httk Documentation

Release 1.2.0.dev1+g3d78bda

Author

Sep 25, 2020

Contents

1 About the High-Throughput Toolkit 3

2 Quickstart 5

3 Reporting bugs 9

4 Citing *httk* in scientific works 11

5 More info and help 13

6 Contributors 15

7 Acknowledgements 17

8 License and redistribution 19

9 Contact 21

10 Full API reference 23

Python Module Index 159

Index 161

This website documents the High-Throughput Toolkit (*httk*). Looking for the Open Materials Database? It is at: http://openmaterialsdb.se

CHAPTER 1

About the High-Throughput Toolkit

The High-Throughput Toolkit (*httk*) is a toolkit for:

- Preparing and running calculations.
- Analyzing the results.
- Store the results and outcome in a global and/or in a personalized database.

httk is an independent implementation of the database-centric high-throughput methodology pioneered by Ceder et al., and others. [see, e.g., Comp. Mat. Sci. 50, 2295 (2011)]. *httk* is presently targeted at atomistic calculations in materials science and electronic structure, but aims to be extended into a library useful also outside those areas.

CHAPTER 2

Quickstart

Httk presently consists of a python library and a few programs. If you just want access to use (rather than develop) the python library, and do not need the external programs, the install is very easy.

(Note: for *httk* version 2.0 we will go over to a single 'script' endpoint, `httk`, for which the pip install step should be sufficient to get a full install.)

2.1 Install to access just the python library

1. You need Python 2.7 and access to pip in your terminal window. (You can get Python and pip, e.g., by installing the Python 2.7 version of Anaconda, https://www.anaconda.com/download, which should give you all you need on Linux, macOS and Windows.)
2. Issue in your terminal window:

   ```
   pip install httk
   ```

   If you at a later point want to upgrade your installation, just issue:

   ```
   pip install httk --upgrade
   ```

You should now be able to simply do `import httk` in your python programs to use the *httk* python library.

2.2 Alternative install: python library + binaries + ability to develop *httk*

1. In addition to Python 2.7 and pip, you also need git. You can get git from here: https://git-scm.com/
2. Issue in your terminal window:

```
git clone https://github.com/rartino/httk
cd httk
pip install --editable . --user
```

If you at a later point want to upgrade your installation, just go back to the *httk* directory and issue:

```
git pull
pip install . --upgrade --user
```

3. To setup the paths to the *httk* programs you also need to run:

```
source /path/to/httk/init.shell
```

where `/path/to/httk` should be the path to where you downloaded *httk* in the steps above. To make this permanent, please add this line to your shell initialization script, e.g., ~/.bashrc

You are now ready to use *httk*.

Notes:

- The above instructions give you access to the latest stable release of httk. To get the latest developer relase (which may or may not work), issue:

```
git checkout devel
pip install . --upgrade --user
```

in your httk directory. To switch back to the stable release, do:

```
git checkout master
pip install . --upgrade --user
```

- An alternative to installing with `pip install` is to just run httk out of the httk directory. In that case, skip the pip install step above and just append `source ~/path/to/httk/init.shell` to your shell init files, with `~/path/to/httk` replaced by the path of your httk directory.)*

2.3 A few simple usage examples

2.3.1 Load a cif file or poscar

This is a very simple example of just loading a structure from a `.cif` file and writing out some information about it.

```
import httk

struct = httk.load("example.cif")

print("Formula:", struct.formula)
print("Volume:", float(struct.uc_volume))
print("Assignments:", struct.uc_formula_symbols)
print("Counts:", struct.uc_counts )
print("Coords:", struct.uc_reduced_coords)
```

Running this generates the output:

```
('Formula:', 'BO2Tl')
('Volume', 509.24213999999984)
('Assignments',['B', 'O', 'Tl'])
('Counts:', [8, 16, 8])
('Coords', FracVector(((1350,4550,4250) , ... , ,10000)))
```

2.3.2 Create structures in code

```
from httk.atomistic import Structure

cell = [[1.0, 0.0, 0.0] ,
        [0.0, 1.0, 0.0] ,
        [0.0, 0.0, 1.0]]
coordgroups = [[
                   [0.5, 0.5, 0.5]
               ],[
                   [0.0, 0.0, 0.0]
               ],[
                   [0.5, 0.0, 0.0], [0.0, 0.5, 0.0], [0.0, 0.0, 0.5]
               ]]

assignments = ['Pb' ,'Ti' ,'O']
volume =62.79
struct = Structure.create(uc_cell = cell,
             uc_reduced_coordgroups = coordgroups,
             assignments = assignments,
             uc_volume = volume)
```

2.3.3 Create database file, store a structure in it, and retrive it

```
import httk, httk.db
from httk.atomistic import Structure

backend = httk.db.backend.Sqlite('example.sqlite')
store = httk.db.store.SqlStore(backend)

tablesalt = httk.load('NaCl.cif')
store.save(tablesalt)

arsenic = httk.load('As.cif')
store.save(arsenic)

# Search for anything with Na
search = store.searcher()
search_struct = search.variable(Structure)
search.add(search_struct.formula_symbols.is_in('Na'))

search.output(search_struct, 'structure')

for match, header in list(search):
    struct = match[0]
    print "Found structure", struct.formula, [str(struct.get_tags()[x]) for x in␣
↪struct.get_tags()]
```

2.3.4 Create database file and store your own data in it

```
#!/usr/bin/env python

import httk, httk.db
from httk.atomistic import Structure

class StructureIsEdible(httk.HttkObject):

    @httk.httk_typed_init({'structure': Structure, 'is_edible': bool})
    def __init__(self, structure, is_edible):
        self.structure = structure
        self.is_edible = is_edible

backend = httk.db.backend.Sqlite('example.sqlite')
store = httk.db.store.SqlStore(backend)

tablesalt = httk.load('NaCl.cif')
edible = StructureIsEdible(tablesalt, True)
store.save(edible)

arsenic = httk.load('As.cif')
edible = StructureIsEdible(arsenic, False)
store.save(edible)
```

2.4 Tutorial

Under `Tutorial/Step1, 2, ...` in your *httk* directory you find a series of code snippets to run to see *httk* in action. You can either just execute them there, or try them out in, e.g., a Jupyter notebook.

In addition to the Tutorial, there is a lot of straightforward examples of various things that can be done with httk in the `Examples` subdirectory. Check the source files for information about what the various examples does.

CHAPTER 3

Reporting bugs

We track our bugs using the issue tracker at github. If you find a bug, please search to see if someone else has reported it here:

https://github.com/rartino/httk/issues

If you cannot find it already reported, please click the 'new issue' button and report the bug.

CHAPTER 4

Citing *httk* in scientific works

This is presently the preferred citation to the httk framework itself:

> The High-Throughput Toolkit (httk), R. Armiento et al., http://httk.openmaterialsdb.se/.

Since *httk* can call upon many other pieces of software quite transparently, it may not be initially obvious what other software should be cited. Unless configured otherwise, *httk* prints out a list of citations when the program ends. You should take note of those citations and include them in your publications if relevant.

CHAPTER 5

More info and help

For more details on installation options refer to *httk Installation Instructions*.

User's guide: see *httk Users' Guide*.

Workflows: for more details on how high-throughput computational workflows are executed via the runmanager.sh program, see *httk Runmanager Details*. This may be useful if you plan to design your own workflows using *httk*.

Developing / contributing to *httk*: see *httk Developers' Guide*

CHAPTER 6

Contributors

For a more complete list of contributors and contributions, see *httk Contributors*.

CHAPTER 7

Acknowledgements

***httk* has kindly been funded in part by:**

- The Swedish Research Council (VR) Grant No. 621-2011-4249
- The Linnaeus Environment at Linköping on Nanoscale Functional Materials (LiLi-NFM) funded by the Swedish Research Council (VR).

CHAPTER 8

License and redistribution

The High-Throughput Toolkit uses the GNU Affero General Public License, which is an open source license that allows redistribution and re-use if the license requirements are met. (Note that this license contains clauses that are not in the GNU Public License, and source code from httk thus cannot be imported into GPL licensed projects.)

The full license text is present in *httk license*.

CHAPTER 9

Contact

Our primary point of contact is email to: httk [at] openmaterialsdb.se (where [at] is replaced by @)

CHAPTER 10

Full API reference

- *Full httk API documentation*
- genindex
- modindex
- search

10.1 httk license

```
                    GNU AFFERO GENERAL PUBLIC LICENSE
                       Version 3, 19 November 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The GNU Affero General Public License is a free, copyleft license for
software and other kinds of works, specifically designed to ensure
cooperation with the community in the case of network server software.

  The licenses for most software and other practical works are designed
to take away your freedom to share and change the works.  By contrast,
our General Public Licenses are intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains free
software for all its users.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
them if you wish), that you receive source code or can get it if you
```

(continues on next page)

(continued from previous page)

```
want it, that you can change the software or use pieces of it in new
free programs, and that you know you can do these things.

  Developers that use our General Public Licenses protect your rights
with two steps: (1) assert copyright on the software, and (2) offer
you this License which gives you legal permission to copy, distribute
and/or modify the software.

  A secondary benefit of defending all users' freedom is that
improvements made in alternate versions of the program, if they
receive widespread use, become available for other developers to
incorporate.  Many developers of free software are heartened and
encouraged by the resulting cooperation.  However, in the case of
software used on network servers, this result may fail to come about.
The GNU General Public License permits making a modified version and
letting the public access it on a server without ever releasing its
source code to the public.

  The GNU Affero General Public License is designed specifically to
ensure that, in such cases, the modified source code becomes available
to the community.  It requires the operator of a network server to
provide the source code of the modified version running there to the
users of that server.  Therefore, public use of a modified version, on
a publicly accessible server, gives the public access to the source
code of the modified version.

  An older license, called the Affero General Public License and
published by Affero, was designed to accomplish similar goals.  This is
a different license, not a version of the Affero GPL, but Affero has
released a new version of the Affero GPL which permits relicensing under
this license.

  The precise terms and conditions for copying, distribution and
modification follow.

                       TERMS AND CONDITIONS

  0. Definitions.

  "This License" refers to version 3 of the GNU Affero General Public License.

  "Copyright" also means copyright-like laws that apply to other kinds of
works, such as semiconductor masks.

  "The Program" refers to any copyrightable work licensed under this
License.  Each licensee is addressed as "you".  "Licensees" and
"recipients" may be individuals or organizations.

  To "modify" a work means to copy from or adapt all or part of the work
in a fashion requiring copyright permission, other than the making of an
exact copy.  The resulting work is called a "modified version" of the
earlier work or a work "based on" the earlier work.

  A "covered work" means either the unmodified Program or a work based
on the Program.

  To "propagate" a work means to do anything with it that, without
```

(continues on next page)

(continued from previous page)

```
permission, would make you directly or secondarily liable for
infringement under applicable copyright law, except executing it on a
computer or modifying a private copy.  Propagation includes copying,
distribution (with or without modification), making available to the
public, and in some countries other activities as well.

  To "convey" a work means any kind of propagation that enables other
parties to make or receive copies.  Mere interaction with a user through
a computer network, with no transfer of a copy, is not conveying.

  An interactive user interface displays "Appropriate Legal Notices"
to the extent that it includes a convenient and prominently visible
feature that (1) displays an appropriate copyright notice, and (2)
tells the user that there is no warranty for the work (except to the
extent that warranties are provided), that licensees may convey the
work under this License, and how to view a copy of this License.  If
the interface presents a list of user commands or options, such as a
menu, a prominent item in the list meets this criterion.

  1. Source Code.

  The "source code" for a work means the preferred form of the work
for making modifications to it.  "Object code" means any non-source
form of a work.

  A "Standard Interface" means an interface that either is an official
standard defined by a recognized standards body, or, in the case of
interfaces specified for a particular programming language, one that
is widely used among developers working in that language.

  The "System Libraries" of an executable work include anything, other
than the work as a whole, that (a) is included in the normal form of
packaging a Major Component, but which is not part of that Major
Component, and (b) serves only to enable use of the work with that
Major Component, or to implement a Standard Interface for which an
implementation is available to the public in source code form.  A
"Major Component", in this context, means a major essential component
(kernel, window system, and so on) of the specific operating system
(if any) on which the executable work runs, or a compiler used to
produce the work, or an object code interpreter used to run it.

  The "Corresponding Source" for a work in object code form means all
the source code needed to generate, install, and (for an executable
work) run the object code and to modify the work, including scripts to
control those activities.  However, it does not include the work's
System Libraries, or general-purpose tools or generally available free
programs which are used unmodified in performing those activities but
which are not part of the work.  For example, Corresponding Source
includes interface definition files associated with source files for
the work, and the source code for shared libraries and dynamically
linked subprograms that the work is specifically designed to require,
such as by intimate data communication or control flow between those
subprograms and other parts of the work.

  The Corresponding Source need not include anything that users
can regenerate automatically from other parts of the Corresponding
Source.
```

(continues on next page)

(continued from previous page)

```
  The Corresponding Source for a work in source code form is that
same work.

  2. Basic Permissions.

  All rights granted under this License are granted for the term of
copyright on the Program, and are irrevocable provided the stated
conditions are met.  This License explicitly affirms your unlimited
permission to run the unmodified Program.  The output from running a
covered work is covered by this License only if the output, given its
content, constitutes a covered work.  This License acknowledges your
rights of fair use or other equivalent, as provided by copyright law.

  You may make, run and propagate covered works that you do not
convey, without conditions so long as your license otherwise remains
in force.  You may convey covered works to others for the sole purpose
of having them make modifications exclusively for you, or provide you
with facilities for running those works, provided that you comply with
the terms of this License in conveying all material for which you do
not control copyright.  Those thus making or running the covered works
for you must do so exclusively on your behalf, under your direction
and control, on terms that prohibit them from making any copies of
your copyrighted material outside their relationship with you.

  Conveying under any other circumstances is permitted solely under
the conditions stated below.  Sublicensing is not allowed; section 10
makes it unnecessary.

  3. Protecting Users' Legal Rights From Anti-Circumvention Law.

  No covered work shall be deemed part of an effective technological
measure under any applicable law fulfilling obligations under article
11 of the WIPO copyright treaty adopted on 20 December 1996, or
similar laws prohibiting or restricting circumvention of such
measures.

  When you convey a covered work, you waive any legal power to forbid
circumvention of technological measures to the extent such circumvention
is effected by exercising rights under this License with respect to
the covered work, and you disclaim any intention to limit operation or
modification of the work as a means of enforcing, against the work's
users, your or third parties' legal rights to forbid circumvention of
technological measures.

  4. Conveying Verbatim Copies.

  You may convey verbatim copies of the Program's source code as you
receive it, in any medium, provided that you conspicuously and
appropriately publish on each copy an appropriate copyright notice;
keep intact all notices stating that this License and any
non-permissive terms added in accord with section 7 apply to the code;
keep intact all notices of the absence of any warranty; and give all
recipients a copy of this License along with the Program.

  You may charge any price or no price for each copy that you convey,
and you may offer support or warranty protection for a fee.
```

(continues on next page)

(continued from previous page)

```
  5. Conveying Modified Source Versions.

  You may convey a work based on the Program, or the modifications to
produce it from the Program, in the form of source code under the
terms of section 4, provided that you also meet all of these conditions:

    a) The work must carry prominent notices stating that you modified
    it, and giving a relevant date.

    b) The work must carry prominent notices stating that it is
    released under this License and any conditions added under section
    7.  This requirement modifies the requirement in section 4 to
    "keep intact all notices".

    c) You must license the entire work, as a whole, under this
    License to anyone who comes into possession of a copy.  This
    License will therefore apply, along with any applicable section 7
    additional terms, to the whole of the work, and all its parts,
    regardless of how they are packaged.  This License gives no
    permission to license the work in any other way, but it does not
    invalidate such permission if you have separately received it.

    d) If the work has interactive user interfaces, each must display
    Appropriate Legal Notices; however, if the Program has interactive
    interfaces that do not display Appropriate Legal Notices, your
    work need not make them do so.

  A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit.  Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

  6. Conveying Non-Source Forms.

  You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the
machine-readable Corresponding Source under the terms of this License,
in one of these ways:

    a) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by the
    Corresponding Source fixed on a durable physical medium
    customarily used for software interchange.

    b) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by a
    written offer, valid for at least three years and valid for as
    long as you offer spare parts or customer support for that product
    model, to give anyone who possesses the object code either (1) a
    copy of the Corresponding Source for all the software in the
    product that is covered by this License, on a durable physical
```

(continues on next page)

(continued from previous page)

```
    medium customarily used for software interchange, for a price no
    more than your reasonable cost of physically performing this
    conveying of source, or (2) access to copy the
    Corresponding Source from a network server at no charge.

    c) Convey individual copies of the object code with a copy of the
    written offer to provide the Corresponding Source.  This
    alternative is allowed only occasionally and noncommercially, and
    only if you received the object code with such an offer, in accord
    with subsection 6b.

    d) Convey the object code by offering access from a designated
    place (gratis or for a charge), and offer equivalent access to the
    Corresponding Source in the same way through the same place at no
    further charge.  You need not require recipients to copy the
    Corresponding Source along with the object code.  If the place to
    copy the object code is a network server, the Corresponding Source
    may be on a different server (operated by you or a third party)
    that supports equivalent copying facilities, provided you maintain
    clear directions next to the object code saying where to find the
    Corresponding Source.  Regardless of what server hosts the
    Corresponding Source, you remain obligated to ensure that it is
    available for as long as needed to satisfy these requirements.

    e) Convey the object code using peer-to-peer transmission, provided
    you inform other peers where the object code and Corresponding
    Source of the work are being offered to the general public at no
    charge under subsection 6d.

  A separable portion of the object code, whose source code is excluded
from the Corresponding Source as a System Library, need not be
included in conveying the object code work.

  A "User Product" is either (1) a "consumer product", which means any
tangible personal property which is normally used for personal, family,
or household purposes, or (2) anything designed or sold for incorporation
into a dwelling.  In determining whether a product is a consumer product,
doubtful cases shall be resolved in favor of coverage.  For a particular
product received by a particular user, "normally used" refers to a
typical or common use of that class of product, regardless of the status
of the particular user or of the way in which the particular user
actually uses, or expects or is expected to use, the product.  A product
is a consumer product regardless of whether the product has substantial
commercial, industrial or non-consumer uses, unless such uses represent
the only significant mode of use of the product.

  "Installation Information" for a User Product means any methods,
procedures, authorization keys, or other information required to install
and execute modified versions of a covered work in that User Product from
a modified version of its Corresponding Source.  The information must
suffice to ensure that the continued functioning of the modified object
code is in no case prevented or interfered with solely because
modification has been made.

  If you convey an object code work under this section in, or with, or
specifically for use in, a User Product, and the conveying occurs as
part of a transaction in which the right of possession and use of the
```

(continues on next page)

(continued from previous page)

```
User Product is transferred to the recipient in perpetuity or for a
fixed term (regardless of how the transaction is characterized), the
Corresponding Source conveyed under this section must be accompanied
by the Installation Information.  But this requirement does not apply
if neither you nor any third party retains the ability to install
modified object code on the User Product (for example, the work has
been installed in ROM).

  The requirement to provide Installation Information does not include a
requirement to continue to provide support service, warranty, or updates
for a work that has been modified or installed by the recipient, or for
the User Product in which it has been modified or installed.  Access to a
network may be denied when the modification itself materially and
adversely affects the operation of the network or violates the rules and
protocols for communication across the network.

  Corresponding Source conveyed, and Installation Information provided,
in accord with this section must be in a format that is publicly
documented (and with an implementation available to the public in
source code form), and must require no special password or key for
unpacking, reading or copying.

  7. Additional Terms.

  "Additional permissions" are terms that supplement the terms of this
License by making exceptions from one or more of its conditions.
Additional permissions that are applicable to the entire Program shall
be treated as though they were included in this License, to the extent
that they are valid under applicable law.  If additional permissions
apply only to part of the Program, that part may be used separately
under those permissions, but the entire Program remains governed by
this License without regard to the additional permissions.

  When you convey a copy of a covered work, you may at your option
remove any additional permissions from that copy, or from any part of
it.  (Additional permissions may be written to require their own
removal in certain cases when you modify the work.)  You may place
additional permissions on material, added by you to a covered work,
for which you have or can give appropriate copyright permission.

  Notwithstanding any other provision of this License, for material you
add to a covered work, you may (if authorized by the copyright holders of
that material) supplement the terms of this License with terms:

    a) Disclaiming warranty or limiting liability differently from the
    terms of sections 15 and 16 of this License; or

    b) Requiring preservation of specified reasonable legal notices or
    author attributions in that material or in the Appropriate Legal
    Notices displayed by works containing it; or

    c) Prohibiting misrepresentation of the origin of that material, or
    requiring that modified versions of such material be marked in
    reasonable ways as different from the original version; or

    d) Limiting the use for publicity purposes of names of licensors or
    authors of the material; or
```

(continues on next page)

(continued from previous page)

```
    e) Declining to grant rights under trademark law for use of some
    trade names, trademarks, or service marks; or

    f) Requiring indemnification of licensors and authors of that
    material by anyone who conveys the material (or modified versions of
    it) with contractual assumptions of liability to the recipient, for
    any liability that these contractual assumptions directly impose on
    those licensors and authors.

  All other non-permissive additional terms are considered "further
restrictions" within the meaning of section 10.  If the Program as you
received it, or any part of it, contains a notice stating that it is
governed by this License along with a term that is a further
restriction, you may remove that term.  If a license document contains
a further restriction but permits relicensing or conveying under this
License, you may add to a covered work material governed by the terms
of that license document, provided that the further restriction does
not survive such relicensing or conveying.

  If you add terms to a covered work in accord with this section, you
must place, in the relevant source files, a statement of the
additional terms that apply to those files, or a notice indicating
where to find the applicable terms.

  Additional terms, permissive or non-permissive, may be stated in the
form of a separately written license, or stated as exceptions;
the above requirements apply either way.

  8. Termination.

  You may not propagate or modify a covered work except as expressly
provided under this License.  Any attempt otherwise to propagate or
modify it is void, and will automatically terminate your rights under
this License (including any patent licenses granted under the third
paragraph of section 11).

  However, if you cease all violation of this License, then your
license from a particular copyright holder is reinstated (a)
provisionally, unless and until the copyright holder explicitly and
finally terminates your license, and (b) permanently, if the copyright
holder fails to notify you of the violation by some reasonable means
prior to 60 days after the cessation.

  Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
violation by some reasonable means, this is the first time you have
received notice of violation of this License (for any work) from that
copyright holder, and you cure the violation prior to 30 days after
your receipt of the notice.

  Termination of your rights under this section does not terminate the
licenses of parties who have received copies or rights from you under
this License.  If your rights have been terminated and not permanently
reinstated, you do not qualify to receive new licenses for the same
material under section 10.
```

(continues on next page)

(continued from previous page)

```
  9. Acceptance Not Required for Having Copies.

  You are not required to accept this License in order to receive or
run a copy of the Program.  Ancillary propagation of a covered work
occurring solely as a consequence of using peer-to-peer transmission
to receive a copy likewise does not require acceptance.  However,
nothing other than this License grants you permission to propagate or
modify any covered work.  These actions infringe copyright if you do
not accept this License.  Therefore, by modifying or propagating a
covered work, you indicate your acceptance of this License to do so.

  10. Automatic Licensing of Downstream Recipients.

  Each time you convey a covered work, the recipient automatically
receives a license from the original licensors, to run, modify and
propagate that work, subject to this License.  You are not responsible
for enforcing compliance by third parties with this License.

  An "entity transaction" is a transaction transferring control of an
organization, or substantially all assets of one, or subdividing an
organization, or merging organizations.  If propagation of a covered
work results from an entity transaction, each party to that
transaction who receives a copy of the work also receives whatever
licenses to the work the party's predecessor in interest had or could
give under the previous paragraph, plus a right to possession of the
Corresponding Source of the work from the predecessor in interest, if
the predecessor has it or can get it with reasonable efforts.

  You may not impose any further restrictions on the exercise of the
rights granted or affirmed under this License.  For example, you may
not impose a license fee, royalty, or other charge for exercise of
rights granted under this License, and you may not initiate litigation
(including a cross-claim or counterclaim in a lawsuit) alleging that
any patent claim is infringed by making, using, selling, offering for
sale, or importing the Program or any portion of it.

  11. Patents.

  A "contributor" is a copyright holder who authorizes use under this
License of the Program or a work on which the Program is based.  The
work thus licensed is called the contributor's "contributor version".

  A contributor's "essential patent claims" are all patent claims
owned or controlled by the contributor, whether already acquired or
hereafter acquired, that would be infringed by some manner, permitted
by this License, of making, using, or selling its contributor version,
but do not include claims that would be infringed only as a
consequence of further modification of the contributor version.  For
purposes of this definition, "control" includes the right to grant
patent sublicenses in a manner consistent with the requirements of
this License.

  Each contributor grants you a non-exclusive, worldwide, royalty-free
patent license under the contributor's essential patent claims, to
make, use, sell, offer for sale, import and otherwise run, modify and
propagate the contents of its contributor version.
```

(continues on next page)

(continued from previous page)

```
  In the following three paragraphs, a "patent license" is any express
agreement or commitment, however denominated, not to enforce a patent
(such as an express permission to practice a patent or covenant not to
sue for patent infringement).  To "grant" such a patent license to a
party means to make such an agreement or commitment not to enforce a
patent against the party.

  If you convey a covered work, knowingly relying on a patent license,
and the Corresponding Source of the work is not available for anyone
to copy, free of charge and under the terms of this License, through a
publicly available network server or other readily accessible means,
then you must either (1) cause the Corresponding Source to be so
available, or (2) arrange to deprive yourself of the benefit of the
patent license for this particular work, or (3) arrange, in a manner
consistent with the requirements of this License, to extend the patent
license to downstream recipients.  "Knowingly relying" means you have
actual knowledge that, but for the patent license, your conveying the
covered work in a country, or your recipient's use of the covered work
in a country, would infringe one or more identifiable patents in that
country that you have reason to believe are valid.

  If, pursuant to or in connection with a single transaction or
arrangement, you convey, or propagate by procuring conveyance of, a
covered work, and grant a patent license to some of the parties
receiving the covered work authorizing them to use, propagate, modify
or convey a specific copy of the covered work, then the patent license
you grant is automatically extended to all recipients of the covered
work and works based on it.

  A patent license is "discriminatory" if it does not include within
the scope of its coverage, prohibits the exercise of, or is
conditioned on the non-exercise of one or more of the rights that are
specifically granted under this License.  You may not convey a covered
work if you are a party to an arrangement with a third party that is
in the business of distributing software, under which you make payment
to the third party based on the extent of your activity of conveying
the work, and under which the third party grants, to any of the
parties who would receive the covered work from you, a discriminatory
patent license (a) in connection with copies of the covered work
conveyed by you (or copies made from those copies), or (b) primarily
for and in connection with specific products or compilations that
contain the covered work, unless you entered into that arrangement,
or that patent license was granted, prior to 28 March 2007.

  Nothing in this License shall be construed as excluding or limiting
any implied license or other defenses to infringement that may
otherwise be available to you under applicable patent law.

  12. No Surrender of Others' Freedom.

  If conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot convey a
covered work so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you may
not convey it at all.  For example, if you agree to terms that obligate you
to collect a royalty for further conveying from those to whom you convey
```

(continues on next page)

(continued from previous page)

```
the Program, the only way you could satisfy both those terms and this
License would be to refrain entirely from conveying the Program.

  13. Remote Network Interaction; Use with the GNU General Public License.

  Notwithstanding any other provision of this License, if you modify the
Program, your modified version must prominently offer all users
interacting with it remotely through a computer network (if your version
supports such interaction) an opportunity to receive the Corresponding
Source of your version by providing access to the Corresponding Source
from a network server at no charge, through some standard or customary
means of facilitating copying of software.  This Corresponding Source
shall include the Corresponding Source for any work covered by version 3
of the GNU General Public License that is incorporated pursuant to the
following paragraph.

  Notwithstanding any other provision of this License, you have
permission to link or combine any covered work with a work licensed
under version 3 of the GNU General Public License into a single
combined work, and to convey the resulting work.  The terms of this
License will continue to apply to the part which is the covered work,
but the work with which it is combined will remain governed by version
3 of the GNU General Public License.

  14. Revised Versions of this License.

  The Free Software Foundation may publish revised and/or new versions of
the GNU Affero General Public License from time to time.  Such new versions
will be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.

  Each version is given a distinguishing version number.  If the
Program specifies that a certain numbered version of the GNU Affero General
Public License "or any later version" applies to it, you have the
option of following the terms and conditions either of that numbered
version or of any later version published by the Free Software
Foundation.  If the Program does not specify a version number of the
GNU Affero General Public License, you may choose any version ever published
by the Free Software Foundation.

  If the Program specifies that a proxy can decide which future
versions of the GNU Affero General Public License can be used, that proxy's
public statement of acceptance of a version permanently authorizes you
to choose that version for the Program.

  Later license versions may give you additional or different
permissions.  However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow a
later version.

  15. Disclaimer of Warranty.

  THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW.  EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
```

(continues on next page)

(continued from previous page)

```
PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM
IS WITH YOU.  SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  16. Limitation of Liability.

  IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS
THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY
GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE
USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF
DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD
PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS),
EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

  17. Interpretation of Sections 15 and 16.

  If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Affero General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.

    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If your software can interact with users remotely through a computer
```

(continues on next page)

(continued from previous page)

```
network, you should also make sure that it provides a way for users to
get its source.  For example, if your program is a web application, its
interface could display a "Source" link that leads users to an archive
of the code.  There are many ways you could offer source, and different
solutions will be better for different programs; see section 13 for the
specific requirements.

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU AGPL, see
<http://www.gnu.org/licenses/>.
```

10.2 *httk* Developers' Guide

10.2.1 Introduction

You likely want to have read the users' guide before reading this.

10.2.2 Short points for experienced developers

- Follow PEP8, except –ignore=E226,E265,E266,E401,E402,E501,W291,W293,W391
- Favor unmutable classes over mutable ones
- For arrays of numbers, use core/FracVector unless you have a reason
- Constructors are generally considered private, use a create(...) static method instead.
- Type conversion should be handled with use(other) methods
- File I/O should be done with the core/ioadapters classes
- Note the plugin system that comes via inheritance from HttkObject

10.2.3 Overview of the python library

- Arrays of numbers: essentially all arrays of numbers within httk.core are implemented using our own vector math class, FracVector. There are many things that can be argued about the pros and cons of re-implementing vector math vs. using numpy vectors. The primary reasons for this design choice was:
  - FracVectors are exact (they are based on fractions), meaning that no information is ever lost about cell shapes and atomic positions, there is no need to handle floating-point 'fussiness' with cutoffs etc. Cell matrices can be exactly inverted, and so on.
  - FracVectors are immutable (but there is a MutableFracVector). They can thus be used as e.g., keys in dictionaries, in sets, etc. This lets us avoid certain type of difficult-to-find bugs where one by mistake mutates a vector that is used elsewhere. (For more info, see the section 'Rant about mutable vs. non-mutable classes' at the end of this document.)
  - FracVectors are implemented in pure Python, making the core part of httk a pure Python library = very easy to install and get up and running
  - FracVectors are easy to convert to floating point arrays when high speed is needed (the opposite conversion is not as easy, requires cutoffs, and will generally not give the exact same results between different computers due to differences in floating point processing.)

- Basic structural classes: we implement our own, rather than using a 'structure' class of another library (e.g., 'Atoms' from ASE). This way we avoid dependencies, but most importantly, our structure classes generally avoid floating point numbers (see discussion about FracVector above). We provide via the 'httk.iface' module conversions to many other structure types in other libraries.

10.2.4 Constructors

The python __init__ constructor is regarded as *private* throughout httk. These constructors should be very light-weight and not sanitize or process their arguments. The arguments to the constructor normally reflect the internal representation of the data and changes when the internal data representation changes as part of future development.

The public constructor should normally be an @classmethod named 'create'. The parameters to create are meant to stay the same even when the internal representation of the data in the class changes. We want 'create' to be as flexible as possible and able to take data on multiple forms. A very common design pattern is that the create method is a "swiss army knife" type creator that can take a multitude of named arguments, and only some set of those arguments are needed to be given. E.g., both these are valid examples of creating a new Structure object:

```
mystruct = Structure.create(cell=mycell, coords=mycoords, counts=mycounts)
mystruct = Structure.create(a=my_a, b=my_b, c=my_c, alpha=my_alpha, beta=my_beta,␣
↪gamma=my_gamma, coords=mycoords, counts=mycounts)
```

Motivation for using create rather than __init__: if __init__ constructors are used as public, one may get into serious limitations in how the internal data representation of the class can be changed later. Also, sometimes it is necessary to create new objects in a way that bypasses any processing of arguments, and this becomes difficult and inelegant if __init__ is already an established public swiss-army-knife type constructor.

10.2.5 The 'use' method

Throughout httk we have another standardized @classmethod method called 'use'. It means "make a best effort to convert the object given into the class on which we call 'use'. E.g.,

```
duck = Duck.use(ducklike)
```

tries to convert ducklike into a Duck, if it is not *already* of type Duck, in which case it is just returned unmodified. The primary difference between 'use' and 'create' is that use always only take one argument (an object we think is 'equivalent' with, e.g., a Duck) and that we generally try to avoid creating a new object if we can.

To better explain the need for this, consider the class 'Structure' and the database class 'DbStructure'. We do not want the 'db' module to leak into the core module (e.g., there should never be any type testing against, e.g., DbStructure or imports from the db submodule into core.) Yet, a Structure and a DbStructure are essentially "the same thing", so methods that expect a 'Structure' with full freedom to use an object as if it is a normal structure is expected to work like this:

```
def do_something(struct):
  struct = Structure.use(struct)
  struct.some_method(...)
```

This saves the need to have to stop and think "wait, is this a function that takes a UnitcellStructure or a Structure?" when using the functions.

One may suggest that it would be better to use object-oriented inheritance for this functionality. However, inheritance typically does not work that great with primitive types (e.g., functions that can take both a string as a file reference, or a Path object, or an IOStream object). Nor does object oriented programming give an unambiguous solution for cross-converting *between subclasses*. Note the following example of the 'use' method:

```
uc_struct = UnitcellStructure()
numpy_stuct = NumpyStructure.use(uc_struct)
# now use numpy_struct in a way that requires NumpyStructure specific methods
```

(Note that there is not yet any NumpyStructure in httk, but will probably be in the future.) In practice NumpyStructure and UnitcellStructure are in different submodules and it makes no sense to make either one inherit from the other, but they (could) both inherit from a common superclass (e.g. 'AbstractStructure'). Nevertheless, even if they do that, there is no obvious way just from object oriented programming to know how to do the above conversion. One could of course 'upcast' UnitcellStructure to AbstractStructure, but the downcast into a NumpyStructure is then not trivial. Also, there could be great benefits in using a conversion 'shortcut' between these two classes that saves time over upcast + a generic downcast.

10.2.6 I/O Adapters

For file io we use httk.core.ioadapters. References to files and output streams can have many types, e.g., strings (i.e., a path), instances of the object Path, instances of Stream, etc. The ioadapters help writing functions that can deal with all these types of references to files comparably easy, without large "if elif elif elif" forks in every such function. Lets say that you write a function that generates some output data:

```
def write_data(fio):
  fio = IoAdapterFileWriter.use(fio)
  f = fio.file
  f.write("OUTPUT")
  fio.close()
```

This allows the input argument 'fio' to be of many, many, different types. You never really need to bother with "converting" your argument before calling write_data. You just *choose* that you want whatever 'fio' was to be turned into an IoAdapterFileWriter, and then you just pick out the 'file' property and use it as a file. You never need to specifically worry about whether fio already was an IoAdapterFileWriter, or just the filename 'output.txt', or a Path object.

10.2.7 Classes and interfaces

A design principle is to keep classes short. As a general rule: only methods that absolutely need to work with the internal data structures of a class should go into the class! Other "methods" should simply be written as regular functions that take one (or more) instances of the class. Put the class in 'classname.py' and the utility methods in 'classnameutils.py'.

The primary benefit of this is that the duck-typing of python allows us to re-use those exact functions even with other objects that fulfill the same API interface as the original class. This cannot be done if they are implemented as instance methods.

However, it is ok to extend the class with convenience methods that are very short calls into functions implemented elsewhere, e.g.,

```
@property
structue.normalized_formula(self):
  return normalized_formula(self)
```

as this helps finding the right method when calling help(object). The difference is that the full implementation is not put into the class iself.

10.2.8 Plugins

To avoid dependences on libraries that you may not have installed, httk implements somewhat unusual 'plugin'-type extensions to any class that inherits from HttkObject.

The practical outcome is that loading a module, e.g., the atomistic visualization module, adds functionality to some objects inside htt.atomistic. E.g.,

```
from httk import *
from httk.atomistic import *
import httk.atomistic.vis
```

This adds, e.g., Structure.vis.show() to show a structure.

In practice this is easy to work with in your own code. We'll use a plugin to the Structure class as example. All you need to do is:

1. create a class that inherits from httk.HttkPlugin, and which implements a method:

   ```
   plugin_init(self, struct)
   ```

   which takes the place of the usual __init__ and gives access to the 'hosting' structure instance.

2. add this to the corresponding HttkObject by:

   ```
   Structure.myplugin = HttkPluginWrapper(MyStructurePluginClass)
   ```

After this has happened during an import, any call on a structure instance, e.g.,

```
struct.myplugin.hello_world()
```

will call the corresponding method in MyStructurePluginClass. Your plugin can also have class methods, which gets called by:

```
Structure.myplugin.classmethod()
```

For a concrete example, look at the structurevisualizerplugin in httk.atomistic.vis.

10.2.9 General recommendations for contributed code

Rule #1: Generally read and follow: http://www.python.org/dev/peps/pep-0008/ You are encouraged to use the pep8 tool (either directly or via your code development platform, but, use: –ignore=E401,E402,E501,W291,W293,W391,E265,E266,E226 (See below for motivations.)

Rule #2: Always organize your code in private sections and a public API. Never write code that depends on private sections outside the class / module / etc.

> It is very very easy for a large Python project to degenerate into a huge pile of code that has such intricate cross-dependences that it is almost impossible to know the implications of a seemingly small change. For example, do you dare changing the internal representation of the data in the X class? You have to be sure no other class reaches into the internal data structures and make assumptions about how they are organized.
>
> The principle of API-oriented organization is simple:
>
> - Every piece of code is either in a private section or part of the public API.
> - Changes to private sections are "easy", as they should never break other code

- Changes to the public API are difficult, and should generally be done only by introducing a new version of the class / module / etc.
- Every public class should be in its own file named after the class, things not meant to be used outside that class should be named with a prefix underscore '_'.

Rule #3: Always make your classes be *immutable* unless you know why you need a mutable class. Do not fall for the pressure of the premature optimization fairy and the idea that "it will be faster if I don't create a new instance". No one cares if you shave 10 ms of the final program execution time, but people will care if your program has bugs. Only optimize code where speed *matters*. See longer rant in section below.

10.2.10 Motivations for/discussions about our digressions from pep8

- E226: missing whitespace around arithmetic operator: This rule as implemented in the pep8 tool is not consistent with the pep0008 standard. Use spaces around arithmetic operators when it adds to readability.
- E265: block comment should start with '# ': We do not want to enforce what can go inside comment sections as they are used rather freely throughout the code right now. This may change in the future.
- E266: too many leading '#' for block comment: see E265
- E401: multiple imports on one line: In this code we put standard system libraries as a single import line to avoid the file preambles to become overly long. All other imports should be each on one line.
- E402: module level import not at top of file: We should generally strive to put all module imports at the top of the file. However, we need to depart from this for conditional imports, especially for our handling of external libraries, and, sometimes for speed optimization (only do slow import X if a function is run that absolutely needs it.)
- E501: line too long: Modern editors allow editing wide source with ease. *Try* to keep lines down under 100 characters, but this rule should be violated if significantly increased readability is obtained by a few even longer lines.
- W291: trailing whitespace: Between all different editors used, this simply generates too many warnings that makes more important pep8 violations more difficult to see. Once in a while we should simply run the files through a tool that removes trailing whitespace.
- W293: blank line contains whitespace: I genuinely disagree with this rule. It is not motivated by the pep0008 standard, but something unmotivated put in by developers of the pep8 tool. Blank lines should be indented to the indentation level of the block that they appear in.
- W391: blank line at end of file: see W291.

10.2.11 A rant about mutable vs. non-mutable classes

While immutable objects incur some overhead due to extra object creation, they generally make programming much easier. For mutable objects you have to learn the internals of the implementation to understand which operations possibly may affect another object.

Consider the following pseudocode for a mutable vector class,:

```
A = MutableVector(((1,2,3,4),(5,6,7,8)))
B = A[0]
B[1] = 7 # does this also change A at the element [0,1]?!
```

You *cannot know the answer*! The answer depends on the internals of MutableVector! However, for an UnMutableVector the answer is trivial ('A' never changes!). Since no one has time to read documentation, the usual programmer will learn when and where a MutableVector affects other vectors by trial-and-error. This leads to bugs!

E.g., let us consider numpy (where vectors are mutable for a good reason: the aim of numpy is to do floating point math at very high speed). Below are some examples of possible assignments operations that can be placed on line 2 in the code above, and a comment that specifies whether the subsequent change of B also changes A. Notice how the behavior is not easy to predict without reading the numpy documentation!:

```
B = A[0]
# Yes, B becomes a reference into A, so changing B also changes A!

B = (A.T)[0].T
# Yes, B is still a reference into A, but with a different shape.
# Changing B also changes A!

B = A.flatten()
# No, flatten() is documented as "returns a copy of the array",
# and indeed, changing B does not change A!

B = A.reshape(8)[0]
# Yes. Despite that this seem to be equivalent to flatten(),
# B becomes a reference into A instead of a copy! Hence, if someone were
# to "clean up the code" by thinking 'flatten is much easier to read'
# and replacing it, they will unintentionally change the behavior of the code!
```

10.2.12 Contributing, License and Redistribution

If you extend the httk framework for yourself, please consider sending your changes back to us. If your changes are generally useful, they will be included in our distribution, which will make your life *much simpler* when you want to upgrade versions.

Presently patches, bug reports, etc., are handled via email, i.e., just email your patches / modified source files to us. (In the future we'll make arrange for a better way, e.g., github.)

The High-Throughput Toolkit uses the GNU Affero General Public License (see the file LICENSE.txt for details), which is an open source license that allows redistribution and re-use if the license requirements are met. (Note that this license contains clauses that are not in the usual GNU Public License, and source code from httk cannot be imported into GPL-only licensed projects.)

If you plan on redistributing / forking httk with major changes, PLEASE edit httk/__init__.py and change the 'version' variable to contain a personal suffix. E.g., set version='1.0.rickard.2'. Then run the command 'make dist'. This creates a httk_v{VERSION}.tgz archive that you can redistribute.

10.2.13 Contact

Our primary point of contact is email to: httk [at] openmaterialsdb.se (where [at] is replaced by @)

10.3 Full httk API documentation

Contents:

10.3.1 httk package

The high-throughput toolkit (httk)

A set of tools and utilities meant to help with:

- Project management, preparation of large-scale computational project.
- **Execution of large-scale computational projects**
  - interface with supercomputer cluster queuing systems, etc.
  - aid with scripting multi-stage runs
  - retrieval of data from supercomputers
- Storage of data in databases
- Search, retrieval and 'processing' of data in storage
- Analysis (especially as a helpful interface against 3:rd party software)

httk.**load**(*ioa*, *ext=None*)
A *very* generic file reader method.

Load a file into a suitable httk object. Try to do the most sane thing possible given the input file. If you know what to expect from the input file, it may be safer to use a targeted method for that file type.

httk.**save**(*obj*, *ioa*, *ext=None*)
A *very* generic file writer method.

Load a file into a suitable httk object. Try to do the most sane thing possible given the input file. If you know what to expect from the input file, it may be safer to use a targeted method for that file type.

httk.**cout**(**args*)

httk.**cerr**(**args*)

class httk.**Code**(*name*, *version*)
Bases: *httk.core.httkobject.HttkObject*

Object for keeping track of httk data about a computer software or script

add_ref(*ref*)

add_refs(*refs*)

add_tag(*tag*, *val*)

add_tags(*tags*)

classmethod create(*name*, *version*, *refs=None*, *tags=None*)
Create a Computation object.

get_refs()

get_tag(*tag*)

get_tags()

class httk.**Computation**(*computation_date*, *description*, *code*, *manifest_hash*, *signatures*, *keys*, *relpath*, *project_counter*, *added_date=None*)
Bases: *httk.core.httkobject.HttkObject*

Object for keeping track of httk data about a specific computation run

add_project(*project*)

add_projects(*projects*)

add_ref(*ref*)

add_refs(*refs*)

add_tag(*tag*, *val*)

add_tags(*tags*)

added_date

classmethod create(*computation_date*, *description*, *code*, *manifest_hash*, *signatures*, *keys*, *project_counter*, *relpath*, *added_date=None*)
Create a Computation object.

get_projects()

get_refs()

get_tag(*tag*)

get_tags()

class httk.**Result**(*computation*)
Bases: *httk.core.httkobject.HttkObject*

Intended as a base class for results tables for computations

classmethod create(*computation*)
Create a Computation object.

class httk.**ComputationRelated**(*main_computation*, *other_computation*, *relation*)
Bases: *httk.core.httkobject.HttkObject*

Object for keeping track of httk data about a specific computation run

classmethod create(*main_computation*, *other_computation*, *relation*)
Create a Computation object.

class httk.**ComputationProject**(*computation*, *project*)
Bases: *httk.core.httkobject.HttkObject*

classmethod create(*computation*, *project*)
Create a Computation object.

class httk.**Author**(*last_name*, *given_names*)
Bases: *httk.core.httkobject.HttkObject*

Object for keeping track of tags for other objects

classmethod create(*last_name*, *given_names*)
Create a Author object.

class httk.**Reference**(*ref*, *authors=None*, *editors=None*, *journal=None*, *journal_issue=None*, *journal_volume=None*, *page_first=None*, *page_last=None*, *title=None*, *year=None*, *book_publisher=None*, *book_publisher_city=None*, *book_title=None*)
Bases: *httk.core.httkobject.HttkObject*

A reference citation

classmethod create(*ref=None*, *authors=None*, *editors=None*, *journal=None*, *journal_issue=None*, *journal_volume=None*, *page_first=None*, *page_last=None*, *title=None*, *year=None*, *book_publisher=None*, *book_publisher_city=None*, *book_title=None*)
Create a Reference object.

class httk.**Project**(*name*, *description*, *project_key*, *keys*)
Bases: *httk.core.httkobject.HttkObject*

add_ref(*ref*)

add_refs(*refs*)

add_tag(*tag*, *val*)

add_tags(*tags*)

classmethod create(*name*, *description*, *project_key*, *keys*)
Create a Project object.

get_refs()

get_tag(*tag*)

get_tags()

class httk.**ProjectRef**(*project*, *reference*)
Bases: *httk.core.httkobject.HttkObject*

class httk.**ProjectTag**(*project*, *tag*, *value*)
Bases: *httk.core.httkobject.HttkObject*

class httk.**FracVector**(*noms*, *denom=1*)
Bases: *httk.core.vectors.vector.Vector*

FracVector is a general *immutable* N-dimensional vector (tensor) class for performing linear algebra with fractional numbers.

A FracVector consists of a multidimensional tuple of integer nominators, and a single shared integer denominator.

Since FracVectors are immutable, every operation on a FracVector returns a new FracVector with the result of the operation. A created FracVector never changes. Hence, they are safe to use as keys in dictionaries, to use in sets, etc.

Note: most methods returns FracVector results that are not simplified (i.e., the FracVector returned does *not* have the smallest possible integer denominator). To return a FracVector with the smallest possible denominator, just call FracVector.simplify() at the last step.

T()
Returns the transpose, A^T.

acos(*prec=None*, *degrees=False*, *limit=False*)
Return a FracVector where every element is the arccos of the element in the source FracVector.

prec = precision (should be set as a fraction) limit = True requires the denominator to be smaller or equal to precision

argmax()
Return the index of the maximum element across all dimensions in the FracVector.

argmin()
Return the index of the minimum element across all dimensions in the FracVector.

asin(*prec=None*, *degrees=False*, *limit=False*)
Return a FracVector where every element is the arcsin of the element in the source FracVector.

prec = precision (should be set as a fraction) limit = True requires the denominator to be smaller or equal to precision

ceil()
Returns the integer that is equal to or just below the value stored in a scalar FracVector.

classmethod chain_vecs(*vecs*)
Optimized chaining of FracVectors.

vecs: a list (or tuple) of fracvectors.

Returns the same thing as FracVector.create(vecs,chain=True)

i.e., removes outermost dimension and chain the sub-sequences. If input=[[1 2 3],[4,5,6]], then FracVector.chain(input) -> [1,2,3,4,5,6]

but this method assumes all vectors share the same denominator (it raises an exception if this is not true)

cos(*prec=None*, *degrees=False*, *limit=False*)

Return a FracVector where every element is the cosine of the element in the source FracVector.

prec = precision (should be set as a fraction) limit = True requires the denominator to be smaller or equal to precision

classmethod create(*noms*, *denom=None*, *simplify=True*, *chain=False*, *min_accuracy=Fraction(1, 10000)*)

Create a FracVector from various types of sequences.

Simplest use:

```
FracVector.create(some_kind_of_sequence)
```

where 'some_kind_of_sequence' can be any nested list or tuple of objects that can be used in the constructor of the Python Fraction class (also works with strings!). If any object found while traveling the items has a .to_fractions() method, it will be called and is expected to return a fraction or list or tuple of fractions.

Optional parameters:

- Invocation with denominator: FracVector.create(nominators,denominator) nominators is any sequence, and denominator a common denominator to divide all nominators with
- simplify: boolean, return a FracVector with the smallest possible denominator.
- chain: boolean, remove outermost dimension and chain the sub-sequences. I.e., if input=[[1 2 3],[4,5,6]], then FracVector.create(input) -> [1,2,3,4,5,6]

Relevant: FracVector itself implements .to_fractions(), and hence, the same constructor allows stacking several FracVector objects like this:

```
vertical_fracvector = FracVector.create([[fracvector1],[fracvector2]])
horizontal_fracvector = FracVector.create([fracvector1,fracvector2],
↪chain=True)
```

- min_accuracy: set to a boolean to adjust the minimum accuracy assumed in string input. The default is 1/10000, i.e. 0.33 = 0.3300 = 33/100, whereas 0.3333 = 1/3. Set it to None to assume infinite accuracy, i.e., convert exactly whatever string is given (unless a standard deviation is given as a parenthesis after the string.)

classmethod create_cos(*data*, *degrees=False*, *limit=False*, *find_best_rational=True*, *prec=Fraction(1, 1000000)*)

Creating a FracVector as the cosine of the argument data. If data are composed by strings, the standard deviation of the numbers are taken into account, and the best possible fractional approximation to the cosines of the data are returned within the standard deviation.

This is not the same as FracVector.create(data).cos(), which creates the best possible fractional approximations of data and then takes cos on that.

classmethod create_exp(*data*, *prec=Fraction(1, 1000000)*, *limit=False*)

Creating a FracVector as the exponent of the argument data. If data are composed by strings, the standard deviation of the numbers are taken into account, and the best possible fractional approximation to the cosines of the data are returned within the standard deviation.

This is not the same as FracVector.create(data).exp(), which creates the best possible fractional approximations of data and then takes exp on that.

classmethod create_sin(*data*, *degrees=False*, *limit=False*, *prec=Fraction(1, 1000000)*)
Creating a FracVector as the sine of the argument data. If data are composed by strings, the standard deviation of the numbers are taken into account, and the best possible fractional approximation to the cosines of the data are returned within the standard deviation.

This is not the same as FracVector.create(data).sin(), which creates the best possible fractional approximations of data and then takes cos on that.

cross(*other*)
Returns the vector cross product of the 3-element 1D vector with the 3-element 1D vector 'other', i.e., A x B.

det()
Returns the determinant of the FracVector as a scalar FracVector.

dim
This property returns a tuple with the dimensionality of each dimension of the FracVector (the noms are assumed to be a nested list of rectangular shape).

dot(*other*)
Returns the vector dot product of the 1D vector with the 1D vector 'other', i.e., A . B or A cdot B. The same as A * B.T().

exp(*prec=None*, *limit=False*)
Return a FracVector where every element is the exponent of the element in the source FracVector.

prec = precision (should be set as a fraction) limit = True requires the denominator to be smaller or equal to precision

classmethod eye(*dims*)
Create a diagonal one-matrix with the given dimensions

flatten()
Returns a FracVector that has been flattened out to a single rowvector

floor()
Returns the integer that is equal to or just below the value stored in a scalar FracVector.

classmethod from_floats(*l*, *resolution=4294967296*)
Create a FracVector from a (nested) list or tuple of floats. You can convert a numpy array with this method if you use A.tolist()

resolution: the resolution used for interpreting the given floating point numbers. Default is 2^32.

classmethod from_tuple(*t*)
Return a FracVector created from the tuple representation: (denom, ... noms...), returned by the to_tuple() method.

ged_prestacked(*other*)

ged_stackedinsert(*pos*, *other*)

get_append(*other*)

get_extend(*other*)

get_insert(*pos*, *other*)

get_prepend(*other*)

get_prextend(*other*)

get_stacked(*other*)

inv()
Returns the matrix inverse, A^-1

lengthsqr()
Returns the square of the length of the vector. The same as A * A.T()

limit_denominator(*max_denom=1000000000*)
Returns a FracVector of reduced resolution.

resolution: each element in the returned FracVector is the closest numerical approximation that can is allowed by a fraction with maximally this denominator. Note: since all elements must be put on a common denominator, the result may have a larger denominator than max_denom

max()
Return the maximum element across all dimensions in the FracVector. max(fracvector) works for a 1D vector.

metric_product(*vecA*, *vecB*)

Returns the result of the metric product using the present square FracVector as the metric matrix. The same as
vecA*self*vecB.T().

min()
Return the minimum element across all dimensions in the FracVector. max(fracvector) works for a 1D vector.

mul(*other*)
Returns the result of multiplying the vector with 'other' using matrix multiplication.

Note that for two 1D FracVectors, A.dot(B) is *not* the same as A.mul(B), but rather: A.mul(B.T()).

nargmax()
Return a list of indices of all maximum elements across all dimensions in the FracVector.

nargmin()
Return a list of indices for all minimum elements across all dimensions in the FracVector.

static nested_map(*op*, **ls*)
Map an operator over a nested tuple. (i.e., the same as the built-in map(), but works recursively on a nested tuple)

static nested_map_fractions(*op*, **ls*)
Map an operator over a nested tuple, but checks every element for a method to_fractions() and uses this to further convert objects into tuples of Fraction.

nom
Returns the integer nominator of a scalar FracVector.

normalize()
Add/remove an integer +/-N to each element to place it in the range [0,1)

normalize_half()
Add/remove an integer +/-N to each element to place it in the range [-1/2,1/2)

This is useful to find the shortest vector C between two points A, B in a space with periodic boundary conditions [0,1
C = (A-B).normalize_half()

classmethod pi(*prec=Fraction(1, 1000000)*, *limit=False*)
Create a scalar FracVector with a rational approximation of pi to precision prec.

classmethod random(*dims*, *minnom=-100*, *maxnom=100*, *denom=100*)
Create a zero matrix with the given dimensions

reciprocal()

classmethod set_common_denom(*A*, *B*)
Used internally to combine two different FracVectors.

Returns a tuple (A2,B2,denom) where A2 is numerically equal to A, and B2 is numerically equal to B, but A2 and B2 are both set on the same shared denominator 'denom' which is the *product* of the denominator of A and B.

set_denominator(*set_denom=1000000000*)
Returns a FracVector of reduced resolution where every element is the closest numerical approximation using this denominator.

sign()
Returns the sign of the scalar FracVector: -1, 0 or 1.

simplify()
Returns a reduced FracVector. I.e., each element has the same numerical value but the new FracVector represents them using the smallest possible shared denominator.

sin(*prec=None*, *degrees=False*, *limit=False*)
Return a FracVector where every element is the sine of the element in the source FracVector.

prec = precision (should be set as a fraction) limit = True requires the denominator to be smaller or equal to precision

sqrt(*prec=None*, *limit=False*)
Return a FracVector where every element is the sqrt of the element in the source FracVector.

prec = precision (should be set as a fraction) limit = True requires the denominator to be smaller or equal to precision

classmethod stack_vecs(*vecs*)
Optimized stacking of FracVectors.

vecs = a list (or tuple) of fracvectors.

Returns the same thing as:

```
FracVector.create(vecs)
```

but only works if all vectors share the same denominator (raises an exception if this is not true)

to_float()
Converts a scalar ExactVector to a single float.

to_floats()
Converts the ExactVector to a list of floats.

to_fraction()
Converts scalar FracVector to a fraction.

to_fractions()
Converts the FracVector to a list of fractions.

to_int()
Converts scalar FracVector to an integer (truncating as necessary).

to_ints()
Converts the FracVector to a list of integers, rounded off as best possible.

to_string(*accuracy=8*)
Converts the ExactVector to a list of strings.

to_strings(*accuracy=8*)
Converts the ExactVector to a list of strings.

to_tuple()
Return a FracVector on tuple representation: (denom, ... noms...).

classmethod use(*old*)
Make sure variable is a FracVector, and if not, convert it.

validate()

classmethod zeros(*dims*)
Create a zero matrix with the given dimensions

class httk.**FracScalar**(*nom*, *denom*)
Bases: *httk.core.vectors.fracvector.FracVector*

Represents the fractional number nom/denom. This is a subclass of FracVector with the purpose of making it clear when a scalar fracvector is needed/used.

classmethod create(*nom*, *denom=None*, *simplify=True*)
Create a FracScalar.

FracScalar(something) something may be any object that can be used in the constructor of the Python Fraction class (also works with strings!).

class httk.**MutableFracVector**(*noms*, *denom*)
Bases: *httk.core.vectors.fracvector.FracVector*, *httk.core.vectors.vector.MutableVector*

Same as FracVector, only, this version allow assignment of elements, e.g.,

```
mfracvec[2,7] = 5
```

and, e.g.,

```
mfracvec[:,7] = [1,2,3,4]
```

Other than this, the FracVector methods exist and do the same, i.e., they return *copies* of the fracvector, rather than modifying it.

However, methods have also been added named with set_* prefixes which performs mutating operations, e.g.,

```
A.set_T()
```

replaces A with its own transpose, whereas

```
A.T()
```

just returns a new MutableFracVector that is the transpose of A, leaving A unmodified.

classmethod from_FracVector(*other*)
Create a MutableFracVector from a FracVector.

invalidate()
Internal method to call when MutableFracVector is changed in such a way that cached properties are invalidated (e.g., _dim)

static nested_inmap(*op*, **ls*)
Like inmap, but work for nested lists

static nested_map(*op*, **ls*)
Map an operator over a nested list. (i.e., the same as the built-in map(), but works recursively on a nested list)

static nested_map_fractions(*op*, **ls*)
Map an operator over a nested list, but checks every element for a method to_fractions() and uses this to further convert objects into lists of Fraction.

set_T()
Changes MutableFracVector inline into own transpose: self -> self.T

set_inv()
Changes MutableFracVector inline into own inverse: self -> self^-1

set_negative()
Changes MutableFracVector inline into own negative: self -> -self

set_normalize()
Add/remove an integer +/-N to each element to place it in the range [0,1)

set_normalize_half()
Add/remove an integer +/-N to each element to place it in the range [-1/2,1/2)

This is useful to find the shortest vector C between two points A, B in a space with periodic boundary conditions [0,1
C = (A-B).normalize_half()

set_set_denominator(*resolution=1000000000*)
Changes MutableFracVector; reduces resolution.

resolution is the new denominator, each element becomes the closest numerical approximation using this denominator.

set_simplify()
Changes MutableFracVector; reduces any common factor between denominator and all nominators

to_FracVector()
Return a FracVector with the values of this MutableFracVector.

classmethod use(*old*)
Make sure variable is a MutableFracVector, and if not, convert it.

validate()

class httk.**IoAdapterFileReader**(*f*, *name=None*, *deletefilename=None*, *close=False*)
Bases: object

Io adapter for easy handling of io.

close()

classmethod use(*other*)

class httk.**IoAdapterFileWriter**(*f*, *name=None*, *close=False*)
Bases: object

Io adapter for access to data as a python file object

close()

classmethod use(*other*)

class httk.**IoAdapterFileAppender**(*f*, *name=None*)
Bases: object

Io adapter for access to data as a python file object

close()

classmethod use(*other*)

class httk.**IoAdapterString**(*string=None*, *name=None*)
Bases: object

Universal io adapter, helps handling the passing of filenames, files, and strings to functions that deal with io

close()

string

classmethod use(*other*)

class httk.**IoAdapterStringList**(*stringlist*, *name=None*)
Bases: object

Universal io adapter, helps handling the passing of filenames, files, and strings to functions that deal with io

classmethod use(*other*)

class httk.**IoAdapterStringList**(*stringlist*, *name=None*)
Bases: object

Universal io adapter, helps handling the passing of filenames, files, and strings to functions that deal with io

classmethod use(*other*)

class httk.**HttkObject**
Bases: object

get_codependent_data()

hexhash

classmethod new_from(*other*)

to(*newtype*)

to_tuple(*use_hexhash=False*)

classmethod types()

classmethod use(*old*)

httk.**httk_typed_property**(*t*)

httk.**httk_typed_init**(*t*, ***kargs*)

httk.**httk_typed_property_delayed**(*t*)

httk.**httk_typed_init_delayed**(*t*, ***kargs*)

class httk.**HttkPluginWrapper**(*plugin=None*)
Bases: object

class httk.**HttkPlugin**
Bases: object

class httk.**HttkPluginPlaceholder**(*plugininfo=None*)
Bases: object

class httk.**Signature**(*signature_data*, *key*)
Bases: *httk.core.httkobject.HttkObject*

classmethod create(*signature_data*, *key*)
Create a Computation object.

class httk.**SignatureKey**(*keydata*, *description*)
Bases: *httk.core.httkobject.HttkObject*

classmethod create(*keydata*, *description*)
Create a Computation object.

Subpackages

httk.analysis package

Subpackages

httk.analysis.matsci package

Subpackages

httk.analysis.matsci.vis package

Submodules

httk.analysis.matsci.vis.matplotlibphasediagramvisualizer module

httk.analysis.matsci.vis.phasediagramvisualizerplugin module

class httk.analysis.matsci.vis.phasediagramvisualizerplugin.**PhaseDiagramVisualizerPlugin**
Bases: *httk.core.httkobject.HttkPlugin*

params()

plugin_init(*phasediagram*)

show(*params={}, backends=['matplotlib'], debug=False*)

wait()

Submodules

httk.analysis.matsci.phasediagram module

class httk.analysis.matsci.phasediagram.**PhaseDiagram**
Bases: object

add_phase(*symbols*, *counts*, *id*, *energy*)
Handles energy=None, for a phase we don't know the energy of.

competing_indices

coord_system

coords()

classmethod create()

hull_competing_indices

hull_competing_phase_lines()

hull_distances

hull_indices

hull_point_coords()

hull_points()

hull_to_interior_competing_phase_lines()

interior_competing_phase_lines()

interior_point_coords()

line_coords()

other_point_coords()

phase_lines

set_hull_data(*hull_indices*, *competing_indices*, *hull_competing_indices*, *hull_distances*, *coord_system*, *phase_lines*)

vis

httk.atomistic package

The httk.atomistic package

Classes and utilities for dealing with high-throughput calculations of atomistic systems.

class httk.atomistic.**Structure**(*assignments*, *rc_sites=None*, *rc_cell=None*, *other_reps=None*)
Bases: *httk.core.httkobject.HttkObject*

A Structure represents N sites of, e.g., atoms or ions, in any periodic or non-periodic arrangement. The structure object is meant to be immutable and assumes that no internal variables are changed after its creation. All methods that 'changes' the object creates and returns a new, updated, structure object.

This is the general heavy weight structure object. For lightweight structure objects, use UnitcellStructure or RepresentativeStructure.

Naming conventions in httk.atomistic:

Structure cell type abbreviations:

rc = Representative cell: only representative atoms are given inside the conventional cell. they need to be replicated by the symmetry elements.

uc = Unit cell: any (imprecisely defined) unit cell (usually the unit cell used to define the structure if it was not done via a representative cell.) with all atoms inside.

pc = Primitive unit cell: a smallest possible unit cell (the standard one) with all atoms inside.

cc = Conventional unit cell: the high symmetry unit cell (rc) with all atoms inside.

For cells:

cell = an abstract name for any reasonable representation of a 'cell' that defines the basis vectors used for representing the structure. When a 'cell' is returned, it is an object of type Cell

basis = a 3x3 sequence-type with (in rows) the three basis vectors (for a periodic system, defining the unit cell, and defines the unit of repetition for the periodic dimensions)

lengths_and_angles = (a,b,c,alpha,beta,gamma): the basis vector lengths and angles

niggli_matrix = ((v1*v1, v2*v2, v3*v3),(2*v2*v3, 2*v1*v3, 2*v2*v3)) where v1, v2, v3 are the vectors forming the basis

metric = ((v1*v1,v1*v2,v1*v3),(v2*v1,v2*v2,v2*v3),(v3*v1,v3*v2,v3*v3))

For sites:

These following prefixes are used to describe types of site specifications: representative cell/rc = only representative atoms are given, which are then to be repeated by structure symmetry group to give all sites

unit cell/uc = all atoms in unitcell

reduced = coordinates given in cell vectors

cartesian = coordinates given as direct cartesian coordinates

sites = used as an abstract name for any sensible representation of a list of coordinates and a cell, when a 'sites' is returned, it is an object of type Sites

counts = number of atoms of each type (one per entry in assignments)

coordgroups = coordinates represented as a 3-level-list of coordinates, e.g. [[[0,0,0],[0.5,0.5,0.5]],[[0.25,0.25,0.25]]] where level-1 list = groups: one group for each equivalent atom

counts and coords = one list with the number of atoms of each type (one per entry in assignments) and a 2-level list of coordinates.

For assignments of atoms, etc. to sites: assignments = abstract name for any representation of assignment of atoms. When returned, will be object of type Assignment.

atomic_numbers = a sequence of integers for the atomic number of each species

occupations = a sequence where the assignments are *repeated* for each coordinate as needed (prefixed with uc or rc depending on which coordinates)

For cell scaling: scaling = abstract name for any representation of cell scaling

scale = multiply all basis vectors with this number

volume = rescaling the cell such that it takes this volume

For periodicity: periodicity = abstract name of a representation of periodicity

pbc = 'periodic boundary conditions' = sequence of True and False for which basis vectors are periodic / non-periodic

nonperiodic_vecs = integer, number of basis vectors, counted from the first, which are non-periodic

For spacegroup: spacegroup = abstract name for any spacegroup representation. When returned, is of type Spacegroup.

hall_symbol = specifically the hall_symbol string representation of the spacegroup

add_ref(*ref*)

add_refs(*refs*)

add_tag(*tag*, *val*)

add_tags(*tags*)

anonymous_formula

anonymous_wyckoff_sequence

cc

cc_formula_parts

clean()

classmethod create(*structure=None*, *assignments=None*, *rc_cell=None*, *rc_basis=None*, *rc_lengths=None*, *rc_angles=None*, *rc_cosangles=None*, *rc_niggli_matrix=None*, *rc_metric=None*, *rc_a=None*, *rc_b=None*, *rc_c=None*, *rc_alpha=None*, *rc_beta=None*, *rc_gamma=None*, *rc_sites=None*, *rc_reduced_coordgroups=None*, *rc_cartesian_coordgroups=None*, *rc_reduced_coords=None*, *rc_cartesian_coords=None*, *rc_reduced_occupationscoords=None*, *rc_cartesian_occupationscoords=None*, *rc_occupancies=None*, *rc_counts=None*, *wyckoff_symbols=None*, *multiplicities=None*, *spacegroup=None*, *hall_symbol=None*, *spacegroupnumber=None*, *setting=None*, *rc_scale=None*, *rc_scaling=None*, *rc_volume=None*, *uc_cell=None*, *uc_basis=None*, *uc_lengths=None*, *uc_angles=None*, *uc_cosangles=None*, *uc_niggli_matrix=None*, *uc_metric=None*, *uc_a=None*, *uc_b=None*, *uc_c=None*, *uc_alpha=None*, *uc_beta=None*, *uc_gamma=None*, *uc_sites=None*, *uc_reduced_coordgroups=None*, *uc_cartesian_coordgroups=None*, *uc_reduced_coords=None*, *uc_cartesian_coords=None*, *uc_reduced_occupationscoords=None*, *uc_cartesian_occupationscoords=None*, *uc_occupancies=None*, *uc_counts=None*, *uc_scale=None*, *uc_scaling=None*, *uc_volume=None*, *uc_is_primitive_cell=False*, *uc_is_conventional_cell=False*, *volume_per_atom=None*, *periodicity=None*, *nonperiodic_vecs=None*, *refs=None*, *tags=None*)

A Structure represents N sites of, e.g., atoms or ions, in any periodic or non-periodic arrangement.

This is a swiss-army-type constructor that allows a selection between a large number of optional arguments.

Note: if redundant and non-compatible information is given, the behavior is undefined. E.g., don't try to call this with a structure + a volume in hopes to get a copy with rescaled volume.

To create a new structure, three primary components are:

- **cell: defines the basis vectors in which reduced coordinates are expressed, and the** unit of repetition (*if* the structure has any periodicity - see the 'periodicity' parameter)
- assignments: a list of 'things' (atoms, ions, etc.) that goes on the sites in the structure
- sites: a sensible representation of location / coordinates of the sites.

Note: *rc_*-prefixes are consistently enforced for any quantity that would be different in a UnitcellStructure. This is to allow for painless change between the various structure-type objects without worrying about accidently using the wrong type of sites object.

Input parameters:

- ONE OF: 'cell'; 'basis', 'length_and_angles'; 'niggli_matrix'; 'metric'; all of: a,b,c, alpha, beta, gamma. (cell requires a Cell object or a very specific format, so unless you know what you are doing, use one of the others.)
- ONE OF: 'assignments', 'atomic_numbers', 'occupancies' (assignments requires an Assignments object or a sequence.), occupations repeats similar site assignments as needed
- ONE OF: 'rc_sites', 'rc_coords' (IF rc_occupations OR rc_counts are also given), 'uc_coords' (IF uc_occupations OR uc_counts are also given) 'rc_B_C', where B=reduced or cartesian, C=coordgroups, coords, or occupationscoords

  Notes:

  - occupationscoords may differ from coords by *order*, since giving occupations as, e.g., ['H','O','H'] does not necessarily have the same order of the coordinates as the format of counts+coords as (2,1), ['H','O'].
  - rc_sites and uc_sites requires a Sites object or a very specific format, so unless you know what you are doing, use one of the others.)
- **ONE OF: scale or volume:** scale = multiply the basis vectors with this scaling factor, volume = the representative (conventional) cell volume (overrides 'scale' if both are given) volume_per_atom = cell volume / number of atoms
- ONE OF periodicity or nonperiodic_vecs

See help(Structure) for more information on the data format of all these data representations.

element_wyckoff_sequence

extended

extensions

find_symmetry()

formula

formula_counts

formula_spaceseparated

formula_symbols

get_refs()

get_tag(*tag*)

get_tags()

hall_symbol

has_rc_repr
Returns True if the structure already contains the representative coordinates + spacegroup, and thus can be queried for this data without launching an expensive symmetry finder operation.

has_uc_repr
Returns True if the structure contains any unit cell-type coordinate representation, and thus can be queried for this data without launching a somewhat expensive cell filling operation.

io

number_of_elements

pbc

pc

pc_a

pc_alpha

pc_b

pc_beta

pc_c

pc_counts

pc_formula_parts

pc_gamma

pc_nbr_atoms

pc_volume

rc

rc_a

rc_alpha

rc_b

rc_basis

rc_beta

rc_c

rc_cartesian_coordgroups

rc_cartesian_coords

rc_cartesian_occupationscoords

rc_cell_orientation

rc_counts

rc_gamma

rc_lengths_and_angles

rc_nbr_atoms

rc_occupancies

rc_occupationssymbols

rc_reduced_coordgroups

rc_reduced_coords

rc_volume

spacegroup

spacegroup_number

spacegroup_number_and_setting

supercell

symbols

tidy()

transform(*matrix*, *max_search_cells=20*, *max_atoms=1000*)

uc

uc_a

uc_alpha

uc_b

uc_basis

uc_beta

uc_c

uc_cartesian_coordgroups

uc_cartesian_coords

uc_cartesian_occupationscoords

uc_cell

uc_cell_orientation

uc_counts

uc_formula

uc_formula_counts

uc_formula_parts

uc_formula_symbols

uc_gamma

uc_lengths_and_angles

uc_nbr_atoms

uc_occupancies

uc_occupationssymbols

uc_reduced_coordgroups

uc_reduced_coords

uc_reduced_occupationscoords

uc_sites

uc_volume

classmethod use(*other*)

volume_per_atom

wyckoff_sequence

class httk.atomistic.**Cell**(*basis*, *lattice_system*, *orientation=1*)

Bases: *httk.core.httkobject.HttkObject*

Represents a cell (e.g., a unitcell, but also possibly just the basis vectors of a non-periodic system)

(The ability to represent the cell for a non-periodic system is also the reason this class is not called Lattice.)

clean()

coordgroups_cartesian_to_reduced(*coordgroups*)

coordgroups_reduced_to_cartesian(*coordgroups*)

coords_cartesian_to_reduced(*coords*)

coords_reduced_to_cartesian(*coords*)

classmethod create(*cell=None*, *basis=None*, *metric=None*, *niggli_matrix=None*, *a=None*, *b=None*, *c=None*, *alpha=None*, *beta=None*, *gamma=None*, *lengths=None*, *angles=None*, *cosangles=None*, *scale=None*, *scaling=None*, *volume=None*, *periodicity=None*, *nonperiodic_vecs=None*, *orientation=1*, *hall=None*, *lattice_system=None*, *eps=0*)

Create a new cell object,

cell: any one of the following:

- a 3x3 array with (in rows) the three basis vectors of the cell (a non-periodic system should conventionally use an identity matrix)
- a dict with a single key 'niggli_matrix' with a 3x2 array with the Niggli Matrix representation of the cell
- a dict with 6 keys, 'a', 'b', 'c', 'alpha', 'beta', 'gamma' giving the cell parameters as floats

scaling: free form input parsed for a scale. positive value = multiply basis vectors by this value negative value = rescale basis vectors so that cell volume becomes abs(value).

scale: set to non-None to multiply all cell vectors with this factor

volume: set to non-None if the basis vectors only give directions, and the volume of the cell should be this value (overrides scale)

periodicity: free form input parsed for periodicity sequence: True/False for each basis vector being periodic integer: number of non-periodic basis vectors

hall: giving the hall symbol makes it possible to determine the lattice system without numerical inaccuracy

lattice_system: any one of: 'cubic', 'hexagonal', 'tetragonal', 'orthorhombic', 'trigonal', 'triclinic', 'monoclinic', 'unknown'

get_axes_standard_order_transform()

get_normalized()

get_normalized_longestvec()

is_point_inside(*cartesian_coord*)

normalization_longestvec_scale

Get the factor with which a normalized version of this cell needs to be multiplied to reproduce this cell.

I.e. self = (normalization_scale)*self.get_normalized()

normalization_scale

scaling()

classmethod use(*other*)

volume

class httk.atomistic.**RepresentativeSites**(*reduced_coordgroups=None*, *cartesian_coordgroups=None*, *reduced_coords=None*, *cartesian_coords=None*, *counts=None*, *hall_symbol=None*, *pbc=None*, *wyckoff_symbols=None*, *multiplicities=None*)

Bases: *httk.atomistic.sites.Sites*

Represents any collection of sites in a unitcell

anonymous_wyckoff_sequence

clean()

classmethod create(*sites=None*, *reduced_coordgroups=None*, *reduced_coords=None*, *counts=None*, *spacegroup=None*, *hall_symbol=None*, *spacegroupnumber=None*, *setting=None*, *periodicity=None*, *wyckoff_symbols=None*, *multiplicities=None*, *occupancies=None*, *pbc=None*)

Create a new sites object

crystal_system

get_uc_sites()

lattice_symbol

lattice_system

tidy()

total_number_of_atoms

wyckoff_sequence

class httk.atomistic.**UnitcellSites**(*reduced_coordgroups=None*, *reduced_coords=None*, *counts=None*, *hall_symbol='P 1'*, *pbc=None*)

Bases: *httk.atomistic.sites.Sites*

Represents any collection of sites in a unitcell

total_number_of_atoms

class httk.atomistic.**Assignments**(*siteassignments*, *extensions=[]*)

Bases: *httk.core.httkobject.HttkObject*

Represents a possible vector of assignments

atomic_numbers

classmethod create(*assignments=None*)

Create a new assignment object,

assignments: a list-style object with one entry per 'atom type'. Any sensible type accepted, most notably, integers (for atom number)

extended

ratios

ratioslist

symbollists

symbols

to_basis()

classmethod use(*old*)

class httk.atomistic.**Compound**(*element_wyckoff_sequence*, *formula*, *spacegroup_number*, *extended*, *extensions*, *wyckoff_sequence*, *anonymous_wyckoff_sequence*, *anonymous_formula*, *formula_symbols*, *formula_counts*, *pbc*)

Bases: *httk.core.httkobject.HttkObject*

add_name(*name*)

add_names(*names*)

add_ref(*ref*)

add_refs(*refs*)

add_tag(*tag*, *val*)

add_tags(*tags*)

anonymous_formula

anonymous_wyckoff_sequence

classmethod create(*base_on_structure=None*, *lift_tags=True*, *lift_refs=True*)
struct: Structure object which forms the basis of this object

formula_counts

formula_symbols

get_names()

get_refs()

get_tag(*tag*)

get_tags()

number_of_elements

wyckoff_sequence

class httk.atomistic.**CompoundStructure**(*compound*, *structure*)
Bases: *httk.core.httkobject.HttkObject*

classmethod create(*compound*, *structure*)

class httk.atomistic.**StructurePhaseDiagram**(*structures*, *energies*, *hull_indices*, *competing_indices*, *hull_competing_indices*, *hull_distances*, *coord_system*, *phase_lines*)
Bases: *httk.core.httkobject.HttkObject*

Represents a phase diagram of structures

classmethod create(*structures*, *energies*)

get_phasediagram()

class httk.atomistic.**StructureRef**(*structure*, *reference*)
Bases: *httk.core.httkobject.HttkObject*

class httk.atomistic.**StructureTag**(*structure*, *tag*, *value*)
Bases: *httk.core.httkobject.HttkObject*

class httk.atomistic.**CompoundTag**(*compound*, *tag*, *value*)
Bases: *httk.core.httkobject.HttkObject*

class httk.atomistic.**CompoundRef**(*compound*, *reference*)
Bases: *httk.core.httkobject.HttkObject*

class httk.atomistic.**UnitcellStructure**(*assignments=None*, *uc_sites=None*, *uc_cell=None*)
Bases: *httk.core.httkobject.HttkObject*

A UnitcellStructure represents N sites of, e.g., atoms or ions, in any periodic or non-periodic arrangement. It keeps track of all the copies of the atoms within a unitcell.

The structure object is meant to be immutable and assumes that no internal variables are changed after its creation. All methods that 'changes' the object creates and returns a new, updated, structure object.

Naming conventions in httk.atomistic:

For cells:

cell = an abstract name for any reasonable representation of a 'cell' that defines the basis vectors used for representing the structure. When a 'cell' is returned, it is an object of type Cell

basis = a 3x3 sequence-type with (in rows) the three basis vectors (for a periodic system, defining the unit cell, and defines the unit of repetition for the periodic dimensions)

lengths_and_angles = (a,b,c,alpha,beta,gamma): the basis vector lengths and angles

niggli_matrix = ((v1*v1, v2*v2, v3*v3),(2*v2*v3, 2*v1*v3, 2*v2*v3)) where v1, v2, v3 are the vectors forming the basis

metric = ((v1*v1,v1*v2,v1*v3),(v2*v1,v2*v2,v2*v3),(v3*v1,v3*v2,v3*v3))

For sites:

These following prefixes are used to describe types of site specifications: representative cell/rc = only representative atoms are given, which are then to be repeated by structure symmetry group to give all sites

unit cell/uc = all atoms in unitcell

reduced = coordinates given in cell vectors

cartesian = coordinates given as direct cartesian coordinates

sites = used as an abstract name for any sensible representation of a list of coordinates and a cell, when a 'sites' is returned, it is an object of type Sites

counts = number of atoms of each type (one per entry in assignments)

coordgroups = coordinates represented as a 3-level-list of coordinates, e.g. [[[0,0,0],[0.5,0.5,0.5]],[[0.25,0.25,0.25]]] where level-1 list = groups: one group for each equivalent atom

counts and coords = one list with the number of atoms of each type (one per entry in assignments) and a 2-level list of coordinates.

For assignments of atoms, etc. to sites: assignments = abstract name for any representation of assignment of atoms. When returned, will be object of type Assignment.

atomic_numbers = a sequence of integers for the atomic number of each species

occupations = a sequence where the assignments are *repeated* for each coordinate as needed (prefixed with uc or rc depending on which coordinates)

For cell scaling: scaling = abstract name for any representation of cell scaling

scale = multiply all basis vectors with this number

volume = rescaling the cell such that it takes this volume

For periodicity: periodicity = abstract name of a representation of periodicity

pbc = 'periodic boundary conditions' = sequence of True and False for which basis vectors are periodic / non-periodic

nonperiodic_vecs = integer, number of basis vectors, counted from the first, which are non-periodic

For spacegroup: spacegroup = abstract name for any spacegroup representation. When returned, is of type Spacegroup.

hall_symbol = specifically the hall_symbol string representation of the spacegroup

classmethod create(*structure=None*, *uc_cell=None*, *uc_basis=None*, *uc_lengths=None*, *uc_angles=None*, *uc_niggli_matrix=None*, *uc_metric=None*, *uc_a=None*, *uc_b=None*, *uc_c=None*, *uc_alpha=None*, *uc_beta=None*, *uc_gamma=None*, *uc_sites=None*, *uc_reduced_coordgroups=None*, *uc_cartesian_coordgroups=None*, *uc_reduced_coords=None*, *uc_cartesian_coords=None*, *uc_reduced_occupationscoords=None*, *uc_cartesian_occupationscoords=None*, *uc_occupancies=None*, *uc_counts=None*, *uc_scale=None*, *uc_scaling=None*, *uc_volume=None*, *volume_per_atom=None*, *assignments=None*, *periodicity=None*, *nonperiodic_vecs=None*, *other_reps=None*, *refs=None*, *tags=None*)

A FullStructure represents N sites of, e.g., atoms or ions, in any periodic or non-periodic arrangement, where the positions of all cites are given (as opposed to a set of unique sites + symmetry operations).

This is a swiss-army-type constructor that allows several different ways to create a FullStructure object.

To create a new structure, three primary components are:

- cell: defines the basis vectors in which reduced coordinates are expressed, and the unit of repetition (*if* the structure has any periodicity - see the 'periodicity' parameter)
- assignments: a list of 'things' (atoms, ions, etc.) that goes on the sites in the structure
- sites: a sensible representation of location / coordinates of the sites.

Note: *uc_*-prefixes are consistently enforced for any quantity that would be different in a UniqueSitesStructure. This is to allow for painless change between the various structure-type objects without worrying about accidently using the wrong type of sites object.

Note: see help(Structure) for parameter naming conventions, i.e., what type of object is expected given a parameter name.

Input parameters:

- ONE OF: 'uc_cell'; 'uc_basis', 'uc_length_and_angles'; 'uc_niggli_matrix'; 'uc_metric'; all of: uc_a,uc_b,uc_c, uc_alpha, uc_beta, uc_gamma. (cell requires a Cell object or a very specific format, so unless you know what you are doing, use one of the others.)
- ONE OF: 'uc_assignments', 'uc_atomic_numbers', 'uc_occupations' (uc_assignments requires an Assignments object or a sequence.), uc_occupations repeats similar site assignments as needed
- ONE OF: 'uc_sites', 'uc_coords' (IF uc_occupations OR uc_counts are also given), or 'uc_B_C', where B=reduced or cartesian, C=coordgroups, coords, or occupationscoords

  Notes:

  - occupationscoords may differ from coords by *order*, since giving occupations as, e.g., ['H','O','H'] does not necessarily have the same order of the coordinates as the format of counts+coords as (2,1), ['H','O'].
  - uc_sites requires a Sites object or a python list on a very specific format, (so unless you know what you are doing, use one of the others.)
- **ONE OF: uc_scale, uc_volume, or volume_per_atom:** scale = multiply the basis vectors with this scaling factor, volume = the unit cell volume (overrides 'scale' if both are given) volume_per_atom = cell volume / number of atoms
- ONE OF periodicity or nonperiodic_vecs

formula_builder

pbc

supercell

transform(*matrix*, *max_search_cells=20*, *max_atoms=1000*)

uc_a

uc_alpha

uc_b

uc_basis

uc_beta

uc_c

uc_cartesian_coordgroups

uc_cartesian_coords

uc_cartesian_occupationscoords

uc_cell_orientation

uc_counts

uc_gamma

uc_lengths_and_angles

uc_reduced_coordgroups

uc_reduced_coords

uc_volume

uc_volume_per_atom

classmethod use(*other*)

class httk.atomistic.**RepresentativeStructure**(*assignments*, *rc_sites=None*, *rc_cell=None*)

Bases: *httk.core.httkobject.HttkObject*

A RepresentativeStructure represents N sites of, e.g., atoms or ions, in any periodic or non-periodic arrangement. It keeps track of a set of representative atoms in a unit cell (the conventional cell) and the symmetry group / operations that are to be applied to them to get all atoms.

This is meant to be a light-weight Structure object. For a heavy-weight with more functionality, use Structure.

The RepresentativeStructure object is meant to be immutable and assumes that no internal variables are changed after its creation. All methods that 'changes' the object creates and returns a new, updated, structure object.

clean()

classmethod create(*structure=None*, *rc_cell=None*, *rc_basis=None*, *rc_lengths=None*, *rc_angles=None*, *rc_niggli_matrix=None*, *rc_metric=None*, *rc_a=None*, *rc_b=None*, *rc_c=None*, *rc_alpha=None*, *rc_beta=None*, *rc_gamma=None*, *rc_sites=None*, *rc_reduced_coordgroups=None*, *rc_cartesian_coordgroups=None*, *rc_reduced_coords=None*, *rc_cartesian_coords=None*, *rc_reduced_occupationscoords=None*, *rc_cartesian_occupationscoords=None*, *rc_occupancies=None*, *rc_counts=None*, *wyckoff_symbols=None*, *multiplicities=None*, *spacegroup=None*, *hall_symbol=None*, *spacegroupnumber=None*, *setting=None*, *rc_scale=None*, *rc_scaling=None*, *rc_volume=None*, *vol_per_atom=None*, *assignments=None*, *periodicity=None*, *nonperiodic_vecs=None*, *refs=None*, *tags=None*)

A Structure represents N sites of, e.g., atoms or ions, in any periodic or non-periodic arrangement.

This is a swiss-army-type constructor that allows a selection between a large number of optional arguments.

To create a new structure, three primary components are:

- **cell: defines the basis vectors in which reduced coordinates are expressed, and the** unit of repetition (*if* the structure has any periodicity - see the 'periodicity' parameter)
- assignments: a list of 'things' (atoms, ions, etc.) that goes on the sites in the structure
- sites: a sensible representation of location / coordinates of the sites.

Note: *rc_*-prefixes are consistently enforced for any quantity that would be different in a UnitcellStructure. This is to allow for painless change between the various structure-type objects without worrying about accidently using the wrong type of sites object.

Input parameters:

- ONE OF: 'cell'; 'basis', 'length_and_angles'; 'niggli_matrix'; 'metric'; all of: a,b,c, alpha, beta, gamma. (cell requires a Cell object or a very specific format, so unless you know what you are doing, use one of the others.)
- ONE OF: 'assignments', 'atomic_numbers', 'occupancies' (assignments requires an Assignments object or a sequence.), occupations repeats similar site assignments as needed
- ONE OF: 'rc_sites', 'rc_coords' (IF rc_occupations OR rc_counts are also given), 'uc_coords' (IF uc_occupations OR uc_counts are also given) 'rc_B_C', where B=reduced or cartesian, C=coordgroups, coords, or occupationscoords

  Notes:

  – occupationscoords may differ from coords by *order*, since giving occupations as, e.g., ['H','O','H'] does not necessarily have the same order of the coordinates as the format of counts+coords as (2,1), ['H','O'].

  – rc_sites and uc_sites requires a Sites object or a very specific format, so unless you know what you are doing, use one of the others.)
- **ONE OF: scale or volume:** scale = multiply the basis vectors with this scaling factor, volume = the representative (conventional) cell volume (overrides 'scale' if both are given) volume_per_atom = cell volume / number of atoms
- ONE OF periodicity or nonperiodic_vecs

See help(Structure) for more information on the data format of all these data representations.

formula_builder

pbc

rc_a

rc_alpha

rc_b

rc_basis

rc_beta

rc_c

rc_cartesian_coordgroups

rc_cartesian_coords

rc_cartesian_occupationscoords

rc_cell_orientation

rc_gamma

rc_lengths_and_angles

rc_volume

uc_volume_per_atom

classmethod use(*other*)

Subpackages

httk.atomistic.atomisticio package

Submodules

httk.atomistic.atomisticio.structure_cif_io module

httk.atomistic.atomisticio.structure_cif_io.**cif_reader_httk_preprocessed**(*ioa*)

httk.atomistic.atomisticio.structure_cif_io.**cif_reader_that_can_only_read_isotropy_cif**(*ioa*)

httk.atomistic.atomisticio.structure_cif_io.**cif_to_struct**(*ioa, backends=['internal', 'cif2cell', 'ase', 'platon']*)

httk.atomistic.atomisticio.structure_cif_io.**cifdata_to_struct**(*cifdata, debug=False*)

httk.atomistic.atomisticio.structure_cif_io.**struct_to_cif**(*struct, ioa, backends=['httk']*)

httk.atomistic.atomisticio.structure_cif_io.**struct_to_cif_httk_simplified**(*struct, ioa, header=None, symops=True*)

httk.atomistic.atomisticio.structure_cif_io.**struct_to_cifdata**(*struct, entryid=None*)

httk.atomistic.atomisticio.structure_io module

httk.atomistic.atomisticio.structure_io.**load_struct**(*ioa*, *ext=None*, *filename=None*)
Load structure data from a file into a Structure

httk.atomistic.atomisticio.structure_io.**save_struct**(*struct*, *ioa*, *ext=None*)
Save structure data from a file into a Structure

httk.atomistic.atomisticio.structureioplugin module

class httk.atomistic.atomisticio.structureioplugin.**StructureIoPlugin**
Bases: *httk.core.httkobject.HttkPlugin*

classmethod load(*ioa*, *ext=None*, *filename=None*)

plugin_init(*struct*)

save(*ioa*, *ext=None*)

httk.atomistic.data package

Submodules

httk.atomistic.data.periodictable module

httk.atomistic.data.periodictable.**atomic_number**(*parse*)
Helper function to produce an atomic symbol if you have some kind of identifier, but does not know what it is.

httk.atomistic.data.periodictable.**atomic_number_isotope**(*parse*)
Helper function to produce an atomic symbol if you have some kind of identifier, but does not know what it is.

httk.atomistic.data.periodictable.**atomic_symbol**(*parse*)
Helper function to produce an atomic symbol if you have some kind of identifier, but does not know what it is.

httk.atomistic.data.periodictable.**most_common_mass**(*parse*)

httk.atomistic.data.spacegroups module

httk.atomistic.data.spacegroups.**find_index**(*parse*)

httk.atomistic.data.spacegroups.**get_proper_hm_symbol**(*parse*)

httk.atomistic.data.spacegroups.**spacegroup_get_hall**(*parse*)

httk.atomistic.data.spacegroups.**spacegroup_get_hm**(*parse*)

httk.atomistic.data.spacegroups.**spacegroup_get_number**(*parse*)

httk.atomistic.data.spacegroups.**spacegroup_get_number_and_setting**(*parse*)

httk.atomistic.data.spacegroups.**spacegroup_get_number_of_settings**(*number*)

httk.atomistic.data.spacegroups.**spacegroup_get_schoenflies**(*parse*)

httk.atomistic.results package

Submodules

httk.atomistic.results.relaxedcellresult module

class httk.atomistic.results.relaxedcellresult.**Result_RelaxedCellResult**(*computation*, *compound*, *relaxed_structure*, *primitive_cell*, *volume_per_atom*, *minimum_energy*)

Bases: *httk.core.computation.Result*

httk.atomistic.results.totalenergyresult module

class httk.atomistic.results.totalenergyresult.**Result_TotalEnergyResult**(*computation*, *structure*, *total_energy*)

Bases: *httk.core.computation.Result*

httk.atomistic.vis package

Submodules

httk.atomistic.vis.asestructurevisualizer module

class httk.atomistic.vis.asestructurevisualizer.**AseStructureVisualizer**(*struct*, *params={}*)

Bases: object

show()

wait()

httk.atomistic.vis.jmolstructurevisualizer module

class httk.atomistic.vis.jmolstructurevisualizer.**JmolStructureVisualizer**(*struct*, *params={}*)

Bases: object

bonds(*on*)

connections()

defaults_publish()

extbonds(*on*)

initialize()

polyhedra(*on*)

postconnect()

preconnect()

refresh()

repeat(*repetitions*)

rotate(*angle*)

save_and_quit(*filename*, *resx=3200*, *resy=2500*)

set_defaults()

show(*repeat=None*)

spin(*on=True*)

stop()

wait()

httk.atomistic.vis.structurephasediagramvisualizerplugin module

class httk.atomistic.vis.structurephasediagramvisualizerplugin.**StructurePhaseDiagramVisual**

Bases: *httk.core.httkobject.HttkPlugin*

plugin_init(*structurephasediagram*)

show(***params*)

httk.atomistic.vis.structurevisualizerplugin module

class httk.atomistic.vis.structurevisualizerplugin.**StructureVisualizerPlugin**

Bases: *httk.core.httkobject.HttkPlugin*

params()

plugin_init(*struct*)

show(*params={}, backends=['jmol', 'ase'], debug=False*)

wait()

Submodules

httk.atomistic.assignment module

class httk.atomistic.assignment.**Assignment**(*atomic_number*, *weight*, *ratio*, *magnetic_moment*)

Bases: *httk.core.httkobject.HttkObject*

Represents a possible vector of assignments

classmethod create(*siteassignment=None, atom=None, weight=None, ratio=None, magnetic_moment=[None, None, None]*)

Create a new siteassignment object site: integer for the site number that this atom is assigned to atomic number or symbol

get_extensions()

get_weight()

symbol

classmethod use(*old*)

httk.atomistic.assignment.**main**()

httk.atomistic.assignments module

class httk.atomistic.assignments.**Assignments**(*siteassignments*, *extensions=[]*)

Bases: *httk.core.httkobject.HttkObject*

Represents a possible vector of assignments

atomic_numbers

classmethod create(*assignments=None*)

Create a new assignment object,

assignments: a list-style object with one entry per 'atom type'. Any sensible type accepted, most notably, integers (for atom number)

extended

ratios

ratioslist

symbollists

symbols

to_basis()

classmethod use(*old*)

httk.atomistic.assignments.**main**()

httk.atomistic.cell module

class httk.atomistic.cell.**Cell**(*basis*, *lattice_system*, *orientation=1*)

Bases: *httk.core.httkobject.HttkObject*

Represents a cell (e.g., a unitcell, but also possibly just the basis vectors of a non-periodic system)

(The ability to represent the cell for a non-periodic system is also the reason this class is not called Lattice.)

clean()

coordgroups_cartesian_to_reduced(*coordgroups*)

coordgroups_reduced_to_cartesian(*coordgroups*)

coords_cartesian_to_reduced(*coords*)

coords_reduced_to_cartesian(*coords*)

classmethod create(*cell=None*, *basis=None*, *metric=None*, *niggli_matrix=None*, *a=None*, *b=None*, *c=None*, *alpha=None*, *beta=None*, *gamma=None*, *lengths=None*, *angles=None*, *cosangles=None*, *scale=None*, *scaling=None*, *volume=None*, *periodicity=None*, *nonperiodic_vecs=None*, *orientation=1*, *hall=None*, *lattice_system=None*, *eps=0*)

Create a new cell object,

cell: any one of the following:

- a 3x3 array with (in rows) the three basis vectors of the cell (a non-periodic system should conventionally use an identity matrix)
- a dict with a single key 'niggli_matrix' with a 3x2 array with the Niggli Matrix representation of the cell
- a dict with 6 keys, 'a', 'b', 'c', 'alpha', 'beta', 'gamma' giving the cell parameters as floats

scaling: free form input parsed for a scale. positive value = multiply basis vectors by this value negative value = rescale basis vectors so that cell volume becomes abs(value).

scale: set to non-None to multiply all cell vectors with this factor

volume: set to non-None if the basis vectors only give directions, and the volume of the cell should be this value (overrides scale)

periodicity: free form input parsed for periodicity sequence: True/False for each basis vector being periodic integer: number of non-periodic basis vectors

hall: giving the hall symbol makes it possible to determine the lattice system without numerical inaccuracy

lattice_system: any one of: 'cubic', 'hexagonal', 'tetragonal', 'orthorhombic', 'trigonal', 'triclinic', 'monoclinic', 'unknown'

get_axes_standard_order_transform()

get_normalized()

get_normalized_longestvec()

is_point_inside(*cartesian_coord*)

normalization_longestvec_scale

Get the factor with which a normalized version of this cell needs to be multiplied to reproduce this cell.

I.e. self = (normalization_scale)*self.get_normalized()

normalization_scale

scaling()

classmethod use(*other*)

volume

httk.atomistic.cell.**main**()

httk.atomistic.cellshape module

class httk.atomistic.cellshape.**CellShape**(*niggli_matrix*, *orientation=1*, *basis=None*)

Bases: *httk.core.httkobject.HttkObject*

Represents a cell (e.g., a unitcell, but also possibly just the basis vectors of a non-periodic system)

basis

clean()

coordgroups_cartesian_to_reduced(*coordgroups*)

coordgroups_reduced_to_cartesian(*coordgroups*)

coords_cartesian_to_reduced(*coords*)

coords_reduced_to_cartesian(*coords*)

classmethod create(*cellshape=None*, *basis=None*, *metric=None*, *niggli_matrix=None*, *a=None*, *b=None*, *c=None*, *alpha=None*, *beta=None*, *gamma=None*, *lengths=None*, *angles=None*, *scale=None*, *scaling=None*, *volume=None*, *periodicity=None*, *nonperiodic_vecs=None*, *orientation=1*)

Create a new cell object,

cell: any one of the following:

- a 3x3 array with (in rows) the three basis vectors of the cell (a non-periodic system should conventionally use an identity matrix)
- a dict with a single key 'niggli_matrix' with a 3x2 array with the Niggli Matrix representation of the cell
- a dict with 6 keys, 'a', 'b', 'c', 'alpha', 'beta', 'gamma' giving the cell parameters as floats

scaling: free form input parsed for a scale. positive value = multiply basis vectors by this value negative value = rescale basis vectors so that cell volume becomes abs(value).

scale: set to non-None to multiply all cell vectors with this factor

volume: set to non-None if the basis vectors only give directions, and the volume of the cell should be this value (overrides scale)

periodicity: free form input parsed for periodicity sequence: True/False for each basis vector being periodic integer: number of non-periodic basis vectors

is_point_inside(*cartesian_coord*)

scaling()

httk.atomistic.cellshape.**main**()

httk.atomistic.cellutils module

httk.atomistic.cellutils.**angles_to_cosangles**(*angles*)

httk.atomistic.cellutils.**basis_determinant**(*basis*)

httk.atomistic.cellutils.**basis_to_niggli_and_orientation**(*basis*)

httk.atomistic.cellutils.**cell_to_basis**(*cell*)

httk.atomistic.cellutils.**get_primitive_to_conventional_basis_transform**(*basis*, *eps=0.0001*)

Figures out how the 'likley' transform of a primitive cell for getting to the conventional basis

This may not be foolproof, and mostly works for re-inverting cells generated by lengths_and_cosangles_to_conventional_basis. (It should only be used when getting something that isn't really the conventional cell does not equal catastrophic failure, just, e.g., a non-optimal representation.)

httk.atomistic.cellutils.**lattice_system_from_lengths_and_cosangles**(*lengths*, *cosangles*, *eps=0*)

Identifies lattice system from a list of cell axis lengths and cosine of angles between them Returns string: 'cubic', 'tetragonal', 'orthorombic', 'hexagonal', 'monoclinic', 'rhombohedral' or 'triclinic'

Note: if axis order is not the standard one (e.g., gamma=120 for hexagonal), the lattice system will come out as triclinic. This way the outcome matches corresponding standard hall symbols, otherwise hall symbol and generated cells not technically match.

If you seek to re-order axes to the standard order, use standard_order_axes_transform on your basis matrix first.

httk.atomistic.cellutils.**lattice_system_from_niggli**(*niggli_matrix*, *eps=0*)

Identifies lattice system from niggli matrix. Returns string: 'cubic', 'tetragonal', 'orthorombic', 'hexagonal', 'monoclinic', 'rhombohedral' or 'triclinic'

Note: if axis order is not the standard one (e.g., gamma=120 for hexagonal), the lattice system will come out as triclinic. This way the outcome matches corresponding standard hall symbols, otherwise hall symbol and generated cells not technically match.

If you seek to re-order axes to the standard order, use standard_order_axes_transform on your basis matrix first.

httk.atomistic.cellutils.**lengths_and_angles_to_niggli**(*lengths*, *angles*)

httk.atomistic.cellutils.**lengths_and_cosangles_to_conventional_basis**(*lengths*, *cosangles*, *lattice_system=None*, *orientation=1*, *eps=0*)

Returns the conventional cell basis given a list of lengths and cosine of angles

Note: if your basis vector order does not follow the conventions for hexagonal and monoclinic cells, you get the triclinic conventional cell.

Conventions: in hexagonal cell gamma=120 degrees, i.e, cosangles[2]=-1/2, in monoclinic cells beta =/= 90 degrees.

httk.atomistic.cellutils.**lengths_and_cosangles_to_niggli**(*lengths*, *cosangles*)

httk.atomistic.cellutils.**main**()

httk.atomistic.cellutils.**metric_to_niggli**(*cell*)

httk.atomistic.cellutils.**niggli_scale_to_vol**(*niggli_matrix*, *scale*)

httk.atomistic.cellutils.**niggli_to_basis**(*niggli_matrix*, *orientation=1*)

httk.atomistic.cellutils.**niggli_to_conventional_basis**(*niggli_matrix*, *lattice_system=None*, *orientation=1*, *eps=0.0001*)

Returns the conventional cell given a niggli_matrix

Note: if your basis vector order does not follow the conventions for hexagonal and monoclinic cells, you get the triclinic conventional cell.

Conventions: in hexagonal cell gamma=120 degrees., in monoclinic cells beta =/= 90 degrees.

httk.atomistic.cellutils.**niggli_to_lengths_and_angles**(*niggli_matrix*)

httk.atomistic.cellutils.**niggli_to_lengths_and_trigangles**(*niggli_matrix*)

httk.atomistic.cellutils.**niggli_to_metric**(*niggli*)

httk.atomistic.cellutils.**scale_to_vol**(*basis*, *scale*)

httk.atomistic.cellutils.**scaling_to_volume**(*basis*, *scaling*)

httk.atomistic.cellutils.**standard_order_axes_transform**(*niggli_matrix*, *lattice_system*, *eps=0*, *return_identity_if_no_transform_needed=False*)

Returns the transform that re-orders the axes to standard order for each possible lattice system.

Note: returns None if no transform is needed, to make it easy to skip the transform in that case. If you want the identity matrix instead, set parameter return_identity_if_no_transform_needed = True,

httk.atomistic.cellutils.**vol_to_scale**(*basis*, *vol*)

httk.atomistic.cli module

httk.atomistic.cli.**main**(*commands*, *args*)

httk.atomistic.compound module

class httk.atomistic.compound.**Compound**(*element_wyckoff_sequence*, *formula*, *spacegroup_number*, *extended*, *extensions*, *wyckoff_sequence*, *anonymous_wyckoff_sequence*, *anonymous_formula*, *formula_symbols*, *formula_counts*, *pbc*)

Bases: *httk.core.httkobject.HttkObject*

add_name(*name*)

add_names(*names*)

add_ref(*ref*)

add_refs(*refs*)

add_tag(*tag*, *val*)

add_tags(*tags*)

anonymous_formula

anonymous_wyckoff_sequence

classmethod create(*base_on_structure=None*, *lift_tags=True*, *lift_refs=True*)

struct: Structure object which forms the basis of this object

formula_counts

formula_symbols

get_names()

get_refs()

get_tag(*tag*)

get_tags()

number_of_elements

wyckoff_sequence

class httk.atomistic.compound.**CompoundName**(*compound*, *name*)
Bases: *httk.core.httkobject.HttkObject*

class httk.atomistic.compound.**CompoundRef**(*compound*, *reference*)
Bases: *httk.core.httkobject.HttkObject*

class httk.atomistic.compound.**CompoundStructure**(*compound*, *structure*)
Bases: *httk.core.httkobject.HttkObject*

classmethod create(*compound*, *structure*)

class httk.atomistic.compound.**CompoundTag**(*compound*, *tag*, *value*)
Bases: *httk.core.httkobject.HttkObject*

class httk.atomistic.compound.**ComputationRelatedCompound**(*computation*, *compound*)
Bases: *httk.core.httkobject.HttkObject*

classmethod create(*computation*, *compound*)

httk.atomistic.compound.**main**()

httk.atomistic.formulautils module

class httk.atomistic.formulautils.**StructureFormulaPlugin**
Bases: *httk.core.httkobject.HttkPlugin*

plugin_init(*struct*)

httk.atomistic.representativesites module

class httk.atomistic.representativesites.**RepresentativeSites**(*reduced_coordgroups=None*, *cartesian_coordgroups=None*, *reduced_coords=None*, *cartesian_coords=None*, *counts=None*, *hall_symbol=None*, *pbc=None*, *wyckoff_symbols=None*, *multiplicities=None*)
Bases: *httk.atomistic.sites.Sites*

Represents any collection of sites in a unitcell

anonymous_wyckoff_sequence

clean()

classmethod create(*sites=None*, *reduced_coordgroups=None*, *reduced_coords=None*, *counts=None*, *spacegroup=None*, *hall_symbol=None*, *spacegroupnumber=None*, *setting=None*, *periodicity=None*, *wyckoff_symbols=None*, *multiplicities=None*, *occupancies=None*, *pbc=None*)
Create a new sites object

crystal_system

get_uc_sites()

lattice_symbol

lattice_system

tidy()

total_number_of_atoms

wyckoff_sequence

httk.atomistic.representativesites.**main**()

httk.atomistic.representativestructure module

class httk.atomistic.representativestructure.**RepresentativeStructure**(*assignments*, *rc_sites=None*, *rc_cell=None*)

Bases: *httk.core.httkobject.HttkObject*

A RepresentativeStructure represents N sites of, e.g., atoms or ions, in any periodic or non-periodic arrangement. It keeps track of a set of representative atoms in a unit cell (the conventional cell) and the symmetry group / operations that are to be applied to them to get all atoms.

This is meant to be a light-weight Structure object. For a heavy-weight with more functionality, use Structure.

The RepresentativeStructure object is meant to be immutable and assumes that no internal variables are changed after its creation. All methods that 'changes' the object creates and returns a new, updated, structure object.

clean()

classmethod create(*structure=None*, *rc_cell=None*, *rc_basis=None*, *rc_lengths=None*, *rc_angles=None*, *rc_niggli_matrix=None*, *rc_metric=None*, *rc_a=None*, *rc_b=None*, *rc_c=None*, *rc_alpha=None*, *rc_beta=None*, *rc_gamma=None*, *rc_sites=None*, *rc_reduced_coordgroups=None*, *rc_cartesian_coordgroups=None*, *rc_reduced_coords=None*, *rc_cartesian_coords=None*, *rc_reduced_occupationscoords=None*, *rc_cartesian_occupationscoords=None*, *rc_occupancies=None*, *rc_counts=None*, *wyckoff_symbols=None*, *multiplicities=None*, *spacegroup=None*, *hall_symbol=None*, *spacegroupnumber=None*, *setting=None*, *rc_scale=None*, *rc_scaling=None*, *rc_volume=None*, *vol_per_atom=None*, *assignments=None*, *periodicity=None*, *nonperiodic_vecs=None*, *refs=None*, *tags=None*)

A Structure represents N sites of, e.g., atoms or ions, in any periodic or non-periodic arrangement.

This is a swiss-army-type constructor that allows a selection between a large number of optional arguments.

To create a new structure, three primary components are:

- **cell: defines the basis vectors in which reduced coordinates are expressed, and the** unit of repetition (*if* the structure has any periodicity - see the 'periodicity' parameter)
- assignments: a list of 'things' (atoms, ions, etc.) that goes on the sites in the structure
- sites: a sensible representation of location / coordinates of the sites.

Note: *rc_*-prefixes are consistently enforced for any quantity that would be different in a UnitcellStructure. This is to allow for painless change between the various structure-type objects without worrying about accidently using the wrong type of sites object.

Input parameters:

- ONE OF: ‘cell’; ‘basis’, ‘length_and_angles’; ‘niggli_matrix’; ‘metric’; all of: a,b,c, alpha, beta, gamma. (cell requires a Cell object or a very specific format, so unless you know what you are doing, use one of the others.)
- ONE OF: ‘assignments’, ‘atomic_numbers’, ‘occupancies’ (assignments requires an Assignments object or a sequence.), occupations repeats similar site assignments as needed
- ONE OF: ‘rc_sites’, ‘rc_coords’ (IF rc_occupations OR rc_counts are also given), ‘uc_coords’ (IF uc_occupations OR uc_counts are also given) ‘rc_B_C’, where B=reduced or cartesian, C=coordgroups, coords, or occupationscoords

  Notes:

  – occupationscoords may differ from coords by *order*, since giving occupations as, e.g., [‘H’,’O’,’H’] does not necessarily have the same order of the coordinates as the format of counts+coords as (2,1), [‘H’,’O’].

  – rc_sites and uc_sites requires a Sites object or a very specific format, so unless you know what you are doing, use one of the others.)
- **ONE OF: scale or volume:** scale = multiply the basis vectors with this scaling factor, volume = the representative (conventional) cell volume (overrides ‘scale’ if both are given) volume_per_atom = cell volume / number of atoms
- ONE OF periodicity or nonperiodic_vecs

See help(Structure) for more information on the data format of all these data representations.

formula_builder

pbc

rc_a

rc_alpha

rc_b

rc_basis

rc_beta

rc_c

rc_cartesian_coordgroups

rc_cartesian_coords

rc_cartesian_occupationscoords

rc_cell_orientation

rc_gamma

rc_lengths_and_angles

rc_volume

uc_volume_per_atom

classmethod use(*other*)

httk.atomistic.representativestructure.**main**()

httk.atomistic.siteassignment module

class httk.atomistic.siteassignment.**SiteAssignment**(*assignments*)

Bases: *httk.core.httkobject.HttkObject*

Represents a possible vector of assignments

atomic_number

atomic_numbers

classmethod create(*assignments=None*)

Create a new assignment object,

assignments: a list-style object with one entry per 'atom type'. Any sensible type accepted, most notably, integers (for atom number)

get_extensions()

ratio

ratios

symbol

symbols

to_basis()

classmethod use(*old*)

httk.atomistic.siteassignment.**main**()

httk.atomistic.sites module

class httk.atomistic.sites.**Sites**(*reduced_coordgroups=None*, *reduced_coords=None*, *counts=None*, *hall_symbol=None*, *pbc=None*)

Bases: *httk.core.httkobject.HttkObject*

Represents any collection of sites in a unitcell

anonymous_formula

clean()

coords_groupnumber

counts

classmethod create(*sites=None*, *reduced_coordgroups=None*, *reduced_coords=None*, *counts=None*, *occupancies=None*, *spacegroup=None*, *hall_symbol=None*, *spacegroupnumber=None*, *setting=None*, *pbc=None*, *periodicity=None*)

Create a new sites object

get_cartesian_coordgroups(*cell*)

get_cartesian_coords(*scale*)

reduced_coordgroups

reduced_coords

total_number_of_atoms

classmethod use(*old*, *cell=None*, *hall_symbol=None*, *periodicity=None*)

httk.atomistic.sites.**main**()

httk.atomistic.sitesutils module

httk.atomistic.sitesutils.**abstract_symbol**(*count*)

httk.atomistic.sitesutils.**anonymous_formula**(*filled_counts*)

httk.atomistic.sitesutils.**clean_coordgroups_and_assignments**(*coordgroups*, *assignments*)

httk.atomistic.sitesutils.**coordgroups_cartesian_to_reduced**(*coordgroups*, *basis*)

httk.atomistic.sitesutils.**coordgroups_reduced_to_cartesian**(*cell*, *coordgroups*)

httk.atomistic.sitesutils.**coordgroups_reduced_to_unitcell**(*coordgroups*, *hall_symbol*, *eps=Fraction(1, 1000)*)

httk.atomistic.sitesutils.**coordgroups_to_coords**(*coordgroups*)

httk.atomistic.sitesutils.**coords_and_counts_to_coordgroups**(*coords*, *counts*)

httk.atomistic.sitesutils.**coords_and_occupancies_to_coordgroups_and_assignments**(*coords*, *occupancies*)

httk.atomistic.sitesutils.**coords_reduced_to_cartesian**(*cell*, *coords*)

httk.atomistic.sitesutils.**coords_to_coordgroups**(*coords*, *counts*)

httk.atomistic.sitesutils.**coordswap**(*fromidx*, *toidx*, *cell*, *coordgroups*)

httk.atomistic.sitesutils.**main**()

httk.atomistic.sitesutils.**normalized_formula_parts**(*assignments*, *ratios*, *counts*)

httk.atomistic.sitesutils.**pbc_to_nonperiodic_vecs**(*pbc*)

httk.atomistic.sitesutils.**periodicity_to_pbc**(*periodicity*)

httk.atomistic.sitesutils.**sites_tidy**(*sites, backends=['platon']*)

httk.atomistic.sitesutils.**sort_coordgroups**(*coordgroups*, *individual_data*)

httk.atomistic.sitesutils.**structure_reduced_coordgroups_to_representative**(*coordgroups, cell, spacegroup, backends=['isotropy']*)

httk.atomistic.spacegroup module

class httk.atomistic.spacegroup.**Spacegroup**(*hall_symbol*)

Bases: *httk.core.httkobject.HttkObject*

Represents a spacegroup

classmethod create(*spacegroup=None*, *hall_symbol=None*, *hm_symbol=None*, *spacegroupnumber=None*, *setting=None*, *symops=None*)

Create a new spacegroup object,

Give ONE OF hall_symbol or spacegroup.

hall_symbol = a string giving the hall symbol of the spacegroup

spacegroup = a spacegroup on any reasonable format that can be parsed, e.g., an integer (spacegroup number)

setting = if only a spacegroup number is given, this allows also specifying a setting.

number

number_and_setting

httk.atomistic.spacegroup.**main**()

httk.atomistic.spacegrouputils module

httk.atomistic.spacegrouputils.**check_symop**(*coordgroups*, *symopv*)

httk.atomistic.spacegrouputils.**crystal_system_from_hall**(*hall_symb*)

httk.atomistic.spacegrouputils.**crystal_system_from_spacegroupnbr**(*spacegroupnr*)

httk.atomistic.spacegrouputils.**filter_hm**(*hm*, *setting=None*, *halls=None*)

httk.atomistic.spacegrouputils.**filter_itcnbr_setting**(*itcnbr*, *setting=None*, *halls=None*)

httk.atomistic.spacegrouputils.**filter_sf**(*sf*, *halls=None*)

httk.atomistic.spacegrouputils.**filter_symops**(*symops*, *halls=None*)

httk.atomistic.spacegrouputils.**get_hall**(*hall*)

httk.atomistic.spacegrouputils.**get_hm_setting**(*hm*, *setting*)

httk.atomistic.spacegrouputils.**get_itcnbr_setting**(*itcnbr*, *setting*)

httk.atomistic.spacegrouputils.**get_nonstandard_hall**(*nonstd_hall*)

httk.atomistic.spacegrouputils.**get_symops**(*hall*)

httk.atomistic.spacegrouputils.**get_symops_strs**(*hall*)

httk.atomistic.spacegrouputils.**get_symopshash**(*hall*)

httk.atomistic.spacegrouputils.**lattice_symbol_from_hall**(*hall*)

httk.atomistic.spacegrouputils.**lattice_system_from_hall**(*hall*)

httk.atomistic.spacegrouputils.**lattice_type_from_hall**(*hall*)

httk.atomistic.spacegrouputils.**main**()

httk.atomistic.spacegrouputils.**reduce_by_symops**(*coordgroups*, *symopvs*, *hall_symbol*)

httk.atomistic.spacegrouputils.**spacegroup_filter**(*parse*)

httk.atomistic.spacegrouputils.**spacegroup_filter_specific**(*hall=None*, *hm=None*, *itcnbr=None*, *setting=None*, *symops=None*, *halls=None*)

httk.atomistic.spacegrouputils.**spacegroup_get_hall**(*parse*)

httk.atomistic.spacegrouputils.**spacegroup_get_hm**(*parse*)

httk.atomistic.spacegrouputils.**spacegroup_get_number**(*parse*)

httk.atomistic.spacegrouputils.**spacegroup_get_number_and_setting**(*parse*)

httk.atomistic.spacegrouputils.**spacegroup_get_schoenflies**(*parse*)

httk.atomistic.spacegrouputils.**spacegroup_parse**(*parse*)

httk.atomistic.spacegrouputils.**symopshash**(*symops*)

httk.atomistic.spacegrouputils.**symopsmatrix**(*symop*)

httk.atomistic.spacegrouputils.**symopstuple**(*symop*, *val_transform=<function val_to_tuple>*)

httk.atomistic.spacegrouputils.**trivial_symmetry_reduce**(*coordgroups*)
Looks for 'trivial' ways to reduce the coordinates in the given coordgroups by a standard set of symmetry operations. This is not a symmetry finder (and it is not intended to be), but for a standard primitive cell taken from a standard conventional cell, it reverses the primitive unit cell coordgroups into the symmetry reduced coordgroups.

httk.atomistic.spacegrouputils.**val_to_tuple**(*val*)

httk.atomistic.spacegrouputils.**wyckoff_symbol_matcher**(*wyckoffs*, *coord*)

httk.atomistic.structure module

class httk.atomistic.structure.**Structure**(*assignments*, *rc_sites=None*, *rc_cell=None*, *other_reps=None*)

Bases: *httk.core.httkobject.HttkObject*

A Structure represents N sites of, e.g., atoms or ions, in any periodic or non-periodic arrangement. The structure object is meant to be immutable and assumes that no internal variables are changed after its creation. All methods that 'changes' the object creates and returns a new, updated, structure object.

This is the general heavy weight structure object. For lightweight structure objects, use UnitcellStructure or RepresentativeStructure.

Naming conventions in httk.atomistic:

Structure cell type abbreviations:

rc = Representative cell: only representative atoms are given inside the conventional cell. they need to be replicated by the symmetry elements.

uc = Unit cell: any (imprecisely defined) unit cell (usually the unit cell used to define the structure if it was not done via a representative cell.) with all atoms inside.

pc = Primitive unit cell: a smallest possible unit cell (the standard one) with all atoms inside.

cc = Conventional unit cell: the high symmetry unit cell (rc) with all atoms inside.

For cells:

cell = an abstract name for any reasonable representation of a 'cell' that defines the basis vectors used for representing the structure. When a 'cell' is returned, it is an object of type Cell

basis = a 3x3 sequence-type with (in rows) the three basis vectors (for a periodic system, defining the unit cell, and defines the unit of repetition for the periodic dimensions)

lengths_and_angles = (a,b,c,alpha,beta,gamma): the basis vector lengths and angles

niggli_matrix = ((v1*v1, v2*v2, v3*v3),(2*v2*v3, 2*v1*v3, 2*v2*v3)) where v1, v2, v3 are the vectors forming the basis

metric = ((v1*v1,v1*v2,v1*v3),(v2*v1,v2*v2,v2*v3),(v3*v1,v3*v2,v3*v3))

For sites:

These following prefixes are used to describe types of site specifications: representative cell/rc = only representative atoms are given, which are then to be repeated by structure symmetry group to give all sites

unit cell/uc = all atoms in unitcell

reduced = coordinates given in cell vectors

cartesian = coordinates given as direct cartesian coordinates

sites = used as an abstract name for any sensible representation of a list of coordinates and a cell, when a 'sites' is returned, it is an object of type Sites

counts = number of atoms of each type (one per entry in assignments)

coordgroups = coordinates represented as a 3-level-list of coordinates, e.g. [[[0,0,0],[0.5,0.5,0.5]],[[0.25,0.25,0.25]]] where level-1 list = groups: one group for each equivalent atom

counts and coords = one list with the number of atoms of each type (one per entry in assignments) and a 2-level list of coordinates.

For assignments of atoms, etc. to sites: assignments = abstract name for any representation of assignment of atoms. When returned, will be object of type Assignment.

atomic_numbers = a sequence of integers for the atomic number of each species

occupations = a sequence where the assignments are *repeated* for each coordinate as needed (prefixed with uc or rc depending on which coordinates)

For cell scaling: scaling = abstract name for any representation of cell scaling

scale = multiply all basis vectors with this number

volume = rescaling the cell such that it takes this volume

For periodicity: periodicity = abstract name of a representation of periodicity

pbc = 'periodic boundary conditions' = sequence of True and False for which basis vectors are periodic / non-periodic

nonperiodic_vecs = integer, number of basis vectors, counted from the first, which are non-periodic

For spacegroup: spacegroup = abstract name for any spacegroup representation. When returned, is of type Spacegroup.

hall_symbol = specifically the hall_symbol string representation of the spacegroup

add_ref(*ref*)

add_refs(*refs*)

add_tag(*tag*, *val*)

add_tags(*tags*)

anonymous_formula

anonymous_wyckoff_sequence

cc

cc_formula_parts

clean()

classmethod create(*structure=None*, *assignments=None*, *rc_cell=None*, *rc_basis=None*, *rc_lengths=None*, *rc_angles=None*, *rc_cosangles=None*, *rc_niggli_matrix=None*, *rc_metric=None*, *rc_a=None*, *rc_b=None*, *rc_c=None*, *rc_alpha=None*, *rc_beta=None*, *rc_gamma=None*, *rc_sites=None*, *rc_reduced_coordgroups=None*, *rc_cartesian_coordgroups=None*, *rc_reduced_coords=None*, *rc_cartesian_coords=None*, *rc_reduced_occupationscoords=None*, *rc_cartesian_occupationscoords=None*, *rc_occupancies=None*, *rc_counts=None*, *wyckoff_symbols=None*, *multiplicities=None*, *spacegroup=None*, *hall_symbol=None*, *spacegroupnumber=None*, *setting=None*, *rc_scale=None*, *rc_scaling=None*, *rc_volume=None*, *uc_cell=None*, *uc_basis=None*, *uc_lengths=None*, *uc_angles=None*, *uc_cosangles=None*, *uc_niggli_matrix=None*, *uc_metric=None*, *uc_a=None*, *uc_b=None*, *uc_c=None*, *uc_alpha=None*, *uc_beta=None*, *uc_gamma=None*, *uc_sites=None*, *uc_reduced_coordgroups=None*, *uc_cartesian_coordgroups=None*, *uc_reduced_coords=None*, *uc_cartesian_coords=None*, *uc_reduced_occupationscoords=None*, *uc_cartesian_occupationscoords=None*, *uc_occupancies=None*, *uc_counts=None*, *uc_scale=None*, *uc_scaling=None*, *uc_volume=None*, *uc_is_primitive_cell=False*, *uc_is_conventional_cell=False*, *volume_per_atom=None*, *periodicity=None*, *nonperiodic_vecs=None*, *refs=None*, *tags=None*)

A Structure represents N sites of, e.g., atoms or ions, in any periodic or non-periodic arrangement.

This is a swiss-army-type constructor that allows a selection between a large number of optional arguments.

Note: if redundant and non-compatible information is given, the behavior is undefined. E.g., don't try to call this with a structure + a volume in hopes to get a copy with rescaled volume.

To create a new structure, three primary components are:

- **cell: defines the basis vectors in which reduced coordinates are expressed, and the** unit of repetition (*if* the structure has any periodicity - see the 'periodicity' parameter)
- assignments: a list of 'things' (atoms, ions, etc.) that goes on the sites in the structure
- sites: a sensible representation of location / coordinates of the sites.

Note: *rc_*-prefixes are consistently enforced for any quantity that would be different in a UnitcellStructure. This is to allow for painless change between the various structure-type objects without worrying about accidently using the wrong type of sites object.

Input parameters:

- ONE OF: 'cell'; 'basis', 'length_and_angles'; 'niggli_matrix'; 'metric'; all of: a,b,c, alpha, beta, gamma. (cell requires a Cell object or a very specific format, so unless you know what you are doing, use one of the others.)
- ONE OF: 'assignments', 'atomic_numbers', 'occupancies' (assignments requires an Assignments object or a sequence.), occupations repeats similar site assignments as needed
- ONE OF: 'rc_sites', 'rc_coords' (IF rc_occupations OR rc_counts are also given), 'uc_coords' (IF uc_occupations OR uc_counts are also given) 'rc_B_C', where B=reduced or cartesian, C=coordgroups, coords, or occupationscoords

  Notes:

  - occupationscoords may differ from coords by *order*, since giving occupations as, e.g., ['H','O','H'] does not necessarily have the same order of the coordinates as the format of counts+coords as (2,1), ['H','O'].
  - rc_sites and uc_sites requires a Sites object or a very specific format, so unless you know what you are doing, use one of the others.)
- **ONE OF: scale or volume:** scale = multiply the basis vectors with this scaling factor, volume = the representative (conventional) cell volume (overrides 'scale' if both are given) volume_per_atom = cell volume / number of atoms
- ONE OF periodicity or nonperiodic_vecs

See help(Structure) for more information on the data format of all these data representations.

element_wyckoff_sequence

extended

extensions

find_symmetry()

formula

formula_counts

formula_spaceseparated

formula_symbols

get_refs()

get_tag(*tag*)

get_tags()

hall_symbol

has_rc_repr
Returns True if the structure already contains the representative coordinates + spacegroup, and thus can be queried for this data without launching an expensive symmetry finder operation.

has_uc_repr
Returns True if the structure contains any unit cell-type coordinate representation, and thus can be queried for this data without launching a somewhat expensive cell filling operation.

io

number_of_elements

pbc

pc

pc_a

pc_alpha

pc_b

pc_beta

pc_c

pc_counts

pc_formula_parts

pc_gamma

pc_nbr_atoms

pc_volume

rc

rc_a

rc_alpha

rc_b

rc_basis

rc_beta

rc_c

rc_cartesian_coordgroups

rc_cartesian_coords

rc_cartesian_occupationscoords

rc_cell_orientation

rc_counts

rc_gamma

rc_lengths_and_angles

rc_nbr_atoms

rc_occupancies

rc_occupationssymbols

rc_reduced_coordgroups

rc_reduced_coords

rc_volume

spacegroup

spacegroup_number

spacegroup_number_and_setting

supercell

symbols

tidy()

transform(*matrix*, *max_search_cells=20*, *max_atoms=1000*)

uc

uc_a

uc_alpha

uc_b

uc_basis

uc_beta

uc_c

uc_cartesian_coordgroups

uc_cartesian_coords

uc_cartesian_occupationscoords

uc_cell

uc_cell_orientation

uc_counts

uc_formula

uc_formula_counts

uc_formula_parts

uc_formula_symbols

uc_gamma

uc_lengths_and_angles

uc_nbr_atoms

uc_occupancies

uc_occupationssymbols

uc_reduced_coordgroups

uc_reduced_coords

uc_reduced_occupationscoords

uc_sites

uc_volume

classmethod use(*other*)

volume_per_atom

wyckoff_sequence

class httk.atomistic.structure.**StructureRef**(*structure*, *reference*)

Bases: *httk.core.httkobject.HttkObject*

class httk.atomistic.structure.**StructureTag**(*structure*, *tag*, *value*)

Bases: *httk.core.httkobject.HttkObject*

httk.atomistic.structure.**main**()

httk.atomistic.structurephasediagram module

class httk.atomistic.structurephasediagram.**StructurePhaseDiagram**(*structures*, *energies*, *hull_indices*, *competing_indices*, *hull_competing_indices*, *hull_distances*, *coord_system*, *phase_lines*)

Bases: *httk.core.httkobject.HttkObject*

Represents a phase diagram of structures

classmethod create(*structures*, *energies*)

get_phasediagram()

class httk.atomistic.structurephasediagram.**StructurePhaseDiagramCompetingIndicies**(*indices*)

Bases: *httk.core.httkobject.HttkObject*

classmethod create(*indices*)

httk.atomistic.structurephasediagram.**main**()

httk.atomistic.structurephasediagram.**setup_phasediagram**(*structures*, *energies*)

httk.atomistic.structureutils module

httk.atomistic.structureutils.**abstract_formula**(*filled_counts*)

httk.atomistic.structureutils.**abstract_symbol**(*count*)

httk.atomistic.structureutils.**basis_determinant**(*basis*)

httk.atomistic.structureutils.**basis_scale_to_vol**(*basis*, *scale*)

httk.atomistic.structureutils.**basis_to_niggli**(*basis*)

httk.atomistic.structureutils.**basis_vol_to_scale**(*basis*, *vol*)

httk.atomistic.structureutils.**cartesian_to_reduced**(*cell*, *coordgroups*)

httk.atomistic.structureutils.**clean_coordgroups_and_assignments**(*coordgroups*, *assignments*)

httk.atomistic.structureutils.**coordgroups_and_assignments_to_coords_and_occupancies**(*coordgroups*, *assignments*)

httk.atomistic.structureutils.**coordgroups_and_assignments_to_symbols**(*coordgroups*, *assignmentobj*)

Return a list of atomic symbols, repeated as needed

httk.atomistic.structureutils.**coordgroups_cartesian_to_reduced**(*coordgroups*, *basis*)

httk.atomistic.structureutils.**coordgroups_reduced_rc_to_unitcellsites**(*coordgroups, basis, hall_symbol, backends=['cif2cell', 'internal', 'ase']*)

httk.atomistic.structureutils.**coordgroups_reduced_uc_to_representative**(*coordgroups, basis, backends=['isotropy']*)

httk.atomistic.structureutils.**coordgroups_to_coords**(*coordgroups*)

httk.atomistic.structureutils.**coords_and_counts_to_coordgroups**(*coords*, *counts*)

httk.atomistic.structureutils.**coords_and_occupancies_to_coordgroups_and_assignments**(*coords*, *occupancies*)

httk.atomistic.structureutils.**coords_to_coordgroups**(*coords*, *counts*)

httk.atomistic.structureutils.**coordswap**(*fromidx*, *toidx*, *cell*, *coordgroups*)

httk.atomistic.structureutils.**get_primitive_basis_transform**(*hall_symbol*)
Transform to be applied to conventional unit cell to give the primitive unit cell

httk.atomistic.structureutils.**internal_coordgroups_reduced_rc_to_unitcellsites**(*coordgroups*, *basis*, *hall_symbol*, *eps=0.001*)

httk.atomistic.structureutils.**lengths_angles_to_niggli**(*lengths*, *angles*)

httk.atomistic.structureutils.**main**()

httk.atomistic.structureutils.**metric_to_niggli**(*cell*)

httk.atomistic.structureutils.**niggli_scale_to_vol**(*niggli_matrix*, *scale*)

httk.atomistic.structureutils.**niggli_to_basis**(*niggli_matrix*, *orientation=1*)

httk.atomistic.structureutils.**niggli_to_cell_old**(*niggli_matrix*, *orientation=1*)

httk.atomistic.structureutils.**niggli_to_lengths_angles**(*niggli_matrix*)

httk.atomistic.structureutils.**niggli_to_metric**(*niggli*)

httk.atomistic.structureutils.**niggli_vol_to_scale**(*niggli_matrix*, *vol*)

httk.atomistic.structureutils.**normalized_formula**(*assignments*, *ratios*, *counts*)

httk.atomistic.structureutils.**normalized_formula_parts**(*assignments*, *ratios*, *counts*)

httk.atomistic.structureutils.**occupations_and_coords_to_assignments_and_coordgroups**(*occupatio*
oc-
cu-
pa-
tions)

httk.atomistic.structureutils.**prototype_formula**(*proto*)

httk.atomistic.structureutils.**reduced_to_cartesian**(*cell*, *coordgroups*)

httk.atomistic.structureutils.**sort_coordgroups**(*coordgroups*, *individual_data*)

httk.atomistic.structureutils.**structure_reduced_uc_to_representative**(*struct,*
back-
ends=['isotropy',
'fake'])

httk.atomistic.structureutils.**structure_tidy**(*struct, backends=['platon']*)

httk.atomistic.structureutils.**structure_to_p1structure**(*struct, backends=['ase']*)

httk.atomistic.structureutils.**structure_to_sgstructure**(*struct,* *back-*
ends=['platon'])

httk.atomistic.structureutils.**transform**(*structure*, *transformation*, *max_search_cells=20*,
max_atoms=1000)

httk.atomistic.supercellutils module

class httk.atomistic.supercellutils.**StructureSupercellPlugin**

Bases: *httk.core.httkobject.HttkPlugin*

cubic(*tolerance=None*, *max_search_cells=1000*)

general(*transformation*, *max_search_cells=20*, *max_atoms=1000*)

orthogonal(*tolerance=None*, *max_search_cells=1000*)

plugin_init(*struct*)

httk.atomistic.supercellutils.**build_cubic_supercell**(*structure*, *tolerance=None*,
max_search_cells=1000)

httk.atomistic.supercellutils.**build_orthogonal_supercell**(*structure,* *tol-*
erance=None,
max_search_cells=1000,
ortho=[True, *True,*
True])

httk.atomistic.supercellutils.**build_supercell_old**(*structure*, *transformation*,
max_search_cells=1000)

httk.atomistic.supercellutils.**cubic_supercell_transformation**(*structure*, *tol-*
erance=None,
max_search_cells=1000)

httk.atomistic.supercellutils.**orthogonal_supercell_transformation**(*structure,*
toler-
ance=None,
or-
tho=[True,
True,
True])

httk.atomistic.unitcellsites module

class httk.atomistic.unitcellsites.**UnitcellSites**(*reduced_coordgroups=None*, *reduced_coords=None*, *counts=None*, *hall_symbol='P 1'*, *pbc=None*)

Bases: *httk.atomistic.sites.Sites*

Represents any collection of sites in a unitcell

total_number_of_atoms

httk.atomistic.unitcellsites.**main**()

httk.atomistic.unitcellstructure module

class httk.atomistic.unitcellstructure.**UnitcellStructure**(*assignments=None*, *uc_sites=None*, *uc_cell=None*)

Bases: *httk.core.httkobject.HttkObject*

A UnitcellStructure represents N sites of, e.g., atoms or ions, in any periodic or non-periodic arrangement. It keeps track of all the copies of the atoms within a unitcell.

The structure object is meant to be immutable and assumes that no internal variables are changed after its creation. All methods that 'changes' the object creates and returns a new, updated, structure object.

Naming conventions in httk.atomistic:

For cells:

cell = an abstract name for any reasonable representation of a 'cell' that defines the basis vectors used for representing the structure. When a 'cell' is returned, it is an object of type Cell

basis = a 3x3 sequence-type with (in rows) the three basis vectors (for a periodic system, defining the unit cell, and defines the unit of repetition for the periodic dimensions)

lengths_and_angles = (a,b,c,alpha,beta,gamma): the basis vector lengths and angles

niggli_matrix = ((v1*v1, v2*v2, v3*v3),(2*v2*v3, 2*v1*v3, 2*v2*v3)) where v1, v2, v3 are the vectors forming the basis

metric = ((v1*v1,v1*v2,v1*v3),(v2*v1,v2*v2,v2*v3),(v3*v1,v3*v2,v3*v3))

For sites:

These following prefixes are used to describe types of site specifications: representative cell/rc = only representative atoms are given, which are then to be repeated by structure symmetry group to give all sites

unit cell/uc = all atoms in unitcell

reduced = coordinates given in cell vectors

cartesian = coordinates given as direct cartesian coordinates

sites = used as an abstract name for any sensible representation of a list of coordinates and a cell, when a 'sites' is returned, it is an object of type Sites

counts = number of atoms of each type (one per entry in assignments)

coordgroups = coordinates represented as a 3-level-list of coordinates, e.g. [[[0,0,0],[0.5,0.5,0.5]],[[0.25,0.25,0.25]]] where level-1 list = groups: one group for each equivalent atom

counts and coords = one list with the number of atoms of each type (one per entry in assignments) and a 2-level list of coordinates.

For assignments of atoms, etc. to sites: assignments = abstract name for any representation of assignment of atoms. When returned, will be object of type Assignment.

atomic_numbers = a sequence of integers for the atomic number of each species

occupations = a sequence where the assignments are *repeated* for each coordinate as needed (prefixed with uc or rc depending on which coordinates)

For cell scaling: scaling = abstract name for any representation of cell scaling

scale = multiply all basis vectors with this number

volume = rescaling the cell such that it takes this volume

For periodicity: periodicity = abstract name of a representation of periodicity

pbc = 'periodic boundary conditions' = sequence of True and False for which basis vectors are periodic / non-periodic

nonperiodic_vecs = integer, number of basis vectors, counted from the first, which are non-periodic

For spacegroup: spacegroup = abstract name for any spacegroup representation. When returned, is of type Spacegroup.

hall_symbol = specifically the hall_symbol string representation of the spacegroup

classmethod create(*structure=None*, *uc_cell=None*, *uc_basis=None*, *uc_lengths=None*, *uc_angles=None*, *uc_niggli_matrix=None*, *uc_metric=None*, *uc_a=None*, *uc_b=None*, *uc_c=None*, *uc_alpha=None*, *uc_beta=None*, *uc_gamma=None*, *uc_sites=None*, *uc_reduced_coordgroups=None*, *uc_cartesian_coordgroups=None*, *uc_reduced_coords=None*, *uc_cartesian_coords=None*, *uc_reduced_occupationscoords=None*, *uc_cartesian_occupationscoords=None*, *uc_occupancies=None*, *uc_counts=None*, *uc_scale=None*, *uc_scaling=None*, *uc_volume=None*, *volume_per_atom=None*, *assignments=None*, *periodicity=None*, *nonperiodic_vecs=None*, *other_reps=None*, *refs=None*, *tags=None*)

A FullStructure represents N sites of, e.g., atoms or ions, in any periodic or non-periodic arrangement, where the positions of all cites are given (as opposed to a set of unique sites + symmetry operations).

This is a swiss-army-type constructor that allows several different ways to create a FullStructure object.

To create a new structure, three primary components are:

- cell: defines the basis vectors in which reduced coordinates are expressed, and the unit of repetition (*if* the structure has any periodicity - see the 'periodicity' parameter)
- assignments: a list of 'things' (atoms, ions, etc.) that goes on the sites in the structure
- sites: a sensible representation of location / coordinates of the sites.

Note: *uc_*-prefixes are consistently enforced for any quantity that would be different in a UniqueSitesStructure. This is to allow for painless change between the various structure-type objects without worrying about accidently using the wrong type of sites object.

Note: see help(Structure) for parameter naming conventions, i.e., what type of object is expected given a parameter name.

Input parameters:

- ONE OF: 'uc_cell'; 'uc_basis', 'uc_length_and_angles'; 'uc_niggli_matrix'; 'uc_metric'; all of: uc_a,uc_b,uc_c, uc_alpha, uc_beta, uc_gamma. (cell requires a Cell object or a very specific format, so unless you know what you are doing, use one of the others.)

- ONE OF: 'uc_assignments', 'uc_atomic_numbers', 'uc_occupations' (uc_assignments requires an Assignments object or a sequence.), uc_occupations repeats similar site assignments as needed
- ONE OF: 'uc_sites', 'uc_coords' (IF uc_occupations OR uc_counts are also given), or 'uc_B_C', where B=reduced or cartesian, C=coordgroups, coords, or occupationscoords

  Notes:

  - occupationscoords may differ from coords by *order*, since giving occupations as, e.g., ['H','O','H'] does not necessarily have the same order of the coordinates as the format of counts+coords as (2,1), ['H','O'].
  - uc_sites requires a Sites object or a python list on a very specific format, (so unless you know what you are doing, use one of the others.)
- **ONE OF: uc_scale, uc_volume, or volume_per_atom:** scale = multiply the basis vectors with this scaling factor, volume = the unit cell volume (overrides 'scale' if both are given) volume_per_atom = cell volume / number of atoms
- ONE OF periodicity or nonperiodic_vecs

formula_builder

pbc

supercell

transform(*matrix*, *max_search_cells=20*, *max_atoms=1000*)

uc_a

uc_alpha

uc_b

uc_basis

uc_beta

uc_c

uc_cartesian_coordgroups

uc_cartesian_coords

uc_cartesian_occupationscoords

uc_cell_orientation

uc_counts

uc_gamma

uc_lengths_and_angles

uc_reduced_coordgroups

uc_reduced_coords

uc_volume

uc_volume_per_atom

classmethod use(*other*)

httk.config package

Submodules

httk.config.config module

Read and setup httk configuration and versioning data.

httk_python_root is derived as the directory config.py is in + ..

config is a configparser.config object where:

- All assignments in a distdata.py file in httk_python_root are read into the section [general]
- Read httk.cfg in httk_python_root
- Using the latest definition of [general]/httk_root, read httk.cfg in that directory
- Read ~/.httk/config

In this config object, the section [general] is looked up for 'httk_root', which is exported as httk_root. If not present, 'root' is looked up in the section 'distdata'. If that is not present, the default of httk_python_root + ../.. is used.

If the file distdata.py in httk_python_root exists, the config object section [distdata] is looked up for version, version_date, and copyright_note, which are exported as httk_version, httk_version_date, httk_copyright_note. If this file does not exist, they identifiers are instead derived using the 'git' command. If that does not work, they are set to 'unknown', except for httk_copyright_note, which is set to a sensible default.

This python file has no dependencies except for the standard library (neither within httk or outside). It will always remain safe to import by itself, e.g.:

```
(cd src/httk/config; python -c "import sys, config; sys.stdout.write(config.httk_
↪version + '\n')")
```

Or:

```
python -c "import sys; here = path.abspath(path.dirname(__file__)); sys.path.insert(1,
↪ os.path.join(here,'src/httk/config')); import config; sys.stdout.write(config.httk_
↪version + '\n')"
```

class httk.config.config.**ExceptionlessConfig**(*config*)
Bases: object

httk.config.config.**determine_version_data**()

httk.config.config.**read_config**()

httk.core package

Subpackages

httk.core.vectors package

Submodules

httk.core.vectors.fracmath module

httk.core.vectors.fracmath.**any_to_fraction**(*arg*, *min_accuracy=Fraction(1, 10000)*)
min_accuracy: we always assume the accuracy is at least this good. i.e., with min_accuracy=1/10000, we take 0.33 to really mean 0.3300, because we assume people meaning 1/3 at least makes the effort to write 0.3333

httk.core.vectors.fracmath.**best_rational_in_interval**(*low*, *high*)

httk.core.vectors.fracmath.**frac_acos**(*x*, *degrees=False*, *prec=Fraction(1, 10000000000)*, *limit=True*)
Return the arccosine of x in radians.

httk.core.vectors.fracmath.**frac_acos_alt**(*x*, *degrees=False*, *prec=Fraction(1, 10000000000)*, *limit=True*)
Return the arc cosine (measured in radians) of Decimal x.

httk.core.vectors.fracmath.**frac_acos_old**(*x*, *degrees=False*, *prec=Fraction(1, 10000000000)*, *limit=True*)
Return the arc cosine (measured in radians) of Decimal x.

httk.core.vectors.fracmath.**frac_asin**(*x*, *degrees=False*, *prec=Fraction(1, 10000000000)*, *limit=True*)
Return the arc sine (measured in radians) of Decimal x.

httk.core.vectors.fracmath.**frac_atan**(*x*, *degrees=False*, *prec=Fraction(1, 10000000000)*, *limit=True*)
Return the arctangent of x in radians.

httk.core.vectors.fracmath.**frac_atan2**(*y*, *x*, *degrees=False*, *prec=Fraction(1, 10000000000)*, *limit=True*)
Return the arctangent of y/x in radians.

Unlike atan(y/x), the signs of both x and y are considered.

httk.core.vectors.fracmath.**frac_atan_old**(*x*, *degrees=False*, *prec=Fraction(1, 10000000000)*, *limit=True*)
Return the arctangent of x in radians.

httk.core.vectors.fracmath.**frac_cos**(*x*, *prec=Fraction(1, 10000000000)*, *limit=True*, *degrees=False*)

httk.core.vectors.fracmath.**frac_exp**(*x*, *prec=Fraction(1, 10000000000)*, *limit=True*)
Return e raised to the power of x.

httk.core.vectors.fracmath.**frac_exp_old**(*x*, *prec=Fraction(1, 10000000000)*, *limit=True*)
Return e raised to the power of x.

httk.core.vectors.fracmath.**frac_log**(*x*, *base=None*, *prec=Fraction(1, 10000000000)*, *limit=True*)
Return the logarithm of x to the given base.

If the base not specified, return the natural logarithm (base e) of x.

TODO: Fix: this fails for moderately large arguments.

httk.core.vectors.fracmath.**frac_log10**(*x*, *prec=Fraction(1, 10000000000)*, *limit=True*)
Return the base 10 logarithm of x.

httk.core.vectors.fracmath.**frac_log_old**(*x*, *base=None*, *prec=Fraction(1, 10000000000)*, *limit=True*)
Return the logarithm of x to the given base.

If the base not specified, return the natural logarithm (base e) of x.

httk.core.vectors.fracmath.**frac_pi**(*prec=Fraction(1, 10000000000)*, *limit=True*)
Compute Pi to the precision prec.

httk.core.vectors.fracmath.**frac_pi_old**(*prec=Fraction(1, 10000000000)*, *limit=True*)
Compute Pi to the precision prec.

httk.core.vectors.fracmath.**frac_sin**(*x*, *prec=Fraction(1, 10000000000)*, *limit=True*, *degrees=False*)

httk.core.vectors.fracmath.**frac_sin_old**(*x*, *prec=Fraction(1, 10000000000)*, *limit=True*, *degrees=False*)

httk.core.vectors.fracmath.**frac_sqrt**(*x*, *prec=Fraction(1, 10000000000)*, *limit=True*)

httk.core.vectors.fracmath.**frac_sqrt_old**(*x*, *prec=Fraction(1, 10000000000)*, *limit=True*)

httk.core.vectors.fracmath.**frac_tan**(*x*, *degrees=False*, *prec=Fraction(1, 10000000000)*, *limit=True*)
Return the tangent of x.

httk.core.vectors.fracmath.**fraction_from_continued_fraction**(*cf*)

httk.core.vectors.fracmath.**get_continued_fraction**(*p*, *q*)

httk.core.vectors.fracmath.**integer_sqrt**(*n*)

httk.core.vectors.fracmath.**is_string**(*arg*)

httk.core.vectors.fracmath.**main**()

httk.core.vectors.fracmath.**run_alot**(*func*, *name*, *mathfun*, *fsmall*, *fmid*, *flarge*, *interval_within_one=False*, *positive=False*, *skip_worst=False*)

httk.core.vectors.fracmath.**string_to_val_and_delta**(*arg*, *min_accuracy=Fraction(1, 10000)*)

httk.core.vectors.fracvector module

class httk.core.vectors.fracvector.**FracScalar**(*nom*, *denom*)
Bases: *httk.core.vectors.fracvector.FracVector*

Represents the fractional number nom/denom. This is a subclass of FracVector with the purpose of making it clear when a scalar fracvector is needed/used.

classmethod create(*nom*, *denom=None*, *simplify=True*)
Create a FracScalar.

FracScalar(something) something may be any object that can be used in the constructor of the Python Fraction class (also works with strings!).

class httk.core.vectors.fracvector.**FracVector**(*noms*, *denom=1*)
Bases: *httk.core.vectors.vector.Vector*

FracVector is a general *immutable* N-dimensional vector (tensor) class for performing linear algebra with fractional numbers.

A FracVector consists of a multidimensional tuple of integer nominators, and a single shared integer denominator.

Since FracVectors are immutable, every operation on a FracVector returns a new FracVector with the result of the operation. A created FracVector never changes. Hence, they are safe to use as keys in dictionaries, to use in sets, etc.

Note: most methods returns FracVector results that are not simplified (i.e., the FracVector returned does *not* have the smallest possible integer denominator). To return a FracVector with the smallest possible denominator, just call FracVector.simplify() at the last step.

T()
Returns the transpose, A^T.

acos(*prec=None*, *degrees=False*, *limit=False*)
Return a FracVector where every element is the arccos of the element in the source FracVector.

prec = precision (should be set as a fraction) limit = True requires the denominator to be smaller or equal to precision

argmax()
Return the index of the maximum element across all dimensions in the FracVector.

argmin()
Return the index of the minimum element across all dimensions in the FracVector.

asin(*prec=None*, *degrees=False*, *limit=False*)
Return a FracVector where every element is the arcsin of the element in the source FracVector.

prec = precision (should be set as a fraction) limit = True requires the denominator to be smaller or equal to precision

ceil()
Returns the integer that is equal to or just below the value stored in a scalar FracVector.

classmethod chain_vecs(*vecs*)
Optimized chaining of FracVectors.

vecs: a list (or tuple) of fracvectors.

Returns the same thing as FracVector.create(vecs,chain=True)

i.e., removes outermost dimension and chain the sub-sequences. If input=[[1 2 3],[4,5,6]], then FracVector.chain(input) -> [1,2,3,4,5,6]

but this method assumes all vectors share the same denominator (it raises an exception if this is not true)

cos(*prec=None*, *degrees=False*, *limit=False*)
Return a FracVector where every element is the cosine of the element in the source FracVector.

prec = precision (should be set as a fraction) limit = True requires the denominator to be smaller or equal to precision

classmethod create(*noms*, *denom=None*, *simplify=True*, *chain=False*, *min_accuracy=Fraction(1, 10000)*)
Create a FracVector from various types of sequences.

Simplest use:

```
FracVector.create(some_kind_of_sequence)
```

where 'some_kind_of_sequence' can be any nested list or tuple of objects that can be used in the constructor of the Python Fraction class (also works with strings!). If any object found while traveling the items has a .to_fractions() method, it will be called and is expected to return a fraction or list or tuple of fractions.

Optional parameters:

- Invocation with denominator: FracVector.create(nominators,denominator) nominators is any sequence, and denominator a common denominator to divide all nominators with
- simplify: boolean, return a FracVector with the smallest possible denominator.

- chain: boolean, remove outermost dimension and chain the sub-sequences. I.e., if input=[[1 2 3],[4,5,6]], then FracVector.create(input) -> [1,2,3,4,5,6]

Relevant: FracVector itself implements .to_fractions(), and hence, the same constructor allows stacking several FracVector objects like this:

```
vertical_fracvector = FracVector.create([[fracvector1],[fracvector2]])
horizontal_fracvector = FracVector.create([fracvector1,fracvector2],
↪chain=True)
```

- min_accuracy: set to a boolean to adjust the minimum accuracy assumed in string input. The default is 1/10000, i.e. 0.33 = 0.3300 = 33/100, whereas 0.3333 = 1/3. Set it to None to assume infinite accuracy, i.e., convert exactly whatever string is given (unless a standard deviation is given as a parenthesis after the string.)

classmethod create_cos(*data*, *degrees=False*, *limit=False*, *find_best_rational=True*, *prec=Fraction(1, 1000000)*)

Creating a FracVector as the cosine of the argument data. If data are composed by strings, the standard deviation of the numbers are taken into account, and the best possible fractional approximation to the cosines of the data are returned within the standard deviation.

This is not the same as FracVector.create(data).cos(), which creates the best possible fractional approximations of data and then takes cos on that.

classmethod create_exp(*data*, *prec=Fraction(1, 1000000)*, *limit=False*)

Creating a FracVector as the exponent of the argument data. If data are composed by strings, the standard deviation of the numbers are taken into account, and the best possible fractional approximation to the cosines of the data are returned within the standard deviation.

This is not the same as FracVector.create(data).exp(), which creates the best possible fractional approximations of data and then takes exp on that.

classmethod create_sin(*data*, *degrees=False*, *limit=False*, *prec=Fraction(1, 1000000)*)

Creating a FracVector as the sine of the argument data. If data are composed by strings, the standard deviation of the numbers are taken into account, and the best possible fractional approximation to the cosines of the data are returned within the standard deviation.

This is not the same as FracVector.create(data).sin(), which creates the best possible fractional approximations of data and then takes cos on that.

cross(*other*)

Returns the vector cross product of the 3-element 1D vector with the 3-element 1D vector 'other', i.e., A x B.

det()

Returns the determinant of the FracVector as a scalar FracVector.

dim

This property returns a tuple with the dimensionality of each dimension of the FracVector (the noms are assumed to be a nested list of rectangular shape).

dot(*other*)

Returns the vector dot product of the 1D vector with the 1D vector 'other', i.e., A . B or A cdot B. The same as A * B.T().

exp(*prec=None*, *limit=False*)

Return a FracVector where every element is the exponent of the element in the source FracVector.

prec = precision (should be set as a fraction) limit = True requires the denominator to be smaller or equal to precision

classmethod eye(*dims*)
Create a diagonal one-matrix with the given dimensions

flatten()
Returns a FracVector that has been flattened out to a single rowvector

floor()
Returns the integer that is equal to or just below the value stored in a scalar FracVector.

classmethod from_floats(*l*, *resolution=4294967296*)
Create a FracVector from a (nested) list or tuple of floats. You can convert a numpy array with this method if you use A.tolist()

resolution: the resolution used for interpreting the given floating point numbers. Default is 2^32.

classmethod from_tuple(*t*)
Return a FracVector created from the tuple representation: (denom, . . . noms. . .), returned by the to_tuple() method.

ged_prestacked(*other*)

ged_stackedinsert(*pos*, *other*)

get_append(*other*)

get_extend(*other*)

get_insert(*pos*, *other*)

get_prepend(*other*)

get_prextend(*other*)

get_stacked(*other*)

inv()
Returns the matrix inverse, A^-1

lengthsqr()
Returns the square of the length of the vector. The same as A * A.T()

limit_denominator(*max_denom=1000000000*)
Returns a FracVector of reduced resolution.

resolution: each element in the returned FracVector is the closest numerical approximation that can is allowed by a fraction with maximally this denominator. Note: since all elements must be put on a common denominator, the result may have a larger denominator than max_denom

max()
Return the maximum element across all dimensions in the FracVector. max(fracvector) works for a 1D vector.

metric_product(*vecA*, *vecB*)

Returns the result of the metric product using the present square FracVector as the metric matrix. The same as vecA*self*vecB.T().

min()
Return the minimum element across all dimensions in the FracVector. max(fracvector) works for a 1D vector.

mul(*other*)
Returns the result of multiplying the vector with 'other' using matrix multiplication.

Note that for two 1D FracVectors, A.dot(B) is *not* the same as A.mul(B), but rather: A.mul(B.T()).

nargmax()
Return a list of indices of all maximum elements across all dimensions in the FracVector.

nargmin()
Return a list of indices for all minimum elements across all dimensions in the FracVector.

static nested_map(*op*, **ls*)
Map an operator over a nested tuple. (i.e., the same as the built-in map(), but works recursively on a nested tuple)

static nested_map_fractions(*op*, **ls*)
Map an operator over a nested tuple, but checks every element for a method to_fractions() and uses this to further convert objects into tuples of Fraction.

nom
Returns the integer nominator of a scalar FracVector.

normalize()
Add/remove an integer +/-N to each element to place it in the range [0,1)

normalize_half()
Add/remove an integer +/-N to each element to place it in the range [-1/2,1/2)

This is useful to find the shortest vector C between two points A, B in a space with periodic boundary conditions [0,1
C = (A-B).normalize_half()

classmethod pi(*prec=Fraction(1, 1000000)*, *limit=False*)
Create a scalar FracVector with a rational approximation of pi to precision prec.

classmethod random(*dims*, *minnom=-100*, *maxnom=100*, *denom=100*)
Create a zero matrix with the given dimensions

reciprocal()

classmethod set_common_denom(*A*, *B*)
Used internally to combine two different FracVectors.

Returns a tuple (A2,B2,denom) where A2 is numerically equal to A, and B2 is numerically equal to B, but A2 and B2 are both set on the same shared denominator 'denom' which is the *product* of the denominator of A and B.

set_denominator(*set_denom=1000000000*)
Returns a FracVector of reduced resolution where every element is the closest numerical approximation using this denominator.

sign()
Returns the sign of the scalar FracVector: -1, 0 or 1.

simplify()
Returns a reduced FracVector. I.e., each element has the same numerical value but the new FracVector represents them using the smallest possible shared denominator.

sin(*prec=None*, *degrees=False*, *limit=False*)
Return a FracVector where every element is the sine of the element in the source FracVector.

prec = precision (should be set as a fraction) limit = True requires the denominator to be smaller or equal to precision

sqrt(*prec=None*, *limit=False*)
Return a FracVector where every element is the sqrt of the element in the source FracVector.

prec = precision (should be set as a fraction) limit = True requires the denominator to be smaller or equal to precision

classmethod stack_vecs(*vecs*)
Optimized stacking of FracVectors.

vecs = a list (or tuple) of fracvectors.

Returns the same thing as:

```
FracVector.create(vecs)
```

but only works if all vectors share the same denominator (raises an exception if this is not true)

to_float()
Converts a scalar ExactVector to a single float.

to_floats()
Converts the ExactVector to a list of floats.

to_fraction()
Converts scalar FracVector to a fraction.

to_fractions()
Converts the FracVector to a list of fractions.

to_int()
Converts scalar FracVector to an integer (truncating as necessary).

to_ints()
Converts the FracVector to a list of integers, rounded off as best possible.

to_string(*accuracy=8*)
Converts the ExactVector to a list of strings.

to_strings(*accuracy=8*)
Converts the ExactVector to a list of strings.

to_tuple()
Return a FracVector on tuple representation: (denom, ... noms...).

classmethod use(*old*)
Make sure variable is a FracVector, and if not, convert it.

validate()

classmethod zeros(*dims*)
Create a zero matrix with the given dimensions

httk.core.vectors.fracvector.**main**()

httk.core.vectors.fracvector.**nested_map_fractions_list**(*op*, **ls*)
Map an operator over a nested list, but checks every element for a method to_fractions() and uses this to further convert objects into lists of Fraction.

httk.core.vectors.fracvector.**nested_map_fractions_tuple**(*op*, **ls*)
Map an operator over a nested tuple, but checks every element for a method to_fractions() and uses this to further convert objects into tuples of Fraction.

httk.core.vectors.fracvector.**nested_map_list**(*op*, **ls*)
Map an operator over a nested list. (i.e., the same as the built-in map(), but works recursively on a nested list)

httk.core.vectors.fracvector.**nested_map_tuple**(*op*, **ls*)
Map an operator over a nested tuple. (i.e., the same as the built-in map(), but works recursively on a nested tuple)

httk.core.vectors.fracvector.**nested_reduce**(*op*, *l*, *initializer=None*)
Same as built-in reduce, but operates on a nested tuple/list/sequence.

httk.core.vectors.fracvector.**nested_reduce_fractions**(*op*, *l*, *initializer=None*)
Same as built-in reduce, but operates on a nested tuple/list/sequence. Also checks every element for a method to_fractions() and uses this to further convert such elements to lists of fractions.

httk.core.vectors.fracvector.**nested_reduce_levels**(*op*, *l*, *level=1*, *initializer=None*)
Same as built-in reduce, but operates on a nested tuple/list/sequence.

httk.core.vectors.fracvector.**tuple_eye**(*dims*, *onepos=0*)
Create a matrix with the given dimensions and 1 on the diagonal

httk.core.vectors.fracvector.**tuple_index**(*dims*, *uppidx=()*)
Create a nested tuple where every element is a tuple indicating the position of that tuple

httk.core.vectors.fracvector.**tuple_random**(*dims*, *minval*, *maxval*)
Create a nested tuple with the given dimensions filled with random numbers between minval and maxval

httk.core.vectors.fracvector.**tuple_slice**(*l*, *key*)
Given a python slice (i.e., what you get to __getitem__ when you write A[3:2]), cut out the relevant nested tuple.

httk.core.vectors.fracvector.**tuple_zeros**(*dims*)
Create a netsted tuple with the given dimensions filled with zeroes

httk.core.vectors.mutablefracvector module

class httk.core.vectors.mutablefracvector.**MutableFracVector**(*noms*, *denom*)
Bases: *httk.core.vectors.fracvector.FracVector*, *httk.core.vectors.vector.MutableVector*

Same as FracVector, only, this version allow assignment of elements, e.g.,

```
mfracvec[2,7] = 5
```

and, e.g.,

```
mfracvec[:,7] = [1,2,3,4]
```

Other than this, the FracVector methods exist and do the same, i.e., they return *copies* of the fracvector, rather than modifying it.

However, methods have also been added named with set_* prefixes which performs mutating operations, e.g.,

```
A.set_T()
```

replaces A with its own transpose, whereas

```
A.T()
```

just returns a new MutableFracVector that is the transpose of A, leaving A unmodified.

classmethod from_FracVector(*other*)
Create a MutableFracVector from a FracVector.

invalidate()
Internal method to call when MutableFracVector is changed in such a way that cached properties are invalidated (e.g., _dim)

static nested_inmap(*op*, **ls*)
Like inmap, but work for nested lists

static nested_map(*op*, **ls*)
Map an operator over a nested list. (i.e., the same as the built-in map(), but works recursively on a nested list)

static nested_map_fractions(*op*, **ls*)
Map an operator over a nested list, but checks every element for a method to_fractions() and uses this to further convert objects into lists of Fraction.

set_T()
Changes MutableFracVector inline into own transpose: self -> self.T

set_inv()
Changes MutableFracVector inline into own inverse: self -> self^-1

set_negative()
Changes MutableFracVector inline into own negative: self -> -self

set_normalize()
Add/remove an integer +/-N to each element to place it in the range [0,1)

set_normalize_half()
Add/remove an integer +/-N to each element to place it in the range [-1/2,1/2)

This is useful to find the shortest vector C between two points A, B in a space with periodic boundary conditions [0,1
C = (A-B).normalize_half()

set_set_denominator(*resolution=1000000000*)
Changes MutableFracVector; reduces resolution.

resolution is the new denominator, each element becomes the closest numerical approximation using this denominator.

set_simplify()
Changes MutableFracVector; reduces any common factor between denominator and all nominators

to_FracVector()
Return a FracVector with the values of this MutableFracVector.

classmethod use(*old*)
Make sure variable is a MutableFracVector, and if not, convert it.

validate()

httk.core.vectors.mutablefracvector.**inmap**(*f*, *x*)
Like built-in map, but work on a list and *replace* the elements in the list with the result of the mapping.

httk.core.vectors.mutablefracvector.**list_set_slice**(*l*, *key*, *values*)

Given: l = list, key = python slice (i.e., what you get to __setitem__ when you write A[3:2]=[2,5]) values = a list of values,

change the elements specified by the slice in key to those given by values.

httk.core.vectors.mutablefracvector.**list_slice**(*l*, *key*)
Given a python slice (i.e., what you get to __getitem__ when you write A[3:2]), cut out the relevant nested list.

httk.core.vectors.mutablefracvector.**main**()

httk.core.vectors.mutablefracvector.**nested_inmap_list**(*op*, **ls*)
Like inmap, but work for nested lists

httk.core.vectors.vector module

class httk.core.vectors.vector.**MutableVector**
Bases: object

class httk.core.vectors.vector.**Scalar**
Bases: *httk.core.vectors.vector.Vector*

Baseclass for scalars

class httk.core.vectors.vector.**Vector**
Bases: object

Defines the general Vector API

classmethod chain_vecs(*vecs*)
Optimized chaining of Vectors.

vecs: a list (or tuple) of vectors.

Returns the same thing as Vector.create(vecs, chain=True)

i.e., removes outermost dimension and chain the sub-sequences. If input=[[1 2 3],[4,5,6]], then Vector.chain(input) -> [1,2,3,4,5,6]

Subclasses may add requirements on the vectors to use this method over <subclass>.create

classmethod create(*data*, *chain=False*)
Create a Vector from various types of sequenced data.

Will return a suitable Vector subclass for the type of data given

classmethod eye(*dims*)
Create a diagonal one-matrix with the given dimensions

ged_prestacked(*other*)

ged_stackedinsert(*pos*, *other*)

get_append(*other*)

get_extend(*other*)

get_insert(*pos*, *other*)

get_prepend(*other*)

get_prextend(*other*)

get_stacked(*other*)

classmethod random(*dims*, *minval=-100*, *maxval=100*)
Create a zero matrix with the given dimensions

classmethod stack_vecs(*vecs*)
Optimized stacking of FracVectors.

vecs = a list (or tuple) of fracvectors.

Returns the same thing as:

```
Vector.create(vecs)
```

Subclasses may add requirements on the vectors to use this method over <subclass>.create

classmethod use(*old*)
Make sure variable is a FracVector, and if not, convert it.

classmethod zeros(*dims*)
Create a zero matrix with the given dimensions

httk.core.vectors.vector.**main**()

httk.core.vectors.vector.**nested_map_fractions_list**(*op*, **ls*)
Map an operator over a nested list, but checks every element for a method to_fractions() and uses this to further convert objects into lists of Fraction.

httk.core.vectors.vector.**nested_map_list**(*op*, **ls*)
Map an operator over a nested list. (i.e., the same as the built-in map(), but works recursively on a nested list)

httk.core.vectors.vector.**nested_reduce**(*op*, *l*, *initializer=None*)
Same as built-in reduce, but operates on a nested tuple/list/sequence.

httk.core.vectors.vector.**nested_reduce_fractions**(*op*, *l*, *initializer=None*)
Same as built-in reduce, but operates on a nested tuple/list/sequence. Also checks every element for a method to_fractions() and uses this to further convert such elements to lists of fractions.

httk.core.vectors.vector.**nested_reduce_levels**(*op*, *l*, *level=1*, *initializer=None*)
Same as built-in reduce, but operates on a nested tuple/list/sequence.

httk.core.vectors.vector.**tuple_eye**(*dims*, *onepos=0*)
Create a matrix with the given dimensions and 1 on the diagonal

httk.core.vectors.vector.**tuple_index**(*dims*, *uppidx=()*)
Create a nested tuple where every element is a tuple indicating the position of that tuple

httk.core.vectors.vector.**tuple_random**(*dims*, *minval*, *maxval*)
Create a nested tuple with the given dimensions filled with random numbers between minval and maxval

httk.core.vectors.vector.**tuple_slice**(*l*, *key*)
Given a python slice (i.e., what you get to __getitem__ when you write A[3:2]), cut out the relevant nested tuple.

httk.core.vectors.vector.**tuple_zeros**(*dims*)
Create a netsted tuple with the given dimensions filled with zeroes

httk.core.vectors.vectormath module

httk.core.vectors.vectormath.**acos**(*x*, ***args*)
Return the arc cosine of x, in radians.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**acosh**(*x*, ***args*)
Return the inverse hyperbolic cosine of x.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**asin**(*x*, ***args*)
Return the arc sine of x, in radians.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**asinh**(*x*, ***args*)
Return the inverse hyperbolic sine of x.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**atan**(*x*, ***args*)
Return the arc tangent of x, in radians.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**atan2**(*x*, *y*, ***args*)
Return atan(y / x), in radians. The result is between -pi and pi. The vector in the plane from the origin to point (x, y) makes this angle with the positive X axis. The point of atan2() is that the signs of both inputs are known to it, so it can compute the correct quadrant for the angle. For example, atan(1) and atan2(1, 1) are both pi/4, but atan2(-1, -1) is -3*pi/4.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**atanh**(*x*, ***args*)
Return the inverse hyperbolic tangent of x.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**ceil**(*x*, ***args*)
Return the ceiling of x, the smallest integer value greater than or equal to x.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**copysign**(*x*, *y*, ***args*)
Return x with the sign of y. If an element of y is zero, abs of the corresponding element in x is returned.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**cos**(*x*, ***args*)
Return the cosine of x radians.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**cosh**(*x*, ***args*)
Return the hyperbolic cosine of x.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**degrees**(*x*, ***args*)
Convert angle x from radians to degrees.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**e**(*x*, ***args*)
Return the value of e represented using the same scalar or vector representation as x.

httk.core.vectors.vectormath.**erf**(*x*, ***args*)
Return the error function at x.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**erfc**(*x*, ***args*)
Return the complementary error function at x.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**exp**(*x*, ***args*)
Return e**x. (For vectors applied to each element.)

httk.core.vectors.vectormath.**expm1**(*x*, ***args*)
Return e**x - 1. (For vectors applied to each element.)

httk.core.vectors.vectormath.**fabs**(*x*, ***args*)
Return the absolute value of x.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**factorial**(*x*, ***args*)
Return x factorial. Raises ValueError if (any element of) x is negative.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**floor**(*x*, ***args*)
Return the floor of x, the largest integer value less than or equal to x.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**fmod**(*x*, *y*, ***args*)
Equivalent to x % y.

httk.core.vectors.vectormath.**frexp**(*x*, ***args*)
Return the mantissa and exponent of x as the pair (m, e). m is a float and e is an integer such that x == m * 2**e exactly. If x is zero, returns (0.0, 0), otherwise 0.5 <= abs(m) < 1.

(For vectors applied to each element and returns tuples nested in lists.)

httk.core.vectors.vectormath.**fsum**(*iterable*, ***args*)
Equivalent to sum(iterable)

httk.core.vectors.vectormath.**gamma**(*x*, ***args*)
Return the Gamma function at x.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**hypot**(*x*, *y*, ***args*)
Return the Euclidean norm, sqrt(x*x + y*y). This is the length of the vector from the origin to point (x, y).

(For vectors applied to each element.)

httk.core.vectors.vectormath.**isanyinf**(*x*, ***args*)
Check if the float x is positive or negative infinity.

(For vectors returns True/False if any element is inf)

httk.core.vectors.vectormath.**isanynan**(*x*, ***args*)
Check if the float x is a NaN (not a number).

(For vectors returns True/False if any element is NaN)

httk.core.vectors.vectormath.**isinf**(*x*, ***args*)
Check if the float x is positive or negative infinity.

(For vectors applied to each element and returns True/False as nested lists.)

httk.core.vectors.vectormath.**isnan**(*x*, ***args*)
Check if the float x is a NaN (not a number).

(For vectors applied to each element and returns True/False as nested lists.)

httk.core.vectors.vectormath.**ldexp**(*x*, ***args*)
Return x * (2**i). This is essentially the inverse of function frexp().

(For vectors applied to each element.)

httk.core.vectors.vectormath.**lgamma**(*x*, ***args*)
Return the natural logarithm of the absolute value of the Gamma function at x.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**log**(*x*, *base=None*, ***args*)
With one argument, return the natural logarithm of x (to base e).

With two arguments, return the logarithm of x to the given base, calculated as log(x)/log(base).

(For vectors applied to each element.)

httk.core.vectors.vectormath.**log10**(*x*, ***args*)
Return the base-10 logarithm of x. This is usually more accurate than log(x, 10).

(For vectors applied to each element.)

httk.core.vectors.vectormath.**log1p**(*x*, ***args*)
Return the natural logarithm of 1+x (base e). The result is calculated in a way which is accurate for x near zero.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**main**()

httk.core.vectors.vectormath.**modf**(*x*, ***args*)
Return the fractional and integer parts of x. Both results carry the sign of x.

(For vectors applied to each element and returns tuples nested in lists.)

httk.core.vectors.vectormath.**pi**(*x*, ***args*)
Return the value of pi represented using the same scalar or vector representation as x.

httk.core.vectors.vectormath.**pow**(*x*, *y*, ***args*)
Return x raised to the power y. Equivalent with x**y

(For vectors applied to each element.)

httk.core.vectors.vectormath.**radians**(*x*, ***args*)
Convert angle x from degrees to radians.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**sign**(*x*, ***args*)
Return the sign of x, equivalent to copysign(1,x).

(For vectors applied to each element.)

httk.core.vectors.vectormath.**sin**(*x*, ***args*)
Return the sine of x radians.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**sinh**(*x*, ***args*)
Return the hyperbolic sine of x.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**sqrt**(*x*, ***args*)
Return the square root of x.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**tan**(*x*, ***args*)
Return the tangent of x radians.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**tanh**(*x*, ***args*)
Return the hyperbolic tangent of x.

(For vectors applied to each element.)

httk.core.vectors.vectormath.**trunc**(*x*, ***args*)
Returns the integer part of x.

(For vectors applied to each element.)

Submodules

httk.core.basic module

Basic help functions

httk.core.basic.**anonymous_symbol_to_int**(*symb*)

httk.core.basic.**breath_first_idxs**(*dim=1*, *start=None*, *end=None*, *perm=True*, *negative=False*)

httk.core.basic.**create_tmpdir**()

httk.core.basic.**destroy_tmpdir**(*tmpdir*)

httk.core.basic.**flatten**(*l*)

httk.core.basic.**int_to_anonymous_symbol**(*i*)

httk.core.basic.**is_unary**(*e*)

httk.core.basic.**main**()

httk.core.basic.**micro_pyawk**(*ioa*, *search*, *results=None*, *debug=False*, *debugfunc=None*, *postdebugfunc=None*)

Small awk-mimicking search routine.

'f' is stream object to search through. 'search' is the "search program", a list of lists/tuples with 3 elements; i.e., [[regex,test,run],[regex,test,run],...] 'results' is a an object that your search program will have access to for storing results.

Here regex is either as a Regex object, or a string that we compile into a Regex. test and run are callable objects.

This function goes through each line in filename, and if regex matches that line *and* test(results,line)==True (or test == None) we execute run(results,match), where match is the match object from running Regex.match.

The default results is an empty dictionary. Passing a results object let you interact with it in run() and test(). Hence, in many occasions it is thus clever to use results=self.

Returns: results

httk.core.basic.**mkdir_p**(*path*)

httk.core.basic.**nested_split**(*s*, *start*, *stop*)

httk.core.basic.**parse_parexpr**(*string*)

Generate parenthesized contents in string as pairs (level, contents).

class httk.core.basic.**rewindable_iterator**(*iterator*)

Bases: object

next()

rewind(*rewindstr=None*)

httk.core.basic.**tuple_to_str**(*t*)

httk.core.citation module

Keep track of citation information for different parts of httk, so that this info can be printed out on program exit. Turn on either explicitly by calling httk.config.print_citations_at_exit() from your program, or implicitly for all software using httk by setting 'auto_print_citations_at_exit=yes' in httk.cfg

Right now this is mostly a proof of concept code, and was added in response to a concern that co-authors of the software would not get credit. We should extend this to add a facility to make it easier to track and acknowledge citations also of the data being used.

httk.core.citation.**add_ext_citation**(*software*, *author*)

httk.core.citation.**add_src_citation**(*module*, *author*)

httk.core.citation.**dont_print_citations_at_exit**()

httk.core.citation.**print_citations**()

httk.core.citation.**print_citations_at_exit**()

httk.core.code module

class httk.core.code.**Code**(*name*, *version*)
Bases: *httk.core.httkobject.HttkObject*

Object for keeping track of httk data about a computer software or script

add_ref(*ref*)

add_refs(*refs*)

add_tag(*tag*, *val*)

add_tags(*tags*)

classmethod create(*name*, *version*, *refs=None*, *tags=None*)
Create a Computation object.

get_refs()

get_tag(*tag*)

get_tags()

class httk.core.code.**CodeRef**(*code*, *reference*)
Bases: *httk.core.httkobject.HttkObject*

class httk.core.code.**CodeTag**(*structure*, *tag*, *value*)
Bases: *httk.core.httkobject.HttkObject*

httk.core.code.**main**()

httk.core.computation module

class httk.core.computation.**Computation**(*computation_date*, *description*, *code*, *manifest_hash*, *signatures*, *keys*, *relpath*, *project_counter*, *added_date=None*)
Bases: *httk.core.httkobject.HttkObject*

Object for keeping track of httk data about a specific computation run

add_project(*project*)

add_projects(*projects*)

add_ref(*ref*)

add_refs(*refs*)

add_tag(*tag*, *val*)

add_tags(*tags*)

added_date

classmethod create(*computation_date*, *description*, *code*, *manifest_hash*, *signatures*, *keys*, *project_counter*, *relpath*, *added_date=None*)
Create a Computation object.

get_projects()

get_refs()

get_tag(*tag*)

get_tags()

class httk.core.computation.**ComputationProject**(*computation*, *project*)
Bases: *httk.core.httkobject.HttkObject*

classmethod create(*computation*, *project*)
Create a Computation object.

class httk.core.computation.**ComputationRef**(*computation*, *reference*)
Bases: *httk.core.httkobject.HttkObject*

class httk.core.computation.**ComputationRelated**(*main_computation*, *other_computation*, *relation*)
Bases: *httk.core.httkobject.HttkObject*

Object for keeping track of httk data about a specific computation run

classmethod create(*main_computation*, *other_computation*, *relation*)
Create a Computation object.

class httk.core.computation.**ComputationTag**(*computation*, *tag*, *value*)
Bases: *httk.core.httkobject.HttkObject*

class httk.core.computation.**Result**(*computation*)
Bases: *httk.core.httkobject.HttkObject*

Intended as a base class for results tables for computations

classmethod create(*computation*)
Create a Computation object.

httk.core.computation.**main**()

httk.core.console module

httk.core.console.**cerr**(**args*)

httk.core.console.**cout**(**args*)

httk.core.crypto module

Provides a few central and very helpful functions for cryptographic hashes, etc.

httk.core.crypto.**generate_keys**(*public_key_path*, *secret_key_path*)
Generates a public and a private key pair and stores them in respective files

httk.core.crypto.**get_crypto_signature**(*message*, *secret_key=None*, *keyfile=None*)

httk.core.crypto.**hexhash_ioa**(*ioa*, *prepend=None*)

httk.core.crypto.**hexhash_str**(*data*, *prepend=None*)

httk.core.crypto.**main**()

httk.core.crypto.**manifest_dir**(*basedir*, *manifestfile*, *excludespath*, *keydir*, *sk*, *pk*, *debug=False*, *force=False*)

httk.core.crypto.**read_keys**(*keydir*)

httk.core.crypto.**sha256file**(*filename*)

httk.core.crypto.**tuple_to_hexhash**(*t*)

httk.core.crypto.**tuple_to_str**(*t*)

httk.core.crypto.**verify_crytpo_signature**(*signature*, *message*, *public_key=None*, *keyfile=None*)

httk.core.crypto.**verify_crytpo_signature_old**(*signature*, *message*, *public_key_path*)

httk.core.ed25519 module

httk.core.ed25519.**H**(*m*)

httk.core.ed25519.**Hint**(*m*)

httk.core.ed25519.**bit**(*h*, *i*)

httk.core.ed25519.**checkvalid**(*s*, *m*, *pk*)

httk.core.ed25519.**decodeint**(*s*)

httk.core.ed25519.**decodepoint**(*s*)

httk.core.ed25519.**edwards**(*P*, *Q*)

httk.core.ed25519.**encodeint**(*y*)

httk.core.ed25519.**encodepoint**(*P*)

httk.core.ed25519.**expmod**(*b*, *e*, *m*)

httk.core.ed25519.**inv**(*x*)

httk.core.ed25519.**isoncurve**(*P*)

httk.core.ed25519.**main**()

httk.core.ed25519.**publickey**(*sk*)

httk.core.ed25519.**scalarmult**(*P*, *e*)

httk.core.ed25519.**signature**(*m*, *sk*, *pk*)

httk.core.ed25519.**xrecover**(*y*)

httk.core.geometry module

Basic geometry helper functions

httk.core.geometry.**hull_z**(*points*, *zs*)

points: a list of points=(x,y,..) with zs= a list of z values; a convex half-hull is constructed over negative z-values

returns data on the following format.:

```
{
  'hull_points': indices in points list for points that make up the convex hull,
   'interior_points':indices for points in the interior,
   'interior_zs':interior_zs
   'zs_on_hull': hull z values for each point (for points on the hull, the value␣
↪of the hull if this point is excluded)
   'closest_points': list of best linear combination of other points for each␣
↪point
   'closest_weights': weights of best linear combination of other points for each␣
↪point
}
```

where hull_points and interior_points are lists of the points on the hull and inside the hull. and

hull_zs is a list of z-values that the hull *would have* at that point, had this point not been included. interior_zs is a list of z-values that the hull has at the interior points.

httk.core.geometry.**is_any_part_of_cube_inside_cell**(*cell*, *midpoint*, *side*)
Checks if any part of a cube is inside the cell spanned by the vectors in cell

httk.core.geometry.**is_point_inside_cell**(*cell*, *point*)
Checks if a given triple-vector is inside the cell given by the basis matrix in cell

httk.core.geometry.**is_point_inside_tetra**(*tetra*, *point*)
Checks if a point is inside the tretrahedra spanned by the coordinates in tetra

httk.core.geometry.**numpy_quickhull_2d**(*sample*)

httk.core.geometry.**simplex_le_solver**(*a*, *b*, *c*)
Minimizie func = a[0]*x + a[1]*y + a[2]*z + ... With constraints:

```
b[0,0]x + b[0,1]y + b[0,2]z + ... <= c[0]
b[1,0]x + b[1,1]y + b[1,2]z + ... <= c[1]
...
x,y,z, ... >= 0
```

Algorithm adapted from 'taw9', http://taw9.hubpages.com/hub/Simplex-Algorithm-in-Python

httk.core.httkobject module

class httk.core.httkobject.**HttkObject**
Bases: object

get_codependent_data()

hexhash

classmethod new_from(*other*)

to(*newtype*)

to_tuple(*use_hexhash=False*)

classmethod types()

classmethod use(*old*)

class httk.core.httkobject.**HttkPlugin**
Bases: object

class httk.core.httkobject.**HttkPluginPlaceholder**(*plugininfo=None*)
Bases: object

class httk.core.httkobject.**HttkPluginWrapper**(*plugin=None*)
Bases: object

class httk.core.httkobject.**HttkTypedProperty**(*property_type*, *fget=None*, *fset=None*, *fdel=None*, *doc=None*)
Bases: property

httk.core.httkobject.**httk_typed_init**(*t*, ***kargs*)

httk.core.httkobject.**httk_typed_init_delayed**(*t*, ***kargs*)

httk.core.httkobject.**httk_typed_property**(*t*)

httk.core.httkobject.**httk_typed_property_delayed**(*t*)

httk.core.httkobject.**httk_typed_property_resolve**(*cls*, *propname*)

httk.core.ioadapters module

class httk.core.ioadapters.**IoAdapterFileAppender**(*f*, *name=None*)
Bases: object

Io adapter for access to data as a python file object

close()

classmethod use(*other*)

class httk.core.ioadapters.**IoAdapterFileReader**(*f*, *name=None*, *deletefilename=None*, *close=False*)
Bases: object

Io adapter for easy handling of io.

close()

classmethod use(*other*)

class httk.core.ioadapters.**IoAdapterFileWriter**(*f*, *name=None*, *close=False*)
Bases: object

Io adapter for access to data as a python file object

close()

classmethod use(*other*)

class httk.core.ioadapters.**IoAdapterFilename**(*filename*, *name=None*, *deletefilename=None*)
Bases: object

Universal io adapter, helps handling the passing of filenames, files, and strings to functions that deal with io

close()

classmethod use(*other*)

class httk.core.ioadapters.**IoAdapterString**(*string=None*, *name=None*)
Bases: object

Universal io adapter, helps handling the passing of filenames, files, and strings to functions that deal with io

close()

string

classmethod use(*other*)

class httk.core.ioadapters.**IoAdapterStringList**(*stringlist*, *name=None*)
Bases: object

Universal io adapter, helps handling the passing of filenames, files, and strings to functions that deal with io

classmethod use(*other*)

httk.core.ioadapters.**cleveropen**(*filename*, *mode*, **args*)

httk.core.ioadapters.**main**()

httk.core.ioadapters.**universal_opener**(*other*)

httk.core.ioadapters.**zdecompressor**(*f*, *mode*, **args*)
Read a classic unix compress .Z type file.

httk.core.miniparser module

LR(1) miniparser

Introduction

A relatively bare-bones LR(1) parser that parses strings into abstract syntax trees (ast) for generic languages. Python 2 and 3 compatible. Language grammars can be given in textual EBNF.

A simple usage example:

```
from miniparser import parser

ls = {
  'ebnf_grammar': """
     S = E ;
     E = T, '+', E ;
     E = T ;
     T = id ;
  """,
  'tokens': {'id': '[a-zA-Z][a-zA-Z0-9_]*'}
}

input_string = "Test + Test"

result = parser(ls, input_string)
print(result)
```

Usage example of a simple grammar for balanced parentheses. This also shows using inline regex via an EBNF special sequence:

```
from miniparser import parser
ls = {
  'ebnf_grammar': """
     Expr = Group
          | Expr , Expr
          | id ;
     Group = '(', Expr, ')' ;
```

(continues on next page)

(continued from previous page)

```
    id = ? [a-zA-Z0-9 _]+ ? ;
  """,
  'remove': ['(',')'],
  'simplify': ['Expr']
}

input_string = "Outer ( Inner ( Inside ) Further outside )"

result = parser(ls, input_string)
print(result)
```

Note: in the above examples, the parse tables are generated on the first call to parse, and then cached inside the 'ls' dict. However, if one wants to pre-generate the parse tables (e.g., for looking at them), that can be done by calling build_ls(ls=ls) before parse. You can, if you want, save the 'ls' variable to disk (e.g. using pickle). However, since a modern computer builds the parse tables in a time comparable with starting up the python interpreter, this may not be so useful.

For documentation on the parameters in the ls dict, see help(build_ls).

Detailed description

This is roughly how the parser operates:

1. It takes as input:

   1.1. An EBNF grammar in text format for the language it is supposed to parse: *ebnf_grammar*.

   1.2. Some other meta-info about the language that defines, e.g., terminals (elements that are not further simplified), etc.

   1.3. A string to parse.

2. The fist time this langague is parsed, the parser builds up the necessary data structures for the language using the function *build_ls*. The steps are:

   2.1. The parser uses itself to parse *ebnf_grammar* into an ast representation of the grammar: *ebnf_grammar_ast*.

   To do this, it uses an already provided ast of the EBNF language itself (but which can also be recreated by the parser itself as shown in the examples at the end of the file under __name__ == "__main__".)

   2.2. The *ebnf_grammar_ast* is translated to a more BNF-like abstract form that expands alteration, optionals, groupings, and repetitions into separate rules: *bnf_grammar_ast*.

   2.3. The *bnf_grammar_ast* is processed into a *rule_table*. This is a dictionary that maps every symbol to a list of possible right hand sides in the production rules.

   2.4. The *rule_table* is used to build a table of the FIRST(symbol) function in LR parsing. It maps all symbols on a list of terminals that may be the very first thing seen in the input when matching that production rule: *first_table*.

   2.5. The *rule_table'and the 'first_table* are used to build the ACTION and GOTO tables in LR parsing. These encode a state machine that for every starting state S tells the machine to either shift or reduce, and when doing so, the state the machine progresses to: *action_table* and *goto_table*.

3. The parse string is processed the python generator *lexer*, which splits the input into lexical tokens.

4. The LR state machine is initialized in its starting state. Tokens are read from the lexer, and shift/reduce actions and state changes are made according to *action_table* and *goto_table*. The results of the parsing are collected on the symbol stack in the from of an ast.

5. When all input has been reduced into the starting symbol, the ast connected to that symbol is returned.

Diagnostic output

- You can add verbosity=<int> as an argument to both the *parser* and the *build_ls* function to get that level of diagnostic output.
- For more fine-tuned output, set verbosity = LogVerbosity(verbosity, [<flags>]) flags can be various flags that can be found in the source code.

  Known flags at the time of writing:

  - *print_all_tokens=True* lets makes the parser have the lexer process all input first and prints all tokens before the parsing starts.
  - <*function name*>*_verbosity* = <*verbosity level*> adjusts the verbosity level for just that one function. For example:

    parser(ls, source, verbosity=LogVerbosity(0,parser_verbosity=3))

  prints out diagnostic output on level 3 for the parser function, but skips any other diagnostic output.
- If you do not want the default behavior of printing diagnostic output on stdout, both parser and build_ls takes the argument logger=<function>, which redirects all diagnostic output to that function. The function should have the signature:

```
logger(*args,**kargs):
```

  where the args is the diagnostic info being printed, and the keyword arguments communicates flags. In particular, pretty=True indicates that complex objects are passed which would benefit from using, e.g., pprint.pprint to typeset the output.

class httk.core.miniparser.**LogVerbosity**(*verbosity*, ***flags*)
Bases: object

Class to send in as keyword argument for verbosity to fine-tune diagnostic output from certain functions.

Set the keyword argument as follows:

```
verbosity = LogVerbosity(verbosity, [<flags>])
```

flags can be various flags that can be found in the source code, e.g., *print_all_tokens=True* lets makes the parser have the lexer process all input first and prints all tokens before the parsing starts.

Specifically, set <*function name*>*_verbosity* = <*verbosity level*> to adjust the verbosity level for just that one function. For example:

```
parser(ls, source, verbosity=LogVerbosity(0,parser_verbosity=3))
```

prints out diagnostic output on level 3 for the parser function, but skips any other diagnostic output.

exception httk.core.miniparser.**ParserError**
Bases: exceptions.Exception

exception httk.core.miniparser.**ParserGrammarError**
Bases: *httk.core.miniparser.ParserError*

exception httk.core.miniparser.**ParserInternalError**
Bases: *httk.core.miniparser.ParserError*

exception httk.core.miniparser.**ParserSyntaxError**(**args*)
Bases: *httk.core.miniparser.ParserError*

httk.core.miniparser.**build_ls**(*ebnf_grammar=None*, *tokens={}*, *partial_tokens={}*, *literals=None*, *precedence=[]*, *ignore=' \t\n'*, *simplify=[]*, *aggregate=[]*, *start=None*, *skip=[]*, *remove=[]*, *comment_markers=[]*, *ls=None*, *verbosity=0*, *logger=<function logger>*)
Build a language specification from an ebnf grammar and some meta-info of the language.

Args:

ebnf_grammar (str): a string containing the ebnf describing the language.

tokens (dict,optional): a dict of token names and the regexs that defines them, they are considered terminals in the parsing. (They may also be defined as production rules in the ebnf, but if so, those definitions are ignored.)

partial_tokens (dict): a dictionary that maps token names on regular expressions for partial token matches. This is used to allow finding longer matches if there is intermediate length input that does not match. E.g., to match 5.32e6 as a number instead as as Number(5.32) + Identifier(e) + Number(6).

literals (list of str): a list of strings of 1 or more characters which define literal symbols of the language (i.e, the tokenizer name the tokens the same as the string), if not given, an attemt is made to auto-extract them from the grammar.

precedence (list,optional): list of tuples of the format (associativity, symbol, ...), the order of this list defines the precedence of those symbols, later in the list = higher precedence. The associativity can be 'left', 'right', or 'noassoc'.

ignore (str,optional): a string of characters, or a list of strings for symbols, which are withheld by the tokenizer. (This is commonly used to skip emitting whitespace tokens, while still supprting whitespace inside tokens, e.g., quoted strings.)

simplify (list,optional): a list of symbol identifiers that are simplified away when the parse tree is generated.

aggregate (list,optional): a list of symbol identifiers that when consituting consequtive nodes are 'flattened', removing the ambiguity of left or right associativity.

start (str,optional): the start (topmost) symbol of the grammar. A successful parsing means reducing all input into this symbol.

remove (list): list of symbols to just skip in the output parse tree (useful to, e.g., skip uninteresting literals).

skip (list): list of rules to completely ignore in the grammar. (useful to skip rules in a complete EBNF which reduces the tokens into single characters entities, when one rather wants to handle those tokens by regex:es by passing the token argument)

ls (dict): As an alternative to giving the above parameters, a dict can be given with the same attributes as the arguments defined above.

httk.core.miniparser.**ebnf_unqote**(*s*)

httk.core.miniparser.**lexer**(*source*, *tokens*, *partial_tokens*, *literals*, *ignore*, *comment_markers=[]*, *verbosity=0*, *logger=<function logger>*)
A generator that turn source into tokens.

Args:

source (str): input string

tokens (dict): a dictonary that maps all tokens of the language on regular expressions that match them.

partial_tokens (dict): a dictionary that maps token names on regular expressions for partial token matches. This is used to allow finding longer matches if there is intermediate length input that does not match. E.g., to match 5.32e6 as a number instead as as Number(5.32) + Identifier(e) + Number(6).

literals (list): a list of single character strings that are to be treated as literals.

httk.core.miniparser.**logger**(**args*, ***kargs*)

This is the default logging function for diagnostic output. It prints the output in *args* on stdout.

Args:

loglevel: the level designated to the diagnostic output

args: list of arguments to print out

kargs: keyword flags. These are: pretty=True: formats the output using pprint.pprint(arg).

httk.core.miniparser.**parser**(*ls*, *source*, *verbosity=0*, *logger=<function logger>*)

This is a fairly straightforward implementation of an LR(1) parser. It should do well for parsing somewhat simple grammars.

The parser takes a language specification (ls), and a string to parse (source). The string is then parsed according to that ls into a syntax tree, which is returned.

An ls is produced by calling the function *build_ls* (see help(build_ls))

Args: ls: language specification produced by build_ls. source: source string to parse.

httk.core.miniparser.**split_chars_strip_comments**(*source*, *comment_markers*)

Helper function for the lexer that reads input and strips comments, while keeping track of absolute position in the file.

Args:

source (str): input string

comment_markers (list of tuples): a list of entries (start_marker, end_marker) that designate comments. A marker can be end-of-line or end with end-of-line, but multiline comment separators are not allowed, i.e., no characters may follow the end-of-line.

httk.core.project module

class httk.core.project.**Project**(*name*, *description*, *project_key*, *keys*)

Bases: *httk.core.httkobject.HttkObject*

add_ref(*ref*)

add_refs(*refs*)

add_tag(*tag*, *val*)

add_tags(*tags*)

classmethod create(*name*, *description*, *project_key*, *keys*)

Create a Project object.

get_refs()

get_tag(*tag*)

get_tags()

class httk.core.project.**ProjectOwner**(*project*, *owner_key*)

Bases: *httk.core.httkobject.HttkObject*

classmethod create(*project*, *owner*)
Create a Project object.

class httk.core.project.**ProjectRef**(*project*, *reference*)
Bases: *httk.core.httkobject.HttkObject*

class httk.core.project.**ProjectTag**(*project*, *tag*, *value*)
Bases: *httk.core.httkobject.HttkObject*

httk.core.project.**main**()

httk.core.reference module

class httk.core.reference.**Author**(*last_name*, *given_names*)
Bases: *httk.core.httkobject.HttkObject*

Object for keeping track of tags for other objects

classmethod create(*last_name*, *given_names*)
Create a Author object.

class httk.core.reference.**Reference**(*ref*, *authors=None*, *editors=None*, *journal=None*, *journal_issue=None*, *journal_volume=None*, *page_first=None*, *page_last=None*, *title=None*, *year=None*, *book_publisher=None*, *book_publisher_city=None*, *book_title=None*)
Bases: *httk.core.httkobject.HttkObject*

A reference citation

classmethod create(*ref=None*, *authors=None*, *editors=None*, *journal=None*, *journal_issue=None*, *journal_volume=None*, *page_first=None*, *page_last=None*, *title=None*, *year=None*, *book_publisher=None*, *book_publisher_city=None*, *book_title=None*)
Create a Reference object.

httk.core.reference.**main**()

httk.core.signature module

class httk.core.signature.**Signature**(*signature_data*, *key*)
Bases: *httk.core.httkobject.HttkObject*

classmethod create(*signature_data*, *key*)
Create a Computation object.

class httk.core.signature.**SignatureKey**(*keydata*, *description*)
Bases: *httk.core.httkobject.HttkObject*

classmethod create(*keydata*, *description*)
Create a Computation object.

httk.core.signature.**main**()

httk.core.template module

httk.core.template.**apply_template**(*template*, *output*, *envglobals=None*, *envlocals=None*)
Simple Python template engine.

The file 'template' is turned into a new file 'output' replacing the following: $name -> the value of the variable 'name' in the scope provided by locals and globals. $(python statement) -> result of evaluating the python statment. ${some python code} -> text on stdout from running that python code.

Note: it is safe for the code inside the template to load the file it eventually will replace.

httk.core.template.**apply_templates**(*inputpath*, *outpath*, *template_suffixes='template'*, *envglobals=None*, *envlocals=None*, *mkdir=True*)

Apply one or a series of templates throughout directory tree.

template_suffixes: string or list of strings that are the suffixes of templates that are to be applied. name: subdirectory in which to apply the template, defaults to last subrun created, or '.' if no subrun have been created.

httk.db package

Subpackages

httk.db.backend package

Submodules

httk.db.backend.sqlite module

This provides a thin abstraction layer for SQL queries, implemented on top of sqlite,3 to make it easier to exchange between SQL databases.

class httk.db.backend.sqlite.**Sqlite**(*filename*)

Bases: object

class SqliteCursor(*db*)

Bases: object

close()

description

execute(*sql*, *values=[]*)

fetchall()

fetchone()

alter(*sql*, *values*, *cursor=None*)

close()

commit()

create_table(*name*, *primkey*, *columnnames*, *columntypes*, *cursor=None*, *index=None*)

cursor()

get_row(*table*, *primkeyname*, *primkey*, *columnnames*, *cursor=None*)

get_rows(*table*, *primkeyname*, *primkeys*, *columnnames*, *cursor=None*)

get_val(*table*, *primkeyname*, *primkey*, *columnname*, *cursor=None*)

insert(*sql*, *values*, *cursor=None*)

insert_row(*name*, *columnnames*, *columnvalues*, *cursor=None*)

modify_structure(*sql*, *values*, *cursor=None*)

query(*sql*, *values*, *cursor=None*)

rollback()

table_exists(*name*, *cursor=None*)

update(*sql*, *values*, *cursor=None*)

update_row(*name*, *primkeyname*, *primkey*, *columnnames*, *columnvalues*, *cursor=None*)

httk.db.backend.sqlite.**db_close**(*connection*)

httk.db.backend.sqlite.**db_open**(*filename*)

httk.db.backend.sqlite.**db_sqlite_close_all**()

httk.db.store package

Stores are abstract keepers of data. The only one properly implemented right now is sqlite, but others are possible. Trivialstore stores data just in the python classes, and dictstore stores all data in a dictionary.

TODO: Note: since a few changes back I think neither trivialstore or dictstore currently works the way they should.

Submodules

httk.db.store.dictstore module

class httk.db.store.dictstore.**DictStore**

Bases: object

Simplified fake database store in a dict, for testing primarily; though it can be used as a fast database-like engine that enables reterival of data

class Keeper(*store*, *table*, *sid*)

Bases: object

puts(***args*)

basics = [<type 'int'>, <type 'float'>, <type 'str'>, <type 'bool'>]

create_table(*table*, *types*)

get(*table*, *sid*, *name*)

insert(*table*, *keyvals*)

new(*table*, *types*, *keyvals*)

put(*table*, *sid*, *name*, *val*)

puts(*table*, *sid*, ***args*)

retrieve(*table*, *types*, *sid*)

httk.db.store.sqlstore module

class httk.db.store.sqlstore.**SqlStore**(*db*)
Bases: object

Keep objects in an sql database

class Keeper(*store*, *table*, *types*, *sid*)
Bases: object

puts(***args*)

basics = [<type 'int'>, <type 'float'>, <type 'str'>, <type 'bool'>, <class 'httk.core

commit()

create_table(*table*, *types*, *cursor=None*)

delay_commit()

get(*table*, *sid*, *types*, *name*)

insert(*table*, *types*, *keyvals*, *cursor=None*, *updatesid=None*)

new(*table*, *types*, *keyvals=None*, *updatesid=None*)

put(*table*, *sid*, *types*, *name*, *val*)

puts(*table*, *sid*, ***args*)

retrieve(*table*, *types*, *sid*)

save(*obj*)

searcher()

httk.db.store.trivialstore module

Submodules

httk.db.filteredcollection module

class httk.db.filteredcollection.**BinaryBooleanOp**(*context*, *operator*, *left*, *right*)
Bases: *httk.db.filteredcollection.Expression*

class httk.db.filteredcollection.**BinaryComparison**(*context*, *operator*, *left*, *right*)
Bases: *httk.db.filteredcollection.Expression*

class httk.db.filteredcollection.**BinaryOp**(*context*, *operator*, *left*, *right*)
Bases: *httk.db.filteredcollection.Expression*

class httk.db.filteredcollection.**DeclaredFunction**(*context*, *name*, *srctable=None*)
Bases: object

class httk.db.filteredcollection.**Expression**(*context*, *exprtype*, **args*)
Bases: object

get_srctable_context()

is_in(**args*)

like(**args*)

class httk.db.filteredcollection.**FCDict**(*data=None*)
Bases: *httk.db.filteredcollection.FilteredCollection*

This implements a filtered collection purely backed by a dictionary and python evaluation.

Note: FCSqliteMemory will usually be faster. (However, you need this class if you need to express filters and expressions using python functions rather than Sqlite functions.)

copy()

data(*outid=None*)

Return an object where the attributes are accessible as properties. I.e. data = myFCDict.data myFCDict.set_filter(data.example == data.otherexample*2)

function(*name*)
Define a python function object for use when expressing filter queries and column expressions. (You cannot define a filter with a "bare function", since it would be called directly at the point of defining the filter.) Validy/existence of this function is not checked until the collection is iterated over.

class httk.db.filteredcollection.**FCMultiDict**(*data=None*)
Bases: *httk.db.filteredcollection.FilteredCollection*

This class allows you to combine a number of filtered collections and put filters on any combination of them together. Just create a separate FilteredCollection from each data source, and pass them in a list to the constructor of this class.

Filters that only apply to one of the FilteredCollections can be put on those collections instead, while a filter that applies to more than one must be set on this class.

add(*filterexpr*)
Append a filter to the filters currently filtering the FilteredCollection. When iterating over the FilteredCollection, a result is only included if it matches all the filters.

copy()

data(*name*, *outid=None*)
Return an object where the attributes of respective filtered collection is accessible as attributes. An example:

languagereview = FCMultiDict('programming':programming_fc, 'review':review_fc) language = languagereview.data('programming').language review = languagereview.data('review') myFCMultiDict.set_filter(language.name == "python" & review.goodness > 9)

subdata(*name*, *table*, *outid=None*, *key='rowid'*, *subkey=None*)
Return an object where the attributes of respective filtered collection is accessible as attributes. An example:

languagereview = FCMultiDict('programming':programming_fc, 'review':review_fc) language = languagereview.data('programming').language review = languagereview.data('review') myFCMultiDict.set_filter(language.name == "python" & review.goodness > 9)

class httk.db.filteredcollection.**FCMultiSqlite**(*dicts=None*)
Bases: *httk.db.filteredcollection.FilteredCollection*

This class allows you to combine a number of filtered collections and put filters on any combination of them together. Just create a separate FilteredCollection from each data source, and pass them in a list to the constructor of this class.

Filters that only apply to one of the FilteredCollections should preferably be put on those collections, while a filter that applies to more than one must be set on this class, *using field definitions made with this class*.

class httk.db.filteredcollection.**FCSqlite**(*sqlstore*)
Bases: *httk.db.filteredcollection.FilteredCollection*

function(*name*)
Define a function object for expressing functions in filter queries. Validity/existence of this function may not be tested until an iteration over matching entries is performed.

sql()

store_table(*name*)
Store the result of the filtered collection in a new table named 'name'.

subtable(*name*, *table*, *outid=None*, *key='rowid'*, *subkey=None*)
Defines a table object to use in filters (for add) and expressions (in set_columns).

table(*name*, *outid=None*)
Defines a table object to use in filters (for add) and expressions (in set_columns).

class httk.db.filteredcollection.**FilteredCollection**
Bases: object

Main interface for filtered collections.

Apart from what is declared here, each subclass should define e.g. 'table', 'column', 'function' methods for defining fields for use for filters (in, e.g., set_filter) and expressions (in, e.g., set_columns).

add(*filterexpr*)
Append a filter to the filters currently filtering the FilteredCollection. When iterating over the FilteredCollection, a result is only included if it matches all the filters.

add_all(*filterexpr*)
Append a filter to the filters currently filtering the FilteredCollection. When iterating over the FilteredCollection, a result is only included if it matches all the filters.

add_sort(*expression*, *direction='ASC'*)
Define which columns should be included in the results when iterating over a FilteredCollection. attributes is a list of tuples consisting of (name,definition) where definition can be any expression in columns.

Default is to show all columns of all tables defined. (See FilteredColleciton.table)

duplicate(*other*)

output(*expression*, *name=None*)
Define which columns should be included in the results when iterating over a FilteredCollection. attributes is a list of tuples consisting of (name,definition) where definition can be any expression in columns.

Default is to show all columns of all tables defined. (See FilteredColleciton.table)

reset()
Clear any filtering done on the data source.

store_table(*name*)
Store the result of the filtered collection in a new table named 'name'.

variable(*obj*, *outid=None*, *parent=None*, *parentkey=None*, *subkey=None*)

class httk.db.filteredcollection.**Function**(*context*, *name*, *srctable*, **args*)
Bases: *httk.db.filteredcollection.Expression*

class httk.db.filteredcollection.**TableOrColumn**(*context*, *name*, *parent=None*, *outid=None*, *key=None*, *subkey=None*, *srctable=None*, *indirection=1*, *classref=None*)
Bases: *httk.db.filteredcollection.Expression*

class httk.db.filteredcollection.**UnaryBooleanOp**(*context*, *operator*, *right*)
Bases: *httk.db.filteredcollection.Expression*

httk.db.filteredcollection.**fc_checkcontext**(*context*, **exprs*)

httk.db.filteredcollection.**fc_eval**(*expr*, *data*)

httk.db.filteredcollection.**fc_get_srctable_context**(**args*)

httk.db.filteredcollection.**fc_sql**(*expr*)

httk.db.filteredcollection.**instantiate_from_store**(*classobj*, *store*, *id*)

httk.db.httkobjdbplugin module

class httk.db.httkobjdbplugin.**HttkObjDbPlugin**
Bases: *httk.core.httkobject.HttkPlugin*

fetch_codependent_data(*store*)

plugin_init(*obj*)

store(*store*, *avoid_duplicate=True*)

store_codependent_data(*store*)

httk.db.storable module

class httk.db.storable.**Storable**(*types=None*, *index=None*)
Bases: object

Superclass for handling various forms of data storage, retreival, etc. Class object representing data should inherit from Storable.

All public variables must be initalized in a call to storable_init() inside __init__(). Other member variables are OK, but must begin with '_', and all methods must handle these variables not being initialized. For private variables that needs to be preserved: let them start with '_' AND declare them in storable_init().

classmethod find_all(*obj*, *store*, *member*, *value*, *types*)
Convinience method to do a very simple search of type: find all entries where member = value.

classmethod find_one(*obj*, *store*, *member*, *value*, *types*)
Convinience^2 method to do a very simple search of type: find one entry where member = value.

storable_init(*store*, *updatesid=None*, ***keyvals*)
All Storable objects need to call this method in __init__(). Name should be a 'somewhat qualified' class name.

trivialstore = <httk.db.storable.TrivialStore object>

classmethod variable(*searcher*, *name*, *types*, *outid=None*, *parent=None*)

class httk.db.storable.**TrivialStore**
Bases: object

Very simple storage class that just stores everything into an individual dictionary, just like regular python objects work

new(*table*, *types*, *keyvals*)

retrieve(*table*, *types*, *sid*)

httk.db.storable.**storable_props**(**props*)

httk.db.storable.**storable_types**(*name*, **keyvals*, ***flags*)

httk.external package

Submodules

httk.external.aflow_ext module

httk.external.aflow_ext.**aflow**(*ioa_in*, *args*, *timeout=30*)

httk.external.aflow_ext.**standard_primitive**(*struct*)

httk.external.ase_glue module

class httk.external.ase_glue.**StructureAsePlugin**

Bases: *httk.core.httkobject.HttkPlugin*

classmethod from_Atoms(*atoms*)

name = 'ase'

plugin_init(*struct*)

to_Atoms()

httk.external.ase_glue.**ase_atoms_to_structure**(*atoms*, *hall_symbol*)

httk.external.ase_glue.**ase_read_structure**(*f*)

httk.external.ase_glue.**ase_write_struct**(*struct*, *ioa*, *format=None*)

httk.external.ase_glue.**coordgroups_reduced_rc_to_unitcellsites**(*coordgroups*, *basis*, *hall_symbol*, *reduce=False*)

httk.external.ase_glue.**ensure_ase_is_imported**()

httk.external.ase_glue.**primitive_from_conventional_cell**(*atoms*, *spacegroup=1*, *setting=1*)

Returns primitive cell given an Atoms object for a conventional cell and it's spacegroup.

Code snippet kindly posted by Jesper Friis, https://listserv.fysik.dtu.dk/pipermail/ase-users/2011-January/000911.html

httk.external.ase_glue.**structure_to_ase_atoms**(*struct*)

httk.external.cif2cell_ext module

httk.external.cif2cell_ext.**cif2cell**(*cwd*, *args*, *timeout=30*)

httk.external.cif2cell_ext.**cif_to_structure_noreduce**(*f*)

httk.external.cif2cell_ext.**cif_to_structure_reduce**(*f*)

httk.external.cif2cell_ext.**coordgroups_reduced_rc_to_unitcellsites**(*coordgroups*, *basis*, *hall_symbol*)

httk.external.cif2cell_ext.**ensure_has_cif2cell**()

httk.external.command module

class httk.external.command.**Command**(*cmd*, *args*, *cwd=None*, *inputstr=None*, *stophook=None*)
Bases: object

receive()

run(*timeout*, *debug=False*)

send(*command*)

start()

stdin

stop()

wait_finish(*timeout=None*)

httk.external.command.**find_executable**(*executables*, *config_name*)

httk.external.gulp_ext module

httk.external.gulp_ext.**jmol**(*cwd*, *args*, *timeout=10*)

httk.external.gulp_ext.**show**(*struct*)

httk.external.isotropy_ext module

httk.external.isotropy_ext.**ensure_has_isotropy**()

httk.external.isotropy_ext.**isotropy**(*cwd*, *args*, *inputstr*, *timeout=30*)

httk.external.isotropy_ext.**struct_process_with_isotropy**(*struct*)

httk.external.isotropy_ext.**uc_reduced_coordgroups_process_with_isotropy**(*coordgroup*, *cell*, *get_wyckoff=False*)

httk.external.jmol module

httk.external.jmol.**ensure_has_cif2cell**()

httk.external.jmol.**main**()

httk.external.jmol.**run**(*cwd*, *args*, *timeout=None*)

httk.external.jmol.**start**(*cwd='./'*, *args=['-I']*)

httk.external.numpy_ext module

httk.external.platon_ext module

httk.external.platon_ext.**addsym**(*struct*)

httk.external.platon_ext.**addsym_spacegroup**(*struct*)

httk.external.platon_ext.**cif_to_sgstructure**(*ioa*)

httk.external.platon_ext.**ensure_has_platon**()

httk.external.platon_ext.**platon**(*cwd*, *args*, *timeout=60*)

httk.external.platon_ext.**structure_addsym_and_tidy**(*struct*)

httk.external.platon_ext.**structure_tidy**(*struct*)

httk.external.platon_ext.**structure_tidy_old**(*struct*)

httk.external.platon_ext.**structure_to_sgstructure**(*struct*)

httk.external.pymatgen_glue module

httk.external.pymatgen_glue.**ensure_pymatgen_is_imported**()

httk.external.pymatgen_glue.**set_mp_key**(*key*)

httk.external.pyspglib_ext module

pyspglib external module

httk.external.pyspglib_ext.**analysis**(*struct*, *symprec=1e-05*)

httk.external.pyspglib_ext.**ensure_pyspg_is_imported**()

httk.external.pyspglib_ext.**primitive**(*struct*, *symprec=1e-05*)

httk.external.pyspglib_ext.**structure_to_spglib_atoms**(*struct*)

httk.external.subimport module

httk.external.subimport.**submodule_import_external**(*modulepath*, *pkg*)

httk.graphics package

Subpackages

httk.graphics.matplotlib package

Submodules

httk.graphics.matplotlib.arrowplot module

httk.graphics.matplotlib.polygonplot module

httk.httkio package

httk Io module

General methods for reading and writing of data, conversions, etc.

Submodules

httk.httkio.cif module

httk.httkio.cif.**main**()

httk.httkio.cif.**read_cif**(*ioa*, *pragmatic=True*, *use_types=False*)
Generic cif reader, given a filename / ioadapter it places all data in a python dictionary.

It returns a tuple: (header, list) Where list are pairs of data blocks names and data blocks

Each data block is a dictionary with tag_name:value

For loops, value is another dictionary with format column_name:value

The optional parameter pragmatic regulates handling of some counter-intuitive aspects of the cif specification, where the default pragmatic=True handles these features the way people usually use them, whereas pragmatic=False means to read the cif file precisely according to the spec. For example, in a multiline text field:

```
;
some text
;
```

Means the string 'nsome text'. For this specific case pragmatic=True removes the leading newline.

set use_types to True to convert things that look like floats and integers to those respective types

httk.httkio.cif.**write_cif**(*ioa*, *data*, *header=None*, *max_line_length=80*, *use_types=False*)
Generic cif writer, given a filename / ioadapter

data = the cif data to write as an (ordered) dictionary of tag_name:value

header = the header (comment) segment

max_line_length = the maximum number of characters allowed on each line. This should not be set < 80 (there is no point, and the length calculating algorithm breaks down at some small line length)

use_types =

if True: always quote values that are of string type. Numeric values are put in the file unquoted (as they should) if False (default): also strings that look like cif numbers are put in the file unquoted

For loops, value is another dictionary with format column_name:value

The optional parameter pragmatic regulates handling of some counter-intuitive aspects of the cif specification, where the default pragmatic=True handles these features the way people usually use them, whereas pragmatic=False means to read the cif file precisely according to the spec. For example, in a multiline text field:

```
;
some text
;
```

Means the string 'nsome text'. For this specific case pragmatic=True removes the leading newline.

set use_types to True to convert things that look like floats and integers to those respective types

httk.httkio.load module

httk.httkio.load.**load**(*ioa*, *ext=None*)
A *very* generic file reader method.

Load a file into a suitable httk object. Try to do the most sane thing possible given the input file. If you know what to expect from the input file, it may be safer to use a targeted method for that file type.

httk.httkio.save module

httk.httkio.save.**save**(*obj*, *ioa*, *ext=None*)
A *very* generic file writer method.

Load a file into a suitable httk object. Try to do the most sane thing possible given the input file. If you know what to expect from the input file, it may be safer to use a targeted method for that file type.

httk.httkweb package

Submodules

httk.httkweb.app_curses module

class httk.httkweb.app_curses.**MyHTMLParser**
Bases: HTMLParser.HTMLParser

handle_data(*data*)

handle_endtag(*tag*)

handle_startendtag(*tag*, *attrs*)

handle_starttag(*tag*, *attrs*)

ignore_close_tags = ['meta', 'link', 'br', 'img', 'input']

ignore_content = ['script', 'style']

text()

class httk.httkweb.app_curses.**WebviewCurses**(*appdir*)
Bases: object

open_url(*url*)

httk.httkweb.app_curses.**render_page**(*stdscr*)

httk.httkweb.app_qt5 module

httk.httkweb.app_qt5.**run_app**(*appdir*, *renderers=None*, *template_engines=None*, *function_handlers=None*, *config='config'*, *debug=True*, *override_global_data=None*)

httk.httkweb.functionhandler_httk module

class httk.httkweb.functionhandler_httk.**FunctionHandlerHttk**(*function_dir*, *function_filename*, *arg_names*, *global_data*, *instanced_template_engine=None*)

Bases: object

execute(*args=None*)

execute_and_format(*args*, *data*)

get_dependency_filenames()

httk.httkweb.helpers module

class httk.httkweb.helpers.**UnquotedStr**(*val*)

Bases: object

httk.httkweb.helpers.**identify**(*topdir*, *relative_url*, *ext_to_class_mapper*, *allow_urls_without_ext=True*)

httk.httkweb.helpers.**read_config**(*srcdir*, *renderers*, *default_global_data=None*, *override_global_data=None*, *config='config'*)

httk.httkweb.helpers.**setup**(*renderers*, *template_engines*, *function_handlers*)

httk.httkweb.helpers.**setup_template_helpers**(*global_data*)

httk.httkweb.publish module

httk.httkweb.publish.**publish**(*srcdir*, *outdir*, *baseurl*, *renderers=None*, *template_engines=None*, *function_handlers=None*, *config='config'*, *override_global_data=None*)

httk.httkweb.render_httk module

class httk.httkweb.render_httk.**RenderHttk**(*render_dir*, *render_filename*, *global_data*)

Bases: object

adornment_chars = ['!', '"', '#', '$', '%', '&', "'", '(', ')', '*', '+', ',', '-', '.

bullet_item_markers = ['- ', '* ', '+ ']

content()

left_punctuation_chars = '\'[({<:"; -'

make_id(*s*)

metadata()

option_list_characters = ['-', '/']

right_punctuation_chars = ']\')}>:,!.?"; -'

rst_light_html_renderer(*content*)

rst_light_parse_textstyle(*content*, *start_marker*, *end_marker*, *style*, *allow_nested=False*, *unescape=True*, *handle_roles=False*, *handle_hyperlinks=False*)

rst_light_parser(*source*)

split_content(*source*)

httk.httkweb.render_rst module

class httk.httkweb.render_rst.**RenderRst**(*render_dir*, *render_filename*, *global_data*)

Bases: object

content()

metadata()

httk.httkweb.serve module

exception httk.httkweb.serve.**WebError**(*message*, *response_code*, *response_msg*, *longmsg=None*, *content_type='text/plain'*, *encoding='utf-8'*)

Bases: exceptions.Exception

httk.httkweb.serve.**serve**(*srcdir*, *port=80*, *baseurl=None*, *renderers=None*, *template_engines=None*, *function_handlers=None*, *debug=True*, *config='config'*, *override_global_data=None*)

httk.httkweb.serve.**startup**(*get_callback*, *post_callback=None*, *port=80*, *netloc=None*, *basepath='/'*, *debug=False*)

httk.httkweb.templateengine_httk module

class httk.httkweb.templateengine_httk.**HttkTemplateFormatter**

Bases: string.Formatter

format_field(*value*, *spec*, *quote=None*, *args=None*, *kwargs=None*)

get_field(*field_name*, *args*, *kwargs*)

vformat(*format_string*, *args*, *kwargs*, *used_args=None*, *recursion_depth=None*)

class httk.httkweb.templateengine_httk.**TemplateEngineHttk**(*template_dir*, *template_filename*, *base_template_filename=None*)

Bases: object

apply(*content=None*, *data=None*, **subcontent*)

get_dependency_filenames()

httk.httkweb.templateengine_templator module

class httk.httkweb.templateengine_templator.**TemplateEngineTemplator**(*template_dir*, *template_filename*, *base_template_filename=None*)

Bases: object

apply(*content=None*, *data=None*, **subcontent*)

get_dependency_filenames()

httk.httkweb.webgenerator module

class httk.httkweb.webgenerator.**Page**(*meta={}*)
Bases: object

update_metadata(*meta*)

class httk.httkweb.webgenerator.**WebGenerator**(*srcdir*, *global_data*, *renderers*, *template_engines*, *function_handlers*)
Bases: object

retrieve(*relative_url*, *query=None*, *allow_urls_without_ext=None*, *all_functions=False*)

httk.iface package

httk Interface module

- The interface between httk and other software. Note: the idea is that this module should be useable without the other software installed. E.g., generation of input files to gulp shouldn't require gulp installed.

Submodules

httk.iface.ase_if module

httk.iface.ase_if.**rc_structure_to_symbols_and_scaled_positions**(*struct*)

httk.iface.ase_if.**uc_structure_to_symbols_and_scaled_positions**(*struct*)

httk.iface.cif2cell_if module

httk.iface.cif2cell_if.**out_to_struct**(*ioa*)
Example input:

```
OUTPUT CELL INFORMATION
Symmetry information:
Trigonal crystal system.
Space group number     : 165
Hall symbol            : -P 3 2"c
Hermann-Mauguin symbol : P-3c1

Bravais lattice vectors :
  0.8660254  -0.5000000   0.0000000
  0.0000000   1.0000000   0.0000000
  0.0000000   0.0000000   1.0231037
All sites, (lattice coordinates):
Atom           a1          a2          a3
La      0.6609000   0.0000000   0.2500000
La      0.3391000   0.0000000   0.7500000
...
F       0.0000000   0.0000000   0.2500000
```

(continues on next page)

(continued from previous page)

```
F       0.0000000   0.0000000   0.7500000

Unit cell volume  : 328.6477016 A^3
Unit cell density :   3.5764559 u/A^3 =   5.9388437 g/cm^3
```

httk.iface.gulp_if module

httk.iface.gulp_if.**generate_fake_potentials**(*species*)

httk.iface.gulp_if.**generate_fake_potentials_try2**(*species*)

httk.iface.gulp_if.**structure_to_gulp**(*iof*, *struct*, *runspec='single conp'*, *postcards=[]*, *potentials=None*)

Writes a file on gulp input format.

httk.iface.isotropy_if module

httk.iface.isotropy_if.**out_to_cif**(*ioa*, *assignments*, *getwyckoff=False*)

httk.iface.isotropy_if.**reduced_coordgroups_to_input**(*coordgroups*, *cell*, *comment='FINDSYM input'*, *accuracy=0.001*)

httk.iface.isotropy_if.**struct_to_input**(*struct*)

httk.iface.jmol_if module

httk.iface.jmol_if.**structure_to_jmol**(*iof*, *struct*, *extbonds=True*, *repeat=None*, *copies=None*)

Converts structure into jmol format.

Example output format:: load data 'model' 1 Computation1 Al 0 0 0 end 'model' { 4 4 4 } supercell "x, y, z " unitcell [2.025 2.025 0 2.025 0 2.025 0 2.025 2.025] set slabByAtom TRUE unitcell {1/1 1/1 1/1} delete (NOT (unitcell OR connected(unitcell))) {connected(unitcell) AND NOT unitcell}.radius = 0 restrict cell={2 2 2} center visible zoom 0

httk.iface.openbabel_if_notstable module

httk.iface.platon_if module

This module is a mess and in need of heavy cleanup.

httk.iface.platon_if.**get_stidy_spacegroup**(*parse*)

httk.iface.platon_if.**platon_lis_to_struct_broken**(*ioa*)

Example input format:

```
                                                        ============
============================================================ Crystal Data␣
↪============================================================
                                                        ============
                        Input Cell  (Lattice Type: P)    -   Temp =   0K
↪  Reduced Cell     (Acta Cryst.(1976),A32,297-298)
```

(continues on next page)

(continued from previous page)

```
--------------------------------------------------------------------------------
 ↪  ------------------------------------------------
a =       3.47100  Angstrom                alpha =            90 Degree
 ↪  a  =    3.471     alpha  =   90.00  V  =    79.6
b =       3.47100                           beta =            90
 ↪  b  =    3.471     beta   =   90.00
c =       6.60300                          gamma =            90
 ↪  c  =    6.603     gamma  =   90.00

...

--------------------------------------------------------------------------------
 ↪------------------------------------------------
Flags Label         Fractional Coordinates (x,y,z)          Orthogonal
 ↪Coordinates (XO,YO,ZO)   Site SSN*SSOF =    S.O.F   Move Type
--------------------------------------------------------------------------------
 ↪------------------------------------------------
-     Ag(1)              1/4            1/4         0.61200      0.8677      0.
 ↪8677      4.0410   4mm  8   1/8          1    -    Met
-     Zr(2)              1/4            1/4         0.13700      0.8677      0.
 ↪8677      0.9046   4mm  8   1/8          1    -    Met
-     Ag(1)a            -1/4           -1/4        -0.61200     -0.8677     -0.
 ↪8677     -4.0410   4mm  8   1/8          1   5.455 Met
-     Zr(2)a            -1/4           -1/4        -0.13700     -0.8677     -0.
 ↪8677     -0.9046   4mm  8   1/8          1   5.455 Met
-     Ag(1)b            -1/4           -1/4         0.38800     -0.8678     -0.
 ↪8678      2.5620   4mm  8   1/8          1   5.456 Met
-     Zr(2)b            -1/4           -1/4         0.86300     -0.8678     -0.
 ↪8678      5.6984   4mm  8   1/8          1   5.456 Met
-     Ag(1)c            -1/4            3/4        -0.61200     -0.8677      2.
 ↪6033     -4.0410   4mm  8   1/8          1   5.465 Met
-     Zr(2)c            -1/4            3/4        -0.13700     -0.8678      2.
 ↪6033     -0.9046   4mm  8   1/8          1   5.465 Met
-     Ag(1)d            -1/4            3/4         0.38800     -0.8678      2.
 ↪6033      2.5620   4mm  8   1/8          1   5.466 Met
-     Zr(2)d            -1/4            3/4         0.86300     -0.8678      2.
 ↪6032      5.6984   4mm  8   1/8          1   5.466 Met
-     Ag(1)e             3/4           -1/4        -0.61200      2.6033     -0.
 ↪8677     -4.0410   4mm  8   1/8          1   5.555 Met
-     Zr(2)e             3/4           -1/4        -0.13700      2.6033     -0.
 ↪8677     -0.9046   4mm  8   1/8          1   5.555 Met
-     Ag(1)f             3/4           -1/4         0.38800      2.6033     -0.
 ↪8678      2.5620   4mm  8   1/8          1   5.556 Met
-     Zr(2)f             3/4           -1/4         0.86300      2.6032     -0.
 ↪8678      5.6984   4mm  8   1/8          1   5.556 Met
-     Ag(1)g             3/4            3/4        -0.61200      2.6033      2.
 ↪6033     -4.0410   4mm  8   1/8          1   5.565 Met
-     Zr(2)g             3/4            3/4        -0.13700      2.6033      2.
 ↪6033     -0.9046   4mm  8   1/8          1   5.565 Met
-     Ag(1)h             3/4            3/4         0.38800      2.6033      2.
 ↪6033      2.5620   4mm  8   1/8          1   5.566 Met
-     Zr(2)h             3/4            3/4         0.86300      2.6032      2.
 ↪6032      5.6984   4mm  8   1/8          1   5.566 Met
================================================================================
```

httk.iface.platon_if.**platon_lis_to_struct_broken2**(*ioa*)

Example input:

```
                                                                 ============
================================================== Crystal Data
↪================================================================
                                                                 ============
                        Input Cell  (Lattice Type: P)   -   Temp =   0K
↪  Reduced Cell     (Acta Cryst.(1976),A32,297-298)
--------------------------------------------------------------------------------
↪  ---------------------------------------------
a =       3.47100  Angstrom               alpha =            90 Degree
↪  a  =    3.471     alpha  =   90.00  V  =    79.6
b =       3.47100                          beta =            90
↪  b  =    3.471     beta   =   90.00
c =       6.60300                         gamma =            90
↪  c  =    6.603     gamma  =   90.00

...

================================================================================
10.0 Angstrom Coordination Sphere Around Atom I = Ag(1)  [ARU = 1555.01]
↪   1/4      1/4  0.61200    0.8677  0.8677  4.0410
--------------------------------------------------------------------------------
↪-------------------------------------------------
 Nr     d(I,J) To  Atom J  Symm_Oper. on Atom J    ARU(J)  Type     Phi      Mu
↪     X        Y        Z        XO      YO      ZO
--------------------------------------------------------------------------------
↪-------------------------------------------------
 1      2.9615 --  Zr(4)  [                    =         ] Intra-135.00  34.03
↪  -1/4     -1/4  0.86300   -0.8678 -0.8678  5.6984
 2      2.9615 --  Zr(4)n [1+x,1+y,z          =  1665.01] Intra  45.00  34.03
↪   3/4      3/4  0.86300    2.6032  2.6032  5.6984
 3      2.9615 --  Zr(4)j [x,1+y,z            =  1565.01] Intra 135.00  34.03
↪  -1/4      3/4  0.86300   -0.8678  2.6032  5.6984
 4      2.9615 --  Zr(4)l [1+x,y,z            =  1655.01] Intra -45.00  34.03
↪   3/4     -1/4  0.86300    2.6032 -0.8678  5.6984
 5      3.1364 --  Zr(3)  [                    =         ] Intra   0.00 -90.00
↪   1/4      1/4  0.13700    0.8677  0.8677  0.9046
```

httk.iface.platon_if.**platon_sites_to_styin**(*ioa*, *sites*, *cell*)

Example input:

```
P 4 B M
    5.5179    5.5179    3.9073   90.0000   90.0000   90.0000
Bi1       0.50000   0.00000   0.54500   0.50000
Ti1       0.00000   0.00000   0.00000
Na1       0.50000   0.00000   0.54500   0.50000
O1        0.00000   0.00000   0.51000
O2        0.72900   0.22900   0.01500
END
END
```

httk.iface.platon_if.**platon_styin_to_sgstruct**(*ioa*)

Example input:

F -4 3 M id=[0] dblock_code=[44325-ICSD] formula= 5.5000 5.5000 5.5000 90.0000 90.0000 90.0000 N Sb1 0.25000 0.25000 0.25000 Al1 0.00000 0.00000 0.00000 END END

httk.iface.platon_if.**platon_styout_to_sgstruct**(*ioa*)

Example input:

```
Results for  id=[0] dblock_code=[44325-I New: F-43m
=================================================

Pearson code : cF   8  Sb  4.0  Al  4.0
Cell parameters :   7.7782  7.7782  7.7782   90.000   90.000   90.000
Space group symbol : F -4 3 m   Number in IT : 216

Setting x,y,z             Origin ( 0.0000 0.0000 0.0000)  Gamma =  0.4330

       Al1       4(c)  1/4     1/4     1/4                            Al   1
      Sb1        4(a)  0       0       0                              Sb   1

Wyckoff sequence : c a

Volume of Unit Cell :   470.5842


OTHER Standardization with Similar Gamma :

Setting -x,-y,-z          Origin ( 0.7500 0.7500 0.7500)  Gamma =  0.4330

      Sb1        4(c)  1/4     1/4     1/4                            Sb   1
       Al1       4(a)  0       0       0                              Al   1

Wyckoff sequence : c a

Volume of Unit Cell :   470.5842
```

httk.iface.platon_if.**platon_styout_to_structure**(*ioa*, *based_on_struct=None*)

Example input:

```
Results for  id=[0] dblock_code=[44325-I New: F-43m
=================================================

Pearson code : cF   8  Sb  4.0  Al  4.0
Cell parameters :   7.7782  7.7782  7.7782   90.000   90.000   90.000
Space group symbol : F -4 3 m   Number in IT : 216

Setting x,y,z             Origin ( 0.0000 0.0000 0.0000)  Gamma =  0.4330

       Al1       4(c)  1/4     1/4     1/4                            Al   1
      Sb1        4(a)  0       0       0                              Sb   1

Wyckoff sequence : c a

Volume of Unit Cell :   470.5842


OTHER Standardization with Similar Gamma :

Setting -x,-y,-z          Origin ( 0.7500 0.7500 0.7500)  Gamma =  0.4330

      Sb1        4(c)  1/4     1/4     1/4                            Sb   1
       Al1       4(a)  0       0       0                              Al   1

Wyckoff sequence : c a

Volume of Unit Cell :   470.5842
```

httk.iface.platon_if.**sites_to_platon**(*ioa*, *sites*, *cell*, *precards*, *postcards*)
Writes a file on PLATONS input format.

httk.iface.platon_if.**structure_to_platon**(*ioa*, *struct*, *precards*, *postcards*)
Writes a file on PLATONS input format.

httk.iface.spglib_if module

httk.iface.spglib_if.**spglib_out_to_struct**(*out*)

httk.iface.vasp_if module

class httk.iface.vasp_if.**OutcarReader**(*ioa*)

parse()

httk.iface.vasp_if.**calculate_kpoints**(*struct*, *dens=20*)

httk.iface.vasp_if.**copy_template**(*dirtemplate*, *dirname*, *templatename*)

httk.iface.vasp_if.**get_magmom**(*symbol*)

httk.iface.vasp_if.**get_magnetizations**(*ionlist*, *high*, *low*)

httk.iface.vasp_if.**get_pseudopotential**(*species*, *poscarspath=None*)

httk.iface.vasp_if.**is_dualmagnetic**(*ion*, *ionlist*)

httk.iface.vasp_if.**magnetization_recurse**(*basemags*, *dualmags*, *high*, *low*)

httk.iface.vasp_if.**poscar_to_strs**(*fio*, *included_decimals=''*)

Parses a file on VASPs POSCAR format. Returns (cell, scale, vol, coords, coords_reduced, counts, occupations, comment)

where cell: 3x3 nested list of *strings* designating the cell scale: *string* representing the overall scale of the cell vol: *string* representing the volume of the cell (only one of scale and vol will be set, the other one = None) coords: Nx3 nested list of *strings* designating the coordinates coords_reduced: bool, true = coords are given in reduced coordinate (in vasp D or Direct), false = coords are given in cartesian coordinates counts: how many atoms of each type occupations: which species of each atom type (integers), or -1, ... -N if no species are given. comment: the comment string given at the top of the file

httk.iface.vasp_if.**poscar_to_structure**(*f*, *included_decimals=''*)

httk.iface.vasp_if.**prepare_single_run**(*dirpath*, *struct*, *poscarspath=None*, *template='t:/vasp/single/static'*, *overwrite=False*)

httk.iface.vasp_if.**read_outcar**(*ioa*)

httk.iface.vasp_if.**structure_to_comment**(*struct*)

httk.iface.vasp_if.**structure_to_poscar**(*f*, *struct*, *fix_negative_determinant=False*, *comment=None*, *primitive_cell=True*)

httk.iface.vasp_if.**write_generic_kpoints_file**(*fio*, *comment=None*, *mp=True*)

httk.iface.vasp_if.**write_kpoints_file**(*fio*, *kpoints*, *comment=None*, *mp=True*, *gamma_centered=False*)

httk.iface.vasp_if.**write_poscar**(*fio*, *cell*, *coords*, *coords_reduced*, *counts*, *occupations*, *comment='Comment'*, *scale='1'*, *vol=None*)
Writes a file on VASPs POSCAR format. Where it says *string* below, any type that works with str(x) is also ok.

Input arguments f: file stream to put output on cell: 3x3 nested list of *strings* designating the cell coords: Nx3 nested list of *strings* designating the coordinates coords_reduced: bool, true = coords are given in reduced coordinate (in vasp D or Direct), false = coords are given in cartesian coordinates counts: how many atoms of each type occupations: which species of each atom type comment: (optional) the comment string given at the top of the file scale: (optional) *string* representing the overall scale of the cell vol: *string* representing the volume of the cell (only one of scale and vol can be set)

httk.task package

Submodules

httk.task.reader module

httk.task.reader.**main**()

httk.task.reader.**read_manifest**(*ioa*, *verify_signature=True*)

httk.task.reader.**reader**(*projectpath*, *inpath*, *excludes=None*, *default_description=None*, *project_counter=0*, *force_remake_manifests=False*)

Read and yield all tasks from the project in path

httk.task.reader.**submit_reader**(*projectpath*, *default_description=None*, *excludes=None*, *project=None*, *project_counter=0*)

Read and yield all tasks from the project in path

For 'submitted' projects that already have manifests and should not be altered in any way.

httk.task.taskmgr module

httk.task.taskmgr.**create_batch_task**(*dirpath*, *template='t:vasp/batch/vasp-relax-formenrg'*, *args=None*, *project='noproject'*, *assignment='unassigned'*, *instantiate_name='ht.instantiate.py'*, *overwrite=False*, *overwrite_head_dir=True*, *remove_instantiate=True*, *name=None*, *priority=3*)

Submodules

httk.cli module

httk.cli.**main**()

httk.versioning module

10.3.2 Indices and tables

- genindex
- modindex
- search

10.4 *httk* Installation Instructions

10.4.1 Installation

There are a few alternative ways to download and install *httk*. Httk presently consists of a python library and a few programs. If you just want access to use the python library, and do not need the external programs, the install is very easy.

Note: for *httk* version 2.0 we will go over to a single program ('python endpoint') `httk`, for which the pip install step should be sufficient to get a full install.

(There are also separate instructions below for advanced users that want to do a direct manual install without the Python pip installed.)

Alternative 1: Install via pip to just access the python library

1. You need Python 2.7 and access to pip in your terminal window. (You can get Python and pip, e.g., by installing the Python 2.7 version of Anaconda, https://www.anaconda.com/download, which should give you all you need on Linux, macOS and Windows.)
2. Issue in your terminal window:

   ```
   pip install httk
   ```

   If you at a later point want to upgrade your installation, just issue:

   ```
   pip install httk --upgrade
   ```

You should now be able to simply do `import httk` in your python programs to use the *httk* python library.

Alternative 2: Install via pip for python library + binaries + ability to develop *httk*

1. In addition to Python 2.7 and pip, you also need git. You can get git from here: https://git-scm.com/
2. Issue in your terminal window:

   ```
   git clone https://github.com/rartino/httk
   cd httk
   pip install --editable . --user
   ```

   If you at a later point want to upgrade your installation, just go back to the *httk* directory and issue:

   ```
   git pull
   pip install . --upgrade --user
   ```

3. To setup the paths to the *httk* programs you also need to run:

   ```
   source /path/to/httk/init.shell
   ```

   where `/path/to/httk` should be the path to where you downloaded *httk* in the steps above. To make this permanent, please add this line to your shell initialization script, e.g., ~/.bashrc

You are now ready to use *httk*.

Notes:

- The above instructions give you access to the latest stable release of httk. To get the latest developer relase (which may or may not work), issue:

```
git checkout devel
pip install . --upgrade --user
```

  in your httk directory. To switch back to the stable release, do:

```
git checkout master
pip install . --upgrade --user
```

- An alternative to installing with `pip install` is to just run httk out of the httk directory. In that case, skip the pip install step above and just append `source ~/path/to/httk/init.shell` to your shell init files, with `~/path/to/httk` replaced by the path of your httk directory.)*

Alternative 3: For experienced users: direct manual install

If you are somewhat familiar with the command line in Linux, Unix, MacOSX or cygwin, and don't want to mess with python, all you need to do is download the archive (see: http://httk.openmaterialsdb.se/downloads.html) uncompress it in a directory of your choosing, and configure your environment in your environment init file (.bashrc or .cshrc) either by inserting `source /path/to/.../httk/init.shell` or by inserting instructions that adds the `httk/bin` directory to your `PATH` environment variable, and the `httk` directory to your `PYTHONPATH` environment variable.

That is all that is needed. As your first test, you can try to run `Examples/0_import_httk/0_import_httk.py`. (Please be aware that the first time you run this command it can be rather slow, since python is creating `*.pyc` files for all httk modules.)

Alternative 4: Step-by-step instructions for installation from archive

Find the latest relase download at this link: https://github.com/rartino/httk/releases/latest, and get the link to the `httk-<version>.tgz` archive.

Run the following in a terminal:

```
mkdir -p ~/bin/python
cd ~/bin/python
curl -L <download link> --output httk-<version>.tgz
tar -zxf httk-<version>.tgz
rm -f httk-<version>.tgz
```

where you have to fill in <download link> and <version> according to the release page.

The archive extaction (tar -zxf) will have created a subdirectory named after the actual version of httk that you downloaded. Check this with the command `ls`. Lets say you see `httk-1.1.2`, then do the following:

```
ln -f -s httk-1.1.2 httk-latest
source ~/bin/python/httk-latest/init.shell
```

If you add the very last line to your `.bashrc` and/or `.cshrc`, httk will work in all new terminals you open. (Or alternatively, just add `~/bin/python/httk-latest/bin/` to your `PATH` environment variable, and `~/bin/python/httk-latest` to your PYTHONPATH environment varibale.) If you cannot figure out how to do this on your system, you will have to re-run `source ~/bin/python/httk-latest/setup.shell` every time you want to use httk.

You can now start using httk. There is no further compiling, etc. required.

As your first test, you can try to run:

```
~/bin/python/httk-latest/Examples/0_import_httk/0_import_httk.py
```

This program simply loads the httk library and prints out its version, if everything works. Please be aware that the first time you run this command it can be rather slow, since python is creating `*.pyc` files for all httk modules.

Upgrade manual installation

This assumes you have followed the step-by-step installation instructions above. To upgrade, first check what version you presently have with:

```
ls ~/bin/python/
```

(look for the highest numbered httk-* directory)

Then find the latest relase download at this link: https://github.com/rartino/httk/releases/latest, and get the link to the `.tar.gz` archive.

Then do this:

```
cd ~/bin/python
rm -f httk-latest.tgz
curl -L <download link> --output httk-<version>.tar.gz
tar -zxf httk-<version>.tgz
rm -f httk-<version>.tar.gz
```

If the new version is, e.g., v1.1.3):

```
cp httk-latest/httk.cfg httk-1.1.3/httk.cfg
ln -f -s httk-1.1.3 ../httk-latest
```

This concludes the upgrade.

10.4.2 Download Source code

The source code of *httk* is available at github: https://github.com/rartino/httk

An archive of the source code of the latest version can be downloaded here: https://github.com/rartino/httk/releases/latest

10.4.3 Windows

These instructions may be expanded in the future. For now, what you need to do is download cygwin and when aksed what software to install, include

> wget, python

After cygwin is installed, start a cygwin terminal and follow the instructions above.

10.4.4 Optional configuration

Edit the `httk.cfg` file in the httk directory to configure paths to other software that you want to use from httk. For programs (e.g., `isotropy`) you want the path to point at the executable. For python libraries, you want the path setting to point at the directory you would include in `PYTHONPATH`, i.e., a directory that typically contains a subdirectory with the name of the package.

Note: if you don't have certain software, don't worry, just leave the line blank. If you have some libraries installed in the system (e.g. 'import ase' works), then you can also leave the lines blank. If you want to make sure *not* to use system libraries, set allow_system_libs=no (this is useful if you are forced to work on a machine with too old versions installed in the system)

10.5 The httk package

This page documents the features of the httk package most relevant for regular users. For a complete listing of members and subpackages, please refer to the full API documentation instead, *Full httk API documentation*.

10.5.1 Introduction

The high-throughput toolkit (httk)

A set of tools and utilities meant to help with:

- Project management, preparation of large-scale computational project.
- **Execution of large-scale computational projects**
  - interface with supercomputer cluster queuing systems, etc.
  - aid with scripting multi-stage runs
  - retrieval of data from supercomputers
- Storage of data in databases
- Search, retrieval and 'processing' of data in storage
- Analysis (especially as a helpful interface against 3:rd party software)

10.5.2 Helpful constants

httk.**httk_root**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

httk.**version**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

10.5.3 Main I/O

httk.**load**(*ioa*, *ext=None*)
A *very* generic file reader method.

Load a file into a suitable httk object. Try to do the most sane thing possible given the input file. If you know what to expect from the input file, it may be safer to use a targeted method for that file type.

httk.**save**(*obj*, *ioa*, *ext=None*)
A *very* generic file writer method.

Load a file into a suitable httk object. Try to do the most sane thing possible given the input file. If you know what to expect from the input file, it may be safer to use a targeted method for that file type.

10.5.4 FracVector

class httk.**FracVector**(*noms*, *denom=1*)

FracVector is a general *immutable* N-dimensional vector (tensor) class for performing linear algebra with fractional numbers.

A FracVector consists of a multidimensional tuple of integer nominators, and a single shared integer denominator.

Since FracVectors are immutable, every operation on a FracVector returns a new FracVector with the result of the operation. A created FracVector never changes. Hence, they are safe to use as keys in dictionaries, to use in sets, etc.

Note: most methods returns FracVector results that are not simplified (i.e., the FracVector returned does *not* have the smallest possible integer denominator). To return a FracVector with the smallest possible denominator, just call FracVector.simplify() at the last step.

class httk.**FracScalar**(*nom*, *denom*)

Represents the fractional number nom/denom. This is a subclass of FracVector with the purpose of making it clear when a scalar fracvector is needed/used.

class httk.**MutableFracVector**(*noms*, *denom*)

Same as FracVector, only, this version allow assignment of elements, e.g.,

```
mfracvec[2,7] = 5
```

and, e.g.,

```
mfracvec[:,7] = [1,2,3,4]
```

Other than this, the FracVector methods exist and do the same, i.e., they return *copies* of the fracvector, rather than modifying it.

However, methods have also been added named with set_* prefixes which performs mutating operations, e.g.,

```
A.set_T()
```

replaces A with its own transpose, whereas

```
A.T()
```

just returns a new MutableFracVector that is the transpose of A, leaving A unmodified.

10.5.5 HttkObject

class httk.**HttkObject**

httk.**httk_typed_property**(*t*)

httk.**httk_typed_init**(*t*, ***kargs*)

httk.**httk_typed_property_delayed**(*t*)

httk.**httk_typed_init_delayed**(*t*, ***kargs*)

httk.**HttkPluginWrapper**(*plugin=None*)

httk.**HttkPlugin**(*main_instance*)

httk.**HttkPluginPlaceholder**(*plugininfo=None*)

10.5.6 HttkObject for Projects and Computations

class httk.**Code**(*name*, *version*)
Object for keeping track of httk data about a computer software or script

class httk.**Computation**(*computation_date*, *description*, *code*, *manifest_hash*, *signatures*, *keys*, *relpath*, *project_counter*, *added_date=None*)
Object for keeping track of httk data about a specific computation run

class httk.**Result**(*computation*)
Intended as a base class for results tables for computations

class httk.**ComputationRelated**(*main_computation*, *other_computation*, *relation*)
Object for keeping track of httk data about a specific computation run

class httk.**ComputationProject**(*computation*, *project*)

class httk.**Author**(*last_name*, *given_names*)
Object for keeping track of tags for other objects

class httk.**Reference**(*ref*, *authors=None*, *editors=None*, *journal=None*, *journal_issue=None*, *journal_volume=None*, *page_first=None*, *page_last=None*, *title=None*, *year=None*, *book_publisher=None*, *book_publisher_city=None*, *book_title=None*)
A reference citation

class httk.**Project**(*name*, *description*, *project_key*, *keys*)

class httk.**ProjectRef**(*project*, *reference*)

class httk.**ProjectTag**(*project*, *tag*, *value*)

10.5.7 IOAdapters

class httk.**IoAdapterFileReader**(*f*, *name=None*, *deletefilename=None*, *close=False*)
Io adapter for easy handling of io.

class httk.**IoAdapterFileWriter**(*f*, *name=None*, *close=False*)
Io adapter for access to data as a python file object

class httk.**IoAdapterFileAppender**(*f*, *name=None*)
Io adapter for access to data as a python file object

class httk.**IoAdapterString**(*string=None*, *name=None*)
Universal io adapter, helps handling the passing of filenames, files, and strings to functions that deal with io

class httk.**IoAdapterStringList**(*stringlist*, *name=None*)
Universal io adapter, helps handling the passing of filenames, files, and strings to functions that deal with io

class httk.**IoAdapterStringList**(*stringlist*, *name=None*)
Universal io adapter, helps handling the passing of filenames, files, and strings to functions that deal with io

10.5.8 Full documentation

For full documentation, see *Full httk API documentation*.

10.6 The httk.atomistic package

This page documents the features of the httk.atomistic package most relevant for regular users. For a complete listing of members and subpackages, please refer to the full API documentation instead, *Full httk API documentation*.

10.6.1 Introduction

The httk.atomistic package

Classes and utilities for dealing with high-throughput calculations of atomistic systems.

10.6.2 Atomistic description

class `httk.atomistic.`**`Structure`**(*assignments*, *rc_sites=None*, *rc_cell=None*, *other_reps=None*)

A Structure represents N sites of, e.g., atoms or ions, in any periodic or non-periodic arrangement. The structure object is meant to be immutable and assumes that no internal variables are changed after its creation. All methods that 'changes' the object creates and returns a new, updated, structure object.

This is the general heavy weight structure object. For lightweight structure objects, use UnitcellStructure or RepresentativeStructure.

Naming conventions in httk.atomistic:

Structure cell type abbreviations:

rc = Representative cell: only representative atoms are given inside the conventional cell. they need to be replicated by the symmetry elements.

uc = Unit cell: any (imprecisely defined) unit cell (usually the unit cell used to define the structure if it was not done via a representative cell.) with all atoms inside.

pc = Primitive unit cell: a smallest possible unit cell (the standard one) with all atoms inside.

cc = Conventional unit cell: the high symmetry unit cell (rc) with all atoms inside.

For cells:

cell = an abstract name for any reasonable representation of a 'cell' that defines the basis vectors used for representing the structure. When a 'cell' is returned, it is an object of type Cell

basis = a 3x3 sequence-type with (in rows) the three basis vectors (for a periodic system, defining the unit cell, and defines the unit of repetition for the periodic dimensions)

lengths_and_angles = (a,b,c,alpha,beta,gamma): the basis vector lengths and angles

niggli_matrix = ((v1*v1, v2*v2, v3*v3),(2*v2*v3, 2*v1*v3, 2*v2*v3)) where v1, v2, v3 are the vectors forming the basis

metric = ((v1*v1,v1*v2,v1*v3),(v2*v1,v2*v2,v2*v3),(v3*v1,v3*v2,v3*v3))

For sites:

These following prefixes are used to describe types of site specifications: representative cell/rc = only representative atoms are given, which are then to be repeated by structure symmetry group to give all sites

unit cell/uc = all atoms in unitcell

reduced = coordinates given in cell vectors

cartesian = coordinates given as direct cartesian coordinates

sites = used as an abstract name for any sensible representation of a list of coordinates and a cell, when a 'sites' is returned, it is an object of type Sites

counts = number of atoms of each type (one per entry in assignments)

coordgroups = coordinates represented as a 3-level-list of coordinates, e.g. [[[0,0,0],[0.5,0.5,0.5]],[[0.25,0.25,0.25]]] where level-1 list = groups: one group for each equivalent atom

counts and coords = one list with the number of atoms of each type (one per entry in assignments) and a 2-level list of coordinates.

For assignments of atoms, etc. to sites: assignments = abstract name for any representation of assignment of atoms. When returned, will be object of type Assignment.

atomic_numbers = a sequence of integers for the atomic number of each species

occupations = a sequence where the assignments are *repeated* for each coordinate as needed (prefixed with uc or rc depending on which coordinates)

For cell scaling: scaling = abstract name for any representation of cell scaling

scale = multiply all basis vectors with this number

volume = rescaling the cell such that it takes this volume

For periodicity: periodicity = abstract name of a representation of periodicity

pbc = 'periodic boundary conditions' = sequence of True and False for which basis vectors are periodic / non-periodic

nonperiodic_vecs = integer, number of basis vectors, counted from the first, which are non-periodic

For spacegroup: spacegroup = abstract name for any spacegroup representation. When returned, is of type Spacegroup.

hall_symbol = specifically the hall_symbol string representation of the spacegroup

class httk.atomistic.**Cell**(*basis*, *lattice_system*, *orientation=1*)

Represents a cell (e.g., a unitcell, but also possibly just the basis vectors of a non-periodic system)

(The ability to represent the cell for a non-periodic system is also the reason this class is not called Lattice.)

class httk.atomistic.**UnitcellStructure**(*assignments=None*, *uc_sites=None*, *uc_cell=None*)

A UnitcellStructure represents N sites of, e.g., atoms or ions, in any periodic or non-periodic arrangement. It keeps track of all the copies of the atoms within a unitcell.

The structure object is meant to be immutable and assumes that no internal variables are changed after its creation. All methods that 'changes' the object creates and returns a new, updated, structure object.

Naming conventions in httk.atomistic:

For cells:

cell = an abstract name for any reasonable representation of a 'cell' that defines the basis vectors used for representing the structure. When a 'cell' is returned, it is an object of type Cell

basis = a 3x3 sequence-type with (in rows) the three basis vectors (for a periodic system, defining the unit cell, and defines the unit of repetition for the periodic dimensions)

lengths_and_angles = (a,b,c,alpha,beta,gamma): the basis vector lengths and angles

niggli_matrix = ((v1*v1, v2*v2, v3*v3),(2*v2*v3, 2*v1*v3, 2*v2*v3)) where v1, v2, v3 are the vectors forming the basis

metric = ((v1*v1,v1*v2,v1*v3),(v2*v1,v2*v2,v2*v3),(v3*v1,v3*v2,v3*v3))

For sites:

These following prefixes are used to describe types of site specifications: representative cell/rc = only representative atoms are given, which are then to be repeated by structure symmetry group to give all sites

unit cell/uc = all atoms in unitcell

reduced = coordinates given in cell vectors

cartesian = coordinates given as direct cartesian coordinates

sites = used as an abstract name for any sensible representation of a list of coordinates and a cell, when a 'sites' is returned, it is an object of type Sites

counts = number of atoms of each type (one per entry in assignments)

coordgroups = coordinates represented as a 3-level-list of coordinates, e.g. [[[0,0,0],[0.5,0.5,0.5]],[[0.25,0.25,0.25]]] where level-1 list = groups: one group for each equivalent atom

counts and coords = one list with the number of atoms of each type (one per entry in assignments) and a 2-level list of coordinates.

For assignments of atoms, etc. to sites: assignments = abstract name for any representation of assignment of atoms. When returned, will be object of type Assignment.

atomic_numbers = a sequence of integers for the atomic number of each species

occupations = a sequence where the assignments are *repeated* for each coordinate as needed (prefixed with uc or rc depending on which coordinates)

For cell scaling: scaling = abstract name for any representation of cell scaling

scale = multiply all basis vectors with this number

volume = rescaling the cell such that it takes this volume

For periodicity: periodicity = abstract name of a representation of periodicity

pbc = 'periodic boundary conditions' = sequence of True and False for which basis vectors are periodic / non-periodic

nonperiodic_vecs = integer, number of basis vectors, counted from the first, which are non-periodic

For spacegroup: spacegroup = abstract name for any spacegroup representation. When returned, is of type Spacegroup.

hall_symbol = specifically the hall_symbol string representation of the spacegroup

class httk.atomistic.**RepresentativeSites**(*reduced_coordgroups=None*, *cartesian_coordgroups=None*, *reduced_coords=None*, *cartesian_coords=None*, *counts=None*, *hall_symbol=None*, *pbc=None*, *wyckoff_symbols=None*, *multiplicities=None*)

Represents any collection of sites in a unitcell

class httk.atomistic.**UnitcellSites**(*reduced_coordgroups=None*, *reduced_coords=None*, *counts=None*, *hall_symbol='P 1'*, *pbc=None*)

Represents any collection of sites in a unitcell

class httk.atomistic.**Assignments**(*siteassignments*, *extensions=[]*)

Represents a possible vector of assignments

class httk.atomistic.**Compound**(*element_wyckoff_sequence*, *formula*, *spacegroup_number*, *extended*, *extensions*, *wyckoff_sequence*, *anonymous_wyckoff_sequence*, *anonymous_formula*, *formula_symbols*, *formula_counts*, *pbc*)

class httk.atomistic.**CompoundStructure**(*compound*, *structure*)

class httk.atomistic.**StructurePhaseDiagram**(*structures*, *energies*, *hull_indices*, *competing_indices*, *hull_competing_indices*, *hull_distances*, *coord_system*, *phase_lines*)

Represents a phase diagram of structures

10.7 Publications

Publications using, or otherwise relating, to the httk framework

10.8 *httk* Runmanager Details

The httk 'taskmanager toolset' is centered around the taskmanager.sh program. This program is responsible for handling a large set of 'tasks' you want to execute on a computer cluster. It can distribute resources between your runs, and re-start them when they break due to, e.g., a computer node breaks, or your job is stopped due to running out of allocated time, etc.

The general philosophy is that 'taskmanager.sh' handles all the tricky parts with overseeing your runs, keeping track of which ones are in which states, and can even restart them automatically when needed. The taskmanager.sh is, in a way, a "second layer of queue system" for your runs.

taskmanager.sh is started in a 'task directory'. It looks in this directory and descends into subdirectories, looking for anything that is setup as a task that is waiting to be run, and then runs it. You can have more than one taskmanager.sh run in the same task directory, taskmanager.sh is very carefully programmed to avoid inference between several instances of itself.

The taskmanager.sh runs until there is nothing more to do in the task directory, at which points it terminates. This is what you typically want if you submit taskmanager.sh to run on supercomputer cluster nodes. Alternatively you can start it with 'taskmanager.sh –daemon", in which case it keeps running forever, looking for new tasks to arrive. You could, e.g., setup a taskmanager daemon running on your own personal computer.

10.8.1 Anatomy of a task

There are a number of conventions you have to follow when setting up a task to be run by taskmanager.sh.

A task is stored in its own directory. The directory name has a very specific format:

```
ht.task.<computer>.<taskid>.<step>.<restarts>.<owner>.<prio>.<status>
```

where: <computer> this is the computer that the task has been assigned to, or 'unassigned'. <prio> is a priority number 1-5. Use '3' as default. <taskid> is a "name" for the task <step> is the present 'step' that a multi-step task is on <restarts> is a counter that keeps track on how many times the task has been restarted, when created should be 0 <owner> 'unclaimed' when created, which is changed into a code belonging to a running taskmanager that presently is handling the task. <status> is one of:

- waitstart: the task is waiting to be started for the first time
- running: the task is currently being executed

- waitstep: the task is partially completed and waits for the next step
- waitsubtasks: the task has split into a number of subtasks and is waiting for them to complete
- finished: the task has successfully run to completion
- broken: the task has returned an error code that specifies that it wants to be set aside as 'broken'.
- stopped: the taskmanager have stopped the job for some reason (timeout, too many restarts, etc.)

The primary component of a task is a "runscript" or a "runprogram" (you can use any language to write these) that is responsible for executing your computational task. The task directory should contain this runscript. It can have either one of two names:

- ht_run: A 'simplified' run script that is meant for simple jobs. "Just run this". If the run breaks (e.g. is stopped by the computer cluster), it will simply be restarted the next time (you are responsible for necessary cleanup).
- ht_steps: Step-scripts allows for more functionality, most importantly, a run can be executed in a series of steps, and re-start is done from the last completed step rather than as a complete do-over.

10.8.2 The ht.parameters file

[*IMPORTANT:* This section describes functionality not yet fully implemented. Presently ht.taskmanger starts all tasks. To handle resources, you presently need to setup e.g. a single cluster as different 'computers']

The run directory may contain a file ht.parameters that, in that case, is consulted by taskmanager.sh before executing the run. The file should be formatted as rows of 'parameter=value'.

Relevant parameters are: 'cores=X' : The task needs to run at at least X cores.

'nodes=X' : The task needs to run at at least X computer nodes.

'memory=X': The task needs at least X amount of memory.

'restart=false': Never restart the run, always re-init it from scratch if possible (if not, set it in a 'broken' state).

If the requirements cannot be fulfilled (at a given time) the process is skipped and taskmanager.sh looks for another process (possibly of lower priority)

Note: taskmanager.sh does not at this time implement a fancy resource management algorithm, but rather just uses a 'greedy' algorithm where it tries to start jobs in priority order. A high-priority job with harsh resource requirements (e.g., many nodes) may thus be starved by a massive amount of small low priority jobs. If this is a problem, you will have to setup a separate 'computer' for jobs that would otherwise starve.

10.8.3 Simplified 'ht_run' runscript

When your 'ht_run' is executed, your current working directory is your task directory. The script gets called with one command line parameter, the name of the <step> in the task directory name. The runscript should simply execute your run.

IMPORTANT: In case your run gets stopped (e.g., by the computer cluster because your job runs out of time, or the computer node it is running on crashes), it needs to handle being re-started with no ill effects, i.e., 'ht_run' will get executed again in an 'unclean' directory. If this is not possible, set 'restart=false' in the ht.parameters file. But note, the latter means your run will end up in a 'broken' state if it needs to be restarted. This is a bad idea for real high-throughput jobs. In this situation, you are strongly recommended to use a ht_steps script instead. (see below)

See APPENDIX A.1 below for an outline of how taskmanager.sh actually process a ht_run-type task. This may be very helpful to understand what actually happens.

10.8.4 The more advanced 'ht_steps' runscript

When your 'ht_steps' is executed, your current working directory is an empty *subdirectory* of your task directory named 'ht.run.<date>'. You should access files in your task directory simply by '../filename', etc. Your 'ht_steps' script is supposed to setup the run in this directory by copying or use symbolic linking ('ln -s') of the appropriate files from your run directory. You should then execute your run, and end your run script in a normal way.

You are 'forced' into using a subdirectory this way rather than simply executing your run in the run directory itself. The motivation for this is to unify task handling for restarts, etc.

When a 'ht_steps' runscript is executed it gets a single parameter set to the <step> part of the task directory name. When it finishes, it should first write a file 'ht.status' in the task directory that contains a simple string naming its next 'step', and then it should return with a specific exit code:

- exit code 2: Waiting for next task
- exit code 3: Subtasks have been created, do not restart again until all are completed.
- exit code 4: Restart me completely
- exit code 5: the run is in a broken state, mark it broken and leave it.

Usually you don't need to think about this, just use the provided httk task api routines for the language being used, and exit the task with, e.g. 'TASK_NEXT' (in bash) or similar. See the corresponding httk task api instructions for more details.

IMPORTANT: a ht_steps script *must be capable of being restarted at the same step*. I.e., if it is started on a 'relax' step, the job may be stopped (running out of runtime) at any time. It may then be restarted again on the same 'relax' step in which case it needs to be able to 're-init' the job and restart (or just continue it, if applicable). The script needs to be written such that it can handle this transparently. For example, some electronic structure software overwrites input files (e.g., VASP overwrites the CHGCAR which sometimes is used as an input file for a run). In this case, one *needs* to write ht_steps to keep around a copy CHGCAR.before so that it can be used to re-initalize the file as the job is restarted. Alternatively, a task may return '4' to indicate that it is in such a broken state that it has to be completely restarted. You are recommended to read the code of some tasks provided along with httk to learn how tasks should be written.

See APPENDIX A.2 for an outline of how taskmanager.sh actually process a ht_steps-type task. This may be very helpful to understand what actually happens.

10.8.5 'ht_steps' subtasks

In a ht_steps script one can create 'subtasks'. This is done is simply by the runscript generating subdirectories with appropriate naming (see section 6.2 Anatomy of a task above.) Note that as soon as the directories fulfill this naming scheme, the run may be executed by another taskmanager.sh process, so one must follow the following process:

1. Create a directory called ht.tmp.task.(something)
2. Populate the directory with necessary files to run as a subtask. (Primarily, a ht_run, or ht_steps)
3. Only when the subtask is ready, *mv ht.tmp.task.<something> ht.task.<something>*

Using specifically the 'ht.tmp.' prefix for your temporary directories has the advantage that such directories are automatically removed when runs are restarted, which avoids leaving half-complete subtask directories in case your job is stopped while creating subtasks.

When a ht_steps script exits with exit code 3, it will be put on hold until all subtasks that reside inside its subdirectories have run to completion. Once this has happened, it will be restarted as usual with 'ht_steps <step>'.

Note that subtasks are handled exactly like regular tasks, so they can themselves create substasks, and so on.

A couple of neat tricks:

- Use a symbolic link (‘ln -s’) to make your subtasks use the same ht_steps script as the topmost task. This way all the run functionality can conveniently be kept inside one and the same script/program.
- Even if your main job uses a ‘ht_steps’ runscript, your subtasks can use ‘ht_run’ scripts to decrease the overhead. (You can even make a symbolic link from the subruns ‘ht_runs’ to your main ‘ht_steps’.)

10.8.6 single_job_taskmanager.sh

There is a ‘light’ version of the taskmanager named single_job_taskmanager.sh that may be helpful in a few situations, e.g.,

- You are in the process of developing a run script and “just want to run through this task” to debug it, with all output in the console.
- You don’t care for the parallelism, resource handling, and restart/continuation capability of the full httk taskmanager, and just want something to put in your cluster submit script that will simply run one task to completion with a minimum of hassle.

You start single_job_taskmanager.sh with the task directory as the current working directory, and it will run that one task to completion. It never ‘restarts’ a task. It thus always create a new ‘run.<date and timestamp>’ and run the task in this directory. It will not rename the task directory itself, and there is no need to follow the naming convention of the task directory at all. It ignores all ‘ht.parameters’ files. Other than this, it mimics the exact functionality of the full task manager both for ‘ht_run’ and ‘ht_steps’ type runscripts.

10.8.7 taskmanager.sh prioritization

The priority order of waiting tasks is the following:

- First it handles tasks of priority 1, then 2, . . . , and last 5.
- It first prioritize finishing tasks that have been started before starting new ones.
- It always runs subtasks ‘depth first’.

10.8.8 Provided helper scripts

In the httk directory, under Execution/tasks-templates/* you can find a number of provided scripts that can be used as-is for your own runs. Reading and understanding them may help you develop / adapt them to your own needs.

10.8.9 Writing runscripts in python

The present aid in the python library for run scripts is limited to use of ready-made templates under Execution/tasks-templates/ Please consult the tutorial Step6.

It is the idea that the httk library will be extended with helper functionality for writing your own runscripts in python. One of the leading design ideas is to make it possible to write scripts that describes how to do a calculation in a *code-independent-way*. I.e., relying on higher-order routines of type ‘converge’ and ‘relax’ which then call out to a specific code.

10.8.10 Writing runsscripts in bash

httk presently come with a helper library of routines for writing runscripts in bash.

There is a general tasks API for bash in: Execution/tasks/ht_tasks_api.sh

and specifically a set of helper routines for runs with the electronic structure software VASP in:
Execution/tasks/vasp/vasptools.sh

10.8.11 APPENDIX A: taskmanager.sh process outlines

The taskmanager.sh process with a *ht_run* runscript

Here is an outline of the process as taskmanager.sh executes a ht_run script:

1. taskmanager.sh looks in the task directory and finds a **.waitstart* directory
2. taskmanager.sh 'adopts' this task by renaming the directory so that it includes a taskmanager-id (an id that pertains to this runmanager.sh instance) This 'locks' the run from being tampered with by other runmanagers.
3. taskmanager.sh executes the ht_run script in this directory.
4. the ht_run script does what it needs to do and simply finishes as usual.
5. taskmanager.sh renames the task directory to both remove the taskmanager-id and so that it now ends with a '.finished' suffix.

IF the taskmanager and the job is stopped at any of the points 3-5 (e.g., the cluster runtime ends and stops the processes), you can simply submit another job with a new taskmanager.sh. This is an outline of what happens then:

1. taskmanager.sh notices a directory named 'ht.task.*.running' that has a filesystem 'ctime' that is > 10 minutes old. This marks an abandonded run, because an alive taskmanager.sh makes sure to update ctime periodically on any ongoing runs.
2. taskmanager.sh 'adopts' this task by renaming the directory so that it removes the old taskmanager-id and replaces it with that of the present instance.
3. taskmanager.sh simply restarts the ht_run scripts in this directory (expecting it to know what to do with regards to cleanup etc.)
4. Everything continues from point #4 and onwards in the regular outline above.

The taskmanager.sh process with a ht_steps runscript

The process outlined in 6.3 changes when a tasks_steps script is used. Steps 1-2 are the same, after that, this happens:

3. taskmanager.sh creates a subdirectory in the task directory named similar to 'ht.run.2014-05-05_12_15_36' (i.e., ht.run.<date and time-stamp>) and makes this directory the current working directory.
4. taskmanager.sh executes 'ht_steps <step>' where step is the name of the .<step>. part of the task directory name.
5. ht_steps executes the apropriate part of the run, writes the ht.status file, and exits with an apropriate exit status.
6. The directory is renamed to remove the taskmanager-id and, depending on the exit status, is made to end with any one of '.finished', 'waitstep' or , 'waitsubtasks'. If '.finished', then this job is complete and will be left alone. Otherwise, continue below.
7. taskmananger.sh goes back to scanning the task directory for runs, but will eventually find this job again.

[If it ends in .waitsubtasks]

8a. subtasks are handled by taskmanager.sh just like any normal tasks. The .waitsubtasks ht_step script itself is not touched until all subtasks in its subdirectories are in a finished state. When this happens, it is restarted following point #4 and onwards.

[If it ends in .waitstep]

8b. taskmananger.sh restart the run following point #4 and onwards.

IF the taskmanager and the job is stopped at any of the points 3-6 (e.g., the cluster runtime ends and stops the processes), you can simply submit another job with a new taskmanager.sh. This is an outline of what happens then:

1. taskmanager.sh notices a directory named 'ht.task.*.running' that has a filesystem 'ctime' that is > 10 minutes old. This marks an abandonded run, because an alive taskmanager.sh makes sure to update ctime periodically on any ongoing runs.
2. taskmanager.sh 'adopts' this task by renaming the directory so that it removes the old taskmanager-id and replaces it with that of the present instance.
3. taskmanager.sh now just continues from point #4 and onwards in the regular outline.

The exception to #3 is if the ht.parameters file (see below) contains 'restart=false'. In that case, the old 'run.*' directory will be removed, and taskmanager.sh instead restarts from #3 in the regular outline.

10.9 *httk* Users' Guide

10.9.1 Introduction

The High-Throughput Toolkit (httk) is a toolkit for preparing and running calculations, analyzing the results, and store them in a global and/or in a personalized database. The word 'high-throughput' refers to the practice of executing a vast number of computational tasks on a supercomputer cluster, in which case proper automatization of all steps is critically important. Httk is presently targeted at atomistic calculations in materials science and electronic structure, but aims to be extended into a library useful also outside those areas.

10.9.2 Importing the httk python library into your program

The easiest way to import the python library if you do atomistic calculations is:

```
from httk import *
from httk.atomistic import *
```

This imports some very often used identifiers into the namespace of your program, e.g., Structure for atomic structures. If you want to avoid wild imports (*from X import* *) you can of course instead do:

```
import httk
import httk.atomistic
```

(Note the need to separately import the atomistic sub-library; it is not imported automatically by import.httk)

To avoid dependences on libraries that you may not have installed, httk implements somewhat unusual 'plugin'-type extensions to its core classes. For example, you can enable visualization of atomic structures, which requires jmol to be installed, by the following:

```
from httk import *
from httk.atomistic import *
import httk.atomistic.vis
```

This adds new visualization calls to the Structure class which can be called, e.g., as:

```
mystructure.vis.show()
```

(Note: if you forget to do 'import httk.atomistic.vis', httk informs you about the need to add this import.)

10.9.3 Example programs

It may be easiest to learn the use of httk by example. There are three such resources available. The presentation httk_overview.pdf shows working code snippets that can be copy+pasted. There are short examples under Examples. Then there is a step-by-step tutorial under Tutorial/ that is intended to showcase the httk features in a natural progressing order.

10.9.4 Interfacing with other software

Interfacing with python libraries

A common need is to use functionality provided by other python libraries outside the standard libraries. Httk tries to help with this. It provides 'glue' modules that lets you import exactly the version you want.

To use the ase python library (Atomic Simulation Environment) together with httk, you typically want to do:

```
import httk.external.ase_glue
import ase
```

The first line imports the httk 'glue' module. It includes helper functionality that makes httk and ase work together. But, it also sets up your python environment so that at the next line 'import ase' actually imports the version of ase that you have configured httk to use. This can, for example, be a specific version in your home directory (which can help avoid an older version provided system-wide on the computational cluster you are using). All you need to do is edit httk.cfg in the main httk directory and set the path to where you have placed the ase library (e.g., in your home directory).

Interact with other programs

Similar to the interface to other python libraries, httk helps you call other (non-python) software packages.

For example, the following code:

```
import httk.external.jmol
```

gives you access to routines for running and interacting with jmol.

Note that subpackages of httk.external raise an exception if you try to import them and the relevant software is missing.

Interface packages

httk also provides 'light' versions of its interface to other software under httk.iface.*. These packages DO NOT require the corresponding software to be installed. This usually includes things such as writing correctly formatted files, etc.

10.9.5 More details on the httk python library

This section covers some design decisions of httk that it may be useful to take note of.

Creating new httk objects

The python default constructor (the '__init__' constructor) that is called when simply doing:

```
struct = Structure(arg1, arg2, ...)
```

should almost *never* be used with httk objects, for several reasons. Perhaps the most important is that it is going to change between version of httk (for more explanation, see the developers' guide).

Instead, almost all httk objects provide a classmethod named **.create* for this purpose instead. I.e.,

```
struct = Structure.create(arg1, arg2, ...)
```

A note about object mutation

Most httk objects assume they stay unaltered after creation (unless clearly spelled out, e.g., 'MutableFracVector'). Hence, methods 'altering' an object normally return a *new copy* of the object with the alterations made. This comes with a number of benefits:

- They can be used as keys in dictionaries
- Less risk for bugs as one part of code alters an object that happens to also be stored and used somewhere else.
- The API becomes more clear, you do not have to wonder if the object itself may be altered by calling a method (it never is.)

It also comes with a drawback

- Code making, say, a series of alterations of an object may becomes more bulky to write.

It is the intention to provide mutable versions where this drawback is of significance. Right now, this more or less only applies to the existence of a MutableFracVector vs the regular FracVector.

Object conversion with the 'use' method

Almost all httk classes contains a **.use()* method for helping with object type conversion. Lets say that you get a Structure object 'structure' which represents structure data fetched out of the database, but you want to have a UnitcellStructure instead, simply do this:

```
unitcellstruct = UnitcellStructure.use(structure)
```

I/O in httk

All I/O in the httk library uses our own framework of IOAdapters classes. This is usually not something you need to worry about; any routine that takes as a parameter an "IOAdapter" 'ioa' will accept a filename or any form of python streaming object in its place. (You may want to check the IOAdapter chapter of the developers' guide to see how this is done in practice, as the IOAdapters may be helpful also in your own routines.)

10.9.6 The httk taskmanager toolset

Apart from the python library, httk also comprises a toolset for executing computational tasks on computer clusters. To avoid issues with incompatible version, this part of httk is mostly written in bash rather than python. If things are working as they should, this is not something you should need to worry about, you can still script your runs in python, or any other language you prefer.

Setting up a computational 'project'

You should first setup a 'top' working directory for your project. Use 'cd' to go to this directory and then run:

```
httk-project-setup project_name
```

Configuring 'computers'

Supercomputer clusters, as well as other computers that you are going to execute runs on can now be setup by the command httk-computer-setup this allows you to configure settings for how to transport runs to this computer and run them there.

After you have configured the computer you also need to run:

```
httk-computer-install
```

to copy necessary httk files to this computer and "prepare it" for executing runs.

Sending tasks to a computer and running them

For this to work you need to have created batch tasks on the right format. For this, please consider closely Step6 of the httk tutorial.

Once you have a directory with runs, execute:

```
httk-tasks-send-to-computer <computer name>
```

and the runs will be copied over. They will not yet be started.

All execution of tasks is done via the taskmanager.sh process, which now needs to be started on the computer. Run:

```
httk-tasks-start-taskmanager <computer name>
```

and it will start up.

You can monitor the status of your compute runs by:

```
httk-tasks-status <computer name>
```

And as soon as one or more of the runs have finished, you can fetch them back with:

```
httk-tasks-receive-from-computer <computer name>
```

This concludes what you need for 'simple' use of the task system. However, for advanced use, you will need to better understand precisely how the taskmanager.sh process operates. This information is present in a separate text: RUNMANAGER_DETAILS.txt.

If you want; how to submit your results to a public database

httk includes tools that, if you want to, makes it easy to submit a project directory so that your data can be made available and searchable in a public database. The normal case would be the Open Materials Database (http://openmaterialsdb.se), run by the same people involved with the httk framework.

First, if you have not yet setup a project directory, do so. I.e., collect all the files that you wish to be part of the submission and do:

```
httk-project-setup project_name
```

This creates a subdirectory *ht_project* in this directory. You must now use a text editor and edit three files in this directory:

1. Edit *ht_project/config* and set *description=A good description of your poject*.
2. Edit *ht_project/license* and write clearly what license you place the data under. For submissions to the Open Materials Database we normally ask for the data to be placed either under a creative commons attribution license, or the public domain. (This can be negotiated, contact the omdb team at contact [at] openmaterialsdb.se.) See http://openmaterialsdb.se/contributorinfo.html for the latest info.
3. Optional: edit *ht_project/references* and insert, one per line, any citations to papers, etc., that you want to associate with this project.

Once your project is setup correctly, you simply have to have the project directory as your current working directory and execute:

```
httk-project-submit
```

(or httk-project-submit <website> if you want to submit somewhere else than the *Open Materials Database*.)

After a series of question and a cryptographic signing of your project files, your files will be submitted to the database.

Note that submitted results are not directly and automatically processed. There is a certain level of manual examination by us to make sure the upload makes sense before we add it to the database.

Furthermore, you can edit the file ht.project/references to add or remove publications even after your result has been submitted. To re-submit updated references, issue the command:

```
httk-project-submit-update-references
```

Finally, should you change your mind about the data being published, you can issue the command:

```
httk-project-submit-withdraw
```

Which will lead to the result eventually being pulled from our data (however, also here some manual work is involved, so the result will not be intimidate.)

10.10 *httk* Contributors

Programming:

- Rickard Armiento, Linköping University, Sweden (ricard [at] ifm.liu.se)
- Christopher Tholander, Linköping University, Sweden.

Some parts of *httk* related to reading structues are heavily inspired by corresponding code in cif2cell by Torbjörn Björkman (Aalto University, Finland).

Database and API design:

- Rickard Armiento
- Peter Steneteg
- Igor Mogyasinz

10.10.1 Acknowledgements

***httk* has kindly been funded in part by:**

- The Swedish Research Council (VR) Grant No. 621-2011-4249.
- The Linnaeus Environment at Linkoping on Nanoscale Functional Materials (LiLi-NFM) funded by The Swedish Research Council.

Python Module Index

h

httk, 40
httk.analysis, 51
httk.analysis.matsci, 51
httk.analysis.matsci.phasediagram, 51
httk.analysis.matsci.vis, 51
httk.analysis.matsci.vis.phasediagramvisualizerplugin, 51
httk.atomistic, 52
httk.atomistic.assignment, 68
httk.atomistic.assignments, 69
httk.atomistic.atomisticio, 65
httk.atomistic.atomisticio.structure_cif_io, 65
httk.atomistic.atomisticio.structure_io, 65
httk.atomistic.atomisticio.structureioplugin, 65
httk.atomistic.cell, 69
httk.atomistic.cellshape, 70
httk.atomistic.cellutils, 71
httk.atomistic.cli, 73
httk.atomistic.compound, 73
httk.atomistic.data, 66
httk.atomistic.data.periodictable, 66
httk.atomistic.data.spacegroups, 66
httk.atomistic.formulautils, 74
httk.atomistic.representativesites, 74
httk.atomistic.representativestructure, 75
httk.atomistic.results, 66
httk.atomistic.results.relaxedcellresult, 67
httk.atomistic.results.totalenergyresult, 67
httk.atomistic.siteassignment, 77
httk.atomistic.sites, 77
httk.atomistic.sitesutils, 78
httk.atomistic.spacegroup, 78
httk.atomistic.spacegrouputils, 79
httk.atomistic.structure, 80
httk.atomistic.structurephasediagram, 86
httk.atomistic.structureutils, 86
httk.atomistic.supercellutils, 88
httk.atomistic.unitcellsites, 89
httk.atomistic.unitcellstructure, 89
httk.atomistic.vis, 67
httk.atomistic.vis.asestructurevisualizer, 67
httk.atomistic.vis.jmolstructurevisualizer, 67
httk.atomistic.vis.structurephasediagramvisualizerp 68
httk.atomistic.vis.structurevisualizerplugin, 68
httk.cli, 138
httk.config, 92
httk.config.config, 92
httk.core, 92
httk.core.basic, 107
httk.core.citation, 107
httk.core.code, 108
httk.core.computation, 108
httk.core.console, 109
httk.core.crypto, 109
httk.core.ed25519, 110
httk.core.geometry, 110
httk.core.httkobject, 111
httk.core.ioadapters, 112
httk.core.miniparser, 113
httk.core.project, 117
httk.core.reference, 118
httk.core.signature, 118
httk.core.template, 118
httk.core.vectors, 92
httk.core.vectors.fracmath, 93
httk.core.vectors.fracvector, 94
httk.core.vectors.mutablefracvector, 100

httk.core.vectors.vector, 102
httk.core.vectors.vectormath, 103
httk.db, 119
httk.db.backend, 119
httk.db.backend.sqlite, 119
httk.db.filteredcollection, 121
httk.db.httkobjdbplugin, 124
httk.db.storable, 124
httk.db.store, 120
httk.db.store.dictstore, 120
httk.db.store.sqlstore, 121
httk.external, 125
httk.external.aflow_ext, 125
httk.external.ase_glue, 125
httk.external.cif2cell_ext, 125
httk.external.command, 126
httk.external.gulp_ext, 126
httk.external.isotropy_ext, 126
httk.external.jmol, 126
httk.external.platon_ext, 126
httk.external.pymatgen_glue, 127
httk.external.pyspglib_ext, 127
httk.external.subimport, 127
httk.graphics, 127
httk.graphics.matplotlib, 127
httk.httkio, 127
httk.httkio.cif, 128
httk.httkio.load, 129
httk.httkio.save, 129
httk.httkweb, 129
httk.httkweb.app_curses, 129
httk.httkweb.app_qt5, 129
httk.httkweb.functionhandler_httk, 130
httk.httkweb.helpers, 130
httk.httkweb.publish, 130
httk.httkweb.render_httk, 130
httk.httkweb.render_rst, 131
httk.httkweb.serve, 131
httk.httkweb.templateengine_httk, 131
httk.httkweb.templateengine_templator, 131
httk.httkweb.webgenerator, 132
httk.iface, 132
httk.iface.ase_if, 132
httk.iface.cif2cell_if, 132
httk.iface.gulp_if, 133
httk.iface.isotropy_if, 133
httk.iface.jmol_if, 133
httk.iface.platon_if, 133
httk.iface.spglib_if, 137
httk.iface.vasp_if, 137
httk.task, 138
httk.task.reader, 138
httk.task.taskmgr, 138
httk.versioning, 138

Index

A

abstract_formula() (*in module httk.atomistic.structureutils*), 86
abstract_symbol() (*in module httk.atomistic.sitesutils*), 78
abstract_symbol() (*in module httk.atomistic.structureutils*), 86
acos() (*httk.core.vectors.fracvector.FracVector method*), 95
acos() (*httk.FracVector method*), 43
acos() (*in module httk.core.vectors.vectormath*), 103
acosh() (*in module httk.core.vectors.vectormath*), 103
add() (*httk.db.filteredcollection.FCMultiDict method*), 122
add() (*httk.db.filteredcollection.FilteredCollection method*), 123
add_all() (*httk.db.filteredcollection.FilteredCollection method*), 123
add_ext_citation() (*in module httk.core.citation*), 108
add_name() (*httk.atomistic.Compound method*), 59
add_name() (*httk.atomistic.compound.Compound method*), 73
add_names() (*httk.atomistic.Compound method*), 59
add_names() (*httk.atomistic.compound.Compound method*), 73
add_phase() (*httk.analysis.matsci.phasediagram.PhaseDiagram method*), 51
add_project() (*httk.Computation method*), 41
add_project() (*httk.core.computation.Computation method*), 108
add_projects() (*httk.Computation method*), 41
add_projects() (*httk.core.computation.Computation method*), 108
add_ref() (*httk.atomistic.Compound method*), 59
add_ref() (*httk.atomistic.compound.Compound method*), 73
add_ref() (*httk.atomistic.Structure method*), 53
add_ref() (*httk.atomistic.structure.Structure method*), 81
add_ref() (*httk.Code method*), 41
add_ref() (*httk.Computation method*), 41
add_ref() (*httk.core.code.Code method*), 108
add_ref() (*httk.core.computation.Computation method*), 108
add_ref() (*httk.core.project.Project method*), 117
add_ref() (*httk.Project method*), 42
add_refs() (*httk.atomistic.Compound method*), 60
add_refs() (*httk.atomistic.compound.Compound method*), 73
add_refs() (*httk.atomistic.Structure method*), 53
add_refs() (*httk.atomistic.structure.Structure method*), 81
add_refs() (*httk.Code method*), 41
add_refs() (*httk.Computation method*), 41
add_refs() (*httk.core.code.Code method*), 108
add_refs() (*httk.core.computation.Computation method*), 108
add_refs() (*httk.core.project.Project method*), 117
add_refs() (*httk.Project method*), 42
add_sort() (*httk.db.filteredcollection.FilteredCollection method*), 123
add_src_citation() (*in module httk.core.citation*), 108
add_tag() (*httk.atomistic.Compound method*), 60
add_tag() (*httk.atomistic.compound.Compound method*), 73
add_tag() (*httk.atomistic.Structure method*), 53
add_tag() (*httk.atomistic.structure.Structure method*), 81
add_tag() (*httk.Code method*), 41
add_tag() (*httk.Computation method*), 41
add_tag() (*httk.core.code.Code method*), 108
add_tag() (*httk.core.computation.Computation method*), 108
add_tag() (*httk.core.project.Project method*), 117
add_tag() (*httk.Project method*), 42
add_tags() (*httk.atomistic.Compound method*), 60
add_tags() (*httk.atomistic.compound.Compound*

method), 73
add_tags() (*httk.atomistic.Structure method*), 53
add_tags() (*httk.atomistic.structure.Structure method*), 81
add_tags() (*httk.Code method*), 41
add_tags() (*httk.Computation method*), 41
add_tags() (*httk.core.code.Code method*), 108
add_tags() (*httk.core.computation.Computation method*), 109
add_tags() (*httk.core.project.Project method*), 117
add_tags() (*httk.Project method*), 43
added_date (*httk.Computation attribute*), 42
added_date (*httk.core.computation.Computation attribute*), 109
addsym() (*in module httk.external.platon_ext*), 126
addsym_spacegroup() (*in module httk.external.platon_ext*), 126
adornment_chars (*httk.httkweb.render_httk.RenderHttk attribute*), 130
aflow() (*in module httk.external.aflow_ext*), 125
alter() (*httk.db.backend.sqlite.Sqlite method*), 119
analysis() (*in module httk.external.pyspglib_ext*), 127
angles_to_cosangles() (*in module httk.atomistic.cellutils*), 71
anonymous_formula (*httk.atomistic.Compound attribute*), 60
anonymous_formula (*httk.atomistic.compound.Compound attribute*), 73
anonymous_formula (*httk.atomistic.sites.Sites attribute*), 77
anonymous_formula (*httk.atomistic.Structure attribute*), 53
anonymous_formula (*httk.atomistic.structure.Structure attribute*), 81
anonymous_formula() (*in module httk.atomistic.sitesutils*), 78
anonymous_symbol_to_int() (*in module httk.core.basic*), 107
anonymous_wyckoff_sequence (*httk.atomistic.Compound attribute*), 60
anonymous_wyckoff_sequence (*httk.atomistic.compound.Compound attribute*), 73
anonymous_wyckoff_sequence (*httk.atomistic.RepresentativeSites attribute*), 58
anonymous_wyckoff_sequence (*httk.atomistic.representativesites.RepresentativeSites attribute*), 74
anonymous_wyckoff_sequence (*httk.atomistic.Structure attribute*), 53
anonymous_wyckoff_sequence (*httk.atomistic.structure.Structure attribute*), 81
any_to_fraction() (*in module httk.core.vectors.fracmath*), 93
apply() (*httk.httkweb.templateengine_httk.TemplateEngineHttk method*), 131
apply() (*httk.httkweb.templateengine_templator.TemplateEngineTemplat method*), 131
apply_template() (*in module httk.core.template*), 118
apply_templates() (*in module httk.core.template*), 119
argmax() (*httk.core.vectors.fracvector.FracVector method*), 95
argmax() (*httk.FracVector method*), 43
argmin() (*httk.core.vectors.fracvector.FracVector method*), 95
argmin() (*httk.FracVector method*), 43
ase_atoms_to_structure() (*in module httk.external.ase_glue*), 125
ase_read_structure() (*in module httk.external.ase_glue*), 125
ase_write_struct() (*in module httk.external.ase_glue*), 125
AseStructureVisualizer (*class in httk.atomistic.vis.asestructurevisualizer*), 67
asin() (*httk.core.vectors.fracvector.FracVector method*), 95
asin() (*httk.FracVector method*), 43
asin() (*in module httk.core.vectors.vectormath*), 103
asinh() (*in module httk.core.vectors.vectormath*), 103
Assignment (*class in httk.atomistic.assignment*), 68
Assignments (*class in httk.atomistic*), 59
Assignments (*class in httk.atomistic.assignments*), 69
atan() (*in module httk.core.vectors.vectormath*), 103
atan2() (*in module httk.core.vectors.vectormath*), 104
atanh() (*in module httk.core.vectors.vectormath*), 104
atomic_number (*httk.atomistic.siteassignment.SiteAssignment attribute*), 77
atomic_number() (*in module httk.atomistic.data.periodictable*), 66
atomic_number_isotope() (*in module httk.atomistic.data.periodictable*), 66
atomic_numbers (*httk.atomistic.Assignments attribute*), 59
atomic_numbers (*httk.atomistic.assignments.Assignments attribute*), 69
atomic_numbers (*httk.atomistic.siteassignment.SiteAssignment attribute*), 77
atomic_symbol() (*in module httk.atomistic.data.periodictable*), 66
Author (*class in httk*), 42
Author (*class in httk.core.reference*), 118

B

basics (*httk.db.store.dictstore.DictStore attribute*), 120
basics (*httk.db.store.sqlstore.SqlStore attribute*), 121
basis (*httk.atomistic.cellshape.CellShape attribute*), 70
basis_determinant() (*in module httk.atomistic.cellutils*), 71
basis_determinant() (*in module httk.atomistic.structureutils*), 86
basis_scale_to_vol() (*in module httk.atomistic.structureutils*), 86
basis_to_niggli() (*in module httk.atomistic.structureutils*), 86
basis_to_niggli_and_orientation() (*in module httk.atomistic.cellutils*), 71
basis_vol_to_scale() (*in module httk.atomistic.structureutils*), 86
best_rational_in_interval() (*in module httk.core.vectors.fracmath*), 93
BinaryBooleanOp (*class in httk.db.filteredcollection*), 121
BinaryComparison (*class in httk.db.filteredcollection*), 121
BinaryOp (*class in httk.db.filteredcollection*), 121
bit() (*in module httk.core.ed25519*), 110
bonds() (*httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method*), 67
breath_first_idxs() (*in module httk.core.basic*), 107
build_cubic_supercell() (*in module httk.atomistic.supercellutils*), 88
build_ls() (*in module httk.core.miniparser*), 116
build_orthogonal_supercell() (*in module httk.atomistic.supercellutils*), 88
build_supercell_old() (*in module httk.atomistic.supercellutils*), 88
bullet_item_markers (*httk.httkweb.render_httk.RenderHttk attribute*), 130

C

calculate_kpoints() (*in module httk.iface.vasp_if*), 137
cartesian_to_reduced() (*in module httk.atomistic.structureutils*), 86
cc (*httk.atomistic.Structure attribute*), 53
cc (*httk.atomistic.structure.Structure attribute*), 81
cc_formula_parts (*httk.atomistic.Structure attribute*), 53
cc_formula_parts (*httk.atomistic.structure.Structure attribute*), 81
ceil() (*httk.core.vectors.fracvector.FracVector method*), 95
ceil() (*httk.FracVector method*), 43
ceil() (*in module httk.core.vectors.vectormath*), 104
Cell (*class in httk.atomistic*), 57
Cell (*class in httk.atomistic.cell*), 69
cell_to_basis() (*in module httk.atomistic.cellutils*), 71
CellShape (*class in httk.atomistic.cellshape*), 70
cerr() (*in module httk*), 41
cerr() (*in module httk.core.console*), 109
chain_vecs() (*httk.core.vectors.fracvector.FracVector class method*), 95
chain_vecs() (*httk.core.vectors.vector.Vector class method*), 102
chain_vecs() (*httk.FracVector class method*), 43
check_symop() (*in module httk.atomistic.spacegrouputils*), 79
checkvalid() (*in module httk.core.ed25519*), 110
cif2cell() (*in module httk.external.cif2cell_ext*), 125
cif_reader_httk_preprocessed() (*in module httk.atomistic.atomisticio.structure_cif_io*), 65
cif_reader_that_can_only_read_isotropy_cif() (*in module httk.atomistic.atomisticio.structure_cif_io*), 65
cif_to_sgstructure() (*in module httk.external.platon_ext*), 127
cif_to_struct() (*in module httk.atomistic.atomisticio.structure_cif_io*), 65
cif_to_structure_noreduce() (*in module httk.external.cif2cell_ext*), 125
cif_to_structure_reduce() (*in module httk.external.cif2cell_ext*), 125
cifdata_to_struct() (*in module httk.atomistic.atomisticio.structure_cif_io*), 65
clean() (*httk.atomistic.Cell method*), 57
clean() (*httk.atomistic.cell.Cell method*), 69
clean() (*httk.atomistic.cellshape.CellShape method*), 70
clean() (*httk.atomistic.RepresentativeSites method*), 58
clean() (*httk.atomistic.representativesites.RepresentativeSites method*), 74
clean() (*httk.atomistic.RepresentativeStructure method*), 63
clean() (*httk.atomistic.representativestructure.RepresentativeStructure method*), 75
clean() (*httk.atomistic.sites.Sites method*), 77
clean() (*httk.atomistic.Structure method*), 54
clean() (*httk.atomistic.structure.Structure method*), 82
clean_coordgroups_and_assignments() (*in module httk.atomistic.sitesutils*), 78
clean_coordgroups_and_assignments() (*in module httk.atomistic.structureutils*), 86
cleveropen() (*in module httk.core.ioadapters*), 113
close() (*httk.core.ioadapters.IoAdapterFileAppender*

method), 112
close() (httk.core.ioadapters.IoAdapterFilename method), 112
close() (httk.core.ioadapters.IoAdapterFileReader method), 112
close() (httk.core.ioadapters.IoAdapterFileWriter method), 112
close() (httk.core.ioadapters.IoAdapterString method), 112
close() (httk.db.backend.sqlite.Sqlite method), 119
close() (httk.db.backend.sqlite.Sqlite.SqliteCursor method), 119
close() (httk.IoAdapterFileAppender method), 49
close() (httk.IoAdapterFileReader method), 49
close() (httk.IoAdapterFileWriter method), 49
close() (httk.IoAdapterString method), 50
Code (class in httk), 41
Code (class in httk.core.code), 108
CodeRef (class in httk.core.code), 108
CodeTag (class in httk.core.code), 108
Command (class in httk.external.command), 126
commit() (httk.db.backend.sqlite.Sqlite method), 119
commit() (httk.db.store.sqlstore.SqlStore method), 121
competing_indices (httk.analysis.matsci.phasediagram.PhaseDiagram attribute), 51
Compound (class in httk.atomistic), 59
Compound (class in httk.atomistic.compound), 73
CompoundName (class in httk.atomistic.compound), 73
CompoundRef (class in httk.atomistic), 60
CompoundRef (class in httk.atomistic.compound), 74
CompoundStructure (class in httk.atomistic), 60
CompoundStructure (class in httk.atomistic.compound), 74
CompoundTag (class in httk.atomistic), 60
CompoundTag (class in httk.atomistic.compound), 74
Computation (class in httk), 41
Computation (class in httk.core.computation), 108
ComputationProject (class in httk), 42
ComputationProject (class in httk.core.computation), 109
ComputationRef (class in httk.core.computation), 109
ComputationRelated (class in httk), 42
ComputationRelated (class in httk.core.computation), 109
ComputationRelatedCompound (class in httk.atomistic.compound), 74
ComputationTag (class in httk.core.computation), 109
connections() (httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method), 67
content() (httk.httkweb.render_httk.RenderHttk method), 130
content() (httk.httkweb.render_rst.RenderRst method), 131
coord_system (httk.analysis.matsci.phasediagram.PhaseDiagram attribute), 51
coordgroups_and_assignments_to_coords_and_occupanci (in module httk.atomistic.structureutils), 86
coordgroups_and_assignments_to_symbols() (in module httk.atomistic.structureutils), 86
coordgroups_cartesian_to_reduced() (httk.atomistic.Cell method), 57
coordgroups_cartesian_to_reduced() (httk.atomistic.cell.Cell method), 69
coordgroups_cartesian_to_reduced() (httk.atomistic.cellshape.CellShape method), 71
coordgroups_cartesian_to_reduced() (in module httk.atomistic.sitesutils), 78
coordgroups_cartesian_to_reduced() (in module httk.atomistic.structureutils), 86
coordgroups_reduced_rc_to_unitcellsites() (in module httk.atomistic.structureutils), 86
coordgroups_reduced_rc_to_unitcellsites() (in module httk.external.ase_glue), 125
coordgroups_reduced_rc_to_unitcellsites() (in module httk.external.cif2cell_ext), 125
coordgroups_reduced_to_cartesian() (httk.atomistic.Cell method), 57
coordgroups_reduced_to_cartesian() (httk.atomistic.cell.Cell method), 69
coordgroups_reduced_to_cartesian() (httk.atomistic.cellshape.CellShape method), 71
coordgroups_reduced_to_cartesian() (in module httk.atomistic.sitesutils), 78
coordgroups_reduced_to_unitcell() (in module httk.atomistic.sitesutils), 78
coordgroups_reduced_uc_to_representative() (in module httk.atomistic.structureutils), 87
coordgroups_to_coords() (in module httk.atomistic.sitesutils), 78
coordgroups_to_coords() (in module httk.atomistic.structureutils), 87
coords() (httk.analysis.matsci.phasediagram.PhaseDiagram method), 51
coords_and_counts_to_coordgroups() (in module httk.atomistic.sitesutils), 78
coords_and_counts_to_coordgroups() (in module httk.atomistic.structureutils), 87
coords_and_occupancies_to_coordgroups_and_assignmer (in module httk.atomistic.sitesutils), 78
coords_and_occupancies_to_coordgroups_and_assignmer (in module httk.atomistic.structureutils), 87
coords_cartesian_to_reduced() (httk.atomistic.Cell method), 57

coords_cartesian_to_reduced() (*httk.atomistic.cell.Cell method*), 69
coords_cartesian_to_reduced() (*httk.atomistic.cellshape.CellShape method*), 71
coords_groupnumber (*httk.atomistic.sites.Sites attribute*), 77
coords_reduced_to_cartesian() (*httk.atomistic.Cell method*), 57
coords_reduced_to_cartesian() (*httk.atomistic.cell.Cell method*), 69
coords_reduced_to_cartesian() (*httk.atomistic.cellshape.CellShape method*), 71
coords_reduced_to_cartesian() (*in module httk.atomistic.sitesutils*), 78
coords_to_coordgroups() (*in module httk.atomistic.sitesutils*), 78
coords_to_coordgroups() (*in module httk.atomistic.structureutils*), 87
coordswap() (*in module httk.atomistic.sitesutils*), 78
coordswap() (*in module httk.atomistic.structureutils*), 87
copy() (*httk.db.filteredcollection.FCDict method*), 122
copy() (*httk.db.filteredcollection.FCMultiDict method*), 122
copy_template() (*in module httk.iface.vasp_if*), 137
copysign() (*in module httk.core.vectors.vectormath*), 104
cos() (*httk.core.vectors.fracvector.FracVector method*), 95
cos() (*httk.FracVector method*), 44
cos() (*in module httk.core.vectors.vectormath*), 104
cosh() (*in module httk.core.vectors.vectormath*), 104
counts (*httk.atomistic.sites.Sites attribute*), 77
cout() (*in module httk*), 41
cout() (*in module httk.core.console*), 109
create() (*httk.analysis.matsci.phasediagram.PhaseDiagram class method*), 51
create() (*httk.atomistic.assignment.Assignment class method*), 68
create() (*httk.atomistic.Assignments class method*), 59
create() (*httk.atomistic.assignments.Assignments class method*), 69
create() (*httk.atomistic.Cell class method*), 57
create() (*httk.atomistic.cell.Cell class method*), 69
create() (*httk.atomistic.cellshape.CellShape class method*), 71
create() (*httk.atomistic.Compound class method*), 60
create() (*httk.atomistic.compound.Compound class method*), 73
create() (*httk.atomistic.compound.CompoundStructure class method*), 74
create() (*httk.atomistic.compound.ComputationRelatedCompound class method*), 74
create() (*httk.atomistic.CompoundStructure class method*), 60
create() (*httk.atomistic.RepresentativeSites class method*), 59
create() (*httk.atomistic.representativesites.RepresentativeSites class method*), 74
create() (*httk.atomistic.RepresentativeStructure class method*), 63
create() (*httk.atomistic.representativestructure.RepresentativeStructure class method*), 75
create() (*httk.atomistic.siteassignment.SiteAssignment class method*), 77
create() (*httk.atomistic.sites.Sites class method*), 77
create() (*httk.atomistic.spacegroup.Spacegroup class method*), 78
create() (*httk.atomistic.Structure class method*), 54
create() (*httk.atomistic.structure.Structure class method*), 82
create() (*httk.atomistic.StructurePhaseDiagram class method*), 60
create() (*httk.atomistic.structurephasediagram.StructurePhaseDiagram class method*), 86
create() (*httk.atomistic.structurephasediagram.StructurePhaseDiagram class method*), 86
create() (*httk.atomistic.UnitcellStructure class method*), 61
create() (*httk.atomistic.unitcellstructure.UnitcellStructure class method*), 90
create() (*httk.Author class method*), 42
create() (*httk.Code class method*), 41
create() (*httk.Computation class method*), 42
create() (*httk.ComputationProject class method*), 42
create() (*httk.ComputationRelated class method*), 42
create() (*httk.core.code.Code class method*), 108
create() (*httk.core.computation.Computation class method*), 109
create() (*httk.core.computation.ComputationProject class method*), 109
create() (*httk.core.computation.ComputationRelated class method*), 109
create() (*httk.core.computation.Result class method*), 109
create() (*httk.core.project.Project class method*), 117
create() (*httk.core.project.ProjectOwner class method*), 117
create() (*httk.core.reference.Author class method*), 118
create() (*httk.core.reference.Reference class method*), 118
create() (*httk.core.signature.Signature class method*), 118
create() (*httk.core.signature.SignatureKey class*

method), 118
create() (httk.core.vectors.fracvector.FracScalar class method), 94
create() (httk.core.vectors.fracvector.FracVector class method), 95
create() (httk.core.vectors.vector.Vector class method), 102
create() (httk.FracScalar class method), 48
create() (httk.FracVector class method), 44
create() (httk.Project class method), 43
create() (httk.Reference class method), 42
create() (httk.Result class method), 42
create() (httk.Signature class method), 50
create() (httk.SignatureKey class method), 51
create_batch_task() (in module httk.task.taskmgr), 138
create_cos() (httk.core.vectors.fracvector.FracVector class method), 96
create_cos() (httk.FracVector class method), 44
create_exp() (httk.core.vectors.fracvector.FracVector class method), 96
create_exp() (httk.FracVector class method), 44
create_sin() (httk.core.vectors.fracvector.FracVector class method), 96
create_sin() (httk.FracVector class method), 45
create_table() (httk.db.backend.sqlite.Sqlite method), 119
create_table() (httk.db.store.dictstore.DictStore method), 120
create_table() (httk.db.store.sqlstore.SqlStore method), 121
create_tmpdir() (in module httk.core.basic), 107
cross() (httk.core.vectors.fracvector.FracVector method), 96
cross() (httk.FracVector method), 45
crystal_system (httk.atomistic.RepresentativeSites attribute), 59
crystal_system (httk.atomistic.representativesites.RepresentativeSites attribute), 74
crystal_system_from_hall() (in module httk.atomistic.spacegrouputils), 79
crystal_system_from_spacegroupnbr() (in module httk.atomistic.spacegrouputils), 79
cubic() (httk.atomistic.supercellutils.StructureSupercellPlugin method), 88
cubic_supercell_transformation() (in module httk.atomistic.supercellutils), 88
cursor() (httk.db.backend.sqlite.Sqlite method), 119

D

data() (httk.db.filteredcollection.FCDict method), 122
data() (httk.db.filteredcollection.FCMultiDict method), 122
db_close() (in module httk.db.backend.sqlite), 120
db_open() (in module httk.db.backend.sqlite), 120
db_sqlite_close_all() (in module httk.db.backend.sqlite), 120
DeclaredFunction (class in httk.db.filteredcollection), 121
decodeint() (in module httk.core.ed25519), 110
decodepoint() (in module httk.core.ed25519), 110
defaults_publish() (httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method), 67
degrees() (in module httk.core.vectors.vectormath), 104
delay_commit() (httk.db.store.sqlstore.SqlStore method), 121
description (httk.db.backend.sqlite.Sqlite.SqliteCursor attribute), 119
destroy_tmpdir() (in module httk.core.basic), 107
det() (httk.core.vectors.fracvector.FracVector method), 96
det() (httk.FracVector method), 45
determine_version_data() (in module httk.config.config), 92
DictStore (class in httk.db.store.dictstore), 120
DictStore.Keeper (class in httk.db.store.dictstore), 120
dim (httk.core.vectors.fracvector.FracVector attribute), 96
dim (httk.FracVector attribute), 45
dont_print_citations_at_exit() (in module httk.core.citation), 108
dot() (httk.core.vectors.fracvector.FracVector method), 96
dot() (httk.FracVector method), 45
duplicate() (httk.db.filteredcollection.FilteredCollection method), 123

E

e() (in module httk.core.vectors.vectormath), 104
ebnf_unqote() (in module httk.core.miniparser), 116
edwards() (in module httk.core.ed25519), 110
element_wyckoff_sequence (httk.atomistic.Structure attribute), 55
element_wyckoff_sequence (httk.atomistic.structure.Structure attribute), 83
encodeint() (in module httk.core.ed25519), 110
encodepoint() (in module httk.core.ed25519), 110
ensure_ase_is_imported() (in module httk.external.ase_glue), 125
ensure_has_cif2cell() (in module httk.external.cif2cell_ext), 126
ensure_has_cif2cell() (in module httk.external.jmol), 126
ensure_has_isotropy() (in module httk.external.isotropy_ext), 126

ensure_has_platon() (*in module httk.external.platon_ext*), 127
ensure_pymatgen_is_imported() (*in module httk.external.pymatgen_glue*), 127
ensure_pyspg_is_imported() (*in module httk.external.pyspglib_ext*), 127
erf() (*in module httk.core.vectors.vectormath*), 104
erfc() (*in module httk.core.vectors.vectormath*), 104
ExceptionlessConfig (*class in httk.config.config*), 92
execute() (*httk.db.backend.sqlite.Sqlite.SqliteCursor method*), 119
execute() (*httk.httkweb.functionhandler_httk.FunctionHandlerHttk method*), 130
execute_and_format() (*httk.httkweb.functionhandler_httk.FunctionHandlerHttk method*), 130
exp() (*httk.core.vectors.fracvector.FracVector method*), 96
exp() (*httk.FracVector method*), 45
exp() (*in module httk.core.vectors.vectormath*), 104
expm1() (*in module httk.core.vectors.vectormath*), 104
expmod() (*in module httk.core.ed25519*), 110
Expression (*class in httk.db.filteredcollection*), 121
extbonds() (*httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method*), 67
extended (*httk.atomistic.Assignments attribute*), 59
extended (*httk.atomistic.assignments.Assignments attribute*), 69
extended (*httk.atomistic.Structure attribute*), 55
extended (*httk.atomistic.structure.Structure attribute*), 83
extensions (*httk.atomistic.Structure attribute*), 55
extensions (*httk.atomistic.structure.Structure attribute*), 83
eye() (*httk.core.vectors.fracvector.FracVector class method*), 96
eye() (*httk.core.vectors.vector.Vector class method*), 102
eye() (*httk.FracVector class method*), 45

F

fabs() (*in module httk.core.vectors.vectormath*), 104
factorial() (*in module httk.core.vectors.vectormath*), 104
fc_checkcontext() (*in module httk.db.filteredcollection*), 124
fc_eval() (*in module httk.db.filteredcollection*), 124
fc_get_srctable_context() (*in module httk.db.filteredcollection*), 124
fc_sql() (*in module httk.db.filteredcollection*), 124
FCDict (*class in httk.db.filteredcollection*), 121
FCMultiDict (*class in httk.db.filteredcollection*), 122
FCMultiSqlite (*class in httk.db.filteredcollection*), 122
FCSqlite (*class in httk.db.filteredcollection*), 122
fetch_codependent_data() (*httk.db.httkobjdbplugin.HttkObjDbPlugin method*), 124
fetchall() (*httk.db.backend.sqlite.Sqlite.SqliteCursor method*), 119
fetchone() (*httk.db.backend.sqlite.Sqlite.SqliteCursor method*), 119
filter_hm() (*in module httk.atomistic.spacegrouputils*), 79
filter_itcnbr_setting() (*in module httk.atomistic.spacegrouputils*), 79
filter_sf() (*in module httk.atomistic.spacegrouputils*), 79
filter_symops() (*in module httk.atomistic.spacegrouputils*), 79
FilteredCollection (*class in httk.db.filteredcollection*), 123
find_all() (*httk.db.storable.Storable class method*), 124
find_executable() (*in module httk.external.command*), 126
find_structure() (*in module httk.atomistic.data.spacegroups*), 66
find_one() (*httk.db.storable.Storable class method*), 124
find_symmetry() (*httk.atomistic.Structure method*), 55
find_symmetry() (*httk.atomistic.structure.Structure method*), 83
flatten() (*httk.core.vectors.fracvector.FracVector method*), 97
flatten() (*httk.FracVector method*), 45
flatten() (*in module httk.core.basic*), 107
floor() (*httk.core.vectors.fracvector.FracVector method*), 97
floor() (*httk.FracVector method*), 45
floor() (*in module httk.core.vectors.vectormath*), 105
fmod() (*in module httk.core.vectors.vectormath*), 105
format_field() (*httk.httkweb.templateengine_httk.HttkTemplateForma method*), 131
formula (*httk.atomistic.Structure attribute*), 55
formula (*httk.atomistic.structure.Structure attribute*), 83
formula_builder (*httk.atomistic.RepresentativeStructure attribute*), 64
formula_builder (*httk.atomistic.representativestructure.Representativ attribute*), 76
formula_builder (*httk.atomistic.UnitcellStructure attribute*), 62
formula_builder (*httk.atomistic.unitcellstructure.UnitcellStructure attribute*), 91

formula_counts (*httk.atomistic.Compound attribute*), 60
formula_counts (*httk.atomistic.compound.Compound attribute*), 73
formula_counts (*httk.atomistic.Structure attribute*), 55
formula_counts (*httk.atomistic.structure.Structure attribute*), 83
formula_spaceseparated (*httk.atomistic.Structure attribute*), 55
formula_spaceseparated (*httk.atomistic.structure.Structure attribute*), 83
formula_symbols (*httk.atomistic.Compound attribute*), 60
formula_symbols (*httk.atomistic.compound.Compound attribute*), 73
formula_symbols (*httk.atomistic.Structure attribute*), 55
formula_symbols (*httk.atomistic.structure.Structure attribute*), 83
frac_acos() (*in module httk.core.vectors.fracmath*), 93
frac_acos_alt() (*in module httk.core.vectors.fracmath*), 93
frac_acos_old() (*in module httk.core.vectors.fracmath*), 93
frac_asin() (*in module httk.core.vectors.fracmath*), 93
frac_atan() (*in module httk.core.vectors.fracmath*), 93
frac_atan2() (*in module httk.core.vectors.fracmath*), 93
frac_atan_old() (*in module httk.core.vectors.fracmath*), 93
frac_cos() (*in module httk.core.vectors.fracmath*), 93
frac_exp() (*in module httk.core.vectors.fracmath*), 93
frac_exp_old() (*in module httk.core.vectors.fracmath*), 93
frac_log() (*in module httk.core.vectors.fracmath*), 93
frac_log10() (*in module httk.core.vectors.fracmath*), 93
frac_log_old() (*in module httk.core.vectors.fracmath*), 93
frac_pi() (*in module httk.core.vectors.fracmath*), 93
frac_pi_old() (*in module httk.core.vectors.fracmath*), 94
frac_sin() (*in module httk.core.vectors.fracmath*), 94
frac_sin_old() (*in module httk.core.vectors.fracmath*), 94
frac_sqrt() (*in module httk.core.vectors.fracmath*), 94
frac_sqrt_old() (*in module httk.core.vectors.fracmath*), 94
frac_tan() (*in module httk.core.vectors.fracmath*), 94
FracScalar (*class in httk*), 48
FracScalar (*class in httk.core.vectors.fracvector*), 94
fraction_from_continued_fraction() (*in module httk.core.vectors.fracmath*), 94
FracVector (*class in httk*), 43
FracVector (*class in httk.core.vectors.fracvector*), 94
frexp() (*in module httk.core.vectors.vectormath*), 105
from_Atoms() (*httk.external.ase_glue.StructureAsePlugin class method*), 125
from_floats() (*httk.core.vectors.fracvector.FracVector class method*), 97
from_floats() (*httk.FracVector class method*), 45
from_FracVector() (*httk.core.vectors.mutablefracvector.MutableFracVector class method*), 100
from_FracVector() (*httk.MutableFracVector class method*), 48
from_tuple() (*httk.core.vectors.fracvector.FracVector class method*), 97
from_tuple() (*httk.FracVector class method*), 45
fsum() (*in module httk.core.vectors.vectormath*), 105
Function (*class in httk.db.filteredcollection*), 123
function() (*httk.db.filteredcollection.FCDict method*), 122
function() (*httk.db.filteredcollection.FCSqlite method*), 123
FunctionHandlerHttk (*class in httk.httkweb.functionhandler_httk*), 130

G

gamma() (*in module httk.core.vectors.vectormath*), 105
ged_prestacked() (*httk.core.vectors.fracvector.FracVector method*), 97
ged_prestacked() (*httk.core.vectors.vector.Vector method*), 102
ged_prestacked() (*httk.FracVector method*), 45
ged_stackedinsert() (*httk.core.vectors.fracvector.FracVector method*), 97
ged_stackedinsert() (*httk.core.vectors.vector.Vector method*), 102
ged_stackedinsert() (*httk.FracVector method*), 45
general() (*httk.atomistic.supercellutils.StructureSupercellPlugin method*), 88
generate_fake_potentials() (*in module httk.iface.gulp_if*), 133
generate_fake_potentials_try2() (*in module httk.iface.gulp_if*), 133
generate_keys() (*in module httk.core.crypto*), 109
get() (*httk.db.store.dictstore.DictStore method*), 120
get() (*httk.db.store.sqlstore.SqlStore method*), 121

get_append() (*httk.core.vectors.fracvector.FracVector method*), 97
get_append() (*httk.core.vectors.vector.Vector method*), 102
get_append() (*httk.FracVector method*), 45
get_axes_standard_order_transform() (*httk.atomistic.Cell method*), 58
get_axes_standard_order_transform() (*httk.atomistic.cell.Cell method*), 70
get_cartesian_coordgroups() (*httk.atomistic.sites.Sites method*), 77
get_cartesian_coords() (*httk.atomistic.sites.Sites method*), 77
get_codependent_data() (*httk.core.httkobject.HttkObject method*), 111
get_codependent_data() (*httk.HttkObject method*), 50
get_continued_fraction() (*in module httk.core.vectors.fracmath*), 94
get_crypto_signature() (*in module httk.core.crypto*), 109
get_dependency_filenames() (*httk.httkweb.functionhandler_httk.FunctionHandlerHttk method*), 130
get_dependency_filenames() (*httk.httkweb.templateengine_httk.TemplateEngineHttk method*), 131
get_dependency_filenames() (*httk.httkweb.templateengine_templator.TemplateEngineTemplator method*), 132
get_extend() (*httk.core.vectors.fracvector.FracVector method*), 97
get_extend() (*httk.core.vectors.vector.Vector method*), 102
get_extend() (*httk.FracVector method*), 45
get_extensions() (*httk.atomistic.assignment.Assignment method*), 69
get_extensions() (*httk.atomistic.siteassignment.SiteAssignment method*), 77
get_field() (*httk.httkweb.templateengine_httk.HttkTemplateFormatter method*), 131
get_hall() (*in module httk.atomistic.spacegrouputils*), 79
get_hm_setting() (*in module httk.atomistic.spacegrouputils*), 79
get_insert() (*httk.core.vectors.fracvector.FracVector method*), 97
get_insert() (*httk.core.vectors.vector.Vector method*), 102
get_insert() (*httk.FracVector method*), 45
get_itcnbr_setting() (*in module httk.atomistic.spacegrouputils*), 79
get_magmom() (*in module httk.iface.vasp_if*), 137
get_magnetizations() (*in module httk.iface.vasp_if*), 137
get_names() (*httk.atomistic.Compound method*), 60
get_names() (*httk.atomistic.compound.Compound method*), 73
get_nonstandard_hall() (*in module httk.atomistic.spacegrouputils*), 79
get_normalized() (*httk.atomistic.Cell method*), 58
get_normalized() (*httk.atomistic.cell.Cell method*), 70
get_normalized_longestvec() (*httk.atomistic.Cell method*), 58
get_normalized_longestvec() (*httk.atomistic.cell.Cell method*), 70
get_phasediagram() (*httk.atomistic.StructurePhaseDiagram method*), 60
get_phasediagram() (*httk.atomistic.structurephasediagram.StructurePhaseDiagram method*), 86
get_prepend() (*httk.core.vectors.fracvector.FracVector method*), 97
get_prepend() (*httk.core.vectors.vector.Vector method*), 102
get_prepend() (*httk.FracVector method*), 45
get_prextend() (*httk.core.vectors.fracvector.FracVector method*), 97
get_prextend() (*httk.core.vectors.vector.Vector method*), 102
get_prextend() (*httk.FracVector method*), 45
get_primitive_basis_transform() (*in module httk.atomistic.structureutils*), 87
get_primitive_to_conventional_basis_transform() (*in module httk.atomistic.cellutils*), 71
get_projects() (*httk.Computation method*), 42
get_projects() (*httk.core.computation.Computation method*), 109
get_proper_hm_symbol() (*in module httk.atomistic.data.spacegroups*), 66
get_pseudopotential() (*in module httk.iface.vasp_if*), 137
get_refs() (*httk.atomistic.Compound method*), 60
get_refs() (*httk.atomistic.compound.Compound method*), 73
get_refs() (*httk.atomistic.Structure method*), 55
get_refs() (*httk.atomistic.structure.Structure method*), 83
get_refs() (*httk.Code method*), 41
get_refs() (*httk.Computation method*), 42
get_refs() (*httk.core.code.Code method*), 108
get_refs() (*httk.core.computation.Computation method*), 109
get_refs() (*httk.core.project.Project method*), 117
get_refs() (*httk.Project method*), 43

get_row() (*httk.db.backend.sqlite.Sqlite method*), 119
get_rows() (*httk.db.backend.sqlite.Sqlite method*), 119
get_srctable_context() (*httk.db.filteredcollection.Expression method*), 121
get_stacked() (*httk.core.vectors.fracvector.FracVector method*), 97
get_stacked() (*httk.core.vectors.vector.Vector method*), 102
get_stacked() (*httk.FracVector method*), 45
get_stidy_spacegroup() (*in module httk.iface.platon_if*), 133
get_symops() (*in module httk.atomistic.spacegrouputils*), 79
get_symops_strs() (*in module httk.atomistic.spacegrouputils*), 79
get_symopshash() (*in module httk.atomistic.spacegrouputils*), 79
get_tag() (*httk.atomistic.Compound method*), 60
get_tag() (*httk.atomistic.compound.Compound method*), 73
get_tag() (*httk.atomistic.Structure method*), 55
get_tag() (*httk.atomistic.structure.Structure method*), 83
get_tag() (*httk.Code method*), 41
get_tag() (*httk.Computation method*), 42
get_tag() (*httk.core.code.Code method*), 108
get_tag() (*httk.core.computation.Computation method*), 109
get_tag() (*httk.core.project.Project method*), 117
get_tag() (*httk.Project method*), 43
get_tags() (*httk.atomistic.Compound method*), 60
get_tags() (*httk.atomistic.compound.Compound method*), 73
get_tags() (*httk.atomistic.Structure method*), 55
get_tags() (*httk.atomistic.structure.Structure method*), 83
get_tags() (*httk.Code method*), 41
get_tags() (*httk.Computation method*), 42
get_tags() (*httk.core.code.Code method*), 108
get_tags() (*httk.core.computation.Computation method*), 109
get_tags() (*httk.core.project.Project method*), 117
get_tags() (*httk.Project method*), 43
get_uc_sites() (*httk.atomistic.RepresentativeSites method*), 59
get_uc_sites() (*httk.atomistic.representativesites.RepresentativeSites method*), 74
get_val() (*httk.db.backend.sqlite.Sqlite method*), 119
get_weight() (*httk.atomistic.assignment.Assignment method*), 69

H

H() (*in module httk.core.ed25519*), 110
hall_symbol (*httk.atomistic.Structure attribute*), 55
hall_symbol (*httk.atomistic.structure.Structure attribute*), 83
handle_data() (*httk.httkweb.app_curses.MyHTMLParser method*), 129
handle_endtag() (*httk.httkweb.app_curses.MyHTMLParser method*), 129
handle_startendtag() (*httk.httkweb.app_curses.MyHTMLParser method*), 129
handle_starttag() (*httk.httkweb.app_curses.MyHTMLParser method*), 129
has_rc_repr (*httk.atomistic.Structure attribute*), 55
has_rc_repr (*httk.atomistic.structure.Structure attribute*), 83
has_uc_repr (*httk.atomistic.Structure attribute*), 55
has_uc_repr (*httk.atomistic.structure.Structure attribute*), 83
hexhash (*httk.core.httkobject.HttkObject attribute*), 111
hexhash (*httk.HttkObject attribute*), 50
hexhash_ioa() (*in module httk.core.crypto*), 109
hexhash_str() (*in module httk.core.crypto*), 109
Hint() (*in module httk.core.ed25519*), 110
httk (*module*), 40
httk.analysis (*module*), 51
httk.analysis.matsci (*module*), 51
httk.analysis.matsci.phasediagram (*module*), 51
httk.analysis.matsci.vis (*module*), 51
httk.analysis.matsci.vis.phasediagramvisualizerplug (*module*), 51
httk.atomistic (*module*), 52
httk.atomistic.assignment (*module*), 68
httk.atomistic.assignments (*module*), 69
httk.atomistic.atomisticio (*module*), 65
httk.atomistic.atomisticio.structure_cif_io (*module*), 65
httk.atomistic.atomisticio.structure_io (*module*), 65
httk.atomistic.atomisticio.structureioplugin (*module*), 65
httk.atomistic.cell (*module*), 69
httk.atomistic.cellshape (*module*), 70
httk.atomistic.cellutils (*module*), 71
httk.atomistic.cli (*module*), 73
httk.atomistic.compound (*module*), 73
httk.atomistic.data (*module*), 66
httk.atomistic.data.periodictable (*module*), 66
httk.atomistic.data.spacegroups (*module*), 66

httk.atomistic.formulautils (*module*), 74
httk.atomistic.representativesites (*module*), 74
httk.atomistic.representativestructure (*module*), 75
httk.atomistic.results (*module*), 66
httk.atomistic.results.relaxedcellresult (*module*), 67
httk.atomistic.results.totalenergyresult (*module*), 67
httk.atomistic.siteassignment (*module*), 77
httk.atomistic.sites (*module*), 77
httk.atomistic.sitesutils (*module*), 78
httk.atomistic.spacegroup (*module*), 78
httk.atomistic.spacegrouputils (*module*), 79
httk.atomistic.structure (*module*), 80
httk.atomistic.structurephasediagram (*module*), 86
httk.atomistic.structureutils (*module*), 86
httk.atomistic.supercellutils (*module*), 88
httk.atomistic.unitcellsites (*module*), 89
httk.atomistic.unitcellstructure (*module*), 89
httk.atomistic.vis (*module*), 67
httk.atomistic.vis.asestructurevisualizer (*module*), 67
httk.atomistic.vis.jmolstructurevisualizer (*module*), 67
httk.atomistic.vis.structurephasediagramvisualizerplugin (*module*), 68
httk.atomistic.vis.structurevisualizerplugin (*module*), 68
httk.cli (*module*), 138
httk.config (*module*), 92
httk.config.config (*module*), 92
httk.core (*module*), 92
httk.core.basic (*module*), 107
httk.core.citation (*module*), 107
httk.core.code (*module*), 108
httk.core.computation (*module*), 108
httk.core.console (*module*), 109
httk.core.crypto (*module*), 109
httk.core.ed25519 (*module*), 110
httk.core.geometry (*module*), 110
httk.core.httkobject (*module*), 111
httk.core.ioadapters (*module*), 112
httk.core.miniparser (*module*), 113
httk.core.project (*module*), 117
httk.core.reference (*module*), 118
httk.core.signature (*module*), 118
httk.core.template (*module*), 118
httk.core.vectors (*module*), 92
httk.core.vectors.fracmath (*module*), 93
httk.core.vectors.fracvector (*module*), 94
httk.core.vectors.mutablefracvector (*module*), 100
httk.core.vectors.vector (*module*), 102
httk.core.vectors.vectormath (*module*), 103
httk.db (*module*), 119
httk.db.backend (*module*), 119
httk.db.backend.sqlite (*module*), 119
httk.db.filteredcollection (*module*), 121
httk.db.httkobjdbplugin (*module*), 124
httk.db.storable (*module*), 124
httk.db.store (*module*), 120
httk.db.store.dictstore (*module*), 120
httk.db.store.sqlstore (*module*), 121
httk.external (*module*), 125
httk.external.aflow_ext (*module*), 125
httk.external.ase_glue (*module*), 125
httk.external.cif2cell_ext (*module*), 125
httk.external.command (*module*), 126
httk.external.gulp_ext (*module*), 126
httk.external.isotropy_ext (*module*), 126
httk.external.jmol (*module*), 126
httk.external.platon_ext (*module*), 126
httk.external.pymatgen_glue (*module*), 127
httk.external.pyspglib_ext (*module*), 127
httk.external.subimport (*module*), 127
httk.graphics (*module*), 127
httk.graphics.matplotlib (*module*), 127
httk.httkio (*module*), 127
httk.httkioplugin (*module*), 128
httk.httkio.load (*module*), 129
httk.httkio.save (*module*), 129
httk.httkweb (*module*), 129
httk.httkweb.app_curses (*module*), 129
httk.httkweb.app_qt5 (*module*), 129
httk.httkweb.functionhandler_httk (*module*), 130
httk.httkweb.helpers (*module*), 130
httk.httkweb.publish (*module*), 130
httk.httkweb.render_httk (*module*), 130
httk.httkweb.render_rst (*module*), 131
httk.httkweb.serve (*module*), 131
httk.httkweb.templateengine_httk (*module*), 131
httk.httkweb.templateengine_templator (*module*), 131
httk.httkweb.webgenerator (*module*), 132
httk.iface (*module*), 132
httk.iface.ase_if (*module*), 132
httk.iface.cif2cell_if (*module*), 132
httk.iface.gulp_if (*module*), 133
httk.iface.isotropy_if (*module*), 133
httk.iface.jmol_if (*module*), 133
httk.iface.platon_if (*module*), 133

httk.iface.spglib_if (*module*), 137
httk.iface.vasp_if (*module*), 137
httk.task (*module*), 138
httk.task.reader (*module*), 138
httk.task.taskmgr (*module*), 138
httk.versioning (*module*), 138
httk_typed_init() (*in module httk*), 50
httk_typed_init() (*in module httk.core.httkobject*), 112
httk_typed_init_delayed() (*in module httk*), 50
httk_typed_init_delayed() (*in module httk.core.httkobject*), 112
httk_typed_property() (*in module httk*), 50
httk_typed_property() (*in module httk.core.httkobject*), 112
httk_typed_property_delayed() (*in module httk*), 50
httk_typed_property_delayed() (*in module httk.core.httkobject*), 112
httk_typed_property_resolve() (*in module httk.core.httkobject*), 112
HttkObjDbPlugin (*class in httk.db.httkobjdbplugin*), 124
HttkObject (*class in httk*), 50
HttkObject (*class in httk.core.httkobject*), 111
HttkPlugin (*class in httk*), 50
HttkPlugin (*class in httk.core.httkobject*), 111
HttkPluginPlaceholder (*class in httk*), 50
HttkPluginPlaceholder (*class in httk.core.httkobject*), 111
HttkPluginWrapper (*class in httk*), 50
HttkPluginWrapper (*class in httk.core.httkobject*), 112
HttkTemplateFormatter (*class in httk.httkweb.templateengine_httk*), 131
HttkTypedProperty (*class in httk.core.httkobject*), 112
hull_competing_indices (*httk.analysis.matsci.phasediagram.PhaseDiagram attribute*), 51
hull_competing_phase_lines() (*httk.analysis.matsci.phasediagram.PhaseDiagram method*), 51
hull_distances (*httk.analysis.matsci.phasediagram.PhaseDiagram attribute*), 52
hull_indices (*httk.analysis.matsci.phasediagram.PhaseDiagram attribute*), 52
hull_point_coords() (*httk.analysis.matsci.phasediagram.PhaseDiagram method*), 52
hull_points() (*httk.analysis.matsci.phasediagram.PhaseDiagram method*), 52
hull_to_interior_competing_phase_lines() (*httk.analysis.matsci.phasediagram.PhaseDiagram method*), 52
hull_z() (*in module httk.core.geometry*), 110
hypot() (*in module httk.core.vectors.vectormath*), 105

I

identify() (*in module httk.httkweb.helpers*), 130
ignore_close_tags (*httk.httkweb.app_curses.MyHTMLParser attribute*), 129
ignore_content (*httk.httkweb.app_curses.MyHTMLParser attribute*), 129
initialize() (*httk.atomistic.vis.jmolstructurevisualizer.JmolStructureV method*), 67
inmap() (*in module httk.core.vectors.mutablefracvector*), 101
insert() (*httk.db.backend.sqlite.Sqlite method*), 119
insert() (*httk.db.store.dictstore.DictStore method*), 120
insert() (*httk.db.store.sqlstore.SqlStore method*), 121
insert_row() (*httk.db.backend.sqlite.Sqlite method*), 119
instantiate_from_store() (*in module httk.db.filteredcollection*), 124
int_to_anonymous_symbol() (*in module httk.core.basic*), 107
integer_sqrt() (*in module httk.core.vectors.fracmath*), 94
interior_competing_phase_lines() (*httk.analysis.matsci.phasediagram.PhaseDiagram method*), 52
interior_point_coords() (*httk.analysis.matsci.phasediagram.PhaseDiagram method*), 52
internal_coordgroups_reduced_rc_to_unitcellsites() (*in module httk.atomistic.structureutils*), 87
inv() (*httk.core.vectors.fracvector.FracVector method*), 97
inv() (*httk.FracVector method*), 46
inv() (*in module httk.core.ed25519*), 110
invalidate() (*httk.core.vectors.mutablefracvector.MutableFracVector method*), 100
invalidate() (*httk.MutableFracVector method*), 48
io (*httk.atomistic.Structure attribute*), 55
io (*httk.atomistic.structure.Structure attribute*), 83
IoAdapterFileAppender (*class in httk*), 49
IoAdapterFileAppender (*class in httk.core.ioadapters*), 112
IoAdapterFilename (*class in httk.core.ioadapters*), 112
IoAdapterFileReader (*class in httk*), 49
IoAdapterFileReader (*class in httk.core.ioadapters*), 112
IoAdapterFileWriter (*class in httk*), 49

IoAdapterFileWriter (*class in httk.core.ioadapters*), 112
IoAdapterString (*class in httk*), 50
IoAdapterString (*class in httk.core.ioadapters*), 112
IoAdapterStringList (*class in httk*), 50
IoAdapterStringList (*class in httk.core.ioadapters*), 113
is_any_part_of_cube_inside_cell() (*in module httk.core.geometry*), 111
is_dualmagnetic() (*in module httk.iface.vasp_if*), 137
is_in() (*httk.db.filteredcollection.Expression method*), 121
is_point_inside() (*httk.atomistic.Cell method*), 58
is_point_inside() (*httk.atomistic.cell.Cell method*), 70
is_point_inside() (*httk.atomistic.cellshape.CellShape method*), 71
is_point_inside_cell() (*in module httk.core.geometry*), 111
is_point_inside_tetra() (*in module httk.core.geometry*), 111
is_string() (*in module httk.core.vectors.fracmath*), 94
is_unary() (*in module httk.core.basic*), 107
isanyinf() (*in module httk.core.vectors.vectormath*), 105
isanynan() (*in module httk.core.vectors.vectormath*), 105
isinf() (*in module httk.core.vectors.vectormath*), 105
isnan() (*in module httk.core.vectors.vectormath*), 105
isoncurve() (*in module httk.core.ed25519*), 110
isotropy() (*in module httk.external.isotropy_ext*), 126

J

jmol() (*in module httk.external.gulp_ext*), 126
JmolStructureVisualizer (*class in httk.atomistic.vis.jmolstructurevisualizer*), 67

L

lattice_symbol (*httk.atomistic.RepresentativeSites attribute*), 59
lattice_symbol (*httk.atomistic.representativesites.RepresentativeSites attribute*), 75
lattice_symbol_from_hall() (*in module httk.atomistic.spacegrouputils*), 79
lattice_system (*httk.atomistic.RepresentativeSites attribute*), 59
lattice_system (*httk.atomistic.representativesites.RepresentativeSites attribute*), 75
lattice_system_from_hall() (*in module httk.atomistic.spacegrouputils*), 79
lattice_system_from_lengths_and_cosangles() (*in module httk.atomistic.cellutils*), 71
lattice_system_from_niggli() (*in module httk.atomistic.cellutils*), 72
lattice_type_from_hall() (*in module httk.atomistic.spacegrouputils*), 79
ldexp() (*in module httk.core.vectors.vectormath*), 105
left_punctuation_chars (*httk.httkweb.render_httk.RenderHttk attribute*), 130
lengths_and_angles_to_niggli() (*in module httk.atomistic.cellutils*), 72
lengths_and_cosangles_to_conventional_basis() (*in module httk.atomistic.cellutils*), 72
lengths_and_cosangles_to_niggli() (*in module httk.atomistic.cellutils*), 72
lengths_angles_to_niggli() (*in module httk.atomistic.structureutils*), 87
lengthsqr() (*httk.core.vectors.fracvector.FracVector method*), 97
lengthsqr() (*httk.FracVector method*), 46
lexer() (*in module httk.core.miniparser*), 116
lgamma() (*in module httk.core.vectors.vectormath*), 105
like() (*httk.db.filteredcollection.Expression method*), 121
limit_denominator() (*httk.core.vectors.fracvector.FracVector method*), 97
limit_denominator() (*httk.FracVector method*), 46
line_coords() (*httk.analysis.matsci.phasediagram.PhaseDiagram method*), 52
list_set_slice() (*in module httk.core.vectors.mutablefracvector*), 101
list_slice() (*in module httk.core.vectors.mutablefracvector*), 101
load() (*httk.atomistic.atomisticio.structureioplugin.StructureIoPlugin class method*), 65
load() (*in module httk*), 41
load() (*in module httk.httkio.load*), 129
load_struct() (*in module httk.atomistic.atomisticio.structure_io*), 65
log() (*in module httk.core.vectors.vectormath*), 105
log10() (*in module httk.core.vectors.vectormath*), 106
log1p() (*in module httk.core.vectors.vectormath*), 106
logger() (*in module httk.core.miniparser*), 117
LogVerbosity (*class in httk.core.miniparser*), 115

M

magnetization_recurse() (*in module httk.iface.vasp_if*), 137

main() (*in module httk.atomistic.assignment*), 69
main() (*in module httk.atomistic.assignments*), 69
main() (*in module httk.atomistic.cell*), 70
main() (*in module httk.atomistic.cellshape*), 71
main() (*in module httk.atomistic.cellutils*), 72
main() (*in module httk.atomistic.cli*), 73
main() (*in module httk.atomistic.compound*), 74
main() (*in module httk.atomistic.representativesites*), 75
main() (*in module httk.atomistic.representativestructure*), 76
main() (*in module httk.atomistic.siteassignment*), 77
main() (*in module httk.atomistic.sites*), 78
main() (*in module httk.atomistic.sitesutils*), 78
main() (*in module httk.atomistic.spacegroup*), 79
main() (*in module httk.atomistic.spacegrouputils*), 79
main() (*in module httk.atomistic.structure*), 85
main() (*in module httk.atomistic.structurephasediagram*), 86
main() (*in module httk.atomistic.structureutils*), 87
main() (*in module httk.atomistic.unitcellsites*), 89
main() (*in module httk.cli*), 138
main() (*in module httk.core.basic*), 107
main() (*in module httk.core.code*), 108
main() (*in module httk.core.computation*), 109
main() (*in module httk.core.crypto*), 110
main() (*in module httk.core.ed25519*), 110
main() (*in module httk.core.ioadapters*), 113
main() (*in module httk.core.project*), 118
main() (*in module httk.core.reference*), 118
main() (*in module httk.core.signature*), 118
main() (*in module httk.core.vectors.fracmath*), 94
main() (*in module httk.core.vectors.fracvector*), 99
main() (*in module httk.core.vectors.mutablefracvector*), 101
main() (*in module httk.core.vectors.vector*), 103
main() (*in module httk.core.vectors.vectormath*), 106
main() (*in module httk.external.jmol*), 126
main() (*in module httk.httkio.cif*), 128
main() (*in module httk.task.reader*), 138
make_id() (*httk.httkweb.render_httk.RenderHttk method*), 130
manifest_dir() (*in module httk.core.crypto*), 110
max() (*httk.core.vectors.fracvector.FracVector method*), 97
max() (*httk.FracVector method*), 46
metadata() (*httk.httkweb.render_httk.RenderHttk method*), 130
metadata() (*httk.httkweb.render_rst.RenderRst method*), 131
metric_product() (*httk.core.vectors.fracvector.FracVector method*), 97
metric_product() (*httk.FracVector method*), 46
metric_to_niggli() (*in module httk.atomistic.cellutils*), 72
metric_to_niggli() (*in module httk.atomistic.structureutils*), 87
micro_pyawk() (*in module httk.core.basic*), 107
min() (*httk.core.vectors.fracvector.FracVector method*), 97
min() (*httk.FracVector method*), 46
mkdir_p() (*in module httk.core.basic*), 107
modf() (*in module httk.core.vectors.vectormath*), 106
modify_structure() (*httk.db.backend.sqlite.Sqlite method*), 119
most_common_mass() (*in module httk.atomistic.data.periodictable*), 66
mul() (*httk.core.vectors.fracvector.FracVector method*), 97
mul() (*httk.FracVector method*), 46
MutableFracVector (*class in httk*), 48
MutableFracVector (*class in httk.core.vectors.mutablefracvector*), 100
MutableVector (*class in httk.core.vectors.vector*), 102
MyHTMLParser (*class in httk.httkweb.app_curses*), 129

N

name (*httk.external.ase_glue.StructureAsePlugin attribute*), 125
nargmax() (*httk.core.vectors.fracvector.FracVector method*), 97
nargmax() (*httk.FracVector method*), 46
nargmin() (*httk.core.vectors.fracvector.FracVector method*), 98
nargmin() (*httk.FracVector method*), 46
nested_inmap() (*httk.core.vectors.mutablefracvector.MutableFracVect static method*), 100
nested_inmap() (*httk.MutableFracVector static method*), 48
nested_inmap_list() (*in module httk.core.vectors.mutablefracvector*), 101
nested_map() (*httk.core.vectors.fracvector.FracVector static method*), 98
nested_map() (*httk.core.vectors.mutablefracvector.MutableFracVector static method*), 101
nested_map() (*httk.FracVector static method*), 46
nested_map() (*httk.MutableFracVector static method*), 48
nested_map_fractions() (*httk.core.vectors.fracvector.FracVector static method*), 98
nested_map_fractions() (*httk.core.vectors.mutablefracvector.MutableFracVector static method*), 101
nested_map_fractions() (*httk.FracVector static method*), 46

nested_map_fractions() (*httk.MutableFracVector static method*), 49
nested_map_fractions_list() (*in module httk.core.vectors.fracvector*), 99
nested_map_fractions_list() (*in module httk.core.vectors.vector*), 103
nested_map_fractions_tuple() (*in module httk.core.vectors.fracvector*), 99
nested_map_list() (*in module httk.core.vectors.fracvector*), 99
nested_map_list() (*in module httk.core.vectors.vector*), 103
nested_map_tuple() (*in module httk.core.vectors.fracvector*), 99
nested_reduce() (*in module httk.core.vectors.fracvector*), 99
nested_reduce() (*in module httk.core.vectors.vector*), 103
nested_reduce_fractions() (*in module httk.core.vectors.fracvector*), 100
nested_reduce_fractions() (*in module httk.core.vectors.vector*), 103
nested_reduce_levels() (*in module httk.core.vectors.fracvector*), 100
nested_reduce_levels() (*in module httk.core.vectors.vector*), 103
nested_split() (*in module httk.core.basic*), 107
new() (*httk.db.storable.TrivialStore method*), 124
new() (*httk.db.store.dictstore.DictStore method*), 120
new() (*httk.db.store.sqlstore.SqlStore method*), 121
new_from() (*httk.core.httkobject.HttkObject class method*), 111
new_from() (*httk.HttkObject class method*), 50
next() (*httk.core.basic.rewindable_iterator method*), 107
niggli_scale_to_vol() (*in module httk.atomistic.cellutils*), 72
niggli_scale_to_vol() (*in module httk.atomistic.structureutils*), 87
niggli_to_basis() (*in module httk.atomistic.cellutils*), 72
niggli_to_basis() (*in module httk.atomistic.structureutils*), 87
niggli_to_cell_old() (*in module httk.atomistic.structureutils*), 87
niggli_to_conventional_basis() (*in module httk.atomistic.cellutils*), 72
niggli_to_lengths_and_angles() (*in module httk.atomistic.cellutils*), 72
niggli_to_lengths_and_trigangles() (*in module httk.atomistic.cellutils*), 72
niggli_to_lengths_angles() (*in module httk.atomistic.structureutils*), 87
niggli_to_metric() (*in module httk.atomistic.cellutils*), 72
niggli_to_metric() (*in module httk.atomistic.structureutils*), 87
niggli_vol_to_scale() (*in module httk.atomistic.structureutils*), 87
nom (*httk.core.vectors.fracvector.FracVector attribute*), 98
nom (*httk.FracVector attribute*), 46
normalization_longestvec_scale (*httk.atomistic.Cell attribute*), 58
normalization_longestvec_scale (*httk.atomistic.cell.Cell attribute*), 70
normalization_scale (*httk.atomistic.Cell attribute*), 58
normalization_scale (*httk.atomistic.cell.Cell attribute*), 70
normalize() (*httk.core.vectors.fracvector.FracVector method*), 98
normalize() (*httk.FracVector method*), 46
normalize_half() (*httk.core.vectors.fracvector.FracVector method*), 98
normalize_half() (*httk.FracVector method*), 46
normalized_formula() (*in module httk.atomistic.structureutils*), 87
normalized_formula_parts() (*in module httk.atomistic.sitesutils*), 78
normalized_formula_parts() (*in module httk.atomistic.structureutils*), 87
number (*httk.atomistic.spacegroup.Spacegroup attribute*), 79
number_and_setting (*httk.atomistic.spacegroup.Spacegroup attribute*), 79
number_of_elements (*httk.atomistic.Compound attribute*), 60
number_of_elements (*httk.atomistic.compound.Compound attribute*), 73
number_of_elements (*httk.atomistic.Structure attribute*), 55
number_of_elements (*httk.atomistic.structure.Structure attribute*), 83
numpy_quickhull_2d() (*in module httk.core.geometry*), 111

O

occupations_and_coords_to_assignments_and_coordgrou (*in module httk.atomistic.structureutils*), 87
open_url() (*httk.httkweb.app_curses.WebviewCurses method*), 129
option_list_characters (*httk.httkweb.render_httk.RenderHttk attribute*), 130

orthogonal() (*httk.atomistic.supercellutils.StructureSupercellPlugin method*), 88
orthogonal_supercell_transformation() (*in module httk.atomistic.supercellutils*), 88
other_point_coords() (*httk.analysis.matsci.phasediagram.PhaseDiagram method*), 52
out_to_cif() (*in module httk.iface.isotropy_if*), 133
out_to_struct() (*in module httk.iface.cif2cell_if*), 132
OutcarReader (*class in httk.iface.vasp_if*), 137
output() (*httk.db.filteredcollection.FilteredCollection method*), 123

P

Page (*class in httk.httkweb.webgenerator*), 132
params() (*httk.analysis.matsci.vis.phasediagramvisualizerplugin.PhaseDiagramVisualizerPlugin method*), 51
params() (*httk.atomistic.vis.structurevisualizerplugin.StructureVisualizerPlugin method*), 68
parse() (*httk.iface.vasp_if.OutcarReader method*), 137
parse_parexpr() (*in module httk.core.basic*), 107
parser() (*in module httk.core.miniparser*), 117
ParserError, 115
ParserGrammarError, 115
ParserInternalError, 115
ParserSyntaxError, 115
pbc (*httk.atomistic.RepresentativeStructure attribute*), 64
pbc (*httk.atomistic.representativestructure.RepresentativeStructure attribute*), 76
pbc (*httk.atomistic.Structure attribute*), 55
pbc (*httk.atomistic.structure.Structure attribute*), 83
pbc (*httk.atomistic.UnitcellStructure attribute*), 62
pbc (*httk.atomistic.unitcellstructure.UnitcellStructure attribute*), 91
pbc_to_nonperiodic_vecs() (*in module httk.atomistic.sitesutils*), 78
pc (*httk.atomistic.Structure attribute*), 55
pc (*httk.atomistic.structure.Structure attribute*), 83
pc_a (*httk.atomistic.Structure attribute*), 55
pc_a (*httk.atomistic.structure.Structure attribute*), 83
pc_alpha (*httk.atomistic.Structure attribute*), 55
pc_alpha (*httk.atomistic.structure.Structure attribute*), 83
pc_b (*httk.atomistic.Structure attribute*), 55
pc_b (*httk.atomistic.structure.Structure attribute*), 83
pc_beta (*httk.atomistic.Structure attribute*), 55
pc_beta (*httk.atomistic.structure.Structure attribute*), 83
pc_c (*httk.atomistic.Structure attribute*), 55
pc_c (*httk.atomistic.structure.Structure attribute*), 83
pc_counts (*httk.atomistic.Structure attribute*), 55
pc_counts (*httk.atomistic.structure.Structure attribute*), 83
pc_formula_parts (*httk.atomistic.Structure attribute*), 55
pc_formula_parts (*httk.atomistic.structure.Structure attribute*), 83
pc_gamma (*httk.atomistic.Structure attribute*), 55
pc_gamma (*httk.atomistic.structure.Structure attribute*), 83
pc_nbr_atoms (*httk.atomistic.Structure attribute*), 56
pc_nbr_atoms (*httk.atomistic.structure.Structure attribute*), 84
pc_volume (*httk.atomistic.Structure attribute*), 56
pc_volume (*httk.atomistic.structure.Structure attribute*), 84
periodicity_to_pbc() (*in module httk.atomistic.sitesutils*), 78
phase_lines (*httk.analysis.matsci.phasediagram.PhaseDiagram attribute*), 52
PhaseDiagram (*class in httk.analysis.matsci.phasediagram*), 51
PhaseDiagramVisualizerPlugin (*class in httk.analysis.matsci.vis.phasediagramvisualizerplugin*), 51
pi() (*httk.core.vectors.fracvector.FracVector class method*), 98
pi() (*httk.FracVector class method*), 46
pi() (*in module httk.core.vectors.vectormath*), 106
platon() (*in module httk.external.platon_ext*), 127
platon_lis_to_struct_broken() (*in module httk.iface.platon_if*), 133
platon_lis_to_struct_broken2() (*in module httk.iface.platon_if*), 134
platon_sites_to_styin() (*in module httk.iface.platon_if*), 135
platon_styin_to_sgstruct() (*in module httk.iface.platon_if*), 135
platon_styout_to_sgstruct() (*in module httk.iface.platon_if*), 135
platon_styout_to_structure() (*in module httk.iface.platon_if*), 136
plugin_init() (*httk.analysis.matsci.vis.phasediagramvisualizerplugin. method*), 51
plugin_init() (*httk.atomistic.atomisticio.structureioplugin.StructureIo method*), 65
plugin_init() (*httk.atomistic.formulautils.StructureFormulaPlugin method*), 74
plugin_init() (*httk.atomistic.supercellutils.StructureSupercellPlugin method*), 88
plugin_init() (*httk.atomistic.vis.structurephasediagramvisualizerplug method*), 68
plugin_init() (*httk.atomistic.vis.structurevisualizerplugin.StructureVi method*), 68
plugin_init() (*httk.db.httkobjdbplugin.HttkObjDbPlugin method*), 124
plugin_init() (*httk.external.ase_glue.StructureAsePlugin*

method), 125
polyhedra() (*httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method*), 68
poscar_to_strs() (*in module httk.iface.vasp_if*), 137
poscar_to_structure() (*in module httk.iface.vasp_if*), 137
postconnect() (*httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method*), 68
pow() (*in module httk.core.vectors.vectormath*), 106
preconnect() (*httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method*), 68
prepare_single_run() (*in module httk.iface.vasp_if*), 137
primitive() (*in module httk.external.pyspglib_ext*), 127
primitive_from_conventional_cell() (*in module httk.external.ase_glue*), 125
print_citations() (*in module httk.core.citation*), 108
print_citations_at_exit() (*in module httk.core.citation*), 108
Project (*class in httk*), 42
Project (*class in httk.core.project*), 117
ProjectOwner (*class in httk.core.project*), 117
ProjectRef (*class in httk*), 43
ProjectRef (*class in httk.core.project*), 118
ProjectTag (*class in httk*), 43
ProjectTag (*class in httk.core.project*), 118
prototype_formula() (*in module httk.atomistic.structureutils*), 88
publickey() (*in module httk.core.ed25519*), 110
publish() (*in module httk.httkweb.publish*), 130
put() (*httk.db.store.dictstore.DictStore method*), 120
put() (*httk.db.store.sqlstore.SqlStore method*), 121
puts() (*httk.db.store.dictstore.DictStore method*), 120
puts() (*httk.db.store.dictstore.DictStore.Keeper method*), 120
puts() (*httk.db.store.sqlstore.SqlStore method*), 121
puts() (*httk.db.store.sqlstore.SqlStore.Keeper method*), 121

Q

query() (*httk.db.backend.sqlite.Sqlite method*), 120

R

radians() (*in module httk.core.vectors.vectormath*), 106
random() (*httk.core.vectors.fracvector.FracVector class method*), 98
random() (*httk.core.vectors.vector.Vector class method*), 102
random() (*httk.FracVector class method*), 46
ratio (*httk.atomistic.siteassignment.SiteAssignment attribute*), 77
ratios (*httk.atomistic.Assignments attribute*), 59
ratios (*httk.atomistic.assignments.Assignments attribute*), 69
ratios (*httk.atomistic.siteassignment.SiteAssignment attribute*), 77
ratioslist (*httk.atomistic.Assignments attribute*), 59
ratioslist (*httk.atomistic.assignments.Assignments attribute*), 69
rc (*httk.atomistic.Structure attribute*), 56
rc (*httk.atomistic.structure.Structure attribute*), 84
rc_a (*httk.atomistic.RepresentativeStructure attribute*), 64
rc_a (*httk.atomistic.representativestructure.RepresentativeStructure attribute*), 76
rc_a (*httk.atomistic.Structure attribute*), 56
rc_a (*httk.atomistic.structure.Structure attribute*), 84
rc_alpha (*httk.atomistic.RepresentativeStructure attribute*), 64
rc_alpha (*httk.atomistic.representativestructure.RepresentativeStructure attribute*), 76
rc_alpha (*httk.atomistic.Structure attribute*), 56
rc_alpha (*httk.atomistic.structure.Structure attribute*), 84
rc_b (*httk.atomistic.RepresentativeStructure attribute*), 64
rc_b (*httk.atomistic.representativestructure.RepresentativeStructure attribute*), 76
rc_b (*httk.atomistic.Structure attribute*), 56
rc_b (*httk.atomistic.structure.Structure attribute*), 84
rc_basis (*httk.atomistic.RepresentativeStructure attribute*), 64
rc_basis (*httk.atomistic.representativestructure.RepresentativeStructure attribute*), 76
rc_basis (*httk.atomistic.Structure attribute*), 56
rc_basis (*httk.atomistic.structure.Structure attribute*), 84
rc_beta (*httk.atomistic.RepresentativeStructure attribute*), 64
rc_beta (*httk.atomistic.representativestructure.RepresentativeStructure attribute*), 76
rc_beta (*httk.atomistic.Structure attribute*), 56
rc_beta (*httk.atomistic.structure.Structure attribute*), 84
rc_c (*httk.atomistic.RepresentativeStructure attribute*), 64
rc_c (*httk.atomistic.representativestructure.RepresentativeStructure attribute*), 76
rc_c (*httk.atomistic.Structure attribute*), 56
rc_c (*httk.atomistic.structure.Structure attribute*), 84
rc_cartesian_coordgroups (*httk.atomistic.RepresentativeStructure attribute*), 64

rc_cartesian_coordgroups (*httk.atomistic.representativestructure.RepresentativeStructure attribute*), 76
rc_cartesian_coordgroups (*httk.atomistic.Structure attribute*), 56
rc_cartesian_coordgroups (*httk.atomistic.structure.Structure attribute*), 84
rc_cartesian_coords (*httk.atomistic.RepresentativeStructure attribute*), 64
rc_cartesian_coords (*httk.atomistic.representativestructure.RepresentativeStructure attribute*), 76
rc_cartesian_coords (*httk.atomistic.Structure attribute*), 56
rc_cartesian_coords (*httk.atomistic.structure.Structure attribute*), 84
rc_cartesian_occupationscoords (*httk.atomistic.RepresentativeStructure attribute*), 64
rc_cartesian_occupationscoords (*httk.atomistic.representativestructure.RepresentativeStructure attribute*), 76
rc_cartesian_occupationscoords (*httk.atomistic.Structure attribute*), 56
rc_cartesian_occupationscoords (*httk.atomistic.structure.Structure attribute*), 84
rc_cell_orientation (*httk.atomistic.RepresentativeStructure attribute*), 64
rc_cell_orientation (*httk.atomistic.representativestructure.RepresentativeStructure attribute*), 76
rc_cell_orientation (*httk.atomistic.Structure attribute*), 56
rc_cell_orientation (*httk.atomistic.structure.Structure attribute*), 84
rc_counts (*httk.atomistic.Structure attribute*), 56
rc_counts (*httk.atomistic.structure.Structure attribute*), 84
rc_gamma (*httk.atomistic.RepresentativeStructure attribute*), 65
rc_gamma (*httk.atomistic.representativestructure.RepresentativeStructure attribute*), 76
rc_gamma (*httk.atomistic.Structure attribute*), 56
rc_gamma (*httk.atomistic.structure.Structure attribute*), 84
rc_lengths_and_angles (*httk.atomistic.RepresentativeStructure attribute*), 65
rc_lengths_and_angles (*httk.atomistic.representativestructure.RepresentativeStructure attribute*), 76
rc_lengths_and_angles (*httk.atomistic.Structure attribute*), 56
rc_lengths_and_angles (*httk.atomistic.structure.Structure attribute*), 84
rc_nbr_atoms (*httk.atomistic.Structure attribute*), 56
rc_nbr_atoms (*httk.atomistic.structure.Structure attribute*), 84
rc_occupancies (*httk.atomistic.Structure attribute*), 56
rc_occupancies (*httk.atomistic.structure.Structure attribute*), 84
rc_occupationssymbols (*httk.atomistic.Structure attribute*), 56
rc_occupationssymbols (*httk.atomistic.structure.Structure attribute*), 84
rc_reduced_coordgroups (*httk.atomistic.Structure attribute*), 56
rc_reduced_coordgroups (*httk.atomistic.structure.Structure attribute*), 84
rc_reduced_coords (*httk.atomistic.Structure attribute*), 56
rc_reduced_coords (*httk.atomistic.structure.Structure attribute*), 84
rc_structure_to_symbols_and_scaled_positions() (*in module httk.iface.ase_if*), 132
rc_volume (*httk.atomistic.RepresentativeStructure attribute*), 65
rc_volume (*httk.atomistic.representativestructure.RepresentativeStructure attribute*), 76
rc_volume (*httk.atomistic.Structure attribute*), 56
rc_volume (*httk.atomistic.structure.Structure attribute*), 84
read() (*in module httk.httkio.cif*), 128
read_config() (*in module httk.config.config*), 92
read_config() (*in module httk.httkweb.helpers*), 130
read_keys() (*in module httk.core.crypto*), 110
read_manifest() (*in module httk.task.reader*), 138
read_outcar() (*in module httk.iface.vasp_if*), 137
reader() (*in module httk.task.reader*), 138
receive() (*httk.external.command.Command method*), 126
reciprocal() (*httk.core.vectors.fracvector.FracVector method*), 98
reciprocal() (*httk.FracVector method*), 46
reduce_by_symops() (*in module httk.atomistic.spacegrouputils*), 79
reduced_coordgroups (*httk.atomistic.sites.Sites attribute*), 77
reduced_coordgroups_to_input() (*in module httk.iface.isotropy_if*), 133
reduced_coords (*httk.atomistic.sites.Sites attribute*), 77
reduced_to_cartesian() (*in module httk.atomistic.structureutils*), 88
Reference (*class in httk*), 42

Reference (*class in httk.core.reference*), 118
refresh() (*httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method*), 68
render_page() (*in module httk.httkweb.app_curses*), 129
RenderHttk (*class in httk.httkweb.render_httk*), 130
RenderRst (*class in httk.httkweb.render_rst*), 131
repeat() (*httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method*), 68
RepresentativeSites (*class in httk.atomistic*), 58
RepresentativeSites (*class in httk.atomistic.representativesites*), 74
RepresentativeStructure (*class in httk.atomistic*), 63
RepresentativeStructure (*class in httk.atomistic.representativestructure*), 75
reset() (*httk.db.filteredcollection.FilteredCollection method*), 123
Result (*class in httk*), 42
Result (*class in httk.core.computation*), 109
Result_RelaxedCellResult (*class in httk.atomistic.results.relaxedcellresult*), 67
Result_TotalEnergyResult (*class in httk.atomistic.results.totalenergyresult*), 67
retrieve() (*httk.db.storable.TrivialStore method*), 124
retrieve() (*httk.db.store.dictstore.DictStore method*), 120
retrieve() (*httk.db.store.sqlstore.SqlStore method*), 121
retrieve() (*httk.httkweb.webgenerator.WebGenerator method*), 132
rewind() (*httk.core.basic.rewindable_iterator method*), 107
rewindable_iterator (*class in httk.core.basic*), 107
right_punctuation_chars (*httk.httkweb.render_httk.RenderHttk attribute*), 130
rollback() (*httk.db.backend.sqlite.Sqlite method*), 120
rotate() (*httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method*), 68
rst_light_html_renderer() (*httk.httkweb.render_httk.RenderHttk method*), 130
rst_light_parse_textstyle() (*httk.httkweb.render_httk.RenderHttk method*), 130
rst_light_parser() (*httk.httkweb.render_httk.RenderHttk method*), 131
run() (*httk.external.command.Command method*), 126
run() (*in module httk.external.jmol*), 126
run_alot() (*in module httk.core.vectors.fracmath*), 94
run_app() (*in module httk.httkweb.app_qt5*), 129

S

save() (*httk.atomistic.atomisticio.structureioplugin.StructureIoPlugin method*), 66
save() (*httk.db.store.sqlstore.SqlStore method*), 121
save() (*in module httk*), 41
save() (*in module httk.httkio.save*), 129
save_and_quit() (*httk.atomistic.vis.jmolstructurevisualizer.JmolStructurevisualizer method*), 68
save_struct() (*in module httk.atomistic.atomisticio.structure_io*), 65
Scalar (*class in httk.core.vectors.vector*), 102
scalarmult() (*in module httk.core.ed25519*), 110
scale_to_vol() (*in module httk.atomistic.cellutils*), 73
scaling() (*httk.atomistic.Cell method*), 58
scaling() (*httk.atomistic.cell.Cell method*), 70
scaling() (*httk.atomistic.cellshape.CellShape method*), 71
scaling_to_volume() (*in module httk.atomistic.cellutils*), 73
searcher() (*httk.db.store.sqlstore.SqlStore method*), 121
send() (*httk.external.command.Command method*), 126
serve() (*in module httk.httkweb.serve*), 131
set_common_denom() (*httk.core.vectors.fracvector.FracVector class method*), 98
set_common_denom() (*httk.FracVector class method*), 47
set_defaults() (*httk.atomistic.vis.jmolstructurevisualizer.JmolStructu method*), 68
set_denominator() (*httk.core.vectors.fracvector.FracVector method*), 98
set_denominator() (*httk.FracVector method*), 47
set_hull_data() (*httk.analysis.matsci.phasediagram.PhaseDiagram method*), 52
set_inv() (*httk.core.vectors.mutablefracvector.MutableFracVector method*), 101
set_inv() (*httk.MutableFracVector method*), 49
set_mp_key() (*in module httk.external.pymatgen_glue*), 127
set_negative() (*httk.core.vectors.mutablefracvector.MutableFracVect method*), 101
set_negative() (*httk.MutableFracVector method*), 49
set_normalize() (*httk.core.vectors.mutablefracvector.MutableFracVe method*), 101
set_normalize() (*httk.MutableFracVector method*), 49

set_normalize_half() (*httk.core.vectors.mutablefracvector.MutableFracVector method*), 101
set_normalize_half() (*httk.MutableFracVector method*), 49
set_set_denominator() (*httk.core.vectors.mutablefracvector.MutableFracVector method*), 101
set_set_denominator() (*httk.MutableFracVector method*), 49
set_simplify() (*httk.core.vectors.mutablefracvector.MutableFracVector method*), 101
set_simplify() (*httk.MutableFracVector method*), 49
set_T() (*httk.core.vectors.mutablefracvector.MutableFracVector method*), 101
set_T() (*httk.MutableFracVector method*), 49
setup() (*in module httk.httkweb.helpers*), 130
setup_phasediagram() (*in module httk.atomistic.structurephasediagram*), 86
setup_template_helpers() (*in module httk.httkweb.helpers*), 130
sha256file() (*in module httk.core.crypto*), 110
show() (*httk.analysis.matsci.vis.phasediagramvisualizerplugin.PhaseDiagramVisualizerPlugin method*), 51
show() (*httk.atomistic.vis.asestructurevisualizer.AseStructureVisualizer method*), 67
show() (*httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method*), 68
show() (*httk.atomistic.vis.structurephasediagramvisualizerplugin.StructurePhaseDiagramVisualizerPlugin method*), 68
show() (*httk.atomistic.vis.structurevisualizerplugin.StructureVisualizerPlugin method*), 68
show() (*in module httk.external.gulp_ext*), 126
sign() (*httk.core.vectors.fracvector.FracVector method*), 98
sign() (*httk.FracVector method*), 47
sign() (*in module httk.core.vectors.vectormath*), 106
Signature (*class in httk*), 50
Signature (*class in httk.core.signature*), 118
signature() (*in module httk.core.ed25519*), 110
SignatureKey (*class in httk*), 50
SignatureKey (*class in httk.core.signature*), 118
simplex_le_solver() (*in module httk.core.geometry*), 111
simplify() (*httk.core.vectors.fracvector.FracVector method*), 98
simplify() (*httk.FracVector method*), 47
sin() (*httk.core.vectors.fracvector.FracVector method*), 98
sin() (*httk.FracVector method*), 47
sin() (*in module httk.core.vectors.vectormath*), 106
sinh() (*in module httk.core.vectors.vectormath*), 106
SiteAssignment (*class in httk.atomistic.siteassignment*), 77
Sites (*class in httk.atomistic.sites*), 77
sites_tidy() (*in module httk.atomistic.sitesutils*), 78
sites_to_platon() (*in module httk.iface.platon_if*), 136
sort_coordgroups() (*in module httk.atomistic.sitesutils*), 78
sort_coordgroups() (*in module httk.atomistic.structureutils*), 88
Spacegroup (*class in httk.atomistic.spacegroup*), 78
spacegroup (*httk.atomistic.Structure attribute*), 56
spacegroup (*httk.atomistic.structure.Structure attribute*), 84
spacegroup_filter() (*in module httk.atomistic.spacegrouputils*), 79
spacegroup_filter_specific() (*in module httk.atomistic.spacegrouputils*), 79
spacegroup_get_hall() (*in module httk.atomistic.data.spacegroups*), 66
spacegroup_get_hall() (*in module httk.atomistic.spacegrouputils*), 79
spacegroup_get_hm() (*in module httk.atomistic.data.spacegroups*), 66
spacegroup_get_hm() (*in module httk.atomistic.spacegrouputils*), 80
spacegroup_get_number() (*in module httk.atomistic.data.spacegroups*), 66
spacegroup_get_number() (*in module httk.atomistic.spacegrouputils*), 80
spacegroup_get_number_and_setting() (*in module httk.atomistic.data.spacegroups*), 66
spacegroup_get_number_and_setting() (*in module httk.atomistic.spacegrouputils*), 80
spacegroup_get_number_of_settings() (*in module httk.atomistic.data.spacegroups*), 66
spacegroup_get_schoenflies() (*in module httk.atomistic.data.spacegroups*), 66
spacegroup_get_schoenflies() (*in module httk.atomistic.spacegrouputils*), 80
spacegroup_number (*httk.atomistic.Structure attribute*), 56
spacegroup_number (*httk.atomistic.structure.Structure attribute*), 84
spacegroup_number_and_setting (*httk.atomistic.Structure attribute*), 56
spacegroup_number_and_setting (*httk.atomistic.structure.Structure attribute*), 84
spacegroup_parse() (*in module httk.atomistic.spacegrouputils*), 80
spglib_out_to_struct() (*in module httk.iface.spglib_if*), 137
spin() (*httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method*), 68
split_chars_strip_comments() (*in module*

httk.core.miniparser), 117
split_content() (*httk.httkweb.render_httk.RenderHttk method*), 131
sql() (*httk.db.filteredcollection.FCSqlite method*), 123
Sqlite (*class in httk.db.backend.sqlite*), 119
Sqlite.SqliteCursor (*class in httk.db.backend.sqlite*), 119
SqlStore (*class in httk.db.store.sqlstore*), 121
SqlStore.Keeper (*class in httk.db.store.sqlstore*), 121
sqrt() (*httk.core.vectors.fracvector.FracVector method*), 98
sqrt() (*httk.FracVector method*), 47
sqrt() (*in module httk.core.vectors.vectormath*), 106
stack_vecs() (*httk.core.vectors.fracvector.FracVector class method*), 98
stack_vecs() (*httk.core.vectors.vector.Vector class method*), 102
stack_vecs() (*httk.FracVector class method*), 47
standard_order_axes_transform() (*in module httk.atomistic.cellutils*), 73
standard_primitive() (*in module httk.external.aflow_ext*), 125
start() (*httk.external.command.Command method*), 126
start() (*in module httk.external.jmol*), 126
startup() (*in module httk.httkweb.serve*), 131
stdin (*httk.external.command.Command attribute*), 126
stop() (*httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method*), 68
stop() (*httk.external.command.Command method*), 126
Storable (*class in httk.db.storable*), 124
storable_init() (*httk.db.storable.Storable method*), 124
storable_props() (*in module httk.db.storable*), 124
storable_types() (*in module httk.db.storable*), 125
store() (*httk.db.httkobjdbplugin.HttkObjDbPlugin method*), 124
store_codependent_data() (*httk.db.httkobjdbplugin.HttkObjDbPlugin method*), 124
store_table() (*httk.db.filteredcollection.FCSqlite method*), 123
store_table() (*httk.db.filteredcollection.FilteredCollection method*), 123
string (*httk.core.ioadapters.IoAdapterString attribute*), 112
string (*httk.IoAdapterString attribute*), 50
string_to_val_and_delta() (*in module httk.core.vectors.fracmath*), 94
struct_process_with_isotropy() (*in module httk.external.isotropy_ext*), 126
struct_to_cif() (*in module httk.atomistic.atomisticio.structure_cif_io*), 65
struct_to_cif_httk_simplified() (*in module httk.atomistic.atomisticio.structure_cif_io*), 65
struct_to_cifdata() (*in module httk.atomistic.atomisticio.structure_cif_io*), 65
struct_to_input() (*in module httk.iface.isotropy_if*), 133
Structure (*class in httk.atomistic*), 52
Structure (*class in httk.atomistic.structure*), 80
structure_addsym_and_tidy() (*in module httk.external.platon_ext*), 127
structure_reduced_coordgroups_to_representative() (*in module httk.atomistic.sitesutils*), 78
structure_reduced_uc_to_representative() (*in module httk.atomistic.structureutils*), 88
structure_tidy() (*in module httk.atomistic.structureutils*), 88
structure_tidy() (*in module httk.external.platon_ext*), 127
structure_tidy_old() (*in module httk.external.platon_ext*), 127
structure_to_ase_atoms() (*in module httk.external.ase_glue*), 125
structure_to_comment() (*in module httk.iface.vasp_if*), 137
structure_to_gulp() (*in module httk.iface.gulp_if*), 133
structure_to_jmol() (*in module httk.iface.jmol_if*), 133
structure_to_p1structure() (*in module httk.atomistic.structureutils*), 88
structure_to_platon() (*in module httk.iface.platon_if*), 137
structure_to_poscar() (*in module httk.iface.vasp_if*), 137
structure_to_sgstructure() (*in module httk.atomistic.structureutils*), 88
structure_to_sgstructure() (*in module httk.external.platon_ext*), 127
structure_to_spglib_atoms() (*in module httk.external.pyspglib_ext*), 127
StructureAsePlugin (*class in httk.external.ase_glue*), 125
StructureFormulaPlugin (*class in httk.atomistic.formulautils*), 74
StructureIoPlugin (*class in httk.atomistic.atomisticio.structureioplugin*), 65
StructurePhaseDiagram (*class in httk.atomistic*), 60
StructurePhaseDiagram (*class in*

httk.atomistic.structurephasediagram), 86
StructurePhaseDiagramCompetingIndicies (*class in httk.atomistic.structurephasediagram*), 86
StructurePhaseDiagramVisualizerPlugin (*class in httk.atomistic.vis.structurephasediagramvisualizerplugin*), 68
StructureRef (*class in httk.atomistic*), 60
StructureRef (*class in httk.atomistic.structure*), 85
StructureSupercellPlugin (*class in httk.atomistic.supercellutils*), 88
StructureTag (*class in httk.atomistic*), 60
StructureTag (*class in httk.atomistic.structure*), 85
StructureVisualizerPlugin (*class in httk.atomistic.vis.structurevisualizerplugin*), 68
subdata() (*httk.db.filteredcollection.FCMultiDict method*), 122
submit_reader() (*in module httk.task.reader*), 138
submodule_import_external() (*in module httk.external.subimport*), 127
subtable() (*httk.db.filteredcollection.FCSqlite method*), 123
supercell (*httk.atomistic.Structure attribute*), 56
supercell (*httk.atomistic.structure.Structure attribute*), 84
supercell (*httk.atomistic.UnitcellStructure attribute*), 62
supercell (*httk.atomistic.unitcellstructure.UnitcellStructure attribute*), 91
symbol (*httk.atomistic.assignment.Assignment attribute*), 69
symbol (*httk.atomistic.siteassignment.SiteAssignment attribute*), 77
symbollists (*httk.atomistic.Assignments attribute*), 59
symbollists (*httk.atomistic.assignments.Assignments attribute*), 69
symbols (*httk.atomistic.Assignments attribute*), 59
symbols (*httk.atomistic.assignments.Assignments attribute*), 69
symbols (*httk.atomistic.siteassignment.SiteAssignment attribute*), 77
symbols (*httk.atomistic.Structure attribute*), 56
symbols (*httk.atomistic.structure.Structure attribute*), 84
symopshash() (*in module httk.atomistic.spacegrouputils*), 80
symopsmatrix() (*in module httk.atomistic.spacegrouputils*), 80
symopstuple() (*in module httk.atomistic.spacegrouputils*), 80

T

T() (*httk.core.vectors.fracvector.FracVector method*), 95
T() (*httk.FracVector method*), 43
table() (*httk.db.filteredcollection.FCSqlite method*), 123
table_exists() (*httk.db.backend.sqlite.Sqlite method*), 120
TableColumn (*class in httk.db.filteredcollection*), 123
tan() (*in module httk.core.vectors.vectormath*), 106
tanh() (*in module httk.core.vectors.vectormath*), 106
TemplateEngineHttk (*class in httk.httkweb.templateengine_httk*), 131
TemplateEngineTemplator (*class in httk.httkweb.templateengine_templator*), 131
text() (*httk.httkweb.app_curses.MyHTMLParser method*), 129
tidy() (*httk.atomistic.RepresentativeSites method*), 59
tidy() (*httk.atomistic.representativesites.RepresentativeSites method*), 75
tidy() (*httk.atomistic.Structure method*), 56
tidy() (*httk.atomistic.structure.Structure method*), 84
to() (*httk.core.httkobject.HttkObject method*), 111
to() (*httk.HttkObject method*), 50
to_Atoms() (*httk.external.ase_glue.StructureAsePlugin method*), 125
to_basis() (*httk.atomistic.Assignments method*), 59
to_basis() (*httk.atomistic.assignments.Assignments method*), 69
to_basis() (*httk.atomistic.siteassignment.SiteAssignment method*), 77
to_float() (*httk.core.vectors.fracvector.FracVector method*), 99
to_float() (*httk.FracVector method*), 47
to_floats() (*httk.core.vectors.fracvector.FracVector method*), 99
to_floats() (*httk.FracVector method*), 47
to_fraction() (*httk.core.vectors.fracvector.FracVector method*), 99
to_fraction() (*httk.FracVector method*), 47
to_fractions() (*httk.core.vectors.fracvector.FracVector method*), 99
to_fractions() (*httk.FracVector method*), 47
to_FracVector() (*httk.core.vectors.mutablefracvector.MutableFracVec method*), 101
to_FracVector() (*httk.MutableFracVector method*), 49
to_int() (*httk.core.vectors.fracvector.FracVector method*), 99
to_int() (*httk.FracVector method*), 47
to_ints() (*httk.core.vectors.fracvector.FracVector method*), 99
to_ints() (*httk.FracVector method*), 47
to_string() (*httk.core.vectors.fracvector.FracVector method*), 99

to_string() (*httk.FracVector method*), 47
to_strings() (*httk.core.vectors.fracvector.FracVector method*), 99
to_strings() (*httk.FracVector method*), 47
to_tuple() (*httk.core.httkobject.HttkObject method*), 111
to_tuple() (*httk.core.vectors.fracvector.FracVector method*), 99
to_tuple() (*httk.FracVector method*), 48
to_tuple() (*httk.HttkObject method*), 50
total_number_of_atoms (*httk.atomistic.RepresentativeSites attribute*), 59
total_number_of_atoms (*httk.atomistic.representativesites.RepresentativeSites attribute*), 75
total_number_of_atoms (*httk.atomistic.sites.Sites attribute*), 77
total_number_of_atoms (*httk.atomistic.UnitcellSites attribute*), 59
total_number_of_atoms (*httk.atomistic.unitcellsites.UnitcellSites attribute*), 89
transform() (*httk.atomistic.Structure method*), 56
transform() (*httk.atomistic.structure.Structure method*), 84
transform() (*httk.atomistic.UnitcellStructure method*), 62
transform() (*httk.atomistic.unitcellstructure.UnitcellStructure method*), 91
transform() (*in module httk.atomistic.structureutils*), 88
trivial_symmetry_reduce() (*in module httk.atomistic.spacegrouputils*), 80
TrivialStore (*class in httk.db.storable*), 124
trivialstore (*httk.db.storable.Storable attribute*), 124
trunc() (*in module httk.core.vectors.vectormath*), 106
tuple_eye() (*in module httk.core.vectors.fracvector*), 100
tuple_eye() (*in module httk.core.vectors.vector*), 103
tuple_index() (*in module httk.core.vectors.fracvector*), 100
tuple_index() (*in module httk.core.vectors.vector*), 103
tuple_random() (*in module httk.core.vectors.fracvector*), 100
tuple_random() (*in module httk.core.vectors.vector*), 103
tuple_slice() (*in module httk.core.vectors.fracvector*), 100
tuple_slice() (*in module httk.core.vectors.vector*), 103
tuple_to_hexhash() (*in module httk.core.crypto*), 110
tuple_to_str() (*in module httk.core.basic*), 107
tuple_to_str() (*in module httk.core.crypto*), 110
tuple_zeros() (*in module httk.core.vectors.fracvector*), 100
tuple_zeros() (*in module httk.core.vectors.vector*), 103
types() (*httk.core.httkobject.HttkObject class method*), 111
types() (*httk.HttkObject class method*), 50

U

uc (*httk.atomistic.Structure attribute*), 56
uc (*httk.atomistic.structure.Structure attribute*), 84
uc_a (*httk.atomistic.Structure attribute*), 56
uc_a (*httk.atomistic.structure.Structure attribute*), 84
uc_a (*httk.atomistic.UnitcellStructure attribute*), 63
uc_a (*httk.atomistic.unitcellstructure.UnitcellStructure attribute*), 91
uc_alpha (*httk.atomistic.Structure attribute*), 56
uc_alpha (*httk.atomistic.structure.Structure attribute*), 84
uc_alpha (*httk.atomistic.UnitcellStructure attribute*), 63
uc_alpha (*httk.atomistic.unitcellstructure.UnitcellStructure attribute*), 91
uc_b (*httk.atomistic.Structure attribute*), 56
uc_b (*httk.atomistic.structure.Structure attribute*), 84
uc_b (*httk.atomistic.UnitcellStructure attribute*), 63
uc_b (*httk.atomistic.unitcellstructure.UnitcellStructure attribute*), 91
uc_basis (*httk.atomistic.Structure attribute*), 56
uc_basis (*httk.atomistic.structure.Structure attribute*), 84
uc_basis (*httk.atomistic.UnitcellStructure attribute*), 63
uc_basis (*httk.atomistic.unitcellstructure.UnitcellStructure attribute*), 91
uc_beta (*httk.atomistic.Structure attribute*), 56
uc_beta (*httk.atomistic.structure.Structure attribute*), 84
uc_beta (*httk.atomistic.UnitcellStructure attribute*), 63
uc_beta (*httk.atomistic.unitcellstructure.UnitcellStructure attribute*), 91
uc_c (*httk.atomistic.Structure attribute*), 56
uc_c (*httk.atomistic.structure.Structure attribute*), 84
uc_c (*httk.atomistic.UnitcellStructure attribute*), 63
uc_c (*httk.atomistic.unitcellstructure.UnitcellStructure attribute*), 91
uc_cartesian_coordgroups (*httk.atomistic.Structure attribute*), 57
uc_cartesian_coordgroups (*httk.atomistic.structure.Structure attribute*), 85

uc_cartesian_coordgroups (*httk.atomistic.UnitcellStructure attribute*), 63
uc_cartesian_coordgroups (*httk.atomistic.unitcellstructure.UnitcellStructure attribute*), 91
uc_cartesian_coords (*httk.atomistic.Structure attribute*), 57
uc_cartesian_coords (*httk.atomistic.structure.Structure attribute*), 85
uc_cartesian_coords (*httk.atomistic.UnitcellStructure attribute*), 63
uc_cartesian_coords (*httk.atomistic.unitcellstructure.UnitcellStructure attribute*), 91
uc_cartesian_occupationscoords (*httk.atomistic.Structure attribute*), 57
uc_cartesian_occupationscoords (*httk.atomistic.structure.Structure attribute*), 85
uc_cartesian_occupationscoords (*httk.atomistic.UnitcellStructure attribute*), 63
uc_cartesian_occupationscoords (*httk.atomistic.unitcellstructure.UnitcellStructure attribute*), 91
uc_cell (*httk.atomistic.Structure attribute*), 57
uc_cell (*httk.atomistic.structure.Structure attribute*), 85
uc_cell_orientation (*httk.atomistic.Structure attribute*), 57
uc_cell_orientation (*httk.atomistic.structure.Structure attribute*), 85
uc_cell_orientation (*httk.atomistic.UnitcellStructure attribute*), 63
uc_cell_orientation (*httk.atomistic.unitcellstructure.UnitcellStructure attribute*), 91
uc_counts (*httk.atomistic.Structure attribute*), 57
uc_counts (*httk.atomistic.structure.Structure attribute*), 85
uc_counts (*httk.atomistic.UnitcellStructure attribute*), 63
uc_counts (*httk.atomistic.unitcellstructure.UnitcellStructure attribute*), 91
uc_formula (*httk.atomistic.Structure attribute*), 57
uc_formula (*httk.atomistic.structure.Structure attribute*), 85
uc_formula_counts (*httk.atomistic.Structure attribute*), 57
uc_formula_counts (*httk.atomistic.structure.Structure attribute*), 85
uc_formula_parts (*httk.atomistic.Structure attribute*), 57
uc_formula_parts (*httk.atomistic.structure.Structure attribute*), 85
uc_formula_symbols (*httk.atomistic.Structure attribute*), 57
uc_formula_symbols (*httk.atomistic.structure.Structure attribute*), 85
uc_gamma (*httk.atomistic.Structure attribute*), 57
uc_gamma (*httk.atomistic.structure.Structure attribute*), 85
uc_gamma (*httk.atomistic.UnitcellStructure attribute*), 63
uc_gamma (*httk.atomistic.unitcellstructure.UnitcellStructure attribute*), 91
uc_lengths_and_angles (*httk.atomistic.Structure attribute*), 57
uc_lengths_and_angles (*httk.atomistic.structure.Structure attribute*), 85
uc_lengths_and_angles (*httk.atomistic.UnitcellStructure attribute*), 63
uc_lengths_and_angles (*httk.atomistic.unitcellstructure.UnitcellStructure attribute*), 91
uc_nbr_atoms (*httk.atomistic.Structure attribute*), 57
uc_nbr_atoms (*httk.atomistic.structure.Structure attribute*), 85
uc_occupancies (*httk.atomistic.Structure attribute*), 57
uc_occupancies (*httk.atomistic.structure.Structure attribute*), 85
uc_occupationssymbols (*httk.atomistic.Structure attribute*), 57
uc_occupationssymbols (*httk.atomistic.structure.Structure attribute*), 85
uc_reduced_coordgroups (*httk.atomistic.Structure attribute*), 57
uc_reduced_coordgroups (*httk.atomistic.structure.Structure attribute*), 85
uc_reduced_coordgroups (*httk.atomistic.UnitcellStructure attribute*), 63
uc_reduced_coordgroups (*httk.atomistic.unitcellstructure.UnitcellStructure attribute*), 91
uc_reduced_coordgroups_process_with_isotropy() (*in module httk.external.isotropy_ext*), 126
uc_reduced_coords (*httk.atomistic.Structure attribute*), 57
uc_reduced_coords (*httk.atomistic.structure.Structure attribute*), 85
uc_reduced_coords (*httk.atomistic.UnitcellStructure attribute*), 63

uc_reduced_coords (*httk.atomistic.unitcellstructure.UnitcellStructure attribute*), 91
uc_reduced_occupationscoords (*httk.atomistic.Structure attribute*), 57
uc_reduced_occupationscoords (*httk.atomistic.structure.Structure attribute*), 85
uc_sites (*httk.atomistic.Structure attribute*), 57
uc_sites (*httk.atomistic.structure.Structure attribute*), 85
uc_structure_to_symbols_and_scaled_positions() (*in module httk.iface.ase_if*), 132
uc_volume (*httk.atomistic.Structure attribute*), 57
uc_volume (*httk.atomistic.structure.Structure attribute*), 85
uc_volume (*httk.atomistic.UnitcellStructure attribute*), 63
uc_volume (*httk.atomistic.unitcellstructure.UnitcellStructure attribute*), 91
uc_volume_per_atom (*httk.atomistic.RepresentativeStructure attribute*), 65
uc_volume_per_atom (*httk.atomistic.representativestructure.RepresentativeStructure attribute*), 76
uc_volume_per_atom (*httk.atomistic.UnitcellStructure attribute*), 63
uc_volume_per_atom (*httk.atomistic.unitcellstructure.UnitcellStructure attribute*), 91
UnaryBooleanOp (*class in httk.db.filteredcollection*), 123
UnitcellSites (*class in httk.atomistic*), 59
UnitcellSites (*class in httk.atomistic.unitcellsites*), 89
UnitcellStructure (*class in httk.atomistic*), 60
UnitcellStructure (*class in httk.atomistic.unitcellstructure*), 89
universal_opener() (*in module httk.core.ioadapters*), 113
UnquotedStr (*class in httk.httkweb.helpers*), 130
update() (*httk.db.backend.sqlite.Sqlite method*), 120
update_metadata() (*httk.httkweb.webgenerator.Page method*), 132
update_row() (*httk.db.backend.sqlite.Sqlite method*), 120
use() (*httk.atomistic.assignment.Assignment class method*), 69
use() (*httk.atomistic.Assignments class method*), 59
use() (*httk.atomistic.assignments.Assignments class method*), 69
use() (*httk.atomistic.Cell class method*), 58
use() (*httk.atomistic.cell.Cell class method*), 70
use() (*httk.atomistic.RepresentativeStructure class method*), 65
use() (*httk.atomistic.representativestructure.RepresentativeStructure class method*), 76
use() (*httk.atomistic.siteassignment.SiteAssignment class method*), 77
use() (*httk.atomistic.sites.Sites class method*), 77
use() (*httk.atomistic.Structure class method*), 57
use() (*httk.atomistic.structure.Structure class method*), 85
use() (*httk.atomistic.UnitcellStructure class method*), 63
use() (*httk.atomistic.unitcellstructure.UnitcellStructure class method*), 91
use() (*httk.core.httkobject.HttkObject class method*), 111
use() (*httk.core.ioadapters.IoAdapterFileAppender class method*), 112
use() (*httk.core.ioadapters.IoAdapterFilename class method*), 112
use() (*httk.core.ioadapters.IoAdapterFileReader class method*), 112
use() (*httk.core.ioadapters.IoAdapterFileWriter class method*), 112
use() (*httk.core.ioadapters.IoAdapterString class method*), 113
use() (*httk.core.ioadapters.IoAdapterStringList class method*), 113
use() (*httk.core.vectors.fracvector.FracVector class method*), 99
use() (*httk.core.vectors.mutablefracvector.MutableFracVector class method*), 101
use() (*httk.core.vectors.vector.Vector class method*), 102
use() (*httk.FracVector class method*), 48
use() (*httk.HttkObject class method*), 50
use() (*httk.IoAdapterFileAppender class method*), 50
use() (*httk.IoAdapterFileReader class method*), 49
use() (*httk.IoAdapterFileWriter class method*), 49
use() (*httk.IoAdapterString class method*), 50
use() (*httk.IoAdapterStringList class method*), 50
use() (*httk.MutableFracVector class method*), 49

V

val_to_tuple() (*in module httk.atomistic.spacegrouputils*), 80
validate() (*httk.core.vectors.fracvector.FracVector method*), 99
validate() (*httk.core.vectors.mutablefracvector.MutableFracVector method*), 101
validate() (*httk.FracVector method*), 48
validate() (*httk.MutableFracVector method*), 49

variable() (*httk.db.filteredcollection.FilteredCollection method*), 123
variable() (*httk.db.storable.Storable class method*), 124
Vector (*class in httk.core.vectors.vector*), 102
verify_crytpo_signature() (*in module httk.core.crypto*), 110
verify_crytpo_signature_old() (*in module httk.core.crypto*), 110
vformat() (*httk.httkweb.templateengine_httk.HttkTemplateFormatter method*), 131
vis (*httk.analysis.matsci.phasediagram.PhaseDiagram attribute*), 52
vol_to_scale() (*in module httk.atomistic.cellutils*), 73
volume (*httk.atomistic.Cell attribute*), 58
volume (*httk.atomistic.cell.Cell attribute*), 70
volume_per_atom (*httk.atomistic.Structure attribute*), 57
volume_per_atom (*httk.atomistic.structure.Structure attribute*), 85

W

wait() (*httk.analysis.matsci.vis.phasediagramvisualizerplugin.PhaseDiagramVisualizerPlugin method*), 51
wait() (*httk.atomistic.vis.asestructurevisualizer.AseStructureVisualizer method*), 67
wait() (*httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method*), 68
wait() (*httk.atomistic.vis.structurevisualizerplugin.StructureVisualizerPlugin method*), 68
wait_finish() (*httk.external.command.Command method*), 126
WebError, 131
WebGenerator (*class in httk.httkweb.webgenerator*), 132
WebviewCurses (*class in httk.httkweb.app_curses*), 129
write_cif() (*in module httk.httkio.cif*), 128
write_generic_kpoints_file() (*in module httk.iface.vasp_if*), 137
write_kpoints_file() (*in module httk.iface.vasp_if*), 137
write_poscar() (*in module httk.iface.vasp_if*), 137
wyckoff_sequence (*httk.atomistic.Compound attribute*), 60
wyckoff_sequence (*httk.atomistic.compound.Compound attribute*), 73
wyckoff_sequence (*httk.atomistic.RepresentativeSites attribute*), 59
wyckoff_sequence (*httk.atomistic.representativesites.RepresentativeSites attribute*), 75
wyckoff_sequence (*httk.atomistic.Structure attribute*), 57
wyckoff_sequence (*httk.atomistic.structure.Structure attribute*), 85
wyckoff_symbol_matcher() (*in module httk.atomistic.spacegrouputils*), 80

X

xrecover() (*in module httk.core.ed25519*), 110

Z

zdecompressor() (*in module httk.core.ioadapters*), 113
zeros() (*httk.core.vectors.fracvector.FracVector class method*), 99
zeros() (*httk.core.vectors.vector.Vector class method*), 103
zeros() (*httk.FracVector class method*), 48