Python-Fitbit Documentation

Release 0.3.0

Issac Kelly, Percy Perez, Brad Pitcher

October 05, 2018

Contents

This is a complete python implementation of the Fitbit API.

It uses oAuth for authentication, it supports both us and si measurements

CHAPTER 1

Quickstart

If you are only retrieving data that doesn't require authorization, then you can use the unauthorized interface:

```
import fitbit
unauth_client = fitbit.Fitbit('<consumer_key>', '<consumer_secret>')
# certain methods do not require user keys
unauth_client.food_units()
```

Here is an example of authorizing with OAuth 2.0:

```
# You'll have to gather the tokens on your own, or use
# ./gather_keys_oauth2.py
authd_client = fitbit.Fitbit('<consumer_key>', '<consumer_secret>',
                             access_token='<access_token>', refresh_token='<refresh_token>')
authd_client.sleep()
```

CHAPTER 2

Fitbit API

Some assumptions you should note. Anywhere it says user_id=None, it assumes the current user_id from the credentials given, and passes a – through the API. Anywhere it says date=None, it should accept either `None` or a `date` or `datetime` object (anything with proper strftime will do), or a string formatted as `%Y-%m-%d`.

class `fitbit.`**Fitbit**(*client_id*, *client_secret*, *access_token=None*, *refresh_token=None*, *expires_at=None*, *refresh_cb=None*, *redirect_uri=None*, *system='en_US'*, ***kwargs*)

Before using this class, create a Fitbit app here. There you will get the client id and secret needed to instantiate this class. When first authorizing a user, make sure to pass the *redirect_uri* keyword arg so fitbit will know where to return to when the authorization is complete. See gather_keys_oauth2.py for a reference implementation of the authorization process. You should save `access_token`, `refresh_token`, and `expires_at` from the returned token for each user you authorize.

When instantiating this class for use with an already authorized user, pass in the `access_token`, `refresh_token`, and `expires_at` keyword arguments. We also strongly recommend passing in a `refresh_cb` keyword argument, which should be a function taking one argument: a token dict. When that argument is present, we will automatically refresh the access token when needed and call this function so that you can save the updated token data. If you don't save the updated information, then you could end up with invalid access and refresh tokens, and the only way to recover from that is to reauthorize the user.

body(*date=None*, *user_id=None*, *data=None*)
Get body data: https://dev.fitbit.com/docs/body/

activities(*date=None*, *user_id=None*, *data=None*)
Get body data: https://dev.fitbit.com/docs/activity/

foods_log(*date=None*, *user_id=None*, *data=None*)
Get food logs data: https://dev.fitbit.com/docs/food-logging/#get-food-logs

foods_log_water(*date=None*, *user_id=None*, *data=None*)
Get water logs data: https://dev.fitbit.com/docs/food-logging/#get-water-logs

sleep(*date=None*, *user_id=None*, *data=None*)
Get sleep data: https://dev.fitbit.com/docs/sleep/

heart(*date=None*, *user_id=None*, *data=None*)
Get heart rate data: https://dev.fitbit.com/docs/heart-rate/

bp(*date=None*, *user_id=None*, *data=None*)
Get blood pressure data: https://dev.fitbit.com/docs/heart-rate/

delete_body(*log_id*)
Delete a body log, given a log id

delete_activities(*log_id*)
Delete an activity log, given a log id

delete_foods_log(*log_id*)
Delete a food log, given a log id

delete_foods_log_water(*log_id*)
Delete a water log, given a log id

delete_sleep(*log_id*)
Delete a sleep log, given a log id

delete_heart(*log_id*)
Delete a heart log, given a log id

delete_bp(*log_id*)
Delete a blood pressure log, given a log id

recent_foods(*user_id=None*, *qualifier=''*)
Get recently logged foods: https://dev.fitbit.com/docs/food-logging/#get-recent-foods

frequent_foods(*user_id=None*, *qualifier=''*)
Get frequently logged foods: https://dev.fitbit.com/docs/food-logging/#get-frequent-foods

favorite_foods(*user_id=None*, *qualifier=''*)
Get favorited foods: https://dev.fitbit.com/docs/food-logging/#get-favorite-foods

recent_activities(*user_id=None*, *qualifier=''*)
Get recently logged activities: https://dev.fitbit.com/docs/activity/#get-recent-activity-types

frequent_activities(*user_id=None*, *qualifier=''*)
Get frequently logged activities: https://dev.fitbit.com/docs/activity/#get-frequent-activities

favorite_activities(*user_id=None*, *qualifier=''*)
Get favorited foods: https://dev.fitbit.com/docs/activity/#get-favorite-activities

accept_invite(*other_user_id*)
Convenience method for respond_to_invite

activities_daily_goal(*calories_out=None*, *active_minutes=None*, *floors=None*, *distance=None*, *steps=None*)
Implements the following APIs for period equal to daily

https://dev.fitbit.com/docs/activity/#get-activity-goals https://dev.fitbit.com/docs/activity/#update-activity-goals

Pass no arguments to get the daily activities goal. Pass any one of the optional arguments to set that component of the daily activities goal.

Arguments: * `calories_out` – New goal value; in an integer format * `active_minutes` – New goal value; in an integer format * `floors` – New goal value; in an integer format * `distance` – New goal value; in the format X.XX or integer * `steps` – New goal value; in an integer format

activities_list()
https://dev.fitbit.com/docs/activity/#browse-activity-types

activities_weekly_goal(*distance=None*, *floors=None*, *steps=None*)
Implements the following APIs for period equal to weekly

https://dev.fitbit.com/docs/activity/#get-activity-goals https://dev.fitbit.com/docs/activity/#update-activity-goals

Pass no arguments to get the weekly activities goal. Pass any one of the optional arguments to set that component of the weekly activities goal.

Arguments: * `distance` – New goal value; in the format X.XX or integer * `floors` – New goal value; in an integer format * `steps` – New goal value; in an integer format

activity_detail(*activity_id*)
https://dev.fitbit.com/docs/activity/#get-activity-type

activity_stats(*user_id=None*, *qualifier=''*)

- https://dev.fitbit.com/docs/activity/#activity-types
- https://dev.fitbit.com/docs/activity/#get-favorite-activities
- https://dev.fitbit.com/docs/activity/#get-recent-activity-types
- https://dev.fitbit.com/docs/activity/#get-frequent-activities

This implements the following methods:

```
recent_activities(user_id=None, qualifier='')
favorite_activities(user_id=None, qualifier='')
frequent_activities(user_id=None, qualifier='')
```

add_alarm(*device_id*, *alarm_time*, *week_days*, *recurring=False*, *enabled=True*, *label=None*, *snooze_length=None*, *snooze_count=None*, *vibe='DEFAULT'*)
https://dev.fitbit.com/docs/devices/#add-alarm alarm_time should be a timezone aware datetime object.

add_favorite_activity(*activity_id*)
https://dev.fitbit.com/docs/activity/#add-favorite-activity

add_favorite_food(*food_id*)
https://dev.fitbit.com/docs/food-logging/#add-favorite-food

body_fat_goal(*fat=None*)
Implements the following APIs

- https://dev.fitbit.com/docs/body/#get-body-goals
- https://dev.fitbit.com/docs/body/#update-body-fat-goal

Pass no arguments to get the body fat goal. Pass a `fat` argument to update the body fat goal.

Arguments: * `fat` – Target body fat in %; in the format X.XX

body_weight_goal(*start_date=None*, *start_weight=None*, *weight=None*)
Implements the following APIs

- https://dev.fitbit.com/docs/body/#get-body-goals
- https://dev.fitbit.com/docs/body/#update-weight-goal

Pass no arguments to get the body weight goal. Pass `start_date`, `start_weight` and optionally `weight` to set the weight goal. `weight` is required if it hasn't been set yet.

Arguments: * `start_date` – Weight goal start date; in the format yyyy-MM-dd * `start_weight` – Weight goal start weight; in the format X.XX * `weight` – Weight goal target weight; in the format X.XX

create_food(*data*)
https://dev.fitbit.com/docs/food-logging/#create-food

delete_alarm(*device_id*, *alarm_id*)
https://dev.fitbit.com/docs/devices/#delete-alarm

delete_favorite_activity(*activity_id*)
https://dev.fitbit.com/docs/activity/#delete-favorite-activity

delete_favorite_food(*food_id*)
https://dev.fitbit.com/docs/food-logging/#delete-favorite-food

food_detail(*food_id*)
https://dev.fitbit.com/docs/food-logging/#get-food

food_goal(*calories=None*, *intensity=None*, *personalized=None*)
Implements the following APIs

https://dev.fitbit.com/docs/food-logging/#get-food-goals https://dev.fitbit.com/docs/food-logging/#update-food-goal

Pass no arguments to get the food goal. Pass at least `calories` or `intensity` and optionally `personalized` to update the food goal.

Arguments: * `calories` – Manual Calorie Consumption Goal; calories, integer; * `intensity` – Food Plan intensity; (MAINTENANCE, EASIER, MEDIUM, KINDAHARD, HARDER); * `personalized` – Food Plan type; `True` or `False`

food_units()
https://dev.fitbit.com/docs/food-logging/#get-food-units

get_alarms(*device_id*)
https://dev.fitbit.com/docs/devices/#get-alarms

get_badges(*user_id=None*)
https://dev.fitbit.com/docs/friends/#badges

get_bodyfat(*base_date=None*, *user_id=None*, *period=None*, *end_date=None*)
https://dev.fitbit.com/docs/body/#get-body-fat-logs base_date should be a datetime.date object (defaults to today), period can be '1d', '7d', '30d', '1w', '1m', '3m', '6m', '1y', 'max' or None end_date should be a datetime.date object, or None.

You can specify period or end_date, or neither, but not both.

get_bodyweight(*base_date=None*, *user_id=None*, *period=None*, *end_date=None*)
https://dev.fitbit.com/docs/body/#get-weight-logs base_date should be a datetime.date object (defaults to today), period can be '1d', '7d', '30d', '1w', '1m', '3m', '6m', '1y', 'max' or None end_date should be a datetime.date object, or None.

You can specify period or end_date, or neither, but not both.

get_devices()
https://dev.fitbit.com/docs/devices/#get-devices

get_friends(*user_id=None*)
https://dev.fitbit.com/docs/friends/#get-friends

get_friends_leaderboard(*period*)
https://dev.fitbit.com/docs/friends/#get-friends-leaderboard

get_meals()
https://dev.fitbit.com/docs/food-logging/#get-meals

get_sleep(*date*)
https://dev.fitbit.com/docs/sleep/#get-sleep-logs date should be a datetime.date object.

intraday_time_series(*resource*, *base_date='today'*, *detail_level='1min'*, *start_time=None*, *end_time=None*)
The intraday time series extends the functionality of the regular time series, but returning data at a more granular level for a single day, defaulting to 1 minute intervals. To access this feature, one must fill out the Private Support form here (see https://dev.fitbit.com/docs/help/). For details on the resources available and more information on how to get access, see:

https://dev.fitbit.com/docs/activity/#get-activity-intraday-time-series

invite_friend(*data*)
https://dev.fitbit.com/docs/friends/#invite-friend

invite_friend_by_email(*email*)
Convenience Method for https://dev.fitbit.com/docs/friends/#invite-friend

invite_friend_by_userid(*user_id*)
Convenience Method for https://dev.fitbit.com/docs/friends/#invite-friend

list_subscriptions(*collection=''*)
https://dev.fitbit.com/docs/subscriptions/#getting-a-list-of-subscriptions

log_activity(*data*)
https://dev.fitbit.com/docs/activity/#log-activity

log_sleep(*start_time*, *duration*)
https://dev.fitbit.com/docs/sleep/#log-sleep start time should be a datetime object. We will be using the year, month, day, hour, and minute.

reject_invite(*other_user_id*)
Convenience method for respond_to_invite

respond_to_invite(*other_user_id*, *accept=True*)
https://dev.fitbit.com/docs/friends/#respond-to-friend-invitation

search_foods(*query*)
https://dev.fitbit.com/docs/food-logging/#search-foods

subscription(*subscription_id*, *subscriber_id*, *collection=None*, *method='POST'*)
https://dev.fitbit.com/docs/subscriptions/

time_series(*resource*, *user_id=None*, *base_date='today'*, *period=None*, *end_date=None*)
The time series is a LOT of methods, (documented at urls below) so they don't get their own method. They all follow the same patterns, and return similar formats.

Taking liberty, this assumes a base_date of today, the current user, and a 1d period.

https://dev.fitbit.com/docs/activity/#activity-time-series https://dev.fitbit.com/docs/body/#body-time-series https://dev.fitbit.com/docs/food-logging/#food-or-water-time-series https://dev.fitbit.com/docs/heart-rate/#heart-rate-time-series https://dev.fitbit.com/docs/sleep/#sleep-time-series

update_alarm(*device_id*, *alarm_id*, *alarm_time*, *week_days*, *recurring=False*, *enabled=True*, *label=None*, *snooze_length=None*, *snooze_count=None*, *vibe='DEFAULT'*)
https://dev.fitbit.com/docs/devices/#update-alarm alarm_time should be a timezone aware datetime object.

user_profile_get(*user_id=None*)
Get a user profile. You can get other user's profile information by passing user_id, or you can get the current user's by not passing a user_id

https://dev.fitbit.com/docs/user/

user_profile_update(*data*)
Set a user profile. You can set your user profile information by passing a dictionary of attributes that will be updated.

https://dev.fitbit.com/docs/user/#update-profile

water_goal(*target=None*)
Implements the following APIs

https://dev.fitbit.com/docs/food-logging/#get-water-goal https://dev.fitbit.com/docs/food-logging/#update-water-goal

Pass no arguments to get the water goal. Pass `target` to update it.

Arguments: * `target` – Target water goal in the format X.X, will be set in unit based on locale

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

T

U

W