

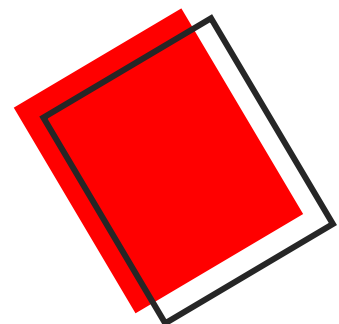
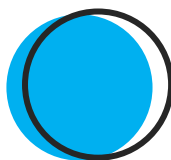
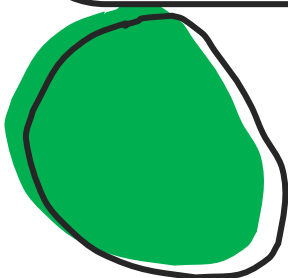
UNIVERSIDAD AUTÓNOMA DE GUERRERO
FACULTAD DE INGENIERÍA
Programabilidad de Redes.

Alumno:

Jared Duarte García.

802 vespertino, 24 de junio de 2024

Skills Exam:
DEVASC Final Skills
Assessment.



Evaluación de habilidades - DEVASC v1.0

Objetivos

Parte 1: Recopilar la documentación de API requerida

Parte 2: Codificar un Bot en Webex Teams.

Antecedentes/Situación

Python se puede utilizar para integrar llamadas de API desde diferentes servicios. En esta evaluación de habilidades, editará y modificará el código Python para un bot de Webex Teams que integra las API de Webex, MapQuest e ISS. El código comprueba continuamente si hay mensajes en una sala de Webex Teams especificada por el usuario, que comienza con una barra diagonal (/) seguida de un nombre de ciudad (por ejemplo, /Washington, DC). Cuando se encuentra un mensaje de este tipo, se extrae el nombre de la ciudad y se utiliza con el servicio de API de MapQuest para obtener las coordenadas GPS de la ciudad especificada. A continuación, estas coordenadas se utilizan con la API de la ISS para encontrar la fecha y hora de la próxima vez que la ISS sobrevolará. Finalmente, se envía un mensaje a la sala Webex Teams, informando al usuario sobre el siguiente sobrevuelo de la ISS para la ciudad consultada.

Una vez finalizado, el bot de Webex Teams:

- Pide al usuario su token de acceso o que use el token de acceso codificado.
- Muestra una lista de los usuarios de las salas de Webex Teams.
- Pregunta al usuario qué sala de Webex Teams debe supervisar para las consultas que comienzan con "/".
- Extrae el nombre de la ciudad de un mensaje que comience con "/" (por ejemplo, /Washington, DC -> Washington, DC).
- Solicita la latitud y longitud de una ciudad especificada mediante la API de MapQuest.
- Solicita la siguiente fecha y hora de sobrevuelo utilizando la API de la ISS.
- Envía la siguiente información de sobrevuelo de la ISS a la sala de Webex Teams especificada.

Recursos necesarios

- Una computadora con el sistema operativo de su elección.
- Acceso de desarrollador y claves API de MapQuest y Webex Teams
- VirtualBox o VMware.
- Máquina virtual (Virtual Machine) DEVASC.
- El script de estudiante **devasc-sa.py**.

Nota: Para proteger entornos de aplicaciones como Webex Teams de bots o intentos de acceso malintencionados, la mayoría de las API limitan la disponibilidad. Si realiza una gran cantidad de llamadas a la API, su llamada a la API puede bloquearse durante un período de tiempo específico. El tiempo de espera suele ser inferior a 5 minutos.

Instrucciones

Parte 1: Recopilar la documentación de API requerida

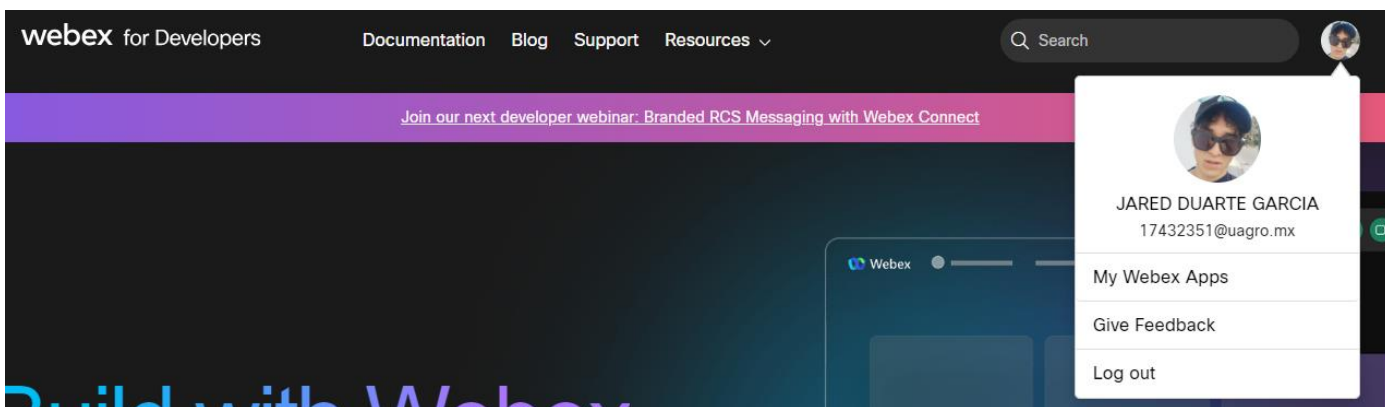
En esta parte, recopilará información de la documentación de la API de Webex, MapQuest e ISS. Esta información será necesaria en la Parte 2 cuando esté codificando el bot de Webex Teams. También debe investigar la biblioteca **time** de Python que usará para convertir marcas de tiempo de época en una fecha y hora legible para humanos.

Paso 1: Ejecute la máquina virtual de DEVASC.

Aunque puede completar esta evaluación de habilidades en otros entornos, estas instrucciones son sólo para la máquina virtual de DEVASC. No se incluyen instrucciones para otros entornos.

Paso 2: Investiga la documentación de las salas de Webex Teams y las API de mensajes.

- a. Inicie sesión en su cuenta de desarrollador para Webex.



- b. Localice y copie su token de acceso personal. ¿Cuál es la duración de su token?

webex for Developers **Documentation** Blog Support Resources ▾

webex for Developers Documentation Blog Support Resources ▾ Search

APIs

- API Behavior Changes
- Partners API Guide
- XML API Deprecation
- Access the API**
- REST API Basics
- Compliance

Webex APIs

- + Admin
- + Webex Calling Beta
- + Webex Calling
- + Webex for Broadworks

Access the Webex API

The Webex APIs give you easy access to the Webex Platform to build [Bots](#), [Integrations](#), or [Guest Issuer](#) apps. If you're ready to start using the Webex APIs, keep reading.

Your Personal Access Token

Bearer *****

This limited-duration personal access token is hidden for your security.

What's possible with the Webex APIs?

The Webex APIs provide your applications with direct access to the Cisco Webex Platform, giving you the ability to:

- [Create a Webex space and invite people](#)
- [Search for people in your company](#)
- [Post messages in a Webex space](#)
- [Get Webex space history](#) or be [notified in real-time](#) when new messages are posted by

What's possible with the Webex APIs?

- Accounts and Authentication
- Personal Access Tokens
- Methods & Content Types
- Next Steps
- Support Policy
- Getting Help

Copy Token

This personal access token is provided to test the API and will be valid for **12 hours** after **logging into** this site. It should not be used in production apps.

Cancel OK

Your Personal Access Token

Bearer *****

This limited-duration personal access token is hidden for your security.

Dura 12 horas.

- c. Busca la URL que mostrará todas las salas a las que pertenece. Registre el método HTTP y la URL:

GET <https://webexapis.com/v1/rooms>.

Evaluación de habilidades - DEVASC v1.0

- d. Busque la dirección URL que enumera todos los mensajes de una sala especificada. Registre el método HTTP y la URL:

GET <https://webexapis.com/v1/messages>

- e. Busque la dirección URL que creará un mensaje para una sala especificada. Registre el método HTTP y la URL.

POST <https://webexapis.com/v1/messages>

Paso 3: Investiga la clave de ubicaciones para la API de direcciones de MapQuest.

- a. Inicie sesión en su cuenta de desarrollador para MapQuest.

The screenshot shows the MapQuest Developer website. At the top is a dark navigation bar with the MapQuest Developer logo and links for Pricing & Plans, Documentation, Blog, Contact Us, and Login. Below the navigation bar is a large heading "Welcome to the family". Underneath this heading are three buttons: "Create New Account", "Login" (highlighted in blue), and "Request New Password". Below these buttons are two input fields: "Username or email address" with the text "Jared-D" and "Password" with masked characters. Below the password field is a "Log In" button and a link "Whoops! Forgot your password?". Below the login section is a "Manage Profile" section. It features an icon of a profile card with a checkmark and an "Edit Profile" button. The profile information is displayed in a table-like structure:

Full name	Jared Duarte Garcia
Company	Universidad Autónoma de Guerrero (UAGro)

- b. Localice y copie su clave de consumidor. ¿Cuándo caduca su llave?

El bot de Webex Teams utilizará los datos de ubicación devueltos por una llamada a la API de direcciones de MapQuest.

Create a New Key

Personal ACTIVE

Transaction Report

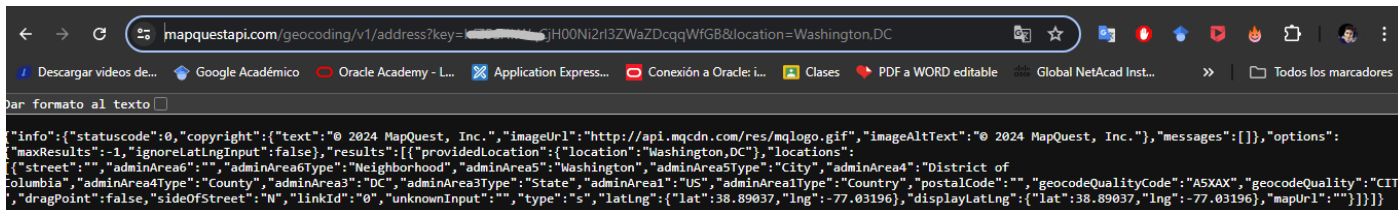
Consumer Key

[Redacted Consumer Key]

ⓧ Revoke Key

No tiene tiempo límite de caducidad.

- c. Abra Chromium y pegue la siguiente URL, reemplazando **your_api_key** por su clave MapQuest:
- https://www.mapquestapi.com/geocoding/v1/address?key=your_api_key&location=Washington,DC



- d. Observe que la clave de ubicaciones de MapQuest incluye claves de latitud y longitud de la ubicación especificada. Registre los valores **lat** y **lng** devueltos por MapQuest para Washington, D.C en el siguiente código.

```

{
  "info": {
    "statusCode": 0,
    "copyright": {
      "text": "© 2020 MapQuest, Inc.",
      "ImageURL": "http://api.mqcdn.com/res/mqlogo.gif ",
      "ImageAltText": "© 2020 MapQuest, Inc."
    },
    "messages": []
  },
  "options": {

```

```
    "MaxResults" :-1,
    "ThumbMaps": true,
    "IgnoreLatLnginput": false
  },
  "results": [
    {
      "providedLocation": {
        "location": "Washington,DC"
      },
      "locations": [
        {
          "street": "",
          "adminArea6": "",
          "adminArea6Type": "Neighborhood",
          "adminArea5": "Washington",
          "AdminArea5Type": "Ciudad",
          "adminArea4": "District of Columbia",
          "adminArea4Type": "County",
          "adminArea3": "DC",
          "adminArea3Type": "State",
          "adminArea1": "US",
          "adminArea1Type": "Country",
          "postalCode": "",
          "geocodeQualityCode": "A5XAX",
          "geocodeQuality": "CITY",
          "dragPoint": false,
          "sideOfStreet": "N",
          "linkID": "282772166",
          "unknownInput": "",
          "type": "s",
          "latlng": {
            "lat": 38.892062,
            "lng" :-77.019912
          },
          "displayLng": {
            "lat": _____,
            "lng": _____
          },
          <output omitted>
        }
      ]
    }
  ]
}
```



```
{
  "info": {
    "statusCode": 0,
    "copyright": {
      "text": "© 2024 MapQuest, Inc.",
      "imageUrl": "http://api.mqcdn.com/res/mqlogo.gif",
      "imageAltText": "© 2024 MapQuest, Inc."
    },
    "messages": []
  },
  "options": {
    "maxResults": -1,
    "ignoreLatLngInput": false
  },
  "results": [
    {
      "providedLocation": {
        "location": "Washington,DC"
      },
      "locations": [
        {
          "street": "",
          "adminArea6": "",
          "adminArea6Type": "Neighborhood",
          "adminArea5": "Washington",
          "adminArea5Type": "City",
          "adminArea4": "District of Columbia",
          "adminArea4Type": "County",
          "adminArea3": "DC",
          "adminArea3Type": "State",
          "adminArea1": "US",
          "adminArea1Type": "Country",
          "postalCode": "",
          "geocodeQualityCode": "A5XAX",
          "geocodeQuality": "CITY",
          "dragPoint": false,
          "sideOfStreet": "N",
          "linkId": "0",

```



```
"unknownInput": "",
"type": "s",
"latLng": {
  "lat": 38.89037,
  "lng": -77.03196
},
"displayLatLng": {
  "lat": 38.89037,
  "lng": -77.03196
},
"mapUrl": ""
}
]
}
}
```

Paso 4: Investigue la documentación de la API de tiempos de paso de la ISS.

- a. Buscar "documentación de la API ISS" en Internet.

The screenshot shows a web browser displaying the Open Notify API documentation. The browser's address bar shows the URL `open-notify.org/Open-Notify-API/ISS-Location-Now/`. The page has a dark header with the "Open Notify" logo and navigation links: "Home", "API Docs" (highlighted in orange), "Source Code", and "About". On the left side, there is a sidebar with a list of links: "API DOCS", "Examples", "ISS Current Location", and "People In Space". The main content area is titled "International Space Station Current Location" and includes a paragraph stating that the ISS is moving at approximately 28,000 km/h. Below this, there is an "Overview" section explaining that the API returns the current latitude and longitude of the ISS along with a Unix timestamp. An "Output" section shows a JSON response from the API endpoint `http://api.open-notify.org/iss-now.json`. The JSON object contains a "message" field with the value "success", a "timestamp" field with the value "UNIX_TIME_STAMP", and an "iss_position" field with a "latitude" value of "CURRENT_LATITUDE".

- b. En el sitio web de documentación de la API de la ISS, haga clic en la documentación de la API para los Tiempos de paso de la ISS

Open Notify API 0.2 documentation

[Open Notify API 0.2 documentation](#)[← Open APIs From Space | International Space Station Pass Predictions →](#)

International Space Station Current Location

GET /iss-now/v1/

GET /iss-now/

GET /iss-now.json

The International Space Station (ISS) is moving at close to 28,000 km/h so its location changes really fast! Where is it right now?

Previous topic

[Open APIs From Space](#)

Next topic

[International Space Station
Pass Predictions](#)
[Show page source](#)

Quick search

This is a simple api to return the current location above the Earth of the ISS. It returns the current latitude and longitude of the space station with a unix timestamp for the time the location was valid. This API takes no inputs.

Status Codes: • **200 OK** – when successful

Response JSON Object:

- **message** (*str*) – Operation status.
- **timestamp** (*int*) – Unix timestamp for this location.
- **iss_position** (*obj*) – Position on Earth directly below the ISS.
- **iss_position.latitude** (*int*) – Latitude
- **iss_position.longitude** (*int*) – Longitude

Example response:

- c. ¿Cuáles son los dos parámetros requeridos (llamados **query strings** en el sitio web) para la API de tiempos de paso de la ISS?

Status • **200 OK** – when successful

Codes: • **400 Bad Request** – if one or more inputs is out of range or invalid

Query Parameters:

- **lat** – latitude in decimal degrees of the ground station. **required** Range: -90, 90
- **lon** – longitude in decimal degrees of the ground station. **required** Range: -180, 180

lat y lon, para la latitud y longitud, respectivamente.

- d. ¿Cuáles son los parámetros opcionales para la API de tiempos de paso de la ISS?

- **alt** – altitude in meters of the ground station. optional. Range: 0, 10,000
- **n** – requested number of predictions. default: 5. May return less than requested

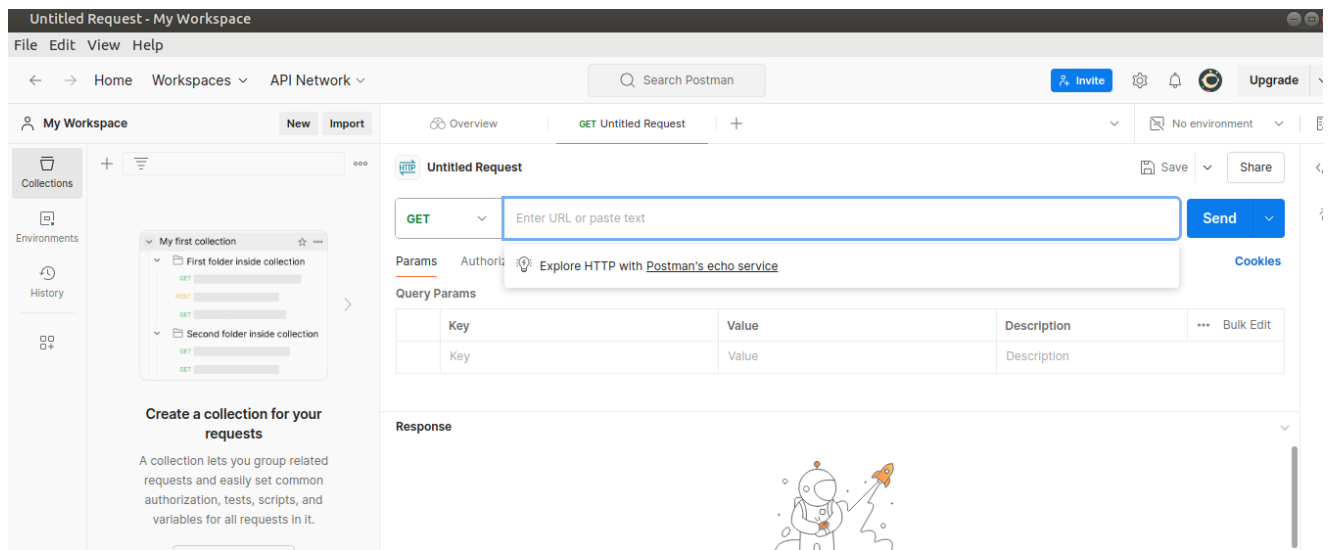
Los parámetros opcionales son “alt” y “n”. El primero para la altitud y el segundo al número solicitado de predicciones.

- e. ¿Cuál es la URL de la API de tiempos de paso de la ISS?

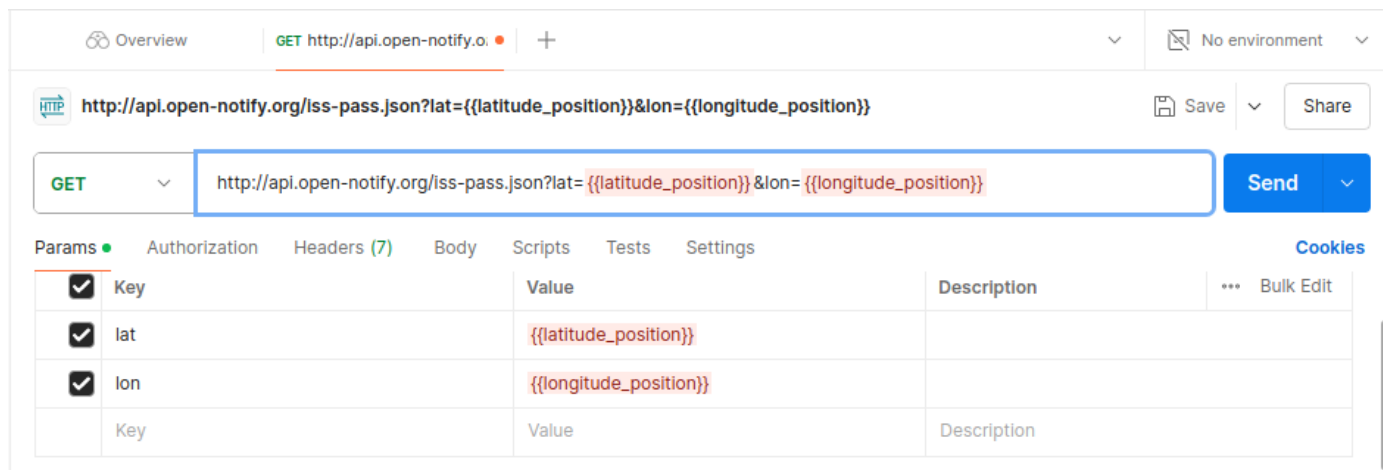
http://api.open-notify.org/iss-pass.json?lat={{latitude_position}}&lon={{longitude_position}}

Paso 5: Investigue la clave de respuesta para la API de tiempos de paso de la ISS.

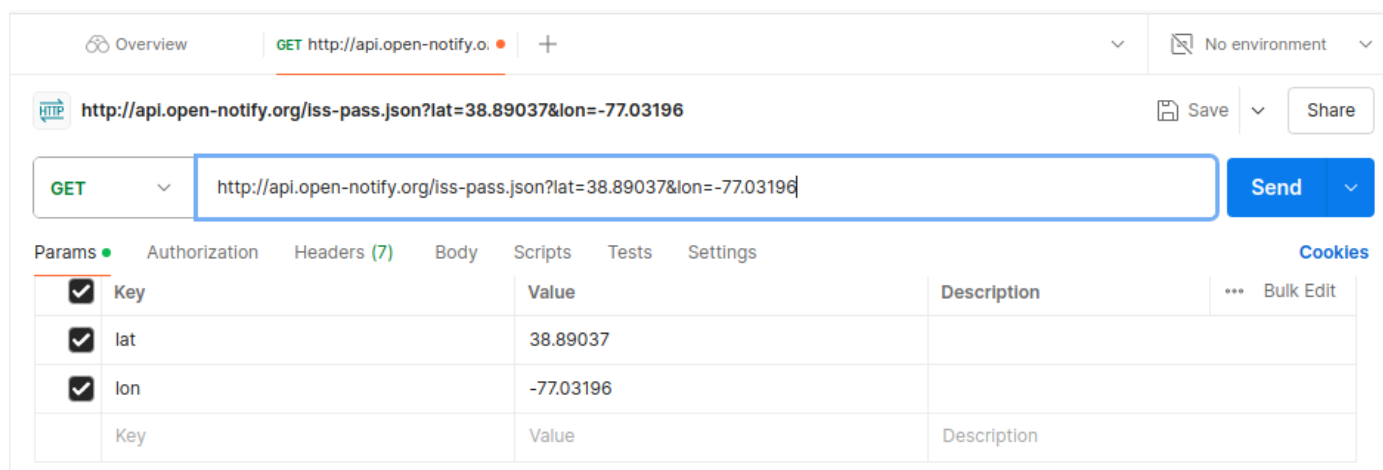
- a. Abra **Postman** y cree una nueva **Solicitud sin título**.



- b. Pegue la URL de tiempos de paso de la ISS.



- c. Reemplace los valores de latitud y longitud por los valores de Washington, D.C.



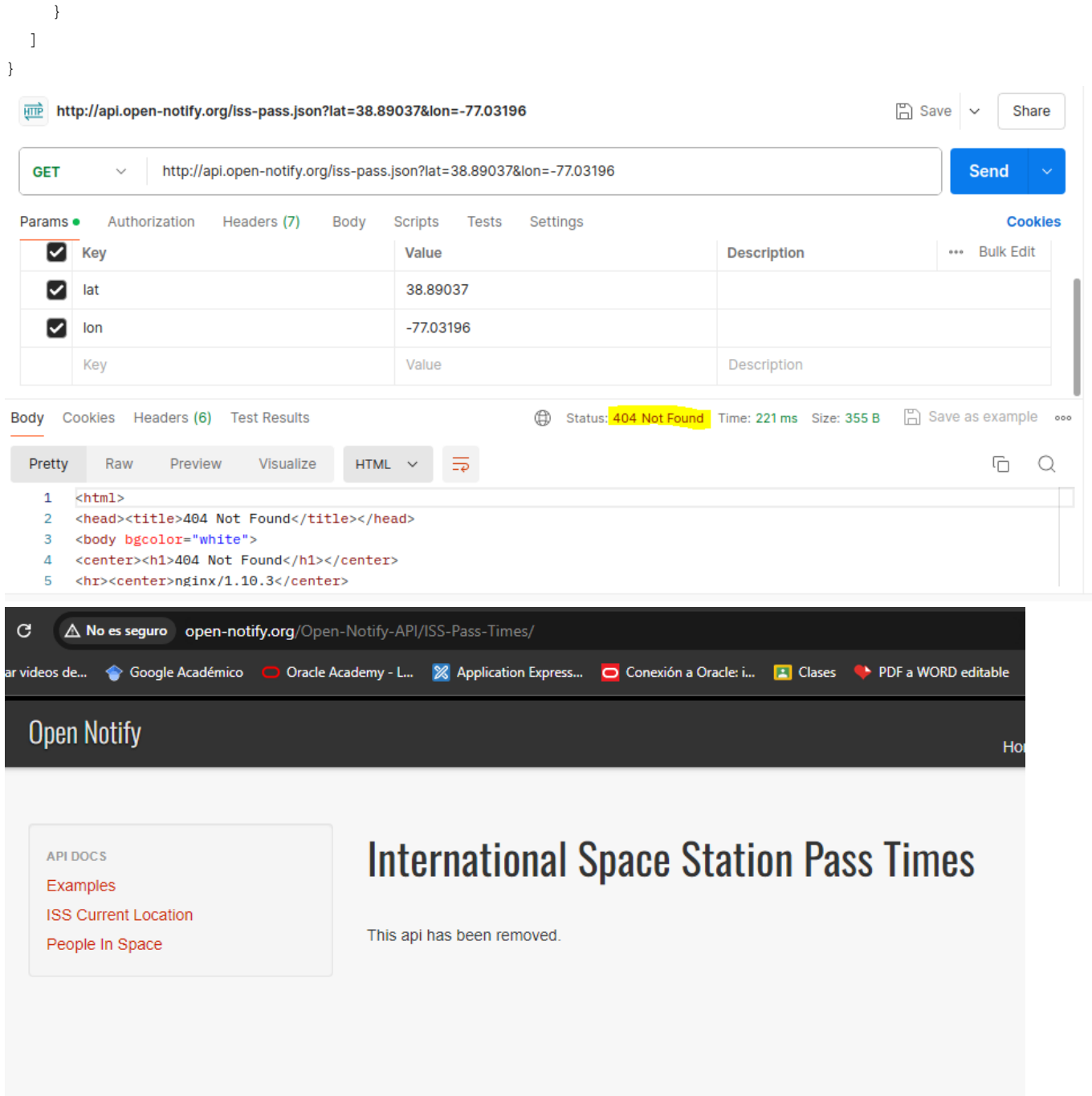
- d. Haga clic en **Send**. Debería obtener resultados similares a los siguientes, aunque sus valores de tiempo serán diferentes. De forma predeterminada, la API de tiempos de paso de la ISS devuelve los siguientes 5 pasos sobre la ubicación especificada.

```
{
  "message": "success",
  "request": {
    "altitud": 100,
    "fecha y hora": 1592669962,
    "latitude": _____,
    "longitude": _____,
    "pases": 5
  },
  "response": [
    {
      "duration": 602,
      "risetime": 1592672814
    },
    {
      "duration": 649,
      "risetime": 1592678613
    },
    {
      "duration": 423,
      "risetime": 1592684494
    },
    {
      "duration": 475,
      "risetime": 1592733041
    },
    {
      "duration": 651,
      "risetime": 1592738727
    }
  ]
}
```

```

    }
  ]
}

```



HTTP <http://api.open-notify.org/iss-pass.json?lat=38.89037&lon=-77.03196> Save Share

GET <http://api.open-notify.org/iss-pass.json?lat=38.89037&lon=-77.03196> Send

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Key	Value	Description
lat	38.89037	
lon	-77.03196	
Key	Value	Description

Body Cookies Headers (6) Test Results Status: 404 Not Found Time: 221 ms Size: 355 B Save as example

Pretty Raw Preview Visualize HTML

```

1 <html>
2 <head><title>404 Not Found</title></head>
3 <body bgcolor="white">
4 <center><h1>404 Not Found</h1></center>
5 <hr><center>nginx/1.10.3</center>

```

Open Notify

API DOCS
Examples
ISS Current Location
People In Space

International Space Station Pass Times

This api has been removed.

Buscando en la web me di cuenta del api había sido borrado y por eso al enviar la solicitud me genera el error 404.

Paso 6: Investigue las marcas de tiempo “epoch” y cómo se convierten a un formato legible para humanos.

En la Parte 2, usará la función **ctime** de la biblioteca **time** de Python para convertir el tiempo epoch en una fecha y hora legibles por humanos. Esa fecha y hora se incorporarán a un mensaje que el bot de Webex Teams publica en una sala.

Busque en Internet la documentación de la biblioteca **time** de Python para responder a las siguientes preguntas. Preferiblemente, debe revisar la documentación en python.org aunque las respuestas se pueden encontrar en otro lugar.

- a. En relación con el tiempo de computadora, ¿qué significa el término "epoch" ?

Una explicación de algunas terminologías y convenciones está en orden.

- Una *epoch* es el punto en el cual el tiempo empieza, el valor retorno de la función `time.gmtime(0)`. El cual es Enero, 1, 1970, 00:00:00(UTC) en todas las plataformas.

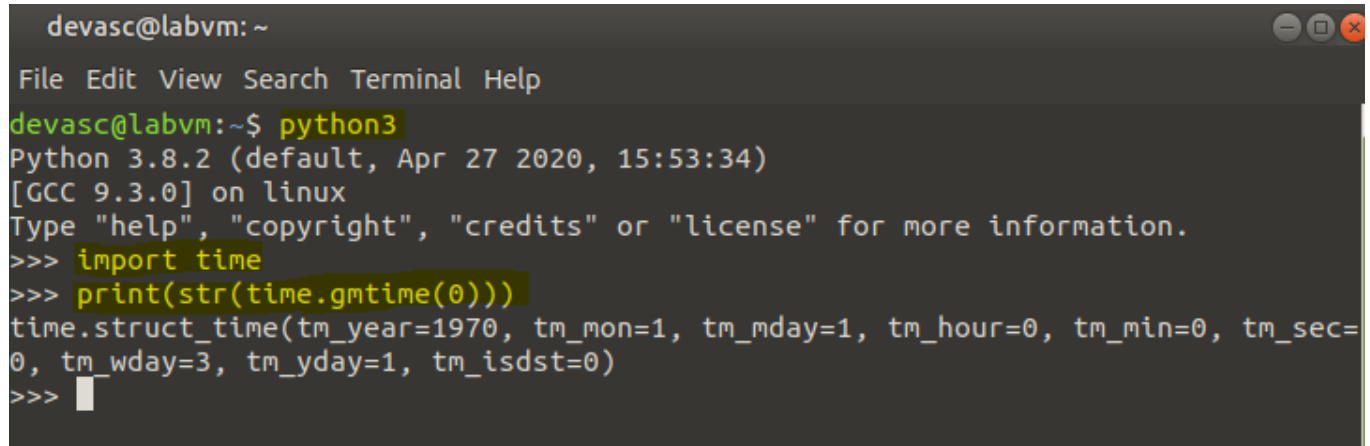
Una *epoch* es el punto en el cual el tiempo empieza, el valor retorno de la función `time.gmtime(0)`. El cual es Enero, 1, 1970, 00:00:00(UTC) en todas las plataformas.

- b. ¿Qué función de la biblioteca **time** devolverá el tiempo epoch en una plataforma dada?

La función `time.gmtime(0)`.

- c. Puede ver el año, mes, día, hora, etc. para el inicio de época con el siguiente código Python. Abra un terminal, inicie Python 3, importe la biblioteca `time` y luego reemplace **<function>** con la función que encontró anteriormente.

```
devasc @labvm: ~$ python3
Python 3.8.2 (predeterminado, 27 de abril de 2020, 15:53:34)
[GCC 9.3.0] en Linux
Escriba "help", "copyright", "credit" o "license" para obtener más información.
>>> import time
>>> print (str (_____))
time.struct_time (tm_year=1970, tm_mon=1, tm_mday=1, tm_hour=0, tm_min=0, tm_sec=0,
tm_wday=3, tm_yday=1, tm_isdst=0)
>>>
```



```
devasc@labvm: ~
File Edit View Search Terminal Help
devasc@labvm:~$ python3
Python 3.8.2 (default, Apr 27 2020, 15:53:34)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import time
>>> print(str(time.gmtime(0)))
time.struct_time(tm_year=1970, tm_mon=1, tm_mday=1, tm_hour=0, tm_min=0, tm_sec=0,
tm_wday=3, tm_yday=1, tm_isdst=0)
>>>
```

- d. El tiempo de inicio de la máquina virtual DEVASC es el mismo que el de Unix. ¿Cuándo comienza "epoch"?

Es distinto porque en la máquina virtual es Apr 27 2020, 15:53:34. Mientras que el "epoch" comienza en Jan 01 1970, 00:00:00 (lo equivalente a 01 Ene 1970, 00:00:00).

- e. El **risetime** en el paso 4 se expresa en el número de segundos desde el tiempo epoch. ¿Qué función de **time** convertirá **risetime** a un formato legible por humanos?

```
time.ctime([secs])
```

Convert a time expressed in seconds since the [epoch](#) to a string of a form: 'Sun Jun 20 23:21:05 1993' representing local time. The day field is two characters long and is space padded if the day is a single digit, e.g.: 'Wed Jun 9 04:26:40 1993'.

Si no se proporciona `secs` o `None`, se utiliza la hora actual retornada por `time()`. `ctime(secs)` es equivalente a `asctime(localtime(secs))`. La información de configuración regional no la utiliza `ctime()`.

Por lo tanto, la función sería “`time.ctime([secs])`”.

- f. ¿Cuál es la fecha del primer `risetime` en el paso 4?

```
>>> print (str (_____))
```

```
<Date and Time is printed here>
```

```
>>>
```

Desafortunadamente ese paso no lo pude completar por un error de que ya no existe la api.

Parte 2: Código de un bot de Webex Teams

Nota: Necesitará el script `devasc-sa.py` abierto y listo para editar para esta parte. Obtenga este script de su instructor.

En esta parte, usará su conocimiento de Python y las API REST, junto con la documentación de API que recopiló en la Parte 1 para finalizar el código del bot de Webex Teams.

Sugerencia: Guarde el archivo original con un nombre diferente en caso de que necesite comenzar de nuevo. Comente el código que aún no está probando. A medida que siga los pasos siguientes, guarde y ejecute el script con frecuencia. Utilice sentencias de impresión temporales para comprobar que las variables contienen los datos esperados.

Paso 1: Importar bibliotecas para solicitudes API, formato JSON y conversión de tiempo epoch.

```
# 1. Import libraries for API requests, JSON formatting, parsing URLs into components,

import requests
import json
import time
```

Paso 2: Complete la instrucción `if` para solicitar al usuario el token de acceso de Webex.

Para este paso, se le proporciona el primer mensaje de usuario y la parte `else` de una instrucción `if/else`. Debe codificar la parte `if` de la declaración que se ejecutará si el usuario dice "N" o "n" para usar el token de acceso codificado. La sentencia `if` comprueba el valor de `choice`, entonces si se introducen "N" o "n", le pide al usuario el valor del token. El valor introducido por el usuario se almacena en la variable `AccessToken`. La variable `AccessToken` debe construirse igual que la versión de la sentencia `else`.

```
# 2. Complete the if statement to ask the user for the Webex access token.
choice = input("Do you wish to use the hard-coded Webex token? (y/n) ")

✓ if choice == "N" or choice == "n":
    accessToken = input("What is your access token? ")
    accessToken = "Bearer " + accessToken
✓ else:
    accessToken = "Bearer NmI0YWIzOTEtYmE2Yi00ZmU5LWE5NzItYjdkY2U3MmYwYzNkNzkxY2RhYWEtYWY"
```

Paso 3: Proporcione la URL a la API de la sala de Webex Teams.

Utilice la documentación de la Parte 1 para especificar la API de la sala de Webex correcta que devolverá una lista de las salas de las que es miembro y las almacenará en la variable `r`.

```
# 3. Provide the URL to the Webex room API.
✓ r = requests.get("https://webexapis.com/v1/rooms",
                  headers = {"Authorization": accessToken})
```

Paso 4: Termine el bucle para imprimir el tipo y el título de cada sala.

La lista de salas almacenadas en la variable `r` se convierten en JSON y se almacenan en la variable `rooms`. Agregue una declaración de impresión que mostrará cada tipo y título de sala.

```
# 4. Create a loop to print the type and title of each room.
print("\nList of available rooms:")
rooms = r.json()["items"]
✓ for room in rooms:
    print ("Type: '" + room["type"] + "' Name: " + room["title"])
```

Paso 5: Proporcione la URL de la API de mensajes de Webex Teams.

Utilice la documentación de la Parte 1 para especificar la API de mensajes de Webex correcta. Cada segundo, el bot hará una llamada a esta API e imprimirá el último mensaje publicado en la sala.

```
# 5. Provide the URL to the Webex messages API.
r = requests.get("https://webexapis.com/v1/messages",
                params = GetParameters,
                headers = {"Authorization": accessToken})
```

Paso 6: Proporcione su clave de consumidor en la API de MapQuest.

Usted documentó esta clave en la Parte 1.

En realidad, en este paso se me solicita una cosa y dentro del script me pide otra. Sin embargo, optare por seguir primeramente las instrucciones en el script.


```
# 6. Provide the URL to the ISS Current Location API.
r = requests.get("http://api.open-notify.org/iss-now.json")

json_data = r.json()

if not json_data["message"] == "success":
    raise Exception("Incorrect reply from Open Notify API. Status code: {}".format(r.status_code))
```

Paso 7: Proporcione la dirección URL a la API de direcciones de MapQuest.

```
# 7. Record the ISS GPS coordinates and timestamp.

lat = json_data["125.8589"]
lng = json_data["-31.4635"]
timestamp = json_data["1719293926"]
```

En mi caso, por lo que entendí coloque la latitud y longitud que me devuelve la url del paso 6. La cual ingrese en Postman y eso me dio como resultado.

The screenshot shows a Postman interface. At the top, the URL bar displays `http://api.open-notify.org/iss-now.json`. Below it, the method is set to `GET`. The `Params` tab is selected, showing a table with two columns: `Key` and `Value`. The `Body` tab is also visible. The response body is shown in the `Body` tab, with the `JSON` format selected. The response is a JSON object with the following structure:

```
{
  "timestamp": 1719293926,
  "iss_position": {
    "longitude": "125.8589",
    "latitude": "-31.4635"
  }
}
```

Paso 8: Proporcione los valores clave de MapQuest para la latitud y la longitud.

Utilice la documentación de la Parte 1 para especificar el formato correcto para almacenar los valores de las claves de latitud y longitud.

```
# 8. Convert the timestamp epoch value to a human readable date and time.  
# Use the time.ctime function to convert the timestamp to a human readable date  
timeString = time.ctime(json_data["1719293926"])
```

Paso 9: Proporcione la URL a la API de tiempos de paso de la ISS.

Utilice la documentación de la Parte 1 para especificar la API de tiempos de paso de la ISS correcta.

Por lo que se me pide en el script tuve que crear una cuenta de graphhopper:

Jared

Last Name:

Duarte García

Email:

17432351@uagro.mx

Password:

.....

Confirm Password:

.....

Country:

Mexico

Company / Organisation:

Universidad Autonoma de Guerrero (UAGro)

☒ I accept the [terms](#) and [privacy policy](#)

☐ I want [the newsletter](#) (max. monthly)

Sign up

Manage your API Keys

Add API Key

Key	Description	Options
afdc51d7-df9d-4774-9582-c0289aeaad31	New	Show Examples Delete

```
# 9. Provide your Graphhopper API consumer key.
```

```
key = "afdc51d7-df9d-4774-9582-c0289aeaad31"
```

Paso 10: Proporcione los valores clave de la ISS para el tiempo de espera y la duración.

Utilice la documentación de la Parte 1 para especificar el formato correcto para almacenar los valores de las claves de tiempo de espera y duración.

```
# 10. Provide the URL to the Graphhopper GeoCoding API.
```

```
# Get location information using the Graphhopper GeoCoding API service using the
GeoURL = "https://graphhopper.com/api/1/geocode"
```

```
loc("&point="+lat+", "+lng
```

```
url = GeoURL + urllib.parse.urlencode({"key":key, "reverse":"true"}) + loc
```

```
r = requests.get(url)
```

Paso 11: Convierta el tiempo de ascenso (risetime) de epoch en una fecha y hora legible para humanos.

En la Parte 1, usted buscó la biblioteca **time**. Use su conocimiento para convertir el valor de epoch por la función **risetime** para que la hora y los datos sean humanamente legibles.

```
# 11. Store the location received from the Graphhopper.
    if len(json_data["hits"]) != 0:
        CountryResult = json_data["hits"][0]["country"]
        NameResult = json_data["hits"][0]["name"]
        if "state" in json_data["hits"][0]:
            StateResult = json_data["hits"][0]["state"]
        if "city" in json_data["hits"][0]:
            CityResult = json_data["hits"][0]["city"]
        if "street" in json_data["hits"][0]:
            StreetResult = json_data["hits"][0]["street"]
        if "housenumber" in json_data["hits"][0]:
            HouseResult = json_data["hits"][0]["housenumber"]
```

Paso 12: Complete el código para dar formato al mensaje de respuesta.

Utilice las variables especificadas para dar formato al mensaje de respuesta que se enviará a la sala Webex Teams. Por ejemplo, un mensaje publicado en la sala tendría el siguiente aspecto, donde la ubicación, el tiempo de ascenso y la duración se muestran en negrita.

En **Austin, Texas**, la ISS sobrevolará el **jue Jun 18 18:42:36 2020** durante **242** segundos.

```
if len(json_data["hits"]) == 0:
    responseMessage = "On {} (GMT), the ISS was flying over a body of water or unpopulated area."
else:
    responseMessage = "On {} (GMT), the ISS was flying over {}, {}".format(timeString, StateResult, CityResult)
    if "state" in json_data["hits"][0]:
        responseMessage += ", {}".format(StateResult)
    if "city" in json_data["hits"][0]:
        responseMessage += ", {}".format(CityResult)
    if "street" in json_data["hits"][0]:
        responseMessage += ", {} Street".format(StreetResult)
    if "housenumber" in json_data["hits"][0]:
        responseMessage += ", {}".format(HouseResult)
    responseMessage += ". ({}\\\"\\\", {}\\\"\\\")".format(lat, lng)
```

Paso 13: Completa el código para publicar el mensaje en la sala de Webex Teams.

El paso final en el programa del bot de Webex, es formatear el mensaje POST API que enviará el **ResponseMessage** a la sala de Webex Teams. Proporcione cada una de las variables necesarias y la URL para que la API de mensajes de Webex complete el código.

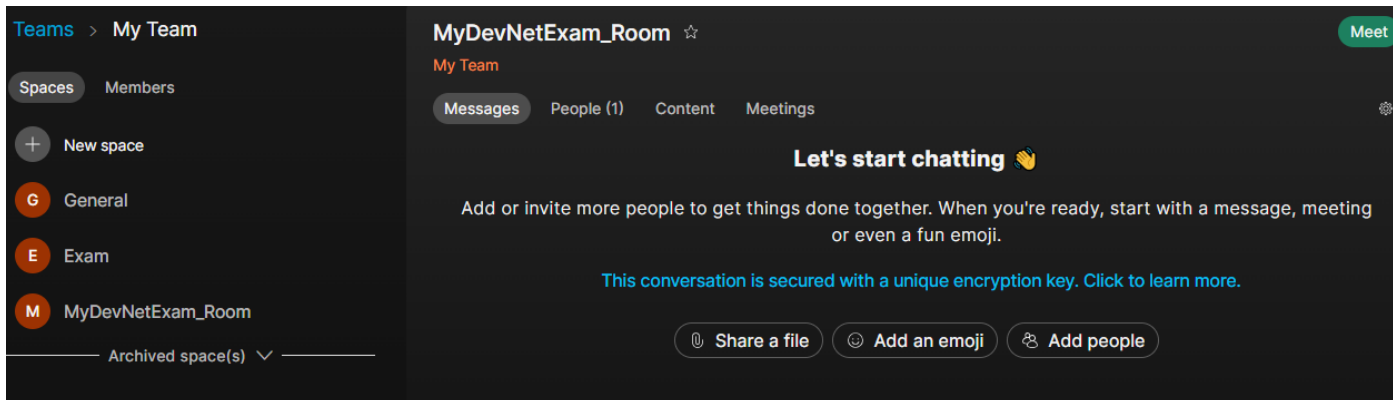
```
# 13. Complete the code to post the message to the Webex room.
# the Webex HTTP headers, including the Authorization and Content-Type
HTTPHeaders = {
    "Authorization": accessToken,
    "Content-Type": "application/json"
}

PostData = {
    "roomId": roomIdToGetMessages,
    "text": responseMessage
}

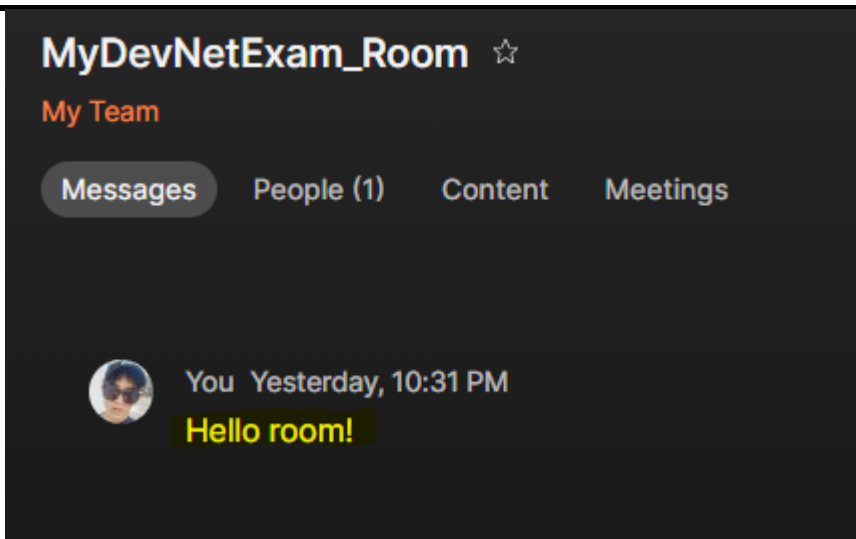
# Post the call to the Webex message API.
r = requests.post("https://webexapis.com/v1/messages",
                  data=json.dumps(PostData),
                  headers=HTTPHeaders)
```

Paso 14: Ejecuta el programa, pruébalo y soluciona los problemas que puedan surgir.

- a. En su cuenta de cliente de Webex Teams, cree una sala con el nombre de su elección, como **MyDevNetExam_Room**.



- b. ¡Publique un mensaje, como: Hello room!, para rellenar la sala con al menos un mensaje.



- c. Ejecute su programa y elija **My DEVASC SA Room** como la sala que supervisará el bot de Webex Teams.
- d. En My DEVASC SA Room, publique una ubicación con el formato **/location**. Los mensajes que comienzan con una barra diagonal inician el bot de Webex para que haga su trabajo. Por ejemplo, cuando se introduce Austin, Texas, debe ocurrir lo siguiente.

En una ventana de terminal:

```
devasc @labvm: ~/ $ python3 devasc-sa_sol.py
¿Desa utilizar el token codificado de Webex? (y/n) y
List of rooms:
Type: 'group' Name: My DEVASC SA Room
<rest of rooms listed>
¿Qué salas deberían ser monitoreadas para/ mensajes de ubicación? My DEVASC SA
Room
Found rooms with the word My DEVASC SA Room
My DEVASC SA Room
Sala encontrada: My DEVASC SA Room
Received message: Hello room!
Received message: Hello room!
Received message: Hello room!
Received message: Hello room!
<continue imprimiendo cada segundo>
```

```

devasc@labvm: ~/Documents
File Edit View Search Terminal Help
devasc@labvm:~/Documents$ python3 devasc-sa.py
Do you wish to use the hard-coded Webex token? (y/n) y

List of available rooms:
Type: 'group' Name: MyDevNetExam_Room
Type: 'group' Name: Exam
Type: 'group' Name: My Team
Type: 'group' Name: JARED's space
Which room should be monitored for the /seconds messages? MyDevNetExam_Room
Found rooms with the word MyDevNetExam_Room
MyDevNetExam_Room
Found room: MyDevNetExam_Room
Received message: Hello room!
Received message: Hello room!
Received message: Hello room!
Received message: Hello room!
Received message: Hello room!
Received message: Hello room!
Received message: Hello room!
^CTraceback (most recent call last):
  File "devasc-sa.py", line 93, in <module>
    time.sleep(1)
KeyboardInterrupt

```

En My DEVASC SA Room, agregue una ubicación.

/Austin, Texas

En la ventana de terminal, se imprime lo siguiente:

Mensaje recibido: /Austin, Texas

Ubicación: Austin, Texas

Coordenadas GPS de ubicación: 30.264979, -97.746598

Envío a Webex Teams: En Austin, Texas, la ISS sobrevolará el sábado 20 de junio 20:18:29 2020 durante 645 segundos.

Mensaje recibido: En Austin, Texas, la ISS sobrevolará el sábado 20 de junio 20:18:29 2020 durante 645 segundos.

Mensaje recibido: En Austin, Texas, la ISS sobrevolará el sábado 20 de junio 20:18:29 2020 durante 645 segundos.

Mensaje recibido: En Austin, Texas, la ISS sobrevolará el sábado 20 de junio 20:18:29 2020 durante 645 segundos.

Mensaje recibido: En Austin, Texas, la ISS sobrevolará el sábado 20 de junio 20:18:29 2020 durante 645 segundos.

<continues to print every 1 second>

En Webex Teams, se muestra lo siguiente.

En Austin, Texas, la ISS sobrevolará el sábado 20 de junio 20:18:29 2020 durante 645 segundos.

- e. En la ventana de su terminal, ingrese **Ctrl+C** para salir del programa.

Script del estudiante

Se adjunta script **devasc-sa.py**. En éste encontrarás comentarios que deberás sustituir por una o más líneas de código para completar una o más acciones.

Retroalimentación. Describe tu nivel habilidades

Me parece muy interesante lo realizado, todo iba genial hasta que me enfrenté al primer problema relacionado a la API de tiempos de paso de la ISS, que fue una de las partes donde me confundí.

Mi nivel de habilidades no es lo suficientemente bueno para que se me facilite en gran medida realizar todas las actividades solicitadas, sin embargo, con lo que aprendí, apuntes y apoyo revisando y consultando dudas en la web, pude resolver algunas de esas dudas, pero no quedó completamente cubierto todo lo solicitado puesto que también la última parte de hacer una prueba colocando una ubicación me diera sus datos (latitud, longitud, etc.), desafortunadamente no quedo la pude completar.

Me quedo con la experiencia de cómo uniendo varias herramientas se pueden hacer cosas muy interesantes como el caso de este script. Que en mi caso me resulto un proceso muy largo y hasta cierto punto tedioso o en el caso de que mi equipo se me cansaba y quería que le dejara descansar un momento.