

---

# hubugs

*Release 0.19.0*

Feb 15, 2020



---

## Contents

---

<b>1</b>	<b>Contents</b>	<b>3</b>
1.1	Background . . . . .	3
1.2	Installation . . . . .	3
1.3	Configuration . . . . .	4
1.4	Usage . . . . .	4
1.5	Templates . . . . .	8
1.6	hubugs . . . . .	12
1.7	Frequently Asked Questions . . . . .	15
1.8	Alternatives . . . . .	16
1.9	Release HOWTO . . . . .	17
1.10	Todo items for documentation . . . . .	18
1.11	API documentation . . . . .	18
1.12	Appendix . . . . .	23
1.13	Glossary . . . . .	25
<b>2</b>	<b>Indices and tables</b>	<b>27</b>
	<b>Python Module Index</b>	<b>29</b>
	<b>Index</b>	<b>31</b>



**Warning:** I don't really use GitHub anymore, so this package is somewhat orphaned. It should still work *and* I will fix bugs when they're pointed out, but hubugs is frankly quite a low priority for me.

*hubugs* is a very simple client for working with GitHub's issue tracker from the command line.

It is written in Python, and works with Python 3.5 and newer. *hubugs* is released under the GPL v3

**Git repository** <https://github.com/JNRowe/hubugs/>

**Issue tracker** <https://github.com/JNRowe/hubugs/issues/>

**Contributors** <https://github.com/JNRowe/hubugs/contributors/>



## 1.1 Background

[GitHub](#) provide a great issue tracker for small projects. It is simple, light and fast. There's only one problem for me, I prefer command line tools for these tasks.

Luckily, the issue tracker exposes your data via a [thoroughly documented API](#), so alternative interfaces are only a SMOP (Small Matter of Programming) away.

### 1.1.1 The requirements

I live at the command line, data that isn't easily manageable from there may as well not exist in my eyes. Therefore:

A simple [command line interface](#) is priority one.

It is nice to customise the views of data, and improve the look and feel of a view over time. Therefore:

Views must be customisable via simple [templates](#)

Data display must be fast, and require the minimum possible network access. Therefore:

Network data must be [cached and compressed](#) where possible

## 1.2 Installation

You can install [hubugs](#) either via PYPI (Python Package Index) or from source.

### 1.2.1 Using PYPI

To install using [pip](#):

```
$ pip install hubugs # to install in Python's site-packages
$ pip install --install-option="--user" hubugs # to install for a single user
```

To install using [easy\\_install](#):

```
$ easy_install hubugs
```

## 1.2.2 From source

If you have downloaded a source tarball you can install it with the following steps:

```
$ python setup.py build
# python setup.py install # to install in Python's site-packages
$ python setup.py install --user # to install for a single user
```

*hubugs* depends on following packages, all of which are available from PYPI:

- [click](#) an excellent package for building command line tools in Python
- [configparser](#) for configuration file handling
- [html2text](#) is used formatting HTML for the terminal
- [httplib2](#) for HTTP communications
- [Jinja2](#) for templating
- [misaka](#) is used for converting issue text to HTML
- [Pygments](#) for syntax highlighting in template output

## 1.3 Configuration

Before **hubugs** can operate on issues you must generate an [OAuth](#) token. **hubugs** provides functionality to do this:

```
hubugs setup
GitHub user? [JNRowe]
GitHub password? <password>
Support private repositories? (Y/n) y
Configuration complete!
```

---

**Note:** You can revoke the generated token at any time from the [GitHub settings](#) page.

---

If you wish to set the authorisation token from the command line you can use the `HUBUGS_TOKEN` environment variable. For example:

```
HUBUGS_TOKEN=xxx hubugs open
```

## 1.4 Usage

The **hubugs** script is the main workhorse of *hubugs*.

Let's start with some basic examples:



```

hubugs list # List the open bugs for the current project
Id Title
5 Handle some GitHub markdown extensions [feature]
6 Sphinx documentation tree [task]
7 Support per project templates [feature]

3 open bugs found

hubugs search markdown # Search for bugs matching markdown
Id Title
5 Handle some GitHub markdown extensions [feature]

1 open bug found matching markdown

hubugs show 6 # Show bug number 6
    Id: 6
    Title: Sphinx documentation tree
    Labels: task
    Created: yesterday by JNRowe
    Updated: yesterday
    State: open
    Comments: 0
Pull request: No

This project deserves some real user documentation, not just a few notes in
`README.rst`.

hubugs comment 6 # Comment on bug 6 using your editor

hubugs comment -m"New comment." 6 # Add comment from command line

```

### 1.4.1 Options

- version**  
show program's version number and exit
- h, --help**  
show program's help message and exit
- pager** <pager>  
pass output through a pager
- no-pager**  
do not pass output through pager
- p** <project>, **--project**=<project>  
GitHub project to operate on
- u** <url>, **--host-url**=<url>  
host to connect to, for GitHub Enterprise support

---

**Note:** You can set a default value for the `--pager` and `--host-url` options by defining `hubugs.pager` or `hubugs.host-url` respectively in your `git` configuration files. Both global and project local settings are supported, see `git-config(1)` for more information.

---

## 1.4.2 Commands

### setup - Generate a new GitHub access token

```
hubugs.py setup [-h] [--local]
```

#### **--local**

set access token for local repository only

### list - List bugs for a project

```
hubugs list [-h] [-s {open,closed,all}] [-l label]
          [-o {number,updated}]
```

**-s** <state>, **--state**=<state>  
state of bugs to operate on

**-l** <label>, **--label**=<label>  
list bugs with specified label

**-o** <order>, **--order**=<order>  
sort order for listing bugs

**-p** <number>, **--page** <number>  
page number

**-r**, **--pull-requests**  
list only pull requests

### search - Search bugs reports in a project

```
hubugs search [-h] [-s {open,closed,all}]
          [-o {number,updated}]
          term
```

**-s** <state>, **--state**=<state>  
state of bugs to operate on

**-o** <order>, **--order**=<order>  
sort order for listing bugs

### show - Show specific bug(s) from a project

```
hubugs show [-h] [-f] [-p] bugs [bugs ...]
```

**-f**, **--full**  
show bug including comments

**-p**, **--patch**  
display patches for pull requests

**-o**, **--patch-only**  
display only the patch content of pull requests

**-b, --browse**  
open bug in web browser

### open - Open a new bug in a project

```
hubugs open [-h] [-a label] [--stdin] [title] [body]
```

**-a label, --add label**  
add label to issue

**--stdin**  
read message from standard input

### comment - Comment on an existing bug in a project

```
hubugs comment [-h] [--stdin] [-m MESSAGE] bugs [bugs ...]
```

**--stdin**  
read message from standard input

**-m <text>, --message=<text>**  
comment text

### edit - Edit an existing bug in a project

```
hubugs edit [-h] [--stdin] [title] [body] bugs [bugs ...]
```

**--stdin**  
read message from standard input

### close - Close an existing bug in a project

```
hubugs close [-h] [--stdin] [-m MESSAGE] bugs [bugs ...]
```

**--stdin**  
read message from standard input

**-m <text>, --message=<text>**  
comment text

### reopen - Reopen a previously closed bug in a project

```
reopen [-h] [--stdin] [-m MESSAGE] bugs [bugs ...]
```

**--stdin**  
read message from standard input

**-m <text>, --message=<text>**  
comment text

### label - Perform labelling actions on an existing bug in a project

```
hubugs label [-h] [-a label] [-r label] bugs [bugs ...]
```

**-a** <label>, **--add**=<label>  
add label to issue

**-r** <label>, **--remove**=<label>  
remove label from issue

### milestone - Add an issue to a milestone

```
hubugs milestone [-h] milestone [bugs [bugs ...]]
```

### milestones - Manage repository milestones

```
hubugs milestones [-h] [-o {due_date,completeness}] [-s {open,closed}]  
[-c milestone] [-l]
```

**-o** <order>, **--order**=<order>  
sort order for listing bugs

**-s** <state>, **--state**=<state>  
state of bugs to operate on

**-c** <name>, **--create**=<name>  
create new milestone

**-l**, **--list**  
list available milestones

## 1.5 Templates

Output is produced from templates using [Jinja](#). Before writing your own templates you should read the awesome [Jinja template designer](#) documentation.

---

**Note:** If you create some cool templates of your own please consider posting them in an [issue](#) or pushing them to a fork on [GitHub](#), so that others can benefit.

---

### 1.5.1 Template locations

Templates are loaded from directories in the following order:

- If it exists, `${XDG_DATA_HOME:~/.local/share}/hubugs/templates`
- Any `hubugs/templates` directory in the directories specified by `XDG_DATA_DIRS`
- The package's `templates` directory

For information on the usage of `XDG_DATA_HOME` and `XDG_DATA_DIRS` read [XDG Base Directory Specification](#).

**Note:** For OS X users there is a fallback to `~/Library/Application Support`, if `XDG_DATA_HOME` is unset.

## 1.5.2 Precedence

The first name match in the order specified above selects the template, so a `view/list.txt` in `$XDG_DATA_HOME/hubugs/templates` overrides the `view/list.txt` provided in the *hubugs* package.

## 1.5.3 Template sets

You can specify the template set to use by defining a *hubugs.templates* setting in your git configuration files. For example:

```
git config --global hubugs.templates my_templates
```

You can also set project specific template sets by editing a repository's config. See *git-config(1)*.

## 1.5.4 Naming

Templates are separated in to two groups. The first group, *view*, is for templates used in directly producing consumable output(such as from the `list` subcommand). The second group, *edit*, is for templates used to generate input files for editing text(such as in the `open` subcommand).

*view* group templates currently include:

- `issue.txt` for formatting a single bug
- `list.txt` for formatting list output

*edit* group templates currently include:

- `default.mkd` for general use, such as in commenting on a bug
- `open.mkd` for opening(or editing) bugs

## 1.5.5 Data

The following variables are available for use in templates

### View group

**columns** (*int*)

The width of the current terminal window

### list.txt data

**project** (*Repository*)

The current project's repository data. See *Repository objects*.

**bugs** (*list*)

Contains the sorted list of bugs to display, if any. See *Bug objects*.

**id\_len** (*int*)

Set to the maximum length of the bug IDs to display

**state** (*str*)

The bug states being searched/listed

**order** (*str*)

The display order

**term** (*str*)

The search term being listed, if any

### issue.txt data

**project** (*Repository*)

The current project's repository data. See *Repository objects*.

**bug** (*list*)

Contains the sorted list of bugs to display, if any. See *Bug objects*

**comments** (*list*)

When displaying a single bug this contains the list of comments associated with a bug, if any. See *Comment objects*

**full** (*bool*)

True, if the user provided the *hubugs show -f* option

**patch** (*str*)

The content found at the location in `Bug.patch_url`, if the user provided the *hubugs show -p* option

**patch\_only** (*bool*)

True, if the user provided the *hubugs show -o* option

### Edit group

**title** (*str*)

The current bug title in `edit` subcommand sessions. See *Bug.title*

**body** (*str*)

The current bug body in `edit` subcommand sessions, if any. See *Bug.body*

**comment\_char** (*str*)

The character to use for comments in templates, defaults to `#`. See `core.commentchar` in *git-config(1)*

## All groups

Jinja templates support object attribute and method access, so an individual `bug` object's `created_at` attribute can be called with a `strftime()` method for custom date output. For example, `{{ bug.created_at.strftime("%a, %e %b %Y %H:%M:%S %z") }}` can be used to output an **RFC 2822**-style date stamp.

If you're authoring your own templates and you find you need extra data for their generation open an [issue](#).

## 1.5.6 Filters

*hubugs* defines the following filters beyond the huge range of excellent [built-in filters](#) in Jinja:

---

**Note:** If you write extra filters that you believe could be of use to other *hubugs* users please consider posting them in an [issue](#) or pushing them to a fork on [GitHub](#), so that others can benefit from your work.

---

### colourise

This filter applies a colour to text, if possible.

For example, to show a bug's `title` attribute in red:

```
{{ bug.title | colourise('red') }}
```

or to display black text on a red background:

```
{{ bug.title | colourise('black on red') }}
```

---

**Note:** This filter is also available under the synonym `colorize`.

---

### highlight

This filter highlights text using [Pygments](#). You can specify the lexer to be used, and also the formatter.

For example, to highlight a chunk of text as Python:

```
{{ text | highlight('python') }}
```

To do the same using the 256-colour mode of [Pygments](#):

```
{{ text | highlight('python', 'terminal256') }}
```

See the output of **pygmentize -L** for the list of available lexers and formatters.

### html2text

This filter converts HTML to a plain text representation using [html2text](#).

### markdown

The purpose of this filter is to convert the [Markdown](#) formatted text from a GitHub issue to html. The excellent [misaka](#) package is used to provide the conversion.

In the default templates it is used to render bug bodies:

```
{{ comment.body | markdown | html2text }}
```

---

**Note:** We ping-pong the conversion from Markdown to HTML as it produces a prettier text representation of the comment. We benefit from uniform newline usage and clean word wrapping of the output.

---

### relative\_time

This filter is used to generate a human-readable relative timestamp from a `datetime` object.

For example, to display a bug's `created_at` attribute as a relative time:

```
{{ bug.created_at | relative_time }}
```

which could produce output such as:

```
about two months ago
```

## 1.6 hubugs

### 1.6.1 Simple client for GitHub issues

### 1.6.2 SYNOPSIS

```
hubugs [option]... <command>
```

### 1.6.3 DESCRIPTION

*hubugs* is a very simple client for working with GitHub's issue tracker. It allows you to perform all the issue related tasks you'd normally perform from the command line.

### 1.6.4 OPTIONS

- |  |   |
|--|---|
| <b>--version</b>                                     | show program's version number and exit  |
| <b>-h, --help</b>                                    | show program's help message and exit  |
| <b>--pager &lt;pager&gt;</b>                         | pass output through a pager   |
| <b>--no-pager</b>                                    | do not pass output through pager  |
| <b>-p &lt;project&gt;, --project=&lt;project&gt;</b> | GitHub project to operate on. You can supply just <project> if you wish to work on one of your own projects, or <user>/<project> to operate on another user's repository. Default is derived from the <code>hubugs.project</code> setting in the git config or the current repository, if possible. |



**-u <url>, --host-url=<url>** host to connect to, for GitHub Enterprise support

## 1.6.5 COMMANDS

### setup

Generate a new GitHub access token

**--local** set access token for local repository only

### list

List bugs for a project

**-s <state>, --state=<state>** state of bugs to operate on  
**-l <label>, --label=<label>** list bugs with specified label  
**-o <order>, --order=<order>** sort order for listing bugs  
**-p <number>, --page <number>** page number  
**-r, --pull-requests** list only pull requests

### search

Search bugs reports in a project

**-s <state>, --state=<state>** state of bugs to operate on  
**-o <order>, --order=<order>** sort order for listing bugs

### show

Show specific bug(s) from a project

**-f, --full** show bug including comments  
**-p, --patch** display patches for pull requests  
**-o, --patch-only** display only the patch content of pull requests  
**-b, --browse** open bug in web browser

### open

Open a new bug in a project

**-a label, --add label** add label to issue  
**--stdin** read message from standard input

### comment

Comment on an existing bug in a project

**--stdin**                    read message from standard input  
**-m <text>, --message=<text>** comment text

### edit

Edit an existing bug in a project

**--stdin**                    read message from standard input

### close

Close an existing bug in a project

**--stdin**                    read message from standard input  
**-m <text>, --message=<text>** comment text

### reopen

Reopen a previously closed bug in a project

**--stdin**                    read message from standard input  
**-m <text>, --message=<text>** comment text

### label

Perform labelling actions on an existing bug in a project

**-a <label>, --add=<label>** add label to issue  
**-r <label>, --remove=<label>** remove label from issue

### milestone

Add an issue to a milestone

### milestones

Manage repository milestones

**-o <order>, --order=<order>** sort order for listing bugs  
**-s <state>, --state=<state>** state of bugs to operate on  
**-c <name>, --create=<name>** create new milestone  
**-l, --list**                    list available milestones

## 1.6.6 CONFIGURATION

You can specify the template set to use by defining a `hubugs.templates` setting in your git configuration files. For example:

```
git config --global hubugs.templates my_templates
```

You can also set project specific template sets by editing a repository's config. See `git-config(1)`.

You can set a default value for the `--pager` and `--host-url` options by defining `hubugs.pager` or `hubugs.host-url` respectively in your git configuration files.

## 1.6.7 BUGS

None known.

## 1.6.8 AUTHOR

Written by James Rowe

## 1.6.9 RESOURCES

Home page, containing full documentation: <http://hubugs.readthedocs.io/>

Issue tracker: <https://github.com/JNRowe/hubugs/issues/>

## 1.6.10 COPYING

Copyright © 2010-2016 James Rowe.

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

hubugs 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 General Public License for more details.

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

## 1.7 Frequently Asked Questions

- *Do you accept template contributions?*
- *Why is the wrapping broken in comments I make?*
- *How do I create headlines when lines beginning with # are scrubbed?*
- *I don't like your choice of template language*

### 1.7.1 Do you accept template contributions?

Yes, if they are somewhat general. And, it is a great way to have me maintain template compatibility for you in case something changes in a future version.

Either open an [issue](#) or push them to a fork on [GitHub](#).

### 1.7.2 Why is the wrapping broken in comments I make?

Unfortunately, GitHub have crippled the newline behaviour of Markdown. If you wrap your comments for readability then you'll be creating new paragraphs with every single line. There is very little that can be done locally to fix this.

The easiest way to workaround the issue is to disable wrapping in your text editor for template files. The files `hubugs` creates are easy to match for automating this within your editor, just use `$TMPDIR/hubugs-* .mkd`.

### 1.7.3 How do I create headlines when lines beginning with # are scrubbed?

Markdown syntax supports two different heading formats, and only one of them requires a `#` at the start of a line. Using the alternative `setext` format is simple:

```
Heading
=====

Sub-heading
-----
```

The other possibility, although not recommended, is to set a custom value for `core.commentchar` in your `git` configuration settings. program:*hubugs* will honour the set value, and strip any lines that begin with that character. You should note however that this is a recent addition to `git`, and other tools may not work correctly with it set to a non-default value.

### 1.7.4 I don't like your choice of template language

[It isn't really a question, but it has come up a couple of times.]

The use of [Jinja](#) should only be an issue if you wish to author your own templates, if you're using the built-in templates you shouldn't notice [Jinja](#) at all. That said. . .

The use of [Jinja](#) seems to be an entry barrier to some people, but it isn't going to change. For the same – invariably pointless and religious – reasons people prefer other templating engines *I* prefer [Jinja](#).

---

**Note:** With all that said, I probably wouldn't be opposed to accepting patches supporting additional *optional* engines ;)

---

## 1.8 Alternatives

Before diving in and spitting out this package I looked at the alternatives below. If I have missed something please drop me a [mail](#).

Both the `ghi` and `github-cli` packages listed below are very useful, and quite usable, clients for GitHub issues. You should definitely try them out before making a decision on what to use, and this would also allow you to highlight any possible bias I may have shown in comparing them ;)

### 1.8.1 ghi

`ghi` is a great issues client, written in [Ruby](#).

I personally didn't like the "feel" of the command line interface, and wanted slightly different output.

`hubugs` provides the following advantages:

- Easily customisable output
- On-disk cache support

and has the following disadvantages:

- `hubugs` requires a significant number of external packages, which may be a problem for some users

### 1.8.2 github-cli

`github-cli` is a fantastic command-line client for GitHub's issues, written in [Python](#). I really liked `github-cli` to the extent of authoring a few patches, but just didn't find it comfortable to use in the long run<sup>1</sup>.

`hubugs` provides the following advantages:

- Easily customisable output
- HTTP compression support
- On-disk cache support
- Pull request integration
- Uses the same method as `git` for choosing an editor to use, which will only affect you if you a custom editor for interacting with `git`

and has the following disadvantages:

- `hubugs` requires a significant number of external packages, which may be a problem for some users

## 1.9 Release HOWTO

### 1.9.1 Test

In the general case tests can be run via `pytest`:

```
$ pytest tests
```

When preparing a release it is important to check that `hubugs` works with all currently supported Python versions, and that the documentation is correct.

### 1.9.2 Prepare release

With the tests passing, perform the following steps

- Update the version data in `hubugs/_version.py`
- Update `NEWS.rst`, if there are any user visible changes

<sup>1</sup> If you look at the timeline for the [hubugs repository](#) and my [github-cli fork](#) you'll see I was using `github-cli` for a few months after spiking `hubugs`.

- Commit the release notes and version changes
- Create a signed tag for the release
- Push the changes, including the new tag, to the GitHub repository

### 1.9.3 Update PyPI

Create and upload the new release tarballs to PyPI:

```
$ ./setup.py sdist --formats=bztar,gztar register upload --sign
```

Fetch the uploaded tarballs, and check for errors.

You should also perform test installations from PyPI, to check the experience *hubugs* users will have.

## 1.10 Todo items for documentation

## 1.11 API documentation

---

**Note:** The documentation in this section is aimed at people wishing to contribute to *hubugs*, and can be skipped if you are simply using the tool from the command line.

---

### 1.11.1 Command line

---

**Note:** The documentation in this section is aimed at people wishing to contribute to *hubugs*, and can be skipped if you are simply using the tool from the command line.

---

`hubugs.setup(globs, local)`  
Setup GitHub access token.

`hubugs.list_bugs(globs, label, page, pull_requests, order, state)`  
Listing bugs.

`hubugs.search(globs, order, state, term)`  
Searching bugs.

`hubugs.show(globs, full, patch, patch_only, browse, bugs)`  
Displaying bugs.

`hubugs.open_bug(globs, add, create, stdin, title, body)`  
Opening new bugs.

`hubugs.comment(globs, message, stdin, bugs)`  
Commenting on bugs.

`hubugs.edit(globs, stdin, title, body, bugs)`  
Editing bugs.

`hubugs.close(globs, stdin, message, bugs)`  
Closing bugs.

`hubugs.reopen(globs, stdin, message, bugs)`  
Reopening closed bugs.

`hubugs.label(globs, add, create, remove, list, bugs)`  
Labelling bugs.

`hubugs.milestone(globs, milestone, bugs)`  
Issue milestones.

`hubugs.milestones(globs, order, state, create, list)`  
Repository milestones.

`hubugs.report_bug(globs)`  
Report a new bug against hubugs.

`hubugs.main()`  
Main command-line entry point.

**Return type** `int`

**Returns** Exit code

### 1.11.2 models

---

**Note:** The documentation in this section is aimed at people wishing to contribute to *hubugs*, and can be skipped if you are simply using the tool from the command line.

---

`hubugs.models.object_hook(__d, __name='unknown')`  
JSON object hook to create dot-accessible objects.

**Parameters**

- `__d` (`Dict[str, str]`) – Dictionary to operate on
- `name` – Fallback name, if dict has no `type` key

`hubugs.models._v2_conv_timestamp(__s)`  
Parse API v2 style timestamps.

**Parameters** `__s` (`str`) – Timestamp to parse

`hubugs.models.from_search(__obj)`  
Support legacy API search results as API v3 issues(-ish).

This is an awful hack to workaround the lack of search support in API v3. It needs to be removed at the first possible opportunity.

**Returns** API v2 issue mangled to look like a API v3 result

**Return type** `Issue`

### 1.11.3 Template

---

**Note:** The documentation in this section is aimed at people wishing to contribute to *hubugs*, and can be skipped if you are simply using the tool from the command line.

---

`hubugs.template.get_template(__group, __name)`

Fetch a Jinja template instance.

**Parameters**

- **\_\_group** (`str`) – Template group identifier
- **\_\_name** (`str`) – Template name

**Return type** `Template`

**Returns** Jinja template instance

## Jinja filter support

`hubugs.template.jinja_filter(__func)`

Simple decorator to add a new filter to Jinja environment.

**Parameters** **\_\_func** (`Callable`) – Function to add to Jinja environment

**Return type** `Callable`

**Returns** Unmodified function

`hubugs.template.colourise(__text, fg=None, bg=None, **kwargs)`

Colourise text.

Returns text untouched if colour output is not enabled

**Parameters**

- **\_\_text** (`str`) – Text to colourise
- **fg** (`Optional[str]`) – Foreground colour
- **bg** (`Optional[str]`) – Background colour
- **kwargs** – Formatting to apply to text

**Return type** `str`

**Returns** Colourised text, when possible

`hubugs.template.highlight(__text, lexer='diff', formatter='terminal')`

Highlight text with pygments.

Returns text untouched if colour output is not enabled

**Parameters**

- **\_\_text** (`str`) – Text to highlight
- **lexer** (`Optional[str]`) – Jinja lexer to use
- **formatter** (`Optional[str]`) – Jinja formatter to use

**Return type** `str`

**Returns** Syntax highlighted output, when possible

`hubugs.template.html2text(__html, width=80, ascii_replacements=False)`

HTML to plain text renderer.

**Parameters**

- **\_\_html** (`str`) – Text to process
- **width** (`Optional[int]`) – Paragraph width



- **ascii\_replacements** (Optional[bool]) – Use psuedo-ascii replacements for Uni-code

**Return type** `str`

**Returns** Rendered text

## User interface support

`hubugs.template.display_bugs(__bugs, __order, **extras)`  
Display bugs to users.

### Parameters

- **\_\_bugs** (List[Dict[str, str]]) – Bugs to display
- **\_\_order** (str) – Sorting order for displaying bugs
- **extras** – Additional values to pass to templates

**Return type** `str`

**Returns** Rendered template output

`hubugs.template.edit_text(edit_type='default', data=None)`  
Edit data with external editor.

### Parameters

- **edit\_type** (Optional[str]) – Template to use in editor
- **data** (Optional[str]) – Information to pass to template

**Return type** `str`

**Returns** User supplied text

### Raises

- *EmptyMessageError* – No message given
- *EmptyMessageError* – Message not edited

## 1.11.4 Utilities

---

**Note:** The documentation in this section is aimed at people wishing to contribute to *hubugs*, and can be skipped if you are simply using the tool from the command line.

---

`hubugs.utils.setup_environment(__project, __host_url)`  
Configure execution environment for commands dispatch.

## Convenience functions

`hubugs.utils.get_editor()`  
Choose a suitable editor.

See *git-var(1)* for details.

**Return type** `List[str]`

**Returns** Users chosen editor, or `vi` if not set

`hubugs.utils.pager(__text, pager=False)`  
Pass output through pager.

**Parameters**

- `__text` (`str`) – Text to page
- `pager` (`Optional[bool]`) – Pager to use

## Git/GitHub support

`hubugs.utils.get_github_api()`  
Create a GitHub API instance.

**Returns** GitHub HTTP session

**Return type** `httplib2.Http`

`hubugs.utils.get_git_config_val(__key, default=None, local_only=False)`  
Fetch a git configuration value.

**Parameters**

- `__key` (`str`) – Configuration value to fetch
- `default` (`Optional[str]`) – Default value to use, if key isn't set
- `local_only` (`Optional[bool]`) – Fetch configuration values from repo config only

**Return type** `str`

**Returns** Git config value, if set

`hubugs.utils.set_git_config_val(__key, __value, local_only=False)`  
Set a git configuration value.

**Parameters**

- `__key` (`str`) – Configuration value to fetch
- `__value` (`str`) – Value to set
- `local_only` (`Optional[bool]`) – Set configuration values from repo config only

`hubugs.utils.get_repo()`  
Extract GitHub project name from git/hg config.

We check the git config for `hubugs.project`, and then fall back to `remote.origin.url`. If both of these fail we check a mercurial root, to satisfy the hg-git users.

**Return type** `str`

**Returns** GitHub project name, including user

`hubugs.utils.sync_labels(__globals, __add, __create)`  
Manage labels for a project.

**Parameters** `globals` – Global argument configuration

**Return type** `List[str]`

**Returns** List of project's label names

## 1.11.5 Errors

---

**Note:** The documentation in this section is aimed at people wishing to contribute to *hubugs*, and can be skipped if you are simply using the tool from the command line.

---

**exception** `hubugs.template.EmptyMessageError`

Error to raise when the user provides an empty message.

**exception** `hubugs.utils.HttpClientError` (*message, response, content*)

Error raised for client error status codes.

**exception** `hubugs.utils.RepoError`

Error raised for invalid repository values.

## 1.12 Appendix

### 1.12.1 Bug objects

`Bug.assigned` (*User*)

The user the issue is assigned to, if any.

`Bug.body` (*str*)

The full body of the issue.

`Bug.body_html` (*str*)

The full body of the issue rendered as HTML.

`Bug.closed_at` (*datetime.datetime*)

The date and time the issue was closed.

`Bug.closed_by` (*User*)

The user the issue was closed by, if closed.

`Bug.comments` (*int*)

The number of comments made on the issue.

`Bug.created_at` (*datetime.datetime*)

The date and time the issue was created.

`Bug.labels` (*list*)

Label objects associated with the issue.

`Bug.milestone` (*str*)

The milestone name associated with the issue, if any.

`Bug.number` (*int*)

The issue's number.

`Bug.pull_request` (*PullRequest*)

Pull request data associated with the issue.

`Bug.state` (*str*)

State of the issue, either open or closed.

`Bug.title` (*str*)

The issue's title.

Bug.**updated\_at** (*datetime.datetime*)  
The date and time when the issue was last updated.

Bug.**user** (*User*)  
The GitHub user that created the issue.

### 1.12.2 Comment objects

Comment.**body** (*str*)  
The full text of the comment.

Comment.**body\_html** (*str*)  
The full text of the comment rendered as HTML.

Comment.**created\_at** (*datetime.datetime*)  
The date and time the comment was created.

Comment.**updated\_at** (*datetime.datetime*)  
The date and time when the comment was last updated.

Comment.**user** (*User*)  
The GitHub user that created the comment.

### 1.12.3 Label objects

Label.**color**  
The colour value for the given label.

Label.**name**  
The name given to the label.

### 1.12.4 PullRequest objects

PullRequest.**diff\_url** (*str*)  
URL for **diff** output associated with the issue.

PullRequest.**patch\_url** (*str*)  
URL for the **git format-patch** output associated with the issue.

### 1.12.5 Repository objects

Repository.**clone\_url** (*str*)  
Clone URL for fetching via HTTP.

Repository.**created\_at** (*datetime.datetime*)  
The date and time the repository was created.

Repository.**description** (*str*)  
The description given to the repository

Repository.**fork** (*bool*)  
Whether the repository is a fork of another on GitHub

Repository.**forks** (*int*)  
The number of forks on GitHub

`Repository.git_url (str)`  
Clone URL for fetching via git protocol.

`Repository.has_downloads (bool)`  
Whether the repository has downloads enabled

`Repository.has_issues (bool)`  
Whether the repository has issues enabled

`Repository.has_wiki (bool)`  
Whether the repository has the wiki enabled

`Repository.homepage (str)`  
The homepage of the repository

`Repository.html_url (str)`  
The main project page on GitHub

`Repository.language (str)`  
The programming language used

`Repository.master_branch (str)`  
The repository's defined master branch

`Repository.mirror_url (str)`  
The original location of a repository, if a mirror

`Repository.name (str)`  
The repository's name

`Repository.open_issues (int)`  
The number of open issues in a repository

`Repository.owner (User)`  
The owner of the repository

`Repository.private (bool)`  
Whether the repository is set as private

`Repository.pushed_at (datetime.datetime)`  
The last time the repository's content was updated

`Repository.size (int)`  
The size of the repository

`Repository.watchers (int)`  
The number of watchers of the repository

### 1.12.6 User objects

`avatar_url (str)`  
The location of the user's avatar

`gravatar_id (str)`  
The hash identifier for fetching images from [gravatar](#)

`login (str)`  
The user's login name on GitHub

## 1.13 Glossary



## CHAPTER 2

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`





### h

`hubugs`, [1](#)

### m

`hubugs.models`, [19](#)

### t

`hubugs.template`, [19](#)

### u

`hubugs.utils`, [21](#)



## Symbols

- local
  - hubugs-setup command line option, 6
- no-pager
  - hubugs command line option, 5
- pager <pager>
  - hubugs command line option, 5
- stdin
  - hubugs-close command line option, 7
  - hubugs-comment command line option, 7
  - hubugs-edit command line option, 7
  - hubugs-open command line option, 7
  - hubugs-reopen command line option, 7
- version
  - hubugs command line option, 5
- a <label>, -add=<label>
  - hubugs-label command line option, 8
- a label, -add label
  - hubugs-open command line option, 7
- b, -browse
  - hubugs-show command line option, 6
- c <name>, -create=<name>
  - hubugs-milestones command line option, 8
- f, -full
  - hubugs-show command line option, 6
- h, -help
  - hubugs command line option, 5
- l <label>, -label=<label>
  - hubugs-list command line option, 6
- l, -list
  - hubugs-milestones command line option, 8
- m <text>, -message=<text>
  - hubugs-close command line option, 7
  - hubugs-comment command line option, 7
  - hubugs-reopen command line option, 7
- o <order>, -order=<order>
  - hubugs-list command line option, 6
  - hubugs-milestones command line option, 8
  - hubugs-search command line option, 6
- o, -patch-only
  - hubugs-show command line option, 6
- p <number>, -page <number>
  - hubugs-list command line option, 6
- p <project>, -project=<project>
  - hubugs command line option, 5
- p, -patch
  - hubugs-show command line option, 6
- r <label>, -remove=<label>
  - hubugs-label command line option, 8
- r, -pull-requests
  - hubugs-list command line option, 6
- s <state>, -state=<state>
  - hubugs-list command line option, 6
  - hubugs-milestones command line option, 8
  - hubugs-search command line option, 6
- u <url>, -host-url=<url>
  - hubugs command line option, 5
- \_v2\_conv\_timestamp() (in module *hubugs.models*), 19

## A

- assignee (*Bug attribute*), 23
- avatar\_url, 25

## B

- body (*Bug attribute*), 23
- body (*built-in variable*), 10
- body (*Comment attribute*), 24
- body\_html (*Bug attribute*), 23
- body\_html (*Comment attribute*), 24
- bug (*built-in variable*), 10
- bugs (*built-in variable*), 10

## C

`clone_url` (*Repository attribute*), 24  
`close()` (*in module hubugs*), 18  
`closed_at` (*Bug attribute*), 23  
`closed_by` (*Bug attribute*), 23  
`color` (*Label attribute*), 24  
`colourise()` (*in module hubugs.template*), 20  
`columns` (*built-in variable*), 9  
`comment()` (*in module hubugs*), 18  
`comment_char` (*built-in variable*), 10  
`comments` (*Bug attribute*), 23  
`comments` (*built-in variable*), 10  
`created_at` (*Bug attribute*), 23  
`created_at` (*Comment attribute*), 24  
`created_at` (*Repository attribute*), 24

## D

`description` (*Repository attribute*), 24  
`diff_url` (*PullRequest attribute*), 24  
`display_bugs()` (*in module hubugs.template*), 21

## E

`edit()` (*in module hubugs*), 18  
`edit_text()` (*in module hubugs.template*), 21  
`EmptyMessageError`, 23  
environment variable  
    HUBUGS\_TOKEN, 4  
    XDG\_DATA\_DIRS, 8, 9  
    XDG\_DATA\_HOME, 9

## F

`fork` (*Repository attribute*), 24  
`forks` (*Repository attribute*), 24  
`from_search()` (*in module hubugs.models*), 19  
`full` (*built-in variable*), 10

## G

`get_editor()` (*in module hubugs.utils*), 21  
`get_git_config_val()` (*in module hubugs.utils*), 22  
`get_github_api()` (*in module hubugs.utils*), 22  
`get_repo()` (*in module hubugs.utils*), 22  
`get_template()` (*in module hubugs.template*), 19  
`git_url` (*Repository attribute*), 24  
`gravatar_id`, 25

## H

`has_downloads` (*Repository attribute*), 25  
`has_issues` (*Repository attribute*), 25  
`has_wiki` (*Repository attribute*), 25  
`highlight()` (*in module hubugs.template*), 20  
`homepage` (*Repository attribute*), 25  
`html2text()` (*in module hubugs.template*), 20

`html_url` (*Repository attribute*), 25  
`HttpClientError`, 23  
`hubugs` (*module*), 1, 18  
`hubugs` command line option  
    -no-pager, 5  
    -pager <pager>, 5  
    -version, 5  
    -h, -help, 5  
    -p <project>, -project=<project>, 5  
    -u <url>, -host-url=<url>, 5  
`hubugs-close` command line option  
    -stdin, 7  
    -m <text>, -message=<text>, 7  
`hubugs-comment` command line option  
    -stdin, 7  
    -m <text>, -message=<text>, 7  
`hubugs-edit` command line option  
    -stdin, 7  
`hubugs-label` command line option  
    -a <label>, -add=<label>, 8  
    -r <label>, -remove=<label>, 8  
`hubugs-list` command line option  
    -l <label>, -label=<label>, 6  
    -o <order>, -order=<order>, 6  
    -p <number>, -page <number>, 6  
    -r, -pull-requests, 6  
    -s <state>, -state=<state>, 6  
`hubugs-milestones` command line option  
    -c <name>, -create=<name>, 8  
    -l, -list, 8  
    -o <order>, -order=<order>, 8  
    -s <state>, -state=<state>, 8  
`hubugs-open` command line option  
    -stdin, 7  
    -a label, -add label, 7  
`hubugs-reopen` command line option  
    -stdin, 7  
    -m <text>, -message=<text>, 7  
`hubugs-search` command line option  
    -o <order>, -order=<order>, 6  
    -s <state>, -state=<state>, 6  
`hubugs-setup` command line option  
    -local, 6  
`hubugs-show` command line option  
    -b, -browse, 6  
    -f, -full, 6  
    -o, -patch-only, 6  
    -p, -patch, 6  
`hubugs.models` (*module*), 19  
`hubugs.template` (*module*), 19  
`hubugs.utils` (*module*), 21  
HUBUGS\_TOKEN, 4

**I**

`id_len` (*built-in variable*), 10

**J**

`jinja_filter()` (*in module hubugs.template*), 20

**L**

`label()` (*in module hubugs*), 19

`labels` (*Bug attribute*), 23

`language` (*Repository attribute*), 25

`list_bugs()` (*in module hubugs*), 18

`login`, 25

**M**

`main()` (*in module hubugs*), 19

`master_branch` (*Repository attribute*), 25

`milestone` (*Bug attribute*), 23

`milestone()` (*in module hubugs*), 19

`milestones()` (*in module hubugs*), 19

`mirror_url` (*Repository attribute*), 25

**N**

`name` (*Label attribute*), 24

`name` (*Repository attribute*), 25

`number` (*Bug attribute*), 23

**O**

`object_hook()` (*in module hubugs.models*), 19

`open_bug()` (*in module hubugs*), 18

`open_issues` (*Repository attribute*), 25

`order` (*built-in variable*), 10

`owner` (*Repository attribute*), 25

**P**

`pager()` (*in module hubugs.utils*), 22

`patch` (*built-in variable*), 10

`patch_only` (*built-in variable*), 10

`patch_url` (*PullRequest attribute*), 24

`private` (*Repository attribute*), 25

`project` (*built-in variable*), 10

`pull_request` (*Bug attribute*), 23

`pushed_at` (*Repository attribute*), 25

**R**

`reopen()` (*in module hubugs*), 18

`RepoError`, 23

`report_bug()` (*in module hubugs*), 19

`RFC`

`RFC 2822`, 11

**S**

`search()` (*in module hubugs*), 18

`set_git_config_val()` (*in module hubugs.utils*), 22

`setup()` (*in module hubugs*), 18

`setup_environment()` (*in module hubugs.utils*), 21

`show()` (*in module hubugs*), 18

`size` (*Repository attribute*), 25

`state` (*Bug attribute*), 23

`state` (*built-in variable*), 10

`sync_labels()` (*in module hubugs.utils*), 22

**T**

`term` (*built-in variable*), 10

`title` (*Bug attribute*), 23

`title` (*built-in variable*), 10

**U**

`updated_at` (*Bug attribute*), 23

`updated_at` (*Comment attribute*), 24

`user` (*Bug attribute*), 24

`user` (*Comment attribute*), 24

**W**

`watchers` (*Repository attribute*), 25

**X**

`XDG_DATA_DIRS`, 8, 9

`XDG_DATA_HOME`, 9