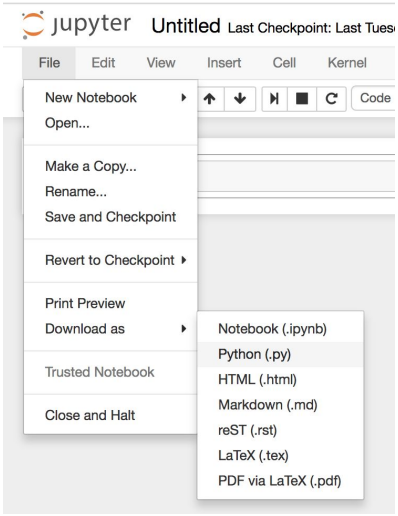
Homework 1: Collect 100 Tweets on a certain topic!
Due: Sept 27 11:59pm

Problem 0, fix AND modify a buggy Python script and scrape some job postings!
Task 1: Please debug the buggy_indeed_ingest.py listed on the CourseWork website.
Task 2: Modify the script such that

- Your search query is NOT “data” (as shown in the script), anything else is fine but you will use this data later in class!
- Your search should not require to be fulltime
- Please do not change the location for this assignment
- You can add 0 to 2 more restrictions if you want (look what Indeed offers!)
- Please have exactly 100 job postings in your data

4 deliverables for this assignment:

- A text file named **{YOUR_UNI}_hw1_0_conda.txt** with the conda command you used to run the script
- Fixed Python code (NOT a jupyter notebook!)
    - You can export your Jupyter notebook into Python code as shown in the screenshot below:



    - This Python file should be named **{YOUR_UNI}_hw1_0_code.py**
- Data:
    - The data should be a JSON file and able to be read via `json.load(open('{YOUR_UNI}_hw1_0_data.json', 'r'))`
    - Please use the format above for your file name
    - Your data should be a dictionary with 2 keys: **request_params** and **job_descriptions** (this should obvious if you just follow the code)
        - Please do not modify the data type for the values as shown in the code
- A pdf of your most recent resume called {YOUR_UNI}_resume.pdf

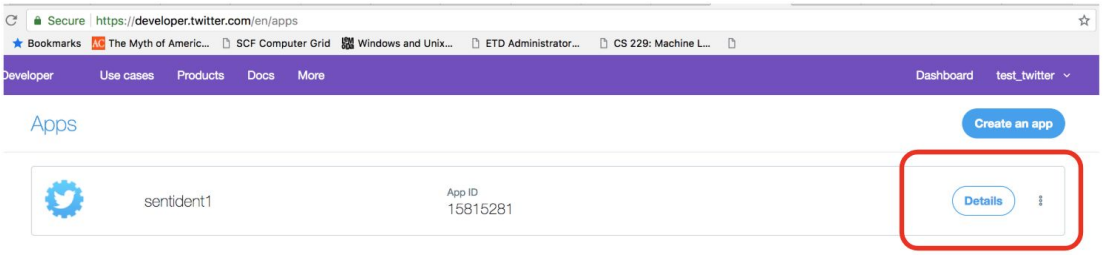
Problem 1, write code to collect Tweets!

2 deliverables for this assignment:

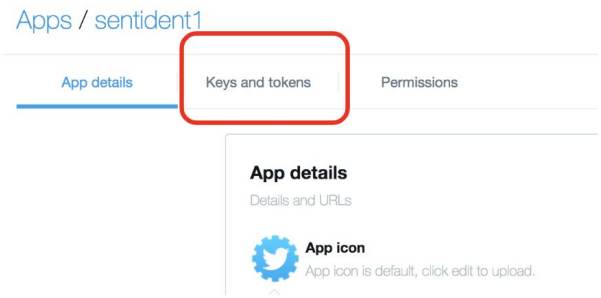
- A text file named **{YOUR_UNI}_hw1_1_conda.txt** with the conda command you used to run the script
- Data:
    - The data should be a JSON file named **{YOUR_UNI}_hw1_1_data.json**
    - The data should have exactly 100 tweets!
    - The data should be in the format of Twitter's default response.
- Do NOT turn in your code

Steps to follow:

- Apply for Twitter Developer Access
    - You'll need a twitter account (sign up!)
    - Take this seriously and do not risk getting banned!
- Getting started
    - Just fill out enough information for your app (project/thesis)
    - Find the credentials for your app
        - Find the Details for your App



        - Find your "Keys and tokens"



        - You'll need both values under **Consumer API Keys**
        - Do not share these or post these in Piazza!
- Encode your credentials
    - You'll need the steps in **Issuing application-only requests**
        - Recall from lecture how to concatenate 2 strings together!

- To "URL encode the consumer key and the consumer secret according to RFC 1738", you'll need a function **urllib.parse.quote**

```
url_encoded_key = quote(key)
```

- When you need to Base64 encode a string called MYSTRING, you'll need the function **base64.b64encode**

```
b64encode(bytes(MYSTRING, 'utf-8')).decode('utf-8')
```

- You should use their example to test that your code is running properly
- Use requests (see Problem0) and the example code below to get your bearer token. The output from the encoding above is named **bearer_b64** in my script below

```
auth_headers = {
      'Authorization': 'Basic ' + bearer_b64,
      'Content-Type': 'application/x-www-form-urlencoded;charset=UTF-8',
      }
auth_response = requests.post(url='https://api.twitter.com/oauth2/token',
                                                    headers=auth_headers,
                                                    data={'grant_type':
'client_credentials'})
```

- Create a variable that contains the same information as the **header** below (this helps Twitter know which App is calling it)

```
headers = {
   'Authorization': 'Bearer ' + auth_response.json()['access_token']
}
```

- Get some free historical tweets
  - You'll need to use the requests library
  - Study the "parameters" section in this page and see what you want/need to specify!
    - Remember we want exactly 100 tweets!
  - The call to Twitter should be very similar to the call to indeed.com except you'll need to pass an additional argument called **headers** to the function. The credentials from the previous step is what you should pass here.
- Extract only the "data" from the request object
  - Try following these instructions then encode this output to a JSON file as done in Problem 0.
  - Your data should be a dictionary with 2 keys: **search_metadata** and **statuses**
  - The value corresponding to statuses should have a length of 100.