Python Reddit API Wrapper Documentation

Release 3.4.0

Bryce Boe

March 23, 2016

CONTENTS

PRAW, an acronym for "Python Reddit API Wrapper", is a python package that allows for simple access to reddit's API. PRAW aims to be as easy to use as possible and is designed to follow all of reddit's API rules. You have to give a useragent that follows the rules, everything else is handled by PRAW so you needn't worry about violating them.

Here's a quick peek, getting the first 5 submissions from the 'hot' section of the 'opensource' subreddit:

```
>>> import praw
>>> r = praw.Reddit(user_agent='my_cool_application')
>>> submissions = r.get_subreddit('opensource').get_hot(limit=5)
>>> [str(x) for x in submissions]
```

This will display something similar to the following:

```
['10 :: Gun.io Debuts Group Funding for Open Source Projects\n Gun.io',
 '24 :: Support the Free Software Foundation',
 '67 :: The 10 Most Important Open Source Projects of 2011',
 '85 :: Plan 9 - A distributed OS with a unified communicatioprotocol  I/O...',
  '2 :: Open-source webOS is dead on arrival ']
```

CHAPTER

ONE

CONTENT PAGES

1.1 Getting Started

In this tutorial we'll go over everything needed to create a bot or program using reddit's API through the Python Reddit API Wrapper (PRAW). We're going to write a program that breaks down a redditor's karma by subreddit, just like the reddit feature. Unlike that, our program can break it down for any redditor, not just us. However, it will be less precise due to limitations in the reddit API, but we're getting ahead of ourselves.

This is a beginners tutorial to PRAW. We'll go over the hows and whys of everything from getting started to writing your first program accessing reddit. What we won't go over in this tutorial is the Python code.

1.1.1 Connecting to reddit

Start by firing up Python and importing PRAW. You can find the installation instructions on the *main page*.

```
>>> import praw
```

Next we need to connect to reddit and identify our script. We do this through the `user_agent` we supply when our script first connects to reddit.

```
>>> user_agent = "Karma breakdown 1.0 by /u/_Daimon_"
>>> r = praw.Reddit(user_agent=user_agent)
```

Care should be taken when we decide on what user_agent to send to reddit. The `user_agent` field is how we uniquely identify our script. The reddit API wiki page has the official and updated recommendations on user_agent strings and everything else. Reading it is *highly recommended*.

In addition to reddit's recommendations, your user_agent string should not contain the keyword `bot`.

1.1.2 Breaking Down Redditor Karma by Subreddit

Now that we've established contact with reddit, it's time for the next step in our script: to break down a user's karma by subreddit. There isn't a function that does this, but luckily it's fairly easy to write the python code to do this ourselves.

We use the function *`get_redditor()`* to get a *`Redditor`* instance that represents a user on reddit. In the following case `user` will provide access to the reddit user "'_Daimon_'".

```
>>> user_name = "_Daimon_"
>>> user = r.get_redditor(user_name)
```

Next we can use the functions *`get_comments()`* and *`get_submitted()`* to get that redditor's comments and submissions. Both are a part of the superclass `Thing` as mentioned on the reddit API wiki page. Both functions can be called with the parameter `limit`, which limits how many things we receive. As a default, reddit returns 25 items.

When the limit is set to `None`, PRAW will try to retrieve all the things. However, due to limitations in the reddit API (not PRAW) we might not get all the things, but more about that later. During development you should be nice and set the limit lower to reduce reddit's workload, if you don't actually need all the results.

```
>>> thing_limit = 10
>>> gen = user.get_submitted(limit=thing_limit)
```

Next we take the generator containing things (either comments or submissions) and iterate through them to create a dictionary with the subreddit display names (like *python* or *askreddit*) as keys and the karma obtained in those subreddits as values.

```
>>> karma_by_subreddit = {}
>>> for thing in gen:
...     subreddit = thing.subreddit.display_name
...     karma_by_subreddit[subreddit] = (karma_by_subreddit.get(subreddit, 0)
...                                      + thing.score)
```

Finally, let's output the karma breakdown in a pretty format.

```
>>> import pprint
>>> pprint.pprint(karma_by_subreddit)
```

And we're done. The program could use a better way of displaying the data, exception catching, etc. If you're interested, you can check out a more fleshed out version of this Karma-Breakdown program.

1.1.3 Obfuscation and API limitations

As I mentioned before there are limits in reddit's API. There is a limit to the amount of things reddit will return before it barfs. Any single reddit listing will display at most 1000 items. This is true for all listings including subreddit submission listings, user submission listings, and user comment listings.

You may also have realized that the karma values change from run to run. This inconsistency is due to reddit's obfuscation of the upvotes and downvotes. The obfuscation is done to everything and everybody to thwart potential cheaters. There's nothing we can do to prevent this.

Another thing you may have noticed is that retrieving a lot of elements take time. reddit allows requests of up to 100 items at once. So if you request <= 100 items PRAW can serve your request in a single API call, but for larger requests PRAW will break it into multiple API calls of 100 items each separated by a small 2 second delay to follow the api guidelines. So requesting 250 items will require 3 api calls and take at least 2x2=4 seconds due to API delay. PRAW does the API calls lazily, i.e. it will not send the next api call until you actually need the data. Meaning the runtime is max(api_delay, code execution time).

Continue to the next tutorial. *Writing a reddit Bot*.

1.1.4 The full Karma Breakdown program.

```
import praw

user_agent = ("Karma breakdown 1.0 by /u/_Daimon_ "
              "github.com/Damgaard/Reddit-Bots/")
r = praw.Reddit(user_agent=user_agent)
thing_limit = 10
user_name = "_Daimon_"
user = r.get_redditor(user_name)
gen = user.get_submitted(limit=thing_limit)
karma_by_subreddit = {}
for thing in gen:
```

```
    subreddit = thing.subreddit.display_name
    karma_by_subreddit[subreddit] = (karma_by_subreddit.get(subreddit, 0)
                                     + thing.score)
import pprint
pprint.pprint(karma_by_subreddit)
```

1.2 Writing a reddit Bot

In the *Getting Started* tutorial, we wrote a script to break down a redditor's karma. In this tutorial we will write a bot.

Bots differ from scripts in a few different ways. First, bots normally run continuously whereas scripts are most often one-off jobs. Second, bots usually automate some task that could be performed by a user, such as posting, commenting or moderating. Bots also present unique design challenges not applicable to writing scripts. We need to make sure that bots keep working continuously, don't unnecessarily perform the same task twice and keep within API Guidelines. This tutorial will introduce you to all three of these problems and show how to use PRAW's and reddit's documentation.

1.2.1 The Problem

From time to time questions are submitted to reddit.com about PRAW, mellort's deprecated fork and the reddit API in general. *u_Daimon_* wants to be notified of these submissions, so he can help the submitter. The bot will monitor the subreddits r/python, r/learnpython and r/redditdev and send *u_Daimon_* a private message, whenever it detects a post with such a question.

We start by importing PRAW and logging in.

```
>>> import time
>>> import praw
>>> r = praw.Reddit('PRAW related-question monitor by /u/_Daimon_ v 1.0. '
...                 'Url: https://praw.readthedocs.org/en/latest/'
...                 'pages/writing_a_bot.html')
>>> r.login()
>>> already_done = [] # Ignore this for now
```

The next step is the main loop, where we look at each of the subreddits in turn. For this tutorial we will implement a subset of the bot, which only looks at the submissions in r/learnpython to make the example code as clear as possible.

```
>>> while True:
>>> subreddit = r.get_subreddit('learnpython')
>>> for submission in subreddit.get_hot(limit=10):
        # Test if it contains a PRAW-related question
```

Finding what we need

Now that we have the submissions, we need to see if they contain a PRAW-related question. We are going to look at the text part of a submission to see if it contains one of the strings "reddit api", "praw" or "mellort". So we're going to go through how you can find out stuff like this on your own.

Start the Python interpreter and compare the output with this r/learnpython post.

```
>>> import praw
>>> from pprint import pprint
>>> r = praw.Reddit('Submission variables testing by /u/_Daimon_')
>>> submission = r.get_submission(submission_id = "105aru")
>>> pprint(vars(submission))
```

```
{'_comment_sort': None,
'_comments': [<praw.objects.Comment object at 0x030FF330>,
              <praw.objects.Comment object at 0x03107B70>,
              <praw.objects.Comment object at 0x03107CF0>],

'_comments_by_id': {u't1_c6aijmu': <praw.objects.Comment object at 0x030FF330>,
                    u't1_c6ailrj': <praw.objects.Comment object at 0x03107B70>,
                    u't1_c6ailxt': <praw.objects.Comment object at 0x03107CF0>,
                    u't1_c6ak4rq': <praw.objects.Comment object at 0x03107C50>,
                    u't1_c6akq4n': <praw.objects.Comment object at 0x03107BB0>,
                    u't1_c6akv1g': <praw.objects.Comment object at 0x031077D0>}
                    ,

'_info_url': 'http://www.reddit.com/api/info/',
'_orphaned': {},
'_populated': True,
'_replaced_more': False,
'_underscore_names': None,
'approved_by': None,
'author': Redditor(user_name='Blackshirt12'),
'author_flair_css_class': u'py32bg',
'author_flair_text': u'',
'banned_by': None,
'clicked': False,
'created': 1348081369.0,
'created_utc': 1348077769.0,
'domain': u'self.learnpython',
'downs': 0,
'edited': 1348083784.0,
'hidden': False,
'id': u'105aru',
'is_self': True,
'likes': None,
'link_flair_css_class': None,
'link_flair_text': None,
'media': None,
'media_embed': {},
'name': u't3_105aru',
'num_comments': 6,
'num_reports': None,
'over_18': False,
'permalink': u'http://www.reddit.com/r/learnpython/comments/105aru/newbie_stri
pping_strings_of_last_character/',
'reddit_session': <praw.Reddit object at 0x029477F0>,
'saved': False,
'score': 1,
'selftext': u'Update: Thanks for the help. Got fixed.\n\nI need to strip 3
strin gs in a list of 4 of their trailing commas to get my formatting right
and to conv ert one of them (a number) to a float but I\'m confused on the
syntax. Also, I do n\'t know of an efficient way of completing the task; I was
planning on stripping each of the three strings on a new line.\n\n    for line
in gradefile:\n linelist = string.split(line)\n        #strip linelist[0],[1],
and [2] of commas\ n        linelist = string.rstrip(linelist[0], ",")',
'selftext_html': u'&lt;!-- SC_OFF --&gt;&lt;div class="md"&gt;&lt;p&gt;Update:
Thanks for the help. Got fixed.&lt;/p&gt;\n\n&lt;p&gt;I need to strip 3
strings in a list of 4 of their trailing commas to get my formatting right and
to convert o ne of them (a number) to a float but I&amp;amp;#39;m confused on the
syntax. Also, I don&amp;amp;#39;t know of an efficient way of completing the task;
```

```
I was planning on stripping each of the three strings on a new
line.&lt;/p&gt;\n\n&lt;pre&gt;&lt;co de&gt;for line in gradefile:\n
linelist = string.split(line)\n    #strip linel ist[0],[1], and [2] of
commas\n    linelist = string.rstrip(linelist[0], &amp;quo
t;,&amp;quot;)\n&lt;/code&gt;&lt;/pre&gt;\n&lt;/div&gt;&lt;!-- SC_ON --&gt;',
'subreddit': <praw.objects.Subreddit object at 0x030FF030>,
'subreddit_id': u't5_2r8ot',
'thumbnail': u'',
'title': u'Newbie: stripping strings of last character',
'ups': 1,
'url': u'http://www.reddit.com/r/learnpython/comments/105aru/newbie_stripping_
strings_of_last_character/'}
>>> pprint(dir(submission))
['__class__',
'__delattr__',
'__dict__',
'__doc__',
'__eq__',
'__format__',
'__getattr__',
'__getattribute__',
'__hash__',
'__init__',
'__module__',
'__ne__',
'__new__',
'__reduce__',
'__reduce_ex__',
'__repr__',
'__setattr__',
'__sizeof__',
'__str__',
'__subclasshook__',
'__unicode__',
'__weakref__',
'_comment_sort',
'_comments',
'_comments_by_id',
'_extract_more_comments',
'_get_json_dict',
'_info_url',
'_insert_comment',
'_orphaned',
'_populate',
'_populated',
'_replaced_more',
'_underscore_names',
'_update_comments',
'add_comment',
'approve',
'approved_by',
'author',
'author_flair_css_class',
'author_flair_text',
'banned_by',
'clear_vote',
'clicked',
'comments',
```

```
'created',
'created_utc',
'delete',
'distinguish',
'domain',
'downs',
'downvote',
'edit',
'edited',
'from_api_response',
'from_id',
'from_url',
'fullname',
'hidden',
'hide',
'id',
'is_self',
'likes',
'link_flair_css_class',
'link_flair_text',
'mark_as_nsfw',
'media',
'media_embed',
'name',
'num_comments',
'num_reports',
'over_18',
'permalink',
'reddit_session',
'refresh',
'remove',
'replace_more_comments',
'report',
'save',
'saved',
'score',
'selftext',
'selftext_html',
'set_flair',
'short_link',
'subreddit',
'subreddit_id',
'thumbnail',
'title',
'undistinguish',
'unhide',
'unmark_as_nsfw',
'unsave',
'ups',
'upvote',
'url',
'vote']
```

`vars` contain the object's attributes and the values they contain. For instance we can see that it has the variable `title` with the value `u'Newbie: stripping strings of last character`. `dir` returns the names in the local scope. You can also use `help` for introspection, if you wish to generate a longer help page. Worth noting is that PRAW contains a lot of property-decorated functions, i.e., functions that are used as variables. So if you're looking for something that behaves like a variable, it might not be in vars. One of these is *`short_link`*, which

returns a much shorter url to the submission and is called as a variable.

Another way of finding out how a reddit page is translated to variables is to look at the .json version of that page. Just append .json to a reddit url to look at the json version, such as the previous r/learnpython post. The variable name reddit uses for a variable is almost certainly the same PRAW uses.

1.2.2 The 3 Bot Problems.

Not Doing The Same Work Twice.

From the information we gained in the previous section, we see that the text portion of a submission is stored in the variable `selftext`. So we test if any of the strings are within the `selftext`, and if they are the bot sends me a message. But *u_Daimon_* should only ever receive a single message per submission. So we need to maintain a list of the submissions we've already notified *u_Daimon_* about. Each `Thing` has a unique ID, so we simply store the used ones in a list and check for membership before mailing. Finally we sleep 30 minutes and restart the main loop.

```
>>> prawWords = ['praw', 'reddit_api', 'mellort']
>>> op_text = submission.selftext.lower()
>>> has_praw = any(string in op_text for string in prawWords)
>>> if submission.id not in already_done and has_praw:
...     msg = '[PRAW related thread](%s)' % submission.short_link
...     r.send_message('_Daimon_', 'PRAW Thread', msg)
...     already_done.append(submission.id)
>>> time.sleep(1800)
```

Note that if the authenticated account has less than 2 link karma then PRAW will prompt for a captcha on stdin. Similar to how reddit would prompt for a captcha if the authenticated user tried to send the message via the webend.

Running Continually.

reddit.com is going to crash and other problems will occur. That's a fact of life. Good bots should be able to take this into account and either exit gracefully or survive the problem. This is a simple bot, where the loss of all data isn't very problematic. So for now we're simply going to accept that it will crash with total loss of data at the first problem encountered.

Keeping Within API Guidelines.

PRAW was designed to make following the API guidelines simple. It will not send a request more often than every 2 seconds and it caches every page for 30 seconds. This can be modified in *The Configuration Files*.

The problem comes when we run multiple bots / scripts at the same time. PRAW cannot share these settings between programs. So there will be at least 2 seconds between program A's requests and at least 2 seconds between program B's requests, but combined their requests may come more often than every 2 seconds. If you wish to run multiple program at the same time. Either combine them into one, ensure from within the programs (such as with message passing) that they don't combined exceed the API guidelines, or *edit the configuration files* to affect how often a program can send a request.

All 3 bot problems will be covered more in-depth in a future tutorial.

For now, you can continue to the next part of our tutorial series. *Comment Parsing*.

1.2.3 The full Question-Discover program

```
"""
Question Discover Program

Tutorial program for PRAW:
See https://github.com/praw-dev/praw/wiki/Writing-A-Bot/
"""

import time

import praw

r = praw.Reddit('PRAW related-question monitor by u/_Daimon_ v 1.0.'
                'Url: https://praw.readthedocs.org/en/latest/'
                'pages/writing_a_bot.html')
r.login()
already_done = []

prawWords = ['praw', 'reddit_api', 'mellort']
while True:
    subreddit = r.get_subreddit('learnpython')
    for submission in subreddit.get_hot(limit=10):
        op_text = submission.selftext.lower()
        has_praw = any(string in op_text for string in prawWords)
        # Test if it contains a PRAW-related question
        if submission.id not in already_done and has_praw:
            msg = '[PRAW related thread](%s)' % submission.short_link
            r.send_message('_Daimon_', 'PRAW Thread', msg)
            already_done.append(submission.id)
    time.sleep(1800)
```

1.3 A Simple "Call and Response" Bot

A common use case for the PRAW library is writing bots that monitor the /all/comments feed for comments meeting particular criteria, triggering an automated response of some kind. To build a bot of this kind, we'll need three things:

1. An effective way to monitor new comments.
2. A way to evaluate if a comment merits a response.
3. A way to trigger the desired action.

We'll need to provide the latter two parts, but PRAW contains an excellent convenience function for the first part: `praw.helpers.comment_stream`, which yields new comments in the order they are authored while handling rate limiting for us. We can constrain our attention to comments generated by a single subreddit, or monitor all public comments on reddit by passing in the subreddit name "all".

Using this helper function effectively reduces most bots to something like the following:

```
import praw

UA = 'my user-agent string. Contact me at /u/MyUsername or my@email.com'
r = praw.Reddit(UA)
r.login()

for c in praw.helpers.comment_stream(r, 'all'):
```

```
    if check_condition(c):
        bot_action(c)
```

Now all we need to do to transform this into a working bot is to provide a *check_condition* function and a *bot_action* function.

1.3.1 Working demo: ListFixerBot

Let's say we wanted a bot to correct bad list formatting with lazily-generated but hopefully improved reddit markdown (snudown). Reddit markdown recognizes lists when a numbered item is numbered like '1.', '2.' etc. Let's try to fix lists where the comment author replaced dots with parens. Our goal is to find comments containing lists that look like:

1. Item one
2. Item two
3. Item three

or even

1. Item one 2) Item two 3) Item three

and modify them so they will appear as:

1. Item one
2. Item two
3. Item three

First, let's write a function to determine if a comment matches what we're looking for. Following the template presented earlier, let's just call this function *check_condition* so we can drop it directly into our existing code.

```
def check_condition(c):
    text = c.body
    tokens = text.split()
    if "1)" in tokens and "2)" in tokens:
        return True
```

We can use regular expressions to replace these parens with dots, and even add preceding line breaks.

```
fixed = re.sub(r'([0-9]+)(\))', r'\n\n\1.', c.body)
```

This expression add two newline characters before each list item to break up list items that aren't separated by newlines (which they need to be). Reddit markdown will treat this as a single line between items. Also, markdown doesn't care what number we use when enumerating a list, so we can just use "1." for each list item and let the markdown interpreter figure out what the correct number should be.

We'll wrap this regular expression with our *bot_action* function to have the bot perform this operation only on the comments that passed the *check_condition* filter. If we include a "verbose" parameter, we can easily choose whether or not the function will output relevant text to the console by default. Keep in mind: reddit supports unicode text in comments.

```
def bot_action(c, verbose=True, respond=False):
    fixed = re.sub(r'(\n?)([0-9]+)(\))', r'\n\n\2.', c.body)

    if verbose:
        print c.body.encode("UTF-8")
        print "\n\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n\n"
        print fixed.encode("UTF-8")
```

We can similarly define an optional input parameter to choose whether or not the bot will actually respond to people with the corrected markdown. NB: If we want our bot to respond, it will need to be logged in. You can create a new account specific to the bot (recommended) or just use an existing login if you have one.

```
def bot_action(c, verbose=True, respond=False):
    fixed = re.sub(r'(\n?)([0-9]+)(\))', r'\n\n\2.', c.body)

    if verbose:
        print c.body.encode("UTF-8")
        print "\n\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n\n"
        print fixed.encode("UTF-8")

    if respond:
        head = "Hi! Let me try to beautify the list in your comment:\n\n"
        c.reply(head + fixed)
```

It's generally considered good practice to create a subreddit whose name matches the name of your bot to centralize discussion, questions, and feature requests relating to your bot. Referencing this subreddit in your bot's comments is a good way to inform people how best to provide feedback.

Our final `bot_action` function looks like this:

```
def bot_action(c, verbose=True, respond=False):
    fixed = re.sub(r'(\n?)([0-9]+)(\))', r'\n\n\2.', c.body)

    if verbose:
        print c.body.encode("UTF-8")
        print "\n\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n\n"
        print fixed.encode("UTF-8")

    if respond:
        head = "Hi! Let me try to beautify the list in  your comment:\n\n"
        tail = "\n\nI am a bot. You can provide feedback in my subreddit: /r/ListFormatFixer"
        c.reply(head + fixed + tail)
```

Here's our completed bot!

```
import re

def check_condition(c):
    text = c.body
    tokens = text.split()
    if "1)" in tokens and "2)" in tokens:
        return True

def bot_action(c, verbose=True, respond=False):
    fixed = re.sub(r'(\n?)([0-9]+)(\))', r'\n\n\2.', c.body)

    if verbose:
        print c.body.encode("UTF-8")
        print "\n\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n\n"
        print fixed.encode("UTF-8")

    if respond:
        head = "Hi! Let me try to beautify the list in  your comment:\n\n"
        tail = "\n\nI am a bot. You can provide feedback in my subreddit: /r/ListFormatFixer"
        c.reply(head + fixed + tail)

if __name__ is '__main__':
    import praw
```

```
    r = praw.Reddit(UA)

    # Provide a descriptive user-agent string. Explain what your bot does, reference
    # yourself as the author, and offer some preferred contact method. A reddit
    # username is sufficient, but nothing wrong with adding an email in here.
    UA = 'ListFormatFixer praw demo, Created by /u/shaggorama'

    # If you want the bot to be able to respond to people, you will need to login.
    # It is strongly recommended you login with oAuth
    # http://praw.readthedocs.org/en/stable/pages/oauth.html

    # NB: This login method is being deprecated soon
    r.login()

    for c in praw.helpers.comment_stream(r, 'all'):
        if check_condition(c):
            # set 'respond=True' to activate bot responses. Must be logged in.
            bot_action(c, respond=False)
```

Keep in mind: bots of this kind are often perceived as annoying and quickly get banned from many subreddits. If/when your bot gets banned, don't take it personally.

1.4 Comment Parsing

A common task for many bots and scripts is to parse a submission´s comments. In this tutorial we will go over how to do that as well as talking about comments in general. To illustrate the problems, we'll write a small script that replies to any comment that contains the text "Hello". Our reply will contain the text " world!".

1.4.1 Submission Comments

As usual, we start by importing PRAW and initializing our contact with reddit.com. We also get a *Submission* object, where our script will do its work.

```
>>> import praw
>>> r = praw.Reddit('Comment Scraper 1.0 by u/_Daimon_ see '
...                 'https://praw.readthedocs.org/en/latest/'
...                 'pages/comment_parsing.html')
>>> submission = r.get_submission(submission_id='11v36o')
```

After getting the *Submission* object we retrieve the comments and look through them to find those that match our criteria. Comments are stored in the attribute *comments* in a comment forest, with each tree root a toplevel comment. E.g., the comments are organized just like when you visit the submission via the website. To get to a lower layer, use *replies* to get the list of replies to the comment. Note that this may include *MoreComments* objects and not just *Comment*.

```
>>> forest_comments = submission.comments
```

As an alternative, we can flatten the comment forest to get a unordered list with the function *praw.helpers.flatten_tree()*. This is the easiest way to iterate through the comments and is preferable when you don't care about a comment's place in the comment forest. We don't, so this is what we are going to use.

```
>>> flat_comments = praw.helpers.flatten_tree(submission.comments)
```

To find out whether any of those comments contains the text we are looking for, we simply iterate through the comments.

```
>>> for comment in flat_comments:
...     if comment.body == "Hello":
...         reply_world(comment)
```

Our program is going to make comments to a submission. If it has bugs, then it might flood a submission with replies or post gibberish. This is bad. So we test the bot in r/test before we let it loose on a "real" subreddit. As it happens, our bot as described so far contains a bug. It doesn't test if we've already replied to a comment before replying. We fix this bug by storing the content_id of every comment we've replied to and test for membership of that list before replying. Just like in *Writing a reddit Bot*.

1.4.2 The number of comments

When we load a submission, the comments for the submission are also loaded, up to a maximum, just like on the website. At reddit.com, this max is 200 comments. If we want more than the maximum number of comments, then we need to replace the *MoreComments* with the *Comment*s they represent. We use the *replace_more_comments()* method to do this. Let's use this function to replace all *MoreComments* with the *Comment*s they represent, so we get all comments in the thread.

```
>>> submission.replace_more_comments(limit=None, threshold=0)
>>> all_comments = submission.comments
```

The number of *MoreComments* PRAW can replace with a single API call is limited. Replacing all *MoreComments* in a thread with many comments will require many API calls and so take a while due to API delay between each API call as specified in the api guidelines.

1.4.3 Getting all recent comments to a subreddit or everywhere

We can get comments made to all subreddits by using *get_comments()* and setting the subreddit argument to the value "all".

```
>>> import praw
>>> r = praw.Reddit('Comment parser example by u/_Daimon_')
>>> all_comments = r.get_comments('all')
```

The results are equivalent to /r/all/comments.

We can also choose to only get the comments from a specific subreddit. This is much simpler than getting all comments made to a reddit and filtering them. It also reduces the load on the reddit.

```
>>> subreddit = r.get_subreddit('python')
>>> subreddit_comments = subreddit.get_comments()
```

The results are equivalent to r/python/comments.

You can use multi-reddits to get the comments from multiple subreddits.

```
>>> multi_reddits = r.get_subreddit('python+learnpython')
>>> multi_reddits_comments = multi_reddits.get_comments()
```

Which is equivalent to r/python+learnpython/comments.

1.4.4 The full program

```
import praw

r = praw.Reddit('Comment Scraper 1.0 by u/_Daimon_ see '
                'https://praw.readthedocs.org/en/latest/'
                'pages/comment_parsing.html')
r.login('bot_username', 'bot_password')
submission = r.get_submission(submission_id='11v36o')
flat_comments = praw.helpers.flatten_tree(submission.comments)
already_done = set()
for comment in flat_comments:
    if comment.body == "Hello" and comment.id not in already_done:
        comment.reply(' world!')
        already_done.add(comment.id)
```

1.4.5 [deleted] comments

When a comment is deleted, in most cases, that comment will not be viewable with a browser nor the API. However, if a comment is made, and then a reply to that comment is made, and *then* the original comment is deleted, that comment will have its `body` and `author` attributes be `NoneType` via the API. The same goes with removed comments, unless the authenticated account is a mod of the subreddit whose comments you are getting. If you are a mod, and said comments are removed comments, they are left intact.

If a comment is made and then the account that left that comment is deleted, the comment body is left intact, while the `author` attribute becomes `NoneType`.

1.5 PRAW and OAuth

OAuth support allows you to use reddit to authenticate on non-reddit websites. It also allows a user to authorize an application to perform different groups of actions on reddit with his account. A moderator can grant an application the right to set flair on his subreddits without simultaneously granting the application the right to submit content, vote, remove content or ban people. Before the moderator would have to grant the application total access, either by giving it the password or by modding an account controlled by the applications.

Note: Support for OAuth is added in version 2.0. This will not work with any previous edition.

1.5.1 A step by step OAuth guide

PRAW simplifies the process of using OAuth greatly. The following is a step-by-step OAuth guide via the interpreter. For real applications you'll need a webserver, but for educational purposes doing it via the interpreter is superior. In the next section there is an *An example webserver*.

Step 1: Create an application.

Go to reddit.com's app page, click on the "are you a developer? create an app" button. Fill out the name, description and about url. Name must be filled out, but the rest doesn't. Write whatever you please. For redirect uri set it to `http://127.0.0.1:65010/authorize_callback`. All four variables can be changed later. Click create app and you should something like the following.

The random string of letters under your app's name is its `client id`. The random string of letters next to secret are your `client_secret` and should not be shared with anybody. At the bottom is the `redirect_uri`.

Step 2: Setting up PRAW.

> **Warning:** This example, like most of the PRAW examples, binds an instance of PRAW to the `r` variable. While we've made no distinction before, `r` (or any instance of PRAW) should not be bound to a global variable due to the fact that a single instance of PRAW cannot concurrently manage multiple distinct user-sessions.
> If you want to persist instances of PRAW across multiple requests in a web application, we recommend that you create an new instance per distinct authentication. Furthermore, if your web application spawns multiple processes, it is highly recommended that you utilize PRAW's *multiprocess* functionality.

We start as usual by importing the PRAW package and creating a *`Reddit`* object with a clear and descriptive useragent that follows the api rules.

```
>>> import praw
>>> r = praw.Reddit('OAuth testing example by u/_Daimon_ ver 0.1 see '
...                 'https://praw.readthedocs.org/en/latest/'
...                 'pages/oauth.html for source')
```

Next we set the app info to match what we got in step 1.

```
>>> r.set_oauth_app_info(client_id='stJlUSUbPQe5lQ',
...                      client_secret='DoNotSHAREWithANYBODY',
...                      redirect_uri='http://127.0.0.1:65010/'
...                                   'authorize_callback')
```

The OAuth app info can be automatically set, check out *The Configuration Files* to see how.

Step 3: Getting authorization from the user.

Now we need to have a user grant us authorization. We do this by sending them to a url, where the access we wish to be granted is listed, then they click 'allow' and are redirected to `redirect_uri` with a code in the url parameters that is needed for step 4.

The url we send them to is generated using *`get_authorize_url()`*. This takes 3 parameters. `state`, which is a string of your choice that represents this client, `scope` which are the reddit scope(s) we ask permission for (see *OAuth*

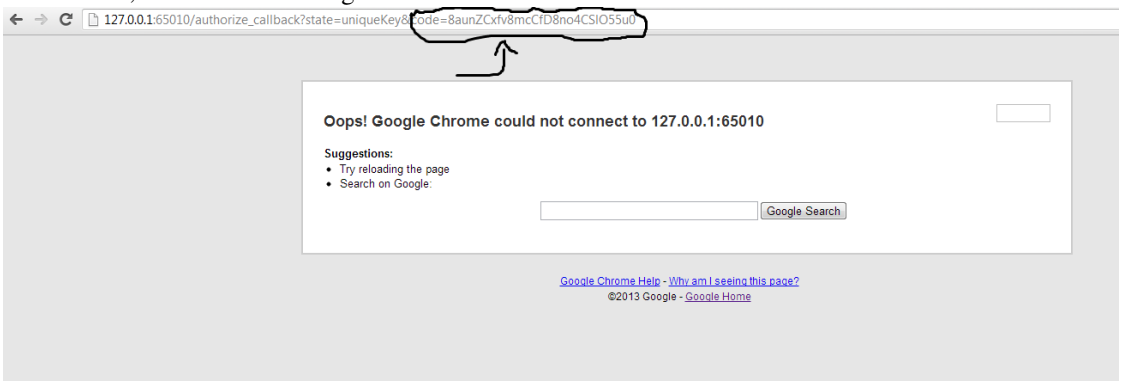
Scopes.) and finally `refreshable` which determines whether we can refresh the access_token (step 6) thus gaining permanent access.

For this tutorial we will need access to the identity scope and be refreshable.

```
>>> url = r.get_authorize_url('uniqueKey', 'identity', True)
>>> import webbrowser
>>> webbrowser.open(url)
>>> # click allow on the displayed web page
```

Step 4: Exchanging the code for an access_token and a refresh_token.

After completing step 3, you're redirected to the `redirect_uri`. Since we don't have a webserver running there at the moment, we'll see something like this. Notice the code in the url.



Now we simply exchange the code for the access information.

```
>>> access_information = r.get_access_information('8aunZCxfv8mcCf'
...                                               'D8no4CSlO55u0')
```

This will overwrite any existing authentication and make subsequent requests to reddit using this authentication unless we set the argument `update_session` to `False`.

`get_access_information()` returns a dict with the `scope`, `access_token` and `refresh_token` of the authenticated user. So later we can swap from one authenticated user to another with

```
>>> r.set_access_credentials(**access_information)
```

If `scope` contains `identity` then `r.user` will be set to the OAuthenticated user with `r.get_access_information` or *`set_access_credentials()`* unless we've set the `update_user` argument to `False`.

Step 5: Use the access.

Now that we've gained access, it's time to use it.

```
>>> authenticated_user = r.get_me()
>>> print authenticated_user.name, authenticated_user.link_karma
```

Step 6: Refreshing the access_token.

An access token lasts for 60 minutes. To get access after that period, we'll need to refresh the access token.

```
>>> r.refresh_access_information(access_information['refresh_token'])
```

This returns a dict, where the `access_token` key has had its value updated. Neither `scope` nor `refresh_token` will have changed.

Note: In version 3.2.0 and higher, PRAW will automatically attempt to refresh the access token if a refresh token is available when it expires. For most personal-use scripts, this eliminates the need to use *`refresh_access_information()`* except when signing in.

1.5.2 An example webserver

To run the example webserver, first install flask.

```
$ pip install flask
```

Then save the code below into a file called example_webserver.py, set the `CLIENT_ID` & `CLIENT_SECRET` to the correct values and run the program. Now you have a webserver running on `http://127.0.0.1:65010` Go there and click on one of the links. You'll be asked to authorize your own application, click allow. Now you'll be redirected back and your user details will be written to the screen.

```
# example_webserver.py #
########################

from flask import Flask, request

import praw

app = Flask(__name__)

CLIENT_ID = 'YOUR_CLIENT_ID'
CLIENT_SECRET = 'YOUR CLIENT SECRET'
REDIRECT_URI = 'http://127.0.0.1:65010/authorize_callback'

@app.route('/')
def homepage():
    link_no_refresh = r.get_authorize_url('UniqueKey')
    link_refresh = r.get_authorize_url('DifferentUniqueKey',
                                       refreshable=True)
    link_no_refresh = "<a href=%s>link</a>" % link_no_refresh
    link_refresh = "<a href=%s>link</a>" % link_refresh
    text = "First link. Not refreshable %s</br></br>" % link_no_refresh
    text += "Second link. Refreshable %s</br></br>" % link_refresh
    return text

@app.route('/authorize_callback')
def authorized():
    state = request.args.get('state', '')
    code = request.args.get('code', '')
    info = r.get_access_information(code)
    user = r.get_me()
    variables_text = "State=%s, code=%s, info=%s." % (state, code,
                                                      str(info))
    text = 'You are %s and have %u link karma.' % (user.name,
                                                   user.link_karma)
    back_link = "<a href='/'>Try again</a>"
    return variables_text + '</br></br>' + text + '</br></br>' + back_link
```

```
if __name__ == '__main__':
    r = praw.Reddit('OAuth Webserver example by u/_Daimon_ ver 0.1. See '
                    'https://praw.readthedocs.org/en/latest/'
                    'pages/oauth.html for more info.')
    r.set_oauth_app_info(CLIENT_ID, CLIENT_SECRET, REDIRECT_URI)
    app.run(debug=True, port=65010)
```

1.5.3 OAuth Scopes.

The following list of access types can be combined in any way you please. Just pass a string containing each scope that you want (if you want several, they should be seperated by spaces, e.g. `"identity submit edit"`) to the scope argument of the `get_authorize_url` method. The description of each scope is identical to the one users will see when they have to authorize your application.

Type	Description	PRAW methods
creddits	Spend my reddit gold creddits on giving gold to other users.	gild
edit	Edit and delete my comments and submissions.	edit, delete
flair	Select my subreddit flair. Change link flair on my submissions.	get_flair_choices, select_flair
history	Access my voting history and comments or submissions I've saved or hidden.	get_hidden, get_saved, get_upvoted, get_downvoted (the last two do not require the `history` scope if upvoted and downvoted posts are made public via the preferences). get_comments now also requires history.
identity	Access my reddit username and signup date.	get_me
modconfig	Manage the configuration, sidebar, and CSS of subreddits I moderate.	get_settings, set_settings, set_stylesheet, upload_image, create_subreddit, update_settings
modcontributors	Add/remove users to approved submitter lists and ban/unban users from subreddits I moderate.	add_ban, remove_ban, add_contributor, remove_contributor, add_wiki_contributor, remove_wiki_contributor, add_wiki_ban, remove_wiki_ban (the last four also require the `modwiki` scope)
modflair	Manage and assign flair in subreddits I moderate.	add_flair_template, clear_flair_template, delete_flair, configure_flair, flair_list, set_flair, set_flair_csv
modlog	Access the moderation log in subreddits I moderate.	get_mod_log
modothers	Invite or remove other moderators from subreddits I moderate.	add_moderator, remove_moderator
modposts	Approve, remove, mark nsfw, and distinguish content in subreddits I moderate.	approve, distinguish, remove, mark_as_nsfw, unmark_as_nsfw, undistinguish.
modself	Accept invitations to moderate a subreddit. Remove myself as a moderator or contributor of subreddits I moderate or contribute to.	accept_moderator_invite
modwiki	Change editors and visibility of wiki pages in subreddits I moderate.	add_editor, get_settings (when used on a WikiPage obejct, not a Subreddit object), edit_settings, remove_editor
mysubreddits	Access the list of subreddits I moderate, contribute to, and subscribe to.	my_contributions, my_moderation, my_reddits
privatemessages	Access my inbox and send private messages to other users.	mark_as_read, mark_as_unread, send_message, get_inbox, get_modmail, get_sent, get_unread
read	Access posts, listings and comments through my account.	get_comments, get_new_by_date (and the other listing funcs), get_submission, get_subreddit, get_content, from_url can now access things in private subreddits that the authenticated user has access to.
report	Report content for rules violations. Hide & show individual submissions.	report, hide, unhide
save	Save and unsave comments and submissions.	save, unsave
submit	Submit links and comments from my account.	add_comment, reply, submit
subscribe	Manage my subreddit subscriptions.	subscribe, unsubscribe
vote	Submit and change my votes on comments and submissions.	clear_vote, upvote, downvote, vote
wikiedit	Edit wiki pages on my behalf	edit_wiki_page
wikiread	Read wiki pages through my account.	get_wiki_page, get_wiki_pages

1.6 Lazy Objects

Each API request to Reddit must be separated by a 2 second delay, as per the API rules. So to get the highest performance, the number of API calls must be kept as low as possible. PRAW uses lazy objects to only make API calls when/if the information is needed.

For instance, if you're doing the following:

```
>>> import praw
>>> r = praw.Reddit(user_agent=UNIQUE_AND_DESCRIPTIVE_USERAGENT)
>>> subreddit = r.get_subreddit('askhistorians')
```

Then *get_subreddit()* didn't send a request to Reddit. Instead it created a lazy *Subreddit* object, that will be filled out with data when/if necessary:

```
>>> for post in subreddit.get_hot():
...     pass
```

Information about the subreddit, like number of subscribers or its description, is not needed to get the hot listing. So PRAW doesn't request it and avoids an unnecessary API call, making the code above run about 2 seconds faster due to lazy objects.

1.6.1 When do the lazy loaded objects become non-lazy?

When the information is needed. It's really that simple. Continuing the code from above:

```
>>> subreddit.has_fetched
False # Data has not been fetched from reddit. It's a lazily loaded object.
>>> subreddit.public_description
u'Questions about the past: answered!'
>>> subreddit.has_fetched
True # No longer lazily loaded.
```

1.6.2 Where are the lazy objects?

PRAW uses lazy objects whenever possible. Objects created with *get_subreddit()* or *get_redditor()* are lazy, unless you call the methods with `fetch=True`. In this case all data about the object will be fetched at creation:

```
>>> non_lazy_subreddit = r.get_subreddit('askhistorians', fetch=True)
>>> non_lazy_subreddit.has_fetched
True
```

When one object references another, the referenced object starts as a lazy object:

```
>> submission = r.get_submission(submission_id="16m0uu")
>> submission.author # Reference to a lazy created Redditor object.
```

Whenever a method returns a generator, such as when you call *get_front_page()* then that's also a lazy object. They don't send API requests to reddit for information until you actually need it by iterating through the generator.

1.6.3 Lazy objects and errors

The downside of using lazy objects is that any error will not happen when the lazy object is created, but instead when the API call is actually made:

```
>> private_subreddit = r.get_subreddit('lounge')
>> private_subreddit.has_fetched
False
>> private_subreddit.subscribers

Traceback (most recent call last):
...
praw.errors.Forbidden: <Response [403]>
```

1.7 Concurrent PRAW Instances

By default, PRAW works great when there is a one-to-one mapping between running PRAW processes, and the IP address / user account that you make requests from. In fact, as of version 2.1.0, PRAW has multithreaded support as long as there is a one-to-one mapping between thread and PRAW *Reddit* object instance. That is, in order to be thread safe, each thread needs to have its own *Reddit* instance [1]. In addition to multithreaded rate-limiting support, PRAW 2.1.0 added multiprocess rate limiting support.

1.7.1 praw-multiprocess

PRAW version 2.1.0 and later install with a program **praw-multiprocess**. This program provides a request handling server that manages the rate-limiting and caching of any PRAW process which directs requests toward it. Starting **praw-multiprocess** is as simple as running `praw-multiprocess` from your terminal / command line assuming you *installed PRAW* properly.

By default **praw-multiprocess** will listen only on *localhost* port *10101*. You can adjust those settings by passing in `--addr` and `--port` arguments respectively. For instance to have **praw-multiprocess** listen on all valid addresses on port 65000, execute via: `praw-multiprocess --addr 0.0.0.0 --port 65000`. For a full list of options execute `praw-multiprocess --help`.

1.7.2 PRAW's MultiprocessingHandler

In order to interact with a **praw-multiprocess** server, PRAW needs to be instructed to use the *MultiprocessHandler* rather than the *DefaultHandler*. In your program you need to pass an instance of *MultiprocessHandler* into the `handler` keyword argument when creating the *Reddit* instance. Below is an example to connect to a **praw-multiprocess** server running with the default arguments:

```
import praw
from praw.handlers import MultiprocessHandler

handler = MultiprocessHandler()
r = praw.Reddit(user_agent='a descriptive user-agent', handler=handler)
```

With this configuration, all network requests made from your program(s) that include the above code will be *proxied* through the *praw-multiprocess* server. All requests made through the same **praw-multiprocess** server will respect reddit's API rate-limiting rules

If instead, you wish to connect to a **praw-multiprocess** server running at address `10.0.0.1` port 65000 then you would create the PRAW instance via:

[1] It is undetermined at this time if the same authentication credentials can be used on multiple instances where the modhash is concerned.

```
import praw
from praw.handlers import MultiprocessHandler

handler = MultiprocessHandler('10.0.0.1', 65000)
r = praw.Reddit(user_agent='a descriptive user-agent', handler=handler)
```

1.7.3 PRAW Multiprocess Resiliency

With all client/server type programs there is the possibility of network issues or simply a lack of an available server. PRAW's *MultiprocessHandler* was created to be quite resilient to such issues. PRAW will retry indefinitely to connect to **praw-multiprocess** server. This means that a **praw-multiprocess** server can be stopped and restarted without any effect on programs utilizing it.

On the other hand, consecutive network failures where the *MultiprocessHandler* has no issue establishing a connection to a **praw-multiprocess** server will result in *ClientException* after three failures. Such failures are **not** expected to occur and if reproducable should be *reported*.

1.8 Contributor Guidelines

PRAW gladly welcomes new contributions. As with most larger projects, we have an established consistent way of doing things. A consistent style increases readability, decreases bug-potential and makes it faster to understand how everything works together.

PRAW follows **PEP 8** and **PEP 257**. You can use `lint.sh` to test for compliance with these PEP's. The following are PRAW-specific guidelines in to those PEP's.

1.8.1 Code

- Objects are sorted alphabetically.
- Things should maintain the same name throughout the code. **kwargs should never be **kw.
- Things should be stored in the same data structure throughout the code.

1.8.2 Testing

- If you're adding functionality, either add tests or suggest how it might be tested.
- In assertEquals, the first value should be the value you're testing and the second the known value.

1.8.3 Documentation

- All publicly available functions, classes and modules should have a docstring.
- Use correct terminology. A subreddits name is something like ' t5_xyfc7'. The correct term for a subreddits "name" like python is its display name.
- When referring to any reddit. Refer to it as 'reddit'. When you are speaking of the specific reddit instance with the website reddit.com, refer to it as 'reddit.com'. This helps prevent any confusion between what is universally true between all reddits and what specifically applies to the most known instance.

1.9 The Configuration Files

PRAW can be configured on the global, user, local and `Reddit` instance levels. This allows for easy custom configuration to fit your needs.

To build the configuration settings, first the global configuration file is loaded. It contains the default settings for all PRAW applications and should never be modified. Then PRAW opens the user level configuration file (if it exists) and any settings here will take precedence over those in the global configuration file. Then PRAW opens the local level configuration file (if it exists) and any settings here will take precedence over those previously defined. Finally you can set configuration settings by giving them as additional arguments when instantiating the `Reddit` object. Settings given this way will take precedence over those previously defined.

```
import praw

user_agent = ("Configuration setting example by /u/_Daimon_. See "
              "https://praw.readthedocs.org/en/latest/pages/configuration_files.html")
r = praw.Reddit(user_agent=user_agent, log_requests=1)
```

1.9.1 Config File Locations

The configuration on all levels is stored in a file called `praw.ini`.

The *global* configuration file is located in the **praw** package location. This file provides the system wide default and should never be modified.

The *user* configuration file location depends on your operating system. Assuming typical operating system installations and the username *foobar* the path for specific operating systems should be:

- **WINDOWS XP:** C:\Documents and Settings\foobar\Application Data\praw.ini
- **WINDOWS Vista/7:** C:\Users\foobar\AppData\Roaming\praw.ini
- **OS with XDG_CONFIG_HOME defined:** $XDG_CONFIG_HOME/praw.ini
- **OS X / Linux:** /home/foobar/.config/praw.ini

The *local* configuration file is located in the current working directory. This location works best if you want script-specific configuration files.

1.9.2 Configuration Variables

The following variables are provided in the [DEFAULT] section of the *global* config file. Each site can overwrite any of these variables.

- *api_request_delay:* A **float** that defines the number of seconds required between calls to the same domain.
- *api_domain:* A **string** that defines the *domain* to use for all standard API requests.
- *check_for_updates:* A **boolean** to indicate whether or not to check for package updates.
- *cache_timeout:* An **integer** that defines the number of seconds to internally cache GET/POST requests based on URL.
- *http_proxy:* A **string** that declares a http proxy to be used. It follows the requests proxy conventions, e.g., `http_proxy: http://user:pass@addr:port`. If no proxy is specified, PRAW will pick up the environment variable for http_proxy, if it has been set.

- *https_proxy:* A **string** that declares a https proxy to be used. It follows the requests proxy conventions, e.g., `https_proxy: http://user:pass@addr:port`. If no proxy is specified, PRAW will pick up the environment variable for https_proxy, if it has been set.
- *log_requests:* An **integer** that determines the level of API call logging.
  - **0:** no logging
  - **1:** log only the request URIs
  - **2:** log the request URIs as well as any POST data
- *oauth_domain:* A **string** that defines the *domain* where OAuth authenticated requests are sent.
- *oauth_https:* A **boolean** that determines whether or not to use HTTPS for oauth connections. This should only be changed for development environments.
- *output_chars_limit:* An **integer** that defines the maximum length of unicode representations of *Comment*, *Message* and *Submission* objects. This is mainly used to fit them within a terminal window line. A negative value means no limit.
- ***permalink_domain:* A string that defines the *domain* that is used for** the display *permalink* for Submissions and Comments.
- ***short_domain:* A string that defines the *domain* that is used for** short urls.
- *store_json_result:* A **boolean** to indicate if json_dict, which contains the original API response, should be stored on every object in the json_dict attribute. Default is `False` as memory usage will double if enabled. For lazy objects, json_dict will be `None` until the data has been fetched.
- *timeout:* Maximum time, a **float**, in seconds, before a single HTTP request times out. urllib2.URLError is raised upon timeout.
- *validate_certs:* A **boolean** to indicate if SSL certificates should be validated or not. If not specified, will default to `True`. This is mainly for testing local reddit installations with self-signed certificates.
- *xxx_kind:* A **string** that maps the *type* returned by json results to a local object. **xxx** is one of: *comment*, *message*, *more*, *redditor*, *submission*, *subreddit*, *userlist*. This variable is needed as the object-to-kind mapping is created dynamically on site creation and thus isn't consistent across sites.

The are additional variables that each site can define. These additional variables are:

- *domain:* (**REQUIRED**) A **string** that provides the domain name, and optionally port, used to connect to the desired reddit site. For reddit proper, this is: *www.reddit.com*. Note that if you are running a custom reddit install, this name needs to match the domain name listed in the reddit configuration ini.
- *user:* A **string** that defines the default username to use when *login* is called without a *user* parameter.
- *pswd:* A **string** that defines the password to use in conjunction with the provided *user*.
- *ssl_domain:* A **string** that defines the *domain* where encrypted requests are sent. This is used for logging in, both OAuth and user/password. When not provided, these requests are sent in plaintext (unencrypted).
- *oauth_client_id:* A **string** that, if given, defines the `client_id` a reddit object is initialized with.
- *oauth_client_secret:* A **string** that, if given, defines the `client_secret` a reddit object is initialized with.
- *oauth_redirect_uri:* A **string** that, if given, defines the `redirect_uri` a reddit object is initialized with. If *oauth_client_id* and *oauth_client_secret* is also given, then *get_authorize_url()* can be run without first setting the oauth settings with running *set_oauth_app_info()*.
- *oauth_refresh_token:* A **string** that, if given, defines the `refresh_token` a reddit object is initialized with. If *oauth_client_id*, *oauth_client_secret*, and *oauth_redirect_uri* are also given, then *refresh_access_information()* can be run with no arguments to acquire new access information without first running *get_authorize_url()* and *get_access_information()*.

Note: The tracking for *api_request_delay* and *cache_timeout* is on a per-domain, not per-site, basis. Essentially, this means that the time since the last request is the time since the last request from any site to the domain in question. Thus, unexpected event timings may occur if these values differ between sites to the same domain.

The Sites

The default provided sites are:

- *reddit:* This site defines the settings for reddit proper. It is used by default if the *site* parameter is not defined when creating the *Reddit* object.
- *local:* This site defines settings for a locally running instance of reddit. The *xxx_kind* mappings may differ so you may need to shadow (overwrite) the 'local' site in your *user*-level or *local*-level `praw.ini` file.

Additional sites can be added to represent other instances of reddit or simply provide an additional set of credentials for easy access to that account. This is done by adding `[YOUR_SITE]` to the `praw.ini` file and then calling it in `praw.Reddit`. For example, you could add the following to your `praw.ini` file:

```
[YOUR_SITE]
domain: www.myredditsite.com
ssl_domain: ssl.myredditsite.com
user: bboe
pswd: this_isn't_my_password
api_request_delay: 7.0
```

From there, to specify the reddit instance of "YOUR_SITE", you would do something like this:

```
import praw

r = praw.Reddit(user_agent='Custom Site Example for PRAW',
                site_name='YOUR_SITE')
```

Of course, you can use any of the above Configuration Variables as well.

Example praw.ini file

The following is an example `praw.ini` file which has 4 sites defined: 2 for a reddit proper accounts and 2 for local reddit testing.

```
[bboe]
domain: www.reddit.com
ssl_domain: ssl.reddit.com
user: bboe
pswd: this_isn't_my_password

[reddit_dev]
domain: www.reddit.com
ssl_domain: ssl.reddit.com
user: someuser
pswd: somepass

[local_dev1]
domain: reddit.local:8000
user: someuser
pswd: somepass

[local_wacky_dev]
domain: reddit.local:8000
```

```
user: someuser
pswd: somepass
api_request_delay: 5.0
default_content_limit: 2
```

1.10 Frequently Asked Questions

This is a list of frequently asked questions and a description of non-obvious behavior in PRAW and reddit.

1.10.1 FAQ

How do I get a comment by ID?

If you have a permalink to a comment you can use get_submission on the permalink, and then the first comment should be the desired one.

```
>>> s = r.get_submission('http://www.reddit.com/r/redditdev/comments/s3vcj/_/c4axeer')
>>> your_comment = s.comments[0]
```

How can I handle captchas myself?

Normally, PRAW will automatically prompt for a response whenever a captcha is required. This works great if you're interactively running a program on the terminal, but may not be desired for other applications. In order to prevent the automatic prompting for captchas, one must add `raise_captcha_exception=True` to the function call:

```
>>> r.submit('reddit_api_test', 'Test Submission', text='Failed Captcha Test',
... raise_captcha_exception=True)
Traceback (most recent call last):
...
praw.errors.InvalidCaptcha: `care to try these again?` on field `captcha`
```

With this added keyword, you program can catch the *InvalidCaptcha* exception and obtain the `captcha_id` via `response['captcha']` of the exception instance.

In order to manually pass the captcha response to the desired function you must add a `captcha` keyword argument with a value of the following format:

```
{'iden' : 'the captcha id', 'captcha': 'the captcha response text'}
```

For instance if the solution to `captcha_id f7UdxDmAEMbyLrb5fRALWJWvI5RPgGve` is `FYEMHB` then the above submission can be made with the following call:

```
>>> captcha = {'iden': 'f7UdxDmAEMbyLrb5fRALWJWvI5RPgGve', 'captcha': 'FYEMHB'}
>>> r.submit('reddit_api_test', 'Test Submission', text='Failed Captcha Test',
... raise_captcha_exception=True, captcha=captcha)
<praw.objects.Submission object at 0x2b726d0>
```

Note that we still add `raise_captcha_exception=True` in case the provided captcha is incorrect.

I made a change, but it doesn't seem to have an effect?

PRAW follows the api guidelines which require that pages not be requested more than every 30 seconds. To do this PRAW has an internal cache, which stores results for 30 seconds and give you the cached result if you request the same page within 30 seconds.

Some commands take a while. Why?

PRAW follows the api guidelines which require a 2 second delay between each API call via CookieAuth. If you are exclusively using OAuth2, you are allowed to change this delay in your `praw.ini` file to be a 1 second delay. This will be the default once CookieAuth is deprecated.

When I print a Comment only part of it is printed. How can I get the rest?

A *`Comment`* is an object which contain a number of attributes with information about the object such as `author`, `body` or `created_utc`. When you use `print` the object string (well unicode) representation of the object is printed. Use `vars` to discover what attributes and their values an object has, see *Writing a reddit Bot* for more details.

Why does the karma change from run to run?

This inconsistency is due to reddit's obfuscation of the upvotes and downvotes. The obfuscation is done to everything and everybody to thwart potential cheaters. There's nothing we can do to prevent this.

How do I report an issue with PRAW?

If you believe you found an issue with PRAW that you can reliably reproduce please file an issue on github. When reporting an issue, please note the version of PRAW you are using (`python -c 'import praw; print(praw.__version__)'`), and provide the code necessary to reproduce the issue. It is strongly suggested that you condense your code as much as possible.

Alternatively, if you cannot reliably reproduce the error, or if you do not wish to create a github account, you can make a submission on /r/redditdev.

1.10.2 Non-obvious behavior and other need to know

- All of the listings (list of stories on subreddit, etc.) are generators, *not* lists. If you need them to be lists, an easy way is to call `list()` with your variable as the argument.
- The default limit for fetching Things is your reddit preferences default, usually 25. You can change this with the `limit` param. If want as many Things as you can then set `limit=None`.
- We can at most get 1000 results from every listing, this is an upstream limitation by reddit. There is nothing we can do to go past this limit. But we may be able to get the results we want with the *`search()`* method instead.

1.11 Change Log

Read /r/changelog to be notified of upstream changes.

1.11.1 Unreleased

- **[BUGFIX]** Prevent built-in method *dir* from causing RedditContentObjects to perform web requests in Python 2.
- **[FEATURE]** Added support for thread locking. See *lock()* and *unlock()*.

1.11.2 3.4.0 (2016-02-21)

Added

- **Added support for modmail muting. See** *add_mute()*, *get_muted()*, *remove_mute()*, *mute_modmail_author()*, and *unmute_modmail_author()*.
- *default_subreddits()*.
- Support for * OAuth scopes.
- *get_traffic()*

Changed

- Image uploads support PNG images as small as 67 bytes.

1.11.3 PRAW 3.3.0

- **[BUGFIX]** Fixed login password prompt issue on windows (#485).
- **[BUGFIX]** Fixed unicode user-agent issue (#483).
- **[BUGFIX]** Fix duplicate request issue with comment and submission streams (#501).
- **[BUGFIX]** Stopped *praw.objects.Redditor.friend()* from raising LoginRequired when using OAuth.
- **[BUGFIX]** Stopped a json-parsing error from occuring in cases where reddit's response to a request was an empty string. `request_json()` will now simply return that empty string.
- **[BUGFIX]** Fix AssertionError when hiding and unhiding under OAuth, raised by stacked scope decorators.
- **[BUGFIX]** Fix AttributeError when hiding and unhiding under OAuth without the "identity" scope, raised when PRAW tried to evict the user's /hidden page from the cache.
- **[CHANGE]** Added messages to all PRAW exceptions (#491).
- **[CHANGE]** Made it easier to send JSON dumps instead of form-encoded data for http requests. Some api-v1 endpoints require the request body to be in the json format.
- **[CHANGE]** Moved and deprecated *praw.objects.LoggedInRedditor.get_friends()* to `praw.AuthenticatedReddit`, leaving a pointer in its place. Previously, `get_friends` was difficult to access because the only instance of *LoggedInRedditor* was the reddit session's *user* attribute, which is only instantiated if the user has the "identity" scope. By moving `get_friends` to the reddit session, it can be used without having to manipulate a *praw.objects.Redditor* intsance's class.
- **[CHANGE]** Removed support for Python 2.6 and Python 3.2 (#532).
- **[FEATURE]** Added support for adding "notes" to your friends. Users with reddit Gold can set the `note` parameter of *praw.objects.Redditor.friend()*. 300 character max enforced by reddit.
- **[FEATURE]** New *praw.objects.Redditor.get_friend_info()* to see info about one of your friends. Includes their name, ID, when you added them, and if you have reddit Gold, your note about them.

1.11.4 PRAW 3.2.1

- **[BUGFIX]** Fixed "multiple values for argument" error when solving captchas.

1.11.5 PRAW 3.2.0

- **[BUGFIX]** Fixed methods which require more than one OAuth scope.
- **[BUGFIX]** Fixed *praw.objects.WikiPage.remove_editor()* raising AssertionError when used through OAuth.
- **[BUGFIX]** Fixed `get_wiki_page()` not sending the OAuth headers.
- **[CHANGE]** *praw.objects.Refreshable.refresh()* will now always return a fresh object. Previously, Subreddits and Redditors would use cache content when available.
- **[CHANGE]** *praw.objects.WikiPage* is now refreshable, and will lazy-load.
- **[FEATURE]** Added methods `leave_moderator()` and `leave_contributor()` to *praw.__init__.AuthenticatedReddit* and *praw.objects.Subreddit*.
- **[FEATURE]** Added support for double stickies. Use boolean parameter *bottom* to choose which sticky to set or get.
- **[FEATURE]** Added methods *praw.objects.Message.collapse()* and *praw.objects.Message.uncollapse()*.
- **[FEATURE]** If an OAuth2 refresh token is available, and PRAW encounters an "Invalid Token" error, it will attempt to refresh the token for you automatically.
- **[REDDIT]** Fixed case where the user could not reply to private messages with the *privatemessages* scope because the endpoint required the *submit* scope. reddit has fixed this quirk, and PRAW now chooses the proper scope.

1.11.6 PRAW 3.1.0

- **[BUGFIX]** Fixed method *get_random_submission* which failed to raise the expected redirect exception.
- **[CHANGE]** Replaced instances of "liked" and "disliked" with "upvoted" and "downvoted". The get_liked and get_disliked methods in `objects.Redditor` still exist, but point to the new methods.
- **[CHANGE]** Fixed the *subreddits* attribute of *praw.objects.Multireddit* being returned as a list of dicts. It is now a list of Subreddit objects.
- **[CHANGE]** The *display_name* attr of `objects.Subreddit` and the *name* attr of `objects.Redditor` are now set when instantiated, and do not lazyload. To guarantee that these strings are properly cased, the user must instantiate the object with *fetch=True*, or call *object.refresh()*.
- **[FEATURE]** Added `get_comment_replies()` and `get_post_replies()` to the *praw.__init__.PrivateMessagesMixin*

1.11.7 PRAW 3.0.0

- **[CHANGE]** All requests should now be through HTTPS.
- **[CHANGE]** All exceptions should be in the PRAW namespace. In particular, there should be no more exceptions under the `requests` namespace.
- **[CHANGE]** All previously deprecated methods have been removed.

- **[CHANGE]** The `display_name` attribute on instances of `Subreddit` is now lazily loaded and will match the casing on the site, not the casing used to construct the Subreddit instance. To quickly fetch the name of an unloaded Subreddit, use `str(sub_instance)`, or `unicode(sub_instance)`.
- **[CHANGE]** Removed `praw.Config` instance attribute `is_reddit`.
- **[CHANGE]** *evict()* now returns the number of items evicted.
- **[CHANGE]** Removed `praw.ini` parameter `decode_html_entities`. Entities, e.g., `&`, `< >`, are now always decoded.
- **[FEATURE]** Added `get_message()` to fetch a single Message object by its ID.
- **[FEATURE]** Added `get_sticky()` to get a Subreddit's sticky post.
- **[FEATURE]** Refresh tokens can be specified in `praw.ini` via `oauth_refresh_token`.
- **[FEATURE]** Added `create_multireddit()` to create a new Multireddit.
- **[FEATURE]** Added `copy_multireddit()` to copy a Multireddit.
- **[FEATURE]** Added `edit_multireddit()` to edit an existing Multireddit.
- **[FEATURE]** Added `get_multireddits()` to get a list of Multireddits belonging to the requested user.
- **[FEATURE]** Added `rename_multireddit()` to rename an existing Multireddit.
- **[FEATURE]** Added `set_suggested_sort()` to change a submission's sort order.
- **[FEATURE]** Added `method` as optional parameter to `request_json()`, so that a request method other than 'POST' can be specified.
- **[FEATURE]** Added *praw.__init__.ReportMixin.hide()* and *praw.__init__.ReportMixin.unhide()*, which accept up to 50 fullnames to be hidden at one time. The appropriate methods in `objects.Hideable` now point here instead.
- **[FEATURE]** Added `add_editor()`, `remove_editor()`, `get_settings()` and `edit_settings()` to `WikiPage` for managing editors and permission levels of individual wiki pages.
- **[REDDIT]** Removed `send_feedback` as it is no longer supported by reddit.
- **[REDDIT]** Added `DeprecationWarning` to `login()` as reddit will stop supporting cookie-based authentication on 2015/08/03.

1.11.8 PRAW 2.1.21

- **[BUGFIX]** Fix assertion error in *replace_more_comments()* with continue this thread links that have more than one child.
- **[BUGFIX]** *refresh()* on *praw.objects.Submission* no longer loses comment sort order and other manually specified parameters.
- **[REDDIT]** Add `hide_ads` as a parameter to *set_settings()*.
- **[REDDIT]** *create_redditor()* no longer requires a captcha
- **[REDDIT]** *create_subreddit()* may require a captcha

1.11.9 PRAW 2.1.20

- **[BUGFIX]** Attempting to lazyload an attribute of a comment that has been removed will explicitly raise a *praw.errors.InvalidComment()* exception, rather than an `IndexError` (issue #339).

- **[BUGFIX]** *replace_more_comments()* handles *continue this thread* type `MoreComments` objects.
- **[FEATURE]** Added *praw.helpers.valid_redditors()*.
- **[FEATURE]** Added a `nsfw` parameter to *get_random_subreddit()* that permits fetching a random NSFW Subreddit. This change also supports fetching these subreddits via `get_subreddit('randnsfw')`.
- **[FEATURE]** Added a `from_sr` parameter to *send_message()* to send the private message from a subreddit you moderate (Like the "From" dropdown box when composing a message).
- **[FEATURE]** Added `Multireddit`
- **[FEATURE]** Added `get_multireddit()` to get a single multireddit obj
- **[FEATURE]** Added `get_my_multireddits()` to get all multireddits owned by the logged in user.
- **[FEATURE]** Added `get_multireddit()` to `Redditor` to quickly get a multireddit belonging to that user.
- **[FEATURE]** *praw.objects.Comment*, *praw.objects.Redditor*, and *praw.objects.Submission* are now gildable.
- **[FEATURE]** *praw.objects.Comment* is now saveable.
- **[REDDIT]** Handle upstream change in reddit's OAuth2 scope parsing.

1.11.10 PRAW 2.1.19

- **[BUGFIX]** Support URLs in *search()*.
- **[BUGFIX]** Fix bug where `json_dict` was set to `None` when it should not have been.
- **[BUGFIX]** Fix *get_subreddit_recommendations()* to work with the updated API route.
- **[BUGFIX]** Track time between requests using `timeit.default_timer`.
- **[CHANGE]** *get_friends()* and *get_banned()* once again work.
- **[CHANGE]** `is_root()` no longer requires fetching submission objects.
- **[REDDIT]** Support `thing_id` lists in *get_info()*.
- **[FEATURE]** Support providing HTTPS proxies, that is, proxies specific to handling HTTPS requests.
- **[FEATURE]** *get_liked()* and *get_disliked()* now accept additional arguments, e.g., limit.
- **[FEATURE]** Add *get_messages()* for specifically retreiving messages (not replies).
- **[REDDIT]** Add `collapse_deleted_comments` as a parameter to *set_settings()*.
- **[REDDIT]** *get_stylesheet()* now supports using the `modconfg` OAuth scope.
- **[REDDIT]** *get_stylesheet()* no longer accepts the `prevstyle` argument.

1.11.11 PRAW 2.1.18

- **[FEATURE]** Add the *get_flair_choices()* method to the *Submission* class, which returns the choices for user flair in the subreddit and the current flair of the authenticated user.
- **[FEATURE]** Add the *get_flair_choices()* method to the *Submission* class, which returns the choices for link flair on this submission as well as it's current flair.
- **[BUGFIX]** Fix python3 issue with `func_defaults`.

- **[REDDIT]** Avoid exceptions caused by upstream changes by reddit with respect to conflicts between json attributes and *RedditContentObject* properties. In such cases, the attribute from reddit will be suffixed with "_reddit".

1.11.12 PRAW 2.1.17

- **[BUGFIX]** Remove the built-in `score` property from comments as reddit provides that attribute as of 2014/06/18.
- **[FEATURE]** *submit()* now supports a `resubmit` argument to allow the submission of an already submitted url.

1.11.13 PRAW 2.1.16

- **[BUGFIX]** Fix incorrect username when building Redditor objects from wikipage submissions.
- **[CHANGE]** Increase the dependency of `update_checker` to 0.10 or later to prevent ImportWarnings (issue 291).
- **[CHANGE]** *get_banned()* now takes a `user_only` argument (default: `True`). When the value is explicitly passed as `False` the return value is not a generator of `Redditor` objects, but a generator of dictionaries whose `name` key corresponds to the `Redditor` object and whose ban-note is at key `note`.
- **[FEATURE]** Enable gathering of duplicate submissions for a Submission object (issue 290).
- **[FEATURE]** Add *praw.__init__.AuthenticatedReddit.delete()*.

1.11.14 PRAW 2.1.15

- **[FEATURE]** Add `save` OAuth scope to *save()* and *unsave()*.
- **[BUGFIX]** Fix Google AppEngine bug with `platform.platform`.
- **[REDDIT]** Using *get_flair()* now requires moderator access. See this /r/redditdev thread
- **[CHANGE]** Increase the dependency of `update_checker` to 0.9 or later.

1.11.15 PRAW 2.1.14

- **[CHANGE]** Increase the dependency of `six` to 1.4 or later.

1.11.16 PRAW 2.1.13

- **[FEATURE]** Support building wheel binary distributions.
- **[FEATURE]** *get_submission()* and *from_url()* now supports url parameters. Both included within the url and explicitly via the "params" argument.
- **[CHANGE]** The dependency on `update_checker` has been increased to >= 0.8.
- **[REDDIT]** Add support for changes to UserLists on reddit.
- **[REDDIT]** Using get_flair_list now requires moderator access. See this /r/redditdev thread
- **[BUGFIX]** Fix configuration parsing for `store_json_result`.
- **[BUGFIX]** Fix duplicate bug in *BoundedSet*.

1.11.17 PRAW 2.1.12

- **[FEATURE]** Add `json_dict` to *RedditContentObject*.
- **[FEATURE]** You can now give configuration settings directly when instantiating a *BaseReddit* object. See the configuration files
- **[BUGFIX]** Fixed a bug that caused an `AttributeError` to be raised when using a deprecated method.

1.11.18 PRAW 2.1.11

- **[FEATURE]** Added *ignore_reports()* and *unignore_reports()* to *Comment* and *Submission*.
- **[BUGFIX]** The `history` scope is not required for *get_comments()*, *get_overview()* and *get_submitted()* despite the official reddit documentation saying so. Redditors may choose to make their voting record public, in which case no authentication is required for *get_disliked()* or *get_liked()*. The `history` scope requirement for the above-mentioned methods has been removed.

1.11.19 PRAW 2.1.10

- **[FEATURE]** Add *get_new_subreddits()* to return the newest subreddits.
- **[FEATURE]** Add the arguments `save` and `send_replies` to *submit()*.
- **[FEATURE]** Create and add `history` scope to *get_comments()*, *get_disliked()*, *get_liked()*, *get_overview()*, *get_submitted()*, *get_hidden()* and *get_saved()*.

1.11.20 PRAW 2.1.9

- **[FEATURE]** `mark_as_nsfw()` and `unmark_as_nsfw()` can now be used if the currently authenticated user is the author of the Submission.
- **[FEATURE]** *get_contributors()* can now be used for accessing the contributor list of protected/private subreddits without requiring moderator access. See issue issue 246.
- **[BUGFIX]** Fixed *Comment* erroneously having the methods `mark_as_nsfw` and `unmark_as_nsfw`, despite comments not being able to be marked as NSFW.
- **[REDDIT]** Update *get_subreddit_recommendations()* to handle changed returned data format.

1.11.21 PRAW 2.1.8

- **[FEATURE]** Add *get_subreddit_recommendations()* to get a recommendation of subreddits based on a list of provided subreddits.
- **[FEATURE]** *Subreddit* now has an `__repr__` method. So it's now possible to identify what subreddit the object represents from the human readable representation of the object.
- **[FEATURE]** Add *praw.__init__.UnauthenticatedReddit.get_rising()* that returns the rising listing of the front page in the context of the currently logged-in user (if any).

1.11.22 PRAW 2.1.7

- **[FEATURE]** Add methods *set_contest_mode()* and *unset_contest_mode()* to *Submission*, for (un)setting of contest modes. See this Reddit post for information about contest mode.
- **[FEATURE]** Move methods *get_liked()* and *get_liked()* to *Redditor* from *LoggedInRedditor*. Redditors can make their likes and dislikes public. Having *get_liked()* and *get_liked()* on *Redditor* allows PRAW to access this info.
- **[FEATURE]** The `has_fetched` attribute has been added to all objects save *Reddit*, see the lazy loading page in PRAW's documentation for more details.
- **[BUGFIX]** Fixed a bug that caused the `timeout` configuration setting to always be the default 45 irrespective of what it was set to in `praw.ini`.

1.11.23 PRAW 2.1.6

- **[BUGFIX]** PRAW automatically retries failed requests to reddit if the error is likely to be a temporary one. This resulted in spamming reddit if the error occurred after content had been saved to reddit's database. Therefore the following methods will no longer retry failed request *upload_image()*, *send_message()*, *submit()*, `send_feedback()`, *reply()* and *add_comment()*. Additionally *request_json()* now has the `retry_on_error` argument, which if set to `True` will prevent retries of the request if it fails.

1.11.24 PRAW 2.1.5

- **[FEATURE]** *select_flair()* method added, can be used to change your flair without moderator access on subreddits that allow it.
- **[FEATURE]** Add *sticky()* and *unsticky()* to sticky and unsticky a submission to the top of a subreddit.
- **[FEATURE]** Add arguments syntax and period to *search()*.
- **[FEATURE]** PRAW will now try to use the http_proxy environment variable for proxy settings, if no proxy is set in the configuration file.
- **[BUGFIX]** *get_stylesheet()* erroneously required moderator access. It now just requires that the authenticated user has access to the subreddit.
- **[BUGFIX]** Fix bug that prevented the usage of *search()* when called from *Subreddit*.

1.11.25 PRAW 2.1.4

- **[FEATURE]** *get_mod_mail()* can now be used to get moderator mail from individual subreddits, instead of all moderated subreddits, just like *get_mod_queue()*.
- **[FEATURE]** Added *get_mentions()* which is a *get_content()* generator for username mentions. Only usable if the authenticated user has gold.
- **[BUGFIX]** Fixed an error in *get_mod_queue()*, *get_reports()*, *get_spam()* and *get_unmoderated()* when calling them from *Reddit* without giving the subreddit argument explicitly.
- **[REDDIT]** New fields `public_traffic` added to *set_settings()* as per the upstream change.

1.11.26 PRAW 2.1.3

- **[FEATURE]** Added *UnauthenticatedReddit.get_random_submission()*.
- **[BUGFIX]** Verify that `sys.stdin` has `closed` attribute before checking if the stream is closed.

1.11.27 PRAW 2.1.2

- **[BUGFIX]** Avoid occasionally processing duplicates in *comment_stream()*.
- **[CHANGE]** *comment_stream()* yields comments in a consitent order (oldest to newest).
- **[FEATURE]** Support fetching submission listings for domains via *get_domain_listing()*.

1.11.28 PRAW 2.1.1

- **[FEATURE]** Added *praw.helpers.comment_stream()* to provide a neverending stream of new comments.
- **[BUGFIX]** Don't cache requests whose responses will result in an exception. This bug was introduced in version 2.1.0.

1.11.29 PRAW 2.1.0

- **[FEATURE]** PRAW now supports proper rate-limiting and shared caching when running multiple processes. See *Concurrent PRAW Instances* for usage information.
- **[CHANGE]** Remove explicit `limit` parameters from functions that utilize *get_content()* but don't alter the limit. This change will result in broken code if the calling code utilizes positional instead of keyword arguments.
- **[CHANGE]** *get_flair()* returns `None` when the redditor does not exist.
- **[CHANGE]** Deprecated `get_all_comments()`. Use *get_comments()* with `all` as the subreddit argument.
- **[CHANGE]** Deprecated `get_my_reddits()`. Use *get_my_subreddits()* instead.
- **[CHANGE]** Deprecated `get_popular_reddits()`. Use *get_popular_subreddits()* instead.
- **[BUGFIX]** Allow editing non-top-level wiki pages fetched using *Subreddit.get_wiki_page()*.
- **[BUGFIX]** Fix a bug in `submit()`. See https://github.com/praw-dev/praw/issues/213.
- **[BUGFIX]** Fix a python 3.3 bug in *upload_image()*. See https://github.com/praw-dev/praw/issues/211.

1.11.30 PRAW 2.0.15

- **[FEATURE]** PRAW can now use a proxy server, see #206. The parameter `http_proxy` (optional) has been added to the configuration file to define a proxy server in the form host:ip or http://login:user@host:ip.

1.11.31 PRAW 2.0.14

- **[BUGFIX]** Prevent potential invalid redirect exception when using *get_wiki_page()*.

1.11.32 PRAW 2.0.13

- **[FEATURE]** Added *get_submissions()* to batch convert fullnames (t3_bas36id) into *Submission* objects.
- **[FEATURE]** Added *get_wiki_banned()* to get a list of wiki banned users.
- **[FEATURE]** Added *add_wiki_ban()* and *remove_wiki_ban()* to manage the list of wiki banned users.
- **[FEATURE]** Added *get_wiki_contributors()* to get a list of wiki contributors.
- **[FEATURE]** Added *add_wiki_contributor()* and *remove_wiki_contributor()* to manage the list of wiki contributors.
- **[FEATURE]** Added *get_wiki_page()* to fetch an individual WikiPage.
- **[FEATURE]** Added *get_wiki_pages()* to get a list of WikiPage objects.
- **[FEATURE]** Wiki pages can be edited through either the *WikiPage.edit()* method of an already existing WikiPage object, or through the *edit_wiki_page()* function. *edit_wiki_page()* is also used to create new wiki pages.
- **[CHANGE]** Deprecated `ban()`, `unban()`, `make_contributor()`, and `make_moderator()` in favor of the consistently named *add_ban()*, *remove_ban()*, *add_contributor()*, and *add_moderator()* respectively.

1.11.33 PRAW 2.0.12

- **[FEATURE]** PRAW can now decode HTML entities, see #186. The parameter `decode_html_entities` (default `False`) has been added to the configuration file to control whether this feature is activated.
- **[FEATURE]** Add *InvalidSubreddit* exception which is raised when attempting to get a listing for a nonexistent subreddit.
- **[FEATURE]** All functions that use the *get_content()* generator function now take `*args, **kwargs`.
- **[BUGFIX]** Requesting user specific data such as *get_unread()* while OAuthenticated as a user, then switching OAuthentication to another user and re-requesting the data within `cache_timeout` would return the cached results matching the previously authenticated user.
- **[BUGFIX]** *friend()* and *unfriend()* used to raise an `AttributeError` when called without user/pswd authentication. It now properly raises *LoginRequired*.

1.11.34 PRAW 2.0.11

- **[FEATURE]** Add the `raise_captcha_exception` argument to `RequireCaptcha` decorator. When `raise_captcha_exception` is `True` (default `False`), PRAW wil not prompt for the captcha information but instead raise a *InvalidCaptcha* exception.
- **[REDDIT]** An upstream change has split new and rising into their own independent listings. Use the new *praw.objects.Subreddit.get_rising()* method instead of the old `get_new_by_rising()` and *get_new()* instead of `get_new_by_date()`.
- **[CHANGE]** The dependency on `update_checker` has been increased from >= 0.4 to >= 0.5.
- **[BUGFIX]** After inviting a moderator invite, the cached set of moderated subreddits would not be updated with the new subreddit. Causing *restrict_access()* to prevent performing moderater actions in the subreddit.

1.11.35 PRAW 2.0.10

- **[FEATURE]** Add *delete_flair()* method to *Subreddit* and *Reddit* objects.

1.11.36 PRAW 2.0.9

- **[FEATURE]** Add parameter `update_user` (default False) to *get_unread()* if it and `unset_has_mail` are both True, then the `user` object in the *Reddit* object will have its `has_mail` attribute set to `False`.
- **[FEATURE]** Add *praw.objects.LoggedInRedditor.get_friends()* and *praw.objects.LoggedInRedditor.get_blocked()*.
- **[FEATURE]** Add the *read* scope to `get_all_comments()` in the *Reddit* object.
- **[FEATURE]** Add the *read* scope to *get_comments()* and the subreddit listings such as *get_new()* in the `Reddit()` and `Subreddit()` object.
- **[BUGFIX]** Fix bug in *MoreComments.comments()*.
- **[CHANGE]** Break *get_friends()* and *get_banned()* until there is an upstream fix to mean that does not require ssl for those endpoints.

1.11.37 PRAW 2.0.8

- **[FEATURE]** Add `unset_has_mail` parameter to *get_unread()*, if it's set to `True`, then it will set `has_mail` for the logged-in user to `False`.

1.11.38 PRAW 2.0.7

- **[REDDIT]** A reddit update broke PRAW's ability to use *login()* if it was authenticated as a logged-in user. This update adds the ability to re-login.
- **[CHANGE]** *get_flair_list()* can now be used when logged-in as a regular user, being logged in as a mod of the subreddit is no longer required.

1.11.39 PRAW 2.0.6

- **[FEATURE]** Add the *get_unmoderated()* method to *Subreddit* and base reddit objects. This returns a listings of submissions that haven't been approved/removed by a moderator.

1.11.40 PRAW 2.0.5

- **[FEATURE]** Add the parameter `gilded_only` to *get_comments()* and `get_all_comments()` methods in *Subreddit* and base reddit objects. If `gilded_only` is set to `True`, then only gilded comments will be returned.
- **[FEATURE]** Add `get_comments()` method to Reddit object. It works like *get_comments()* in Subreddit objects except it takes the subreddit as the first argument.

1.11.41 PRAW 2.0.4

- **[BUGFIX]** Fix python 3 failure within the test suite introduced in 2.0.3.

1.11.42 PRAW 2.0.3

- **[FEATURE]** Add *delete_image()* method to *Subreddit* objects (also callable on the base reddit object with the subreddit as the first argument).
- **[CHANGE]** PRAW now requires version 0.4 of `update_checker`.

1.11.43 PRAW 2.0.2

- **[BUGFIX]** Fixed bug when comparing *MoreComments* classes in Python 3.x.

1.11.44 PRAW 2.0.1

- **[BUGFIX]** Fix bug with `limit=None` in method *replace_more_comments()* in *Submission* object.

1.11.45 PRAW 2.0.0

- **[FEATURE]** Support reddit OAuth2 scopes (passwordless authentication). See *PRAW and OAuth* for usage information.
- **[FEATURE]** Maximize the number of items fetched when explicit limits are set thus reducing the number of requests up to 4x in some cases.
- **[FEATURE]** Add the following API methods to *Subreddit* objects (also callable on the base reddit object with the subreddit as the first argument):
  - *accept_moderator_invite()* – accept a pending moderator invite.
  - *get_mod_log()* – return ModAction objects for each item (run vars(item), to see available attributes).
  - *configure_flair()* – interface to subreddit flair options.
  - *upload_image()* – upload an image for the subreddit header or use in CSS.
- **[FEATURE]** Support 'admin' and *special* distinguishing of items via *distinguish()*.
- **[FEATURE]** Ability to specify max-character limit for object-to-string representations via `output_chars_limit` in `praw.ini`.
- **[CHANGE]** Remove `comments_flat` property of *Submission* objects. The new *praw.helpers.flatten_tree()* can be used to flatten comment trees.
- **[CHANGE]** Remove `all_comments` and `all_comments_flat` properties of Submission objects. The now public method *replace_more_comments()* must now be explicitly called to replace instances of *MoreComments* within the comment tree.
- **[CHANGE]** The `content_id` attribute of *RedditContentObject* has been renamed to *fullname*.
- **[CHANGE]** The `info` base Reddit instance method has been renamed to *get_info()*.
- **[CHANGE]** `get_saved_links` has been renamed to *get_saved()* and moved to the *LoggedInRedditor* (`r.user`) namespace.
- **[CHANGE]** The Subreddit `get_info` method has been renamed to *from_url()* and supports parameters for changing the number of comments to fetch and by what sort method.
- **[CHANGE]** The *get_submission()* method also now supports parameters for changing the number of comments to fetch and by what sort method.

- **[CHANGE]** *mark_as_nsfw()* and *unmark_as_nsfw()* can no longer be used on *Subreddit* objects. Use `update_settings(nsfw=True)` instead.
- **[CHANGE]** Remove depreciated method `compose_message`.
- **[CHANGE]** Refactored and add a number of exception classes (docs, source) This includes the renaming of:
  - `BadCaptcha` to *InvalidCaptcha*.
  - `NonExistantUser` to *InvalidUser*.
- **[CHANGE]** Simplify content-limit handling and remove the following no-longer necessary parameters from `praw.ini`:
  - `comment_limit`
  - `comment_sort`
  - `default_content_limit`
  - `gold_comments_max`
  - `more_comments_max`
  - `regular_comments_max`
- **[CHANGE]** Move the following methods from *LoggedInRedditor* to base reddit object.
  - *get_unread()*
  - *get_inbox()*
  - *get_mod_mail()*
  - *get_sent()*

1.11.46 PRAW 1.0.16

- **[FEATURE]** Add support for /r/random.

1.11.47 PRAW 1.0.15

- **[FEATURE]** Added the functions *Hideable()* and *unhide()* to *Submission*.
- **[FEATURE]** Added function *is_username_available()* to *Reddit*.

1.11.48 PRAW 1.0.14

- **[FEATURE]** Extended functionality to Python 3.3.

1.11.49 PRAW 1.0.13

- **[BUGFIX]** Fixed non-equality bug. Before comparing two PRAW objects with != would always return `True`.
- **[FEATURE]** Added the function `my_contributions` to *LoggedInRedditor*. Use this to find the subreddits where the user is an approved contributor.
- **[CHANGE]** Voting on something will now force the next call to *get_liked()* or *get_disliked()* to re-query from the reddit rather than use the cache.

1.11.50 PRAW 1.0.12

- **[FEATURE]** Support for optional 'prev' values added.

1.11.51 PRAW 1.0.11

- **[FEATURE]** Added *get_top()* to *Reddit*.

1.11.52 PRAW 1.0.10

- **[FEATURE]** Allow for the OS to not be identified when searching for `praw.ini`.

1.11.53 PRAW 1.0.9

- **[FEATURE]** Added the functions *mark_as_nsfw()* and *unmark_as_nsfw()* to *Submission* and *Subreddit*.

1.11.54 PRAW 1.0.8

- **[CHANGE]** Printing a *Submission* to `sys.stdout` will now limit the output length to 80 chars, just like *Comment* does.
- **[FEATURE]** The maximum amount of comments that can be retrieved alongside a submission for gold and regular accounts has been exported to `praw.ini`.
- **[REDDIT]** Checks for login/moderator in *get_moderators()* and *get_flair()* for Subreddit are no longer necessary.
- **[FEATURE]** Added the function *refresh()* to *Submission*, *Subreddit* and *Redditor*. This will make PRAW re-query either reddit or the cache, depending on whether the last call was within `cache_timeout`, for the latest values and update the objects values.
- **[FEATURE]** Added functions *get_liked()*, *get_disliked()* and *get_hidden()* to *LoggedInRedditor* to allow you to get the Things the user has upvoted, downvoted or hidden.
- **[BUGFIX]** Temporary bugfix until prevstyles become optional.
- **[FEATURE]** Added prevstyle to set_stylesheet requests.
- **[BUGFIX]** Putting in `user` or `pswd` to `praw.ini` without values will no longer make it impossible to login.
- **[FEATURE]** You can now have just `user` filled out in `praw.ini` to ease login while remaining safe.

1.11.55 PRAW 1.0.7

- **[REDDIT]** New fields `prev_description_id` and `prev_public_description_id` added to *set_settings()* as per the upstream change.

1.11.56 PRAW 1.0.6

- **[CHANGE]** `compose_message` has been renamed to *send_message()* in *Reddit* and *LoggedInRedditor*. `compose_message` is now depreciated and will be removed around the end of 2012.

1.11.57 PRAW 1.0.5

- **[FEATURE]** `get_popular_reddits()` added to *Reddit*.

1.11.58 PRAW 1.0.4

- **[FEATURE]** Added *get_new()* and *get_controversial()* to *Reddit*.

1.11.59 PRAW 1.0.3

- **[REDDIT]** The logged in / moderator checks for `flair_list` in *Reddit* are no longer needed and have been removed.

1.11.60 PRAW 1.0.2

- **[FEATURE]** `score` property wrapped function have been added to *Comment*.

1.11.61 PRAW 1.0.1

- **[FEATURE]** `require_moderator` decorator now supports multi-reddits.
- **[FEATURE]** Rudimentary logging of the http requests have been implemented.

1.11.62 PRAW 1.0.0

1.12 Code Overview

Here you will find an overview of PRAW's objects and methods, but not the objects attributes which are generated dynamically from reddit's responses and are thus impossible to accurately describe statically. In *Writing a reddit Bot* there is a longer discussion of how to introspect PRAW, which you can use in conjunction with this nice visual overview.

1.12.1 `praw` Package

Python Reddit API Wrapper.

PRAW, an acronym for "Python Reddit API Wrapper", is a python package that allows for simple access to reddit's API. PRAW aims to be as easy to use as possible and is designed to follow all of reddit's API rules. You have to give a useragent, everything else is handled by PRAW so you needn't worry about violating them.

More information about PRAW can be found at https://github.com/praw-dev/praw

class `praw.__init__.`**`AuthenticatedReddit`**(**args*, ***kwargs*)

Bases: *praw.__init__.OAuth2Reddit*, *praw.__init__.UnauthenticatedReddit*

This class adds the methods necessary for authenticating with reddit.

Authentication can either be login based (through *login()*), or OAuth2 based (via *set_access_credentials()*).

You should **not** directly instantiate instances of this class. Use *Reddit* instead.

Initialize an AuthenticatedReddit instance.

accept_moderator_invite(*subreddit*)
Accept a moderator invite to the given subreddit.

Callable upon an instance of Subreddit with no arguments.

Returns The json response from the server.

clear_authentication()
Clear any existing authentication on the reddit object.

This function is implicitly called on *login* and *set_access_credentials*.

delete(*password*, *message=u''*)
Delete the currently authenticated redditor.

WARNING!

This action is IRREVERSIBLE. Use only if you're okay with NEVER accessing this reddit account again.

Parameters

- **password** – password for currently authenticated account
- **message** – optional 'reason for deletion' message.

Returns json response from the server.

edit_wiki_page(*subreddit*, *page*, *content*, *reason=u''*)
Create or edit a wiki page with title *page* for *subreddit*.

Returns The json response from the server.

get_access_information(*code*, *update_session=True*)
Return the access information for an OAuth2 authorization grant.

Parameters

- **code** – the code received in the request from the OAuth2 server
- **update_session** – Update the current session with the retrieved token(s).

Returns A dictionary with the key/value pairs for access_token, refresh_token and scope. The refresh_token value will be done when the OAuth2 grant is not refreshable.

get_flair_choices(*subreddit*, *link=None*)
Return available flair choices and current flair.

Parameters **link** – If link is given, return the flair options for this submission. Not normally given directly, but instead set by calling the flair_choices method for Submission objects. Use the default for the session's user.

Returns A dictionary with 2 keys. 'current' containing current flair settings for the authenticated user and 'choices' containing a list of possible flair choices.

get_friends(***params*)
Return a UserList of Redditors with whom the user is friends.

get_me()
Return a LoggedInRedditor object.

Note: This function is only intended to be used with an 'identity' providing OAuth2 grant.

has_scope(*scope*)
Return True if OAuth2 authorized for the passed in scope(s).

is_logged_in()
Return True when the session is authenticated via username/password.

Username and passwords are provided via *login()*.

is_oauth_session()
Return True when the current session is an OAuth2 session.

login(*username=None*, *password=None*, ***kwargs*)
Login to a reddit site.

DEPRECATED. Will be removed in a future version of PRAW.

https://www.reddit.com/comments/2ujhkr/ https://www.reddit.com/comments/37e2mv/

Look for username first in parameter, then praw.ini and finally if both were empty get it from stdin. Look for password in parameter, then praw.ini (but only if username matches that in praw.ini) and finally if they both are empty get it with getpass. Add the variables `user` (username) and `pswd` (password) to your praw.ini file to allow for auto-login.

A successful login will overwrite any existing authentication.

refresh_access_information(*refresh_token=None*, *update_session=True*)
Return updated access information for an OAuth2 authorization grant.

Parameters

- **refresh_token** – The refresh token used to obtain the updated information. When not provided, use the stored refresh_token.
- **update_session** – Update the session with the returned data.

Returns A dictionary with the key/value pairs for `access_token`, `refresh_token` and `scope`. The `refresh_token` value will be None when the OAuth2 grant is not refreshable. The `scope` value will be a set containing the scopes the tokens are valid for.

select_flair(*item*, *flair_template_id=u''*, *flair_text=u''*)
Select user flair or link flair on subreddits.

This can only be used for assigning your own name flair or link flair on your own submissions. For assigning other's flairs using moderator access, see *set_flair()*.

Parameters

- **item** – A string, Subreddit object (for user flair), or Submission object (for link flair). If `item` is a string it will be treated as the name of a Subreddit.
- **flair_template_id** – The id for the desired flair template. Use the *get_flair_choices()* and *get_flair_choices()* methods to find the ids for the available user and link flair choices.
- **flair_text** – A string containing the custom flair text. Used on subreddits that allow it.

Returns The json response from the server.

set_access_credentials(*scope*, *access_token*, *refresh_token=None*, *update_user=True*)
Set the credentials used for OAuth2 authentication.

Calling this function will overwrite any currently existing access credentials.

Parameters

- **scope** – A set of reddit scopes the tokens provide access to
- **access_token** – the access token of the authentication

- **refresh_token** – the refresh token of the authentication
- **update_user** – Whether or not to set the user attribute for identity scopes

class praw.__init__.**BaseReddit**(*user_agent*, *site_name=None*, *handler=None*, *disable_update_check=False*, ***kwargs*)

Bases: object

A base class that allows access to reddit's API.

You should **not** directly instantiate instances of this class. Use *Reddit* instead.

Initialize our connection with a reddit server.

The user_agent is how your application identifies itself. Read the official API guidelines for user_agents https://github.com/reddit/reddit/wiki/API. Applications using default user_agents such as "Python/urllib" are drastically limited.

site_name allows you to specify which reddit you want to connect to. The installation defaults are reddit.com, if you only need to connect to reddit.com then you can safely ignore this. If you want to connect to another reddit, set site_name to the name of that reddit. This must match with an entry in praw.ini. If site_name is None, then the site name will be looked for in the environment variable REDDIT_SITE. If it is not found there, the default site name reddit matching reddit.com will be used.

disable_update_check allows you to prevent an update check from occurring in spite of the check_for_updates setting in praw.ini.

All additional parameters specified via kwargs will be used to initialize the Config object. This can be used to specify configuration settings during instantiation of the Reddit instance. See https://praw.readthedocs.org/en/latest/pages/configuration_files.html for more details.

RETRY_CODES = [502, 503, 504]

evict(*urls*)

Evict url(s) from the cache.

Parameters **urls** – An iterable containing normalized urls.

Returns The number of items removed from the cache.

get_content(*url*, *params=None*, *limit=0*, *place_holder=None*, *root_field=u'data'*, *thing_field=u'children'*, *after_field=u'after'*, *object_filter=None*, ***kwargs*)

A generator method to return reddit content from a URL.

Starts at the initial url, and fetches content using the *after* JSON data until *limit* entries have been fetched, or the *place_holder* has been reached.

Parameters

- **url** – the url to start fetching content from
- **params** – dictionary containing extra GET data to put in the url
- **limit** – the number of content entries to fetch. If limit <= 0, fetch the default for your account (25 for unauthenticated users). If limit is None, then fetch as many entries as possible (reddit returns at most 100 per request, however, PRAW will automatically make additional requests as necessary).
- **place_holder** (*a string corresponding to a reddit base36 id without prefix, e.g. 'asdfasdf'*) – if not None, the method will fetch *limit* content, stopping if it finds content with *id* equal to *place_holder*. The place_holder item is the last item to be yielded from this generator. Note that the use of *place_holder* is not 100% reliable as the place holder item may no longer exist due to being removed or deleted.

- **root_field** – indicates the field in the json response that holds the data. Most objects use 'data', however some (flairlist) don't have the 'data' object. Use None for the root object.
- **thing_field** – indicates the field under the root_field which contains the list of things. Most objects use 'children'.
- **after_field** – indicates the field which holds the after item element
- **object_filter** – if set to an integer value, fetch content from the corresponding list index in the JSON response. For example the JSON response for submission duplicates is a list of objects, and the object we want to fetch from is at index 1. So we set object_filter=1 to filter out the other useless list elements.

Returns a list of reddit content, of type Subreddit, Comment, Submission or user flair.

request(*url*, *params=None*, *data=None*, *retry_on_error=True*, *method=None*)
Make a HTTP request and return the response.

Parameters

- **url** – the url to grab content from.
- **params** – a dictionary containing the GET data to put in the url
- **data** – a dictionary containing the extra data to submit
- **retry_on_error** – if True retry the request, if it fails, for up to 3 attempts
- **method** – The HTTP method to use in the request.

Returns The HTTP response.

request_json(*url*, *params=None*, *data=None*, *as_objects=True*, *retry_on_error=True*, *method=None*)
Get the JSON processed from a page.

Parameters

- **url** – the url to grab content from.
- **params** – a dictionary containing the GET data to put in the url
- **data** – a dictionary containing the extra data to submit
- **as_objects** – if True return reddit objects else raw json dict.
- **retry_on_error** – if True retry the request, if it fails, for up to 3 attempts

Returns JSON processed page

update_checked = False

class praw.__init__.**Config**(*site_name*, ***kwargs*)
Bases: object

A class containing the configuration for a reddit site.

Initialize PRAW's configuration.

API_PATHS = {u'authorize': u'api/v1/authorize/', u'comment': u'api/comment/', u'flairconfig': u'api/flairconfig/', u'leav

WWW_PATHS = set([u'authorize'])

short_domain
Return the short domain of the reddit server.

Used to generate the shortlink. For reddit.com the short_domain is redd.it.

static ua_string(*praw_info*)
Return the user-agent string.

The user-agent string contains PRAW version and platform version info.

class praw.__init__.**ModConfigMixin**(**args*, ***kwargs*)
Bases: *praw.__init__.AuthenticatedReddit*

Adds methods requiring the 'modconfig' scope (or mod access).

You should **not** directly instantiate instances of this class. Use *Reddit* instead.

Initialize an AuthenticatedReddit instance.

create_subreddit(*name*, *title*, *description=u''*, *language=u'en'*, *subreddit_type=u'public'*, *content_options=u'any'*, *over_18=False*, *default_set=True*, *show_media=False*, *domain=u''*, *wikimode=u'disabled'*, *captcha=None*, ***kwargs*)
Create a new subreddit.

Returns The json response from the server.

This function may result in a captcha challenge. PRAW will automatically prompt you for a response. See *How can I handle captchas myself?* if you want to manually handle captchas.

delete_image(*subreddit*, *name=None*, *header=False*)
Delete an image from the subreddit.

Parameters

- **name** – The name of the image if removing a CSS image.
- **header** – When true, delete the subreddit header.

Returns The json response from the server.

get_settings(*subreddit*, ***params*)
Return the settings for the given subreddit.

set_settings(*subreddit*, *title*, *public_description=u''*, *description=u''*, *language=u'en'*, *subreddit_type=u'public'*, *content_options=u'any'*, *over_18=False*, *default_set=True*, *show_media=False*, *domain=u''*, *domain_css=False*, *domain_sidebar=False*, *header_hover_text=u''*, *wikimode=u'disabled'*, *wiki_edit_age=30*, *wiki_edit_karma=100*, *submit_link_label=u''*, *submit_text_label=u''*, *exclude_banned_modqueue=False*, *comment_score_hide_mins=0*, *public_traffic=False*, *collapse_deleted_comments=False*, *spam_comments=u'low'*, *spam_links=u'high'*, *spam_selfposts=u'high'*, *submit_text=u''*, *hide_ads=False*, *suggested_comment_sort=u''*, *key_color=u''*, ***kwargs*)
Set the settings for the given subreddit.

Parameters **subreddit** – Must be a subreddit object.

Returns The json response from the server.

set_stylesheet(*subreddit*, *stylesheet*)
Set stylesheet for the given subreddit.

Returns The json response from the server.

update_settings(*subreddit*, ***kwargs*)
Update only the given settings for the given subreddit.

The settings to update must be given by keyword and match one of the parameter names in *set_settings*.

Returns The json response from the server.

upload_image(*subreddit*, *image_path*, *name=None*, *header=False*)
Upload an image to the subreddit.

Parameters

- **image_path** – A path to the jpg or png image you want to upload.
- **name** – The name to provide the image. When None the name will be filename less any extension.
- **header** – When True, upload the image as the subreddit header.

Returns A link to the uploaded image. Raises an exception otherwise.

class praw.__init__.**ModFlairMixin**(**args*, ***kwargs*)
Bases: *praw.__init__.AuthenticatedReddit*

Adds methods requiring the 'modflair' scope (or mod access).

You should **not** directly instantiate instances of this class. Use *Reddit* instead.

Initialize an AuthenticatedReddit instance.

add_flair_template(*subreddit*, *text=u''*, *css_class=u''*, *text_editable=False*, *is_link=False*)
Add a flair template to the given subreddit.

Returns The json response from the server.

clear_flair_templates(*subreddit*, *is_link=False*)
Clear flair templates for the given subreddit.

Returns The json response from the server.

configure_flair(*subreddit*, *flair_enabled=False*, *flair_position=u'right'*, *flair_self_assign=False*, *link_flair_enabled=False*, *link_flair_position=u'left'*, *link_flair_self_assign=False*)
Configure the flair setting for the given subreddit.

Returns The json response from the server.

delete_flair(*subreddit*, *user*)
Delete the flair for the given user on the given subreddit.

Returns The json response from the server.

get_flair_list(*subreddit*, **args*, ***kwargs*)
Return a get_content generator of flair mappings.

Parameters **subreddit** – Either a Subreddit object or the name of the subreddit to return the flair list for.

The additional parameters are passed directly into *get_content()*. Note: the *url*, *root_field*, *thing_field*, and *after_field* parameters cannot be altered.

set_flair(*subreddit*, *item*, *flair_text=u''*, *flair_css_class=u''*)
Set flair for the user in the given subreddit.

item can be a string, Redditor object, or Submission object. If *item* is a string it will be treated as the name of a Redditor.

This method can only be called by a subreddit moderator with flair permissions. To set flair on yourself or your own links use *select_flair()*.

Returns The json response from the server.

set_flair_csv(*subreddit*, *flair_mapping*)
Set flair for a group of users in the given subreddit.

flair_mapping should be a list of dictionaries with the following keys: *user*: the user name, *flair_text*: the flair text for the user (optional), *flair_css_class*: the flair css class for the user (optional)

Returns The json response from the server.

class praw.__init__.**ModLogMixin**(**args*, ***kwargs*)
Bases: *praw.__init__.AuthenticatedReddit*

Adds methods requiring the 'modlog' scope (or mod access).

You should **not** directly instantiate instances of this class. Use *Reddit* instead.

Initialize an AuthenticatedReddit instance.

get_mod_log(*subreddit*, *mod=None*, *action=None*, **args*, ***kwargs*)
Return a get_content generator for moderation log items.

Parameters

- **subreddit** – Either a Subreddit object or the name of the subreddit to return the modlog for.
- **mod** – If given, only return the actions made by this moderator. Both a moderator name or Redditor object can be used here.
- **action** – If given, only return entries for the specified action.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

class praw.__init__.**ModOnlyMixin**(**args*, ***kwargs*)
Bases: *praw.__init__.AuthenticatedReddit*

Adds methods requiring the logged in moderator access.

You should **not** directly instantiate instances of this class. Use *Reddit* instead.

Initialize an AuthenticatedReddit instance.

get_banned(*subreddit*, *user_only=True*, **args*, ***kwargs*)
Return a get_content generator of banned users for the subreddit.

Parameters

- **subreddit** – The subreddit to get the banned user list for.
- **user_only** – When False, the generator yields a dictionary of data associated with the server response for that user. In such cases, the Redditor will be in key 'name' (default: True).

get_contributors(*subreddit*, **args*, ***kwargs*)
Return a get_content generator of contributors for the given subreddit.

If it's a public subreddit, then authentication as a moderator of the subreddit is required. For protected/private subreddits only access is required. See issue #246.

get_edited(*subreddit=u'mod'*, **args*, ***kwargs*)
Return a get_content generator of edited items.

Parameters **subreddit** – Either a Subreddit object or the name of the subreddit to return the edited items for. Defaults to *mod* which includes items for all the subreddits you moderate.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_mod_mail(*subreddit=u'mod'*, **args*, ***kwargs*)
Return a get_content generator for moderator messages.

Parameters **subreddit** – Either a Subreddit object or the name of the subreddit to return the moderator mail from. Defaults to *mod* which includes items for all the subreddits you moderate.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_mod_queue(*subreddit=u'mod'*, **args*, ***kwargs*)
Return a get_content generator for the moderator queue.

Parameters **subreddit** – Either a Subreddit object or the name of the subreddit to return the modqueue for. Defaults to *mod* which includes items for all the subreddits you moderate.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_muted(*subreddit*, *user_only=True*, **args*, ***kwargs*)
Return a get_content generator for modmail-muted users.

Parameters **subreddit** – Either a Subreddit object or the name of a subreddit to get the list of muted users from.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_reports(*subreddit=u'mod'*, **args*, ***kwargs*)
Return a get_content generator of reported items.

Parameters **subreddit** – Either a Subreddit object or the name of the subreddit to return the reported items. Defaults to *mod* which includes items for all the subreddits you moderate.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_spam(*subreddit=u'mod'*, **args*, ***kwargs*)
Return a get_content generator of spam-filtered items.

Parameters **subreddit** – Either a Subreddit object or the name of the subreddit to return the spam-filtered items for. Defaults to *mod* which includes items for all the subreddits you moderate.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_stylesheet(*subreddit*, ***params*)
Return the stylesheet and images for the given subreddit.

get_unmoderated(*subreddit=u'mod'*, **args*, ***kwargs*)
Return a get_content generator of unmoderated submissions.

Parameters **subreddit** – Either a Subreddit object or the name of the subreddit to return the unmoderated submissions for. Defaults to *mod* which includes items for all the subreddits you moderate.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_wiki_banned(*subreddit*, **args*, ***kwargs*)
Return a get_content generator of users banned from the wiki.

get_wiki_contributors (*subreddit*, **args*, ***kwargs*)
Return a get_content generator of wiki contributors.

The returned users are those who have been approved as a wiki contributor by the moderators of the subreddit, Whether or not they've actually contributed to the wiki is irrellevant, their approval as wiki contributors is all that matters.

class praw.__init__.**ModSelfMixin** (**args*, ***kwargs*)
Bases: *praw.__init__.AuthenticatedReddit*

Adds methods pertaining to the 'modself' OAuth scope (or login).

You should **not** directly instantiate instances of this class. Use *Reddit* instead.

Initialize an AuthenticatedReddit instance.

leave_contributor (*subreddit*)
Abdicate approved submitter status in a subreddit. Use with care.

Parameters **subreddit** – The name of the subreddit to leave *status* from.

Returns the json response from the server.

leave_moderator (*subreddit*)
Abdicate moderator status in a subreddit. Use with care.

Parameters **subreddit** – The name of the subreddit to leave *status* from.

Returns the json response from the server.

class praw.__init__.**MultiredditMixin** (**args*, ***kwargs*)
Bases: *praw.__init__.AuthenticatedReddit*

Adds methods pertaining to multireddits.

You should **not** directly instantiate instances of this class. Use *Reddit* instead.

Initialize an AuthenticatedReddit instance.

MULTI_PATH = u'/user/{0}/m/{1}'

copy_multireddit (*from_redditor*, *from_name*, *to_name=None*, **args*, ***kwargs*)
Copy a multireddit.

Parameters

- **from_redditor** – The username or Redditor object for the user who owns the original multireddit
- **from_name** – The name of the multireddit, belonging to from_redditor
- **to_name** – The name to copy the multireddit as. If None, uses the name of the original

The additional parameters are passed directly into *request_json()*

create_multireddit (*name*, *description_md=None*, *icon_name=None*, *key_color=None*, *subreddits=None*, *visibility=None*, *weighting_scheme=None*, *overwrite=False*, **args*, ***kwargs*)
Create a new multireddit.

Parameters

- **name** – The name of the new multireddit.
- **description_md** – Optional description for the multireddit, formatted in markdown.

- **icon_name** – Optional, choose an icon name from this list: `art and design`, `ask`, `books`, `business`, `cars`, `comics`, `cute animals`, `diy`, `entertainment`, `food and drink`, `funny`, `games`, `grooming`, `health`, `life advice`, `military`, `models pinup`, `music`, `news`, `philosophy`, `pictures and gifs`, `science`, `shopping`, `sports`, `style`, `tech`, `travel`, `unusual stories`, `video`, or `None`.
- **key_color** – Optional rgb hex color code of the form *#xxxxxx*.
- **subreddits** – Optional list of subreddit names or Subreddit objects to initialize the Multireddit with. You can always add more later with *add_subreddit()*.
- **visibility** – Choose a privacy setting from this list: `public`, `private`, `hidden`. Defaults to private if blank.
- **weighting_scheme** – Choose a weighting scheme from this list: `classic`, `fresh`. Defaults to classic if blank.
- **overwrite** – Allow for overwriting / updating multireddits. If False, and the multi name already exists, throw 409 error. If True, and the multi name already exists, use the given properties to update that multi. If True, and the multi name does not exist, create it normally.

Returns The newly created Multireddit object.

The additional parameters are passed directly into *request_json()*

delete_multireddit(*name*, **args*, ***kwargs*)
Delete a Multireddit.

Any additional parameters are passed directly into *request()*

edit_multireddit(**args*, ***kwargs*)
Edit a multireddit, or create one if it doesn't already exist.

See *create_multireddit()* for accepted parameters.

get_multireddit(*redditor*, *multi*, **args*, ***kwargs*)
Return a Multireddit object for the author and name specified.

Parameters

- **redditor** – The username or Redditor object of the user who owns the multireddit.
- **multi** – The name of the multireddit to fetch.

The additional parameters are passed directly into the *Multireddit* constructor.

get_multireddits(*redditor*, **args*, ***kwargs*)
Return a list of multireddits belonging to a redditor.

Parameters **redditor** – The username or Redditor object to find multireddits from.

Returns The json response from the server

The additional parameters are passed directly into *request_json()*

If the requested redditor is the current user, all multireddits are visible. Otherwise, only public multireddits are returned.

rename_multireddit(*current_name*, *new_name*, **args*, ***kwargs*)
Rename a Multireddit.

Parameters

- **current_name** – The name of the multireddit to rename

- **new_name** – The new name to assign to this multireddit

The additional parameters are passed directly into *request_json()*

class praw.__init__.**MySubredditsMixin**(**args*, ***kwargs*)
Bases: *praw.__init__.AuthenticatedReddit*

Adds methods requiring the 'mysubreddits' scope (or login).

You should **not** directly instantiate instances of this class. Use *Reddit* instead.

Initialize an AuthenticatedReddit instance.

get_my_contributions(**args*, ***kwargs*)
Return a get_content generator of subreddits.

The Subreddits generated are those where the session's user is a contributor.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_my_moderation(**args*, ***kwargs*)
Return a get_content generator of subreddits.

The Subreddits generated are those where the session's user is a moderator.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_my_multireddits()
Return a list of the authenticated Redditor's Multireddits.

get_my_subreddits(**args*, ***kwargs*)
Return a get_content generator of subreddits.

The subreddits generated are those that hat the session's user is subscribed to.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

class praw.__init__.**OAuth2Reddit**(**args*, ***kwargs*)
Bases: *praw.__init__.BaseReddit*

Provides functionality for obtaining reddit OAuth2 access tokens.

You should **not** directly instantiate instances of this class. Use *Reddit* instead.

Initialize an OAuth2Reddit instance.

get_access_information(*code*)
Return the access information for an OAuth2 authorization grant.

Parameters **code** – the code received in the request from the OAuth2 server

Returns A dictionary with the key/value pairs for `access_token`, `refresh_token` and `scope`. The `refresh_token` value will be None when the OAuth2 grant is not refreshable. The `scope` value will be a set containing the scopes the tokens are valid for.

get_authorize_url(*state*, *scope=u'identity'*, *refreshable=False*)
Return the URL to send the user to for OAuth2 authorization.

Parameters

- **state** – a unique string of your choice that represents this individual client
- **scope** – the reddit scope to ask permissions for. Multiple scopes can be enabled by passing in a container of strings.

- **refreshable** – when True, a permanent "refreshable" token is issued

has_oauth_app_info
Return True when OAuth credentials are associated with the instance.

The necessary credentials are: `client_id`, `client_secret` and `redirect_uri`.

refresh_access_information(*refresh_token*)
Return updated access information for an OAuth2 authorization grant.

Parameters **refresh_token** – the refresh token used to obtain the updated information

Returns A dictionary with the key/value pairs for access_token, refresh_token and scope. The refresh_token value will be done when the OAuth2 grant is not refreshable. The scope value will be a set containing the scopes the tokens are valid for.

set_oauth_app_info(*client_id*, *client_secret*, *redirect_uri*)
Set the app information to use with OAuth2.

This function need only be called if your praw.ini site configuration does not already contain the necessary information.

Go to https://www.reddit.com/prefs/apps/ to discover the appropriate values for your application.

Parameters

- **client_id** – the client_id of your application
- **client_secret** – the client_secret of your application
- **redirect_uri** – the redirect_uri of your application

class praw.__init__.**PrivateMessagesMixin**(**args*, ***kwargs*)
Bases: *praw.__init__.AuthenticatedReddit*

Adds methods requiring the 'privatemessages' scope (or login).

You should **not** directly instantiate instances of this class. Use *Reddit* instead.

Initialize an AuthenticatedReddit instance.

get_comment_replies(**args*, ***kwargs*)
Return a get_content generator for inboxed comment replies.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_inbox(**args*, ***kwargs*)
Return a get_content generator for inbox (messages and comments).

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_mentions(**args*, ***kwargs*)
Return a get_content generator for username mentions.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_message(*message_id*, **args*, ***kwargs*)
Return a Message object corresponding to the given ID.

Parameters **message_id** – The ID or Fullname for a Message

The additional parameters are passed directly into *from_id()* of Message, and subsequently into *request_json()*.

get_messages(**args*, ***kwargs*)
Return a get_content generator for inbox (messages only).

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_post_replies(**args*, ***kwargs*)
Return a get_content generator for inboxed submission replies.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_sent(**args*, ***kwargs*)
Return a get_content generator for sent messages.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_unread(*unset_has_mail=False*, *update_user=False*, **args*, ***kwargs*)
Return a get_content generator for unread messages.

Parameters

- **unset_has_mail** – When True, clear the has_mail flag (orangered) for the user.
- **update_user** – If both *unset_has_mail* and *update user* is True, set the *has_mail* attribute of the logged-in user to False.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

send_message(*recipient*, *subject*, *message*, *from_sr=None*, *captcha=None*, ***kwargs*)
Send a message to a redditor or a subreddit's moderators (mod mail).

Parameters

- **recipient** – A Redditor or Subreddit instance to send a message to. A string can also be used in which case the string is treated as a redditor unless it is prefixed with either '/r/' or '#', in which case it will be treated as a subreddit.
- **subject** – The subject of the message to send.
- **message** – The actual message content.
- **from_sr** – A Subreddit instance or string to send the message from. When provided, messages are sent from the subreddit rather than from the authenticated user. Note that the authenticated user must be a moderator of the subreddit and have mail permissions.

Returns The json response from the server.

This function may result in a captcha challenge. PRAW will automatically prompt you for a response. See *How can I handle captchas myself?* if you want to manually handle captchas.

class praw.__init__.**Reddit**(**args*, ***kwargs*)
Bases: *praw.__init__.ModConfigMixin*, *praw.__init__.ModFlairMixin*, *praw.__init__.ModLogMixin*, *praw.__init__.ModOnlyMixin*, *praw.__init__.ModSelfMixin*, *praw.__init__.MultiredditMixin*, *praw.__init__.MySubredditsMixin*, *praw.__init__.PrivateMessagesMixin*, *praw.__init__.ReportMixin*, *praw.__init__.SubmitMixin*, *praw.__init__.SubscribeMixin*

Provides access to reddit's API.

See *BaseReddit*'s documentation for descriptions of the initialization parameters.

Initialize an AuthenticatedReddit instance.

class praw.__init__.**ReportMixin**(**args*, ***kwargs*)
Bases: *praw.__init__.AuthenticatedReddit*

Adds methods requiring the 'report' scope (or login).

You should **not** directly instantiate instances of this class. Use *Reddit* instead.

Initialize an AuthenticatedReddit instance.

hide(*thing_id*, *_unhide=False*)
Hide up to 50 objects in the context of the logged in user.

Parameters

- **thing_id** – A single fullname or list of fullnames, representing objects which will be hidden.
- **_unhide** – If True, unhide the object(s) instead. Use *unhide()* rather than setting this manually.

Returns The json response from the server.

unhide(*thing_id*)
Unhide up to 50 objects in the context of the logged in user.

Parameters **thing_id** – A single fullname or list of fullnames, representing objects which will be unhidden.

Returns The json response from the server.

class praw.__init__.**SubmitMixin**(**args*, ***kwargs*)
Bases: *praw.__init__.AuthenticatedReddit*

Adds methods requiring the 'submit' scope (or login).

You should **not** directly instantiate instances of this class. Use *Reddit* instead.

Initialize an AuthenticatedReddit instance.

submit(*subreddit*, *title*, *text=None*, *url=None*, *captcha=None*, *save=None*, *send_replies=None*, *resubmit=None*, ***kwargs*)
Submit a new link to the given subreddit.

Accepts either a Subreddit object or a str containing the subreddit's display name.

Parameters

- **resubmit** – If True, submit the link even if it has already been submitted.
- **save** – If True the new Submission will be saved after creation.
- **send_replies** – If True, inbox replies will be received when people comment on the submission. If set to None, the default of True for text posts and False for link posts will be used.

Returns The newly created Submission object if the reddit instance can access it. Otherwise, return the url to the submission.

This function may result in a captcha challenge. PRAW will automatically prompt you for a response. See *How can I handle captchas myself?* if you want to manually handle captchas.

class praw.__init__.**SubscribeMixin**(**args*, ***kwargs*)
Bases: *praw.__init__.AuthenticatedReddit*

Adds methods requiring the 'subscribe' scope (or login).

You should **not** directly instantiate instances of this class. Use *Reddit* instead.

Initialize an AuthenticatedReddit instance.

subscribe(*subreddit*, *unsubscribe=False*)
Subscribe to the given subreddit.

Parameters

- **subreddit** – Either the subreddit name or a subreddit object.
- **unsubscribe** – When True, unsubscribe.

Returns The json response from the server.

unsubscribe(*subreddit*)
Unsubscribe from the given subreddit.

Parameters **subreddit** – Either the subreddit name or a subreddit object.

Returns The json response from the server.

class praw.__init__.**UnauthenticatedReddit**(**args*, ***kwargs*)
Bases: *praw.__init__.BaseReddit*

This mixin provides bindings for basic functions of reddit's API.

None of these functions require authenticated access to reddit's API.

You should **not** directly instantiate instances of this class. Use *Reddit* instead.

Initialize an UnauthenticatedReddit instance.

create_redditor(*user_name*, *password*, *email=u''*)
Register a new user.

Returns The json response from the server.

default_subreddits(**args*, ***kwargs*)
Return a get_content generator for the default subreddits.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_comments(*subreddit*, *gilded_only=False*, **args*, ***kwargs*)
Return a get_content generator for comments in the given subreddit.

Parameters **gilded_only** – If True only return gilded comments.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_controversial(**args*, ***kwargs*)
Return a get_content generator for controversial submissions.

Corresponds to submissions provided by `https://www.reddit.com/controversial/` for the session.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_domain_listing(*domain*, *sort=u'hot'*, *period=None*, **args*, ***kwargs*)
Return a get_content generator for submissions by domain.

Corresponds to the submissions provided by `https://www.reddit.com/domain/{domain}`.

Parameters

- **domain** – The domain to generate a submission listing for.
- **sort** – When provided must be one of 'hot', 'new', 'rising', 'controversial, or 'top'. Defaults to 'hot'.
- **period** – When sort is either 'controversial', or 'top' the period can be either None (for account default), 'all', 'year', 'month', 'week', 'day', or 'hour'.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_flair(*subreddit*, *redditor*, ***params*)
Return the flair for a user on the given subreddit.

Parameters

- **subreddit** – Can be either a Subreddit object or the name of a subreddit.
- **redditor** – Can be either a Redditor object or the name of a redditor.

Returns None if the user doesn't exist, otherwise a dictionary containing the keys *flair_css_class*, *flair_text*, and *user*.

get_front_page(**args*, ***kwargs*)
Return a get_content generator for the front page submissions.

Corresponds to the submissions provided by `https://www.reddit.com/` for the session.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_info(*url=None*, *thing_id=None*, *limit=None*)
Look up existing items by thing_id (fullname) or url.

Parameters

- **url** – The url to lookup.
- **thing_id** – A single thing_id, or a list of thing_ids. A thing_id can be any one of Comment (`t1_`), Link (`t3_`), or Subreddit (`t5_`) to lookup by fullname.
- **limit** – The maximum number of Submissions to return when looking up by url. When None, uses account default settings.

Returns When a single `thing_id` is provided, return the corresponding thing object, or `None` if not found. When a list of `thing_id``s or a ``url` is provided return a list of thing objects (up to `limit`). `None` is returned if any one of the thing_ids or the URL is invalid.

get_moderators(*subreddit*, ***kwargs*)
Return the list of moderators for the given subreddit.

get_new(**args*, ***kwargs*)
Return a get_content generator for new submissions.

Corresponds to the submissions provided by `https://www.reddit.com/new/` for the session.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_new_subreddits(**args*, ***kwargs*)
Return a get_content generator for the newest subreddits.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_popular_subreddits(**args*, ***kwargs*)
Return a get_content generator for the most active subreddits.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_random_submission(*subreddit=u'all'*)
Return a random Submission object.

Parameters **subreddit** – Limit the submission to the specified subreddit(s). Default: all

get_random_subreddit(*nsfw=False*)
Return a random Subreddit object.

Parameters **nsfw** – When true, return a random NSFW Subreddit object. Calling in this manner will set the 'over18' cookie for the duration of the PRAW session.

get_redditor(*user_name*, **args*, ***kwargs*)
Return a Redditor instance for the user_name specified.

The additional parameters are passed directly into the *Redditor* constructor.

get_rising(**args*, ***kwargs*)
Return a get_content generator for rising submissions.

Corresponds to the submissions provided by `https://www.reddit.com/rising/` for the session.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_sticky(*subreddit*, *bottom=False*)
Return a Submission object for the sticky of the subreddit.

Parameters **bottom** – Get the top or bottom sticky. If the subreddit has only a single sticky, it is considered the top one.

get_submission(*url=None*, *submission_id=None*, *comment_limit=0*, *comment_sort=None*, *params=None*)
Return a Submission object for the given url or submission_id.

Parameters

- **comment_limit** – The desired number of comments to fetch. If <= 0 fetch the default number for the session's user. If None, fetch the maximum possible.
- **comment_sort** – The sort order for retrieved comments. When None use the default for the session's user.
- **params** – Dictionary containing extra GET data to put in the url.

get_submissions(*fullnames*, **args*, ***kwargs*)
Generate Submission objects for each item provided in *fullnames*.

A submission fullname looks like *t3_<base36_id>*. Submissions are yielded in the same order they appear in *fullnames*.

Up to 100 items are batched at a time – this happens transparently.

The additional parameters are passed directly into *get_content()*. Note: the *url* and *limit* parameters cannot be altered.

get_subreddit(*subreddit_name*, **args*, ***kwargs*)
Return a Subreddit object for the subreddit_name specified.

The additional parameters are passed directly into the *Subreddit* constructor.

get_subreddit_recommendations(*subreddits*, *omit=None*)
Return a list of recommended subreddits as Subreddit objects.

Subreddits with activity less than a certain threshold, will not have any recommendations due to lack of data.

Parameters

- **subreddits** – A list of subreddits (either names or Subreddit objects) to base the recommendations on.
- **omit** – A list of subreddits (either names or Subreddit objects) that will be filtered out of the result.

get_top(**args*, ***kwargs*)
Return a get_content generator for top submissions.

Corresponds to the submissions provided by `https://www.reddit.com/top/` for the session.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_traffic(*subreddit*)
Return the json dictionary containing traffic stats for a subreddit.

Parameters **subreddit** – The subreddit whose /about/traffic page we will collect.

get_wiki_page(*subreddit*, *page*)
Return a WikiPage object for the subreddit and page provided.

get_wiki_pages(*subreddit*)
Return a list of WikiPage objects for the subreddit.

is_username_available(*username*)
Return True if username is valid and available, otherwise False.

search(*query*, *subreddit=None*, *sort=None*, *syntax=None*, *period=None*, **args*, ***kwargs*)
Return a generator for submissions that match the search query.

Parameters

- **query** – The query string to search for. If query is a URL only submissions which link to that URL will be returned.
- **subreddit** – Limit search results to the subreddit if provided.
- **sort** – The sort order of the results.
- **syntax** – The syntax of the search query.
- **period** – The time period of the results.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

See https://www.reddit.com/wiki/search for more information on how to build a search query.

search_reddit_names(*query*)
Return subreddits whose display name contains the query.

1.12.2 objects Module

Contains code about objects such as Submissions, Redditors or Commments.

There are two main groups of objects in this file. The first are objects that correspond to a Thing or part of a Thing as specified in reddit's API overview, https://github.com/reddit/reddit/wiki/API. The second gives functionality that extends over multiple Things. An object that extends from Saveable indicates that it can be saved and unsaved in the context of a logged in user.

class praw.objects.**Comment**(*reddit_session*, *json_dict*)

Bases: *praw.objects.Editable*, *praw.objects.Gildable*, *praw.objects.Inboxable*, *praw.objects.Moderatable*, *praw.objects.Refreshable*, *praw.objects.Reportable*, *praw.objects.Saveable*, *praw.objects.Voteable*

A class that represents a reddit comments.

Construct an instance of the Comment object.

is_root
Return True when the comment is a top level comment.

permalink
Return a permalink to the comment.

replies
Return a list of the comment replies to this comment.

If the comment is not from a submission, *replies()* will always be an empty list unless you call refresh() before calling :meth:`replies() due to a limitation in reddit's API.

submission
Return the Submission object this comment belongs to.

class praw.objects.**Editable**(*reddit_session*, *json_dict=None*, *fetch=True*, *info_url=None*, *underscore_names=None*, *uniq=None*)

Bases: *praw.objects.RedditContentObject*

Interface for Reddit content objects that can be edited and deleted.

Create a new object from the dict of attributes returned by the API.

The fetch parameter specifies whether to retrieve the object's information from the API (only matters when it isn't provided using json_dict).

delete()
Delete this object.

Returns The json response from the server.

edit(*text*)
Replace the body of the object with *text*.

Returns The updated object.

class praw.objects.**Gildable**(*reddit_session*, *json_dict=None*, *fetch=True*, *info_url=None*, *underscore_names=None*, *uniq=None*)

Bases: *praw.objects.RedditContentObject*

Interface for RedditContentObjects that can be gilded.

Create a new object from the dict of attributes returned by the API.

The fetch parameter specifies whether to retrieve the object's information from the API (only matters when it isn't provided using json_dict).

gild(*months=None*)
Gild the Redditor or author of the content.

Parameters **months** – Specifies the number of months to gild. This parameter is Only valid when the instance called upon is of type Redditor. When not provided, the value defaults to 1.

Returns True on success, otherwise raises an exception.

class praw.objects.**Hideable**(*reddit_session*, *json_dict=None*, *fetch=True*, *info_url=None*, *underscore_names=None*, *uniq=None*)

Bases: *praw.objects.RedditContentObject*

Interface for objects that can be hidden.

Create a new object from the dict of attributes returned by the API.

The fetch parameter specifies whether to retrieve the object's information from the API (only matters when it isn't provided using json_dict).

hide(*_unhide=False*)

Hide object in the context of the logged in user.

Parameters **_unhide** – If True, unhide the item instead. Use *unhide()* instead of setting this manually.

Returns The json response from the server.

unhide()

Unhide object in the context of the logged in user.

Returns The json response from the server.

class praw.objects.**Inboxable**(*reddit_session*, *json_dict=None*, *fetch=True*, *info_url=None*, *underscore_names=None*, *uniq=None*)

Bases: *praw.objects.RedditContentObject*

Interface for objects that appear in the inbox (orangereds).

Create a new object from the dict of attributes returned by the API.

The fetch parameter specifies whether to retrieve the object's information from the API (only matters when it isn't provided using json_dict).

mark_as_read()

Mark object as read.

Returns The json response from the server.

mark_as_unread()

Mark object as unread.

Returns The json response from the server.

reply(*text*)

Reply to object with the specified text.

Returns A Comment object for the newly created comment (reply).

class praw.objects.**LoggedInRedditor**(*reddit_session*, *user_name=None*, *json_dict=None*, *fetch=False*, ***kwargs*)

Bases: *praw.objects.Redditor*

A class representing a currently logged in Redditor.

Construct an instance of the Redditor object.

get_blocked()

Return a UserList of Redditors with whom the user has blocked.

get_cached_moderated_reddits()
Return a cached dictionary of the user's moderated reddits.

This list is used internally. Consider using the *get_my_moderation* function instead.

get_friends(***params*)
Return a UserList of Redditors with whom the user is friends.

This method has been moved to `praw.AuthenticatedReddit`.

get_hidden(*sort=u'new'*, *time=u'all'*, **args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

Parameters

- **sort** – Specify the sort order of the results if applicable (one of `'hot'`, `'new'`, `'top'`, `'controversial'`).
- **time** – Specify the time-period to return submissions if applicable (one of `'hour'`, `'day'`, `'week'`, `'month'`, `'year'`, `'all'`).

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_multireddit(**args*, ***kwargs*)
Return a Multireddit object for the author and name specified.

See *MultiredditMixin.get_multireddit()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_multireddits(**args*, ***kwargs*)
Return a list of multireddits belonging to a redditor.

See *MultiredditMixin.get_multireddits()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_saved(*sort=u'new'*, *time=u'all'*, **args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

Parameters

- **sort** – Specify the sort order of the results if applicable (one of `'hot'`, `'new'`, `'top'`, `'controversial'`).
- **time** – Specify the time-period to return submissions if applicable (one of `'hour'`, `'day'`, `'week'`, `'month'`, `'year'`, `'all'`).

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

class `praw.objects.`**Message**(*reddit_session*, *json_dict*)
Bases: *praw.objects.Inboxable*

A class for private messages.

Construct an instance of the Message object.

collapse()
Collapse a private message or modmail.

static from_id(*reddit_session*, *message_id*, **args*, ***kwargs*)
Request the url for a Message and return a Message object.

Parameters

- **reddit_session** – The session to make the request with.

- **message_id** – The ID of the message to request.

The additional parameters are passed directly into *request_json()*.

mute_modmail_author(*_unmute=False*)
Mute the sender of this modmail message.

Parameters **_unmute** – Unmute the user instead. Please use *unmute_modmail_author()* instead of setting this directly.

uncollapse()
Uncollapse a private message or modmail.

unmute_modmail_author()
Unmute the sender of this modmail message.

class praw.objects.**Messageable**(*reddit_session*, *json_dict=None*, *fetch=True*, *info_url=None*, *underscore_names=None*, *uniq=None*)
Bases: *praw.objects.RedditContentObject*

Interface for RedditContentObjects that can be messaged.

Create a new object from the dict of attributes returned by the API.

The fetch parameter specifies whether to retrieve the object's information from the API (only matters when it isn't provided using json_dict).

send_message(**args*, ***kwargs*)
Send a message to a redditor or a subreddit's moderators (mod mail).

See *PrivateMessagesMixin.send_message()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

class praw.objects.**ModAction**(*reddit_session*, *json_dict=None*, *fetch=False*)
Bases: *praw.objects.RedditContentObject*

A moderator action.

Construct an instance of the ModAction object.

class praw.objects.**Moderatable**(*reddit_session*, *json_dict=None*, *fetch=True*, *info_url=None*, *underscore_names=None*, *uniq=None*)
Bases: *praw.objects.RedditContentObject*

Interface for Reddit content objects that have can be moderated.

Create a new object from the dict of attributes returned by the API.

The fetch parameter specifies whether to retrieve the object's information from the API (only matters when it isn't provided using json_dict).

approve()
Approve object.

This reverts a removal, resets the report counter, marks it with a green check mark (only visible to other moderators) on the website view and sets the approved_by attribute to the logged in user.

Returns The json response from the server.

distinguish(*as_made_by=u'mod'*)
Distinguish object as made by mod, admin or special.

Distinguished objects have a different author color. With Reddit Enhancement Suite it is the background color that changes.

Returns The json response from the server.

ignore_reports()
Ignore future reports on this object.

This prevents future reports from causing notifications or appearing in the various moderation listing. The report count will still increment.

remove(*spam=False*)
Remove object. This is the moderator version of delete.

The object is removed from the subreddit listings and placed into the spam listing. If spam is set to True, then the automatic spam filter will try to remove objects with similar attributes in the future.

Returns The json response from the server.

undistinguish()
Remove mod, admin or special distinguishing on object.

Returns The json response from the server.

unignore_reports()
Remove ignoring of future reports on this object.

Undoes 'ignore_reports'. Future reports will now cause notifications and appear in the various moderation listings.

class praw.objects.**MoreComments**(*reddit_session*, *json_dict*)
Bases: *praw.objects.RedditContentObject*

A class indicating there are more comments.

Construct an instance of the MoreComment object.

comments(*update=True*)
Fetch and return the comments for a single MoreComments object.

class praw.objects.**Multireddit**(*reddit_session*, *author=None*, *name=None*, *json_dict=None*, *fetch=False*, ***kwargs*)
Bases: *praw.objects.Refreshable*

A class for users' Multireddits.

Construct an instance of the Multireddit object.

add_subreddit(*subreddit*, *_delete=False*, **args*, ***kwargs*)
Add a subreddit to the multireddit.

Parameters **subreddit** – The subreddit name or Subreddit object to add

The additional parameters are passed directly into *request_json()*.

copy(*to_name*)
Copy this multireddit.

Convenience function that utilizes *MultiredditMixin.copy_multireddit()* populating both the *from_redditor* and *from_name* parameters.

delete()
Delete this multireddit.

Convenience function that utilizes *MultiredditMixin.delete_multireddit()* populating the *name* parameter.

edit(**args*, ***kwargs*)
Edit this multireddit.

Convenience function that utilizes *MultiredditMixin.edit_multireddit()* populating the *name* parameter.

classmethod from_api_response(*reddit_session*, *json_dict*)
Return an instance of the appropriate class from the json dict.

get_controversial(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_controversial_from_all(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_controversial_from_day(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_controversial_from_hour(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_controversial_from_month(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_controversial_from_week(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_controversial_from_year(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_hot(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_new(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_rising(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_top(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_top_from_all(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_top_from_day(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_top_from_hour(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_top_from_month(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_top_from_week(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_top_from_year(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

remove_subreddit(*subreddit*, **args*, ***kwargs*)
Remove a subreddit from the user's multireddit.

rename(*new_name*, **args*, ***kwargs*)
Rename this multireddit.

This function is a handy shortcut to `rename_multireddit()` of the reddit_session.

class praw.objects.**PRAWListing**(*reddit_session*, *json_dict=None*, *fetch=False*)
Bases: *praw.objects.RedditContentObject*

An abstract class to coerce a listing into RedditContentObjects.

Construct an instance of the PRAWListing object.

CHILD_ATTRIBUTE = None

class praw.objects.**RedditContentObject**(*reddit_session*, *json_dict=None*, *fetch=True*, *info_url=None*, *underscore_names=None*, *uniq=None*)
Bases: object

Base class that represents actual reddit objects.

Create a new object from the dict of attributes returned by the API.

The fetch parameter specifies whether to retrieve the object's information from the API (only matters when it isn't provided using json_dict).

classmethod from_api_response(*reddit_session*, *json_dict*)
Return an instance of the appropriate class from the json_dict.

fullname
Return the object's fullname.

A fullname is an object's kind mapping like *t3* followed by an underscore and the object's base36 id, e.g., *t1_c5s96e0*.

has_fetched
Return whether the object has been fully fetched from reddit.

class praw.objects.**Redditor**(*reddit_session*, *user_name=None*, *json_dict=None*, *fetch=False*, ***kwargs*)
Bases: *praw.objects.Gildable*, *praw.objects.Messageable*, *praw.objects.Refreshable*

A class representing the users of reddit.

Construct an instance of the Redditor object.

friend(*note=None*, *_unfriend=False*)
Friend the user.

Parameters

- **note** – A personal note about the user. Requires reddit Gold.
- **_unfriend** – Unfriend the user. Please use *unfriend()* instead of setting this parameter manually.

Returns The json response from the server.

get_comments(*sort=u'new'*, *time=u'all'*, **args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

Parameters

- **sort** – Specify the sort order of the results if applicable (one of 'hot', 'new', 'top', 'controversial').
- **time** – Specify the time-period to return submissions if applicable (one of 'hour', 'day', 'week', 'month', 'year', 'all').

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_disliked(**args*, ***kwargs*)
Return a listing of the Submissions the user has downvoted.

This method points to *get_downvoted()*, as the "disliked" name is being phased out.

get_downvoted(**args*, ***kwargs*)
Return a listing of the Submissions the user has downvoted.

Returns get_content generator of Submission items.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

As a default, this listing is only accessible by the user. Thereby requiring either user/pswd authentication or OAuth authentication with the 'history' scope. Users may choose to make their voting record public by changing a user preference. In this case, no authentication will be needed to access this listing.

get_friend_info()
Return information about this friend, including personal notes.

The personal note can be added or overwritten with :meth:friend, but only if the user has reddit Gold.

Returns The json response from the server.

get_liked(**args*, ***kwargs*)
Return a listing of the Submissions the user has upvoted.

This method points to *get_upvoted()*, as the "liked" name is being phased out.

get_multireddit(**args*, ***kwargs*)
Return a Multireddit object for the author and name specified.

See *MultiredditMixin.get_multireddit()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_multireddits(**args*, ***kwargs*)
Return a list of multireddits belonging to a redditor.

See *MultiredditMixin.get_multireddits()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_overview(*sort=u'new'*, *time=u'all'*, **args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

Parameters

- **sort** – Specify the sort order of the results if applicable (one of `'hot'`, `'new'`, `'top'`, `'controversial'`).
- **time** – Specify the time-period to return submissions if applicable (one of `'hour'`, `'day'`, `'week'`, `'month'`, `'year'`, `'all'`).

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_submitted(*sort=u'new'*, *time=u'all'*, **args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

Parameters

- **sort** – Specify the sort order of the results if applicable (one of `'hot'`, `'new'`, `'top'`, `'controversial'`).
- **time** – Specify the time-period to return submissions if applicable (one of `'hour'`, `'day'`, `'week'`, `'month'`, `'year'`, `'all'`).

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_upvoted(**args*, ***kwargs*)
Return a listing of the Submissions the user has upvoted.

Returns get_content generator of Submission items.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

As a default, this listing is only accessible by the user. Thereby requirering either user/pswd authentication or OAuth authentication with the 'history' scope. Users may choose to make their voting record public by changing a user preference. In this case, no authentication will be needed to access this listing.

mark_as_read(*messages*, *unread=False*)
Mark message(s) as read or unread.

Returns The json response from the server.

unfriend()
Unfriend the user.

Returns The json response from the server.

class praw.objects.**Refreshable**(*reddit_session*, *json_dict=None*, *fetch=True*, *info_url=None*, *underscore_names=None*, *uniq=None*)
Bases: *praw.objects.RedditContentObject*

Interface for objects that can be refreshed.

Create a new object from the dict of attributes returned by the API.

The fetch parameter specifies whether to retrieve the object's information from the API (only matters when it isn't provided using json_dict).

refresh()
Re-query to update object with latest values. Return the object.

Any listing, such as the submissions on a subreddits top page, will automatically be refreshed serverside. Refreshing a submission will also refresh all its comments.

In the rare case of a submissions's comment[0] being deleted or removed in between its original retrieval and refresh, or inconsistencies between different endpoints resulting in this, an IndexError will be thrown.

class praw.objects.**Reportable**(*reddit_session*, *json_dict=None*, *fetch=True*, *info_url=None*, *underscore_names=None*, *uniq=None*)
Bases: *praw.objects.RedditContentObject*

Interface for RedditContentObjects that can be reported.

Create a new object from the dict of attributes returned by the API.

The fetch parameter specifies whether to retrieve the object's information from the API (only matters when it isn't provided using json_dict).

report(*reason=None*)
Report this object to the moderators.

Parameters **reason** – The user-supplied reason for reporting a comment or submission. Default: None (blank reason)

Returns The json response from the server.

class praw.objects.**Saveable**(*reddit_session*, *json_dict=None*, *fetch=True*, *info_url=None*, *underscore_names=None*, *uniq=None*)
Bases: *praw.objects.RedditContentObject*

Interface for RedditContentObjects that can be saved.

Create a new object from the dict of attributes returned by the API.

The fetch parameter specifies whether to retrieve the object's information from the API (only matters when it isn't provided using json_dict).

save(*unsave=False*)
Save the object.

Returns The json response from the server.

unsave()
Unsave the object.

Returns The json response from the server.

class praw.objects.**Submission**(*reddit_session*, *json_dict*)
Bases: *praw.objects.Editable*, *praw.objects.Gildable*, *praw.objects.Hideable*, *praw.objects.Moderatable*, *praw.objects.Refreshable*, *praw.objects.Reportable*, *praw.objects.Saveable*, *praw.objects.Voteable*

A class for submissions to reddit.

Construct an instance of the Subreddit object.

add_comment(*text*)
Comment on the submission using the specified text.

Returns A Comment object for the newly created comment.

comments
Return forest of comments, with top-level comments as tree roots.

May contain instances of MoreComment objects. To easily replace these objects with Comment objects, use the replace_more_comments method then fetch this attribute. Use comment replies to walk down the tree. To get an unnested, flat list of comments from this attribute use helpers.flatten_tree.

static from_id(*reddit_session*, *subreddit_id*)
Return an edit-only submission object based on the id.

static from_json(*json_response*)
Return a submission object from the json response.

static from_url(*reddit_session*, *url*, *comment_limit=0*, *comment_sort=None*, *comments_only=False*, *params=None*)
Request the url and return a Submission object.

Parameters

- **reddit_session** – The session to make the request with.
- **url** – The url to build the Submission object from.
- **comment_limit** – The desired number of comments to fetch. If <= 0 fetch the default number for the session's user. If None, fetch the maximum possible.
- **comment_sort** – The sort order for retrieved comments. When None use the default for the session's user.
- **comments_only** – Return only the list of comments.
- **params** – dictionary containing extra GET data to put in the url.

get_duplicates(**args*, ***kwargs*)
Return a get_content generator for the submission's duplicates.

Returns get_content generator iterating over Submission objects.

The additional parameters are passed directly into *get_content()*. Note: the *url* and *object_filter* parameters cannot be altered.

get_flair_choices(**args*, ***kwargs*)
Return available link flair choices and current flair.

Convenience function for *get_flair_choices()* populating both the *subreddit* and *link* parameters.

Returns The json response from the server.

lock()
Lock thread.

Requires that the currently authenticated user has the modposts oauth scope or has user/password authentication as a mod of the subreddit.

Returns The json response from the server.

mark_as_nsfw(*unmark_nsfw=False*)
Mark as Not Safe For Work.

Requires that the currently authenticated user is the author of the submission, has the modposts oauth scope or has user/password authentication as a mod of the subreddit.

Returns The json response from the server.

replace_more_comments(*limit=32*, *threshold=1*)
Update the comment tree by replacing instances of MoreComments.

Parameters

- **limit** – The maximum number of MoreComments objects to replace. Each replacement requires 1 API request. Set to None to have no limit, or to 0 to make no extra requests. Default: 32
- **threshold** – The minimum number of children comments a MoreComments object must have in order to be replaced. Default: 1

Returns A list of MoreComments objects that were not replaced.

Note that after making this call, the *comments* attribute of the submission will no longer contain any MoreComments objects. Items that weren't replaced are still removed from the tree, and will be included in the returned list.

select_flair(**args*, ***kwargs*)
Select user flair or link flair on subreddits.

See *AuthenticatedReddit.select_flair()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

set_contest_mode(*state=True*)
Set 'Contest Mode' for the comments of this submission.

Contest mode have the following effects:

- The comment thread will default to being sorted randomly.
- Replies to top-level comments will be hidden behind "[show replies]" buttons.
- Scores will be hidden from non-moderators.
- Scores accessed through the API (mobile apps, bots) will be obscured to "1" for non-moderators.

Source for effects: https://www.reddit.com/159bww/

Returns The json response from the server.

set_flair(**args*, ***kwargs*)
Set flair for this submission.

Convenience function that utilizes *ModFlairMixin.set_flair()* populating both the *subreddit* and *item* parameters.

Returns The json response from the server.

set_suggested_sort(*sort=u'blank'*)
Set 'Suggested Sort' for the comments of the submission.

Comments can be sorted in one of (confidence, top, new, hot, controversial, old, random, qa, blank).

Returns The json response from the server.

short_link
Return a short link to the submission.

The short link points to a page on the short_domain that redirects to the main. For example http://redd.it/eorhm is a short link for https://www.reddit.com/r/announcements/comments/eorhm/reddit_30_less_typing/.

sticky(*bottom=True*)
Sticky a post in its subreddit.

If there is already a stickied post in the designated slot it will be unstickied.

Parameters **bottom** – Set this as the top or bottom sticky. If no top sticky exists, this submission will become the top sticky regardless.

Returns The json response from the server

unlock()
Lock thread.

Requires that the currently authenticated user has the modposts oauth scope or has user/password authentication as a mod of the subreddit.

Returns The json response from the server.

unmark_as_nsfw()
Mark as Safe For Work.

Returns The json response from the server.

unset_contest_mode()
Unset 'Contest Mode' for the comments of this submission.

Contest mode have the following effects:

- The comment thread will default to being sorted randomly.
- Replies to top-level comments will be hidden behind "[show replies]" buttons.
- Scores will be hidden from non-moderators.
- Scores accessed through the API (mobile apps, bots) will be obscured to "1" for non-moderators.

Source for effects: http://www.reddit.com/159bww/

Returns The json response from the server.

unsticky()
Unsticky this post.

Returns The json response from the server

class praw.objects.**Subreddit**(*reddit_session*, *subreddit_name=None*, *json_dict=None*, *fetch=False*, ***kwargs*)

Bases: *praw.objects.Messageable*, *praw.objects.Refreshable*

A class for Subreddits.

Construct an instance of the Subreddit object.

accept_moderator_invite(**args*, ***kwargs*)
Accept a moderator invite to the given subreddit.

See *AuthenticatedReddit.accept_moderator_invite()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

add_ban(*thing*, *user*, ***kwargs*)

add_contributor(*thing*, *user*, ***kwargs*)

add_flair_template(**args*, ***kwargs*)
Add a flair template to the given subreddit.

See *ModFlairMixin.add_flair_template()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

add_moderator(*thing*, *user*, ***kwargs*)

add_mute(*thing*, *user*, ***kwargs*)

add_wiki_ban(*thing*, *user*, ***kwargs*)

add_wiki_contributor(*thing*, *user*, ***kwargs*)

clear_all_flair()
Remove all user flair on this subreddit.

Returns The json response from the server when there is flair to clear, otherwise returns None.

clear_flair_templates(**args*, ***kwargs*)
Clear flair templates for the given subreddit.

See *ModFlairMixin.clear_flair_templates()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

configure_flair(**args*, ***kwargs*)
Configure the flair setting for the given subreddit.

See *ModFlairMixin.configure_flair()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

delete_flair(**args*, ***kwargs*)
Delete the flair for the given user on the given subreddit.

See *ModFlairMixin.delete_flair()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

delete_image(**args*, ***kwargs*)
Delete an image from the subreddit.

See *ModConfigMixin.delete_image()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

edit_wiki_page(**args*, ***kwargs*)
Create or edit a wiki page with title *page* for *subreddit*.

See *AuthenticatedReddit.edit_wiki_page()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_banned(**args*, ***kwargs*)
Return a get_content generator of banned users for the subreddit.

See *ModOnlyMixin.get_banned()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_comments(**args*, ***kwargs*)
Return a get_content generator for comments in the given subreddit.

See *UnauthenticatedReddit.get_comments()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_contributors(**args*, ***kwargs*)
See *ModOnlyMixin.get_contributors()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_controversial(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_controversial_from_all(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_controversial_from_day(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_controversial_from_hour(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_controversial_from_month(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_controversial_from_week(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_controversial_from_year(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_edited(**args*, ***kwargs*)
Return a get_content generator of edited items.

See *ModOnlyMixin.get_edited()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_flair(**args*, ***kwargs*)
Return the flair for a user on the given subreddit.

See *UnauthenticatedReddit.get_flair()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_flair_choices(**args*, ***kwargs*)
Return available flair choices and current flair.

See *AuthenticatedReddit.get_flair_choices()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_flair_list(**args*, ***kwargs*)
Return a get_content generator of flair mappings.

See *ModFlairMixin.get_flair_list()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_hot(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_mod_log(**args*, ***kwargs*)
Return a get_content generator for moderation log items.

See *ModLogMixin.get_mod_log()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_mod_mail(**args*, ***kwargs*)
Return a get_content generator for moderator messages.

See *ModOnlyMixin.get_mod_mail()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_mod_queue(**args*, ***kwargs*)
Return a get_content generator for the moderator queue.

See *ModOnlyMixin.get_mod_queue()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_moderators(**args*, ***kwargs*)
Return the list of moderators for the given subreddit.

See *UnauthenticatedReddit.get_moderators()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_muted(**args*, ***kwargs*)
Return a get_content generator for modmail-muted users.

See *ModOnlyMixin.get_muted()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_new(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_random_submission(**args*, ***kwargs*)
Return a random Submission object.

See *UnauthenticatedReddit.get_random_submission()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_reports(**args*, ***kwargs*)
Return a get_content generator of reported items.

See *ModOnlyMixin.get_reports()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_rising(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_settings(**args*, ***kwargs*)
Return the settings for the given subreddit.

See *ModConfigMixin.get_settings()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_spam(**args*, ***kwargs*)
Return a get_content generator of spam-filtered items.

See *ModOnlyMixin.get_spam()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_sticky(**args*, ***kwargs*)
Return a Submission object for the sticky of the subreddit.

See *UnauthenticatedReddit.get_sticky()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_stylesheet(**args*, ***kwargs*)
Return the stylesheet and images for the given subreddit.

See *ModOnlyMixin.get_stylesheet()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_top(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_top_from_all(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_top_from_day(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_top_from_hour(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_top_from_month(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_top_from_week(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_top_from_year(**args*, ***kwargs*)
Return a get_content generator for some RedditContentObject type.

The additional parameters are passed directly into *get_content()*. Note: the *url* parameter cannot be altered.

get_traffic(**args*, ***kwargs*)
Return the json dictionary containing traffic stats for a subreddit.

See *UnauthenticatedReddit.get_traffic()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_unmoderated(**args*, ***kwargs*)
Return a get_content generator of unmoderated submissions.

See *ModOnlyMixin.get_unmoderated()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_wiki_banned(**args*, ***kwargs*)
Return a get_content generator of users banned from the wiki.

See *ModOnlyMixin.get_wiki_banned()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_wiki_contributors(**args*, ***kwargs*)
Return a get_content generator of wiki contributors.

See *ModOnlyMixin.get_wiki_contributors()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_wiki_page(**args*, ***kwargs*)
Return a WikiPage object for the subreddit and page provided.

See *UnauthenticatedReddit.get_wiki_page()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

get_wiki_pages(**args*, ***kwargs*)
Return a list of WikiPage objects for the subreddit.

See *UnauthenticatedReddit.get_wiki_pages()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

leave_contributor(**args*, ***kwargs*)
Abdicate approved submitter status in a subreddit. Use with care.

See *ModSelfMixin.leave_contributor()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

leave_moderator(**args*, ***kwargs*)
Abdicate moderator status in a subreddit. Use with care.

See *ModSelfMixin.leave_moderator()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

remove_ban(*thing*, *user*, ***kwargs*)

remove_contributor(*thing*, *user*, ***kwargs*)

remove_moderator(*thing*, *user*, ***kwargs*)

remove_mute(*thing*, *user*, ***kwargs*)

remove_wiki_ban(*thing*, *user*, ***kwargs*)

remove_wiki_contributor(*thing*, *user*, ***kwargs*)

search(**args*, ***kwargs*)
Return a generator for submissions that match the search query.

See *UnauthenticatedReddit.search()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

select_flair(**args*, ***kwargs*)
Select user flair or link flair on subreddits.

See *AuthenticatedReddit.select_flair()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

set_flair(**args*, ***kwargs*)
Set flair for the user in the given subreddit.

See *ModFlairMixin.set_flair()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

set_flair_csv(**args*, ***kwargs*)
Set flair for a group of users in the given subreddit.

See *ModFlairMixin.set_flair_csv()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

set_settings(**args*, ***kwargs*)
Set the settings for the given subreddit.

See *ModConfigMixin.set_settings()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

set_stylesheet(**args*, ***kwargs*)
Set stylesheet for the given subreddit.

See *ModConfigMixin.set_stylesheet()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

submit(**args*, ***kwargs*)
Submit a new link to the given subreddit.

See *SubmitMixin.submit()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

subscribe(**args*, ***kwargs*)
Subscribe to the given subreddit.

See *SubscribeMixin.subscribe()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

unsubscribe(**args*, ***kwargs*)
Unsubscribe from the given subreddit.

See *SubscribeMixin.unsubscribe()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

update_settings(**args*, ***kwargs*)
Update only the given settings for the given subreddit.

See *ModConfigMixin.update_settings()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

upload_image(**args*, ***kwargs*)
Upload an image to the subreddit.

See *ModConfigMixin.upload_image()* for complete usage. Note that you should exclude the subreddit parameter when calling this convenience method.

class praw.objects.**UserList**(*reddit_session*, *json_dict=None*, *fetch=False*)
Bases: *praw.objects.PRAWListing*

A list of Redditors. Works just like a regular list.

Construct an instance of the PRAWListing object.

CHILD_ATTRIBUTE = u'children'

class praw.objects.**Voteable**(*reddit_session*, *json_dict=None*, *fetch=True*, *info_url=None*, *underscore_names=None*, *uniq=None*)
Bases: *praw.objects.RedditContentObject*

Interface for RedditContentObjects that can be voted on.

Create a new object from the dict of attributes returned by the API.

The fetch parameter specifies whether to retrieve the object's information from the API (only matters when it isn't provided using json_dict).

clear_vote()
Remove the logged in user's vote on the object.

Running this on an object with no existing vote has no adverse effects.

Note: votes must be cast by humans. That is, API clients proxying a human's action one-for-one are OK, but bots deciding how to vote on content or amplifying a human's vote are not. See the reddit rules for more details on what constitutes vote cheating.

Source for note: http://www.reddit.com/dev/api#POST_api_vote

Returns The json response from the server.

downvote()
Downvote object. If there already is a vote, replace it.

Note: votes must be cast by humans. That is, API clients proxying a human's action one-for-one are OK, but bots deciding how to vote on content or amplifying a human's vote are not. See the reddit rules for more details on what constitutes vote cheating.

Source for note: http://www.reddit.com/dev/api#POST_api_vote

Returns The json response from the server.

upvote()
Upvote object. If there already is a vote, replace it.

Note: votes must be cast by humans. That is, API clients proxying a human's action one-for-one are OK, but bots deciding how to vote on content or amplifying a human's vote are not. See the reddit rules for more details on what constitutes vote cheating.

Source for note: http://www.reddit.com/dev/api#POST_api_vote

Returns The json response from the server.

vote(*direction=0*)
Vote for the given item in the direction specified.

Note: votes must be cast by humans. That is, API clients proxying a human's action one-for-one are OK, but bots deciding how to vote on content or amplifying a human's vote are not. See the reddit rules for more details on what constitutes vote cheating.

Source for note: http://www.reddit.com/dev/api#POST_api_vote

Returns The json response from the server.

class praw.objects.**WikiPage**(*reddit_session*, *subreddit=None*, *page=None*, *json_dict=None*, *fetch=False*, ***kwargs*)
Bases: *praw.objects.Refreshable*

An individual WikiPage object.

Construct an instance of the WikiPage object.

add_editor(*username*, *_delete=False*, **args*, ***kwargs*)
Add an editor to this wiki page.

Parameters

- **username** – The name or Redditor object of the user to add.
- **_delete** – If True, remove the user as an editor instead. Please use *remove_editor()* rather than setting it manually.

Additional parameters are passed into *request_json()*.

edit(**args*, ***kwargs*)
Edit the wiki page.

Convenience function that utilizes *AuthenticatedReddit.edit_wiki_page()* populating both the subreddit and page parameters.

edit_settings(*permlevel*, *listed*, **args*, ***kwargs*)
Edit the settings for this individual wiki page.

Parameters

- **permlevel** – Who can edit this page? (0) use subreddit wiki permissions, (1) only approved wiki contributors for this page may edit (see *add_editor()*), (2) only mods may edit and view
- **listed** – Show this page on the listing? True - Appear in /wiki/pages False - Do not appear in /wiki/pages

Returns The updated settings data.

Additional parameters are passed into request_json().

classmethod from_api_response(*reddit_session*, *json_dict*)
Return an instance of the appropriate class from the json_dict.

get_settings(**args*, ***kwargs*)
Return the settings for this wiki page.

Includes permission level, names of editors, and whether the page is listed on /wiki/pages.

Additional parameters are passed into *request_json()*

remove_editor(*username*, **args*, ***kwargs*)
Remove an editor from this wiki page.

Parameters **username** – The name or Redditor object of the user to remove.

This method points to *add_editor()* with _delete=True.

Additional parameters are are passed to *add_editor()* and subsequently into *request_json()*.

class praw.objects.**WikiPageListing**(*reddit_session*, *json_dict=None*, *fetch=False*)
Bases: *praw.objects.PRAWListing*

A list of WikiPages. Works just like a regular list.

Construct an instance of the PRAWListing object.

CHILD_ATTRIBUTE = u'_tmp'

1.12.3 helpers Module

Helper functions.

The functions here provide functionality that is often needed by programs using PRAW, but which isn't part of reddit's API.

class praw.helpers.**BoundedSet**(*max_items*)
Bases: object

A set with a maximum size that evicts the oldest items when necessary.

This class does not implement the complete set interface.

Construct an instance of the BoundedSet.

add(*item*)
Add an item to the set discarding the oldest item if necessary.

praw.helpers.**comment_stream**(*reddit_session*, *subreddit*, *limit=None*, *verbosity=1*)
Indefinitely yield new comments from the provided subreddit.

Comments are yielded from oldest to newest.

Parameters

- **reddit_session** – The reddit_session to make requests from. In all the examples this is assigned to the variable r.
- **subreddit** – Either a subreddit object, or the name of a subreddit. Use *all* to get the comment stream for all comments made to reddit.
- **limit** – The maximum number of comments to fetch in a single iteration. When None, fetch all available comments (reddit limits this to 1000 (or multiple of 1000 for multi-subreddits). If this number is too small, comments may be missed.
- **verbosity** – A number that controls the amount of output produced to stderr. <= 0: no output; >= 1: output the total number of comments processed and provide the short-term number of comments processed per second; >= 2: output when additional delays are added in order to avoid subsequent unexpected http errors. >= 3: output debugging information regarding the comment stream. (Default: 1)

praw.helpers.**convert_id36_to_numeric_id**(*id36*)
Convert strings representing base36 numbers into an integer.

praw.helpers.**convert_numeric_id_to_id36**(*numeric_id*)
Convert an integer into its base36 string representation.

This method has been cleaned up slightly to improve readability. For more info see:

https://github.com/reddit/reddit/blob/master/r2/r2/lib/utils/_utils.pyx

https://www.reddit.com/r/redditdev/comments/n624n/submission_ids_question/

https://en.wikipedia.org/wiki/Base36

praw.helpers.**flatten_tree**(*tree*, *nested_attr=u'replies'*, *depth_first=False*)
Return a flattened version of the passed in tree.

Parameters

- **nested_attr** – The attribute name that contains the nested items. Defaults to `replies` which is suitable for comments.
- **depth_first** – When true, add to the list in a depth-first manner rather than the default breadth-first manner.

praw.helpers.**normalize_url**(*url*)
Return url after stripping trailing .json and trailing slashes.

praw.helpers.**submission_stream**(*reddit_session*, *subreddit*, *limit=None*, *verbosity=1*)
Indefinitely yield new submissions from the provided subreddit.

Submissions are yielded from oldest to newest.

Parameters

- **reddit_session** – The reddit_session to make requests from. In all the examples this is assigned to the variable `r`.
- **subreddit** – Either a subreddit object, or the name of a subreddit. Use *all* to get the submissions stream for all submissions made to reddit.
- **limit** – The maximum number of submissions to fetch in a single iteration. When None, fetch all available submissions (reddit limits this to 1000 (or multiple of 1000 for multi-subreddits). If this number is too small, submissions may be missed. Since there isn't a limit to the number of submissions that can be retrieved from r/all, the limit will be set to 1000 when limit is None.
- **verbosity** – A number that controls the amount of output produced to stderr. <= 0: no output; >= 1: output the total number of submissions processed and provide the short-term number of submissions processed per second; >= 2: output when additional delays are added in order to avoid subsequent unexpected http errors. >= 3: output debugging information regarding the submission stream. (Default: 1)

praw.helpers.**submissions_between**(*reddit_session*, *subreddit*, *lowest_timestamp=None*, *highest_timestamp=None*, *newest_first=True*, *extra_cloudsearch_fields=None*, *verbosity=1*)
Yield submissions between two timestamps.

If both `highest_timestamp` and `lowest_timestamp` are unspecified, yields all submissions in the `subreddit`.

Submissions are yielded from newest to oldest(like in the "new" queue).

Parameters

- **reddit_session** – The reddit_session to make requests from. In all the examples this is assigned to the variable `r`.

- **subreddit** – Either a subreddit object, or the name of a subreddit. Use *all* to get the submissions stream for all submissions made to reddit.
- **lowest_timestamp** – The lower bound for `created_utc` atributed of submissions. (Default: subreddit's created_utc or 0 when subreddit == "all").
- **highest_timestamp** – The upper bound for `created_utc` attribute of submissions. (Default: current unix time) NOTE: both highest_timestamp and lowest_timestamp are proper unix timestamps(just like `created_utc` attributes)
- **newest_first** – If set to true, yields submissions from newest to oldest. Otherwise yields submissions from oldest to newest
- **extra_cloudsearch_fields** – Allows extra filtering of results by parameters like author, self. Full list is available here: https://www.reddit.com/wiki/search
- **verbosity** – A number that controls the amount of output produced to stderr. <= 0: no output; >= 1: output the total number of submissions processed; >= 2: output debugging information regarding the search queries. (Default: 1)

`praw.helpers.`**valid_redditors**(*redditors*, *sub*)

Return a verified list of valid Redditor instances.

Parameters

- **redditors** – A list comprised of Redditor instances and/or strings that are to be verified as actual redditor accounts.
- **sub** – A Subreddit instance that the authenticated account has flair changing permission on.

Note: Flair will be unset for all valid redditors in *redditors* on the subreddit *mod_sub*.

1.12.4 `errors` Module

Error classes.

Includes two main exceptions: ClientException, when something goes wrong on our end, and APIExeception for when something goes wrong on the server side. A number of classes extend these two main exceptions for more specific exceptions.

exception `praw.errors.`**APIException**(*error_type*, *message*, *field=u''*, *response=None*)

Bases: *praw.errors.PRAWException*

Base exception class for the reddit API error message exceptions.

All exceptions of this type should have their own subclass.

Construct an APIException.

Parameters

- **error_type** – The error type set on reddit's end.
- **message** – The associated message for the error.
- **field** – The input field associated with the error, or ''.
- **response** – The HTTP response that resulted in the exception.

exception `praw.errors.`**AlreadyModerator**(*error_type*, *message*, *field=u''*, *response=None*)

Bases: *praw.errors.APIException*

Used to indicate that a user is already a moderator of a subreddit.

Construct an APIException.

Parameters

- **error_type** – The error type set on reddit's end.
- **message** – The associated message for the error.
- **field** – The input field associated with the error, or ''.
- **response** – The HTTP response that resulted in the exception.

ERROR_TYPE = u'ALREADY_MODERATOR'

exception praw.errors.**AlreadySubmitted**(*error_type*, *message*, *field=u''*, *response=None*)
Bases: *praw.errors.APIException*

An exception to indicate that a URL was previously submitted.

Construct an APIException.

Parameters

- **error_type** – The error type set on reddit's end.
- **message** – The associated message for the error.
- **field** – The input field associated with the error, or ''.
- **response** – The HTTP response that resulted in the exception.

ERROR_TYPE = u'ALREADY_SUB'

exception praw.errors.**BadCSS**(*error_type*, *message*, *field=u''*, *response=None*)
Bases: *praw.errors.APIException*

An exception to indicate bad CSS (such as invalid) was used.

Construct an APIException.

Parameters

- **error_type** – The error type set on reddit's end.
- **message** – The associated message for the error.
- **field** – The input field associated with the error, or ''.
- **response** – The HTTP response that resulted in the exception.

ERROR_TYPE = u'BAD_CSS'

exception praw.errors.**BadCSSName**(*error_type*, *message*, *field=u''*, *response=None*)
Bases: *praw.errors.APIException*

An exception to indicate a bad CSS name (such as invalid) was used.

Construct an APIException.

Parameters

- **error_type** – The error type set on reddit's end.
- **message** – The associated message for the error.
- **field** – The input field associated with the error, or ''.
- **response** – The HTTP response that resulted in the exception.

ERROR_TYPE = u'BAD_CSS_NAME'

exception praw.errors.**BadUsername**(*error_type*, *message*, *field=u''*, *response=None*)
Bases: *praw.errors.APIException*

An exception to indicate an invalid username was used.

Construct an APIException.

Parameters

- **error_type** – The error type set on reddit's end.
- **message** – The associated message for the error.
- **field** – The input field associated with the error, or ''.
- **response** – The HTTP response that resulted in the exception.

ERROR_TYPE = u'BAD_USERNAME'

exception praw.errors.**ClientException**(*message=None*)
Bases: *praw.errors.PRAWException*

Base exception class for errors that don't involve the remote API.

Construct a ClientException.

Parameters **message** – The error message to display.

exception praw.errors.**ExceptionList**(*errors*)
Bases: *praw.errors.APIException*

Raised when more than one exception occurred.

Construct an ExceptionList.

Parameters **errors** – The list of errors.

exception praw.errors.**Forbidden**(*_raw*, *message=None*)
Bases: *praw.errors.HTTPException*

Raised when the user does not have permission to the entity.

Construct a HTTPException.

Params _raw The internal request library response object. This object is mapped to attribute *_raw* whose format may change at any time.

exception praw.errors.**HTTPException**(*_raw*, *message=None*)
Bases: *praw.errors.PRAWException*

Base class for HTTP related exceptions.

Construct a HTTPException.

Params _raw The internal request library response object. This object is mapped to attribute *_raw* whose format may change at any time.

exception praw.errors.**InsufficientCreddits**(*error_type*, *message*, *field=u''*, *response=None*)
Bases: *praw.errors.APIException*

Raised when there are not enough creddits to complete the action.

Construct an APIException.

Parameters

- **error_type** – The error type set on reddit's end.
- **message** – The associated message for the error.

- **field** – The input field associated with the error, or ‘’.
- **response** – The HTTP response that resulted in the exception.

ERROR_TYPE = u’INSUFFICIENT_CREDDITS’

exception praw.errors.**InvalidCaptcha**(*error_type*, *message*, *field=u’‘*, *response=None*)
Bases: *praw.errors.APIException*

An exception for when an incorrect captcha error is returned.

Construct an APIException.

Parameters

- **error_type** – The error type set on reddit’s end.
- **message** – The associated message for the error.
- **field** – The input field associated with the error, or ‘’.
- **response** – The HTTP response that resulted in the exception.

ERROR_TYPE = u’BAD_CAPTCHA’

exception praw.errors.**InvalidComment**
Bases: *praw.errors.PRAWException*

Indicate that the comment is no longer available on reddit.

ERROR_TYPE = u’DELETED_COMMENT’

exception praw.errors.**InvalidEmails**(*error_type*, *message*, *field=u’‘*, *response=None*)
Bases: *praw.errors.APIException*

An exception for when invalid emails are provided.

Construct an APIException.

Parameters

- **error_type** – The error type set on reddit’s end.
- **message** – The associated message for the error.
- **field** – The input field associated with the error, or ‘’.
- **response** – The HTTP response that resulted in the exception.

ERROR_TYPE = u’BAD_EMAILS’

exception praw.errors.**InvalidFlairTarget**(*error_type*, *message*, *field=u’‘*, *response=None*)
Bases: *praw.errors.APIException*

An exception raised when an invalid user is passed as a flair target.

Construct an APIException.

Parameters

- **error_type** – The error type set on reddit’s end.
- **message** – The associated message for the error.
- **field** – The input field associated with the error, or ‘’.
- **response** – The HTTP response that resulted in the exception.

ERROR_TYPE = u’BAD_FLAIR_TARGET’

exception praw.errors.**InvalidInvite**(*error_type*, *message*, *field=u''*, *response=None*)
Bases: *praw.errors.APIException*

Raised when attempting to accept a nonexistent moderator invite.

Construct an APIException.

Parameters

- **error_type** – The error type set on reddit's end.
- **message** – The associated message for the error.
- **field** – The input field associated with the error, or ''.
- **response** – The HTTP response that resulted in the exception.

ERROR_TYPE = u'NO_INVITE_FOUND'

exception praw.errors.**InvalidSubmission**
Bases: *praw.errors.PRAWException*

Indicates that the submission is no longer available on reddit.

ERROR_TYPE = u'DELETED_LINK'

exception praw.errors.**InvalidSubreddit**
Bases: *praw.errors.PRAWException*

Indicates that an invalid subreddit name was supplied.

ERROR_TYPE = u'SUBREDDIT_NOEXIST'

exception praw.errors.**InvalidUser**(*error_type*, *message*, *field=u''*, *response=None*)
Bases: *praw.errors.APIException*

An exception for when a user doesn't exist.

Construct an APIException.

Parameters

- **error_type** – The error type set on reddit's end.
- **message** – The associated message for the error.
- **field** – The input field associated with the error, or ''.
- **response** – The HTTP response that resulted in the exception.

ERROR_TYPE = u'USER_DOESNT_EXIST'

exception praw.errors.**InvalidUserPass**(*error_type*, *message*, *field=u''*, *response=None*)
Bases: *praw.errors.APIException*

An exception for failed logins.

Construct an APIException.

Parameters

- **error_type** – The error type set on reddit's end.
- **message** – The associated message for the error.
- **field** – The input field associated with the error, or ''.
- **response** – The HTTP response that resulted in the exception.

ERROR_TYPE = u'WRONG_PASSWORD'

exception praw.errors.**LoginOrScopeRequired**(*function*, *scope*, *message=None*)
Bases: *praw.errors.OAuthScopeRequired*, *praw.errors.LoginRequired*

Indicates that either a logged in session or OAuth2 scope is required.

The attribute *scope* will contain the name of the necessary scope.

Construct a LoginOrScopeRequired exception.

Parameters

- **function** – The function that requires authentication.
- **scope** – The scope that is required if not logged in.
- **message** – A custom message to associate with the exception. Default: *function* requires a logged in session or the OAuth2 scope *scope*

exception praw.errors.**LoginRequired**(*function*, *message=None*)
Bases: *praw.errors.ClientException*

Indicates that a logged in session is required.

This exception is raised on a preemptive basis, whereas NotLoggedIn occurs in response to a lack of credentials on a privileged API call.

Construct a LoginRequired exception.

Parameters

- **function** – The function that requires login-based authentication.
- **message** – A custom message to associate with the exception. Default: *function* requires a logged in session

exception praw.errors.**ModeratorOrScopeRequired**(*function*, *scope*)
Bases: *praw.errors.LoginOrScopeRequired*, *praw.errors.ModeratorRequired*

Indicates that a moderator of the sub or OAuth2 scope is required.

The attribute *scope* will contain the name of the necessary scope.

Construct a ModeratorOrScopeRequired exception.

Parameters

- **function** – The function that requires moderator authentication or a moderator scope..
- **scope** – The scope that is required if not logged in with moderator access..

exception praw.errors.**ModeratorRequired**(*function*)
Bases: *praw.errors.LoginRequired*

Indicates that a moderator of the subreddit is required.

Construct a ModeratorRequired exception.

Parameters **function** – The function that requires moderator access.

exception praw.errors.**NotFound**(*_raw*, *message=None*)
Bases: *praw.errors.HTTPException*

Raised when the requested entity is not found.

Construct a HTTPException.

Params _raw The internal request library response object. This object is mapped to attribute *_raw* whose format may change at any time.

exception praw.errors.**NotLoggedIn**(*error_type*, *message*, *field=u''*, *response=None*)
Bases: *praw.errors.APIException*

An exception for when a Reddit user isn't logged in.

Construct an APIException.

Parameters

- **error_type** – The error type set on reddit's end.
- **message** – The associated message for the error.
- **field** – The input field associated with the error, or ''.
- **response** – The HTTP response that resulted in the exception.

ERROR_TYPE = u'USER_REQUIRED'

exception praw.errors.**NotModified**(*response*)
Bases: *praw.errors.APIException*

An exception raised when reddit returns {'error': 304}.

This error indicates that the requested content was not modified and is being requested too frequently. Such an error usually occurs when multiple instances of PRAW are running concurrently or in rapid succession.

Construct an instance of the NotModified exception.

This error does not have an error_type, message, nor field.

exception praw.errors.**OAuthAppRequired**(*message=None*)
Bases: *praw.errors.ClientException*

Raised when an OAuth client cannot be initialized.

This occurs when any one of the OAuth config values are not set.

Construct a ClientException.

Parameters **message** – The error message to display.

exception praw.errors.**OAuthException**(*message*, *url*)
Bases: *praw.errors.PRAWException*

Base exception class for OAuth API calls.

Attribute *message* contains the error message. Attribute *url* contains the url that resulted in the error.

Construct a OAuthException.

Parameters

- **message** – The message associated with the exception.
- **url** – The url that resulted in error.

exception praw.errors.**OAuthInsufficientScope**(*message*, *url*)
Bases: *praw.errors.OAuthException*

Raised when the current OAuth scope is not sufficient for the action.

This indicates the access token is valid, but not for the desired action.

Construct a OAuthException.

Parameters

- **message** – The message associated with the exception.

- **url** – The url that resulted in error.

exception praw.errors.**OAuthInvalidGrant**(*message*, *url*)
Bases: *praw.errors.OAuthException*

Raised when the code to retrieve access information is not valid.

Construct a OAuthException.

Parameters

- **message** – The message associated with the exception.
- **url** – The url that resulted in error.

exception praw.errors.**OAuthInvalidToken**(*message*, *url*)
Bases: *praw.errors.OAuthException*

Raised when the current OAuth access token is not valid.

Construct a OAuthException.

Parameters

- **message** – The message associated with the exception.
- **url** – The url that resulted in error.

exception praw.errors.**OAuthScopeRequired**(*function*, *scope*, *message=None*)
Bases: *praw.errors.ClientException*

Indicates that an OAuth2 scope is required to make the function call.

The attribute *scope* will contain the name of the necessary scope.

Contruct an OAuthScopeRequiredClientException.

Parameters

- **function** – The function that requires a scope.
- **scope** – The scope required for the function.
- **message** – A custom message to associate with the exception. Default: *function* requires the OAuth2 scope *scope*

exception praw.errors.**PRAWException**
Bases: exceptions.Exception

The base PRAW Exception class.

Ideally, this can be caught to handle any exception from PRAW.

exception praw.errors.**RateLimitExceeded**(*error_type*, *message*, *field*, *response*)
Bases: *praw.errors.APIException*

An exception for when something has happened too frequently.

Contains a *sleep_time* attribute for the number of seconds that must transpire prior to the next request.

Construct an instance of the RateLimitExceeded exception.

The parameters match that of *APIException*.

The *sleep_time* attribute is extracted from the response object.

ERROR_TYPE = u'RATELIMIT'

exception praw.errors.**RedirectException**(*request_url*, *response_url*, *message=None*)
Bases: *praw.errors.PRAWException*

Raised when a redirect response occurs that is not expected.

Construct a RedirectException.

Parameters

- **request_url** – The url requested.
- **response_url** – The url being redirected to.
- **message** – A custom message to associate with the exception.

exception praw.errors.**SubredditExists**(*error_type*, *message*, *field=u''*, *response=None*)
Bases: *praw.errors.APIException*

An exception to indicate that a subreddit name is not available.

Construct an APIException.

Parameters

- **error_type** – The error type set on reddit's end.
- **message** – The associated message for the error.
- **field** – The input field associated with the error, or ''.
- **response** – The HTTP response that resulted in the exception.

ERROR_TYPE = u'SUBREDDIT_EXISTS'

exception praw.errors.**UsernameExists**(*error_type*, *message*, *field=u''*, *response=None*)
Bases: *praw.errors.APIException*

An exception to indicate that a username is not available.

Construct an APIException.

Parameters

- **error_type** – The error type set on reddit's end.
- **message** – The associated message for the error.
- **field** – The input field associated with the error, or ''.
- **response** – The HTTP response that resulted in the exception.

ERROR_TYPE = u'USERNAME_TAKEN'

1.12.5 handlers Module

Provides classes that handle request dispatching.

class praw.handlers.**DefaultHandler**
Bases: *praw.handlers.RateLimitHandler*

Extends the RateLimitHandler to add thread-safe caching support.

Establish the HTTP session.

ca_lock = <thread.lock object>

cache = {}

cache_hit_callback = None

classmethod clear_cache()
Remove all items from the cache.

classmethod evict(*urls*)
Remove items from cache matching URLs.

Return the number of items removed.

request(*_cache_key*, *_cache_ignore*, *_cache_timeout*, ***kwargs*)
Responsible for dispatching the request and returning the result.

Network level exceptions should be raised and only `requests.Response` should be returned.

Parameters

- **request** – A `requests.PreparedRequest` object containing all the data necessary to perform the request.
- **proxies** – A dictionary of proxy settings to be utilized for the request.
- **timeout** – Specifies the maximum time that the actual HTTP request can take.
- **verify** – Specifies if SSL certificates should be validated.

`**_` should be added to the method call to ignore the extra arguments intended for the cache handler.

timeouts = {}

static with_cache(*function*)
Return a decorator that interacts with a handler's cache.

This decorator must be applied to a DefaultHandler class method or instance method as it assumes *cache*, *ca_lock* and *timeouts* are available.

class `praw.handlers.`**MultiprocessHandler**(*host=u'localhost'*, *port=10101*)
Bases: `object`

A PRAW handler to interact with the PRAW multi-process server.

Construct an instance of the MultiprocessHandler.

evict(*urls*)
Forward the eviction to the server and return its response.

request(***kwargs*)
Forward the request to the server and return its HTTP response.

class `praw.handlers.`**RateLimitHandler**
Bases: `object`

The base handler that provides thread-safe rate limiting enforcement.

While this handler is threadsafe, PRAW is not thread safe when the same *Reddit* instance is being utilized from multiple threads.

Establish the HTTP session.

classmethod evict(*urls*)
Method utilized to evict entries for the given urls.

Parameters **urls** – An iterable containing normalized urls.

Returns The number of items removed from the cache.

By default this method returns False as a cache need not be present.

last_call = {}

static rate_limit (*function*)
Return a decorator that enforces API request limit guidelines.

We are allowed to make a API request every api_request_delay seconds as specified in praw.ini. This value may differ from reddit to reddit. For reddit.com it is 2. Any function decorated with this will be forced to delay _rate_delay seconds from the calling of the last function decorated with this before executing.

This decorator must be applied to a RateLimitHandler class method or instance method as it assumes *rl_lock* and *last_call* are available.

request (*_rate_domain*, *_rate_delay*, ***kwargs*)
Responsible for dispatching the request and returning the result.

Network level exceptions should be raised and only `requests.Response` should be returned.

Parameters

- **request** – A `requests.PreparedRequest` object containing all the data necessary to perform the request.
- **proxies** – A dictionary of proxy settings to be utilized for the request.
- **timeout** – Specifies the maximum time that the actual HTTP request can take.
- **verify** – Specifies if SSL certificates should be validated.

`**_` should be added to the method call to ignore the extra arguments intended for the cache handler.

rl_lock = <thread.lock object>

1.12.6 `decorators` Module

Decorators.

They mainly do two things: ensure API guidelines are followed and prevent unnecessary failed API requests by testing that the call can be made first. Also, they can limit the length of output strings and parse json response for certain errors.

`praw.decorators.`**alias_function** (*function*, *class_name*)
Create a RedditContentObject function mapped to a BaseReddit function.

The BaseReddit classes define the majority of the API's functions. The first argument for many of these functions is the RedditContentObject that they operate on. This factory returns functions appropriate to be called on a RedditContent object that maps to the corresponding BaseReddit function.

`praw.decorators.`**deprecated** (*msg=u''*)
Deprecate decorated method.

`praw.decorators.`**limit_chars** (*func*)
Truncate the string returned from a function and return the result.

`praw.decorators.`**oauth_generator** (*func*)
Set the _use_oauth keyword argument to True when appropriate.

This is needed because generator functions may be called at anytime, and PRAW relies on the Reddit._use_oauth value at original call time to know when to make OAuth requests.

Returned data is not modified.

`praw.decorators.`**raise_api_exceptions** (*func*)
Raise client side exception(s) when present in the API response.

Returned data is not modified.

praw.decorators.**require_captcha**(*func*)
Return a decorator for methods that require captchas.

praw.decorators.**require_oauth**(*func*)
Verify that the OAuth functions can be used prior to use.

Returned data is not modified.

praw.decorators.**restrict_access**(*scope*, *mod=None*, *login=None*, *oauth_only=False*)
Restrict function access unless the user has the necessary permissions.

Raises one of the following exceptions when appropriate:

- LoginRequired
- LoginOrOAuthRequired * the scope attribute will provide the necessary scope name
- ModeratorRequired
- ModeratorOrOAuthRequired * the scope attribute will provide the necessary scope name

Parameters

- **scope** – Indicate the scope that is required for the API call. None or False must be passed to indicate that no scope handles the API call. All scopes save for *read* imply login=True. Scopes with 'mod' in their name imply mod=True.
- **mod** – Indicate that a moderator is required. Implies login=True.
- **login** – Indicate that a login is required.
- **oauth_only** – Indicate that only OAuth is supported for the function.

Returned data is not modified.

This decorator assumes that all mod required functions fit one of these categories:

•have the subreddit as the first argument (Reddit instance functions) or have a subreddit keyword argument

•are called upon a subreddit object (Subreddit RedditContentObject)

•are called upon a RedditContent object with attribute subreddit

1.13 Useful Apps/Scripts

Here are some scripts that people have contributed so far. Feel free to edit this page to add in more.

PRAWtools by BBoe A collection of tools that utilize PRAW. Two current tools are `modutils` a program useful to subreddit moderators, and `subreddit_stats`, a tool to compute submission / comment statistics for a subreddit.

prawoauth2 by Avinash Sajjanshetty `prawoauth2` is a helper library for PRAW which makes writing Reddit bots/apps using OAuth2 super easy, simple and fun.

AutoModerator by Deimos A bot for automating straightforward reddit moderation tasks and improving upon the existing spam-filter.

r.doqdoq A website that displays reddit stories under the guise of Python or Java code.

reddit Notifier for Gnome3 Integrates with Unity and Gnome Shell to display new reddit mail as it arrives.

Link Unscripter A bot for replying to posts <and comments, eventually> that contain javascript-required links to provide non-javascript alternatives.

ClockStalker Examines a redditor's posting history and creates a comment with a nice activity overview. ClockStalker uses an older version of PRAW, the `reddit`, module. It should, but may not, work with the latest version of PRAW.

Butcher bot by u/xiphirx Handles routine tasks on r/Diablo such as the removal of images/memes and bandwagon-esque titles.

r/diablo flair infographic generator by u/xiphirx Creates beautiful infographics.

Groompbot by u/AndrewNeo Posts new videos from YouTube to a subreddit.

newsfrbot by u/keepthepace Parses RSS feeds from some major french publications and posts them to relevant subreddits.

reddit-modbot A relatively lightweight script for automating reddit moderating tasks. It was written as a simpler alternative to AutoModerator by Deimos.

reddit-giveaway-bot A bot that automatically manages giveaway. One feature gives out product keys to the first N commenters.

DailyProgBot A simple challenge-queue submission bot for r/DailyProgrammer. Users submit challenges through a Google Documents form, then the bot crawls said form, posting the appropriate challenge on the appropriate day of the week.

VideoLinkBot by u/shaggorama A bot that aggregates video links in a response comment where multiple videos links appear in reply to a submission (uses a slightly out-of-date version of PRAW, currently requires `Submission.all_comments_flat`).

reddit-analysis by u/rhiever Scrapes a specific subreddit or user and prints out all of the commonly-used words in the past month. Contains a data file containing a list of words that should be considered common.

AlienFeed by u/Jawerty AlienFeed is a command line application made for displaying and interacting with reddit submissions. The client can return a list containing the top submissions in a subreddit, and even open the links up if you'd like.

ALTcointip by u/im14 ALTcointip bot allows redditors to gift (tip) various cryptocoins (Litecoin, PPCoin, Namecoin, etc) to each other as a way of saying thanks.

RedditAgain by Karan Goel Migrate an old reddit account to a new one. Backs up existing content and submissions, and copies subscriptions to a new account.

reddit-cloud by Paul Nechifor Generates word clouds for submissions (or users), posts them to Imgur and the URL to reddit. It's what I run on u/WordCloudBot2.

Reddit-to-Diigo-Copier by Doug Copies your reddit saved links to a Diigo account of your choice. Makes use of PRAW and the Diigo API.

NetflixBot by Alan Wright Parses comments for calls and determines if a movie is availble for streaming on Netflix. Makes use of PRAW and the NetflixRouletteAPI. Run on u/NetflixBot.

RemindMeBot by Joey Called upon with the `RemindMeBot!` command on any subreddit, the user is able to set a reminder for themselves about the current thread or comment. The bot will then send them a private message with the date they specified. u/RemindMeBot.

RedditRover by DarkMio A plugin based Reddit Multi Bot Framework intended for new and advanced programmers to host a wide variety of Reddit bots without mangling with all the ins and outs of Reddit, PRAW and API limitations.

Reddit Keyword Tracking Bot by Jermell Beane <requires Kivy> A bot that will watch any subreddits and email you with updates when it finds words that matter to you. settings are configured via a GUI making it easy for people who don't know how to edit python scripts.

Reddit-Paper by Cameron Gagnon. Command line interface program that will download the top 5 images from r/earthporn, r/spaceporn, etc. and set them as the computer's wallpaper. Currently only tested and used on Ubuntu, but more OS's coming soon.

QR Codify by JstnPwll Fetches username mentions and converts the comment data into a QR code. The code is then posted via ASCII characters as a followup comment.

EVE Killmail Reddit Bot by ArnoldM904 Searches for comments in /r/eve that contain zkillboard killmail links. Then replies to comments with web-scraped TL;DR of the information.

ButtsBot by Judson Dunaway-Barlow A silly bot that posts a picture of a (clothed) butt from the Astros team whenever somebody in the /r/Astros subreddit uses any of a few certain keywords in a comment.

GoodReads Bot by Avinash Sajjanshetty A bot which powers /u/goodreadsbot on Reddit, posts information of a book whenever someone posts a link to Goodreads.

Self-Destruct Bot by diceroll123 Removes posts after a specified amount of time. (For time-sensitive posts.)

Subtitle Bot by arrayofchar Finds youtube links and see if closed captioning (aka subtitle) is available. Formats subtitle into paragraphs based on standard deviation of timestamp gaps. If subtitle is longer than Reddit comment limit, puts it in Pastebin.

Reddit Countdown by matchu Implements a countdown in a subreddit's sidebar to help Reddit communities count down to important events.

<Your Script Here> Edit this page on github to add your script to this list.

1.14 Exceptions

This page documents the exceptions that can occur while running PRAW and what they mean. The exceptions can be divided into three rough categories and a full list of the `ClientException` s and `APIException` s that can occur can be found in the errors module.

1.14.1 ClientException

Something went wrong on the client side of the request. All exceptions of this nature inherit from the exception class `ClientException`. Most of these exceptions occur when you try to do something you don't have authorization to do. For instance trying to remove a submission in a subreddit where the logged-in user is not a moderator will throw a `ModeratorRequired` error.

1.14.2 APIException

Something went wrong on the server side of the request. All exceptions of this nature inherit from the exception class `APIException`. They deal with all sorts of errors that can occur when communicating with a remote API such as trying to login with the incorrect password, which raise a `InvalidUserPass`.

1.14.3 HTTPException

All other errors. The most common occurrence is when reddit returns a non-200 status code. This will raise an exception that is either an object of the *`HTTPException`* or one of its subclasses.

Each of these exceptions will likely have an associated HTTP response status code. The meanings of some of these status codes are:

301, 302

Redirects. Are automatically handled in PRAW, but may result in a `RedirectException` if an unexpected redirect is encountered.

403 (Forbidden)

This will occur if you try to access a restricted resource. For instance a private subreddit that the currently logged-in user doesn't have access to.

```
>>> import praw
>>> r = praw.Reddit('404 test by u/_Daimon_')
>>> r.get_subreddit('lounge', fetch=True)
```

404 (NotFound)

Indicates that the requested resource does not exist.

500

An internal error happened on the server. Sometimes there's a temporary hiccup that cause this and repeating the request will not re-raise the issue. If it's consistently thrown when you call the same PRAW method with the same arguments, then there's either a bug in the way PRAW parses arguments or in the way reddit handles them. Create a submission on r/redditdev so that the right people become aware of the issue and can solve it.

502, 503, 504

A temporary issue at reddit's end. Usually only happens when the servers are under very heavy pressure. Since it's a temporary issue, PRAW will automatically retry the request for you. If you're seeing this error then PRAW has either failed with this request 3 times in a row or it's a request that adds something to reddit's database like *add_comment()*. In this case, the error may be thrown after the comment was added to reddit's database, so retrying the request could result in duplicate comments. To prevent duplication such requests are not retried on errors.

CHAPTER

TWO

REFERENCES AND OTHER RELEVANT PAGES

- PRAW's Source Code
- reddit's Source Code
- reddit's API Wiki Page
- reddit's API Documentation
- reddit Markdown Primer
- reddit.com's FAQ
- reddit.com's Status Twitterbot. Tweets when reddit goes up or down
- r/changelog. Significant changes to reddit's codebase will be announced here in non-developer speak
- r/redditdev. Ask questions about reddit's codebase, PRAW and other API clients here

CHAPTER

THREE

INSTALLATION

PRAW is supported on python 2.7, 3.3, 3.4 and 3.5. The recommended way to install is via pip

```
$ pip install praw
```

If you want to run the development version of PRAW try:

```
$ pip install --upgrade https://github.com/praw-dev/praw/archive/master.zip
```

If you don't have `pip` installed, then the Hitchhiker's Guide to Python has a section for setting it up on Windows, Mac and Linux. There is also a Stack overflow question on installing pip on Windows that might prove helpful.

Alternatively you can do it via easy_install

```
$ easy_install praw
```

CHAPTER

FOUR

SUPPORT

The official place to ask questions about PRAW, reddit and other API wrappers is r/redditdev. If the question is more about Python and less about PRAW, such as "what are generators", then you're likely to get more, faster and more in-depth answers in r/learnpython.

If you've uncovered a bug or have a feature request, then make an issue on our project page at github.

Please note that this project is released with a Contributor Code of Conduct. By participating in this project you agree to abide by its terms.

CHAPTER

FIVE

LICENSE

All of the code contained here is licensed by the GNU GPLv3.

CHAPTER

SIX

A FEW SHORT EXAMPLES

Note: These example are intended to be completed in order. While you are free to skip down to what you want to accomplish, please check the previous examples for any `NameErrors` you might encounter.

1. Import the package

```
>>> import praw
```

2. Create the Reddit object (requires a user-agent):

```
>>> r = praw.Reddit(user_agent='Test Script by /u/bboe')
```

3. Logging in:

```
>>> r.login('username', 'password')
```

4. Send a message (requires login):

```
>>> r.send_message('user', 'Subject Line', 'You are awesome!')
```

5. Mark all unread messages as read (requires login):

```
>>> for msg in r.get_unread(limit=None):
...     msg.mark_as_read()
```

6. Get the top submissions for /r/python:

```
>>> submissions = r.get_subreddit('python').get_top(limit=10)
```

7. Get comments from a given submission:

```
>>> submission = next(submissions)
>>> submission.comments
```

8. Comment on a submission (requires login):

```
>>> submission.add_comment('text')
```

9. Reply to a comment (requires login):

```
>>> comment = submission.comments[0]
>>> comment.reply('test')
```

10. Voting (requires login):

```
>>> # item can be a comment or submission
>>> item.upvote()
>>> item.downvote()
>>> item.clear_vote()
```

11. Deleting (requires login):

```
>>> # item can be a comment or submission
>>> item.delete()
```

12. Saving a submission (requires login):

```
>>> submission.save()
>>> submission.unsave()
```

13. Create a SELF submission (requires login):

```
>>> r.submit('reddit_api_test', 'submission title', text='body')
```

14. Create a URL submission (requires login):

```
>>> r.submit('reddit_api_test', 'Google!', url='http://google.com')
```

15. Get user karma:

```
>>> user = r.get_redditor('ketralnis')
>>> user.link_karma
>>> user.comment_karma
```

16. Get saved links (requires login):

```
>>> r.user.get_saved()
```

17. Get content newer than a comment or submission's id:

```
>>> r.get_subreddit('python').get_top(limit=None,
                                      place_holder=submission.id)
```

18. (Un)subscribe to a subreddit (requires login):

```
>>> r.get_subreddit('python').subscribe()
>>> r.get_subreddit('python').unsubscribe()
```

19. (Un)friend a user:

```
>>> r.get_redditor('ketralnis').friend()
>>> r.get_redditor('ketralnis').unfriend()
```

20. Create a subreddit:

```
>>> r.create_subreddit(short_title='MyIncredibleSubreddit',
...                    full_title='my Incredibly Cool Subreddit',
...                    description='It is incredible!')
```

21. Get flair mappings for a particular subreddit (requires mod privileges):

```
>>> item = next(r.get_subreddit('python').get_flair_list())
>>> item['user']
>>> item['flair_text']
>>> item['flair_css_class']
```

22. Set / update user flair (requires mod privileges):

```
>>> r.get_subreddit('python').set_flair('user', 'text flair', 'css-class')
```

23. Clear user flair (requires mod privileges):

```
>>> r.get_subreddit('python').set_flair('user')
```

24. Bulk set user flair (requires mod privileges):

```
>>> flair_mapping = [{'user':'user', 'flair_text':'dev'},
...                  {'user':'pyapitestuser3', 'flair_css_class':'css2'},
...                  {'user':'pyapitestuser2', 'flair_text':'AWESOME',
...                  'flair_css_class':'css'}]
>>> r.get_subreddit('python').set_flair_csv(flair_mapping)
```

25. Add flair templates (requires mod privileges):

```
>>> r.get_subreddit('python').add_flair_template(text='editable',
...                                              css_class='foo',
...                                              text_editable=True)
```

26. Clear flair templates (requires mod privileges):

```
>>> r.get_subreddit('python').clear_flair_templates()
```

CHAPTER

SEVEN

USEFUL SCRIPTS

AutoModerator by Deimos A bot for automating straightforward reddit moderation tasks and improving upon the existing spam-filter.

ClockStalker Examines a redditor's posting history and creates a comment with a nice activity overview. ClockStalker uses an older version of PRAW, the `reddit`, module. It should, but may not, work with the latest version of PRAW.

DailyProgBot A simple challenge-queue submission bot for r/DailyProgrammer. Users submit challenges through a Google Documents form, then the bot crawls said form, posting the appropriate challenge on the appropriate day of the week.

p

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W