

---

# **pybot**

***Release 1.0.1***

**Aug 16, 2023**



---

## Contents:

---

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>bot module</b>                              | <b>1</b>  |
| <b>2</b> | <b>botbrain module</b>                         | <b>3</b>  |
| <b>3</b> | <b>conferror module</b>                        | <b>5</b>  |
| <b>4</b> | <b>confman module</b>                          | <b>7</b>  |
| <b>5</b> | <b>db module</b>                               | <b>9</b>  |
| <b>6</b> | <b>event module</b>                            | <b>11</b> |
| <b>7</b> | <b>lite module</b>                             | <b>13</b> |
| <b>8</b> | <b>logger module</b>                           | <b>15</b> |
| <b>9</b> | <b>modules package</b>                         | <b>17</b> |
| 9.1      | Subpackages                                    | 17        |
| 9.1.1    | modules.snippets package                       | 17        |
| 9.1.1.1  | Submodules                                     | 17        |
| 9.1.1.2  | modules.snippets.commandtest module            | 17        |
| 9.1.1.3  | modules.snippets.d10 module                    | 17        |
| 9.1.1.4  | modules.snippets.d12 module                    | 17        |
| 9.1.1.5  | modules.snippets.d4 module                     | 17        |
| 9.1.1.6  | modules.snippets.d6 module                     | 17        |
| 9.1.1.7  | modules.snippets.d8 module                     | 18        |
| 9.1.1.8  | modules.snippets.downtime module               | 18        |
| 9.1.1.9  | modules.snippets.platypus module               | 18        |
| 9.1.1.10 | modules.snippets.reflect module                | 18        |
| 9.1.1.11 | modules.snippets.second_in_command_test module | 18        |
| 9.1.1.12 | modules.snippets.snippetutil module            | 18        |
| 9.1.1.13 | Module contents                                | 18        |
| 9.2      | Submodules                                     | 18        |
| 9.3      | modules.basemodule module                      | 18        |
| 9.4      | modules.bofh module                            | 19        |
| 9.5      | modules.bonk module                            | 19        |
| 9.6      | modules.choose module                          | 19        |
| 9.7      | modules.ctof module                            | 19        |

|           |                                  |           |
|-----------|----------------------------------|-----------|
| 9.8       | modules.d20 module               | 19        |
| 9.9       | modules.dad module               | 20        |
| 9.10      | modules.dance module             | 20        |
| 9.11      | modules.debugger module          | 20        |
| 9.12      | modules.diabeetus module         | 20        |
| 9.13      | modules.disconnect_yeller module | 21        |
| 9.14      | modules.example module           | 21        |
| 9.15      | modules.examplerderived module   | 21        |
| 9.16      | modules.ftoc module              | 21        |
| 9.17      | modules.hello module             | 22        |
| 9.18      | modules.help module              | 22        |
| 9.19      | modules.howdy module             | 22        |
| 9.20      | modules.isup module              | 22        |
| 9.21      | modules.jimmies module           | 23        |
| 9.22      | modules.jury module              | 23        |
| 9.23      | modules.kanbomodule module       | 23        |
| 9.24      | modules.lastfm module            | 23        |
| 9.25      | modules.meme module              | 23        |
| 9.26      | modules.module module            | 24        |
| 9.27      | modules.nicklist module          | 25        |
| 9.28      | modules.part module              | 25        |
| 9.29      | modules.pimp module              | 25        |
| 9.30      | modules.qdb module               | 25        |
| 9.31      | modules.r6 module                | 26        |
| 9.32      | modules.recap module             | 26        |
| 9.33      | modules.redditinfo module        | 27        |
| 9.34      | modules.replace module           | 27        |
| 9.35      | modules.replay module            | 28        |
| 9.36      | modules.seen module              | 28        |
| 9.37      | modules.shortener module         | 28        |
| 9.38      | modules.tell module              | 28        |
| 9.39      | modules.told module              | 29        |
| 9.40      | modules.twitterposter module     | 29        |
| 9.41      | modules.tzone module             | 30        |
| 9.42      | modules.uptime module            | 30        |
| 9.43      | modules.vyos module              | 30        |
| 9.44      | modules.weather module           | 30        |
| 9.45      | modules.welcome module           | 31        |
| 9.46      | modules.youtube module           | 31        |
| 9.47      | modules.yth module               | 31        |
| 9.48      | Module contents                  | 31        |
| <b>10</b> | <b>pybot module</b>              | <b>33</b> |
| <b>11</b> | <b>stats module</b>              | <b>35</b> |
| <b>12</b> | <b>util module</b>               | <b>37</b> |
| <b>13</b> | <b>version module</b>            | <b>39</b> |
| <b>14</b> | <b>webwriter module</b>          | <b>41</b> |
| <b>15</b> | <b>Indices and tables</b>        | <b>43</b> |
|           | <b>Python Module Index</b>       | <b>45</b> |





# CHAPTER 1

---

bot module

---





## CHAPTER 2

---

### botbrain module

---

```
class botbrain.BotBrain (microphone, bot=None)  
    Bases: object  
  
    BRAINDEBUG = False  
  
    getMicrophone ()  
  
    notice (channel, thing)  
  
    respond (usr, channel, message)  
  
    say (channel, thing)
```



## CHAPTER 3

---

### conferror module

---

**exception** `conferror.ConfError` (*error*)  
Bases: `Exception`



## CHAPTER 4

---

### confman module

---

```
class confman.ConfManager (conf=None)
```

```
    Bases: object
```

Singleton class. Opens and parses a JSON-formatted conf file from (generally) the running user's home folder. Looks for .pybotrc. This allows each thread to know only its own network name, and always get back the information specified for that network from the confman.

```
    getChannels (net)
```

```
    getDBName (net)
```

```
    getDBPass (net)
```

```
    getDBType ()
```

```
    getDBUsername (net)
```

```
    getIRCPass (net)
```

```
    getNetwork ()
```

```
    getNetworks ()
```

```
    getNick (net)
```

```
    getNumChannels (net)
```

```
    getNumNets ()
```

```
    getOwner (net)
```

```
    getPort (net)
```

```
    getTimeout (net)
```



## CHAPTER 5

---

### db module

---

```
class db.DB(bot=None)
    Bases: object

    Handles connecting to the database and reading and writing data. Currently supports only MySQL/mariadb, and
    that probably needs to change.

    age = datetime.datetime(2023, 8, 16, 22, 20, 26, 141330)
    e (sql)
    getImgs ()
    getSeen (who)
    insert (where, which, what)
    insertImg (user, url, channel)
    isAdmin (username)
    replace (where, which, what)
    select (where, what)
    updateSeen (who, statement, event)
```





**class** `event.Event` (*\_type*)Bases: `object`

Allows event type definition. The definition accepts a regex. Every event can be triggered by specific lines, messages, `message_id` or users. Eventually (see `time_event` branch for proof-of-concept implementation) time-sensitive events will be triggerable as well.

Each line received by the bot is passed to each module in the `modules_list`. If the module determines the line matches what the event cares about, the event calls each of its subscribers itself, which contains all the information the module needs to respond appropriately.

**To use:** `e = Event("__my_type__") e.define("some_regex") bot.register_event(e, calling_module)`

**define** (*definition=None, msg\_definition=None, user\_definition=None, message\_id=None, mode=None, case\_insensitive=False*)

Define ourself by general line (*definition*), *msg\_definition* (what someone says in a channel or PM), *user\_definition* (the user who said the thing), or *message\_id* (like 376 for MOTD or 422 for no MOTD) Currently, an event is defined by only one type of definition. If one were to remove the returns after each self. set, an event could be defined and triggered by any of several definitions.

Args: *definition*: string. regex allowed. *msg\_definition*: string. regex allowed. this is what someone would say in a channel. like "hello, pybot". *user\_definition*: string. the user that said the thing. like 'hlmtre' or 'BoneKin'. *message\_id*: the numerical ID of low-level IRC protocol stuff. 376, for example, tells clients 'hey, this is the MOTD.'

**matches** (*line*)

Fills out the event object per line, and returns True or False if the line matches one of our definitions. Args: *line*: string. The entire incoming line.

Return: boolean; True or False.

**notifySubscribers** (*line*)

Fills out the object with all necessary information, then notifies subscribers with itself (an event with all the line information parsed out) as an argument. Args: *line*: string

**subscribe** (*e*)

Append passed-in event to our list of subscribing modules.

Args: e: event.

## CHAPTER 7

---

lite module

---

```
class lite.SQLiteDB (bot=None)
    Bases: object
    e (sql)
    getImgs ()
    insertImg (user, url, channel)
    isAdmin (username)
```



## CHAPTER 8

---

### logger module

---

```
class logger.Logger
```

```
    Bases: object
```

```
    CRITICAL = 0
```

```
    INFO = 2
```

```
    WARNING = 1
```

```
    levels = ['CRITICAL', 'WARNING', 'INFO']
```

```
    write (level, line, nick=None, location=None)
```

Write out to the logfile of either default location or otherwise specified. Includes calling class in its logged line.

Args: level: enumerated thing from the logger class. line: string. thing to write out to the logger. nick: string. determines filename. location: where to write the logfile out to.

Returns: nothing.



## 9.1 Subpackages

### 9.1.1 modules.snippets package

#### 9.1.1.1 Submodules

#### 9.1.1.2 modules.snippets.commandtest module

`modules.snippets.commandtest.test_function` (*bot, message, channel*)

#### 9.1.1.3 modules.snippets.d10 module

`modules.snippets.d10.d10` (*bot, message, channel*)

#### 9.1.1.4 modules.snippets.d12 module

`modules.snippets.d12.d12` (*bot, message, channel*)

#### 9.1.1.5 modules.snippets.d4 module

`modules.snippets.d4.d4` (*bot, message, channel*)

#### 9.1.1.6 modules.snippets.d6 module

`modules.snippets.d6.d4` (*bot, message, channel*)

### 9.1.1.7 modules.snippets.d8 module

`modules.snippets.d8.d4 (bot, message, channel)`

### 9.1.1.8 modules.snippets.downtime module

### 9.1.1.9 modules.snippets.platypus module

`modules.snippets.platypus.platform_info (bot, message, channel)`

### 9.1.1.10 modules.snippets.reflect module

`modules.snippets.reflect.reflect_reload (bot, message, channel)`

### 9.1.1.11 modules.snippets.second\_in\_command\_test module

### 9.1.1.12 modules.snippets.snippetutil module

`modules.snippets.snippetutil.reload (bot, message, channel)`

### 9.1.1.13 Module contents

## 9.2 Submodules

## 9.3 modules.basemodule module

**class** `modules.basemodule.BaseModule` (*events=None, printer\_handle=None, bot=None, say=None*)

Bases: `object`

A base module class for deriving modules (anything you fine folk write, probably) to inherit from. The nice this is this allows you to define your own `post_init` and `handle` functions.

In your module's `post_init`, define and register your own events, and pass your module in.

```
def MyModule(BaseModule):
    def post_init(self):
        e = Event("__wee__")
        e.define("foo")
        self.bot.register_event(e, self)
```

Bam, you've got the things you need (a bot handle, mostly) and by extending `BaseModule` you implement the right things to be called without error. Elzar.

**handle** (*event*)

**post\_init** ()

Called after `init` is set up and builds out our basic module's needs. Allows you to do your own post-processing when inheriting from `BaseModule`.



## 9.4 modules.bofh module

```
class modules.bofh.Bofh (events=None, printer_handle=None, bot=None, say=None)
    Bases: modules.basemodule.BaseModule

    handle (event)

    post_init ()
        Called after init is set up and builds out our basic module's needs. Allows you to do your own post-
        processing when inheriting from BaseModule.
```

## 9.5 modules.bonk module

```
class modules.bonk.Bonk (events=None, printer_handle=None, bot=None, say=None)
    Bases: modules.basemodule.BaseModule

    get_bonked (bonkee="")

    handle (event)

    post_init ()
        Called after init is set up and builds out our basic module's needs. Allows you to do your own post-
        processing when inheriting from BaseModule.
```

## 9.6 modules.choose module

```
class modules.choose.Choose (events=None, printer_handle=None, bot=None, say=None)
    Bases: modules.basemodule.BaseModule

    handle (event)

    post_init ()
        Called after init is set up and builds out our basic module's needs. Allows you to do your own post-
        processing when inheriting from BaseModule.
```

## 9.7 modules.ctof module

```
class modules.ctof.Ctof (events=None, printer_handle=None, bot=None, say=None)
    Bases: modules.basemodule.BaseModule

    handle (event)

    post_init ()
        Called after init is set up and builds out our basic module's needs. Allows you to do your own post-
        processing when inheriting from BaseModule.
```

## 9.8 modules.d20 module

```
class modules.d20.D20 (events=None, printer_handle=None, bot=None, say=None)
    Bases: modules.basemodule.BaseModule

    handle (event)
```

**post\_init()**

Called after init is set up and builds out our basic module's needs. Allows you to do your own post-processing when inheriting from BaseModule.

## 9.9 modules.dad module

**class** `modules.dad.Dad` (*events=None, printer\_handle=None, bot=None, say=None*)

Bases: `modules.basemodule.BaseModule`

**handle** (*event*)

**post\_init()**

Called after init is set up and builds out our basic module's needs. Allows you to do your own post-processing when inheriting from BaseModule.

## 9.10 modules.dance module

**class** `modules.dance.Dance` (*events=None, printer\_handle=None, bot=None, say=None*)

Bases: `modules.basemodule.BaseModule`

**handle** (*event*)

**post\_init()**

Called after init is set up and builds out our basic module's needs. Allows you to do your own post-processing when inheriting from BaseModule.

## 9.11 modules.debugger module

**class** `modules.debugger.Debugger` (*events=None, printer\_handle=None, bot=None, say=None*)

Bases: `modules.basemodule.BaseModule`

**handle** (*event*)

**mem\_store\_delete** (*mem\_store\_key*)

**post\_init()**

Called after init is set up and builds out our basic module's needs. Allows you to do your own post-processing when inheriting from BaseModule.

**pretty** (*d, event, indent=0*)

**recurse** (*obj*)

## 9.12 modules.diabeetus module

**class** `modules.diabeetus.Diabeetus` (*events=None, printer\_handle=None, bot=None, say=None*)

Bases: `modules.basemodule.BaseModule`

**get\_glucose** (*channel*)

get the glucose

**handle** (*event*)

**post\_init()**

Called after init is set up and builds out our basic module's needs. Allows you to do your own post-processing when inheriting from BaseModule.

## 9.13 modules.disconnect\_yeller module

```
class modules.disconnect_yeller.Disconnect_Yeller(events=None,
                                                    printer_handle=None, bot=None,
                                                    say=None)
```

Bases: *modules.basemodule.BaseModule*

**handle**(event)

**post\_init()**

Called after init is set up and builds out our basic module's needs. Allows you to do your own post-processing when inheriting from BaseModule.

## 9.14 modules.example module

```
class modules.example.Example(events=None, printer_handle=None, bot=None, say=None)
```

Bases: object

**handle**(event)

## 9.15 modules.examplerderived module

```
class modules.examplerderived.ExampleDerived(events=None, printer_handle=None,
                                                bot=None, say=None)
```

Bases: *modules.basemodule.BaseModule*

**handle**(event)

**post\_init()**

Called after init is set up and builds out our basic module's needs. Allows you to do your own post-processing when inheriting from BaseModule.

## 9.16 modules.ftoc module

```
class modules.ftoc.Ftoc(events=None, printer_handle=None, bot=None, say=None)
```

Bases: *modules.basemodule.BaseModule*

**handle**(event)

**post\_init()**

Called after init is set up and builds out our basic module's needs. Allows you to do your own post-processing when inheriting from BaseModule.

## 9.17 modules.hello module

```
class modules.hello.Hello(events=None, printer_handle=None, bot=None, say=None)
    Bases: modules.basemodule.BaseModule

    handle (event)

    post_init ()
        Called after init is set up and builds out our basic module's needs. Allows you to do your own post-
        processing when inheriting from BaseModule.
```

## 9.18 modules.help module

```
class modules.help.Help(events=None, printer_handle=None, bot=None, say=None)
    Bases: modules.basemodule.BaseModule

    get_help_lines ()

    handle (event)

    individual_help (cmd, event)

    post_init ()
        Called after init is set up and builds out our basic module's needs. Allows you to do your own post-
        processing when inheriting from BaseModule.
```

## 9.19 modules.howdy module

```
class modules.howdy.Howdy(events=None, printer_handle=None, bot=None, say=None)
    Bases: modules.basemodule.BaseModule

    handle (event)

    post_init ()
        Called after init is set up and builds out our basic module's needs. Allows you to do your own post-
        processing when inheriting from BaseModule.
```

## 9.20 modules.isup module

```
class modules.isup.Isup(events=None, printer_handle=None, bot=None, say=None)
    Bases: modules.basemodule.BaseModule

    takes a url and determines if the site hosted there is up

    handle (event)

    post_init ()
        Called after init is set up and builds out our basic module's needs. Allows you to do your own post-
        processing when inheriting from BaseModule.
```

## 9.21 modules.jimmies module

```
class modules.jimmies.Jimmies (events=None, printer_handle=None, bot=None, say=None)
    Bases: modules.basemodule.BaseModule

    get_jimmies_status ()
        Randomly selects and returns a string with a “jimmies” status.

    handle (event)

    post_init ()
        Called after init is set up and builds out our basic module’s needs. Allows you to do your own post-
        processing when inheriting from BaseModule.
```

## 9.22 modules.jury module

```
class modules.jury.Jury (events=None, printer_handle=None, bot=None, say=None)
    Bases: modules.basemodule.BaseModule

    handle (event)

    post_init ()
        Called after init is set up and builds out our basic module’s needs. Allows you to do your own post-
        processing when inheriting from BaseModule.
```

## 9.23 modules.kanbomodule module

```
class modules.kanbomodule.KanboModule (events=None, printer_handle=None, bot=None, say=None)
    Bases: modules.basemodule.BaseModule

    handle (event)

    post_init ()
        Called after init is set up and builds out our basic module’s needs. Allows you to do your own post-
        processing when inheriting from BaseModule.
```

## 9.24 modules.lastfm module

```
class modules.lastfm.LastFM (events=None, printer_handle=None, bot=None, say=None)
    Bases: modules.basemodule.BaseModule

    handle (event)

    post_init ()
        Called after init is set up and builds out our basic module’s needs. Allows you to do your own post-
        processing when inheriting from BaseModule.
```

## 9.25 modules.meme module

```
class modules.meme.PhonyMc
    Bases: object
```

```
imgflip_password = 'None'
imgflip_userid = 'None'
class modules.meme.meme (events=None, printer_handle=None, bot=None, say=None)
    Bases: object
    check_rate (nick)
        Check to see if the given nick has allowed enough time to pass before calling meme again. Return True
        and set the new last meme time if true. Warn nick and return False if not.
    compare_description (meme_name, user_description)
        compares two strings. if greater than 67% similarity, returns true
    contains_url (line)
        Given a string, returns True if there is a url present
    create_ignore_nicks_tuple ()
        creates a tuple with all nicks from self.ignore_list in <>
    create_meme (meme_id, top_line, bottom_line)
        Given a meme id from imgflip and two lines, top and bottom, submit a request to imgflip for a new meme
        and return the URL
    format_string (line)
        Given an appropriate line, strip out <nick>. Otherwise return unmodified line
    get_last_meme_time (nick)
        Given a channel name, return the last time .meme was called in that channel, return 0 if never used
    get_line (array_of_lines)
        Given an array of lines from which to pick, randomly select an appropriate line, clean it up, and return the
        string.
    get_random_flavor ()
        Change up the flavor text when returning memes. It got boring before
    get_random_meme_id ()
        Selects a random id from the top_memes_list
    get_top_memes ()
        Makes an API call to imgflip to get top 100 most popular memes. Returns a list of results
    get_user_lines (channel, nick)
        Given a specific nick and channel, create a list of all their lines in the buffer
    handle (event)
    is_valid_line (line)
        Given a line from the qdb buffer, return True if certain conditions are met that make it good for meme
        selection. Return False if not
    set_last_meme_time (nick)
        Upon calling meme, set the last time it was used by that nick
```

## 9.26 modules.module module

```
class modules.module.Module (events=None, printer_handle=None, bot=None, say=None)
    Bases: object
    handle (event)
```

```

load(modulename)
unload(modulename)
unload_event(eventname)

```

## 9.27 modules.nicklist module

```

class modules.nicklist.Nicklist(events=None, printer_handle=None, bot=None, say=None)
    Bases: modules.basemodule.BaseModule

    handle(event)

    post_init()
        Called after init is set up and builds out our basic module's needs. Allows you to do your own post-
        processing when inheriting from BaseModule.

```

## 9.28 modules.part module

```

class modules.part.Part(events=None, printer_handle=None, bot=None, say=None)
    Bases: modules.basemodule.BaseModule

    This command should be used as a private message to the bot or else it will not work

    handle(event)

    post_init()
        Called after init is set up and builds out our basic module's needs. Allows you to do your own post-
        processing when inheriting from BaseModule.

```

## 9.29 modules.pimp module

```

class modules.pimp.Pimp(events=None, printer_handle=None, bot=None, say=None)
    Bases: modules.basemodule.BaseModule

    handle(event)

    post_init()
        Called after init is set up and builds out our basic module's needs. Allows you to do your own post-
        processing when inheriting from BaseModule.

```

## 9.30 modules.qdb module

```

class modules.qdb.QDB(events=None, printer_handle=None, bot=None, say=None)
    Bases: object

    add_buffer(event=None, debug=False)
        Takes a channel name and line passed to it and stores them in the bot's mem_store dict for future access.
        The dict will have channel as key. The value to that key will be a list of formatted lines of activity. If the
        buffer size is not yet exceeded, lines are just added. If the buffer is maxed out, the oldest line is removed
        and newest one inserted at the beginning.

```

**add\_recently\_submitted** (*q\_id, submission*)

Takes a string, submission, and adds it to the list of recent submissions. Also we do length checking, only keep record of the previous MAX\_HISTORY\_SIZE quotes.

**delete** (*user, post\_id="", passcode=""*)

A special function that allows certain users to delete posts

**format\_line** (*event*)

Takes an event and formats a string appropriate for quotation from it

**get\_qdb\_submission** (*channel=None, start\_msg="", end\_msg="", strict=False*)

Given two strings, start\_msg and end\_msg, this function will assemble a submission for the QDB. start\_msg is a substring to search for and identify a starting line. end\_msg similarly is used to search for the last desired line in the submission. This function returns a string ready for submission to the QDB if it finds the desired selection. If not, it returns None.

**handle** (*event*)

**recently\_submitted** (*submission*)

Checks to see if the given submission is string is at least 75% similar to the strings in the list of recently submitted quotes. Returns the id of the quote if it was recently submitted. If not, returns -1.

**strip\_formatting** (*msg*)

Uses regex to replace any special formatting in IRC (bold, colors) with nothing

**submit** (*qdb\_submission, debug=False*)

Given a string, qdb\_submission, this function will upload the string to hlmtr's qdb server. Returns a string with status of submission. If it worked, includes a link to new quote.

## 9.31 modules.r6 module

**class** `modules.r6.R6` (*events=None, printer\_handle=None, bot=None, say=None*)

Bases: `modules.basemodule.BaseModule`

Takes specified stats from r6tab and prints them to irc channel

**api\_get** (*name*)

Needed to set user agent so request would not be blocked, without this a 503 status code is returned

**handle** (*event*)

**post\_init** ()

Called after init is set up and builds out our basic module's needs. Allows you to do your own post-processing when inheriting from BaseModule.

**print\_stats** (*ids, js, choice*)

## 9.32 modules.recap module

**class** `modules.recap.recap` (*events=None, printer\_handle=None, bot=None, say=None*)

Bases: `modules.basemodule.BaseModule`

**check\_rate** (*channel*)

Check to see if the given channel has allowed enough time to pass before calling recap again. Return True and set the new time limit if true. Return False if not.

**contains\_url** (*line*)

Given a string, returns True if there is a url present



**create\_ignore\_nicks\_tuple()**  
creates a tuple with all nicks from self.ignore\_list in <>

**dramatize\_line(line)**  
Pass a valid line in, return line with some random type of dramatic formatting

**get\_episode()**  
Return a list with two elements: a random show title and episode name

**get\_lines(channel)**  
Given a channel, searches the qdb buffer for 4 random, suitable lines.

**get\_timediff(channel)**  
Return how much time remains in the function lockdown

**handle(event)**

**post\_init()**  
Called after init is set up and builds out our basic module's needs. Allows you to do your own post-processing when inheriting from BaseModule.

**reset\_timer(channel)**  
If there's an error getting a recap, call this to reset lockdown timer

**scramble\_nick(nick)**  
Given a valid nick in the format <nickname>, scramble a vowel in the nick to avoid beeping the user

**valid\_line(line)**  
Returns True if a given line matches all requirements for validity: Not an action line, longer than minimum length, not spoken by ignored nicks, no URLs

### 9.33 modules.redditinfo module

### 9.34 modules.replace module

**class modules.replace.Replace(events=None, printer\_handle=None, bot=None, say=None)**  
Bases: *modules.basemodule.BaseModule*

**add\_buffer(event=None, debug=False)**  
Takes a channel name and line passed to it and stores them in the bot's mem\_store dict for future access. The dict will have channel as key. The value to that key will be a list of formatted lines of activity. If the buffer size is not yet exceeded, lines are just added. If the buffer is maxed out, the oldest line is removed and newest one inserted at the beginning.

**format\_line(event)**  
Takes an event and formats a string appropriate for quotation from it

**get\_replacement\_message(channel=None, find\_msg="")**  
Looks through the mem\_store to find the most recent message containing find\_msg

**handle(event)**

**post\_init()**  
Called after init is set up and builds out our basic module's needs. Allows you to do your own post-processing when inheriting from BaseModule.

## 9.35 modules.replay module

```
class modules.replay.Replay(events=None, printer_handle=None, bot=None, say=None)
    Bases: object

    get_replacement_message(channel=None, find_msg="")
        Looks through the mem_store to find the most recent message containing find_msg

    handle(event)

    is_number(e)
```

## 9.36 modules.seen module

```
class modules.seen.Seen(events=None, printer_handle=None, bot=None, say=None)
    Bases: modules.basemodule.BaseModule

    handle(event)

    mem_store_init()

    post_init()
        Called after init is set up and builds out our basic module's needs. Allows you to do your own post-
        processing when inheriting from BaseModule.
```

## 9.37 modules.shortener module

```
class modules.shortener.Shortener(events=None, printer_handle=None, bot=None, say=None)
    Bases: modules.basemodule.BaseModule

    handle(event)

    post_init()
        Called after init is set up and builds out our basic module's needs. Allows you to do your own post-
        processing when inheriting from BaseModule.

    reddit_link(link)
```

## 9.38 modules.tell module

```
class modules.tell.Notice(subj, obj, message)
    Bases: object

class modules.tell.Tell(events=None, printer_handle=None, bot=None, say=None)
    Bases: modules.basemodule.BaseModule

    handle(event)

    post_init()
        Because of the way this module works we have to make sure to set our event like we normally would with
        __tell__, but we cannot define our event with "^tell" like we normally would as it will only look for that
        line to trigger the event and the user being told will never receive his message since the bot is only looking
        for .tell and not the user in the PRIVMSG
```

We will set the .tell trigger in our handle function “if event.msg.startswith(“tell”):” and set define to PRIVMSG so it searches all lines from users. While simultaneously looking for the .tell trigger from the user.

This is because we actually need 2 things for this module to work.

1.) The user needs to be able to leave a message for someone else using “.tell someuser <Insert message here>”

2.) **The user who the .tell message is directed towards will be determined by the PRIVMSG definition.**

This is determined in the “else” block that searches every line not starting with .tell. If the user matches the stored user from the previous tell trigger, the event will be triggered and pybot will spit out text into the proper channel every time the intended user says something in chat until the buffer is out of .tell events.

## 9.39 modules.told module

```
class modules.told.Told(events=None, printer_handle=None, bot=None, say=None)
```

Bases: *modules.basemodule.BaseModule*

```
get_told_status(target)
```

Randomly selects and returns a string with a “told” status.

```
handle(event)
```

```
post_init()
```

Called after init is set up and builds out our basic module’s needs. Allows you to do your own post-processing when inheriting from BaseModule.

## 9.40 modules.twitterposter module

```
class modules.twitterposter.TwitterPoster(events=None, printer_handle=None, bot=None,
                                           say=None)
```

Bases: *modules.basemodule.BaseModule*

```
class PhonyPt
```

Bases: object

```
access_token = ''
```

```
access_token_secret = ''
```

```
api_key = ''
```

```
api_secret = ''
```

```
handle(event)
```

```
post_init()
```

Called after init is set up and builds out our basic module’s needs. Allows you to do your own post-processing when inheriting from BaseModule.

```
pt = <modules.twitterposter.TwitterPoster.PhyonPt object>
```

```
user_to_track = 'bhhorg'
```

## 9.41 modules.tzone module

**class** `modules.tzone.Tzone` (*events=None, printer\_handle=None, bot=None, say=None*)

Bases: `modules.basemodule.BaseModule`

**handle** (*event*)

**post\_init** ()

Called after init is set up and builds out our basic module's needs. Allows you to do your own post-processing when inheriting from BaseModule.

**request\_api** (*location*)

Takes the location provided and determines whether its a valid request and will return either the time of the location or a message instructing you how to the make the proper call

## 9.42 modules.uptime module

**class** `modules.uptime.Uptime` (*events=None, printer\_handle=None, bot=None, say=None*)

Bases: `modules.basemodule.BaseModule`

**handle** (*event*)

**post\_init** ()

Called after init is set up and builds out our basic module's needs. Allows you to do your own post-processing when inheriting from BaseModule.

## 9.43 modules.vyos module

Works only in hlmtre's specifically configured environment and when his house has not burned down

**class** `modules.vyos.Vyos` (*events=None, printer\_handle=None, bot=None, say=None*)

Bases: `modules.basemodule.BaseModule`

**handle** (*event*)

**ping** (*nick*)

**post\_init** ()

Called after init is set up and builds out our basic module's needs. Allows you to do your own post-processing when inheriting from BaseModule.

## 9.44 modules.weather module

**class** `modules.weather.Weather` (*events=None, printer\_handle=None, bot=None, say=None*)

Bases: `modules.basemodule.BaseModule`

**get\_api\_request** (*x, y*)

Simple form the query string and return it.

**get\_conditions** (*query, channel*)

given a fully formed query to the OpenWeatherMap API, format an output string

```

get_lat_long_from_bing (location)
    go grab the latitude/longitude from bing's really excellent location API. Returns: tuple of x,y coordinates
    - (0,0) on error

handle (event)

post_init ()
    Called after init is set up and builds out our basic module's needs. Allows you to do your own post-
    processing when inheriting from BaseModule.

```

## 9.45 modules.welcome module

```

class modules.welcome.Welcome (events=None, printer_handle=None, bot=None, say=None)
    Bases: object

    handle (event)

```

## 9.46 modules.youtube module

```

class modules.youtube.Youtube (events=None, printer_handle=None, bot=None, say=None)
    Bases: modules.basemodule.BaseModule

    handle (event)

    post_init ()
        Called after init is set up and builds out our basic module's needs. Allows you to do your own post-
        processing when inheriting from BaseModule.

    print_video_title (event, url, video_tag)

```

## 9.47 modules.yth module

```

class modules.yth.YTH (events=None, printer_handle=None, bot=None, say=None)
    Bases: modules.basemodule.BaseModule

    handle (event)

    post_init ()
        Called after init is set up and builds out our basic module's needs. Allows you to do your own post-
        processing when inheriting from BaseModule.

```

## 9.48 Module contents



## CHAPTER 10

---

pybot module

---





## CHAPTER 11

---

stats module

---

```
class stats.Stats
    Bases: object
    db = <db.DB object>
```



## CHAPTER 12

---

### util module

---

```
class util.bcolors
    Bases: object

    Allows for prettyprinting to the console for debugging.

    CYAN = '\x1b[36m'
    ENDC = '\x1b[0m'
    FAIL = '\x1b[91m'
    GREEN = '\x1b[32m'
    HEADER = '\x1b[95m'
    OKBLUE = '\x1b[94m'
    OKGREEN = '\x1b[92m'
    WARNING = '\x1b[93m'
    YELLOW = '\x1b[33m'

util.commands(*command_list)
util.depends(self, module_name)
util.parse_line(line)
    returns an object with a nice set of line-pulled-apart members
util.strip_nick(nick)
    Clean up nicks of their op levels (&Schooly_D, ~BoneKin, etc)
```



## CHAPTER 13

---

version module

---



## CHAPTER 14

---

### webwriter module

---

**class** webwriter.**WebWriter**

Bases: object

Handles writing out to teh home folder of the running user. Creates a very simple web page that just lists all the images posted in a channel the bot's in. Probably in trouble for cp because of teh 4cdn links.





## CHAPTER 15

---

### Indices and tables

---

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



### b

botbrain, 3

### c

conferror, 5

confman, 7

### d

db, 9

### e

event, 11

### l

lite, 13

logger, 15

### m

modules, 31

modules.basemodule, 18

modules.bofh, 19

modules.bonk, 19

modules.choose, 19

modules.ctof, 19

modules.d20, 19

modules.dad, 20

modules.dance, 20

modules.debugger, 20

modules.diabeetus, 20

modules.disconnect\_yeller, 21

modules.example, 21

modules.examplerderived, 21

modules.ftoc, 21

modules.hello, 22

modules.help, 22

modules.howdy, 22

modules.isup, 22

modules.jimmies, 23

modules.jury, 23

modules.kanbomodule, 23

modules.lastfm, 23

modules.meme, 23

modules.module, 24

modules.nicklist, 25

modules.part, 25

modules.pimp, 25

modules.qdb, 25

modules.r6, 26

modules.recap, 26

modules.replace, 27

modules.replay, 28

modules.seen, 28

modules.shortener, 28

modules.snippets, 18

modules.snippets.commandtest, 17

modules.snippets.d10, 17

modules.snippets.d12, 17

modules.snippets.d4, 17

modules.snippets.d6, 17

modules.snippets.d8, 18

modules.snippets.downtime, 18

modules.snippets.platypus, 18

modules.snippets.reflect, 18

modules.snippets.snippetutil, 18

modules.tell, 28

modules.told, 29

modules.twitterposter, 29

modules.tzone, 30

modules.uptime, 30

modules.vyos, 30

modules.weather, 30

modules.welcome, 31

modules.youtube, 31

modules.yth, 31

### p

pybot, 33

## **S**

stats, [35](#)

## **U**

util, [37](#)

## **V**

version, [39](#)

## **W**

webwriter, [41](#)

## A

access\_token (modules.twitterposter.TwitterPoster.PhyPy attribute), 29

access\_token\_secret (modules.twitterposter.TwitterPoster.PhyPy attribute), 29

add\_buffer() (modules.qdb.QDB method), 25

add\_buffer() (modules.replace.Replace method), 27

add\_recently\_submitted() (modules.qdb.QDB method), 25

age (db.DB attribute), 9

api\_get() (modules.r6.R6 method), 26

api\_key (modules.twitterposter.TwitterPoster.PhyPy attribute), 29

api\_secret (modules.twitterposter.TwitterPoster.PhyPy attribute), 29

## B

BaseModule (class in modules.basemodule), 18

bcolors (class in util), 37

Bofh (class in modules.bofh), 19

Bonk (class in modules.bonk), 19

BotBrain (class in botbrain), 3

botbrain (module), 3

BRAINDEBUG (botbrain.BotBrain attribute), 3

## C

check\_rate() (modules.meme.meme method), 24

check\_rate() (modules.recap.recap method), 26

Choose (class in modules.choose), 19

commands() (in module util), 37

compare\_description() (modules.meme.meme method), 24

ConfError, 5

conferror (module), 5

confman (module), 7

ConfManager (class in confman), 7

contains\_url() (modules.meme.meme method), 24

contains\_url() (modules.recap.recap method), 26

create\_ignore\_nicks\_tuple() (modules.meme.meme method), 24

create\_ignore\_nicks\_tuple() (modules.recap.recap method), 27

create\_meme() (modules.meme.meme method), 24

CRITICAL (logger.Logger attribute), 15

Ctof (class in modules.ctof), 19

CYAN (util.bcolors attribute), 37

## D

d10() (in module modules.snippets.d10), 17

d12() (in module modules.snippets.d12), 17

D20 (class in modules.d20), 19

d4() (in module modules.snippets.d4), 17

d4() (in module modules.snippets.d6), 17

d4() (in module modules.snippets.d8), 18

Dad (class in modules.dad), 20

Dance (class in modules.dance), 20

DB (class in db), 9

db (module), 9

db (stats.Stats attribute), 35

Debugger (class in modules.debugger), 20

define() (event.Event method), 11

delete() (modules.qdb.QDB method), 26

depends() (in module util), 37

Diabeetus (class in modules.diabeetus), 20

Disconnect\_Yeller (class in modules.disconnect\_yeller), 21

dramatize\_line() (modules.recap.recap method), 27

## E

e() (db.DB method), 9

e() (lite.SqliteDB method), 13

ENDC (util.bcolors attribute), 37

Event (class in event), 11

event (module), 11

Example (class in modules.example), 21

ExampleDerived (class in *modules.examplederived*), 21

## F

FAIL (*util.bcolors* attribute), 37

format\_line() (*modules.qdb.QDB* method), 26

format\_line() (*modules.replace.Replace* method), 27

format\_string() (*modules.meme.meme* method), 24

Ftoc (class in *modules.ftoc*), 21

## G

get\_api\_request() (*modules.weather.Weather* method), 30

get\_bonked() (*modules.bonk.Bonk* method), 19

get\_conditions() (*modules.weather.Weather* method), 30

get\_episode() (*modules.recap.recap* method), 27

get\_glucose() (*modules.diabeetus.Diabeetus* method), 20

get\_help\_lines() (*modules.help.Help* method), 22

get\_jimmies\_status() (*modules.jimmies.Jimmies* method), 23

get\_last\_meme\_time() (*modules.meme.meme* method), 24

get\_lat\_long\_from\_bing() (*modules.weather.Weather* method), 30

get\_line() (*modules.meme.meme* method), 24

get\_lines() (*modules.recap.recap* method), 27

get\_qdb\_submission() (*modules.qdb.QDB* method), 26

get\_random\_flavor() (*modules.meme.meme* method), 24

get\_random\_meme\_id() (*modules.meme.meme* method), 24

get\_replacement\_message() (*modules.replace.Replace* method), 27

get\_replacement\_message() (*modules.replay.Replay* method), 28

get\_timediff() (*modules.recap.recap* method), 27

get\_told\_status() (*modules.told.Told* method), 29

get\_top\_memes() (*modules.meme.meme* method), 24

get\_user\_lines() (*modules.meme.meme* method), 24

getChannels() (*confman.ConfManager* method), 7

getDBName() (*confman.ConfManager* method), 7

getDBPass() (*confman.ConfManager* method), 7

getDBType() (*confman.ConfManager* method), 7

getDBUsername() (*confman.ConfManager* method), 7

getImgs() (*db.DB* method), 9

getImgs() (*lite.SQLiteDB* method), 13

getIRCPass() (*confman.ConfManager* method), 7

getMicrophone() (*botbrain.BotBrain* method), 3

getNetwork() (*confman.ConfManager* method), 7

getNetworks() (*confman.ConfManager* method), 7

getNick() (*confman.ConfManager* method), 7

getNumChannels() (*confman.ConfManager* method), 7

getNumNets() (*confman.ConfManager* method), 7

getOwner() (*confman.ConfManager* method), 7

getPort() (*confman.ConfManager* method), 7

getSeen() (*db.DB* method), 9

getTimeout() (*confman.ConfManager* method), 7

GREEN (*util.bcolors* attribute), 37

## H

handle() (*modules.basemodule.BaseModule* method), 18

handle() (*modules.bofh.Bofh* method), 19

handle() (*modules.bonk.Bonk* method), 19

handle() (*modules.choose.Choose* method), 19

handle() (*modules.ctof.Ctof* method), 19

handle() (*modules.d20.D20* method), 19

handle() (*modules.dad.Dad* method), 20

handle() (*modules.dance.Dance* method), 20

handle() (*modules.debugger.Debugger* method), 20

handle() (*modules.diabeetus.Diabeetus* method), 20

handle() (*modules.disconnect\_yeller.Disconnect\_Yeller* method), 21

handle() (*modules.example.Example* method), 21

handle() (*modules.examplederived.ExampleDerived* method), 21

handle() (*modules.ftoc.Ftoc* method), 21

handle() (*modules.hello.Hello* method), 22

handle() (*modules.help.Help* method), 22

handle() (*modules.howdy.Howdy* method), 22

handle() (*modules.isup.Isup* method), 22

handle() (*modules.jimmies.Jimmies* method), 23

handle() (*modules.jury.Jury* method), 23

handle() (*modules.kanbomodule.KanboModule* method), 23

handle() (*modules.lastfm.LastFM* method), 23

handle() (*modules.meme.meme* method), 24

handle() (*modules.module.Module* method), 24

handle() (*modules.nicklist.Nicklist* method), 25

handle() (*modules.part.Part* method), 25

handle() (*modules.pimp.Pimp* method), 25

handle() (*modules.qdb.QDB* method), 26

handle() (*modules.r6.R6* method), 26

handle() (*modules.recap.recap* method), 27

handle() (*modules.replace.Replace* method), 27

handle() (*modules.replay.Replay* method), 28

handle() (*modules.seen.Seen* method), 28

handle() (*modules.shortener.Shortener* method), 28

handle() (*modules.tell.Tell* method), 28

handle() (*modules.told.Told* method), 29

[handle\(\)](#) (*modules.twitterposter.TwitterPoster method*), 29  
[handle\(\)](#) (*modules.tzone.Tzone method*), 30  
[handle\(\)](#) (*modules.uptime.Uptime method*), 30  
[handle\(\)](#) (*modules.vyos.Vyos method*), 30  
[handle\(\)](#) (*modules.weather.Weather method*), 31  
[handle\(\)](#) (*modules.welcome.Welcome method*), 31  
[handle\(\)](#) (*modules.youtube.Youtube method*), 31  
[handle\(\)](#) (*modules.yth.YTH method*), 31  
[HEADER](#) (*util.bcolors attribute*), 37  
[Hello](#) (*class in modules.hello*), 22  
[Help](#) (*class in modules.help*), 22  
[Howdy](#) (*class in modules.howdy*), 22

**I**

[imgflip\\_password](#) (*modules.meme.PhonyMc attribute*), 23  
[imgflip\\_userid](#) (*modules.meme.PhonyMc attribute*), 24  
[individual\\_help\(\)](#) (*modules.help.Help method*), 22  
[INFO](#) (*logger.Logger attribute*), 15  
[insert\(\)](#) (*db.DB method*), 9  
[insertImg\(\)](#) (*db.DB method*), 9  
[insertImg\(\)](#) (*lite.SqliteDB method*), 13  
[is\\_number\(\)](#) (*modules.replay.Replay method*), 28  
[is\\_valid\\_line\(\)](#) (*modules.meme.meme method*), 24  
[isAdmin\(\)](#) (*db.DB method*), 9  
[isAdmin\(\)](#) (*lite.SqliteDB method*), 13  
[Isup](#) (*class in modules.isup*), 22

**J**

[Jimmies](#) (*class in modules.jimmies*), 23  
[Jury](#) (*class in modules.jury*), 23

**K**

[KanboModule](#) (*class in modules.kanbomodule*), 23

**L**

[LastFM](#) (*class in modules.lastfm*), 23  
[levels](#) (*logger.Logger attribute*), 15  
[lite](#) (*module*), 13  
[load\(\)](#) (*modules.module.Module method*), 24  
[Logger](#) (*class in logger*), 15  
[logger](#) (*module*), 15

**M**

[matches\(\)](#) (*event.Event method*), 11  
[mem\\_store\\_delete\(\)](#) (*modules.debugger.Debugger method*), 20  
[mem\\_store\\_init\(\)](#) (*modules.seen.Seen method*), 28  
[meme](#) (*class in modules.meme*), 24  
[Module](#) (*class in modules.module*), 24  
[modules](#) (*module*), 31  
[modules.basemodule](#) (*module*), 18  
[modules.bofh](#) (*module*), 19  
[modules.bonk](#) (*module*), 19  
[modules.choose](#) (*module*), 19  
[modules.ctof](#) (*module*), 19  
[modules.d20](#) (*module*), 19  
[modules.dad](#) (*module*), 20  
[modules.dance](#) (*module*), 20  
[modules.debugger](#) (*module*), 20  
[modules.diabeetus](#) (*module*), 20  
[modules.disconnect\\_yeller](#) (*module*), 21  
[modules.example](#) (*module*), 21  
[modules.examplerderived](#) (*module*), 21  
[modules.ftoc](#) (*module*), 21  
[modules.hello](#) (*module*), 22  
[modules.help](#) (*module*), 22  
[modules.howdy](#) (*module*), 22  
[modules.isup](#) (*module*), 22  
[modules.jimmies](#) (*module*), 23  
[modules.jury](#) (*module*), 23  
[modules.kanbomodule](#) (*module*), 23  
[modules.lastfm](#) (*module*), 23  
[modules.meme](#) (*module*), 23  
[modules.module](#) (*module*), 24  
[modules.nicklist](#) (*module*), 25  
[modules.part](#) (*module*), 25  
[modules.pimp](#) (*module*), 25  
[modules.qdb](#) (*module*), 25  
[modules.r6](#) (*module*), 26  
[modules.recap](#) (*module*), 26  
[modules.replace](#) (*module*), 27  
[modules.replay](#) (*module*), 28  
[modules.seen](#) (*module*), 28  
[modules.shortener](#) (*module*), 28  
[modules.snippets](#) (*module*), 18  
[modules.snippets.commandtest](#) (*module*), 17  
[modules.snippets.d10](#) (*module*), 17  
[modules.snippets.d12](#) (*module*), 17  
[modules.snippets.d4](#) (*module*), 17  
[modules.snippets.d6](#) (*module*), 17  
[modules.snippets.d8](#) (*module*), 18  
[modules.snippets.downtime](#) (*module*), 18  
[modules.snippets.platypus](#) (*module*), 18  
[modules.snippets.reflect](#) (*module*), 18  
[modules.snippets.snippetutil](#) (*module*), 18  
[modules.tell](#) (*module*), 28  
[modules.told](#) (*module*), 29  
[modules.twitterposter](#) (*module*), 29  
[modules.tzone](#) (*module*), 30  
[modules.uptime](#) (*module*), 30  
[modules.vyos](#) (*module*), 30  
[modules.weather](#) (*module*), 30  
[modules.welcome](#) (*module*), 31

`modules.youtube (module)`, 31

`modules.yth (module)`, 31

## N

`Nicklist (class in modules.nicklist)`, 25

`Notice (class in modules.tell)`, 28

`notice () (botbrain.BotBrain method)`, 3

`notifySubscribers () (event.Event method)`, 11

## O

`OKBLUE (util.bcolors attribute)`, 37

`OKGREEN (util.bcolors attribute)`, 37

## P

`parse_line () (in module util)`, 37

`Part (class in modules.part)`, 25

`PhonyMc (class in modules.meme)`, 23

`Pimp (class in modules.pimp)`, 25

`ping () (modules.vyos.Vyos method)`, 30

`platform_info () (in module modules.snippets.platypus)`, 18

`post_init () (modules.basemodule.BaseModule method)`, 18

`post_init () (modules.bofh.Bofh method)`, 19

`post_init () (modules.bonk.Bonk method)`, 19

`post_init () (modules.choose.Choose method)`, 19

`post_init () (modules.ctof.Ctof method)`, 19

`post_init () (modules.d20.D20 method)`, 20

`post_init () (modules.dad.Dad method)`, 20

`post_init () (modules.dance.Dance method)`, 20

`post_init () (modules.debugger.Debugger method)`, 20

`post_init () (modules.diabeetus.Diabeetus method)`, 20

`post_init () (modules.disconnect_yeller.Disconnect_Yeller method)`, 21

`post_init () (modules.examplederived.ExampleDerived method)`, 21

`post_init () (modules.ftoc.Ftoc method)`, 21

`post_init () (modules.hello.Hello method)`, 22

`post_init () (modules.help.Help method)`, 22

`post_init () (modules.howdy.Howdy method)`, 22

`post_init () (modules.isup.Isup method)`, 22

`post_init () (modules.jimmies.Jimmies method)`, 23

`post_init () (modules.jury.Jury method)`, 23

`post_init () (modules.kanbomodule.KanboModule method)`, 23

`post_init () (modules.lastfm.LastFM method)`, 23

`post_init () (modules.nicklist.Nicklist method)`, 25

`post_init () (modules.part.Part method)`, 25

`post_init () (modules.pimp.Pimp method)`, 25

`post_init () (modules.r6.R6 method)`, 26

`post_init () (modules.recap.recap method)`, 27

`post_init () (modules.replace.Replace method)`, 27

`post_init () (modules.seen.Seen method)`, 28

`post_init () (modules.shortener.Shortener method)`, 28

`post_init () (modules.tell.Tell method)`, 28

`post_init () (modules.told.Told method)`, 29

`post_init () (modules.twitterposter.TwitterPoster method)`, 29

`post_init () (modules.tzone.Tzone method)`, 30

`post_init () (modules.uptime.Uptime method)`, 30

`post_init () (modules.vyos.Vyos method)`, 30

`post_init () (modules.weather.Weather method)`, 31

`post_init () (modules.youtube.Youtube method)`, 31

`post_init () (modules.yth.YTH method)`, 31

`pretty () (modules.debugger.Debugger method)`, 20

`print_stats () (modules.r6.R6 method)`, 26

`print_video_title () (modules.youtube.Youtube method)`, 31

`pt (modules.twitterposter.TwitterPoster attribute)`, 29

`pybot (module)`, 33

## Q

`QDB (class in modules.qdb)`, 25

## R

`R6 (class in modules.r6)`, 26

`recap (class in modules.recap)`, 26

`recently_submitted () (modules.qdb.QDB method)`, 26

`recurse () (modules.debugger.Debugger method)`, 20

`reddit_link () (modules.shortener.Shortener method)`, 28

`reflect_reload () (in module modules.snippets.reflect)`, 18

`reload () (in module modules.snippets.snippetutil)`, 18

`Replace (class in modules.replace)`, 27

`replace () (db.DB method)`, 9

`Replay (class in modules.replay)`, 28

`request_api () (modules.tzone.Tzone method)`, 30

`reset_timer () (modules.recap.recap method)`, 27

`respond () (botbrain.BotBrain method)`, 3

## S

`say () (botbrain.BotBrain method)`, 3

`scramble_nick () (modules.recap.recap method)`, 27

`Seen (class in modules.seen)`, 28

`select () (db.DB method)`, 9

`set_last_meme_time () (modules.meme.meme method)`, 24

`Shortener (class in modules.shortener)`, 28

`SqliteDB (class in lite)`, 13

`Stats (class in stats)`, 35

`stats (module)`, 35

`strip_formatting () (modules.qdb.QDB method)`, 26



`strip_nick()` (*in module util*), 37  
`submit()` (*modules.qdb.QDB method*), 26  
`subscribe()` (*event.Event method*), 11

## T

`Tell` (*class in modules.tell*), 28  
`test_function()` (*in module modules.snippets.commandtest*), 17  
`Told` (*class in modules.told*), 29  
`TwitterPoster` (*class in modules.twitterposter*), 29  
`TwitterPoster.PhyonPt` (*class in modules.twitterposter*), 29  
`Tzone` (*class in modules.tzone*), 30

## U

`unload()` (*modules.module.Module method*), 25  
`unload_event()` (*modules.module.Module method*), 25  
`updateSeen()` (*db.DB method*), 9  
`Uptime` (*class in modules.uptime*), 30  
`user_to_track` (*modules.twitterposter.TwitterPoster attribute*), 29  
`util` (*module*), 37

## V

`valid_line()` (*modules.recap.recap method*), 27  
`version` (*module*), 39  
`Vyos` (*class in modules.vyos*), 30

## W

`WARNING` (*logger.Logger attribute*), 15  
`WARNING` (*util.bcolors attribute*), 37  
`Weather` (*class in modules.weather*), 30  
`WebWriter` (*class in webwriter*), 41  
`webwriter` (*module*), 41  
`Welcome` (*class in modules.welcome*), 31  
`write()` (*logger.Logger method*), 15

## Y

`YELLOW` (*util.bcolors attribute*), 37  
`Youtube` (*class in modules.youtube*), 31  
`YTH` (*class in modules.yth*), 31