

Follow

<https://www.linkedin.com/in/stephenjhsu/>

Jan 9 • 5 min read

Collect Your Own Fitbit Data with Python

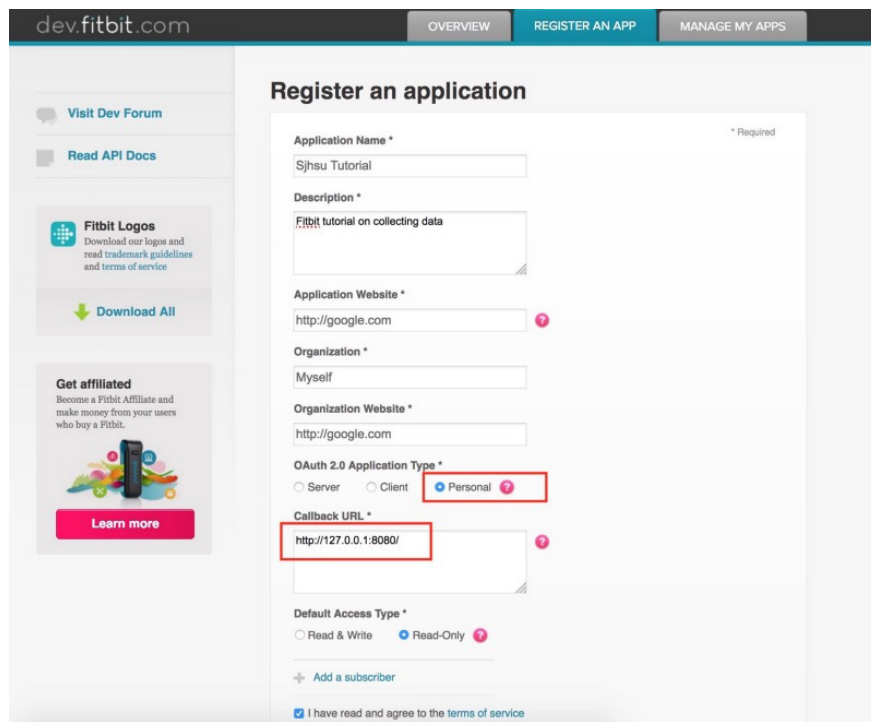


Python Generated Fitbit Cloud

So you've got your Fitbit over the Christmas break and you've got some New Years Resolutions. You go online and see the graphs on your dashboard but you're still not pleased. You want more data, more graphs, and more information. Well say no more, because I'm going to teach you how to collect your own Fitbit data using nothing but a little Python code. With this tutorial, you can get your elusive minute by minute data (also known as intraday data), which is not readily available when you first get your Fitbit.

Step 1: Set up your account and create the app

The first thing you'll need to do is create a [Fitbit account](#). Once you've done that, you can go to [dev.fitbit.com](#). Under "Manage", go to "Register An App". This will lead you to a page that looks like:



The screenshot shows the 'dev.fitbit.com' website with a navigation bar containing 'OVERVIEW', 'REGISTER AN APP', and 'MANAGE MY APPS'. The main content area is titled 'Register an application'. On the left sidebar, there are links for 'Visit Dev Forum', 'Read API Docs', 'Fitbit Logos' (with a 'Download All' button), and 'Get affiliated' (with a 'Learn more' button). The registration form includes the following fields:

- Application Name ***: Sjsu Tutorial
- Description ***: Fitbit tutorial on collecting data
- Application Website ***: http://google.com
- Organization ***: Myself
- Organization Website ***: http://google.com
- OAuth 2.0 Application Type ***: Radio buttons for Server, Client, and Personal (selected).
- Callback URL ***: http://127.0.0.1:8080/
- Default Access Type ***: Radio buttons for Read & Write and Read-Only (selected).

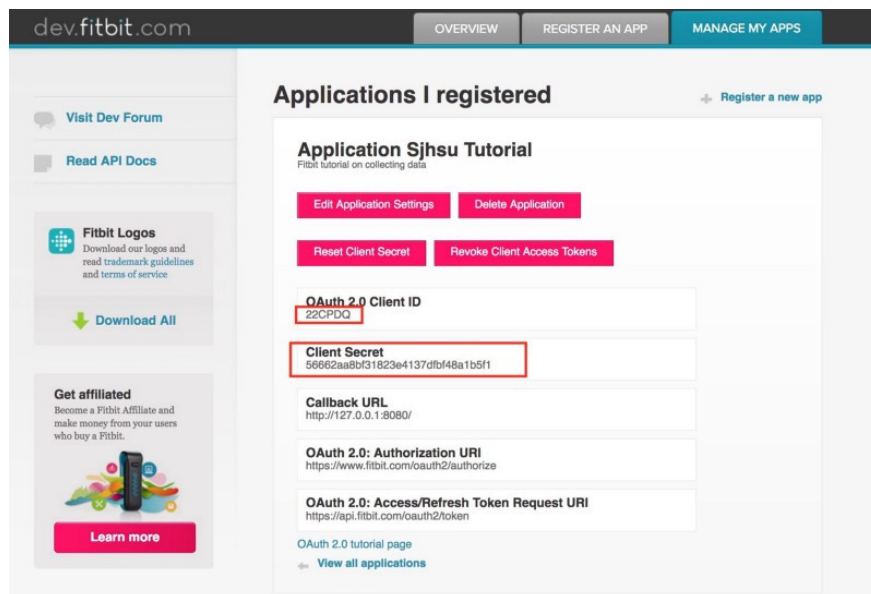
At the bottom of the form, there is a link 'Add a subscriber' and a checkbox 'I have read and agree to the terms of service' which is checked.

Application Registration

For the application website and organization website, name it anything starting with “http://” or “https://”. Secondly, make sure the OAuth 2.0 Application Type is “Personal” as this is key to allowing us to download our intraday data. Lastly, make sure the Callback URL is “http://127.0.0.1:8080/” in order to get our Fitbit API to connect properly. After that, click on the agreement box and submit.

NOTE: depending on the app, we may need an additional step to fill out a form in order to gain permission to our intraday data at this link. Fitbit is supportive of personal projects and any other non profit research, so these should already be callable, but commercial apps might take longer to be approved.

After that, you’ll be redirected to a page looking like this:



Registered Application

The parts we will need from this page are the OAuth 2.0 Client ID and the Client Secret.

Step 2: The API

Once Step 1 is completed, our next step is to use a [Fitbit unofficial API](#). Click the green button on the right side to download the repo and afterwards unzip the file. After that, open up the command line to change directories to the directory containing the unzipped files and run a quick 'sudo pip install -r requirements/base.txt'.

Now we're finally ready to start coding. First things first, we will need to import the necessary packages and bring in our Client_ID and Client_Secret from earlier to authorize ourselves.

```
import fitbit
import gather_keys_oauth2 as OAuth2
import pandas as pd
import datetime

CLIENT_ID = '22CPDQ'
CLIENT_SECRET = '56662aa8bf31823e4137dfbf48a1b5f1'
```

Using the ID and Secret, we can obtain the access and refresh tokens that authorize us to get our data.

```
server = OAuth2.OAuth2Server(CLIENT_ID, CLIENT_SECRET)
server.browser_authorize()

ACCESS_TOKEN =
str(server.fitbit.client.session.token['access_token'])
REFRESH_TOKEN =
str(server.fitbit.client.session.token['refresh_token'])

auth2_client = fitbit.Fitbit(CLIENT_ID, CLIENT_SECRET,
oauth2=True, access_token=ACCESS_TOKEN,
refresh_token=REFRESH_TOKEN)
```



Sjhsu Tutorial by **Myself** would like the ability to access the following data in your Fitbit account

Warning! This app is not using HTTPS to securely obtain your permission.

- ☒ food and water logs ⓘ
- ☒ location and GPS
- ☒ sleep
- ☒ Fitbit devices and settings
- ☒ friends ⓘ
- ☒ activity and exercise
- ☒ weight ⓘ
- ☒ heart rate
- ☒ profile ⓘ

Deny

Allow

Data shared with Sjhsu Tutorial will be governed by Myself's privacy policy and terms of service. You can revoke this consent at any time in your Fitbit [account settings](#). More information about these permissions can be found [here](#).

What the page should look like after authorization and signing in

We're in! ... but not so fast. We've authenticated ourselves but we still haven't gotten our data. Before jumping in, I have one handy trick that will save a lot of manual typing: many of the calls require a date format of (YYYY-MM-DD) as a string format. So if you were to collect your data daily, we'll have to change this string manually everyday. Instead, we can import the useful 'datetime' package and let it do the formatting for us instead.

```

yesterday = str((datetime.datetime.now() -
datetime.timedelta(days=1)).strftime("%Y%m%d"))

yesterday2 = str((datetime.datetime.now() -
datetime.timedelta(days=1)).strftime("%Y-%m-%d"))

today = str(datetime.datetime.now().strftime("%Y%m%d"))

```

Step 3: Acquire the data

We can finally now call for our data.

```

fit_statsHR =
auth2_client.intraday_time_series('activities/heart',
base_date=yesterday2, detail_level='1sec')

```

```

: fitbitHR['activities-heart-intraday']
: {u'dataset': [{u'time': u'00:00:08', u'value': 66},
  {u'time': u'00:00:13', u'value': 65},
  {u'time': u'00:00:23', u'value': 65},
  {u'time': u'00:00:28', u'value': 65},
  {u'time': u'00:00:38', u'value': 66},
  {u'time': u'00:00:43', u'value': 65},
  {u'time': u'00:00:53', u'value': 64},
  {u'time': u'00:00:58', u'value': 63},
  {u'time': u'00:01:08', u'value': 62},
  {u'time': u'00:01:18', u'value': 63},
  {u'time': u'00:01:23', u'value': 64},
  {u'time': u'00:01:33', u'value': 65},
  {u'time': u'00:01:38', u'value': 68},
  {u'time': u'00:01:48', u'value': 66},
  {u'time': u'00:02:03', u'value': 65},
  {u'time': u'00:02:08', u'value': 66},
  {u'time': u'00:02:13', u'value': 65},
  {u'time': u'00:02:18', u'value': 64},
  {u'time': u'00:02:23', u'value': 65},
  {u'time': u'00:02:28', u'value': 64}]}

```

Heart Rate Data

While this data looks readable to us, it's not yet ready to be saved on our computer. The following bit of code reads the dictionary format and iterates through the dictionary values—saving the respective time and value values as lists before combining both into a pandas data frame.

```

time_list = []
val_list = []

for i in fit_statsHR['activities-heart-intraday']
['dataset']:

```

```
val_list.append(i['value'])
time_list.append(i['time'])

heartdf = pd.DataFrame({'Heart
Rate':val_list,'Time':time_list})
```

```
heartdf = pd.DataFrame({'Heart Rate':val_list,'Time':time_list})
```

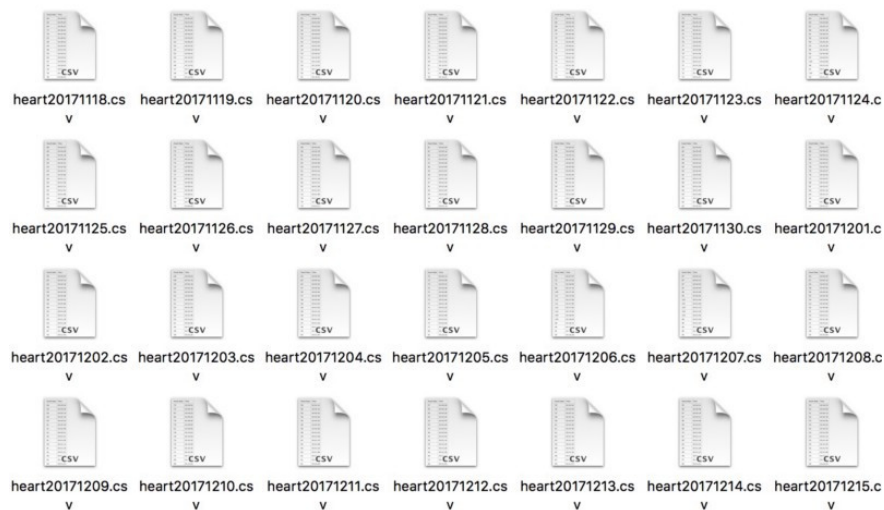
```
heartdf
```

	Heart Rate	Time
0	93	00:00:00
1	93	00:01:00
2	77	00:02:00
3	79	00:03:00
4	84	00:04:00
5	76	00:05:00
6	65	00:06:00

Our Newly Created Fitbit pandas data frame

Now to finally save our data locally. Thinking ahead, we'll be calling our data about once a data (1 file = 1 day), so we'll want to have a good naming convention that prevents any future mixups or overriding data. My preferred format is to go with heartYYMMDD.csv. The following code takes our heart data frame and saves them as a CSV in the /Downloads/python-fitbit-master/Heart/ directory.

```
heartdf.to_csv('/Users/shsu/Downloads/python-fitbit-
master/Heart/heart'+ \
               yesterday+'.csv', \
               columns=['Time','Heart Rate'], header=True, \
               index = False)
```



Small army of heart rate files

But wait! There's still more!

Reading the [documentation](#), there's plenty of other data we can still collect like:

Our sleep log data:

```

"""Sleep data on the night of ...."""

fit_stats1 = auth2_client.sleep(date='today')
stime_list = []
sval_list = []

for i in fit_stats1['sleep'][0]['minuteData']:
    stime_list.append(i['dateTime'])
    sval_list.append(i['value'])

sleepdf = pd.DataFrame({'State':sval_list,
                        'Time':stime_list})

sleepdf['Interpreted'] =
sleepdf['State'].map({'2':'Awake','3':'Very
Awake','1':'Asleep'})
sleepdf.to_csv('/Users/shsu/Downloads/python-fitbit-
master/Sleep/sleep' + \
               today+'.csv', \
               columns =
['Time','State','Interpreted'],header=True,
               index = False)

```

	A	B	C	D	E	F	G
1	State	Time	Interpreted				
2	1	0:27:00	Asleep				
3	1	0:28:00	Asleep				
4	1	0:29:00	Asleep				
5	1	0:30:00	Asleep				
6	2	0:31:00	Awake				
7	1	0:32:00	Asleep				
8	1	0:33:00	Asleep				
9	1	0:34:00	Asleep				

Sleep Log

Or our sleep summary:

```

"""Sleep Summary on the night of ...."""

fit_statsSum = auth2_client.sleep(date='today')['sleep'][0]

ssummarydf =
pd.DataFrame({'Date':fit_statsSum['dateOfSleep'],
              'MainSleep':fit_statsSum['isMainSleep'],
              'Efficiency':fit_statsSum['efficiency'],
              'Duration':fit_statsSum['duration'],
              'Minutes
Asleep':fit_statsSum['minutesAsleep'],
              'Minutes Awake':fit_statsSum['minutesAwake'],
              'Awakenings':fit_statsSum['awakeCount'],
              'Restless
Count':fit_statsSum['restlessCount'],
              'Restless
Duration':fit_statsSum['restlessDuration'],
              'Time in Bed':fit_statsSum['timeInBed']
              }, index=[0])

```


	A	B	C	D	E	F	G	H	I	J
1	Awakenings	Date	Duration	Efficiency	MainSleep	Minutes Asle	Minutes Aw	Restless Cou	Restless Duri	Time in Bed
2	2	11/18/17	26400000	92	TRUE	405	35	15	26	440
3	0	11/19/17	29040000	96	TRUE	465	19	16	19	484
4	1	11/20/17	28440000	96	TRUE	453	21	11	17	474
5	0	11/21/17	26280000	91	TRUE	400	38	18	38	438
6	2	11/22/17	25320000	93	TRUE	391	31	17	29	422
7	4	11/23/17	24180000	94	TRUE	377	26	12	19	403
8	1	11/24/17	29460000	97	TRUE	468	14	8	19	491

Sleep Summary

And that's all you need to get started on collecting all your Fitbit data! So play around some more, read the Python Fitbit documentation, get lost in it for a whole, find your way out, and see overall how nifty your heart data is. In case you wanted a feel for what you can make with that data, here is a [link](#) to one of my analyses of my own heart data.

Thanks for taking the time to read my tutorial and feel free to leave a comment or connect on [LinkedIn](#) as I'll be posting more tutorials on data mining and data science.

References:

1. [Orcas Fitbit API Github](#)
2. [Python-Fitbit documentation](#)
3. [Official Fitbit API documentation](#)
4. [Fitbit Intraday Access Form](#)
5. [Personal Github with all the code](#)
6. [Pandas data frame documentation](#)

