# CET-1112 – Practical Assessment

Note: You will need to search for information online and navigate the web to find solutions. You will have 2.5 Hours to complete this assessment.

# Objectives

Part 1: Collect Required API Documentation

Part 2: Code a Webex Teams Bot

# Background / Scenario

Python can be used to integrate API calls from different services. In this Skills Assessment, you will edit and modify Python code for a Webex Teams bot that integrates Webex, MapQuest, and Sunrise/Sunset APIs. The code continuously checks for messages in a user specified Webex Teams room that begins with a forward slash (/) followed by a city name (e.g. /Washington, DC). When such a message is found, the city name is extracted and used with the MapQuest API service to get the GPS coordinates for the specified city. Next, these coordinates are used with the Sunrise/Sunset API to find the relevant sunrise and sunset times. Finally, a message is sent back to the Webex Teams room informing the user about the days sunrise and sunset for the queried city.

When completed, the Webex Teams bot will:

* Ask the user 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 queries starting with “/”.
* Extract the city name of a message starting with “/” (e.g. /Toronto, ON -> Toronto, ON).
* Request the latitude and longitude of a specified city using the MapQuest API.
* Request the sunrise and sunset times using the Sunrise/Sunset API.
* Send the sunrise and sunset information back to the specified Webex Teams room.

# Required Resources

* One personal computer matching the minimum system requirements of this course.
* MapQuest and Webex Teams developer access and API keys
* Virtual Box Software
* DEVASC Virtual Machine
* The **devbot.py** student script – available on Moodle

**Notes**:

* 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.
* If there have been changes to the API, it is up to you to resolve what commands are required to solve the problem at hand, and adjust your script accordingly.
* You must decide what evidence is sufficient for each component – this may be code, output, screenshots or other written evidence, and will be marked in red. You decide and may need to justify why what you have provided is valid.

# Instructions

## Collect Required API Documentation

In this Part, you will collect information from the Webex, MapQuest, and Sunrise/Sunset API documentation. This information will be required in Part 2 when you are coding the Webex Teams bot.

### Launch the DEVASC VM.

Although it is possible to complete this Skills Assessment in other environments, these instructions are for the DEVASC VM only. Instructions for other environments will not be provided.

### Investigate the documentation for Webex Teams rooms and message APIs.

* + - 1. Login to your developer account for Webex ( <https://developer.webex.com/> )
      2. Investigate the Webex Developer Documentation ( <https://developer.webex.com/docs/> )
         1. Locate and copy your personal access token. What is the lifetime for your token?

Personal Token: M2RiNWNhOTQtN2YxMy00MmJlLWJmNDktNDNhM2I4YzU5MDk2OWE4ZjYxMGUtNDFl\_PC75\_47fe537e-27d1-4e32-b2dc-2c26e4aa4fa0

Lifetime: 12 Hour after login

* + - * 1. Find the URL that will *list all the rooms to which you belong*.

Record the HTTP method and URL: GET[https://webexapis.com/v1/rooms](https://developer.webex.com/docs/api/v1/rooms/list-rooms)

* + - * 1. Find the URL that *list all the messages for a specified room*.
        2. GET[https://webexapis.com/v1/messages](https://developer.webex.com/docs/api/v1/messages/list-messages)Find the URL that *will create a message for a specified room*.

Record the HTTP method and URL: POST[https://webexapis.com/v1/messages](https://developer.webex.com/docs/api/v1/messages/create-a-message)

### Investigate the locations key for the MapQuest address API.

* + - 1. Create a developer account for MapQuest ( <https://developer.mapquest.com/> )
      2. Investigate the MapQuest Developer Documentation ( <https://developer.mapquest.com/documentation> )
         1. Which API endpoint should be used to obtain a locations latitutude and longitude?

MapQuest Endpoint:

* + - * 1. Locate and copy your Consumer Key. When does your key expire?

Consumer Key: **wTiF9uMdbPR3EF4mfUvtF2tU5roj9Dat**

Key Expiry: no expire

* + - 1. Open Chromium and paste in the following URL, replacing **your\_api\_key** with your MapQuest consumer key:

*https://www.mapquestapi.com/geocoding/v1/address?key=****your\_api\_key****&location=Sudbury,ON*

* + - 1. Notice that the MapQuest API includes keys for latitude and longitude of the location you entered. Record the **lat** and **lng** values returned by MapQuest for Sudbury, ON below.

Latitude (lat): 46.5255

Longitude (lng): -80.93551

### Investigate the documentation for the World Time API.

* + - 1. Investigate the Sunrise/Sunset API Documentation ( <https://sunrise-sunset.org/api> )
      2. On the Sunrise/Sunset API documentation website, review the information on request parameters.
      3. What are the two required parameters (also called **query strings**) for the Sunrise/Sunset API?

Required Parameters: lat for latitude and lng for longitude

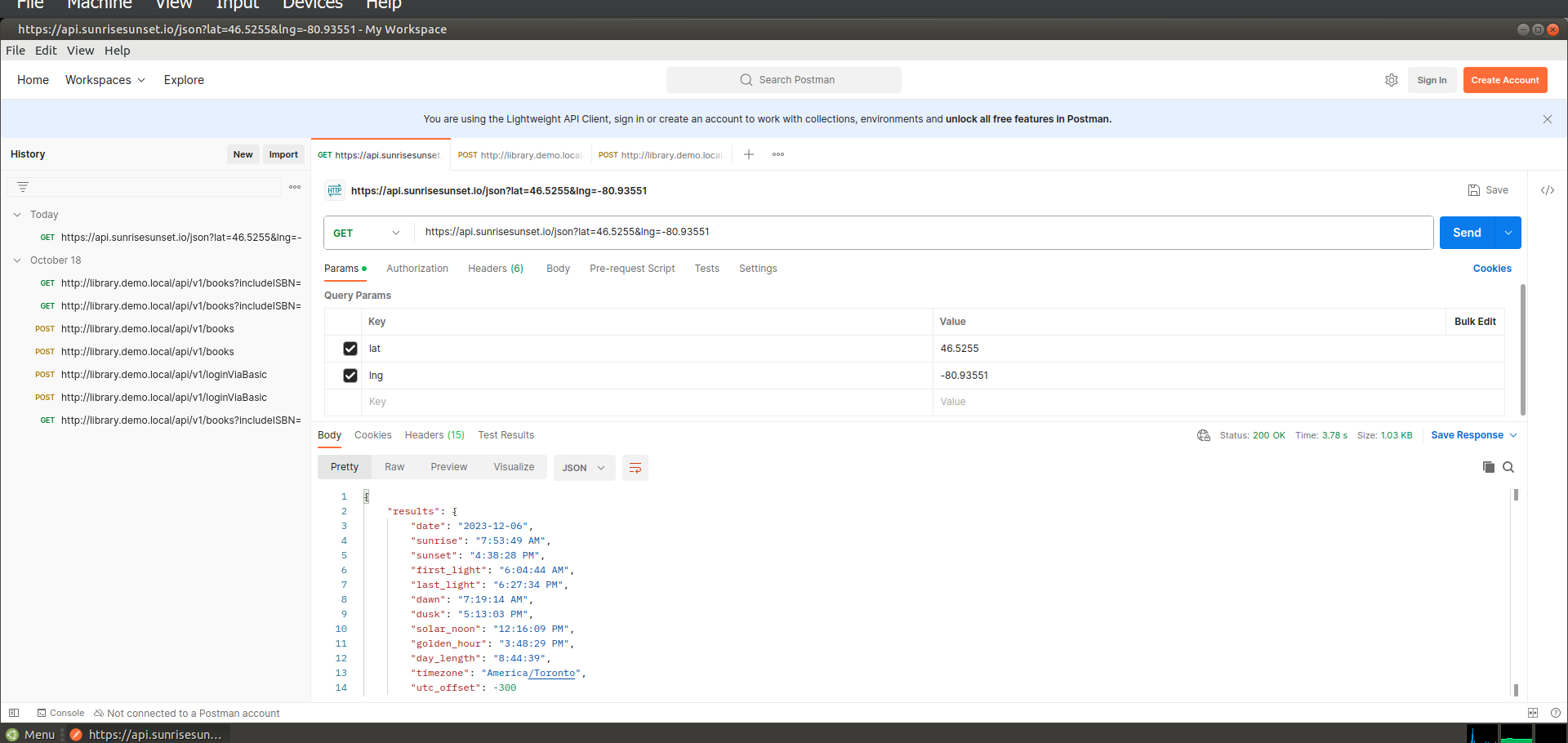
* + - 1. What are the optional parameters for the Sunrise/Sunset API?

Optional Parameters:date, timezone, date\_start, date\_end

* + - 1. What is the URL Sunrise/Sunset API?

REST API URL: https://api.sunrisesunset.io/json?lat=38.907192&lng=-77.036873

* + - 1. Open **Postman** and create a new **Untitled Request**.
      2. Paste in the Sunrise/Sunset URL.
      3. Replace the latitude and longitude values with the values for Sudbury, ON
      4. Click **Send**. You should get output similar to the following, although your values will be different.



A computer code with numbers and digits

Description automatically generated

## Code a Webex Teams Bot

**Note**: You will need the **devbot.py** script open and ready to edit for this Part. Obtain this script from your instructor.

In this Part, you will use your understanding of Python and REST APIs along with the API documentation you gathered in Part 1 to finish the code for the Webex Teams bot.

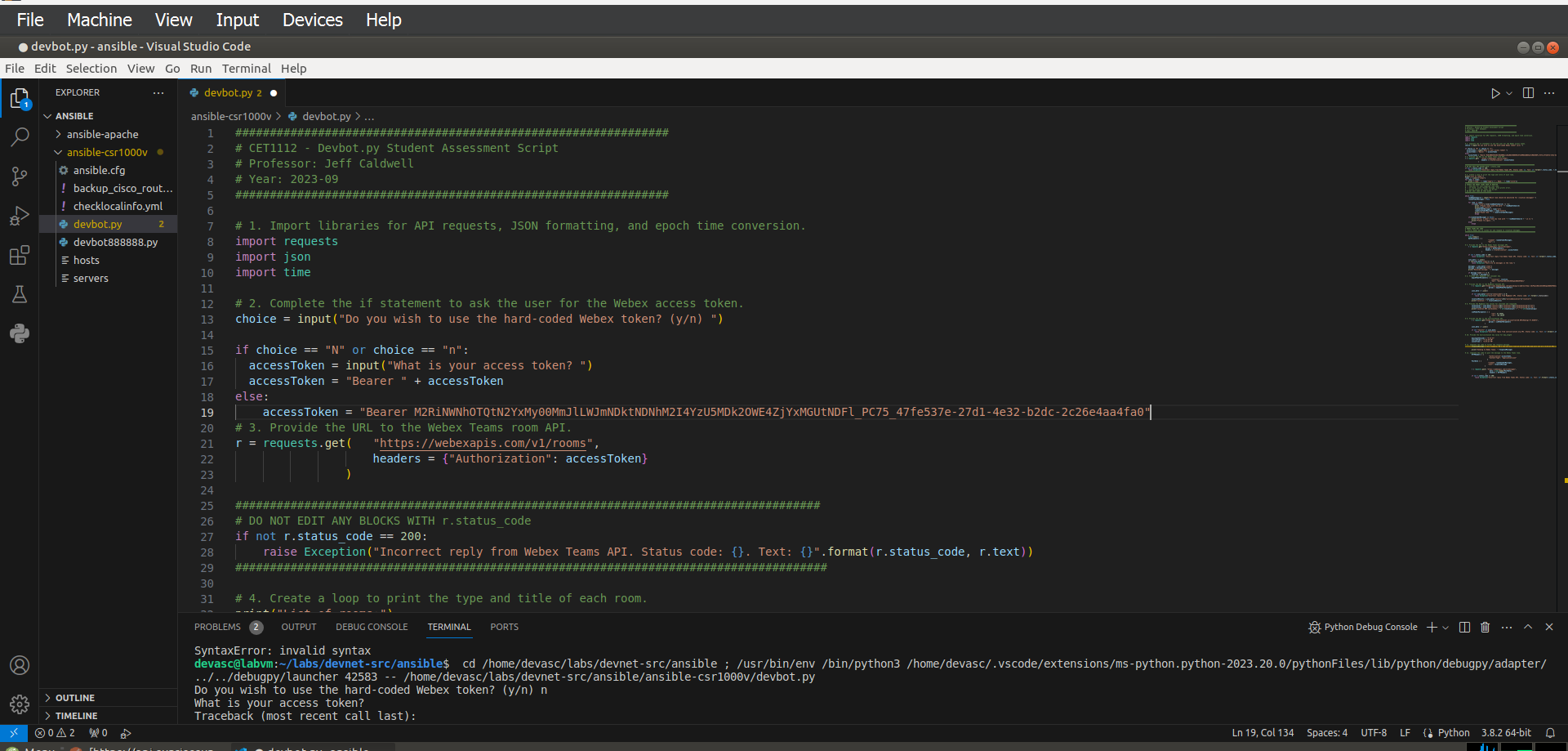
Each section of the script that requires modification is clearly marked with ***<!!!REPLACEME !!>***

**Hint**: Save the original file as a different name in case you need to start over. Comment out the code you are not yet testing. As you work through the following steps, save and run your script often.

**Pro Tip**: Use temporary print statements to verify that your variables contain the data you expect.

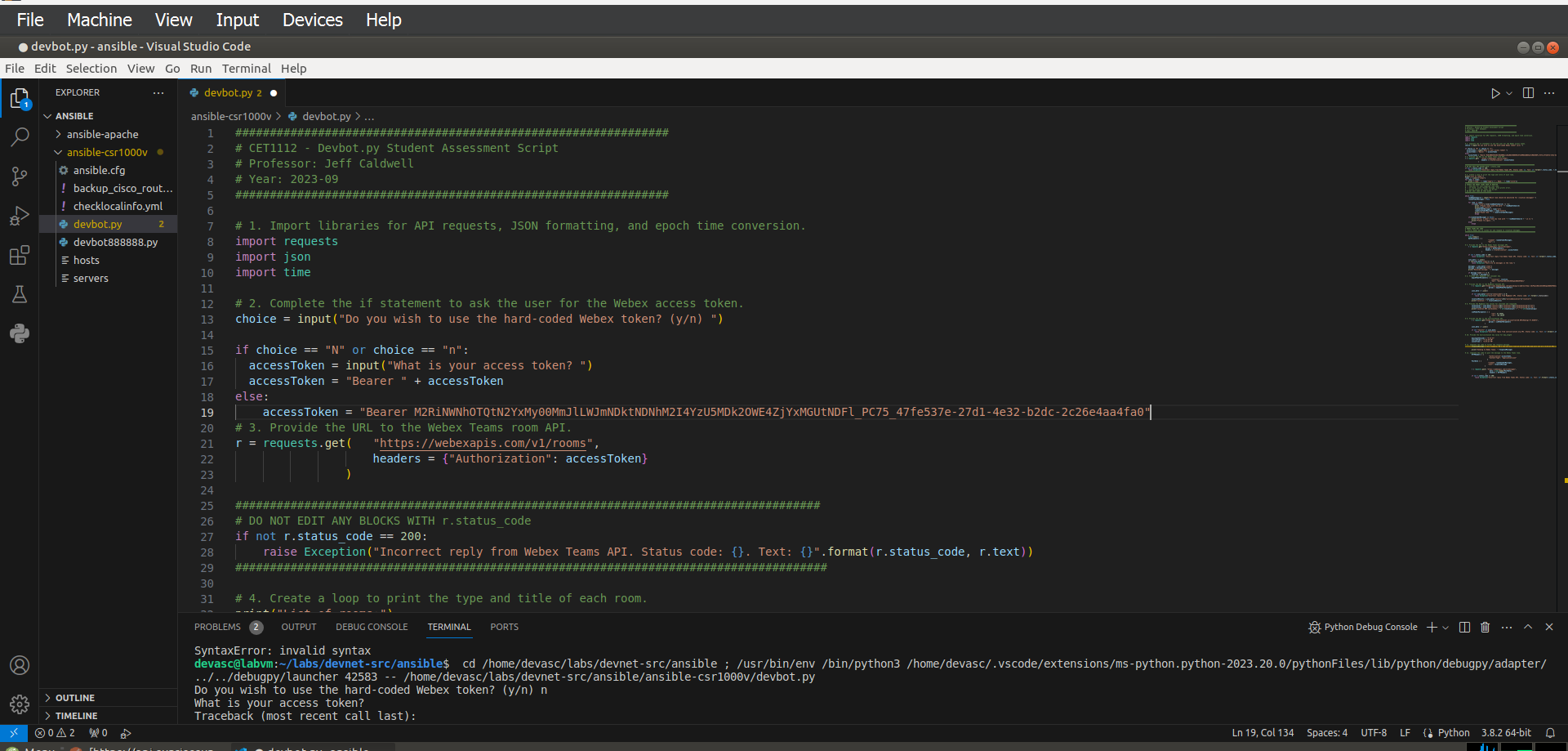
### Import libraries for API requests and JSON formatting.

Use the Python documentation and by searching the internet, determine what libraries need to be imported into your script for these tasks.



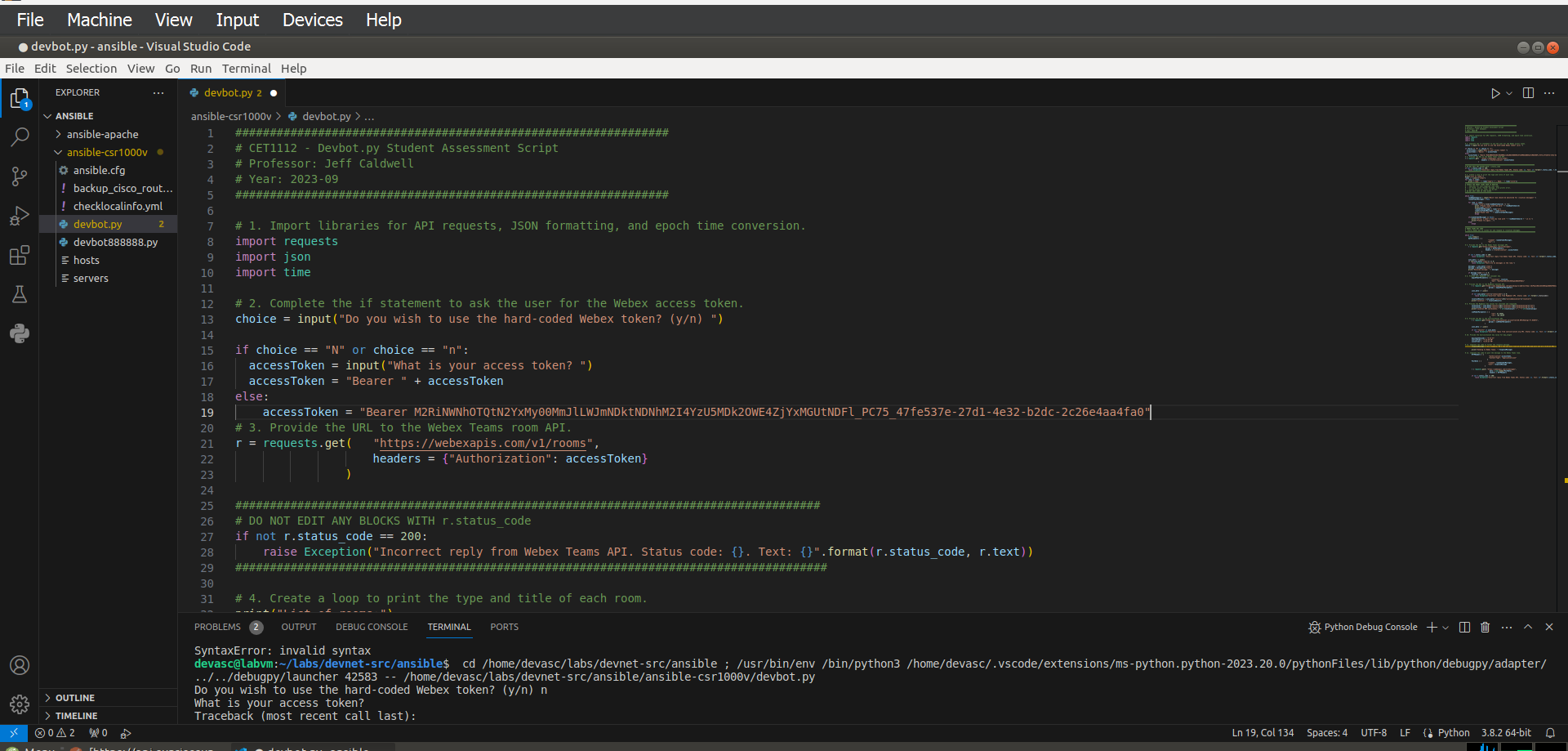
### Complete the if statement to ask the user for the Webex access token.

For this step, you are provided with the first user prompt and the **else** portion of an **if/else** statement. You need to code the **if** portion of the statement that will execute if the user says “N” or “n” to using the hard-coded access token. The **if** statement checks the value of **choice**, then if “N” or “n” are entered, asks the user for the value of the token. The user-entered value is then stored in the **accessToken** variable. The **accessToken** variable should be constructed just like the version of the **else** statement.



### Provide the URL to the Webex Teams room API.

Use your documentation from Part 1 to specify the correct Webex Teams room API that will return a list of the rooms that you are a member of and store them in the **r** variable.



### Finish the loop to print the type and title of each room.

The list of rooms stored in the **r** variable is converted to JSON and stored in the **rooms** variable. Add a print statement that will display each room type and title.

A screenshot of a computer

Description automatically generated

### Provide the URL to the Webex Teams messages API.

Use your documentation from Part 1 to specify the correct Webex Teams messages API. Every second, the bot will make a call to this API and print the latest message posted to the room.

A screenshot of a computer

Description automatically generated

### Provide your MapQuest API consumer key.

You documented this key in Part 1.

wTiF9uMdbPR3EF4mfUvtF2tU5roj9Dat

### Provide the URL to the MapQuest Geocoding API.

A screen shot of a computer program

Description automatically generated

### Provide the MapQuest key values for latitude and longitude.

Use your documentation in Part 1 to specify the correct format to store the values for the latitude and longitude keys.

A screen shot of a computer program

Description automatically generated

### Provide the URL to the Sunrise/Sunset API.

Use your documentation from Part 1 to specify the sunrise/sunset API request.

A screen shot of a computer program

Description automatically generated

### Collect and store the Sunrise/Sunset key values for sunrise, sunset and day length.

Use your documentation from Part 1 to correctly collect and store the key values from the API.

A screen shot of a computer program

Description automatically generated

### Complete the code to format the response message.

Use the variables that have been specified to format the response message that will be sent to the Webex Teams room. For example, a posted message in the room would look like the following where the location, risetime, and duration are shown in bold.

In **Sudbury, ON** the sun will rise at **XXX** and will set at **XXX**. The day will last **XXX** seconds.

A screenshot of a computer program

Description automatically generated

### Complete the code to post the message to the Webex Teams room.

The final step in the Webex Teams bot program is to format the API POST message that will send the **responseMessage** to the Webex Teams room. Provide each of the required variables and the URL for the Webex Teams message API to complete the code.

### Run your program, test it, and troubleshoot as necessary.

* + - 1. In your Webex Teams client, create a room with a name of your choice, such as **My DEVASC SA Room**.
      2. Post a message, such as **Hello Room!**, to populate the room with at least one message.
      3. Run your program and choose **My DEVASC SA Room** as the room the Webex Teams bot will monitor.
      4. In the My DEVASC SA Room, post a location in the format **/location**. Messages that begin with a forward slash initiate the Webex Team bot to do its work. For example, the following should occur when Toronto, ON is entered.

In a terminal window:

devasc@labvm:~/$ **python3 devbot.py**

Do you wish to use the hard-coded Webex token? (y/n) **y**

List of rooms:

Type: 'group' Name: My DEVASC SA Room

<rest of rooms listed>

Which room should be monitored for /location messages? **My DEVASC SA Room**

Found rooms with the word My DEVASC SA Room

My DEVASC SA Room

Found room : My DEVASC SA Room

Received message: Hello room!

Received message: Hello room!

Received message: Hello room!

Received message: Hello room!

<continues to print every 1 second>

In Webex Teams My DEVASC SA Room, trigger the bot by adding a location.

A screen shot of a computer

Description automatically generated

**/Toronto, ON**

In the terminal window, the following prints:

Received message: /Toronto, ON

Location: Toronto, Ontario

Location GPS coordinates: 43.6532, 79.3832

Sending to Webex Teams: In Toronto, Ontario the sun will rise at XXX and will set at XXX. The day will last XXX seconds.

Received message: In Toronto, Ontario the sun will rise at XXX and will set at XXX. The day will last XXX seconds.

<…remaining lines omitted>

In Webex Teams, the following displays in your selected room.

*In* ***Toronto, ON*** *the sun will rise at* ***XXX*** *and will set at* ***XXX****. The day will last* ***XXX*** *seconds.*

* + - 1. In your terminal window, enter **Ctrl+C** to exit the program.
      2. A screenshot of a computer program

         Description automatically generated

### Upload your Python script file to GitHub

* + - 1. Initialize a git repository in the directory you have your devbot.py file stored.
      2. Login to your GitHub account (created earlier in the course) and investigate documentation as needed ( <https://docs.github.com/en/get-started> )
      3. Create a matching remote repository on GitHub. This must be created as a **public** **repository** to allow your instructor access to the file.

Repository Name: final exam

* + - 1. Add the remote repository URL as the remote/origin for your local git project.

Terminal command to add remote repository:

* + - 1. Commit your code and “push” the code to your GitHub profile.

GitHub Repository URL: https://github.com/Alish8866791500/final-exam.git

### Submit your Practical Assessment Document

1. Submit your Practical Assessment document including your answers, screenshots and any other “proof” and the link to your scripts GitHub repository URL to Moodle.