

---

# httk Documentation

*Release 1.1.11*

**Author**

**Sep 24, 2018**



---

## Contents

---

<b>1</b>	<b>About the High-Throughput Toolkit</b>	<b>3</b>
<b>2</b>	<b>Getting started with <i>httk</i></b>	<b>5</b>
<b>3</b>	<b>Reporting bugs</b>	<b>7</b>
<b>4</b>	<b>Developing / contributing to <i>httk</i></b>	<b>9</b>
<b>5</b>	<b>Citing <i>httk</i> in scientific works</b>	<b>11</b>
<b>6</b>	<b>Contributors</b>	<b>13</b>
<b>7</b>	<b>Acknowledgements</b>	<b>15</b>
<b>8</b>	<b>License and redistribution</b>	<b>17</b>
<b>9</b>	<b>Contact</b>	<b>19</b>
<b>10</b>	<b>Full API reference</b>	<b>21</b>
	<b>Python Module Index</b>	<b>151</b>



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



---

## About the High-Throughput Toolkit

---

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. *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.



---

### Getting started with *httk*

---

## 2.1 Download

The *httk* is freely available under an open source license, the GNU Affero General Public License.

The latest release of *httk* is 1.1.11.

**Download the latest release at github here:** <https://github.com/rartino/httk/releases/latest>

## 2.2 Installation

Installation information is found in the *httk Installation Instructions*.

## 2.3 User's guide

For information on basic usage of the *httk*, see *httk Users' Guide*.

More tricky details on how high-throughput computational tasks are executed via the `runmanager.sh` program are presented in *httk Runmanager Details*. This is useful if you plan to write your own intricate run-scripts using *httk*.



## 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

---

### Developing / contributing to *httk*

---

Read the *httk Developers' Guide*



---

### 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 the *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 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).



---

### 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 htk thus cannot be imported into GPL licensed projects.)

The full license text is present in [htk 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 htk API documentation*
- genindex
- modindex
- search

### 10.1 htk 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/>](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 immutable 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_struct = 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 itself.

## 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 UnMutableVec-tor 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 httk Downloads

The *httk* is freely available under an open source license, the GNU Affero General Public License.

The latest release of httk is 1.1.11.

**Download the latest release at github here:** <https://github.com/rartino/httk/releases/latest>

## 10.4 Full httk API documentation

Contents:

### 10.4.1 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, rel-  
path, 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, jour-  
nal_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.,  $A \times B$ .

**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.,  $A \cdot B$  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,  $A^{-1}$

**lengthsq** ()  
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 be 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  $\pm N$  to each element to place it in the range  $[0,1)$

**set\_normalize\_half** ()

Add/remove an integer  $\pm N$  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.HttpObject
    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.HttpkPluginWrapper(plugin=None)
    Bases: object

class httk.HttpkPlugin
    Bases: object

class httk.HttpkPluginPlaceholder(plugininfo=None)
    Bases: object

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

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

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

    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.HttpkPlugin

    params()

    plugin_init(phasediagram)

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

    wait()
```

## Submodules

### httpk.analysis.matsci.phasediagram module

```

class httpk.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
    
```

### httpk.atomistic package

The httpk.atomistic package

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

```

class httpk.atomistic.Structure (assignments, rc_sites=None, rc_cell=None, other_reps=None)
    Bases: httpk.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 httpk.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

pbv = '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 accidentally 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\_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, anonym-
                               mous_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, com-
                                             peting_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, nonperi-
odic_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 sites 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 accidentally 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, space-
group=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 accidentally 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.atomistico package

## Submodules

### httk.atomistic.atomistico.structure\_cif\_io module

```

httk.atomistic.atomistico.structure_cif_io.cif_reader_httk_preprocessed(ioa)
httk.atomistic.atomistico.structure_cif_io.cif_reader_that_can_only_read_isotropy_cif(ioa)
httk.atomistic.atomistico.structure_cif_io.cif_to_struct(ioa, back-
                                                         ends=['internal',
                                                         'cif2cell', 'ase', 'platon'])
httk.atomistic.atomistico.structure_cif_io.cifdata_to_struct(cifdata, de-
                                                         bug=False)
httk.atomistic.atomistico.structure_cif_io.struct_to_cif(struct, ioa, back-
                                                         ends=['httk'])
httk.atomistic.atomistico.structure_cif_io.struct_to_cif_httk_simplified(struct,
                                                                           ioa,
                                                                           header=None,
                                                                           symops=True)
httk.atomistic.atomistico.structure_cif_io.struct_to_cifdata(struct, en-
                                                         tryid=None)

```

## httk.atomistic.atomistico.structure\_io module

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

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

## httk.atomistic.atomistico.structureioplugin module

**class** `httk.atomistic.atomistico.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,  
                                                                           com-  
                                                                           pound,  
                                                                           re-  
                                                                           laxed_structure,  
                                                                           prim-  
                                                                           i-  
                                                                           tive_cell,  
                                                                           vol-  
                                                                           ume_per_atom,  
                                                                           min-  
                                                                           i-  
                                                                           mum_energy)
```

Bases: *httk.core.computation.Result*

#### httk.atomistic.results.totalenergyresult module

```
class httk.atomistic.results.totalenergyresult.Result_TotalEnergyResult (computation,  
                                                                           struc-  
                                                                           ture,  
                                                                           to-  
                                                                           tal_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.StructurePhaseDiagramVisual1
    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, mag-
    netic_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’, ‘orthorhombic’, ‘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’, ‘orthorhombic’, ‘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*,  
*cosan-*  
*gles*,  
*lat-*  
*tice\_system=None*,  
*orien-*  
*ta-*  
*tion=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  $\neq$  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*,  
*lat-*  
*tice\_system=None*,  
*orien-*  
*tation=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  $\beta \neq 90$  degrees.

```
httpk.atomistic.cellutils.niggli_to_lengths_and_angles (niggli_matrix)
httpk.atomistic.cellutils.niggli_to_lengths_and_trigangles (niggli_matrix)
httpk.atomistic.cellutils.niggli_to_metric (niggli)
httpk.atomistic.cellutils.scale_to_vol (basis, scale)
httpk.atomistic.cellutils.scaling_to_volume (basis, scaling)
httpk.atomistic.cellutils.standard_order_axes_transform (niggli_matrix,          lat-
                                                         tice_system, eps=0, re-
                                                         turn_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`,

```
httpk.atomistic.cellutils.vol_to_scale (basis, vol)
```

## httpk.atomistic.cli module

```
httpk.atomistic.cli.main (commands, args)
```

## httpk.atomistic.compound module

```
class httpk.atomistic.compound.Compound (element_wyckoff_sequence, formula, space-
                                           group_number, extended, extensions, wyck-
                                           off_sequence, anonymous_wyckoff_sequence,
                                           anonymous_formula, formula_symbols, for-
                                           mula_counts, pbc)
```

Bases: `httpk.core.httkobject.HttpkObject`

```
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 httpk.atomistic.compound.CompoundName(compound, name)
    Bases: httpk.core.httkobject.HttkObject

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

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

    classmethod create(compound, structure)

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

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

    classmethod create(computation, compound)

httpk.atomistic.compound.main()
    
```

### httpk.atomistic.formulautils module

```

class httpk.atomistic.formulautils.StructureFormulaPlugin
    Bases: httpk.core.httkobject.HttkPlugin

    plugin_init(struct)
    
```

### httpk.atomistic.representativesites module

```

class httpk.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: httpk.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, spacegroupnum-  
ber=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, space-  
group=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 accidentally 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, oc-  
cu-  
pan-  
cies*)

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.**pbcs\_to\_nonperiodic\_vecs** (*pbcs*)

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, space-  
group, back-  
ends=['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, spacegroupnum-
                        ber=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 (space-
                      group 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 (spacegroupnbr)
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, space-
                    group=None, hall_symbol=None, spacegroupnumber=None, set-
                    ting=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, vol-
                    ume_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 accidentally 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**

**pbcr**

**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_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.StructurePhaseDiagramCompetingIndices(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,
                                                                        assignments,
                                                                        tobj)

    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,
                                                                    back-
                                                                    ends=['cif2cell',
                                                                    'in-
                                                                    ter-
                                                                    nal',
                                                                    'ase'])

httk.atomistic.structureutils.coordgroups_reduced_uc_to_representative(coordgroups,
                                                                    ba-
                                                                    sis,
                                                                    back-
                                                                    ends=['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,
                                                                    oc-
                                                                    cu-
                                                                    pan-
                                                                    cies)

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,
                                                                    ba-
                                                                    sis,
                                                                    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 (occupations,
                                                                                          coordgroups)
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,
                                                                       backends=['isotropy',
                                                                       'fake'])
httk.atomistic.structureutils.structure_tidy (struct, backends=['platon'])
httk.atomistic.structureutils.structure_to_plstructure (struct, backends=['ase'])
httk.atomistic.structureutils.structure_to_sgstructure (struct, backends=['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, tolerance=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, tolerance=None,
                                                              max_search_cells=1000)
httk.atomistic.supercellutils.orthogonal_supercell_transformation (structure, tolerance=None,
                                                                    ortho=[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

pbv = '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, nonperi-
odic_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 sites 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 accidentally 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**

**ttransform** (*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

httk core module

Basic utilities and data definitions that are used throughout the httk code.

**A few of the most important components:**

**fracvector:** our general matrix object used to allow exact representation of arrays to allow, e.g., exact matching of coordinates to existing structures in the database.

ioadapters: our classes for generic handling of IO to files, streams, etc.

structure: our basic definition of a “structure of atoms”

## 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 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_cos(x, prec=Fraction(1, 10000000000), limit=True, de-
                                   grees=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_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.
```

```
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_pi(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, de-
                                   grees=False)
```

```
httk.core.vectors.fracmath.frac_sqrt(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)
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.,  $A \times B$ .

**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.,  $A \cdot B$  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 be 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 nested 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  $\pm N$  to each element to place it in the range  $[0,1)$

**set\_normalize\_half** ()  
Add/remove an integer  $\pm N$  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 nested 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  $0.5 \leq \text{abs}(m) < 1$ .

(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,  $\sqrt{x^2 + y^2}$ . 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  $\log(x)/\log(\text{base})$ .  
 (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_idx`(*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_sequence(arg)`

`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_path)
```

```
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_crypto_signature (signature, message, public_key)
```

```
httk.core.crypto.verify_crypto_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.scalar_mult(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 tetrahedra spanned by the coordinates in tetra

httk.core.geometry.numpy_quickhull_2d(sample)

httk.core.geometry.simplex_le_solver(a, b, c)
    Minimize func =  $a[0]*x + a[1]*y + a[2]*z + \dots$  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, deletefile-
                                                name=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.universal_opener (other)
httk.core.ioadapters.zdecompressor (f, mode, *args)
    Read a classic unix compress .Z type file.
```

## 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, outputpath, template_suffixes='template', envglob-
                                     als=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: subdi-
    rectory 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 retrieval 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)
```

```
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.) Validity/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') myFC-
MultiDict.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:

```
language = FCMultiDict('programming':programming_fc, 'review':review_fc) language
= language.data('programming').language review = language.data('review') myFC-
MultiDict.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 `FilteredCollection.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 `FilteredCollection.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*, *class-*  
*ref=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, retrieval, etc. Class object representing data should inherit from Storable.

All public variables must be initialized 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*)

Convenience method to do a very simple search of type: find all entries where member = value.

**classmethod** **find\_one** (*obj*, *store*, *member*, *value*, *types*)

Convenience^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_spplib_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 'some 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 'some 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.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
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)
-----
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

```

(continues on next page)

(continued from previous page)

-	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
```

(continues on next page)

(continued from previous page)

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, templatenam*)

`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*)

**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)
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.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
                                    ate_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

## 10.4.2 Indices and tables

- [genindex](#)
- [modindex](#)
- [search](#)

## 10.5 *httk* Installation Instructions

### 10.5.1 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 release (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 source archive

Find the latest release download at this link: <https://github.com/rartino/httk/releases/latest>, and get the link to the `.tar.gz` archive.

Run the following in a terminal:

```
mkdir -p ~/bin/python
cd ~/bin/python
curl -L <download link> --output httk-<version>.tar.gz
tar -zxf httk-<version>.tar.gz
rm -f httk-<version>.tar.gz
```

where you have to fill in <download link> and <version> according to the release page.

The archive extraction (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-1.0.0`, then do the following:

```
ln -f -s httk-1.0.0 httk-latest
source ~/bin/python/httk-latest/setup.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 variable.) 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 release 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., `v1.0.1`):

```
cp httk-latest/httk.cfg httk-1.0.1/httk.cfg
ln -f -s httk-1.0.1 ../httk-latest
```

This concludes the upgrade.

## 10.5.2 Windows

These instructions may be expanded in the future. For now, what you need to do is download cygwin and when asked what software to install, include

wget, python

After cygwin is installed, start a cygwin terminal and follow the instructions above.

## 10.5.3 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)

## 10.6 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](#).

### 10.6.1 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)

### 10.6.2 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.

### 10.6.3 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.

### 10.6.4 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.

## 10.6.5 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)
```

## 10.6.6 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, rel-
                        path, 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, jour-
                        nal_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)
```

## 10.6.7 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

## 10.6.8 Full documentation

For full documentation, see [Full httk API documentation](#).

## 10.7 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](#).

### 10.7.1 Introduction

The httk.atomistic package

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

### 10.7.2 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_coors=None, cartesian_coors=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_coors=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

## 10.8 Publications

Publications using, or otherwise relating, to the httk framework

## 10.9 Getting started with *httk*

### 10.9.1 Download

The *httk* is freely available under an open source license, the GNU Affero General Public License.

The latest release of httk is 1.1.11.

**Download the latest release at github here:** <https://github.com/rartino/httk/releases/latest>

### 10.9.2 Installation

Installation information is found in the *httk Installation Instructions*.

### 10.9.3 User's guide

For information on basic usage of the *httk*, see *httk Users' Guide*.

More tricky details on how high-throughput computational tasks are executed via the `runmanager.sh` program are presented in *httk Runmanager Details*. This is useful if you plan to write your own intricate run-scripts using *httk*.

## 10.10 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.

## 10.11 Developing / contributing to *httk*

Read the *httk Developers' Guide*

## 10.12 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 the *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.

## 10.13 Contributors

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

## 10.14 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).

## 10.15 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](#).

## 10.16 Contact

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

## 10.17 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.

### 10.17.1 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.

### 10.17.2 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.

### 10.17.3 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.

### 10.17.4 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-initialize 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.

### 10.17.5 ‘ht\_steps’ subtasks

In a ht\_steps script one can create ‘subtasks’. This is done 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 subtasks, 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'.)

### 10.17.6 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.

### 10.17.7 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'.

### 10.17.8 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.

### 10.17.9 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.

### 10.17.10 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

### 10.17.11 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 abandoned 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 appropriate part of the run, writes the *ht.status* file, and exits with an appropriate 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. taskmanager.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. `taskmanager.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 abandoned 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.

## 10.18 *httk* Users' Guide

### 10.18.1 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.

### 10.18.2 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.)

### 10.18.3 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.

### 10.18.4 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.

### 10.18.5 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.)

## 10.18.6 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 project*.
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](mailto:contact@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.)

## 10.19 *httk* Contributors

### Programming:

- Rickard Armiento, Linköping University, Sweden ([ricard \[at\] ifm.liu.se](mailto:ricard@ifm.liu.se))
- Christopher Tholander, Linköping University, Sweden.

Some parts of *httk* related to reading structures are heavily inspired by corresponding code in *cif2cell* by Torbjörn Björkman (Aalto University, Finland).

### Database and API design:

- Rickard Armiento
- Peter Steneteg
- Igor Mogyasinz

## 10.19.1 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.

### h

httk, 39  
 httk.analysis, 49  
 httk.analysis.matsci, 49  
 httk.analysis.matsci.phasediagram, 50  
 httk.analysis.matsci.vis, 49  
 httk.analysis.matsci.vis.phasediagramvisualizerplugin, 49  
 httk.atomistic, 50  
 httk.atomistic.assignment, 66  
 httk.atomistic.assignments, 67  
 httk.atomistic.atomistico, 63  
 httk.atomistic.atomistico.structure\_cif\_io, 63  
 httk.atomistic.atomistico.structure\_io, 64  
 httk.atomistic.atomistico.structureioplugn, 64  
 httk.atomistic.cell, 67  
 httk.atomistic.cellshape, 69  
 httk.atomistic.cellutils, 69  
 httk.atomistic.cli, 71  
 httk.atomistic.compound, 71  
 httk.atomistic.data, 64  
 httk.atomistic.data.periodictable, 64  
 httk.atomistic.data.spacegroups, 64  
 httk.atomistic.formulautils, 72  
 httk.atomistic.representativesites, 72  
 httk.atomistic.representativestructure, 73  
 httk.atomistic.results, 65  
 httk.atomistic.results.relaxedcellresult, 65  
 httk.atomistic.results.totalenergyresult, 65  
 httk.atomistic.siteassignment, 75  
 httk.atomistic.sites, 75  
 httk.atomistic.sitesutils, 76  
 httk.atomistic.spacegroup, 77  
 httk.atomistic.spacegrouputils, 77  
 httk.atomistic.structure, 78  
 httk.atomistic.structurephasediagram, 84  
 httk.atomistic.structureutils, 84  
 httk.atomistic.supercellutils, 86  
 httk.atomistic.unitcellsites, 87  
 httk.atomistic.unitcellstructure, 87  
 httk.atomistic.vis, 65  
 httk.atomistic.vis.asestructurevisualizer, 65  
 httk.atomistic.vis.jmolstructurevisualizer, 65  
 httk.atomistic.vis.structurephasediagramvisualizer, 66  
 httk.atomistic.vis.structurevisualizerplugin, 66  
 httk.cli, 129  
 httk.config, 90  
 httk.config.config, 90  
 httk.core, 90  
 httk.core.basic, 104  
 httk.core.citation, 105  
 httk.core.code, 105  
 httk.core.computation, 106  
 httk.core.console, 107  
 httk.core.crypto, 107  
 httk.core.ed25519, 108  
 httk.core.geometry, 108  
 httk.core.httkobject, 109  
 httk.core.ioadapters, 110  
 httk.core.project, 111  
 httk.core.reference, 111  
 httk.core.signature, 112  
 httk.core.template, 112  
 httk.core.vectors, 91  
 httk.core.vectors.fracmath, 91  
 httk.core.vectors.fracvector, 92  
 httk.core.vectors.mutablefracvector, 98  
 httk.core.vectors.vector, 99

- httk.core.vectors.vectormath, 101
- httk.db, 112
  - httk.db.backend, 112
    - httk.db.backend.sqlite, 112
  - httk.db.filteredcollection, 114
  - httk.db.httkobjdbplugin, 117
  - httk.db.storable, 117
  - httk.db.store, 113
    - httk.db.store.dictstore, 113
    - httk.db.store.sqlstore, 114
- httk.external, 118
  - httk.external.aflow\_ext, 118
  - httk.external.ase\_glue, 118
  - httk.external.cif2cell\_ext, 119
  - httk.external.command, 119
  - httk.external.gulp\_ext, 119
  - httk.external.isotropy\_ext, 119
  - httk.external.jmol, 119
  - httk.external.platon\_ext, 120
  - httk.external.pymatgen\_glue, 120
  - httk.external.pyspglib\_ext, 120
  - httk.external.subimport, 120
- httk.graphics, 121
  - httk.graphics.matplotlib, 121
- httk.httkio, 121
  - httk.httkio.cif, 121
  - httk.httkio.load, 122
  - httk.httkio.save, 122
- httk.iface, 122
  - httk.iface.ase\_if, 122
  - httk.iface.cif2cell\_if, 122
  - httk.iface.gulp\_if, 123
  - httk.iface.isotropy\_if, 123
  - httk.iface.jmol\_if, 123
  - httk.iface.platon\_if, 124
  - httk.iface.spglib\_if, 127
  - httk.iface.vasp\_if, 127
- httk.task, 128
  - httk.task.reader, 128
  - httk.task.taskmgr, 128
- httk.versioning, 129

## A

- `abstract_formula()` (in module `httk.atomistic.structureutils`), 84
- `abstract_symbol()` (in module `httk.atomistic.sitesutils`), 76
- `abstract_symbol()` (in module `httk.atomistic.structureutils`), 84
- `acos()` (`httk.core.vectors.fracvector.FracVector` method), 92
- `acos()` (`httk.FracVector` method), 41
- `acos()` (in module `httk.core.vectors.vectormath`), 101
- `acosh()` (in module `httk.core.vectors.vectormath`), 101
- `add()` (`httk.db.filteredcollection.FCMultiDict` method), 115
- `add()` (`httk.db.filteredcollection.FilteredCollection` method), 116
- `add_all()` (`httk.db.filteredcollection.FilteredCollection` method), 116
- `add_ext_citation()` (in module `httk.core.citation`), 105
- `add_name()` (`httk.atomistic.Compound` method), 58
- `add_name()` (`httk.atomistic.compound.Compound` method), 71
- `add_names()` (`httk.atomistic.Compound` method), 58
- `add_names()` (`httk.atomistic.compound.Compound` method), 71
- `add_phase()` (`httk.analysis.matsci.phasediagram.PhaseDiagram` method), 50
- `add_project()` (`httk.Computation` method), 40
- `add_project()` (`httk.core.computation.Computation` method), 106
- `add_projects()` (`httk.Computation` method), 40
- `add_projects()` (`httk.core.computation.Computation` method), 106
- `add_ref()` (`httk.atomistic.Compound` method), 58
- `add_ref()` (`httk.atomistic.compound.Compound` method), 71
- `add_ref()` (`httk.atomistic.Structure` method), 52
- `add_ref()` (`httk.atomistic.structure.Structure` method), 79
- `add_ref()` (`httk.Code` method), 39
- `add_ref()` (`httk.Computation` method), 40
- `add_ref()` (`httk.core.code.Code` method), 105
- `add_ref()` (`httk.core.computation.Computation` method), 106
- `add_refs()` (`httk.core.project.Project` method), 111
- `add_refs()` (`httk.Project` method), 41
- `add_refs()` (`httk.atomistic.Compound` method), 58
- `add_refs()` (`httk.atomistic.compound.Compound` method), 71
- `add_refs()` (`httk.atomistic.Structure` method), 52
- `add_refs()` (`httk.atomistic.structure.Structure` method), 79
- `add_refs()` (`httk.Code` method), 39
- `add_refs()` (`httk.Computation` method), 40
- `add_refs()` (`httk.core.code.Code` method), 105
- `add_refs()` (`httk.core.computation.Computation` method), 106
- `add_refs()` (`httk.core.project.Project` method), 111
- `add_refs()` (`httk.Project` method), 41
- `add_sort()` (`httk.db.filteredcollection.FilteredCollection` method), 116
- `add_src_citation()` (in module `httk.core.citation`), 105
- `add_tag()` (`httk.atomistic.Compound` method), 58
- `add_tag()` (`httk.atomistic.compound.Compound` method), 71
- `add_tag()` (`httk.atomistic.Structure` method), 52
- `add_tag()` (`httk.atomistic.structure.Structure` method), 79
- `add_tag()` (`httk.Code` method), 39
- `add_tag()` (`httk.Computation` method), 40
- `add_tag()` (`httk.core.code.Code` method), 106
- `add_tag()` (`httk.core.computation.Computation` method), 106
- `add_tag()` (`httk.core.project.Project` method), 111
- `add_tag()` (`httk.Project` method), 41
- `add_tags()` (`httk.atomistic.Compound` method), 58
- `add_tags()` (`httk.atomistic.compound.Compound` method), 71
- `add_tags()` (`httk.atomistic.Structure` method), 52
- `add_tags()` (`httk.atomistic.structure.Structure` method), 80
- `add_tags()` (`httk.Code` method), 39
- `add_tags()` (`httk.Computation` method), 40
- `add_tags()` (`httk.core.code.Code` method), 106

[add\\_tags\(\) \(httk.core.computation.Computation method\), 106](#)  
[add\\_tags\(\) \(httk.core.project.Project method\), 111](#)  
[add\\_tags\(\) \(httk.Project method\), 41](#)  
[added\\_date \(httk.Computation attribute\), 40](#)  
[added\\_date \(httk.core.computation.Computation attribute\), 106](#)  
[addsym\(\) \(in module httk.external.platon\\_ext\), 120](#)  
[addsym\\_spacegroup\(\) \(in module httk.external.platon\\_ext\), 120](#)  
[aflow\(\) \(in module httk.external.aflow\\_ext\), 118](#)  
[alter\(\) \(httk.db.backend.sqlite.Sqlite method\), 113](#)  
[analysis\(\) \(in module httk.external.pyspglib\\_ext\), 120](#)  
[angles\\_to\\_cosangles\(\) \(in module httk.atomistic.cellutils\), 69](#)  
[anonymous\\_formula \(httk.atomistic.Compound attribute\), 58](#)  
[anonymous\\_formula \(httk.atomistic.compound.Compound attribute\), 71](#)  
[anonymous\\_formula \(httk.atomistic.sites.Sites attribute\), 75](#)  
[anonymous\\_formula \(httk.atomistic.Structure attribute\), 52](#)  
[anonymous\\_formula \(httk.atomistic.structure.Structure attribute\), 80](#)  
[anonymous\\_formula\(\) \(in module httk.atomistic.sitesutils\), 76](#)  
[anonymous\\_symbol\\_to\\_int\(\) \(in module httk.core.basic\), 104](#)  
[anonymous\\_wyckoff\\_sequence \(httk.atomistic.Compound attribute\), 58](#)  
[anonymous\\_wyckoff\\_sequence \(httk.atomistic.compound.Compound attribute\), 71](#)  
[anonymous\\_wyckoff\\_sequence \(httk.atomistic.RepresentativeSites attribute\), 57](#)  
[anonymous\\_wyckoff\\_sequence \(httk.atomistic.representativesites.RepresentativeSites attribute\), 72](#)  
[anonymous\\_wyckoff\\_sequence \(httk.atomistic.Structure attribute\), 52](#)  
[anonymous\\_wyckoff\\_sequence \(httk.atomistic.structure.Structure attribute\), 80](#)  
[any\\_to\\_fraction\(\) \(in module httk.core.vectors.fracmath\), 91](#)  
[apply\\_template\(\) \(in module httk.core.template\), 112](#)  
[apply\\_templates\(\) \(in module httk.core.template\), 112](#)  
[argmax\(\) \(httk.core.vectors.fracvector.FracVector method\), 92](#)  
[argmax\(\) \(httk.FracVector method\), 41](#)  
[argmin\(\) \(httk.core.vectors.fracvector.FracVector method\), 92](#)  
[argmin\(\) \(httk.FracVector method\), 41](#)  
[ase\\_atoms\\_to\\_structure\(\) \(in module httk.external.ase\\_glue\), 118](#)  
[ase\\_read\\_structure\(\) \(in module httk.external.ase\\_glue\), 118](#)  
[ase\\_write\\_struct\(\) \(in module httk.external.ase\\_glue\), 118](#)  
[AseStructureVisualizer \(class in httk.atomistic.vis.asestructurevisualizer\), 65](#)  
[asin\(\) \(httk.core.vectors.fracvector.FracVector method\), 92](#)  
[asin\(\) \(httk.FracVector method\), 41](#)  
[asin\(\) \(in module httk.core.vectors.vectormath\), 101](#)  
[asinh\(\) \(in module httk.core.vectors.vectormath\), 101](#)  
[Assignment \(class in httk.atomistic.assignment\), 66](#)  
[Assignments \(class in httk.atomistic\), 57](#)  
[Assignments \(class in httk.atomistic.assignments\), 67](#)  
[atan\(\) \(in module httk.core.vectors.vectormath\), 101](#)  
[atan2\(\) \(in module httk.core.vectors.vectormath\), 101](#)  
[atanh\(\) \(in module httk.core.vectors.vectormath\), 101](#)  
[atomic\\_number \(httk.atomistic.siteassignment.SiteAssignment attribute\), 75](#)  
[atomic\\_number\(\) \(in module httk.atomistic.data.periodictable\), 64](#)  
[atomic\\_number\\_isotope\(\) \(in module httk.atomistic.data.periodictable\), 64](#)  
[atomic\\_numbers \(httk.atomistic.Assignments attribute\), 57](#)  
[atomic\\_numbers \(httk.atomistic.assignments.Assignments attribute\), 67](#)  
[atomic\\_numbers \(httk.atomistic.siteassignment.SiteAssignment attribute\), 75](#)  
[atomic\\_symbol\(\) \(in module httk.atomistic.data.periodictable\), 64](#)  
[Author \(class in httk\), 40](#)  
[Author \(class in httk.core.reference\), 111](#)

## B

[basics \(httk.db.store.dictstore.DictStore attribute\), 114](#)  
[basics \(httk.db.store.sqlstore.SqlStore attribute\), 114](#)  
[basis \(httk.atomistic.cellshape.CellShape attribute\), 69](#)  
[basis\\_determinant\(\) \(in module httk.atomistic.cellutils\), 69](#)  
[basis\\_determinant\(\) \(in module httk.atomistic.structureutils\), 84](#)  
[basis\\_scale\\_to\\_vol\(\) \(in module httk.atomistic.structureutils\), 84](#)  
[basis\\_to\\_niggli\(\) \(in module httk.atomistic.structureutils\), 84](#)  
[basis\\_to\\_niggli\\_and\\_orientation\(\) \(in module httk.atomistic.cellutils\), 69](#)  
[basis\\_vol\\_to\\_scale\(\) \(in module httk.atomistic.structureutils\), 84](#)  
[best\\_rational\\_in\\_interval\(\) \(in module httk.core.vectors.fracmath\), 91](#)

- BinaryBooleanOp (class in `httk.db.filteredcollection`), 114
- BinaryComparison (class in `httk.db.filteredcollection`), 114
- BinaryOp (class in `httk.db.filteredcollection`), 115
- bit() (in module `httk.core.ed25519`), 108
- bonds() (`httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer` method), 65
- breath\_first\_idx() (in module `httk.core.basic`), 104
- build\_cubic\_supercell() (in module `httk.atomistic.supercellutils`), 86
- build\_orthogonal\_supercell() (in module `httk.atomistic.supercellutils`), 86
- build\_supercell\_old() (in module `httk.atomistic.supercellutils`), 86
- ## C
- calculate\_kpoints() (in module `httk.iface.vasp_if`), 127
- cartesian\_to\_reduced() (in module `httk.atomistic.structureutils`), 84
- cc (`httk.atomistic.Structure` attribute), 52
- cc (`httk.atomistic.structure.Structure` attribute), 80
- cc\_formula\_parts (`httk.atomistic.Structure` attribute), 52
- cc\_formula\_parts (`httk.atomistic.structure.Structure` attribute), 80
- ceil() (`httk.core.vectors.fracvector.FracVector` method), 92
- ceil() (`httk.FracVector` method), 42
- ceil() (in module `httk.core.vectors.vectormath`), 101
- Cell (class in `httk.atomistic`), 55
- Cell (class in `httk.atomistic.cell`), 67
- cell\_to\_basis() (in module `httk.atomistic.cellutils`), 69
- CellShape (class in `httk.atomistic.cellshape`), 69
- cerr() (in module `httk`), 39
- cerr() (in module `httk.core.console`), 107
- chain\_vecs() (`httk.core.vectors.fracvector.FracVector` class method), 93
- chain\_vecs() (`httk.core.vectors.vector.Vector` class method), 99
- chain\_vecs() (`httk.FracVector` class method), 42
- check\_symop() (in module `httk.atomistic.spacegrouputils`), 77
- checkvalid() (in module `httk.core.ed25519`), 108
- cif2cell() (in module `httk.external.cif2cell_ext`), 119
- cif\_reader\_httk\_preprocessed() (in module `httk.atomistic.atomistico.structure_cif_io`), 63
- cif\_reader\_that\_can\_only\_read\_isotropy\_cif() (in module `httk.atomistic.atomistico.structure_cif_io`), 63
- cif\_to\_sgstructure() (in module `httk.external.platon_ext`), 120
- cif\_to\_struct() (in module `httk.atomistic.atomistico.structure_cif_io`), 63
- cif\_to\_structure\_noreduce() (in module `httk.external.cif2cell_ext`), 119
- cif\_to\_structure\_reduce() (in module `httk.external.cif2cell_ext`), 119
- cifdata\_to\_struct() (in module `httk.atomistic.atomistico.structure_cif_io`), 63
- clean() (`httk.atomistic.Cell` method), 56
- clean() (`httk.atomistic.cell.Cell` method), 67
- clean() (`httk.atomistic.cellshape.CellShape` method), 69
- clean() (`httk.atomistic.RepresentativeSites` method), 57
- clean() (`httk.atomistic.representativesites.RepresentativeSites` method), 72
- clean() (`httk.atomistic.RepresentativeStructure` method), 61
- clean() (`httk.atomistic.representativestructure.RepresentativeStructure` method), 73
- clean() (`httk.atomistic.sites.Sites` method), 75
- clean() (`httk.atomistic.Structure` method), 52
- clean() (`httk.atomistic.structure.Structure` method), 80
- clean\_coordgroups\_and\_assignments() (in module `httk.atomistic.sitesutils`), 76
- clean\_coordgroups\_and\_assignments() (in module `httk.atomistic.structureutils`), 84
- cleveropen() (in module `httk.core.ioadapters`), 110
- close() (`httk.core.ioadapters.IoAdapterFileAppender` method), 110
- close() (`httk.core.ioadapters.IoAdapterFilename` method), 110
- close() (`httk.core.ioadapters.IoAdapterFileReader` method), 110
- close() (`httk.core.ioadapters.IoAdapterFileWriter` method), 110
- close() (`httk.core.ioadapters.IoAdapterString` method), 110
- close() (`httk.db.backend.sqlite.Sqlite` method), 113
- close() (`httk.db.backend.sqlite.Sqlite.SqliteCursor` method), 112
- close() (`httk.IoAdapterFileAppender` method), 48
- close() (`httk.IoAdapterFileReader` method), 47
- close() (`httk.IoAdapterFileWriter` method), 48
- close() (`httk.IoAdapterString` method), 48
- Code (class in `httk`), 39
- Code (class in `httk.core.code`), 105
- CodeRef (class in `httk.core.code`), 106
- CodeTag (class in `httk.core.code`), 106
- Command (class in `httk.external.command`), 119
- commit() (`httk.db.backend.sqlite.Sqlite` method), 113
- commit() (`httk.db.store.sqlstore.SqlStore` method), 114
- competing\_indices (`httk.analysis.matsci.phasediagram.PhaseDiagram` attribute), 50
- Compound (class in `httk.atomistic`), 58
- Compound (class in `httk.atomistic.compound`), 71
- CompoundName (class in `httk.atomistic.compound`), 72

- CompoundRef (class in httk.atomistic), 59
- CompoundRef (class in httk.atomistic.compound), 72
- CompoundStructure (class in httk.atomistic), 58
- CompoundStructure (class in httk.atomistic.compound), 72
- CompoundTag (class in httk.atomistic), 58
- CompoundTag (class in httk.atomistic.compound), 72
- Computation (class in httk), 39
- Computation (class in httk.core.computation), 106
- ComputationProject (class in httk), 40
- ComputationProject (class in httk.core.computation), 106
- ComputationRef (class in httk.core.computation), 106
- ComputationRelated (class in httk), 40
- ComputationRelated (class in httk.core.computation), 106
- ComputationRelatedCompound (class in httk.atomistic.compound), 72
- ComputationTag (class in httk.core.computation), 107
- connections() (httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method), 65
- coord\_system (httk.analysis.matsci.phasediagram.PhaseDiagram attribute), 50
- coordgroups\_and\_assignments\_to\_coords\_and\_occupancies() (in module httk.atomistic.structureutils), 84
- coordgroups\_and\_assignments\_to\_symbols() (in module httk.atomistic.structureutils), 84
- coordgroups\_cartesian\_to\_reduced() (httk.atomistic.Cell method), 56
- coordgroups\_cartesian\_to\_reduced() (httk.atomistic.cell.Cell method), 67
- coordgroups\_cartesian\_to\_reduced() (httk.atomistic.cellshape.CellShape method), 69
- coordgroups\_cartesian\_to\_reduced() (in module httk.atomistic.sitesutils), 76
- coordgroups\_cartesian\_to\_reduced() (in module httk.atomistic.structureutils), 84
- coordgroups\_reduced\_rc\_to\_unitcellsites() (in module httk.atomistic.structureutils), 84
- coordgroups\_reduced\_rc\_to\_unitcellsites() (in module httk.external.ase\_glue), 118
- coordgroups\_reduced\_rc\_to\_unitcellsites() (in module httk.external.cif2cell\_ext), 119
- coordgroups\_reduced\_to\_cartesian() (httk.atomistic.Cell method), 56
- coordgroups\_reduced\_to\_cartesian() (httk.atomistic.cell.Cell method), 67
- coordgroups\_reduced\_to\_cartesian() (httk.atomistic.cellshape.CellShape method), 69
- coordgroups\_reduced\_to\_cartesian() (in module httk.atomistic.sitesutils), 76
- coordgroups\_reduced\_to\_unitcell() (in module httk.atomistic.sitesutils), 76
- coordgroups\_reduced\_uc\_to\_representative() (in module httk.atomistic.structureutils), 85
- coordgroups\_to\_coords() (in module httk.atomistic.sitesutils), 76
- coordgroups\_to\_coords() (in module httk.atomistic.structureutils), 85
- coords() (httk.analysis.matsci.phasediagram.PhaseDiagram method), 50
- coords\_and\_counts\_to\_coordgroups() (in module httk.atomistic.sitesutils), 76
- coords\_and\_counts\_to\_coordgroups() (in module httk.atomistic.structureutils), 85
- coords\_and\_occupancies\_to\_coordgroups\_and\_assignments() (in module httk.atomistic.sitesutils), 76
- coords\_and\_occupancies\_to\_coordgroups\_and\_assignments() (in module httk.atomistic.structureutils), 85
- coords\_cartesian\_to\_reduced() (httk.atomistic.Cell method), 56
- coords\_cartesian\_to\_reduced() (httk.atomistic.cell.Cell method), 68
- coords\_cartesian\_to\_reduced() (httk.atomistic.cellshape.CellShape method), 69
- coords\_groupnumber (httk.atomistic.sites.Sites attribute), 75
- coords\_reduced\_to\_cartesian() (httk.atomistic.Cell method), 56
- coords\_reduced\_to\_cartesian() (httk.atomistic.cell.Cell method), 68
- coords\_reduced\_to\_cartesian() (httk.atomistic.cellshape.CellShape method), 69
- coords\_reduced\_to\_cartesian() (in module httk.atomistic.sitesutils), 76
- coords\_to\_coordgroups() (in module httk.atomistic.sitesutils), 76
- coords\_to\_coordgroups() (in module httk.atomistic.structureutils), 85
- coordswap() (in module httk.atomistic.sitesutils), 76
- coordswap() (in module httk.atomistic.structureutils), 85
- copy() (httk.db.filteredcollection.FCDict method), 115
- copy() (httk.db.filteredcollection.FCMultiDict method), 115
- copy\_template() (in module httk.iface.vasp\_if), 127
- copysign() (in module httk.core.vectors.vectormath), 101
- cos() (httk.core.vectors.fracvector.FracVector method), 93
- cos() (httk.FracVector method), 42
- cos() (in module httk.core.vectors.vectormath), 101
- cosh() (in module httk.core.vectors.vectormath), 102
- counts (httk.atomistic.sites.Sites attribute), 75
- cout() (in module httk), 39
- cout() (in module httk.core.console), 107
- create() (httk.analysis.matsci.phasediagram.PhaseDiagram class method), 50

- create() (httk.atomistic.assignment.Assignment class method), 67  
 create() (httk.atomistic.Assignments class method), 57  
 create() (httk.atomistic.assignments.Assignments class method), 67  
 create() (httk.atomistic.Cell class method), 56  
 create() (httk.atomistic.cell.Cell class method), 68  
 create() (httk.atomistic.cellshape.CellShape class method), 69  
 create() (httk.atomistic.Compound class method), 58  
 create() (httk.atomistic.compound.Compound class method), 71  
 create() (httk.atomistic.compound.CompoundStructure class method), 72  
 create() (httk.atomistic.compound.ComputationRelatedCompound class method), 72  
 create() (httk.atomistic.CompoundStructure class method), 58  
 create() (httk.atomistic.RepresentativeSites class method), 57  
 create() (httk.atomistic.representativesites.RepresentativeSites class method), 72  
 create() (httk.atomistic.RepresentativeStructure class method), 62  
 create() (httk.atomistic.representativestructure.RepresentativeStructure class method), 73  
 create() (httk.atomistic.siteassignment.SiteAssignment class method), 75  
 create() (httk.atomistic.sites.Sites class method), 75  
 create() (httk.atomistic.spacegroup.Spacegroup class method), 77  
 create() (httk.atomistic.Structure class method), 52  
 create() (httk.atomistic.structure.Structure class method), 80  
 create() (httk.atomistic.StructurePhaseDiagram class method), 58  
 create() (httk.atomistic.structurephasediagram.StructurePhaseDiagram class method), 84  
 create() (httk.atomistic.structurephasediagram.StructurePhaseDiagramCompound class method), 84  
 create() (httk.atomistic.UnitcellStructure class method), 60  
 create() (httk.atomistic.unitcellstructure.UnitcellStructure class method), 88  
 create() (httk.Author class method), 40  
 create() (httk.Code class method), 39  
 create() (httk.Computation class method), 40  
 create() (httk.ComputationProject class method), 40  
 create() (httk.ComputationRelated class method), 40  
 create() (httk.core.code.Code class method), 106  
 create() (httk.core.computation.Computation class method), 106  
 create() (httk.core.computation.ComputationProject class method), 106  
 create() (httk.core.computation.ComputationRelated class method), 107  
 create() (httk.core.computation.Result class method), 107  
 create() (httk.core.project.Project class method), 111  
 create() (httk.core.project.ProjectOwner class method), 111  
 create() (httk.core.reference.Author class method), 111  
 create() (httk.core.reference.Reference class method), 111  
 create() (httk.core.signature.Signature class method), 112  
 create() (httk.core.signature.SignatureKey class method), 112  
 create() (httk.core.vectors.fracvector.FracScalar class method), 92  
 create() (httk.core.vectors.fracvector.FracVector class method), 93  
 create() (httk.core.vectors.vector.Vector class method), 99  
 create() (httk.FracScalar class method), 46  
 create() (httk.FracVector class method), 42  
 create() (httk.Project class method), 41  
 create() (httk.Reference class method), 41  
 create() (httk.Result class method), 40  
 create() (httk.Signature class method), 49  
 create() (httk.SignatureKey class method), 49  
 create\_batch\_task() (in module httk.task.taskmgr), 128  
 create\_structures() (httk.core.vectors.fracvector.FracVector class method), 93  
 create\_cos() (httk.FracVector class method), 42  
 create\_exp() (httk.core.vectors.fracvector.FracVector class method), 94  
 create\_exp() (httk.FracVector class method), 43  
 create\_sin() (httk.core.vectors.fracvector.FracVector class method), 94  
 create\_sin() (httk.FracVector class method), 43  
 create\_table() (httk.db.backend.sqlite.Sqlite method), 113  
 create\_table() (httk.db.store.dictstore.DictStore method), 114  
 create\_table() (httk.db.store.sqlstore.SqlStore method), 114  
 create\_table\_compiling\_bulk() (in module httk.core.basic), 104  
 cross() (httk.core.vectors.fracvector.FracVector method), 94  
 cross() (httk.FracVector method), 43  
 crystal\_system (httk.atomistic.RepresentativeSites attribute), 57  
 crystal\_system (httk.atomistic.representativesites.RepresentativeSites attribute), 73  
 crystal\_system\_from\_hall() (in module httk.atomistic.spacegrouputils), 77  
 crystal\_system\_from\_spacegroupnbr() (in module httk.atomistic.spacegrouputils), 77  
 cubic() (httk.atomistic.supercellutils.StructureSupercellPlugin method), 86  
 cubic\_supercell\_transformation() (in module httk.atomistic.supercellutils), 86

cursor() (httk.db.backend.sqlite.SQLite method), 113

## D

data() (httk.db.filteredcollection.FCDict method), 115

data() (httk.db.filteredcollection.FCMultiDict method), 115

db\_close() (in module httk.db.backend.sqlite), 113

db\_open() (in module httk.db.backend.sqlite), 113

db\_sqlite\_close\_all() (in module httk.db.backend.sqlite), 113

DeclaredFunction (class in httk.db.filteredcollection), 115

decodeint() (in module httk.core.ed25519), 108

decodepoint() (in module httk.core.ed25519), 108

defaults\_publish() (httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method), 66

degrees() (in module httk.core.vectors.vectormath), 102

delay\_commit() (httk.db.store.sqlstore.SqlStore method), 114

description (httk.db.backend.sqlite.SQLite.SQLiteCursor attribute), 112

destroy\_tmpdir() (in module httk.core.basic), 104

det() (httk.core.vectors.fracvector.FracVector method), 94

det() (httk.FracVector method), 43

determine\_version\_data() (in module httk.config.config), 90

DictStore (class in httk.db.store.dictstore), 113

DictStore.Keeper (class in httk.db.store.dictstore), 113

dim (httk.core.vectors.fracvector.FracVector attribute), 94

dim (httk.FracVector attribute), 43

dont\_print\_citations\_at\_exit() (in module httk.core.citation), 105

dot() (httk.core.vectors.fracvector.FracVector method), 94

dot() (httk.FracVector method), 43

duplicate() (httk.db.filteredcollection.FilteredCollection method), 116

## E

e() (in module httk.core.vectors.vectormath), 102

edwards() (in module httk.core.ed25519), 108

element\_wyckoff\_sequence (httk.atomistic.Structure attribute), 53

element\_wyckoff\_sequence (httk.atomistic.structure.Structure attribute), 81

encodeint() (in module httk.core.ed25519), 108

encodepoint() (in module httk.core.ed25519), 108

ensure\_ase\_is\_imported() (in module httk.external.ase\_glue), 118

ensure\_has\_cif2cell() (in module httk.external.cif2cell\_ext), 119

ensure\_has\_cif2cell() (in module httk.external.jmol), 119

ensure\_has\_isotropy() (in module httk.external.isotropy\_ext), 119

ensure\_has\_platon() (in module httk.external.platon\_ext), 120

ensure\_pymatgen\_is\_imported() (in module httk.external.pymatgen\_glue), 120

ensure\_pyspg\_is\_imported() (in module httk.external.pyspglib\_ext), 120

erf() (in module httk.core.vectors.vectormath), 102

erfc() (in module httk.core.vectors.vectormath), 102

ExceptionlessConfig (class in httk.config.config), 90

execute() (httk.db.backend.sqlite.SQLite.SQLiteCursor method), 112

exp() (httk.core.vectors.fracvector.FracVector method), 94

exp() (httk.FracVector method), 43

exp() (in module httk.core.vectors.vectormath), 102

expm1() (in module httk.core.vectors.vectormath), 102

expmod() (in module httk.core.ed25519), 108

Expression (class in httk.db.filteredcollection), 115

extbonds() (httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method), 66

extended (httk.atomistic.Assignments attribute), 57

extended (httk.atomistic.assignments.Assignments attribute), 67

extended (httk.atomistic.Structure attribute), 53

extended (httk.atomistic.structure.Structure attribute), 81

extensions (httk.atomistic.Structure attribute), 53

extensions (httk.atomistic.structure.Structure attribute), 81

eye() (httk.core.vectors.fracvector.FracVector class method), 94

eye() (httk.core.vectors.vector.Vector class method), 100

eye() (httk.FracVector class method), 43

## F

fabs() (in module httk.core.vectors.vectormath), 102

factorial() (in module httk.core.vectors.vectormath), 102

fc\_checkcontext() (in module httk.db.filteredcollection), 117

fc\_eval() (in module httk.db.filteredcollection), 117

fc\_get\_srctable\_context() (in module httk.db.filteredcollection), 117

fc\_sql() (in module httk.db.filteredcollection), 117

FCDict (class in httk.db.filteredcollection), 115

FCMultiDict (class in httk.db.filteredcollection), 115

FCMultiSqlite (class in httk.db.filteredcollection), 116

FCSqlite (class in httk.db.filteredcollection), 116

fetch\_codependent\_data() (httk.db.httkobjdbplugin.HttkObjDbPlugin method), 117

fetchall() (httk.db.backend.sqlite.SQLite.SQLiteCursor method), 113

fetchone() (httk.db.backend.sqlite.SQLite.SQLiteCursor method), 113

filter\_hm() (in module httk.atomistic.spacegrouputils), 77

filter\_itcnbr\_setting() (in module httk.atomistic.spacegrouputils), 77

- filter\_sf() (in module httk.atomistic.spacegrouputils), 77  
 filter\_symops() (in module httk.atomistic.spacegrouputils), 77  
 FilteredCollection (class in httk.db.filteredcollection), 116  
 find\_all() (httk.db.storable.Storable class method), 117  
 find\_executable() (in module httk.external.command), 119  
 find\_index() (in module httk.atomistic.data.spacegroups), 64  
 find\_one() (httk.db.storable.Storable class method), 117  
 find\_symmetry() (httk.atomistic.Structure method), 53  
 find\_symmetry() (httk.atomistic.structure.Structure method), 81  
 flatten() (httk.core.vectors.fracvector.FracVector method), 94  
 flatten() (httk.FracVector method), 43  
 flatten() (in module httk.core.basic), 104  
 floor() (httk.core.vectors.fracvector.FracVector method), 94  
 floor() (httk.FracVector method), 43  
 floor() (in module httk.core.vectors.vectormath), 102  
 fmod() (in module httk.core.vectors.vectormath), 102  
 formula (httk.atomistic.Structure attribute), 53  
 formula (httk.atomistic.structure.Structure attribute), 81  
 formula\_builder (httk.atomistic.RepresentativeStructure attribute), 62  
 formula\_builder (httk.atomistic.representativestructure.RepresentativeStructure attribute), 74  
 formula\_builder (httk.atomistic.UnitcellStructure attribute), 61  
 formula\_builder (httk.atomistic.unitcellstructure.UnitcellStructure attribute), 89  
 formula\_counts (httk.atomistic.Compound attribute), 58  
 formula\_counts (httk.atomistic.compound.Compound attribute), 71  
 formula\_counts (httk.atomistic.Structure attribute), 53  
 formula\_counts (httk.atomistic.structure.Structure attribute), 81  
 formula\_spaceseparated (httk.atomistic.Structure attribute), 53  
 formula\_spaceseparated (httk.atomistic.structure.Structure attribute), 81  
 formula\_symbols (httk.atomistic.Compound attribute), 58  
 formula\_symbols (httk.atomistic.compound.Compound attribute), 71  
 formula\_symbols (httk.atomistic.Structure attribute), 53  
 formula\_symbols (httk.atomistic.structure.Structure attribute), 81  
 frac\_acos() (in module httk.core.vectors.fracmath), 91  
 frac\_asin() (in module httk.core.vectors.fracmath), 91  
 frac\_atan() (in module httk.core.vectors.fracmath), 91  
 frac\_atan2() (in module httk.core.vectors.fracmath), 91  
 frac\_cos() (in module httk.core.vectors.fracmath), 91  
 frac\_exp() (in module httk.core.vectors.fracmath), 91  
 frac\_log() (in module httk.core.vectors.fracmath), 91  
 frac\_log10() (in module httk.core.vectors.fracmath), 91  
 frac\_pi() (in module httk.core.vectors.fracmath), 91  
 frac\_sin() (in module httk.core.vectors.fracmath), 91  
 frac\_sqrt() (in module httk.core.vectors.fracmath), 91  
 frac\_tan() (in module httk.core.vectors.fracmath), 91  
 FracScalar (class in httk), 46  
 FracScalar (class in httk.core.vectors.fracvector), 92  
 fraction\_from\_continued\_fraction() (in module httk.core.vectors.fracmath), 91  
 FracVector (class in httk), 41  
 FracVector (class in httk.core.vectors.fracvector), 92  
 frexp() (in module httk.core.vectors.vectormath), 102  
 from\_Atoms() (httk.external.ase\_glue.StructureAsePlugin class method), 118  
 from\_floats() (httk.core.vectors.fracvector.FracVector class method), 94  
 from\_floats() (httk.FracVector class method), 43  
 from\_FracVector() (httk.core.vectors.mutablefracvector.MutableFracVector class method), 98  
 from\_FracVector() (httk.MutableFracVector class method), 47  
 from\_tuple() (httk.core.vectors.fracvector.FracVector class method), 94  
 from\_tuple() (httk.FracVector class method), 43  
 fsum() (in module httk.core.vectors.vectormath), 102  
 FunctionStructure (class in httk.db.filteredcollection), 117  
 function() (httk.db.filteredcollection.FCDict method), 115  
 function() (httk.db.filteredcollection.FCSqlite method), 116
- ## G
- gamma() (in module httk.core.vectors.vectormath), 102  
 ged\_prestacked() (httk.core.vectors.fracvector.FracVector method), 94  
 ged\_prestacked() (httk.core.vectors.vector.Vector method), 100  
 ged\_prestacked() (httk.FracVector method), 44  
 ged\_stackedinsert() (httk.core.vectors.fracvector.FracVector method), 94  
 ged\_stackedinsert() (httk.core.vectors.vector.Vector method), 100  
 ged\_stackedinsert() (httk.FracVector method), 44  
 general() (httk.atomistic.supercellutils.StructureSupercellPlugin method), 86  
 generate\_fake\_potentials() (in module httk.iface.gulp\_if), 123  
 generate\_fake\_potentials\_try2() (in module httk.iface.gulp\_if), 123  
 generate\_keys() (in module httk.core.crypto), 107  
 get() (httk.db.store.dictstore.DictStore method), 114  
 get() (httk.db.store.sqlstore.SqlStore method), 114  
 get\_append() (httk.core.vectors.fracvector.FracVector method), 94

[get\\_append\(\) \(httk.core.vectors.vector.Vector method\), 100](#)  
[get\\_append\(\) \(httk.FracVector method\), 44](#)  
[get\\_axes\\_standard\\_order\\_transform\(\) \(httk.atomistic.Cell method\), 56](#)  
[get\\_axes\\_standard\\_order\\_transform\(\) \(httk.atomistic.cell.Cell method\), 68](#)  
[get\\_cartesian\\_coordgroups\(\) \(httk.atomistic.sites.Sites method\), 75](#)  
[get\\_cartesian\\_coords\(\) \(httk.atomistic.sites.Sites method\), 75](#)  
[get\\_codependent\\_data\(\) \(httk.core.httkobject.HttkObject method\), 109](#)  
[get\\_codependent\\_data\(\) \(httk.HttkObject method\), 48](#)  
[get\\_continued\\_fraction\(\) \(in module httk.core.vectors.fracmath\), 91](#)  
[get\\_crypto\\_signature\(\) \(in module httk.core.crypto\), 107](#)  
[get\\_extend\(\) \(httk.core.vectors.fracvector.FracVector method\), 95](#)  
[get\\_extend\(\) \(httk.core.vectors.vector.Vector method\), 100](#)  
[get\\_extend\(\) \(httk.FracVector method\), 44](#)  
[get\\_extensions\(\) \(httk.atomistic.assignment.Assignment method\), 67](#)  
[get\\_extensions\(\) \(httk.atomistic.siteassignment.SiteAssignment method\), 75](#)  
[get\\_hall\(\) \(in module httk.atomistic.spacegrouputils\), 77](#)  
[get\\_hm\\_setting\(\) \(in module httk.atomistic.spacegrouputils\), 77](#)  
[get\\_insert\(\) \(httk.core.vectors.fracvector.FracVector method\), 95](#)  
[get\\_insert\(\) \(httk.core.vectors.vector.Vector method\), 100](#)  
[get\\_insert\(\) \(httk.FracVector method\), 44](#)  
[get\\_itcnbr\\_setting\(\) \(in module httk.atomistic.spacegrouputils\), 77](#)  
[get\\_magmom\(\) \(in module httk.iface.vasp\\_if\), 127](#)  
[get\\_magnetizations\(\) \(in module httk.iface.vasp\\_if\), 127](#)  
[get\\_names\(\) \(httk.atomistic.Compound method\), 58](#)  
[get\\_names\(\) \(httk.atomistic.compound.Compound method\), 71](#)  
[get\\_nonstandard\\_hall\(\) \(in module httk.atomistic.spacegrouputils\), 77](#)  
[get\\_normalized\(\) \(httk.atomistic.Cell method\), 56](#)  
[get\\_normalized\(\) \(httk.atomistic.cell.Cell method\), 68](#)  
[get\\_normalized\\_longestvec\(\) \(httk.atomistic.Cell method\), 56](#)  
[get\\_normalized\\_longestvec\(\) \(httk.atomistic.cell.Cell method\), 68](#)  
[get\\_phasediagram\(\) \(httk.atomistic.StructurePhaseDiagram method\), 58](#)  
[get\\_phasediagram\(\) \(httk.atomistic.structurephasediagram.StructurePhaseDiagram method\), 84](#)  
[get\\_prepend\(\) \(httk.core.vectors.fracvector.FracVector method\), 95](#)  
[get\\_prepend\(\) \(httk.core.vectors.vector.Vector method\), 100](#)  
[get\\_prepend\(\) \(httk.FracVector method\), 44](#)  
[get\\_prextend\(\) \(httk.core.vectors.fracvector.FracVector method\), 95](#)  
[get\\_prextend\(\) \(httk.core.vectors.vector.Vector method\), 100](#)  
[get\\_prextend\(\) \(httk.FracVector method\), 44](#)  
[get\\_primitive\\_basis\\_transform\(\) \(in module httk.atomistic.structureutils\), 85](#)  
[get\\_primitive\\_to\\_conventional\\_basis\\_transform\(\) \(in module httk.atomistic.cellutils\), 69](#)  
[get\\_projects\(\) \(httk.Computation method\), 40](#)  
[get\\_projects\(\) \(httk.core.computation.Computation method\), 106](#)  
[get\\_proper\\_hm\\_symbol\(\) \(in module httk.atomistic.data.spacegroups\), 64](#)  
[get\\_pseudopotential\(\) \(in module httk.iface.vasp\\_if\), 127](#)  
[get\\_refs\(\) \(httk.atomistic.Compound method\), 58](#)  
[get\\_refs\(\) \(httk.atomistic.compound.Compound method\), 71](#)  
[get\\_refs\(\) \(httk.atomistic.Structure method\), 53](#)  
[get\\_refs\(\) \(httk.atomistic.structure.Structure method\), 81](#)  
[get\\_refs\(\) \(httk.Code method\), 39](#)  
[get\\_refs\(\) \(httk.Computation method\), 40](#)  
[get\\_refs\(\) \(httk.core.code.Code method\), 106](#)  
[get\\_refs\(\) \(httk.core.computation.Computation method\), 106](#)  
[get\\_refs\(\) \(httk.core.project.Project method\), 111](#)  
[get\\_refs\(\) \(httk.Project method\), 41](#)  
[get\\_row\(\) \(httk.db.backend.sqlite.Sqlite method\), 113](#)  
[get\\_rows\(\) \(httk.db.backend.sqlite.Sqlite method\), 113](#)  
[get\\_srctable\\_context\(\) \(httk.db.filteredcollection.Expression method\), 115](#)  
[get\\_stacked\(\) \(httk.core.vectors.fracvector.FracVector method\), 95](#)  
[get\\_stacked\(\) \(httk.core.vectors.vector.Vector method\), 100](#)  
[get\\_stacked\(\) \(httk.FracVector method\), 44](#)  
[get\\_stidy\\_spacegroup\(\) \(in module httk.iface.platon\\_if\), 124](#)  
[get\\_symops\(\) \(in module httk.atomistic.spacegrouputils\), 77](#)  
[get\\_symops\\_strs\(\) \(in module httk.atomistic.spacegrouputils\), 77](#)  
[get\\_symopshash\(\) \(in module httk.atomistic.spacegrouputils\), 77](#)  
[get\\_tag\(\) \(httk.atomistic.Compound method\), 58](#)  
[get\\_tag\(\) \(httk.atomistic.compound.Compound method\), 72](#)  
[get\\_tag\(\) \(httk.atomistic.Structure method\), 53](#)  
[get\\_tag\(\) \(httk.atomistic.structure.Structure method\), 81](#)  
[get\\_tag\(\) \(httk.Code method\), 39](#)  
[get\\_tag\(\) \(httk.Computation method\), 40](#)

- `get_tag()` (httk.core.code.Code method), 106
  - `get_tag()` (httk.core.computation.Computation method), 106
  - `get_tag()` (httk.core.project.Project method), 111
  - `get_tag()` (httk.Project method), 41
  - `get_tags()` (httk.atomistic.Compound method), 58
  - `get_tags()` (httk.atomistic.compound.Compound method), 72
  - `get_tags()` (httk.atomistic.Structure method), 53
  - `get_tags()` (httk.atomistic.structure.Structure method), 81
  - `get_tags()` (httk.Code method), 39
  - `get_tags()` (httk.Computation method), 40
  - `get_tags()` (httk.core.code.Code method), 106
  - `get_tags()` (httk.core.computation.Computation method), 106
  - `get_tags()` (httk.core.project.Project method), 111
  - `get_tags()` (httk.Project method), 41
  - `get_uc_sites()` (httk.atomistic.RepresentativeSites method), 57
  - `get_uc_sites()` (httk.atomistic.representativesites.RepresentativeSites method), 73
  - `get_val()` (httk.db.backend.sqlite.Sqlite method), 113
  - `get_weight()` (httk.atomistic.assignment.Assignment method), 67
- ## H
- `H()` (in module httk.core.ed25519), 108
  - `hall_symbol` (httk.atomistic.Structure attribute), 53
  - `hall_symbol` (httk.atomistic.structure.Structure attribute), 81
  - `has_rc_repr` (httk.atomistic.Structure attribute), 53
  - `has_rc_repr` (httk.atomistic.structure.Structure attribute), 81
  - `has_uc_repr` (httk.atomistic.Structure attribute), 53
  - `has_uc_repr` (httk.atomistic.structure.Structure attribute), 81
  - `hexhash` (httk.core.httkobject.HttkObject attribute), 109
  - `hexhash` (httk.HttkObject attribute), 48
  - `hexhash_ioa()` (in module httk.core.crypto), 107
  - `hexhash_str()` (in module httk.core.crypto), 107
  - `Hint()` (in module httk.core.ed25519), 108
  - `httk` (module), 39
  - `httk.analysis` (module), 49
  - `httk.analysis.matsci` (module), 49
  - `httk.analysis.matsci.phasediagram` (module), 50
  - `httk.analysis.matsci.vis` (module), 49
  - `httk.analysis.matsci.vis.phasediagramvisualizerplugin` (module), 49
  - `httk.atomistic` (module), 50
  - `httk.atomistic.assignment` (module), 66
  - `httk.atomistic.assignments` (module), 67
  - `httk.atomistic.atomistico` (module), 63
  - `httk.atomistic.atomistico.structure_cif_io` (module), 63
  - `httk.atomistic.atomistico.structure_io` (module), 64
  - `httk.atomistic.atomistico.structureioplugin` (module), 64
  - `httk.atomistic.cell` (module), 67
  - `httk.atomistic.cellshape` (module), 69
  - `httk.atomistic.cellutils` (module), 69
  - `httk.atomistic.cli` (module), 71
  - `httk.atomistic.compound` (module), 71
  - `httk.atomistic.data` (module), 64
  - `httk.atomistic.data.periodictable` (module), 64
  - `httk.atomistic.data.spacegroups` (module), 64
  - `httk.atomistic.formulautils` (module), 72
  - `httk.atomistic.representativesites` (module), 72
  - `httk.atomistic.representativestructure` (module), 73
  - `httk.atomistic.results` (module), 65
  - `httk.atomistic.results.relaxedcellresult` (module), 65
  - `httk.atomistic.results.totalenergyresult` (module), 65
  - `httk.atomistic.siteassignment` (module), 75
  - `httk.atomistic.sites` (module), 75
  - `httk.atomistic.sitesutils` (module), 76
  - `httk.atomistic.spacegroup` (module), 77
  - `httk.atomistic.spacegrouputils` (module), 77
  - `httk.atomistic.structure` (module), 78
  - `httk.atomistic.structurephasediagram` (module), 84
  - `httk.atomistic.structureutils` (module), 84
  - `httk.atomistic.supercellutils` (module), 86
  - `httk.atomistic.unitcellsites` (module), 87
  - `httk.atomistic.unitcellstructure` (module), 87
  - `httk.atomistic.vis` (module), 65
  - `httk.atomistic.vis.asestructurevisualizer` (module), 65
  - `httk.atomistic.vis.jmolstructurevisualizer` (module), 65
  - `httk.atomistic.vis.structurephasediagramvisualizerplugin` (module), 66
  - `httk.atomistic.vis.structurevisualizerplugin` (module), 66
  - `httk.cli` (module), 129
  - `httk.config` (module), 90
  - `httk.config.config` (module), 90
  - `httk.core` (module), 90
  - `httk.core.basic` (module), 104
  - `httk.core.citation` (module), 105
  - `httk.core.code` (module), 105
  - `httk.core.computation` (module), 106
  - `httk.core.console` (module), 107
  - `httk.core.crypto` (module), 107
  - `httk.core.ed25519` (module), 108
  - `httk.core.geometry` (module), 108
  - `httk.core.httkobject` (module), 109
  - `httk.core.ioadapters` (module), 110
  - `httk.core.project` (module), 111
  - `httk.core.reference` (module), 111
  - `httk.core.signature` (module), 112
  - `httk.core.template` (module), 112
  - `httk.core.vectors` (module), 91
  - `httk.core.vectors.fracmath` (module), 91
  - `httk.core.vectors.fracvector` (module), 92
  - `httk.core.vectors.mutablefracvector` (module), 98

[httk.core.vectors.vector \(module\)](#), 99  
[httk.core.vectors.vectormath \(module\)](#), 101  
[httk.db \(module\)](#), 112  
[httk.db.backend \(module\)](#), 112  
[httk.db.backend.sqlite \(module\)](#), 112  
[httk.db.filteredcollection \(module\)](#), 114  
[httk.db.httkobjdbplugin \(module\)](#), 117  
[httk.db.storable \(module\)](#), 117  
[httk.db.store \(module\)](#), 113  
[httk.db.store.dictstore \(module\)](#), 113  
[httk.db.store.sqlstore \(module\)](#), 114  
[httk.external \(module\)](#), 118  
[httk.external.afLOW\\_ext \(module\)](#), 118  
[httk.external.ase\\_glue \(module\)](#), 118  
[httk.external.cif2cell\\_ext \(module\)](#), 119  
[httk.external.command \(module\)](#), 119  
[httk.external.gulp\\_ext \(module\)](#), 119  
[httk.external.isotropy\\_ext \(module\)](#), 119  
[httk.external.jmol \(module\)](#), 119  
[httk.external.platon\\_ext \(module\)](#), 120  
[httk.external.pymatgen\\_glue \(module\)](#), 120  
[httk.external.pyspglib\\_ext \(module\)](#), 120  
[httk.external.subimport \(module\)](#), 120  
[httk.graphics \(module\)](#), 121  
[httk.graphics.matplotlib \(module\)](#), 121  
[httk.httkio \(module\)](#), 121  
[httk.httkio.cif \(module\)](#), 121  
[httk.httkio.load \(module\)](#), 122  
[httk.httkio.save \(module\)](#), 122  
[httk.iface \(module\)](#), 122  
[httk.iface.ase\\_if \(module\)](#), 122  
[httk.iface.cif2cell\\_if \(module\)](#), 122  
[httk.iface.gulp\\_if \(module\)](#), 123  
[httk.iface.isotropy\\_if \(module\)](#), 123  
[httk.iface.jmol\\_if \(module\)](#), 123  
[httk.iface.platon\\_if \(module\)](#), 124  
[httk.iface.spglib\\_if \(module\)](#), 127  
[httk.iface.vasp\\_if \(module\)](#), 127  
[httk.task \(module\)](#), 128  
[httk.task.reader \(module\)](#), 128  
[httk.task.taskmgr \(module\)](#), 128  
[httk.versioning \(module\)](#), 129  
[httk\\_typed\\_init\(\) \(in module httk\)](#), 48  
[httk\\_typed\\_init\(\) \(in module httk.core.httkobject\)](#), 109  
[httk\\_typed\\_init\\_delayed\(\) \(in module httk\)](#), 48  
[httk\\_typed\\_init\\_delayed\(\) \(in module httk.core.httkobject\)](#), 109  
[httk\\_typed\\_property\(\) \(in module httk\)](#), 48  
[httk\\_typed\\_property\(\) \(in module httk.core.httkobject\)](#), 109  
[httk\\_typed\\_property\\_delayed\(\) \(in module httk\)](#), 48  
[httk\\_typed\\_property\\_delayed\(\) \(in module httk.core.httkobject\)](#), 109  
[httk\\_typed\\_property\\_resolve\(\) \(in module httk.core.httkobject\)](#), 109  
[HttkObjDbPlugin \(class in httk.db.httkobjdbplugin\)](#), 117  
[HttkObject \(class in httk\)](#), 48  
[HttkObject \(class in httk.core.httkobject\)](#), 109  
[HttkPlugin \(class in httk\)](#), 49  
[HttkPlugin \(class in httk.core.httkobject\)](#), 109  
[HttkPluginPlaceholder \(class in httk\)](#), 49  
[HttkPluginPlaceholder \(class in httk.core.httkobject\)](#), 109  
[HttkPluginWrapper \(class in httk\)](#), 49  
[HttkPluginWrapper \(class in httk.core.httkobject\)](#), 109  
[HttkTypedProperty \(class in httk.core.httkobject\)](#), 109  
[hull\\_competing\\_indices \(httk.analysis.matsci.phasediagram.PhaseDiagram attribute\)](#), 50  
[hull\\_competing\\_phase\\_lines\(\) \(httk.analysis.matsci.phasediagram.PhaseDiagram method\)](#), 50  
[hull\\_distances \(httk.analysis.matsci.phasediagram.PhaseDiagram attribute\)](#), 50  
[hull\\_indices \(httk.analysis.matsci.phasediagram.PhaseDiagram attribute\)](#), 50  
[hull\\_point\\_coords\(\) \(httk.analysis.matsci.phasediagram.PhaseDiagram method\)](#), 50  
[hull\\_points\(\) \(httk.analysis.matsci.phasediagram.PhaseDiagram method\)](#), 50  
[hull\\_to\\_interior\\_competing\\_phase\\_lines\(\) \(httk.analysis.matsci.phasediagram.PhaseDiagram method\)](#), 50  
[hull\\_z\(\) \(in module httk.core.geometry\)](#), 108  
[hypot\(\) \(in module httk.core.vectors.vectormath\)](#), 102  
  
[initialize\(\) \(httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method\)](#), 66  
[inmap\(\) \(in module httk.core.vectors.mutablefracvector\)](#), 99  
[insert\(\) \(httk.db.backend.sqlite.SQLite method\)](#), 113  
[insert\(\) \(httk.db.store.dictstore.DictStore method\)](#), 114  
[insert\(\) \(httk.db.store.sqlstore.SqlStore method\)](#), 114  
[insert\\_row\(\) \(httk.db.backend.sqlite.SQLite method\)](#), 113  
[instantiate\\_from\\_store\(\) \(in module httk.db.filteredcollection\)](#), 117  
[int\\_to\\_anonymous\\_symbol\(\) \(in module httk.core.basic\)](#), 104  
[integer\\_sqrt\(\) \(in module httk.core.vectors.fracmath\)](#), 91  
[interior\\_competing\\_phase\\_lines\(\) \(httk.analysis.matsci.phasediagram.PhaseDiagram method\)](#), 50  
[interior\\_point\\_coords\(\) \(httk.analysis.matsci.phasediagram.PhaseDiagram method\)](#), 50  
[internal\\_coordgroups\\_reduced\\_rc\\_to\\_unitcellsites\(\) \(in module httk.atomistic.structureutils\)](#), 85  
[inv\(\) \(httk.core.vectors.fracvector.FracVector method\)](#), 95  
[inv\(\) \(httk.FracVector method\)](#), 44

- inv() (in module `httk.core.ed25519`), 108  
 invalidate() (`httk.core.vectors.mutablefracvector.MutableFracVector` attribute), 57  
     method), 98  
 invalidate() (`httk.MutableFracVector` method), 47  
 io (`httk.atomistic.Structure` attribute), 54  
 io (`httk.atomistic.structure.Structure` attribute), 81  
 IoAdapterFileAppender (class in `httk`), 48  
 IoAdapterFileAppender (class in `httk.core.ioadapters`), 110  
 IoAdapterFilename (class in `httk.core.ioadapters`), 110  
 IoAdapterFileReader (class in `httk`), 47  
 IoAdapterFileReader (class in `httk.core.ioadapters`), 110  
 IoAdapterFileWriter (class in `httk`), 48  
 IoAdapterFileWriter (class in `httk.core.ioadapters`), 110  
 IoAdapterString (class in `httk`), 48  
 IoAdapterString (class in `httk.core.ioadapters`), 110  
 IoAdapterStringList (class in `httk`), 48  
 IoAdapterStringList (class in `httk.core.ioadapters`), 110  
 is\_any\_part\_of\_cube\_inside\_cell() (in module `httk.core.geometry`), 108  
 is\_dualmagnetic() (in module `httk.iface.vasp_if`), 127  
 is\_in() (`httk.db.filteredcollection.Expression` method), 115  
 is\_point\_inside() (`httk.atomistic.Cell` method), 56  
 is\_point\_inside() (`httk.atomistic.cell.Cell` method), 68  
 is\_point\_inside() (`httk.atomistic.cellshape.CellShape` method), 69  
 is\_point\_inside\_cell() (in module `httk.core.geometry`), 109  
 is\_point\_inside\_tetra() (in module `httk.core.geometry`), 109  
 is\_sequence() (in module `httk.core.basic`), 104  
 is\_string() (in module `httk.core.vectors.fracmath`), 92  
 is\_unary() (in module `httk.core.basic`), 104  
 isanyinf() (in module `httk.core.vectors.vectormath`), 103  
 isanynan() (in module `httk.core.vectors.vectormath`), 103  
 isinf() (in module `httk.core.vectors.vectormath`), 103  
 isnan() (in module `httk.core.vectors.vectormath`), 103  
 isoncurve() (in module `httk.core.ed25519`), 108  
 isotropy() (in module `httk.external.isotropy_ext`), 119
- ## J
- jmol() (in module `httk.external.gulp_ext`), 119  
 JmolStructureVisualizer (class in `httk.atomistic.vis.jmolstructurevisualizer`), 65
- ## L
- lattice\_symbol (`httk.atomistic.RepresentativeSites` attribute), 57  
 lattice\_symbol (`httk.atomistic.representativesites.RepresentativeSites` attribute), 73  
 lattice\_symbol\_from\_hall() (in module `httk.atomistic.spacegrouputils`), 77  
 lattice\_system (`httk.atomistic.RepresentativeSites` attribute), 57  
 lattice\_system (`httk.atomistic.representativesites.RepresentativeSites` attribute), 73  
 lattice\_system\_from\_hall() (in module `httk.atomistic.spacegrouputils`), 77  
 lattice\_system\_from\_lengths\_and\_cosangles() (in module `httk.atomistic.cellutils`), 70  
 lattice\_system\_from\_niggli() (in module `httk.atomistic.cellutils`), 70  
 lattice\_type\_from\_hall() (in module `httk.atomistic.spacegrouputils`), 77  
 ldexp() (in module `httk.core.vectors.vectormath`), 103  
 lengths\_and\_angles\_to\_niggli() (in module `httk.atomistic.cellutils`), 70  
 lengths\_and\_cosangles\_to\_conventional\_basis() (in module `httk.atomistic.cellutils`), 70  
 lengths\_and\_cosangles\_to\_niggli() (in module `httk.atomistic.cellutils`), 70  
 lengths\_angles\_to\_niggli() (in module `httk.atomistic.structureutils`), 85  
 lengthsqr() (`httk.core.vectors.fracvector.FracVector` method), 95  
 lengthsqr() (`httk.FracVector` method), 44  
 lgamma() (in module `httk.core.vectors.vectormath`), 103  
 limit\_denominator() (`httk.core.vectors.fracvector.FracVector` method), 95  
 limit\_denominator() (`httk.FracVector` method), 44  
 line\_coords() (`httk.analysis.matsci.phasediagram.PhaseDiagram` method), 50  
 list\_set\_slice() (in module `httk.core.vectors.mutablefracvector`), 99  
 list\_slice() (in module `httk.core.vectors.mutablefracvector`), 99  
 load() (`httk.atomistic.atomistico.structureioplugin.StructureIoPlugin` class method), 64  
 load() (in module `httk`), 39  
 load() (in module `httk.httkio.load`), 122  
 load\_struct() (in module `httk.atomistic.atomistico.structure_io`), 64  
 log() (in module `httk.core.vectors.vectormath`), 103  
 log10() (in module `httk.core.vectors.vectormath`), 103  
 log1p() (in module `httk.core.vectors.vectormath`), 103
- ## M
- magnetization\_recurse() (in module `httk.iface.vasp_if`), 127  
 main() (in module `httk.atomistic.assignment`), 67  
 main() (in module `httk.atomistic.assignments`), 67  
 main() (in module `httk.atomistic.cell`), 68  
 main() (in module `httk.atomistic.cellshape`), 69  
 main() (in module `httk.atomistic.cellutils`), 70  
 main() (in module `httk.atomistic.cli`), 71  
 main() (in module `httk.atomistic.compound`), 72

main() (in module htk.atomistic.representativesites), 73  
 main() (in module htk.atomistic.representativestructure), 75  
 main() (in module htk.atomistic.siteassignment), 75  
 main() (in module htk.atomistic.sites), 76  
 main() (in module htk.atomistic.sitesutils), 76  
 main() (in module htk.atomistic.spacegroup), 77  
 main() (in module htk.atomistic.spacegrouputils), 77  
 main() (in module htk.atomistic.structure), 83  
 main() (in module htk.atomistic.structurephasediagram), 84  
 main() (in module htk.atomistic.structureutils), 85  
 main() (in module htk.atomistic.unitcellsites), 87  
 main() (in module htk.cli), 129  
 main() (in module htk.core.basic), 104  
 main() (in module htk.core.code), 106  
 main() (in module htk.core.computation), 107  
 main() (in module htk.core.crypto), 107  
 main() (in module htk.core.ed25519), 108  
 main() (in module htk.core.project), 111  
 main() (in module htk.core.reference), 111  
 main() (in module htk.core.signature), 112  
 main() (in module htk.core.vectors.fracmath), 92  
 main() (in module htk.core.vectors.fracvector), 97  
 main() (in module htk.core.vectors.mutablefracvector), 99  
 main() (in module htk.core.vectors.vector), 100  
 main() (in module htk.core.vectors.vectormath), 103  
 main() (in module htk.external.jmol), 119  
 main() (in module htk.htkio.cif), 121  
 manifest\_dir() (in module htk.core.crypto), 107  
 max() (htk.core.vectors.fracvector.FracVector method), 95  
 max() (htk.FracVector method), 44  
 metric\_product() (htk.core.vectors.fracvector.FracVector method), 95  
 metric\_product() (htk.FracVector method), 44  
 metric\_to\_niggli() (in module htk.atomistic.cellutils), 70  
 metric\_to\_niggli() (in module htk.atomistic.structureutils), 85  
 micro\_pyawk() (in module htk.core.basic), 104  
 min() (htk.core.vectors.fracvector.FracVector method), 95  
 min() (htk.FracVector method), 44  
 mkdir\_p() (in module htk.core.basic), 105  
 modf() (in module htk.core.vectors.vectormath), 103  
 modify\_structure() (htk.db.backend.sqlite.Sqlite method), 113  
 most\_common\_mass() (in module htk.atomistic.data.periodictable), 64  
 mul() (htk.core.vectors.fracvector.FracVector method), 95  
 mul() (htk.FracVector method), 44  
 MutableFracVector (class in htk), 46

MutableFracVector (class in htk.core.vectors.mutablefracvector), 98  
 MutableVector (class in htk.core.vectors.vector), 99

## N

name (htk.external.ase\_glue.StructureAsePlugin attribute), 118  
 nargmax() (htk.core.vectors.fracvector.FracVector method), 95  
 nargmax() (htk.FracVector method), 44  
 nargmin() (htk.core.vectors.fracvector.FracVector method), 95  
 nargmin() (htk.FracVector method), 44  
 nested\_inmap() (htk.core.vectors.mutablefracvector.MutableFracVector static method), 98  
 nested\_inmap() (htk.MutableFracVector static method), 47  
 nested\_inmap\_list() (in module htk.core.vectors.mutablefracvector), 99  
 nested\_map() (htk.core.vectors.fracvector.FracVector static method), 95  
 nested\_map() (htk.core.vectors.mutablefracvector.MutableFracVector static method), 98  
 nested\_map() (htk.FracVector static method), 44  
 nested\_map() (htk.MutableFracVector static method), 47  
 nested\_map\_fractions() (htk.core.vectors.fracvector.FracVector static method), 95  
 nested\_map\_fractions() (htk.core.vectors.mutablefracvector.MutableFracVector static method), 98  
 nested\_map\_fractions() (htk.FracVector static method), 44  
 nested\_map\_fractions() (htk.MutableFracVector static method), 47  
 nested\_map\_fractions\_list() (in module htk.core.vectors.fracvector), 97  
 nested\_map\_fractions\_list() (in module htk.core.vectors.vector), 100  
 nested\_map\_fractions\_tuple() (in module htk.core.vectors.fracvector), 97  
 nested\_map\_list() (in module htk.core.vectors.fracvector), 97  
 nested\_map\_list() (in module htk.core.vectors.vector), 100  
 nested\_map\_tuple() (in module htk.core.vectors.fracvector), 97  
 nested\_reduce() (in module htk.core.vectors.fracvector), 97  
 nested\_reduce() (in module htk.core.vectors.vector), 100  
 nested\_reduce\_fractions() (in module htk.core.vectors.fracvector), 97  
 nested\_reduce\_fractions() (in module htk.core.vectors.vector), 100  
 nested\_reduce\_levels() (in module htk.core.vectors.fracvector), 97

- [nested\\_reduce\\_levels\(\)](#) (in module [htt看.core.vectors.vector](#)), 100  
[nested\\_split\(\)](#) (in module [htt看.core.basic](#)), 105  
[new\(\)](#) ([htt看.db.storable.TrivialStore](#) method), 118  
[new\(\)](#) ([htt看.db.store.dictstore.DictStore](#) method), 114  
[new\(\)](#) ([htt看.db.store.sqlstore.SqlStore](#) method), 114  
[new\\_from\(\)](#) ([htt看.core.httkobject.HttkObject](#) class method), 109  
[new\\_from\(\)](#) ([htt看.HttkObject](#) class method), 48  
[next\(\)](#) ([htt看.core.basic.rewindable\\_iterator](#) method), 105  
[niggli\\_scale\\_to\\_vol\(\)](#) (in module [htt看.atomistic.cellutils](#)), 70  
[niggli\\_scale\\_to\\_vol\(\)](#) (in module [htt看.atomistic.structureutils](#)), 85  
[niggli\\_to\\_basis\(\)](#) (in module [htt看.atomistic.cellutils](#)), 70  
[niggli\\_to\\_basis\(\)](#) (in module [htt看.atomistic.structureutils](#)), 85  
[niggli\\_to\\_cell\\_old\(\)](#) (in module [htt看.atomistic.structureutils](#)), 85  
[niggli\\_to\\_conventional\\_basis\(\)](#) (in module [htt看.atomistic.cellutils](#)), 70  
[niggli\\_to\\_lengths\\_and\\_angles\(\)](#) (in module [htt看.atomistic.cellutils](#)), 71  
[niggli\\_to\\_lengths\\_and\\_trigangles\(\)](#) (in module [htt看.atomistic.cellutils](#)), 71  
[niggli\\_to\\_lengths\\_angles\(\)](#) (in module [htt看.atomistic.structureutils](#)), 85  
[niggli\\_to\\_metric\(\)](#) (in module [htt看.atomistic.cellutils](#)), 71  
[niggli\\_to\\_metric\(\)](#) (in module [htt看.atomistic.structureutils](#)), 85  
[niggli\\_vol\\_to\\_scale\(\)](#) (in module [htt看.atomistic.structureutils](#)), 85  
[nom](#) ([htt看.core.vectors.fracvector.FracVector](#) attribute), 95  
[nom](#) ([htt看.FracVector](#) attribute), 44  
[normalization\\_longestvec\\_scale](#) ([htt看.atomistic.Cell](#) attribute), 56  
[normalization\\_longestvec\\_scale](#) ([htt看.atomistic.cell.Cell](#) attribute), 68  
[normalization\\_scale](#) ([htt看.atomistic.Cell](#) attribute), 56  
[normalization\\_scale](#) ([htt看.atomistic.cell.Cell](#) attribute), 68  
[normalize\(\)](#) ([htt看.core.vectors.fracvector.FracVector](#) method), 95  
[normalize\(\)](#) ([htt看.FracVector](#) method), 45  
[normalize\\_half\(\)](#) ([htt看.core.vectors.fracvector.FracVector](#) method), 95  
[normalize\\_half\(\)](#) ([htt看.FracVector](#) method), 45  
[normalized\\_formula\(\)](#) (in module [htt看.atomistic.structureutils](#)), 85  
[normalized\\_formula\\_parts\(\)](#) (in module [htt看.atomistic.sitesutils](#)), 76  
[normalized\\_formula\\_parts\(\)](#) (in module [htt看.atomistic.structureutils](#)), 85  
[number](#) ([htt看.atomistic.spacegroup.Spacegroup](#) attribute), 77  
[number\\_and\\_setting](#) ([htt看.atomistic.spacegroup.Spacegroup](#) attribute), 77  
[number\\_of\\_elements](#) ([htt看.atomistic.Compound](#) attribute), 58  
[number\\_of\\_elements](#) ([htt看.atomistic.compound.Compound](#) attribute), 72  
[number\\_of\\_elements](#) ([htt看.atomistic.Structure](#) attribute), 54  
[number\\_of\\_elements](#) ([htt看.atomistic.structure.Structure](#) attribute), 81  
[numpy\\_quickhull\\_2d\(\)](#) (in module [htt看.core.geometry](#)), 109
- ## O
- [occupations\\_and\\_coords\\_to\\_assignments\\_and\\_coordgroups\(\)](#) (in module [htt看.atomistic.structureutils](#)), 85  
[orthogonal\(\)](#) ([htt看.atomistic.supercellutils.StructureSupercellPlugin](#) method), 86  
[orthogonal\\_supercell\\_transformation\(\)](#) (in module [htt看.atomistic.supercellutils](#)), 86  
[other\\_point\\_coords\(\)](#) ([htt看.analysis.matsci.phasediagram.PhaseDiagram](#) method), 50  
[out\\_to\\_cif\(\)](#) (in module [htt看.iface.isotropy\\_if](#)), 123  
[out\\_to\\_struct\(\)](#) (in module [htt看.iface.cif2cell\\_if](#)), 122  
[OutcarReader](#) (class in [htt看.iface.vasp\\_if](#)), 127  
[output\(\)](#) ([htt看.db.filteredcollection.FilteredCollection](#) method), 116
- ## P
- [params\(\)](#) ([htt看.analysis.matsci.vis.phasediagramvisualizerplugin.PhaseDiagramVisualizerPlugin](#) method), 49  
[params\(\)](#) ([htt看.atomistic.vis.structurevisualizerplugin.StructureVisualizerPlugin](#) method), 66  
[parse\(\)](#) ([htt看.iface.vasp\\_if.OutcarReader](#) method), 127  
[parse\\_parexpr\(\)](#) (in module [htt看.core.basic](#)), 105  
[pbc](#) ([htt看.atomistic.RepresentativeStructure](#) attribute), 62  
[pbc](#) ([htt看.atomistic.representativestructure.RepresentativeStructure](#) attribute), 74  
[pbc](#) ([htt看.atomistic.Structure](#) attribute), 54  
[pbc](#) ([htt看.atomistic.structure.Structure](#) attribute), 81  
[pbc](#) ([htt看.atomistic.UnitcellStructure](#) attribute), 61  
[pbc](#) ([htt看.atomistic.unitcellstructure.UnitcellStructure](#) attribute), 89  
[pbc\\_to\\_nonperiodic\\_vecs\(\)](#) (in module [htt看.atomistic.sitesutils](#)), 76  
[pc](#) ([htt看.atomistic.Structure](#) attribute), 54  
[pc](#) ([htt看.atomistic.structure.Structure](#) attribute), 81  
[pc\\_a](#) ([htt看.atomistic.Structure](#) attribute), 54  
[pc\\_a](#) ([htt看.atomistic.structure.Structure](#) attribute), 81  
[pc\\_alpha](#) ([htt看.atomistic.Structure](#) attribute), 54  
[pc\\_alpha](#) ([htt看.atomistic.structure.Structure](#) attribute), 81  
[pc\\_b](#) ([htt看.atomistic.Structure](#) attribute), 54  
[pc\\_b](#) ([htt看.atomistic.structure.Structure](#) attribute), 81

pc\_beta (httk.atomistic.Structure attribute), 54  
 pc\_beta (httk.atomistic.structure.Structure attribute), 82  
 pc\_c (httk.atomistic.Structure attribute), 54  
 pc\_c (httk.atomistic.structure.Structure attribute), 82  
 pc\_counts (httk.atomistic.Structure attribute), 54  
 pc\_counts (httk.atomistic.structure.Structure attribute), 82  
 pc\_formula\_parts (httk.atomistic.Structure attribute), 54  
 pc\_formula\_parts (httk.atomistic.structure.Structure attribute), 82  
 pc\_gamma (httk.atomistic.Structure attribute), 54  
 pc\_gamma (httk.atomistic.structure.Structure attribute), 82  
 pc\_nbr\_atoms (httk.atomistic.Structure attribute), 54  
 pc\_nbr\_atoms (httk.atomistic.structure.Structure attribute), 82  
 pc\_volume (httk.atomistic.Structure attribute), 54  
 pc\_volume (httk.atomistic.structure.Structure attribute), 82  
 periodicity\_to\_pbc() (in module httk.atomistic.sitesutils), 76  
 phase\_lines (httk.analysis.matsci.phasediagram.PhaseDiagram attribute), 50  
 PhaseDiagram (class in httk.analysis.matsci.phasediagram), 50  
 PhaseDiagramVisualizerPlugin (class in httk.analysis.matsci.vis.phasediagramvisualizerplugin), 49  
 pi() (httk.core.vectors.fracvector.FracVector class method), 96  
 pi() (httk.FracVector class method), 45  
 pi() (in module httk.core.vectors.vectormath), 103  
 platon() (in module httk.external.platon\_ext), 120  
 platon\_lis\_to\_struct\_broken() (in module httk.iface.platon\_if), 124  
 platon\_lis\_to\_struct\_broken2() (in module httk.iface.platon\_if), 125  
 platon\_sites\_to\_styin() (in module httk.iface.platon\_if), 125  
 platon\_styin\_to\_sgstruct() (in module httk.iface.platon\_if), 126  
 platon\_styout\_to\_sgstruct() (in module httk.iface.platon\_if), 126  
 platon\_styout\_to\_structure() (in module httk.iface.platon\_if), 126  
 plugin\_init() (httk.analysis.matsci.vis.phasediagramvisualizerplugin.PhaseDiagramVisualizerPlugin method), 49  
 plugin\_init() (httk.atomistic.atomistico.structureioplugin.StructureIoPlugin method), 64  
 plugin\_init() (httk.atomistic.formulautils.StructureFormulaPlugin method), 72  
 plugin\_init() (httk.atomistic.supercellutils.StructureSupercellPlugin method), 86  
 plugin\_init() (httk.atomistic.vis.structurephasediagramvisualizerplugin.StructurePhaseDiagramVisualizerPlugin method), 66  
 plugin\_init() (httk.db.httkobjdbplugin.HttkObjDbPlugin method), 117  
 plugin\_init() (httk.external.ase\_glue.StructureAsePlugin method), 118  
 polyhedra() (httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method), 66  
 poscar\_to\_strs() (in module httk.iface.vasp\_if), 127  
 poscar\_to\_structure() (in module httk.iface.vasp\_if), 128  
 postconnect() (httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method), 66  
 pow() (in module httk.core.vectors.vectormath), 103  
 preconnect() (httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method), 66  
 prepare\_single\_run() (in module httk.iface.vasp\_if), 128  
 primitive() (in module httk.external.pyspglib\_ext), 120  
 primitive\_from\_conventional\_cell() (in module httk.external.ase\_glue), 118  
 print\_citations() (in module httk.core.citation), 105  
 print\_citations\_at\_exit() (in module httk.core.citation), 105  
 Project (class in httk), 41  
 Project (class in httk.core.project), 111  
 ProjectOwner (class in httk.core.project), 111  
 ProjectRef (class in httk), 41  
 ProjectRef (class in httk.core.project), 111  
 ProjectTag (class in httk), 41  
 ProjectTag (class in httk.core.project), 111  
 prototype\_formula() (in module httk.atomistic.structureutils), 86  
 publickey() (in module httk.core.ed25519), 108  
 put() (httk.db.store.dictstore.DictStore method), 114  
 put() (httk.db.store.sqlstore.SqlStore method), 114  
 puts() (httk.db.store.dictstore.DictStore method), 114  
 puts() (httk.db.store.dictstore.DictStore.Keeper method), 113  
 puts() (httk.db.store.sqlstore.SqlStore method), 114  
 puts() (httk.db.store.sqlstore.SqlStore.Keeper method), 114

## Q

query() (httk.db.backend.sqlite.Sqlite method), 113

## R

radians() (in module httk.core.vectors.vectormath), 103  
 random() (httk.core.vectors.fracvector.FracVector class method), 96  
 random() (httk.core.vectors.vector.Vector class method), 100  
 random() (httk.FracVector class method), 45  
 ratio (httk.atomistic.siteassignment.SiteAssignment attribute), 73

ratios (httk.atomistic.Assignments attribute), 57  
 ratios (httk.atomistic.assignments.Assignments attribute), 67  
 ratios (httk.atomistic.siteassignment.SiteAssignment attribute), 75  
 ratioslist (httk.atomistic.Assignments attribute), 57  
 ratioslist (httk.atomistic.assignments.Assignments attribute), 67  
 rc (httk.atomistic.Structure attribute), 54  
 rc (httk.atomistic.structure.Structure attribute), 82  
 rc\_a (httk.atomistic.RepresentativeStructure attribute), 63  
 rc\_a (httk.atomistic.representativestructure.RepresentativeStructure attribute), 74  
 rc\_a (httk.atomistic.Structure attribute), 54  
 rc\_a (httk.atomistic.structure.Structure attribute), 82  
 rc\_alpha (httk.atomistic.RepresentativeStructure attribute), 63  
 rc\_alpha (httk.atomistic.representativestructure.RepresentativeStructure attribute), 74  
 rc\_alpha (httk.atomistic.Structure attribute), 54  
 rc\_alpha (httk.atomistic.structure.Structure attribute), 82  
 rc\_b (httk.atomistic.RepresentativeStructure attribute), 63  
 rc\_b (httk.atomistic.representativestructure.RepresentativeStructure attribute), 74  
 rc\_b (httk.atomistic.Structure attribute), 54  
 rc\_b (httk.atomistic.structure.Structure attribute), 82  
 rc\_basis (httk.atomistic.RepresentativeStructure attribute), 63  
 rc\_basis (httk.atomistic.representativestructure.RepresentativeStructure attribute), 74  
 rc\_basis (httk.atomistic.Structure attribute), 54  
 rc\_basis (httk.atomistic.structure.Structure attribute), 82  
 rc\_beta (httk.atomistic.RepresentativeStructure attribute), 63  
 rc\_beta (httk.atomistic.representativestructure.RepresentativeStructure attribute), 74  
 rc\_beta (httk.atomistic.Structure attribute), 54  
 rc\_beta (httk.atomistic.structure.Structure attribute), 82  
 rc\_c (httk.atomistic.RepresentativeStructure attribute), 63  
 rc\_c (httk.atomistic.representativestructure.RepresentativeStructure attribute), 74  
 rc\_c (httk.atomistic.Structure attribute), 54  
 rc\_c (httk.atomistic.structure.Structure attribute), 82  
 rc\_cartesian\_coordgroups (httk.atomistic.RepresentativeStructure attribute), 63  
 rc\_cartesian\_coordgroups (httk.atomistic.representativestructure.RepresentativeStructure attribute), 74  
 rc\_cartesian\_coordgroups (httk.atomistic.Structure attribute), 54  
 rc\_cartesian\_coordgroups (httk.atomistic.structure.Structure attribute), 82  
 rc\_cartesian\_coords (httk.atomistic.RepresentativeStructure attribute), 63  
 rc\_cartesian\_coords (httk.atomistic.representativestructure.RepresentativeStructure attribute), 74  
 rc\_cartesian\_coords (httk.atomistic.Structure attribute), 54  
 rc\_cartesian\_coords (httk.atomistic.structure.Structure attribute), 82  
 rc\_cartesian\_occupationscoords (httk.atomistic.RepresentativeStructure attribute), 63  
 rc\_cartesian\_occupationscoords (httk.atomistic.representativestructure.RepresentativeStructure attribute), 74  
 rc\_cartesian\_occupationscoords (httk.atomistic.Structure attribute), 54  
 rc\_cartesian\_occupationscoords (httk.atomistic.structure.Structure attribute), 82  
 rc\_cell\_orientation (httk.atomistic.RepresentativeStructure attribute), 63  
 rc\_cell\_orientation (httk.atomistic.representativestructure.RepresentativeStructure attribute), 74  
 rc\_cell\_orientation (httk.atomistic.Structure attribute), 54  
 rc\_cell\_orientation (httk.atomistic.structure.Structure attribute), 82  
 rc\_counts (httk.atomistic.Structure attribute), 54  
 rc\_counts (httk.atomistic.structure.Structure attribute), 82  
 rc\_gamma (httk.atomistic.RepresentativeStructure attribute), 63  
 rc\_gamma (httk.atomistic.representativestructure.RepresentativeStructure attribute), 74  
 rc\_gamma (httk.atomistic.Structure attribute), 54  
 rc\_gamma (httk.atomistic.structure.Structure attribute), 82  
 rc\_lengths\_and\_angles (httk.atomistic.RepresentativeStructure attribute), 63  
 rc\_lengths\_and\_angles (httk.atomistic.representativestructure.RepresentativeStructure attribute), 74  
 rc\_lengths\_and\_angles (httk.atomistic.Structure attribute), 54  
 rc\_lengths\_and\_angles (httk.atomistic.structure.Structure attribute), 82  
 rc\_nbr\_atoms (httk.atomistic.Structure attribute), 54  
 rc\_nbr\_atoms (httk.atomistic.structure.Structure attribute), 82  
 rc\_occupancies (httk.atomistic.Structure attribute), 54  
 rc\_occupancies (httk.atomistic.structure.Structure attribute), 82  
 rc\_occupationssymbols (httk.atomistic.Structure attribute), 54  
 rc\_occupationssymbols (httk.atomistic.structure.Structure attribute), 82  
 rc\_reduced\_coordgroups (httk.atomistic.Structure attribute), 54  
 rc\_reduced\_coordgroups (httk.atomistic.structure.Structure attribute), 82

- attribute), 82
  - rc\_reduced\_coords (httk.atomistic.Structure attribute), 54
  - rc\_reduced\_coords (httk.atomistic.structure.Structure attribute), 82
  - rc\_structure\_to\_symbols\_and\_scaled\_positions() (in module httk.iface.ase\_if), 122
  - rc\_volume (httk.atomistic.RepresentativeStructure attribute), 63
  - rc\_volume (httk.atomistic.representativestructure.RepresentativeStructure attribute), 74
  - rc\_volume (httk.atomistic.Structure attribute), 54
  - rc\_volume (httk.atomistic.structure.Structure attribute), 82
  - read\_cif() (in module httk.httkio.cif), 121
  - read\_config() (in module httk.config.config), 90
  - read\_keys() (in module httk.core.crypto), 107
  - read\_manifest() (in module httk.task.reader), 128
  - read\_outcar() (in module httk.iface.vasp\_if), 128
  - reader() (in module httk.task.reader), 128
  - receive() (httk.external.command.Command method), 119
  - reciprocal() (httk.core.vectors.fracvector.FracVector method), 96
  - reciprocal() (httk.FracVector method), 45
  - reduce\_by\_symops() (in module httk.atomistic.spacegrouputils), 77
  - reduced\_coordgroups (httk.atomistic.sites.Sites attribute), 75
  - reduced\_coordgroups\_to\_input() (in module httk.iface.isotropy\_if), 123
  - reduced\_coords (httk.atomistic.sites.Sites attribute), 76
  - reduced\_to\_cartesian() (in module httk.atomistic.structureutils), 86
  - Reference (class in httk), 40
  - Reference (class in httk.core.reference), 111
  - refresh() (httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method), 66
  - repeat() (httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method), 66
  - RepresentativeSites (class in httk.atomistic), 57
  - RepresentativeSites (class in httk.atomistic.representativesites), 72
  - RepresentativeStructure (class in httk.atomistic), 61
  - RepresentativeStructure (class in httk.atomistic.representativestructure), 73
  - reset() (httk.db.filteredcollection.FilteredCollection method), 116
  - Result (class in httk), 40
  - Result (class in httk.core.computation), 107
  - Result\_RelaxedCellResult (class in httk.atomistic.results.relaxedcellresult), 65
  - Result\_TotalEnergyResult (class in httk.atomistic.results.totalenergyresult), 65
  - retrieve() (httk.db.storable.TrivialStore method), 118
  - retrieve() (httk.db.store.dictstore.DictStore method), 114
  - retrieve() (httk.db.store.sqlstore.SqlStore method), 114
  - rewind() (httk.core.basic.rewindable\_iterator method), 105
  - rewindable\_iterator (class in httk.core.basic), 105
  - rollback() (httk.db.backend.sqlite.Sqlite method), 113
  - rotate() (httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method), 66
  - run() (httk.external.command.Command method), 119
  - run() (in module httk.external.jmol), 119
  - run\_alot() (in module httk.core.vectors.fracmath), 92
- ## S
- save() (httk.atomistic.atomistico.structureioplugin.StructureIoPlugin method), 64
  - save() (httk.db.store.sqlstore.SqlStore method), 114
  - save() (in module httk), 39
  - save() (in module httk.httkio.save), 122
  - save\_and\_quit() (httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method), 66
  - save\_struct() (in module httk.atomistic.atomistico.structure\_io), 64
  - Scalar (class in httk.core.vectors.vector), 99
  - scalarmult() (in module httk.core.ed25519), 108
  - scale\_to\_vol() (in module httk.atomistic.cellutils), 71
  - scaling() (httk.atomistic.Cell method), 56
  - scaling() (httk.atomistic.cell.Cell method), 68
  - scaling() (httk.atomistic.cellshape.CellShape method), 69
  - scaling\_to\_volume() (in module httk.atomistic.cellutils), 71
  - searcher() (httk.db.store.sqlstore.SqlStore method), 114
  - send() (httk.external.command.Command method), 119
  - set\_common\_denom() (httk.core.vectors.fracvector.FracVector class method), 96
  - set\_common\_denom() (httk.FracVector class method), 45
  - set\_denominator() (httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method), 66
  - set\_denominator() (httk.core.vectors.fracvector.FracVector method), 96
  - set\_denominator() (httk.FracVector method), 45
  - set\_hull\_data() (httk.analysis.matsci.phasediagram.PhaseDiagram method), 50
  - set\_inv() (httk.core.vectors.mutablefracvector.MutableFracVector method), 98
  - set\_inv() (httk.MutableFracVector method), 47
  - set\_mp\_key() (in module httk.external.pympatgen\_glue), 120
  - set\_negative() (httk.core.vectors.mutablefracvector.MutableFracVector method), 98
  - set\_negative() (httk.MutableFracVector method), 47
  - set\_normalize() (httk.core.vectors.mutablefracvector.MutableFracVector method), 98
  - set\_normalize() (httk.MutableFracVector method), 47

set\_normalize\_half() (httk.core.vectors.mutablefracvector.MutableFracVector method), 98  
 set\_normalize\_half() (httk.MutableFracVector method), 47  
 set\_set\_denominator() (httk.core.vectors.mutablefracvector.MutableFracVector method), 99  
 set\_set\_denominator() (httk.MutableFracVector method), 47  
 set\_simplify() (httk.core.vectors.mutablefracvector.MutableFracVector method), 99  
 set\_simplify() (httk.MutableFracVector method), 47  
 set\_T() (httk.core.vectors.mutablefracvector.MutableFracVector method), 98  
 set\_T() (httk.MutableFracVector method), 47  
 setup\_phasediagram() (in module httk.atomistic.structurephasediagram), 84  
 sha256file() (in module httk.core.crypto), 107  
 show() (httk.analysis.matsci.vis.phasediagramvisualizerplugin.PhaseDiagramVisualizerPlugin method), 49  
 show() (httk.atomistic.vis.asestructurevisualizer.AseStructureVisualizer method), 65  
 show() (httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method), 66  
 show() (httk.atomistic.vis.structurephasediagramvisualizerplugin.StructurePhasedDiagramVisualizerPlugin method), 66  
 show() (httk.atomistic.vis.structurevisualizerplugin.StructureVisualizerPlugin method), 66  
 show() (in module httk.external.gulp\_ext), 119  
 sign() (httk.core.vectors.fracvector.FracVector method), 96  
 sign() (httk.FracVector method), 45  
 sign() (in module httk.core.vectors.vectormath), 104  
 Signature (class in httk), 49  
 Signature (class in httk.core.signature), 112  
 signature() (in module httk.core.ed25519), 108  
 SignatureKey (class in httk), 49  
 SignatureKey (class in httk.core.signature), 112  
 simplex\_le\_solver() (in module httk.core.geometry), 109  
 simplify() (httk.core.vectors.fracvector.FracVector method), 96  
 simplify() (httk.FracVector method), 45  
 sin() (httk.core.vectors.fracvector.FracVector method), 96  
 sin() (httk.FracVector method), 45  
 sin() (in module httk.core.vectors.vectormath), 104  
 sinh() (in module httk.core.vectors.vectormath), 104  
 SiteAssignment (class in httk.atomistic.siteassignment), 75  
 Sites (class in httk.atomistic.sites), 75  
 sites\_tidy() (in module httk.atomistic.sitesutils), 76  
 sites\_to\_platon() (in module httk.iface.platon\_if), 127  
 sort\_coordgroups() (in module httk.atomistic.sitesutils), 76  
 sort\_coordgroups() (in module httk.atomistic.structureutils), 86  
 spacegroup (httk.atomistic.SpaceGroup class in httk.atomistic.spacegroup), 77  
 spacegroup (httk.atomistic.Structure attribute), 54  
 spacegroup (httk.atomistic.structure.Structure attribute), 82  
 spacegroup\_filter\_specific() (in module httk.atomistic.spacegrouputils), 77  
 spacegroup\_get\_hall() (in module httk.atomistic.data.spacegroups), 64  
 spacegroup\_get\_hall() (in module httk.atomistic.spacegrouputils), 78  
 spacegroup\_get\_hm() (in module httk.atomistic.data.spacegroups), 64  
 spacegroup\_get\_hm() (in module httk.atomistic.spacegrouputils), 78  
 spacegroup\_get\_number() (in module httk.atomistic.spacegrouputils), 78  
 spacegroup\_get\_number() (in module httk.atomistic.data.spacegroups), 64  
 spacegroup\_get\_number\_and\_setting() (in module httk.atomistic.spacegrouputils), 78  
 spacegroup\_get\_number\_and\_setting() (in module httk.atomistic.data.spacegroups), 64  
 spacegroup\_get\_number\_of\_settings() (in module httk.atomistic.spacegrouputils), 78  
 spacegroup\_get\_schoenflies() (in module httk.atomistic.data.spacegroups), 64  
 spacegroup\_get\_schoenflies() (in module httk.atomistic.spacegrouputils), 78  
 spacegroup\_number (httk.atomistic.Structure attribute), 55  
 spacegroup\_number (httk.atomistic.structure.Structure attribute), 82  
 spacegroup\_number\_and\_setting (httk.atomistic.Structure attribute), 55  
 spacegroup\_number\_and\_setting (httk.atomistic.structure.Structure attribute), 82  
 spacegroup\_parse() (in module httk.atomistic.spacegrouputils), 78  
 spglib\_out\_to\_struct() (in module httk.iface.spglib\_if), 127  
 spin() (httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method), 66  
 sql() (httk.db.filteredcollection.FCSqlite method), 116  
 Sqlite (class in httk.db.backend.sqlite), 112  
 Sqlite.SqliteCursor (class in httk.db.backend.sqlite), 112  
 SqlStore (class in httk.db.store.sqlstore), 114  
 SqlStore.Keeper (class in httk.db.store.sqlstore), 114  
 sqrt() (httk.core.vectors.fracvector.FracVector method), 96  
 sqrt() (httk.FracVector method), 45  
 sqrt() (in module httk.core.vectors.vectormath), 104  
 stack\_vecs() (httk.core.vectors.fracvector.FracVector

- class method), 96
- stack\_vecs() (httk.core.vectors.vector.Vector class method), 100
- stack\_vecs() (httk.FracVector class method), 45
- standard\_order\_axes\_transform() (in module httk.atomistic.cellutils), 71
- standard\_primitive() (in module httk.external.afflow\_ext), 118
- start() (httk.external.command.Command method), 119
- start() (in module httk.external.jmol), 120
- stdin (httk.external.command.Command attribute), 119
- stop() (httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method), 66
- stop() (httk.external.command.Command method), 119
- Storable (class in httk.db.storable), 117
- storable\_init() (httk.db.storable.Storable method), 117
- storable\_props() (in module httk.db.storable), 118
- storable\_types() (in module httk.db.storable), 118
- store() (httk.db.httkobjdbplugin.HttkObjDbPlugin method), 117
- store\_codependent\_data() (httk.db.httkobjdbplugin.HttkObjDbPlugin method), 117
- store\_table() (httk.db.filteredcollection.FCSqlite method), 116
- store\_table() (httk.db.filteredcollection.FilteredCollection method), 116
- string (httk.core.ioadapters.IoAdapterString attribute), 110
- string (httk.IoAdapterString attribute), 48
- string\_to\_val\_and\_delta() (in module httk.core.vectors.fracmath), 92
- struct\_process\_with\_isotropy() (in module httk.external.isotropy\_ext), 119
- struct\_to\_cif() (in module httk.atomistic.atomisticio.structure\_cif\_io), 63
- struct\_to\_cif\_httk\_simplified() (in module httk.atomistic.atomisticio.structure\_cif\_io), 63
- struct\_to\_cifdata() (in module httk.atomistic.atomisticio.structure\_cif\_io), 63
- struct\_to\_input() (in module httk.iface.isotropy\_if), 123
- Structure (class in httk.atomistic), 50
- Structure (class in httk.atomistic.structure), 78
- structure\_addsym\_and\_tidy() (in module httk.external.platon\_ext), 120
- structure\_reduced\_coordgroups\_to\_representative() (in module httk.atomistic.sitesutils), 76
- structure\_reduced\_uc\_to\_representative() (in module httk.atomistic.structureutils), 86
- structure\_tidy() (in module httk.atomistic.structureutils), 86
- structure\_tidy() (in module httk.external.platon\_ext), 120
- structure\_tidy\_old() (in module httk.external.platon\_ext), 120
- structure\_to\_ase\_atoms() (in module httk.external.ase\_glue), 118
- structure\_to\_comment() (in module httk.iface.vasp\_if), 128
- structure\_to\_gulp() (in module httk.iface.gulp\_if), 123
- structure\_to\_jmol() (in module httk.iface.jmol\_if), 123
- structure\_to\_plstructure() (in module httk.atomistic.structureutils), 86
- StructureVisualizerPlaton() (in module httk.iface.platon\_if), 127
- structure\_to\_poscar() (in module httk.iface.vasp\_if), 128
- structure\_to\_sgstructure() (in module httk.atomistic.structureutils), 86
- structure\_to\_sgstructure() (in module httk.external.platon\_ext), 120
- structure\_to\_spglib\_atoms() (in module httk.external.pyspglib\_ext), 120
- StructureAsePlugin (class in httk.external.ase\_glue), 118
- StructureFormulaPlugin (class in httk.atomistic.formulautils), 72
- StructureIoPlugin (class in httk.atomistic.atomisticio.structureioplugin), 64
- StructurePhaseDiagram (class in httk.atomistic), 58
- StructurePhaseDiagram (class in httk.atomistic.structurephasediagram), 84
- StructurePhaseDiagramCompetingIndicies (class in httk.atomistic.structurephasediagram), 84
- StructurePhaseDiagramVisualizerPlugin (class in httk.atomistic.vis.structurephasediagramvisualizerplugin), 66
- StructureRef (class in httk.atomistic), 58
- StructureRef (class in httk.atomistic.structure), 83
- StructureSupercellPlugin (class in httk.atomistic.supercellutils), 86
- StructureTag (class in httk.atomistic), 58
- StructureTag (class in httk.atomistic.structure), 83
- StructureVisualizerPlugin (class in httk.atomistic.vis.structurevisualizerplugin), 66
- subdata() (httk.db.filteredcollection.FCMultiDict method), 115
- submit\_reader() (in module httk.task.reader), 128
- submodule\_import\_external() (in module httk.external.subimport), 120
- subtable() (httk.db.filteredcollection.FCSqlite method), 116
- supercell (httk.atomistic.Structure attribute), 55
- supercell (httk.atomistic.structure.Structure attribute), 82
- supercell (httk.atomistic.UnitcellStructure attribute), 61
- supercell (httk.atomistic.unitcellstructure.UnitcellStructure attribute), 89

- symbol (httk.atomistic.assignment.Assignment attribute), 67
  - symbol (httk.atomistic.siteassignment.SiteAssignment attribute), 75
  - sybollists (httk.atomistic.Assignments attribute), 57
  - sybollists (httk.atomistic.assignments.Assignments attribute), 67
  - symbols (httk.atomistic.Assignments attribute), 57
  - symbols (httk.atomistic.assignments.Assignments attribute), 67
  - symbols (httk.atomistic.siteassignment.SiteAssignment attribute), 75
  - symbols (httk.atomistic.Structure attribute), 55
  - symbols (httk.atomistic.structure.Structure attribute), 82
  - symopshash() (in module httk.atomistic.spacegrouputils), 78
  - symopsmatrix() (in module httk.atomistic.spacegrouputils), 78
  - symopstuple() (in module httk.atomistic.spacegrouputils), 78
- ## T
- T() (httk.core.vectors.fracvector.FracVector method), 92
  - T() (httk.FracVector method), 41
  - table() (httk.db.filteredcollection.FCSqlite method), 116
  - table\_exists() (httk.db.backend.sqlite.Sqlite method), 113
  - TableOrColumn (class in httk.db.filteredcollection), 117
  - tan() (in module httk.core.vectors.vectormath), 104
  - tanh() (in module httk.core.vectors.vectormath), 104
  - tidy() (httk.atomistic.RepresentativeSites method), 57
  - tidy() (httk.atomistic.representativesites.RepresentativeSites method), 73
  - tidy() (httk.atomistic.Structure method), 55
  - tidy() (httk.atomistic.structure.Structure method), 82
  - to() (httk.core.httkobject.HttkObject method), 109
  - to() (httk.HttkObject method), 48
  - to\_Atoms() (httk.external.ase\_glue.StructureAsePlugin method), 118
  - to\_basis() (httk.atomistic.Assignments method), 58
  - to\_basis() (httk.atomistic.assignments.Assignments method), 67
  - to\_basis() (httk.atomistic.siteassignment.SiteAssignment method), 75
  - to\_float() (httk.core.vectors.fracvector.FracVector method), 96
  - to\_float() (httk.FracVector method), 45
  - to\_floats() (httk.core.vectors.fracvector.FracVector method), 96
  - to\_floats() (httk.FracVector method), 45
  - to\_fraction() (httk.core.vectors.fracvector.FracVector method), 96
  - to\_fraction() (httk.FracVector method), 46
  - to\_fractions() (httk.core.vectors.fracvector.FracVector method), 96
  - to\_fractions() (httk.FracVector method), 46
  - to\_FracVector() (httk.core.vectors.mutablefracvector.MutableFracVector method), 99
  - to\_FracVector() (httk.MutableFracVector method), 47
  - to\_int() (httk.core.vectors.fracvector.FracVector method), 96
  - to\_int() (httk.FracVector method), 46
  - to\_ints() (httk.core.vectors.fracvector.FracVector method), 97
  - to\_ints() (httk.FracVector method), 46
  - to\_string() (httk.core.vectors.fracvector.FracVector method), 97
  - to\_string() (httk.FracVector method), 46
  - to\_strings() (httk.core.vectors.fracvector.FracVector method), 97
  - to\_strings() (httk.FracVector method), 46
  - to\_tuple() (httk.core.httkobject.HttkObject method), 109
  - to\_tuple() (httk.core.vectors.fracvector.FracVector method), 97
  - to\_tuple() (httk.FracVector method), 46
  - to\_tuple() (httk.HttkObject method), 48
  - total\_number\_of\_atoms (httk.atomistic.RepresentativeSites attribute), 57
  - total\_number\_of\_atoms (httk.atomistic.representativesites.RepresentativeSites attribute), 73
  - total\_number\_of\_atoms (httk.atomistic.sites.Sites attribute), 76
  - total\_number\_of\_atoms (httk.atomistic.UnitcellSites attribute), 57
  - total\_number\_of\_atoms (httk.atomistic.unitcellsites.UnitcellSites attribute), 87
  - transform() (httk.atomistic.Structure method), 55
  - transform() (httk.atomistic.structure.Structure method), 82
  - transform() (httk.atomistic.UnitcellStructure method), 61
  - transform() (httk.atomistic.unitcellstructure.UnitcellStructure method), 89
  - transform() (in module httk.atomistic.structureutils), 86
  - trivial\_symmetry\_reduce() (in module httk.atomistic.spacegrouputils), 78
  - TrivialStore (class in httk.db.storable), 118
  - trivialstore (httk.db.storable.Storable attribute), 117
  - trunc() (in module httk.core.vectors.vectormath), 104
  - tuple\_eye() (in module httk.core.vectors.fracvector), 97
  - tuple\_eye() (in module httk.core.vectors.vector), 100
  - tuple\_index() (in module httk.core.vectors.fracvector), 97
  - tuple\_index() (in module httk.core.vectors.vector), 100
  - tuple\_random() (in module httk.core.vectors.fracvector), 97
  - tuple\_random() (in module httk.core.vectors.vector), 101
  - tuple\_slice() (in module httk.core.vectors.fracvector), 97
  - tuple\_slice() (in module httk.core.vectors.vector), 101
  - tuple\_to\_hexhash() (in module httk.core.crypto), 107
  - tuple\_to\_str() (in module httk.core.basic), 105

tuple\_to\_str() (in module httk.core.crypto), 107  
 tuple\_zeros() (in module httk.core.vectors.fracvector), 97  
 tuple\_zeros() (in module httk.core.vectors.vector), 101  
 types() (httk.core.httkobject.HttkObject class method), 109  
 types() (httk.HttkObject class method), 48

## U

uc (httk.atomistic.Structure attribute), 55  
 uc (httk.atomistic.structure.Structure attribute), 82  
 uc\_a (httk.atomistic.Structure attribute), 55  
 uc\_a (httk.atomistic.structure.Structure attribute), 82  
 uc\_a (httk.atomistic.UnitcellStructure attribute), 61  
 uc\_a (httk.atomistic.unitcellstructure.UnitcellStructure attribute), 89  
 uc\_alpha (httk.atomistic.Structure attribute), 55  
 uc\_alpha (httk.atomistic.structure.Structure attribute), 83  
 uc\_alpha (httk.atomistic.UnitcellStructure attribute), 61  
 uc\_alpha (httk.atomistic.unitcellstructure.UnitcellStructure attribute), 89  
 uc\_b (httk.atomistic.Structure attribute), 55  
 uc\_b (httk.atomistic.structure.Structure attribute), 83  
 uc\_b (httk.atomistic.UnitcellStructure attribute), 61  
 uc\_b (httk.atomistic.unitcellstructure.UnitcellStructure attribute), 89  
 uc\_basis (httk.atomistic.Structure attribute), 55  
 uc\_basis (httk.atomistic.structure.Structure attribute), 83  
 uc\_basis (httk.atomistic.UnitcellStructure attribute), 61  
 uc\_basis (httk.atomistic.unitcellstructure.UnitcellStructure attribute), 89  
 uc\_beta (httk.atomistic.Structure attribute), 55  
 uc\_beta (httk.atomistic.structure.Structure attribute), 83  
 uc\_beta (httk.atomistic.UnitcellStructure attribute), 61  
 uc\_beta (httk.atomistic.unitcellstructure.UnitcellStructure attribute), 89  
 uc\_c (httk.atomistic.Structure attribute), 55  
 uc\_c (httk.atomistic.structure.Structure attribute), 83  
 uc\_c (httk.atomistic.UnitcellStructure attribute), 61  
 uc\_c (httk.atomistic.unitcellstructure.UnitcellStructure attribute), 89  
 uc\_cartesian\_coordgroups (httk.atomistic.Structure attribute), 55  
 uc\_cartesian\_coordgroups (httk.atomistic.structure.Structure attribute), 83  
 uc\_cartesian\_coordgroups (httk.atomistic.UnitcellStructure attribute), 61  
 uc\_cartesian\_coordgroups (httk.atomistic.unitcellstructure.UnitcellStructure attribute), 89  
 uc\_cartesian\_coords (httk.atomistic.Structure attribute), 55  
 uc\_cartesian\_coords (httk.atomistic.structure.Structure attribute), 83  
 uc\_cartesian\_coords (httk.atomistic.UnitcellStructure attribute), 61  
 uc\_cartesian\_coords (httk.atomistic.unitcellstructure.UnitcellStructure attribute), 89  
 uc\_cartesian\_occupationscoords (httk.atomistic.Structure attribute), 55  
 uc\_cartesian\_occupationscoords (httk.atomistic.structure.Structure attribute), 83  
 uc\_cartesian\_occupationscoords (httk.atomistic.UnitcellStructure attribute), 61  
 uc\_cartesian\_occupationscoords (httk.atomistic.unitcellstructure.UnitcellStructure attribute), 89  
 uc\_cell (httk.atomistic.Structure attribute), 55  
 uc\_cell (httk.atomistic.structure.Structure attribute), 83  
 uc\_cell\_orientation (httk.atomistic.Structure attribute), 55  
 uc\_cell\_orientation (httk.atomistic.structure.Structure attribute), 83  
 uc\_cell\_orientation (httk.atomistic.UnitcellStructure attribute), 61  
 uc\_cell\_orientation (httk.atomistic.unitcellstructure.UnitcellStructure attribute), 89  
 uc\_counts (httk.atomistic.Structure attribute), 55  
 uc\_counts (httk.atomistic.structure.Structure attribute), 83  
 uc\_counts (httk.atomistic.UnitcellStructure attribute), 61  
 uc\_counts (httk.atomistic.unitcellstructure.UnitcellStructure attribute), 89  
 uc\_formula\_counts (httk.atomistic.Structure attribute), 55  
 uc\_formula\_counts (httk.atomistic.structure.Structure attribute), 83  
 uc\_formula\_parts (httk.atomistic.Structure attribute), 55  
 uc\_formula\_parts (httk.atomistic.structure.Structure attribute), 83  
 uc\_formula\_symbols (httk.atomistic.Structure attribute), 55  
 uc\_formula\_symbols (httk.atomistic.structure.Structure attribute), 83  
 uc\_gamma (httk.atomistic.Structure attribute), 55  
 uc\_gamma (httk.atomistic.structure.Structure attribute), 83  
 uc\_gamma (httk.atomistic.UnitcellStructure attribute), 61  
 uc\_gamma (httk.atomistic.unitcellstructure.UnitcellStructure attribute), 89  
 uc\_lengths\_and\_angles (httk.atomistic.Structure attribute), 55  
 uc\_lengths\_and\_angles (httk.atomistic.structure.Structure attribute), 83  
 uc\_lengths\_and\_angles (httk.atomistic.UnitcellStructure attribute), 61  
 uc\_lengths\_and\_angles (httk.atomistic.unitcellstructure.UnitcellStructure attribute), 89  
 uc\_nbr\_atoms (httk.atomistic.Structure attribute), 55

uc\_nbr\_atoms (httk.atomistic.structure.Structure attribute), 83  
 uc\_occupancies (httk.atomistic.Structure attribute), 55  
 uc\_occupancies (httk.atomistic.structure.Structure attribute), 83  
 uc\_occupationssymbols (httk.atomistic.Structure attribute), 55  
 uc\_occupationssymbols (httk.atomistic.structure.Structure attribute), 83  
 uc\_reduced\_coordgroups (httk.atomistic.Structure attribute), 55  
 uc\_reduced\_coordgroups (httk.atomistic.structure.Structure attribute), 83  
 uc\_reduced\_coordgroups (httk.atomistic.UnitcellStructure attribute), 61  
 uc\_reduced\_coordgroups (httk.atomistic.unitcellstructure.UnitcellStructure attribute), 89  
 uc\_reduced\_coordgroups\_process\_with\_isotropy() (in module httk.external.isotropy\_ext), 119  
 uc\_reduced\_coors (httk.atomistic.Structure attribute), 55  
 uc\_reduced\_coors (httk.atomistic.structure.Structure attribute), 83  
 uc\_reduced\_coors (httk.atomistic.UnitcellStructure attribute), 61  
 uc\_reduced\_coors (httk.atomistic.unitcellstructure.UnitcellStructure attribute), 89  
 uc\_reduced\_occupationscoors (httk.atomistic.Structure attribute), 55  
 uc\_reduced\_occupationscoors (httk.atomistic.structure.Structure attribute), 83  
 uc\_sites (httk.atomistic.Structure attribute), 55  
 uc\_sites (httk.atomistic.structure.Structure attribute), 83  
 uc\_structure\_to\_symbols\_and\_scaled\_positions() (in module httk.iface.ase\_if), 122  
 uc\_volume (httk.atomistic.Structure attribute), 55  
 uc\_volume (httk.atomistic.structure.Structure attribute), 83  
 uc\_volume (httk.atomistic.UnitcellStructure attribute), 61  
 uc\_volume (httk.atomistic.unitcellstructure.UnitcellStructure attribute), 89  
 uc\_volume\_per\_atom (httk.atomistic.RepresentativeStructure attribute), 63  
 uc\_volume\_per\_atom (httk.atomistic.representativestructure.RepresentativeStructure attribute), 74  
 uc\_volume\_per\_atom (httk.atomistic.UnitcellStructure attribute), 61  
 uc\_volume\_per\_atom (httk.atomistic.unitcellstructure.UnitcellStructure attribute), 89  
 UnaryBooleanOp (class in httk.db.filteredcollection), 117  
 UnitcellSites (class in httk.atomistic), 57  
 UnitcellSites (class in httk.atomistic.unitcellsites), 87  
 UnitcellStructure (class in httk.atomistic), 59  
 UnitcellStructure (class in httk.atomistic.unitcellstructure), 87  
 universal\_opener() (in module httk.core.ioadapters), 110  
 update() (httk.db.backend.sqlite.Sqlite method), 113  
 update\_row() (httk.db.backend.sqlite.Sqlite method), 113  
 use() (httk.atomistic.assignment.Assignment class method), 67  
 use() (httk.atomistic.Assignments class method), 58  
 use() (httk.atomistic.assignments.Assignments class method), 67  
 use() (httk.atomistic.Cell class method), 56  
 use() (httk.atomistic.cell.Cell class method), 68  
 use() (httk.atomistic.RepresentativeStructure class method), 63  
 use() (httk.atomistic.representativestructure.RepresentativeStructure class method), 74  
 use() (httk.atomistic.siteassignment.SiteAssignment class method), 75  
 use() (httk.atomistic.sites.Sites class method), 76  
 use() (httk.atomistic.Structure class method), 55  
 use() (httk.atomistic.structure.Structure class method), 83  
 use() (httk.atomistic.UnitcellStructure class method), 61  
 use() (httk.atomistic.unitcellstructure.UnitcellStructure class method), 89  
 use() (httk.core.httkobject.HttkObject class method), 109  
 use() (httk.core.ioadapters.IoAdapterFileAppender class method), 110  
 use() (httk.core.ioadapters.IoAdapterFilename class method), 110  
 use() (httk.core.ioadapters.IoAdapterFileReader class method), 110  
 use() (httk.core.ioadapters.IoAdapterFileWriter class method), 110  
 use() (httk.core.ioadapters.IoAdapterString class method), 110  
 use() (httk.core.ioadapters.IoAdapterStringList class method), 110  
 use() (httk.core.vectors.fracvector.FracVector class method), 97  
 use() (httk.core.vectors.mutablefracvector.MutableFracVector class method), 99  
 use() (httk.core.vectors.vector.Vector class method), 100  
 use() (httk.FracVector class method), 46  
 use() (httk.HttkObject class method), 48  
 use() (httk.IoAdapterFileAppender class method), 48  
 use() (httk.IoAdapterFileReader class method), 48  
 use() (httk.IoAdapterFileWriter class method), 48  
 use() (httk.IoAdapterString class method), 48  
 use() (httk.IoAdapterStringList class method), 48  
 use() (httk.MutableFracVector class method), 47

## V

val\_to\_tuple() (in module httk.atomistic.spacegrouputils), 78

[validate\(\)](#) (httk.core.vectors.fracvector.FracVector method), [97](#)  
[validate\(\)](#) (httk.core.vectors.mutablefracvector.MutableFracVector method), [99](#)  
[validate\(\)](#) (httk.FracVector method), [46](#)  
[validate\(\)](#) (httk.MutableFracVector method), [47](#)  
[variable\(\)](#) (httk.db.filteredcollection.FilteredCollection method), [117](#)  
[variable\(\)](#) (httk.db.storable.Storable class method), [118](#)  
[Vector](#) (class in httk.core.vectors.vector), [99](#)  
[verify\\_crypto\\_signature\(\)](#) (in module httk.core.crypto), [107](#)  
[verify\\_crypto\\_signature\\_old\(\)](#) (in module httk.core.crypto), [107](#)  
[vis](#) (httk.analysis.matsci.phasediagram.PhaseDiagram attribute), [50](#)  
[vol\\_to\\_scale\(\)](#) (in module httk.atomistic.cellutils), [71](#)  
[volume](#) (httk.atomistic.Cell attribute), [57](#)  
[volume](#) (httk.atomistic.cell.Cell attribute), [68](#)  
[volume\\_per\\_atom](#) (httk.atomistic.Structure attribute), [55](#)  
[volume\\_per\\_atom](#) (httk.atomistic.structure.Structure attribute), [83](#)

## W

[wait\(\)](#) (httk.analysis.matsci.vis.phasediagramvisualizerplugin.PhaseDiagramVisualizerPlugin method), [49](#)  
[wait\(\)](#) (httk.atomistic.vis.asestructurevisualizer.AseStructureVisualizer method), [65](#)  
[wait\(\)](#) (httk.atomistic.vis.jmolstructurevisualizer.JmolStructureVisualizer method), [66](#)  
[wait\(\)](#) (httk.atomistic.vis.structurevisualizerplugin.StructureVisualizerPlugin method), [66](#)  
[wait\\_finish\(\)](#) (httk.external.command.Command method), [119](#)  
[write\\_cif\(\)](#) (in module httk.httkio.cif), [121](#)  
[write\\_generic\\_kpoints\\_file\(\)](#) (in module httk.iface.vasp\_if), [128](#)  
[write\\_kpoints\\_file\(\)](#) (in module httk.iface.vasp\_if), [128](#)  
[write\\_poscar\(\)](#) (in module httk.iface.vasp\_if), [128](#)  
[wyckoff\\_sequence](#) (httk.atomistic.Compound attribute), [58](#)  
[wyckoff\\_sequence](#) (httk.atomistic.compound.Compound attribute), [72](#)  
[wyckoff\\_sequence](#) (httk.atomistic.RepresentativeSites attribute), [57](#)  
[wyckoff\\_sequence](#) (httk.atomistic.representativesites.RepresentativeSites attribute), [73](#)  
[wyckoff\\_sequence](#) (httk.atomistic.Structure attribute), [55](#)  
[wyckoff\\_sequence](#) (httk.atomistic.structure.Structure attribute), [83](#)  
[wyckoff\\_symbol\\_matcher\(\)](#) (in module httk.atomistic.spacegrouputils), [78](#)

**X**  
[xrecover\(\)](#) (in module httk.core.ed25519), [108](#)

**Z**  
[zdecompressor\(\)](#) (in module httk.core.ioadapters), [110](#)  
[zeros\(\)](#) (httk.core.vectors.fracvector.FracVector class method), [97](#)  
[zeros\(\)](#) (httk.core.vectors.vector.Vector class method), [100](#)  
[zeros\(\)](#) (httk.FracVector class method), [46](#)