HTTP Request/Response Cycle - Lab

Introduction

In this lab, we'll make use of the `requests` module commands and properties seen in the previous lesson, to extract information for a web service called **"Open Notify"** to access NASA's space data.

Objectives

You will be able to:

- Explain the HTTP request/response cycle
- List the status codes of responses and their meanings
- Obtain and interpret status codes from responses
- Make HTTP GET and POST 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 the following information from open notify.

- Current Location of the International Space Station
- Number of People in Space
- Overhead Pass Predictions for the International Space Station

API endpoints

Open Notify 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 the following endpoints:

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

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

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

Current location of International Space Station

The first endpoint we'll look at on Open Notify 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 the 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
- Interpret the returned code

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

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

```
200
```

```
# Your comments
# code 200 -- everything went okay, and the result has been returned.
```

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

```
print(response.text)
```

```
{"timestamp": 1648752857, "message": "success", "iss_position": {"longitude":
"119.4328", "latitude": "-44.9325"}}
```

```
# Interpret your results using the API - where is the space station right now ?
```

Number of people in space

Let's repeat the above for the second 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 the 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.

```
# 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))
```

```
10
{'message': 'success', 'people': [{'name': 'Zhai Zhigang', 'craft': 'Shenzhou
13'}, {'name': 'Wang Yaping', 'craft': 'Shenzhou 13'}, {'name': 'Ye Guangfu',
'craft': 'Shenzhou 13'}, {'name': 'Raja Chari', 'craft': 'ISS'}, {'name': 'Tom
Marshburn', 'craft': 'ISS'}, {'name': 'Kayla Barron', 'craft': 'ISS'}, {'name':
'Matthias Maurer', 'craft': 'ISS'}, {'name': 'Oleg Artemyev', 'craft': 'ISS'},
{'name': 'Denis Matveev', 'craft': 'ISS'}, {'name': 'Sergey Korsakov.', 'craft':
'ISS'}], 'number': 10}
```

```
# Interpret the Results - How many people are in space and what are their names
```

Level Up (Optional) - Investigating other API endpoints

Next pass of International space station for a given location

Let's repeat the exercise for another endpoint `iss-pass.json` . This endpoint is used to query the next pass of the space station on a given location.

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

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

```
400
```

```
# 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.
```

The status code for the end point returned is 400, which indicates a client error. To see more detailed information, we can use `response.text` instead of `response.status_code` .

```
# Make the same request, but use response.text

response = requests.get("http://api.open-notify.org/iss-pass.json")
print(response.text)
```

```
{
  "message": "failure",
  "reason": "Latitude must be specified"
}
```

This endpoint is not visible from the documentation, but we can navigate to the source code of Open Notify and see if we can gain more information on how to use this endpoint.

Click on the `Source Code` Tab, and navigate to the Open Notify GitHub repository.

https://github.com/open-notify/Open-Notify-API

If we look at the API spec this endpoint, we see that the ISS Pass endpoint requires two parameters.

https://github.com/open-notify/Open-Notify-API/blob/master/app.py#L103

> The API returns a list of upcoming ISS passes for a particular location formatted as JSON. As input it expects a latitude/longitude pair, altitude and how many results to return. All fields are required.

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 Open Notify passing in the lat long parameters as k:v pairs in a dictionary
- Check the status code and interpret
- Print the header information and the returned content

```
# 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=parameter
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': 'Thu, 31 Mar 2022 18:54:40 GMT', 'Content-
Type': 'application/json', 'Content-Length': '519', 'Connection': 'keep-alive',
'Via': '1.1 vegur'}
{
  "message": "success",
  "request": {
    "altitude": 100,
    "datetime": 1648752200,
    "latitude": 40.71,
    "longitude": -74.0,
    "passes": 5
  },
  "response": [
    {
      "duration": 647,
      "risetime": 1648755267
    },
    {
      "duration": 580,
      "risetime": 1648761143
    },
    {
      "duration": 570,
      "risetime": 1648767017
    },
    {
      "duration": 639,
      "risetime": 1648772833
    },
    {
      "duration": 627,
      "risetime": 1648778642
    }
  ]
}
```

```
# Check the API and interpret your results - when will ISS pass over NEW York next ?
```

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'll look at connecting to APIs which are not OPEN, i.e. we would need to pass in some authentication information and filter the results.

Releases

No releases published

Packages

No packages published

Contributors 8

Languages

● **Jupyter Notebook** 100.0%