Lab - Using Python to Integrate MapQuest and ISS Location API with Webex Teams APIs

Objectives

In this lab, you will complete the following objectives:

- Use Postman with the ISS location API.
- Use Python to ask for the access token.
- Add the URL for the Webex Teams List Rooms API endpoint.
- Add the MapQuest API key.
- Set the longitude key as returned by the MapQuest API.
- Convert the Unix epoch timestamp.
- Add the URL for the Webex Teams messages endpoint.
- Execute the program.

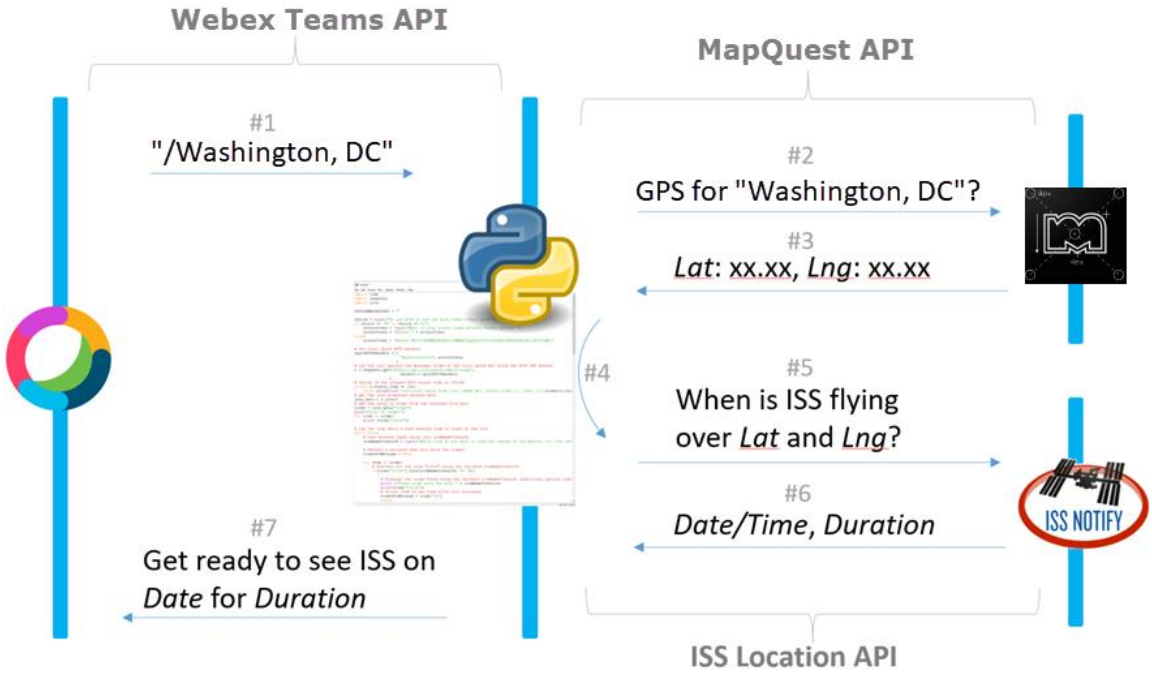
Background / Scenario

Python can be used to integrate API calls from different services. In this lab, you will edit and modify Python code that uses MapQuest and ISS location APIs. The code continuously checks for messages in a user specified Webex Teams room starting with "/" followed by a city name (e.g. /Washington, DC). When such message is found, the city name is extracted and in a first step, it is used with the MapQuest API service to get the GPS coordinates in the form of latitude and longitude of the specified city. In the second step, the latitude and longitude are used with the ISS location API to find the date and time of the next the ISS will fly over the specified coordinates. In the last step, a summary message is sent back to the Webex Teams room informing about the next flyover of the ISS over the queried city.

When completed, the **09_iss-flyover-to-webex_teams-bot.py** program will:

- Ask the users for their access token or to use the hard-coded access token.
- Display a list of the user's Webex Teams rooms.
- Ask the user which Webex Teams room to monitor for the queries starting with "/".
- Extract the city name of a message starting with "/" (e.g. /Washington, DC -> Washington, DC)
- Request the latitude and longitude of a specified city using the MapQuest API.
- Request the next flyover date and time using the ISS location API.

- Send the next ISS flyover information back to the specified Webex Teams room.



Required Resources

- Webex Teams account
- Postman application
- Webex Teams desktop application
- Python 3 with IDLE
- Python code files

Note: To protect application environments like Webex Teams from bots or malicious access attempts, most APIs rate limit availability. If you make a large number of the same API calls, your API call may be blocked for a specific amount of time. The timeout is usually less than 5 minutes.

Step 1: Use Postman with the ISS location API.

In this step, you will use Postman to make a request to the ISS location API and examine the returned JSON data.

a. Use the MapQuest API (as introduced in Chapter 1) to discover the GPS coordinates for “Washington, DC”. Replace “your_api_key” with your key in the following URL:

https://www.mapquestapi.com/geocoding/v1/address?key=your_api_key&location=Washington,DC

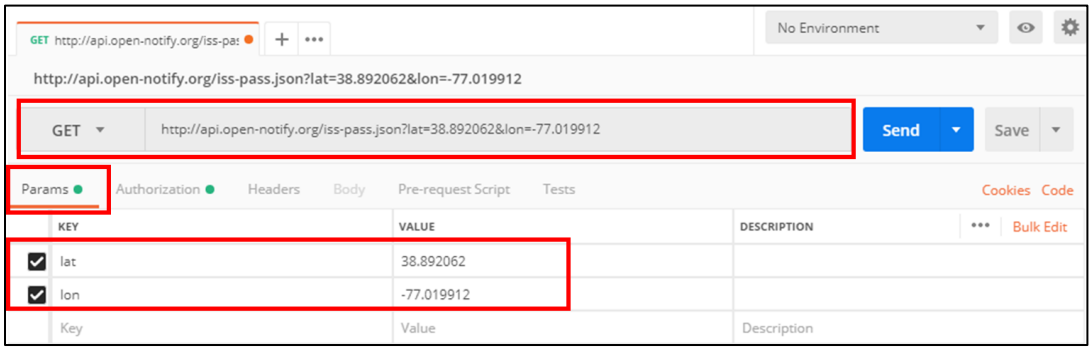
b. Start **Postman** and create a new tab under **Builder**.

c. You will use the default HTTP GET method. Add the URL **http://api.open-notify.org/iss-pass.json**.

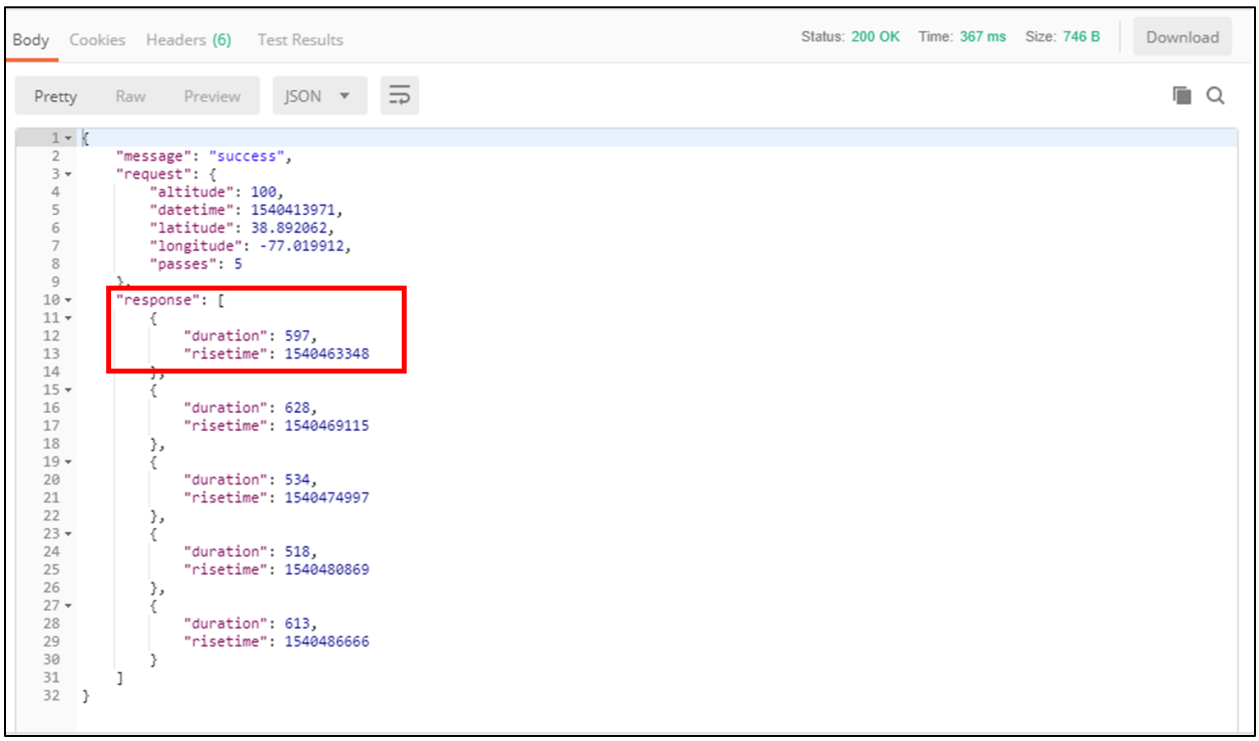
d. The ISS location API requires the **lat** and **lon** GET parameters for the latitude and longitude coordinates to discover when the ISS flies over that place.

You can find the API documentation at http://open-notify-api.readthedocs.io/en/latest/iss_pass.html.

- Select **Params**, if necessary.
- In the **Key** field, enter **lat** for latitude.
- In the **Value** field, enter **38.892062** for Washington, DC.
- In the **Key** field, enter **lon** for longitude.
- In the **Value** field, enter **-77.019912** for Washington, DC.
- Click **Send**.



e. Examine the various key/value JSON pairs returned by the API call:



f. The “response” key contains a list of upcoming ISS flyovers. The first item in the list is the first flyover. The “risetime” is when the ISS starts the flyover and is encoded as Unix epoch timestamp in seconds.

You can convert the epoch timestamp into a human readable form using https://www.epochconverter.com or using the following Python code:

```
import time
print(str(time.ctime(1540231033)))
```

Step 2: Use Python to ask for the access token.

Create the code to request from the user which access token to be used. Prompt the users to enter their access token if the hard-coded token will not be used. Refer to the previous labs or the solution file, if necessary.

Step 3: Add the URL for the Webex Teams List Rooms endpoint.

a. Go to **Student Step #3** in the code and locate the statement:

```
r = requests.get( "<!!!REPLACEME with URL for Webex Teams rooms API!!!>",
                      headers={"Authorization": accessToken}
                  )
```

b. Replace **<!!!REPLACEME with URL for Webex Teams API!!!>** with **https://api.ciscospark.com/v1/rooms**.

Step 4: Add the MapQuest API key.

a. Go to **Student Step #4**.

b. Replace **<!!!REPLACEME with your MapQuest API Key!!!>** with the MapQuest API Key from Chapter 1.

Step 5: Set the longitude key as returned by the MapQuest API.

a. Go to **Student Step #5**.

b. Replace **<!!!REPLACEME with the longitude key!!!>** with the longitude key as retuned by the MapQuest API (hint: see Step 1a).

Step 6: Convert the Unix epoch timestamp.

a. Go to **Student Step #6**.

b. Replace the **<!!!REPLACEME with a function that converts Epoch timestamp to human readable time!!!>** with a Python function call that converts a Unix Epoch Timestamp to a human readable time (hint: see Step 1f).

Step 7: Add the URL for the Webex Teams messages endpoint.

a. Go to **Student Step #7**.

b. Replace **<!!!REPLACEME with URL for Webex Teams messages API!!!>** with **https://api.ciscospark.com/v1/messages**.

Step 8: Execute the program.

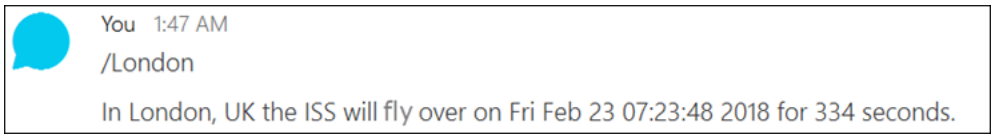
a. Save your program and then choose **Run** > **Run Module** to execute the program in IDLE.

b. Debug as necessary.

Common Python problems include:

- **Punctuation and symbols:** " ",' ', :, =,==, (), { }, [ ] are meaningful.
- **Paired symbols:** must match (open and close).
- **Indentation:** standard indentation is four (or multiples of four) spaces, control structures additional indentation.

c. After typing /location (e.g. /London) in the monitored Webex Teams room, you should see a similar reply:



d. The solution file is: **09_iss-flyover-to-webex_teams-bot_sol.py**.