HTTP Request/Response Cycle - Lab

Introduction

In this lab, we shall attempt to bring together everything we have thus far learned about APIs, Client/Server Communication principles and Python's `requests` library. We shall make use of the `requests` module's methods and properties seen in the previous lesson, to extract information for a web service called **"Open Notify"**, in order to access NASA's space data.

Objectives

You will be able to:

- Understand and explain the HTTP Request/Response cycle
- Make http requests in Python using the 'requests' library

Open Notify

Open Notify is an an open source project to provide a simple programming interface for some of NASA's awesome data. This takes live raw data from NASA's systems and turn them into APIs related to space and spacecraft. We can access following information from open notify.

- Current Location of the International Space Station
  - (The ISS is currently over 22.014° N, -101.057° E)
- Overhead Pass Predictions for the International Space Station
- Number of People in Space
  - (There are currently 3 humans in space)

API endpoints

OpenNotify has several API endpoints.

> An endpoint is a server route that is used to retrieve different data from the API.

For example, the `/comments` endpoint on the Reddit API might retrieve information about comments, whereas the `/users` endpoint might retrieve data about users. To access them, you would add the endpoint to the base url of the API.

For the OpenNotify API, we have following end points

1. Current Location of the International Space Station `/iss-now.json`
2. Overhead Pass Predictions for the International Space Station `/iss-pass.json`
3. Number of People in Space `/astros.json`

The json extension simple tells us that the data is being returned in a JSON format.

In this lab, we'll be querying a this API to retrieve live data about the International Space Station (ISS). Details on OpenNofity , endpoints, syntax and services it offers can be viewed Here

Current location of International Space Station

The first endpoint we'll look at on OpenNotify is the `iss-now.json` endpoint (current location of international space station). This endpoint gets the current latitude and longitude of the International Space Station. Perform following tasks

- Make a get request to get the latest position of the international space station from the opennotify api's iss_now endpoint at http://api.open-notify.org/iss-now.json
- Check the status code of the response

- Check the status code of the response
- Interpret the returned code

In [2]:
```
import requests
# Make a get request to get the latest position of the international space station from the opennot
response = requests.get("http://api.open-notify.org/iss-now.json")

# Print the status code of the response.
print(response.status_code)
```

```
200
```

In [3]:
```
# Your comments
# code 200 -- everything went okay, and the result has been returned .
```

- Print the contents of the response and identify its current location

In [4]:
```
print(response.text)
```

```
{"timestamp": 1539962082, "message": "success", "iss_position": {"latitude": "-36.5945", "longitud
e": "-174.1912"}}
```

In [5]:
```
# Interpret your results using the API - where is the space station right now ?
```

Check the next pass of International space station for a given location

Let's repeat the above for the second endpoint `iss_pass.json` . This end point is used to query the next pass of the space station on a given location. Let's just run as above and record your observations

In [6]:
```
# Make a get request to get the latest position of the international space station from the opennot
response = requests.get("http://api.open-notify.org/iss-pass.json")

# Print the status code of the response.
print(response.status_code)
```

```
400
```

In [7]:
```
# Your comments
# code 400 -- the server thinks you made a bad request.
# This can happen when you don't send along the right data, among other things.
```

So clearly there is something wrong as we had a 400 response. This is how you should always test your responses for validity.

if we look at the documentation for the OpenNotify API, we see that the `iss-pass.json` endpoint requires two parameters.

> The `/iss-Pass.json` endpoint returns when the ISS will next pass over a given location on earth. In order to compute this, we need to pass the coordinates of the location to the API. We do this by passing two parameters -- latitude and longitude.

We can do this by adding an optional keyword argument, params, to our request. In this case, there are two parameters we need to pass:

- lat -- The latitude of the location we want.
- lon -- The longitude of the location we want.

Perform the following tasks :

- Set parameters to reflect the lat and long of New York (40.71, -74)
- Send a get request to OpenNotify passing in the lat long parameters as k:v pairs in a dictionary i.e. {"lat": 40.71, "lon": -74}
- Check the status code and interpret
- Print the header information and the returned content

In [8]:
```
# Set up the parameters we want to pass to the API.
# This is the latitude and longitude of New York City.
parameters = {"lat": 40.71, "lon": -74}

# Make a get request with the parameters.
response = requests.get("http://api.open-notify.org/iss-pass.json", params=parameters)
print(response.status_code)

# Print the content of the response (the data the server returned)
print(dict(response.headers))
print(response.text)
```

```
200
{'Server': 'nginx/1.10.3', 'Date': 'Fri, 19 Oct 2018 15:14:49 GMT', 'Content-Type': 'application/j
son', 'Content-Length': '519', 'Connection': 'keep-alive', 'Via': '1.1 vegur'}
{
```

```
{
  "message": "success",
  "request": {
    "altitude": 100,
    "datetime": 1539962089,
    "latitude": 40.71,
    "longitude": -74.0,
    "passes": 5
  },
  "response": [
    {
      "duration": 577,
      "risetime": 1539963737
    },
    {
      "duration": 554,
      "risetime": 1539969601
    },
    {
      "duration": 621,
      "risetime": 1539975406
    },
    {
      "duration": 628,
      "risetime": 1539981195
    },
    {
      "duration": 285,
      "risetime": 1539987108
    }
  ]
}
```


In [9]:
```
# Check the API and interpret your results - when will ISS pass over NEW York next ?
```

Finding the number of people in space

OpenNotify has one more API endpoint, `/astros.json`. It tells you how many people are currently in space. The format of the responses can be studied HERE.

Read the above documentation and perform following tasks:

- Get the response from astros.json endpoint
- Count how many people are currently in space
- List the names of people currently in space.

In [10]:
```
# Get the response from the API endpoint.
response = requests.get("http://api.open-notify.org/astros.json")
data = response.json()

# 9 people are currently in space.
print(data["number"])
print(dict(data))
```

```
3
{'people': [{'name': 'Sergey Prokopyev', 'craft': 'ISS'}, {'name': 'Alexander Gerst', 'craft': 'IS
S'}, {'name': 'Serena Aunon-Chancellor', 'craft': 'ISS'}], 'message': 'success', 'number': 3}
```

In [11]:
```
# Interpret the Results - How many people are in space and what are their names
```

Summary

In this lesson we saw how we can use request and response methods to query an Open API. We also saw how to look at the contents returned with the API calls and how to parse them. Next we shall look at connecting to APIs which are not OPEN , i.e. we would need to paass in some authentication information and filter the results.