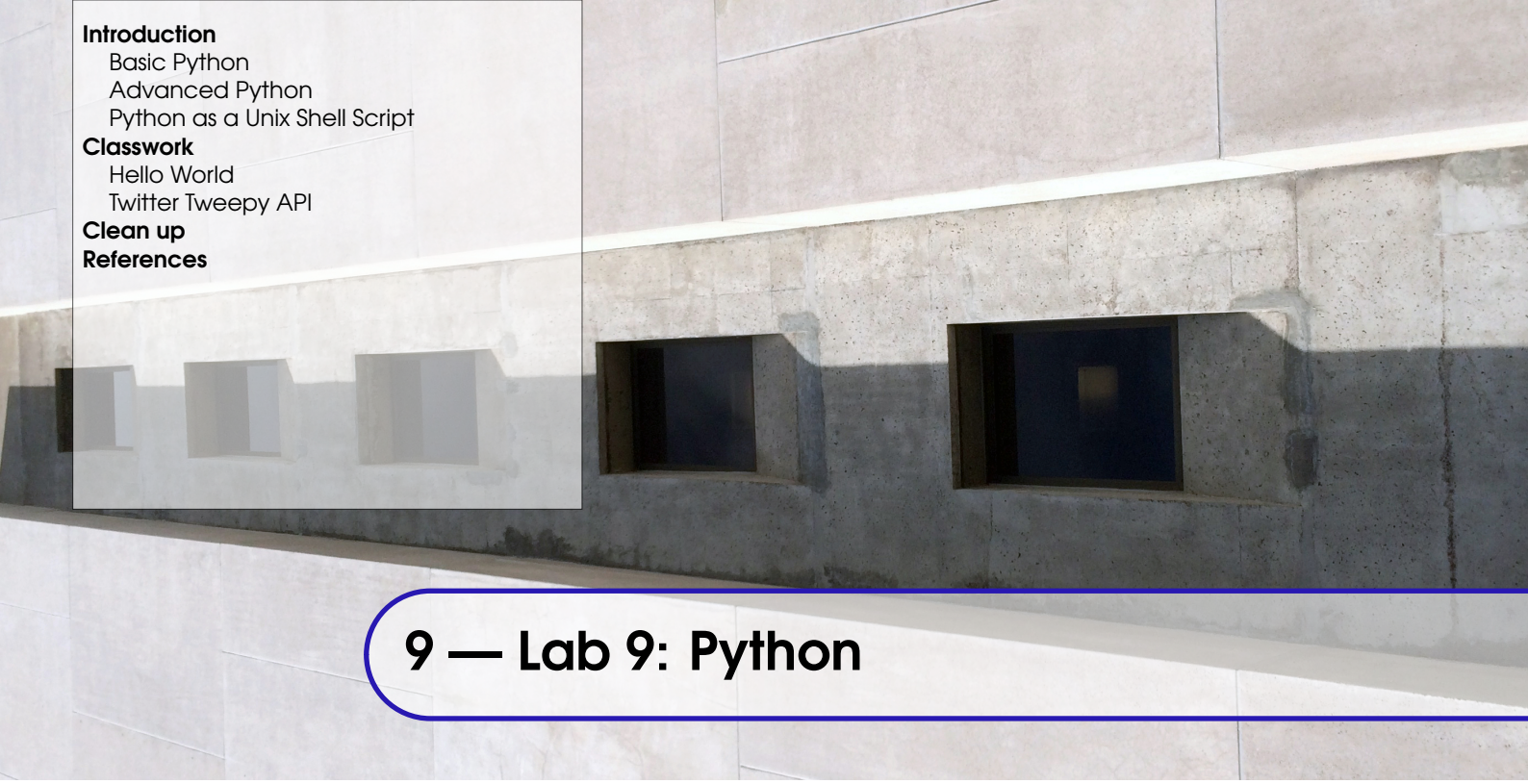

9 — Lab 9: Python

9.1 Introduction

For the last few labs, you have been working with web technologies. For today's class, you'll return to using your unix shell account to create a script to run from the unix command line.

9.1.1 Basic Python

If you have never used Python before, go to the Python section of Codeacademy (`http://www.codecademy.com/tracks/python`). Start from Python Syntax section, and work your way through the Exam Statistics section. If you feel comfortable with some of these topics, you may skip forward until you find the level you feel comfortable beginning. Continue at Python as Unix Script, below.

9.1.2 Advanced Python

At Codeacademy (`http://www.codecademy.com/tracks/python`) start the tutorial at the Advanced Topics in Python section and work forward until the end of the tutorials. If you feel you need reinforcement for any concept you encounter, go back and review the subject in the earlier tutorials on this site.

9.1.3 Python as a Unix Shell Script

Unix Shebang and Python

In the unix shell, it is possible to create text files that the shell will interpret as programs. To tell the shell that the file is a program, on the very first line of the file, you must put a sharp #) and a bang (!) (sharp and bang are old programming jargon). Combining Sharp and BANG gives us SheBANG, pronounced she-bang (like the song). In your shell script, the line containing the shebang must be the first line of your file. On the same line, following the #!, the location of the interpreter that will run the script is specified. At the time of execution, the #! line is read, interpreted, then thrown out, and the rest of the file is sent to the interpreter to be executed[1].

On thecity, the Python interpreter is located at /usr/bin/python. Therefore, to create a Python script on thecity, the very first line (no blank lines before) must read `#!  /usr/bin/python`

[1] For the gory shebang details, see `http://en.wikipedia.org/wiki/Shebang_(Unix)`.

Permission to Execute

The last thing that you must make sure is that the file that you wish to execute as a program from the unix shell command line is flagged as an executable file. To do this, you can type `chmod a+x filename`, where *filename* is the name of the file you wish to make executable.

9.2 Classwork

9.2.1 Hello World

In your unix shell on thecity, using your favorite editor, create a Python program named hello.py. It should simply print `hello world` to the console.

9.2.2 Twitter Tweepy API

Introduction

For lab nine you will be learning a little bit of python while gaining some experience with a Python library used to access the Twitter API. The Python Library we will use will be Tweepy. Now unfortunately we cannot do this work on our own thecity server as it does not have the correct packages to run the Tweepy library. Therefore, for this lab you will be working on a Amazon Web Services Server that has been made for you. You will be connecting to it the same way you do to connect to the thecity server. The host name is 52.8.87.226, your *username* will be the first part of your sfsu email address (i.e *username*@mail.sfsu.edu), and the password will be you SFSU student ID number. You can do this either through `ssh` or Putty.

Creating Twitter Account

If you already have an account you may skip to the next section where we will create a Twitter application.

In order to collect tweets from a Twitter stream we must obtain access keys for the Twitter API. To obtain access keys we must first have a Twitter account. To do this please navigate to `https://twitter.com` and make an account. Unfortunately, you will need attach a mobile number to access the Twitter API. When doing so you will receive a text with a verification code. Enter this code when prompted. **This will/can be removed later in the clean up section.** You may skip some steps after creating an account, But you should confirm your email address when asked in order to make the Twitter Application process easier.

Creating Twitter Dev Application

Now that we have a working Twitter account, we need to create a Twitter application to obtain access keys.

1. While logged into your Twitter account navigate to `https://apps.twitter.com/app/new`.
2. On this page fill out the Application Details. For Name and Description please provide you own information. For website, you may enter your SFSU website or just simply type a place holder(https://www.placeholdercsc412.com). Leave the call back URL blank. Make sure to accept the Twitter Development Agreement at the bottom.

3. After clicking Create your twitter Application button a the bottom, you should be redirected to the following page:

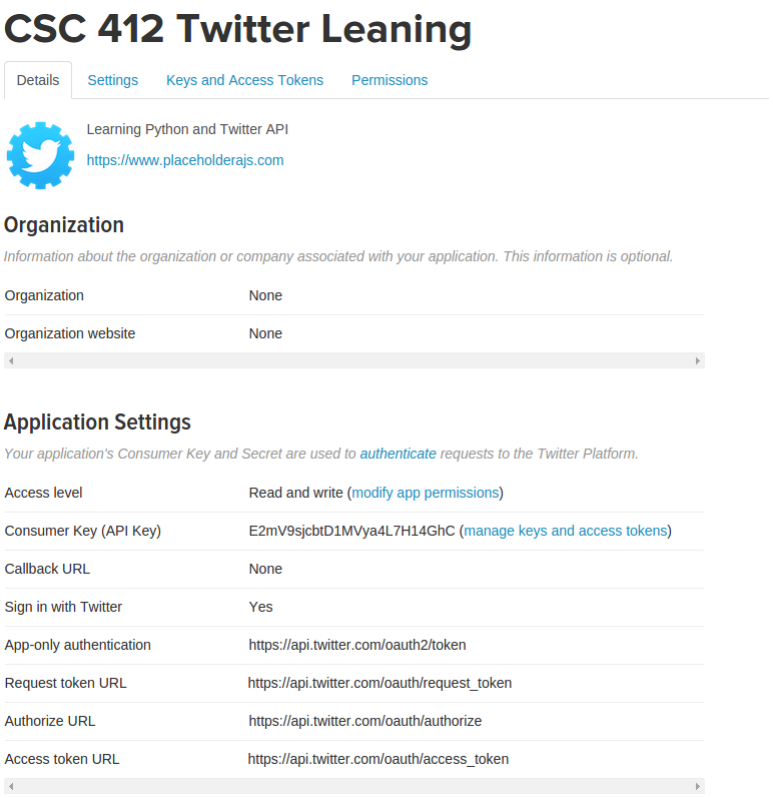

Figure 9.1: Center of Page of Creating Twitter App

4. Next click on the tab labeled *Keys and Access Tokens*.
5. Near the bottom click the button Create my access token.
6. Take note of the following four values:
   - Consumer Key
   - Consumer Secret
   - Access Token
   - Access Token Secret

   We will be using these keys to make requests to the Twitter API.

Testing Creation of Twitter Application

Next we will run a quick python script that will test whether or not the previous steps were successful.

1. First log into the AWS server that has been created. The hostname is : 52.8.87.226. The *username* will be the first part of your sfsu email address (i.e *username*@mail.sfsu.edu). Password will be your student ID.
2. With your favorite text editor, open a file name *twitterStream.py*

   ```
   vim twitterStream.py ⏎
   ```

3. Next type the following lines of code into the python file:

```
1 import tweepy
2
3 consumer_key = 'your API key'
4 consumer_key_secret = 'your API key secret'
```

```
access_token = 'you Access Token'
access_token_secret = 'your Access Token Secret'

key = tweepy.OAuthHandler(consumer_key,consumer_key_secret)
key.set_access_token(access_token,access_token_secret)

api = tweepy.API(key)
print api.me().name
```

You will need to to replace the values for the variables consumer_key, consumer_key_secret, access_token, access_token_secret with they keys from your twitter application page. Picture below. Use YOUR KEYS.

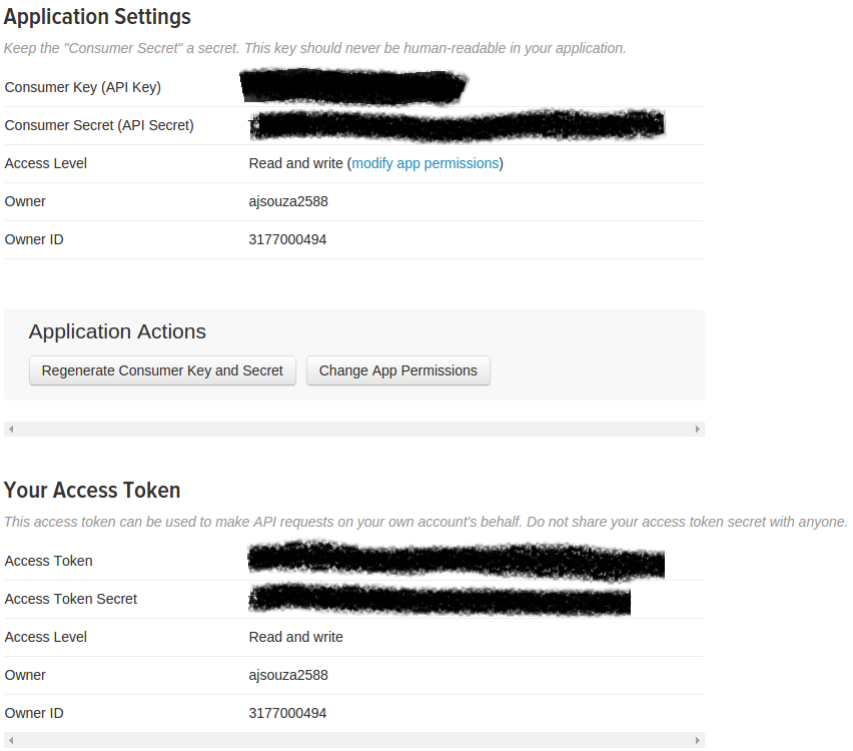

Figure 9.2: Twitter API Access Keys

4. Next run run the python script with the following command:

```
python twitterStream.py
```

5. If everything works correctly you should see your name printed.(At least the full name you supplied when you created your twitter account).
6. Make sure to keep this file, we will be adding code to it in the next section.

If errors occur make sure to check the key values, make sure they are typed correctly. Make sure your python is formatted correctly, remember in Python, one extra space randomly in your file can cause the script not to run.

Simple Python Script to Record Tweets

Next we are going to modify the previously created file *twitterStream.py* and add a few more lines of code. Once we are done and we have no errors, what we should see are tweets printed to the console.

1. First, open the file *twitterStream.py*.
2. Delete the follow lines of code:

```
key = tweepy.OAuthHandler(consumer_key, consumer_key_secret)
key.set_access_token(access_token, access_token_secret)

api = tweepy.API(key)
print api.me().name
```

3. Next, modify the code so it resembles the code snippet below:

```
import tweepy

## You can learn more about OAuth here: https://dev.twitter.com/docs/
    auth/oauth
consumer_key = 'your API key'
consumer_key_secret = 'your API key secret'
access_token = 'you Access Token'
access_token_secret = 'your Access Token Secret'

class TweetListener(tweepy.StreamListener):
  def on_status(self, status):
    print('Tweet text: ' + status.text)
    return True

  def on_error(self, status_code):
    print('Got an error with status code: ' + str(status_code))
    return True

  def on_timeout(self):
    print('Timeout..')
    return True
```

- class TweetListener(tweepy.StreamListener): , this is the class we are going to use that will allow us to receive tweets from Twitter. The TweetListener class takes tweepy.StreamListener as its argument. This allows us to inherit methods from the StreamListener class. We will be overriding some of the methods in StreamListener so they do what we need.
- on_status

```
def on_status(self, status):
    print('Tweet text: ' + status.text)
    return True
```

  The on_status method we have defined is similar to action listener in Java. Once a certain event happens, this method will be executed. In our case, anytime a status update happens on things we are following this will print the tweet to the screen.
- on_error

```
def on_error(self, status_code):
  print('Got an error with status code: ' + str(status_code))
  return True
```

  The on_error method will only execute when an error has occurred while the Stream is active. This for debug purposes.
- on_timeout

```
def on_timeout(self):
    print('Timeout..')
    return True
```

  The on_timeout method is called when the stream connection has timed out. In order to keep the stream alive we return true. This will allow us to run the stream for long time intervals.

These methods were inherited from StreamListener. Make sure they are define in the class TwitterListener with the correct indentation. In StreamListener, these functions actually do not contain any code besides a return statement.

4. Finally, add the code snippet to the file. Note that this code is not in the scope of the class. So do not indent the if statement when typing it.

```
if __name__ == '__main__':
  listener = TweetListener()
  auth = tweepy.OAuthHandler(consumer_key, consumer_key_secret)
  auth.set_access_token(access_token, access_token_secret)

  stream = tweepy.Stream(auth, listener)
  stream.filter(follow=[], track=['#Warriors'])
```

This code snippet will be our "main" method. Even though python does not have main methods, the Python interpreter will define some special labels. Since we are executing our python script as a standalone file, it is given the name('__name__') '__main__'. If this script was imported into another file its name would be different. In most cases its name would be the name of the module itself. What this allows us to do is, if we wanted to run this as a standalone file we can, and the code inside the if body will run. But if what we wanted to use it as piece of a larger application, this code snippet would be skipped. The code does the following things:

(a) Create an instance of the TwitterListener class.
(b) Setup the authentication with our given keys to allow us access to the twitter API.
(c) Creates a Stream with our auth instance and our TwitterListener instance
(d) The last statement filters our stream. In our case we only collect tweets that have the hashtag (#Warriors). You can use the track list to track certain hashtags. You can use the follow list to follow to certain users. You are welcome to change these if you wish to follow or collect tweets from something else. Not that following something that has low traffic(unpopular) may make it look the python script is not working.

In the end the file should look like the following:

```
import tweepy

## You can learn more about OAuth here: https://dev.twitter.com/docs/
    auth/oauth
consumer_key = 'your API key'
consumer_key_secret = 'your API key secret'
access_token = 'you Access Token'
access_token_secret = 'your Access Token Secret'


class TweetListener(tweepy.StreamListener):
  def on_status(self, status):
    print('Tweet text: ' + status.text)
    return True

  def on_error(self, status_code):
    print('Got an error with status code: ' + str(status_code))
    return True

  def on_timeout(self):
    print('Timeout..')
    return True

if __name__ == '__main__':
  listener = TweetListener()
  auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
```

```
26  auth.set_access_token(access_token, access_token_secret)
27
28  stream = tweepy.Stream(auth, listener)
29  stream.filter(follow=[], track=['#Warriors'])
```

Run the script with the following command:

```
python twitterStream.py
```

A quick way to see the script working is to track a specific hashtag. For exmaple track #csc412. Then go to your twitter account and post a tweet with that hashtag. Almost instantly, you will see your tweet pop up in your console. Note: you need to run the python script before posting the tweet.

9.3 Clean up

You are welcomed to keep you twitter application after the lab. If you do, you can skip this section.

If you want to remove your mobile number from the twitter account:

1. log into your Twitter account
2. Click on the icon the looks like an egg near the top right hand corner of the page
3. Select the Settings tab.
4. In the menu on the left hand side, select the tab labeled Mobile.
5. Then find the button labeled Delete Mobile
6. And the mobile number attached to the account is removed.

If you want to remove your Twitter Application:

1. log into your Twitter account
2. Click on this link `https://apps.twitter.com/app`
3. Select your Twitter Application
4. Then click the Delete Application button at the bottom.

9.4 References

`http://www.tweepy.org/`
`https://github.com/tweepy/tweepy`
`http://www.afterhoursprogramming.com/tutorial/Python/Introduction/`