# Lesson 4 - Web API

May 5, 2019

# 1 Lesson 4 - Web API

## 1.1 Requesting information from the web

### 1.1.1 Python 'requests' module.

- This module provides functions to send a HTTP request and get the response from the server
- Requests is a third party module. If not installed, we will need to do "pip install requests" in the mac terminal or in the command pronpt of windows.
- http://docs.python-requests.org/en/master/user/quickstart/#make-a-request

### 1.1.2 Using 'requests' module

Use the requests module to make a HTTP request to http://www.github.com/ibm - Check the status of the request - Display the response header information

```
<img src="HTTPresponse.gif"  width="200" title="HTTP response packet">
```

*Get status code for the request*

```
In [1]: import requests
        url = 'http://www.github.com/ibm'
        response = requests.get(url)

        print(response.status_code)
```

```
200
```

*Get header information*

```
In [ ]: import requests
        url = 'http://www.github.com/ibm'
        response = requests.get(url)

        print(response.status_code)

        if response.status_code == 200:
            print('Response status - OK ')
            print(response.headers)
        else:
            print('Error making the HTTP request ',response.status_code  )
```

*Get the body Information*

```
In [ ]: import requests
        url = 'http://www.github.com/ibm'
        response = requests.get(url)

        print(response.status_code)

        if response.status_code == 200:
            print('Response status - OK ')
            print(response.text)
        else:
            print('Error making the HTTP request ',response.status_code  )
```

## 1.2 Using a Web API to Collect Data

- An application programming interface is a set of functions that you call to get access to some service.
- An API is basically a list of functions and datatsructures for interfacing with websites's data.

The way these work is similar to viewing a web page. When you point your browser to a website, you do it with a URL (http://www.github.com/ibm for instance). Github sends you back data containing HTML, CSS, and Javascript. Your browser uses this data to construct the page that you see. The API works similarly, you request data with a URL (http://api.github.com/org/ibm), but instead of getting HTML and such, you get data formatted as JSON.

### 1.2.1 Access data using web APIs

*Write a program to access all the public OSS projects hosted by IBM on github.com using the web apis*

#### Step 1: Access the Web API service and check rate limits

```
In [9]: import requests

        url = "https://api.github.com/orgs/ibm"
        response = requests.get(url)

        if response.status_code == 200:
            print('Response status - OK ')
            print(response.headers['X-RateLimit-Remaining'])
        else:
            print('Error making the HTTP request ',response.status_code  )
```

```
Response status - OK
53
```

**Step 2: Authentication (if required)** Authenticate requests to increase the API request limit. Access data that requires authentication.

**Basic Authentication**

- Pass the userid and password as parameters in the response.get function
- Little risky and prone to hacking. Create dummy user ID and password

**OAUTH**

- OAuth 2 is an authorization framework that enables a user to connect to their account using a third party application
- While this is more secure thant the basic authentication (i.e. passing the userid and password while you make a http request), it is a little more difficult to code.
- It needs a personal token and a consumer key to be generated and passed to the webserver

Unfortunately different websites have different ways of generating and using the token and consumer keys. Hence we will need to write the authorization code for each website seperately. HOwever, every website provides detailed information on how you can generate and send the token and keys.

```
In [ ]: import requests

        def GithubAPI(url):
            """ Make a HTTP request for the given URL and send the response body
            back to the calling function"""
            # Use basic authentication
            response = requests.get(url, auth=("ENTER USER ID","ENTER PASSWORD"))
            if response.status_code == 200:
                print('Response status - OK ')
                print(response.headers['X-RateLimit-Remaining'])
                return response.text
            else:
                print('Error making the HTTP request ',response.status_code  )
                return None

        def main():
            url = "https://api.github.com/orgs/ibm"
            txt_response = GithubAPI(url)

            if txt_response:
                print(txt_response)

        main()
```

**Step 3: Parse the response** The *json* module gives us functions to convert the JSON response to a python readable data structure.

*Write a program to get the number of OSS projects started by IBM*

```
In [13]: import requests
         import json

         def GithubAPI(url):
             """ Make a HTTP request for the given URL and send the response body
             back to the calling function"""
             response = requests.get(url)
             if response.status_code == 200:
                 print('Response status - OK ')
                 return response.json()
             else:
                 print('Error making the HTTP request ',response.status_code  )
                 return None

         def main():
             url = "https://api.github.com/orgs/ibm"
             txt_response = GithubAPI(url)

             if txt_response:
                 print('The number of public repos are : ',txt_response['public_repos'])

         main()
```

```
Response status - OK
The number of public repos are :  851
```

**Step 3: Follow the url information from the Web API to find what you need** *Let us collect the information regarding the different projects started by IBM*

```
In [ ]: import requests
        import json

        def GithubAPI(url):
            """ Make a HTTP request for the given URL and send the response body
            back to the calling function"""
            response = requests.get(url, auth("ENTER USER ID","ENTER PASSWORD"))
            if response.status_code == 200:
                print('Response status - OK ')
                return response.json()
            else:
                print('Error making the HTTP request ',response.status_code  )
                return None

        def main():
            url = "https://api.github.com/orgs/ibm"
            response_json = GithubAPI(url)
```

```
    if response_json:
        print('The number of public repos are : ',response_json['public_repos'])
        repo_url = response_json['repos_url']
        repo_response = GithubAPI(repo_url)
        for repo in repo_response:
            print([repo['id'],repo['name']])

main()
```

**Step 4: Paginate to get data from other pages** *Traverse the pages if the data is spread across multiple pages*

In [ ]:
```
import requests
import json

def GithubAPI(url):
    """ Make a HTTP request for the given URL and send the response body
    back to the calling function"""
    response = requests.get(url, auth = ("ENTER USER ID","ENTER PASSWORD"))
    if response.status_code == 200:
        print('Response status - OK ')
        return response.json()
    else:
        print('Error making the HTTP request ',response.status_code  )
        return None

def main():
    url = "https://api.github.com/orgs/ibm"
    response_json = GithubAPI(url)

    if response_json:
        print('The number of public repos are : ',response_json['public_repos'])
        repo_url = response_json['repos_url']
        total_no =  response_json['public_repos']
        per_page = 100
        page_count = 1
        while page_count < total_no/100:
            #Display 100 repos per page and traverse the pages untill we get the last
            page_url = repo_url+"?per_page=100&page_no="+str(page_count)
            print(page_url)
            repo_response = GithubAPI(page_url)
            # Increment page number
            page_count = page_count+1
            for repo in repo_response:
                print([repo['id'],repo['name']])

main()
```

## 1.3 Write a CSV

Lets try to write the repos into a CSV file.

*Write a code to append data row wise to a csv file*

```
In [ ]: import csv
        WRITE_CSV = "C:/Users/kmpoo/Dropbox/HEC/Teaching/Python for PhD Mar 2018/python4phd/Se
        with open(WRITE_CSV, 'at',encoding = 'utf-8', newline='') as csv_obj:
            write = csv.writer(csv_obj) # Note it is csv.writer not reader

            write.writerow(['REPO ID','REPO NAME'])
```

*What do you think will happen if we use 'wt' as mode instead of 'at' ?*

*Write a program so that you save the IBM repositories into the CSV file. So that each row is a new repository and column 1 is the ID and column 2 is the name*

```
In [ ]: #Enter code here

        import requests
        import json
        import csv

        WRITE_CSV = "C:/Users/kmpoo/Dropbox/HEC/Teaching/Python for PhD Mar 2018/python4phd/Se

        def appendcsv(data_list):
            with open(WRITE_CSV, 'at',encoding = 'utf-8', newline='') as csv_obj:
                write = csv.writer(csv_obj) # Note it is csv.writer not reader
                write.writerow(data_list)

        def GithubAPI(url):
            """ Make a HTTP request for the given URL and send the response body
            back to the calling function"""
            response = requests.get(url, auth = ("ENTER USER ID","ENTER PASSWORD"))
            if response.status_code == 200:
                print('Response status - OK ')
                return response.json()
            else:
                print('Error making the HTTP request ',response.status_code  )
                return None

        def main():
            url = "https://api.github.com/orgs/ibm"
            response_json = GithubAPI(url)

            if response_json:
                print('The number of public repos are : ',response_json['public_repos'])
                repo_url = response_json['repos_url']
                total_no =  response_json['public_repos']
                per_page = 100
                page_count = 1
                while page_count < total_no/100:
                    #Display 100 repos per page and traverse the pages untill we get the last p
                    page_url = repo_url+"?per_page=100&page_no="+str(page_count)
                    print(page_url)
                    repo_response = GithubAPI(page_url)
                    # Increment page number
                    page_count = page_count+1
                    for repo in repo_response:
                        print([repo['id'],repo['name']])
                        appendcsv([repo['id'],repo['name']])


        main()
```