CFG Advanced Python Course - Session 4 (Summer Intensive 2015)
Using external APIs

APIs
APIs are of vital importance to developers; they enable you to programmatically access data from various different services such as Twitter, Facebook and Spotify to name a few. APIs enable you to create applications powered by data from any service of your liking.

Mandrill
When building a website, for example a landing page for what is to come, one vital thing to do is capture the details of people that might be interested in what you are going to launch.

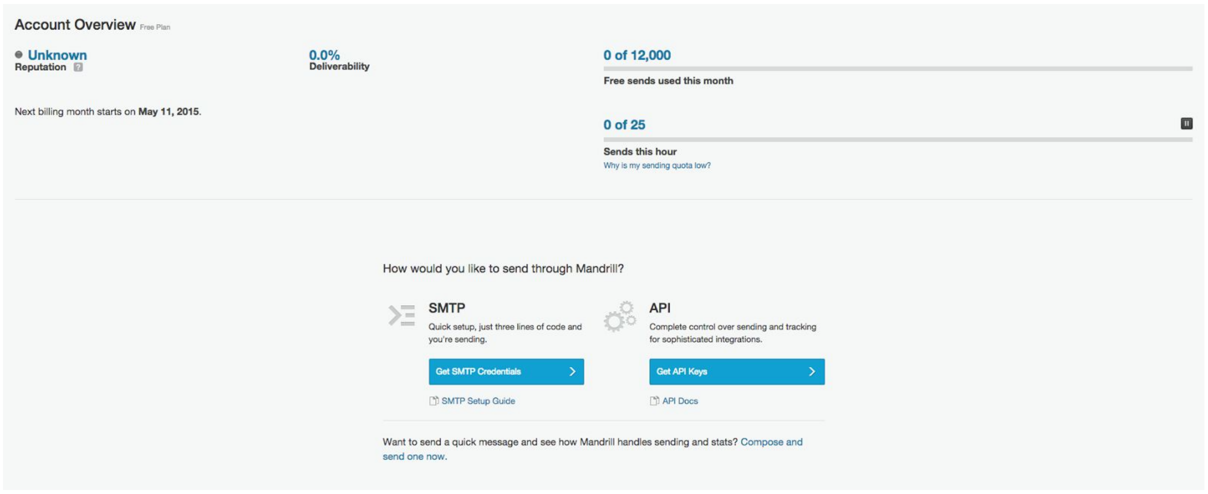
Most importantly, you want to capture their email address so you can communicate things to them. In this session we will learn how to capture a user's e-mail address (and any other details for that matter) and programmatically send them an email to confirm that they have signed up to your awesome website's newsletter!

This will be an interactive session based on what we did in **Session 3**. if you look at what we did during that session under the **GET & POST** section, you will see that we already learnt how to capture a user's name and e-mail and print it out on our terminal. Now we are going to use that data to send an e-mail.

We will be using Mandrill, an email infrastructure service - or in simpler terms, a service that enables you to programmatically send e-mails. Mandrill has a Python library that makes it easy to send e-mails, so go ahead and head over to their Python documentation and follow the instructions on how to install the Python client and get started.

You should all already have a Mandrill account, as this was your homework from last session. If not, go ahead and create an account. If you get an error message like 'too many people are trying to create an account from the same location', then wait a little bit; this is Mandrill's way to safeguard against spammers creating accounts programmatically to send spam e-mail.

Once you log into your Mandrill account, you should see a page like the following:



Account Overview Free Plan

● Unknown
Reputation

0.0%
Deliverability

0 of 12,000
Free sends used this month

Next billing month starts on **May 11, 2015**.

0 of 25
Sends this hour
Why is my sending quota low?

How would you like to send through Mandrill?

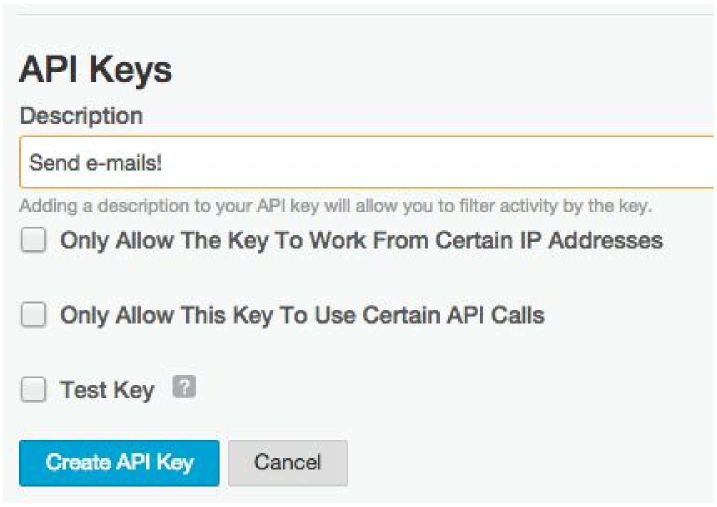
SMTP
Quick setup, just three lines of code and you're sending.
Get SMTP Credentials
SMTP Setup Guide

API
Complete control over sending and tracking for sophisticated integrations.
Get API Keys
API Docs

Want to send a quick message and see how Mandrill handles sending and stats? Compose and send one now.

Hurray, Mandrill is giving us **12,000 free emails per month**!

In order to programmatically send e-mails, we need to get an API guy, so go ahead and click on **Get API Keys**. On the next page that you are presented with, click on **+ New API Key**. You should see a simple form like this:



API Keys

Description

Send e-mails!

Adding a description to your API key will allow you to filter activity by the key.

☐ Only Allow The Key To Work From Certain IP Addresses

☐ Only Allow This Key To Use Certain API Calls

☐ Test Key

Add a description to make it easier to track your API Keys and leave the rest of the boxes unchecked.

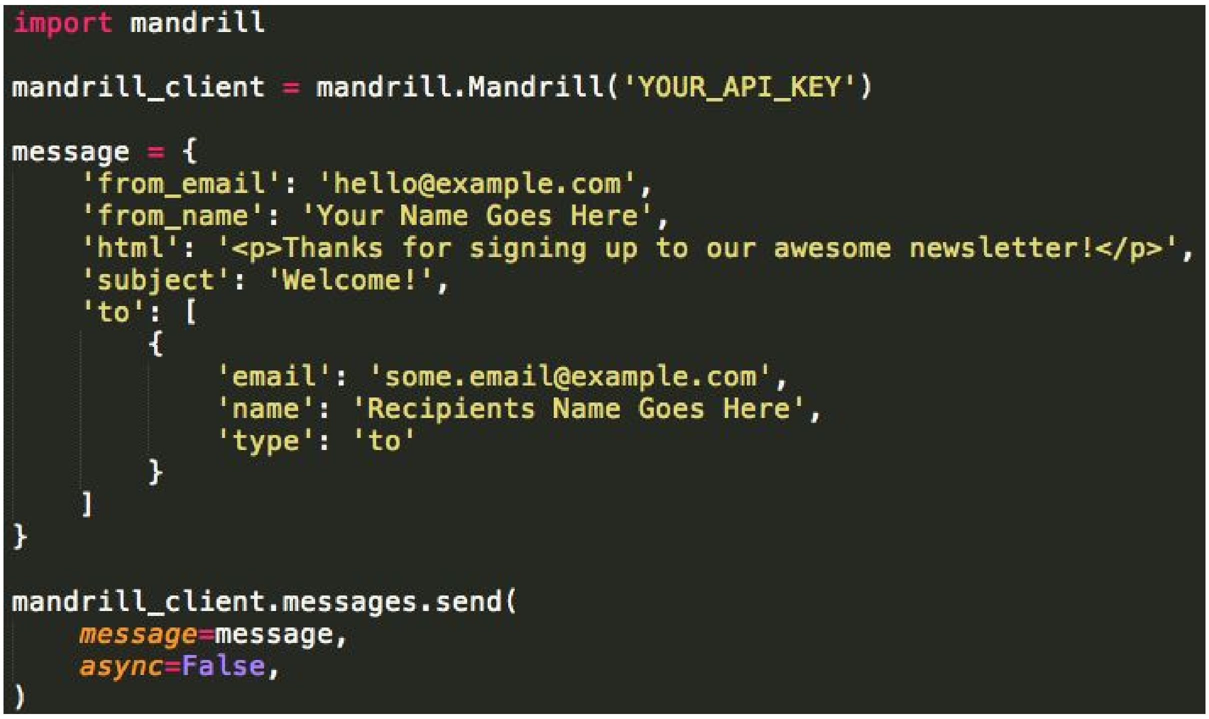
You might wonder, why would you ever want to create multiple API Keys. This is especially useful when your website grows and you start sending e-mails for different functions; registration emails, newsletters etc. In those cases, you would want to use a different API Key for each function to make life easier when managing what each component does.

After adding a description, hit **Create API Key** - you should now see a key listed! You will notice that there is a little On/Off toggle to the right of the key - ensure that is **On**:

If you now go back to Mandrill's Python documentation, you can see instructions on how to use the library - to authenticate yourself with Mandrill via Python this is as simple as writing two lines of code:

```
import mandrill
mandrill_client = mandrill.Mandrill('YOUR_API_KEY')
```

Next up, we want to actually send an e-mail, so we can look at the this part of the documentation which you can access from the left hand side of the Python documentation. The most minimal way to send an e-mail would be the following:



```
import mandrill

mandrill_client = mandrill.Mandrill('YOUR_API_KEY')

message = {
    'from_email': 'hello@example.com',
    'from_name': 'Your Name Goes Here',
    'html': '<p>Thanks for signing up to our awesome newsletter!</p>',
    'subject': 'Welcome!',
    'to': [
        {
            'email': 'some.email@example.com',
            'name': 'Recipients Name Goes Here',
            'type': 'to'
        }
    ]
}

mandrill_client.messages.send(
    message=message,
    async=False,
)
```

This example is completely independent from a Flask app, so what you should do now is figure out how to apply the above code in the context of your Flask app that we created over the past few sessions.

Ensure that you are sending an email to a valid email address, set your own e-mail address as the **from_email** and use the email and name submitted via the form that we were previously printing.

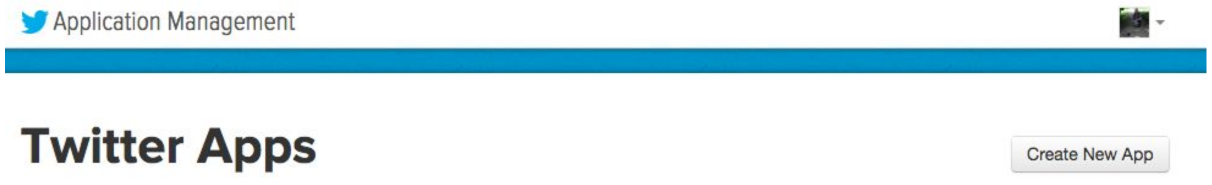
Read through the documentation and figure out what other things you can do with Mandrill - Can you send an email to multiple recipients? Can you add attachments?

Twitter
Twitter provides a set of REST APIs in order to programmatically access, read & write Twitter data. You can access Twitter's REST APIs in many different ways; there are a whole bunch of libraries for any programming language you can imagine. In the context of this course, we will be using tweepy, a Python library enabling programmatic access to Twitter's REST APIs.

As we all know by now, installing a Python library such as tweepy is as simple as doing **pip install tweepy**. Mac users, as always, you might need to add a **sudo** at the start of that command.

As per last session's homework, you should all have a Twitter account, so go ahead and head over to apps.twitter.com; log in with your Twitter account if you are not already logged in.

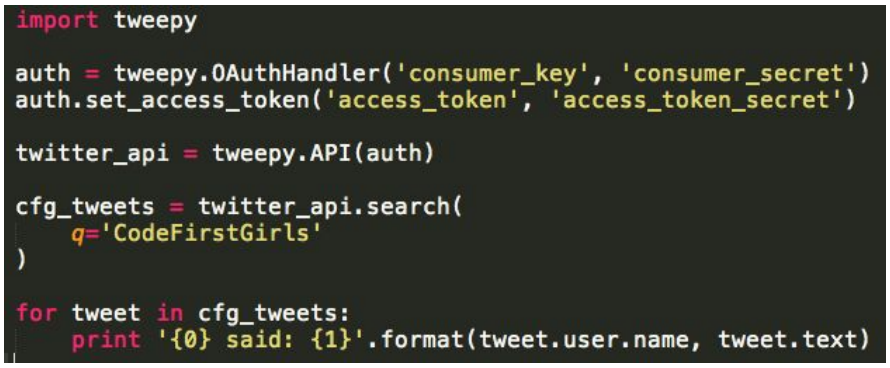
Once logged in, you should see the following:



We will create a new application, so click on the **Create New App** button.

You will now be presented with a form that you need to fill in with all your application details; fill in the name, description and website fields. Before pressing the **Create your Twitter application** button, ensure you select that you agree to the Developer Agreement.

Once you do that, if everything works out fine you will be taken to your application page where you can see some basic information. Click on the **Keys and Access Tokens** tab, where you will get a chance to generate a consumer key & secret as well as an access token & secret. We will use these keys and secrets in our Python code to authenticate ourselves with Twitter so that it knows we have a genuine application to access data. Let's try get some Twitter data using Tweepy, for example let's get Tweets that mention CodeFirstGirls:



```python
import tweepy

auth = tweepy.OAuthHandler('consumer_key', 'consumer_secret')
auth.set_access_token('access_token', 'access_token_secret')

twitter_api = tweepy.API(auth)

cfg_tweets = twitter_api.search(
    q='CodeFirstGirls'
)

for tweet in cfg_tweets:
    print '{0} said: {1}'.format(tweet.user.name, tweet.text)
```

That's cool right? That's just a small glimpse of what you can actually do using Twitter's API! You can look at tweepy's API reference to discover some more things that you can do with it.

Task

- Get information related to the CodeFirstGirls Twitter user
- Get a list of all followers for the CodeFirstGirls Twitter user
- Figure out how to post a tweet using tweepy