

! Note

You are not using the most up to date version of the library. [3.6.0](#) is the newest version.

API Reference

This page contains some basic documentation for the Tweepy module.

tweepy.api – Twitter API wrapper

```
class API([auth_handler=None] [, host='api.twitter.com'] [, search_host='search.twitter.com'] [, cache=None] [, api_root='/1'] [, search_root=""] [, retry_count=0] [, retry_delay=0] [, retry_errors=None] [, timeout=60] [, parser=ModelParser] [, compression=False] [, wait_on_rate_limit=False] [, wait_on_rate_limit_notify=False] [, proxy=None])
```

This class provides a wrapper for the API as provided by Twitter. The functions provided in this class are listed below.

Parameters:

- **auth_handler** – authentication handler to be used
- **host** – general API host
- **search_host** – search API host
- **cache** – cache backend to use
- **api_root** – general API path root
- **search_root** – search API path root
- **retry_count** – default number of retries to attempt when error occurs
- **retry_delay** – number of seconds to wait between retries
- **retry_errors** – which HTTP status codes to retry
- **timeout** – The maximum amount of time to wait for a response from Twitter
- **parser** – The object to use for parsing the response from Twitter
- **compression** – Whether or not to use GZIP compression for requests
- **wait_on_rate_limit** – Whether or not to automatically wait for rate limits to replenish
- **wait_on_rate_limit_notify** – Whether or not to print a notification when Tweepy is waiting for rate limits to replenish
- **proxy** – The full url to an HTTPS proxy to use for connecting to Twitter.

Timeline methods

API.home_timeline([*since_id*] [, *max_id*] [, *count*] [, *page*])

Returns the 20 most recent statuses, including retweets, posted by the authenticating user and that user's friends. This is the equivalent of /timeline/home on the Web.

Parameters:

- **since_id** – Returns only statuses with an ID greater than (that is, more recent than) the specified ID.
- **max_id** – Returns only statuses with an ID less than (that is, older than) or equal to the specified ID.
- **count** – Specifies the number of statuses to retrieve.
- **page** – Specifies the page of results to retrieve. Note: there are pagination limits.

Return type: list of `Status` objects

API.statuses_lookup(*id* [, *include_entities*] [, *trim_user*] [, *map*])

Returns full Tweet objects for up to 100 tweets per request, specified by the *id* parameter.

Parameters:

- **id** – A list of Tweet IDs to lookup, up to 100
- **include_entities** – A boolean indicating whether or not to include [entities] (<https://dev.twitter.com/docs/entities>) in the returned tweets. Defaults to False.
- **trim_user** – A boolean indicating if user IDs should be provided, instead of full user information. Defaults to False.
- **map** – A boolean indicating whether or not to include tweets that cannot be shown, but with a value of None. Defaults to False.

Return type: list of `Status` objects

API.user_timeline([*id/user_id/screen_name*] [, *since_id*] [, *max_id*] [, *count*] [, *page*])

Returns the 20 most recent statuses posted from the authenticating user or the user specified. It's also possible to request another user's timeline via the *id* parameter.

- Parameters:**
- **id** – Specifies the ID or screen name of the user.
 - **user_id** – Specifies the ID of the user. Helpful for disambiguating when a valid user ID is also a valid screen name.
 - **screen_name** – Specifies the screen name of the user. Helpful for disambiguating when a valid screen name is also a user ID.
 - **since_id** – Returns only statuses with an ID greater than (that is, more recent than) the specified ID.
 - **max_id** – Returns only statuses with an ID less than (that is, older than) or equal to the specified ID.
 - **count** – Specifies the number of statuses to retrieve.
 - **page** – Specifies the page of results to retrieve. Note: there are pagination limits.

Return type: list of `Status` objects

`API.retweets_of_me([since_id] [,max_id] [,count] [,page])`

Returns the 20 most recent tweets of the authenticated user that have been retweeted by others.

- Parameters:**
- **since_id** – Returns only statuses with an ID greater than (that is, more recent than) the specified ID.
 - **max_id** – Returns only statuses with an ID less than (that is, older than) or equal to the specified ID.
 - **count** – Specifies the number of statuses to retrieve.
 - **page** – Specifies the page of results to retrieve. Note: there are pagination limits.

Return type: list of `Status` objects

Status methods

`API.get_status(id)`

Returns a single status specified by the ID parameter.

- Parameters:** `id` – The numerical ID of the status.

Return type: `Status` object

`API.update_status(status [,in_reply_to_status_id] [,lat] [,long] [,source] [,place_id])`

Update the authenticated user's status. Statuses that are duplicates or too long will be silently ignored.

Parameters:

- **status** – The text of your status update.
- **in_reply_to_status_id** – The ID of an existing status that the update is in reply to.
- **lat** – The location's latitude that this tweet refers to.
- **long** – The location's longitude that this tweet refers to.
- **source** – Source of the update. Only supported by Identi.ca. Twitter ignores this parameter.
- **place_id** – Twitter ID of location which is listed in the Tweet if geolocation is enabled for the user.

Return type:

`Status` object

API.update_with_media(*filename* [, *status*] [, *in_reply_to_status_id*] [, *lat*] [, *long*] [, *source*] [, *place_id*] [, *file*])

Update the authenticated user's status. Statuses that are duplicates or too long will be silently ignored.

Parameters:

- **filename** – The filename of the image to upload. This will automatically be opened unless *file* is specified
- **status** – The text of your status update.
- **in_reply_to_status_id** – The ID of an existing status that the update is in reply to.
- **lat** – The location's latitude that this tweet refers to.
- **long** – The location's longitude that this tweet refers to.
- **source** – Source of the update. Only supported by Identi.ca. Twitter ignores this parameter.
- **place_id** – Twitter ID of location which is listed in the Tweet if geolocation is enabled for the user.
- **file** – A file object, which will be used instead of opening *filename*. *filename* is still required, for MIME type detection and to use as a form field in the POST data

Return type:

`Status` object

API.destroy_status(*id*)

Destroy the status specified by the id parameter. The authenticated user must be the author of the status to destroy.

Parameters: `id` – The numerical ID of the status.

Return type: `status` object

API.retweet(`id`)

Retweets a tweet. Requires the id of the tweet you are retweeting.

Parameters: `id` – The numerical ID of the status.

Return type: `status` object

API.retweets(`id` [, `count`])

Returns up to 100 of the first retweets of the given tweet.

Parameters:

- `id` – The numerical ID of the status.
- `count` – Specifies the number of retweets to retrieve.

Return type: list of `status` objects

User methods

API.get_user(`id/user_id/screen_name`)

Returns information about the specified user.

Parameters:

- `id` – Specifies the ID or screen name of the user.
- `user_id` – Specifies the ID of the user. Helpful for disambiguating when a valid user ID is also a valid screen name.
- `screen_name` – Specifies the screen name of the user. Helpful for disambiguating when a valid screen name is also a user ID.

Return type: `user` object

API.me()

Returns the authenticated user's information.

Return type: `user` object

API.followers([*id/screen_name/user_id*] [, *cursor*])

Returns an user's followers ordered in which they were added 100 at a time. If no user is specified by id/screen name, it defaults to the authenticated user.

Parameters:

- ***id*** – Specifies the ID or screen name of the user.
- ***user_id*** – Specifies the ID of the user. Helpful for disambiguating when a valid user ID is also a valid screen name.
- ***screen_name*** – Specifies the screen name of the user. Helpful for disambiguating when a valid screen name is also a user ID.
- ***cursor*** – Breaks the results into pages. Provide a value of -1 to begin paging. Provide values as returned to in the response body's next_cursor and previous_cursor attributes to page back and forth in the list.

Return type: list of `user` objects

API.search_users(*q* [, *per_page*] [, *page*])

Run a search for users similar to Find People button on Twitter.com; the same results returned by people search on Twitter.com will be returned by using this API (about being listed in the People Search). It is only possible to retrieve the first 1000 matches from this API.

Parameters:

- ***q*** – The query to run against people search.
- ***per_page*** – Specifies the number of statuses to retrieve. May not be greater than 20.
- ***page*** – Specifies the page of results to retrieve. Note: there are pagination limits.

Return type: list of `user` objects

Direct Message Methods

API.direct_messages([*since_id*] [, *max_id*] [, *count*] [, *page*] [, *full_text*])

Returns direct messages sent to the authenticating user.

- Parameters:**
- **since_id** – Returns only statuses with an ID greater than (that is, more recent than) the specified ID.
 - **max_id** – Returns only statuses with an ID less than (that is, older than) or equal to the specified ID.
 - **count** – Specifies the number of statuses to retrieve.
 - **page** – Specifies the page of results to retrieve. Note: there are pagination limits.
 - **full_text** – A boolean indicating whether or not the full text of a message should be returned. If False the message text returned will be truncated to 140 chars. Defaults to False.

Return type: list of `DirectMessage` objects

`API.get_direct_message([id] [, full_text])`

Returns a specific direct message.

- Parameters:**
- **id** – `|id|`
 - **full_text** – A boolean indicating whether or not the full text of a message should be returned. If False the message text returned will be truncated to 140 chars. Defaults to False.

Return type: `DirectMessage` object

`API.sent_direct_messages([since_id] [, max_id] [, count] [, page] [, full_text])`

Returns direct messages sent by the authenticating user.

- Parameters:**
- **since_id** – Returns only statuses with an ID greater than (that is, more recent than) the specified ID.
 - **max_id** – Returns only statuses with an ID less than (that is, older than) or equal to the specified ID.
 - **count** – Specifies the number of statuses to retrieve.
 - **page** – Specifies the page of results to retrieve. Note: there are pagination limits.
 - **full_text** – A boolean indicating whether or not the full text of a message should be returned. If False the message text returned will be truncated to 140 chars. Defaults to False.

Return type: list of `DirectMessage` objects

API.send_direct_message(*user/screen_name/user_id*, *text*)

Sends a new direct message to the specified user from the authenticating user.

- Parameters:**
- **user** – The ID or screen name of the recipient user.
 - **screen_name** – screen name of the recipient user
 - **user_id** – user id of the recipient user

Return type: `DirectMessage` object

API.destroy_direct_message(*id*)

Destroy a direct message. Authenticating user must be the recipient of the direct message.

- Parameters:** *id* – The ID of the direct message to destroy.

Return type: `DirectMessage` object

Friendship Methods

API.create_friendship(*id/screen_name/user_id* [, *follow*])

Create a new friendship with the specified user (aka follow).

- Parameters:**
- **id** – Specifies the ID or screen name of the user.
 - **screen_name** – Specifies the screen name of the user. Helpful for disambiguating when a valid screen name is also a user ID.
 - **user_id** – Specifies the ID of the user. Helpful for disambiguating when a valid user ID is also a valid screen name.
 - **follow** – Enable notifications for the target user in addition to becoming friends.

Return type: `User` object

API.destroy_friendship(*id/screen_name/user_id*)

Destroy a friendship with the specified user (aka unfollow).

- Parameters:**
- **id** – Specifies the ID or screen name of the user.
 - **screen_name** – Specifies the screen name of the user. Helpful for disambiguating when a valid screen name is also a user ID.
 - **user_id** – Specifies the ID of the user. Helpful for disambiguating when a valid user ID is also a valid screen name.

Return type: `User` object

API.exists_friendship(*user_a, user_b*)

Checks if a friendship exists between two users. Will return True if *user_a* follows *user_b*, otherwise False.

- Parameters:**
- **user_a** – The ID or screen_name of the subject user.
 - **user_b** – The ID or screen_name of the user to test for following.

Return type: True/False

API.show_friendship(*source_id/source_screen_name, target_id/target_screen_name*)

Returns detailed information about the relationship between two users.

- Parameters:**
- **source_id** – The user_id of the subject user.
 - **source_screen_name** – The screen_name of the subject user.
 - **target_id** – The user_id of the target user.
 - **target_screen_name** – The screen_name of the target user.

Return type: `Friendship` object

API.friends_ids(*id/screen_name/user_id [, cursor]*)

Returns an array containing the IDs of users being followed by the specified user.

- Parameters:**
- **id** – Specifies the ID or screen name of the user.
 - **screen_name** – Specifies the screen name of the user. Helpful for disambiguating when a valid screen name is also a user ID.
 - **user_id** – Specifies the ID of the user. Helpful for disambiguating when a valid user ID is also a valid screen name.
 - **cursor** – Breaks the results into pages. Provide a value of -1 to begin paging. Provide values as returned to in the response body's next_cursor and previous_cursor attributes to page back and forth in the list.

Return type: list of Integers

API.followers_ids(*id/screen_name/user_id*)

Returns an array containing the IDs of users following the specified user.

- Parameters:**
- **id** – Specifies the ID or screen name of the user.
 - **screen_name** – Specifies the screen name of the user. Helpful for disambiguating when a valid screen name is also a user ID.
 - **user_id** – Specifies the ID of the user. Helpful for disambiguating when a valid user ID is also a valid screen name.
 - **cursor** – Breaks the results into pages. Provide a value of -1 to begin paging. Provide values as returned to in the response body's next_cursor and previous_cursor attributes to page back and forth in the list.

Return type: list of Integers

Account Methods

API.verify_credentials()

Verify the supplied user credentials are valid.

Return type: `user` object if credentials are valid, otherwise False

API.rate_limit_status()

Returns the remaining number of API requests available to the requesting user before the API limit is reached for the current hour. Calls to rate_limit_status do not count against the rate limit. If authentication credentials are provided, the rate limit status for the authenticating user is returned. Otherwise, the rate limit status for the requester's IP address is returned.

Return type: JSON object

API.set_delivery_device(*device*)

Sets which device Twitter delivers updates to for the authenticating user. Sending “none” as the device parameter will disable SMS updates.

Parameters: **device** – Must be one of: sms, none

Return type: User object

API.update_profile_colors([*profile_background_color*] [, *profile_text_color*] [, *profile_link_color*] [, *profile_sidebar_fill_color*] [, *profile_sidebar_border_color*])

Sets one or more hex values that control the color scheme of the authenticating user’s profile page on twitter.com.

Parameters:

- **profile_background_color** –
- **profile_text_color** –
- **profile_link_color** –
- **profile_sidebar_fill_color** –
- **profile_sidebar_border_color** –

Return type: User object

API.update_profile_image(*filename*)

Update the authenticating user’s profile image. Valid formats: GIF, JPG, or PNG

Parameters: **filename** – local path to image file to upload. Not a remote URL!

Return type: User object

API.update_profile_background_image(*filename*)

Update authenticating user’s background image. Valid formats: GIF, JPG, or PNG

Parameters: **filename** – local path to image file to upload. Not a remote URL!

Return type: User object

API.update_profile([*name*] [, *url*] [, *location*] [, *description*])

Sets values that users are able to set under the “Account” tab of their settings page.

- Parameters:**
- **name** – Maximum of 20 characters
 - **url** – Maximum of 100 characters. Will be prepended with “<http://>” if not present
 - **location** – Maximum of 30 characters
 - **description** – Maximum of 160 characters

Return type: `user` object

Favorite Methods

`API.favorites([id][,page])`

Returns the favorite statuses for the authenticating user or user specified by the ID parameter.

- Parameters:**
- **id** – The ID or screen name of the user to request favorites
 - **page** – Specifies the page of results to retrieve. Note: there are pagination limits.

Return type: list of `status` objects

`API.create_favorite(id)`

Favorites the status specified in the ID parameter as the authenticating user.

- Parameters:** `id` – The numerical ID of the status.

Return type: `status` object

`API.destroy_favorite(id)`

Un-favorites the status specified in the ID parameter as the authenticating user.

- Parameters:** `id` – The numerical ID of the status.

Return type: `status` object

Block Methods

`API.create_block(id/screen_name/user_id)`

Blocks the user specified in the ID parameter as the authenticating user. Destroys a friendship to the blocked user if it exists.

- Parameters:**
- **id** – Specifies the ID or screen name of the user.
 - **screen_name** – Specifies the screen name of the user. Helpful for disambiguating when a valid screen name is also a user ID.
 - **user_id** – Specifies the ID of the user. Helpful for disambiguating when a valid user ID is also a valid screen name.

Return type: `User` object

API.destroy_block(*id/screen_name/user_id*)

Un-blocks the user specified in the ID parameter for the authenticating user.

- Parameters:**
- **id** – Specifies the ID or screen name of the user.
 - **screen_name** – Specifies the screen name of the user. Helpful for disambiguating when a valid screen name is also a user ID.
 - **user_id** – Specifies the ID of the user. Helpful for disambiguating when a valid user ID is also a valid screen name.

Return type: `User` object

API.blocks([*page*])

Returns an array of user objects that the authenticating user is blocking.

Parameters: `page` – Specifies the page of results to retrieve. Note: there are pagination limits.

Return type: list of `User` objects

API.blocks_ids()

Returns an array of numeric user ids the authenticating user is blocking.

Return type: list of Integers

Spam Reporting Methods

API.report_spam([*id/user_id/screen_name*])

The user specified in the id is blocked by the authenticated user and reported as a spammer.

- Parameters:**
- **id** – Specifies the ID or screen name of the user.
 - **screen_name** – Specifies the screen name of the user. Helpful for disambiguating when a valid screen name is also a user ID.
 - **user_id** – Specifies the ID of the user. Helpful for disambiguating when a valid user ID is also a valid screen name.

Return type: `User` object

Saved Searches Methods

API.`saved_searches()`

Returns the authenticated user's saved search queries.

Return type: list of `SavedSearch` objects

API.`get_saved_search(id)`

Retrieve the data for a saved search owned by the authenticating user specified by the given id.

Parameters: `id` – The id of the saved search to be retrieved.

Return type: `SavedSearch` object

API.`create_saved_search(query)`

Creates a saved search for the authenticated user.

Parameters: `query` – The query of the search the user would like to save.

Return type: `SavedSearch` object

API.`destroy_saved_search(id)`

Destroys a saved search for the authenticated user. The search specified by id must be owned by the authenticating user.

Parameters: `id` – The id of the saved search to be deleted.

Return type: `SavedSearch` object

Help Methods

API.search(*q* [, *lang*] [, *locale*] [, *rpp*] [, *page*] [, *since_id*] [, *geocode*] [, *show_user*])

Returns tweets that match a specified query.

Parameters:

- ***q*** – the search query string
- ***lang*** – Restricts tweets to the given language, given by an ISO 639-1 code.
- ***locale*** – Specify the language of the query you are sending. This is intended for language-specific clients and the default should work in the majority of cases.
- ***rpp*** – The number of tweets to return per page, up to a max of 100.
- ***page*** – The page number (starting at 1) to return, up to a max of roughly 1500 results (based on *rpp* * *page*).
- ***since_id*** – Returns only statuses with an ID greater than (that is, more recent than) the specified ID.
- ***geocode*** – Returns tweets by users located within a given radius of the given latitude/longitude. The location is preferentially taking from the Geotagging API, but will fall back to their Twitter profile. The parameter value is specified by “latitude,longitude,radius”, where radius units must be specified as either “mi” (miles) or “km” (kilometers). Note that you cannot use the near operator via the API to geocode arbitrary locations; however you can use this geocode parameter to search near geocodes directly.
- ***show_user*** – When true, prepends “<user>:” to the beginning of the tweet. This is useful for readers that do not display Atom’s author field. The default is false.

Return type: list of `SearchResult` objects

List Methods

API.create_list(*name* [, *mode*] [, *description*])

Creates a new list for the authenticated user. Accounts are limited to 20 lists.

Parameters:

- ***name*** – The name of the new list.
- ***mode*** – Whether your list is public or private. Values can be public or private. Lists are public by default if no mode is specified.
- ***description*** – The description of the list you are creating.

Return type: `List` object

API.destroy_list(*slug*)

Deletes the specified list. Must be owned by the authenticated user.

Parameters: `slug` – the slug name or numerical ID of the list

Return type: `List` object

API.update_list(`slug` [, `name`] [, `mode`] [, `description`])

Updates the specified list. Note: this current throws a 500. Twitter is looking into the issue.

Parameters:

- `slug` – the slug name or numerical ID of the list
- `name` – What you'd like to change the lists name to.
- `mode` – Whether your list is public or private. Values can be public or private.
Lists are public by default if no mode is specified.
- `description` – What you'd like to change the list description to.

Return type: `List` object

API.lists([`cursor`])

List the lists of the specified user. Private lists will be included if the authenticated users is the same as the user who's lists are being returned.

Parameters: `cursor` – Breaks the results into pages. Provide a value of -1 to begin paging.
Provide values as returned to in the response body's next_cursor and previous_cursor attributes to page back and forth in the list.

Return type: list of `List` objects

API.lists_memberships([`cursor`])

List the lists the specified user has been added to.

Parameters: `cursor` – Breaks the results into pages. Provide a value of -1 to begin paging.
Provide values as returned to in the response body's next_cursor and previous_cursor attributes to page back and forth in the list.

Return type: list of `List` objects

API.lists_subscriptions([`cursor`])

List the lists the specified user follows.

Parameters: `cursor` – Breaks the results into pages. Provide a value of -1 to begin paging.
Provide values as returned to in the response body's next_cursor and previous_cursor attributes to page back and forth in the list.

Return type: list of `List` objects

API.list_timeline(owner, slug [, since_id] [, max_id] [, per_page] [, page])

Show tweet timeline for members of the specified list.

Parameters:

- **owner** – the screen name of the owner of the list
- **slug** – the slug name or numerical ID of the list
- **since_id** – Returns only statuses with an ID greater than (that is, more recent than) the specified ID.
- **max_id** – Returns only statuses with an ID less than (that is, older than) or equal to the specified ID.
- **per_page** – Number of results per a page
- **page** – Specifies the page of results to retrieve. Note: there are pagination limits.

Return type: list of `status` objects

API.get_list(owner, slug)

Show the specified list. Private lists will only be shown if the authenticated user owns the specified list.

Parameters:

- **owner** – the screen name of the owner of the list
- **slug** – the slug name or numerical ID of the list

Return type: `List` object

API.add_list_member(slug, id)

Add a member to a list. The authenticated user must own the list to be able to add members to it. Lists are limited to having 500 members.

Parameters:

- **slug** – the slug name or numerical ID of the list
- **id** – the ID of the user to add as a member

Return type: `List` object

API.remove_list_member(slug, id)

Removes the specified member from the list. The authenticated user must be the list's owner to remove members from the list.

- Parameters:**
- **slug** – the slug name or numerical ID of the list
 - **id** – the ID of the user to remove as a member

Return type: `List` object

API.list_members(*owner, slug, cursor*)

Returns the members of the specified list.

- Parameters:**
- **owner** – the screen name of the owner of the list
 - **slug** – the slug name or numerical ID of the list
 - **cursor** – Breaks the results into pages. Provide a value of -1 to begin paging. Provide values as returned to in the response body's next_cursor and previous_cursor attributes to page back and forth in the list.

Return type: list of `User` objects

API.is_list_member(*owner, slug, id*)

Check if a user is a member of the specified list.

- Parameters:**
- **owner** – the screen name of the owner of the list
 - **slug** – the slug name or numerical ID of the list
 - **id** – the ID of the user to check

Return type: `User` object if user is a member of list, otherwise False.

API.subscribe_list(*owner, slug*)

Make the authenticated user follow the specified list.

- Parameters:**
- **owner** – the screen name of the owner of the list
 - **slug** – the slug name or numerical ID of the list

Return type: `List` object

API.unsubscribe_list(*owner, slug*)

Unsubscribes the authenticated user from the specified list.

- Parameters:**
- **owner** – the screen name of the owner of the list
 - **slug** – the slug name or numerical ID of the list

Return type: `List` object

API.list_subscribers(*owner, slug [, cursor]*)

Returns the subscribers of the specified list.

- Parameters:**
- **owner** – the screen name of the owner of the list
 - **slug** – the slug name or numerical ID of the list
 - **cursor** – Breaks the results into pages. Provide a value of -1 to begin paging. Provide values as returned to in the response body's next_cursor and previous_cursor attributes to page back and forth in the list.

Return type: list of `User` objects

API.is_subscribed_list(*owner, slug, id*)

Check if the specified user is a subscriber of the specified list.

- Parameters:**
- **owner** – the screen name of the owner of the list
 - **slug** – the slug name or numerical ID of the list
 - **id** – the ID of the user to check

Return type: `User` object if user is subscribed to the list, otherwise False.

Trends Methods

API.trends_available()

Returns the locations that Twitter has trending topic information for. The response is an array of “locations” that encode the location’s WOEID (a Yahoo! Where On Earth ID) and some other human-readable information such as a canonical name and country the location belongs in.

Return type: `JSON` object

API.trends_place(*id [, exclude]*)

Returns the top 10 trending topics for a specific WOEID, if trending information is available for it.

The response is an array of “trend” objects that encode the name of the trending topic, the query parameter that can be used to search for the topic on Twitter Search, and the Twitter Search URL.

This information is cached for 5 minutes. Requesting more frequently than that will not return any more data, and will count against your rate limit usage.

- Parameters:**
- **id** – The Yahoo! Where On Earth ID of the location to return trending information for. Global information is available by using 1 as the WOEID.
 - **exclude** – Setting this equal to hashtags will remove all hashtags from the trends list.

Return type: JSON object

API.trends_closest(*lat, long*)

Returns the locations that Twitter has trending topic information for, closest to a specified location.

The response is an array of “locations” that encode the location’s WOEID and some other human-readable information such as a canonical name and country the location belongs in.

A WOEID is a Yahoo! Where On Earth ID.

- Parameters:**
- **lat** – If provided with a long parameter the available trend locations will be sorted by distance, nearest to furthest, to the co-ordinate pair. The valid ranges for longitude is -180.0 to +180.0 (West is negative, East is positive) inclusive.
 - **long** – If provided with a lat parameter the available trend locations will be sorted by distance, nearest to furthest, to the co-ordinate pair. The valid ranges for longitude is -180.0 to +180.0 (West is negative, East is positive) inclusive.

Return type: JSON object

Geo Methods

API.reverse_geocode([*lat*] [, *long*] [, *accuracy*] [, *granularity*] [, *max_results*])

Given a latitude and longitude, looks for places (cities and neighbourhoods) whose IDs can be specified in a call to `update_status()` to appear as the name of the location. This call provides a detailed response about the location in question; the `nearby_places()` function should be preferred for getting a list of places nearby without great detail.

- Parameters:**
- `lat` – The location's latitude.
 - `long` – The location's longitude.
 - `accuracy` – Specify the “region” in which to search, such as a number (then this is a radius in meters, but it can also take a string that is suffixed with ft to specify feet). If this is not passed in, then it is assumed to be 0m
 - `granularity` – Assumed to be ‘neighborhood’ by default; can also be ‘city’.
 - `max_results` – A hint as to the maximum number of results to return. This is only a guideline, which may not be adhered to.

`API.reverse_geocode([lat] [, long] [, ip] [, accuracy] [, granularity] [, max_results])`

Given a latitude and longitude, looks for nearby places (cities and neighbourhoods) whose IDs can be specified in a call to `update_status()` to appear as the name of the location. This call provides a detailed response about the location in question; the `nearby_places()` function should be preferred for getting a list of places nearby without great detail.

- Parameters:**
- `lat` – The location's latitude.
 - `long` – The location's longitude.
 - `ip` – The location's IP address. Twitter will attempt to geolocate using the IP address.
 - `accuracy` – Specify the “region” in which to search, such as a number (then this is a radius in meters, but it can also take a string that is suffixed with ft to specify feet). If this is not passed in, then it is assumed to be 0m
 - `granularity` – Assumed to be ‘neighborhood’ by default; can also be ‘city’.
 - `max_results` – A hint as to the maximum number of results to return. This is only a guideline, which may not be adhered to.

`API.geo_id(id)`

Given *id* of a place, provide more details about that place.

- Parameters:** `id` – Valid Twitter ID of a location.

`tweepy.error` – Exceptions

The exceptions are available in the `tweepy` module directly, which means `tweepy.error` itself does not need to be imported. For example, `tweepy.error.TweepError` is available as `tweepy.TweepError`.

exception `TweepError`

The main exception Tweepy uses. Is raised for a number of things.

When a `TweepError` is raised due to an error Twitter responded with, the error code (as described in the API documentation) can be accessed at `TweepError.message[0]['code']`. Note, however, that `TweepError`s also may be raised with other things as message (for example plain error reason strings).

exception `RateLimitError`

Is raised when an API method fails due to hitting Twitter's rate limit. Makes for easy handling of the rate limit specifically.

Inherits from `TweepError`, so `except TweepError` will catch a `RateLimitError` too.