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

## 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 librería **time** de Python que usará para convertir marcas de tiempo de época en una fecha y hora legible para humanos.

### 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.

### Investigue la documentación de las salas de Webex Teams y las API de mensajes.

* + - 1. Inicie sesión en su cuenta de desarrollador para Webex.
      2. Localice y copie su token de acceso personal. ¿Cuál es la duración de su token? \_\_\_\_\_\_\_\_\_\_\_\_\_
      3. Busca la URL que mostrará todas las salas a las que pertenece. Registre el método HTTP y la URL:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + - 1. Busque la dirección URL que enumera todos los mensajes de una sala especificada. Registre el método HTTP y la URL:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + - 1. Busque la dirección URL que creará un mensaje para una sala especificada. Registre el método HTTP y la URL.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Investigue la clave de ubicaciones para la API de direcciones de MapQuest.

* + - 1. Inicie sesión en su cuenta de desarrollador para MapQuest.
      2. 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.

* + - 1. 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

* + - 1. 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,

"Ing" ፦77.019912

},

"displayLng": {

"lat": \_\_\_\_\_\_\_\_\_\_\_\_\_\_,

"lng": \_\_\_\_\_\_\_\_\_\_\_\_\_\_

},

<output omitted>

}

]

}

]

}

### Investigue la documentación de la API de tiempos de paso de la ISS.

* + - 1. Buscar "documentación de la API ISS" en Internet.
      2. 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**
      3. ¿Cuáles son los dos parámetros requeridos (llamados **query strings** en el sitio web) para la API de tiempos de paso de la ISS?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + - 1. ¿Cuáles son los parámetros opcionales para la API de tiempos de paso de la ISS?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + - 1. ¿Cuál es la URL de la API de tiempos de paso de la ISS?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Investigue la clave de respuesta para la API de tiempos de paso de la ISS.

* + - 1. Abra **Postman** y cree una nueva **Solicitud sin título**.
      2. Pegue la URL de tiempos de paso de la ISS.
      3. Reemplace los valores de latitud y longitud por los valores de Washington, D.C.
      4. 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

}

]

}

### 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 librería **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.

* + - 1. En relación con el tiempo de computadora, ¿qué significa el término "epoch" ?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + - 1. ¿Qué función de la librería **time** devolverá el tiempo epoch en una plataforma dada?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + - 1. 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 librería 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)

>>>

* + - 1. El tiempo de inicio de la máquina virtual DEVASC es el mismo que el de Unix. ¿Cuándo comienza "epoch"?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + - 1. 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?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + - 1. ¿Cuál es la fecha del primer risetime en el paso 4?

>>> **print (str (**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**)))**

<Date and Time is printed here>

>>>

## 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.

### Importar librerías para solicitudes API, formato JSON y conversión de tiempo epoch.

### 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**.

### 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**.

### 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.

### 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.

### Proporcione su clave de consumidor en la API de MapQuest.

Usted documentó esta clave en la Parte 1.

### Proporcione la dirección URL a la API de direcciones de MapQuest.

### 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.

### 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.

### 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.

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

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

### 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.

### Complete 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.

### Ejecute el programa, pruébelo y solucione los problemas necesarios.

* + - 1. En su cuenta de cliente de Webex Teams, cree una sala con el nombre de su elección, como **My DEVASC SA Room**.
      2. ¡Publique un mensaje, como: Hello room!, para rellenar la sala con al menos un mensaje.
      3. Ejecute su programa y elija **My DEVASC SA Room** como la sala que supervisará el bot de Webex Teams.
      4. 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>

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.**

* + - 1. En la ventana de su terminal, ingrese **Ctrl+C** para salir del programa.

# Script del estudiante

A continuación, se muestra el contenido del script **devasc-sa.py**. Sin embargo, le recomendamos que utilice el archivo **devasc-sa.py** que obtenga de su instructor.

Copiar y pegar la siguiente secuencia de comandos desde un documento PDF o Word puede ser problemático.

**ADVERTENCIA**: Usted es responsable de corregir los errores de formato de código si decide copiar y pegar lo siguiente:

###############################################################

# Este programa:

# - Pide al usuario que ingrese un token de acceso o use el token de acceso codificado.

# - Enumera las salas de Webex Teams del usuario.

# - Pregunta al usuario qué sala de Webex Teams debe supervisar las solicitudes "/location".

# - Supervisa cada segundo la sala de Webex Teams seleccionada en busca de mensajes "/location".

# - Descubre las coordenadas GPS para la "ubicación" usando la API de MapQuest.

# - Descubre la fecha y la hora del siguiente paso de ISS sobre la "ubicación" usando la API de ISS

# - Formatea y envía los resultados a la sala de Webex Teams.

#

# El estudiante debe:

# 1. Importar librerías para solicitudes, JSON y tiempo.

# 2. Complete la declaración if para solicitar al usuario el token de acceso de Webex Temas.

# 3. Proporcione la URL de la API de sala de Webex Temas.

# 4. Cree un bucle para imprimir el tipo y el título de cada sala.

# 5. Proporcione la URL de la API de mensajes de Webex Teams.

# 6. Proporcione la clave de consumidor de la API de MapQuest.

# 7. Proporcione la URL de la API de direcciones de MapQuest.

# 8. Proporcione los valores clave de MapQuest para obtener la latitud y la longitud.

# 9. Proporcione la URL de la API de tiempos de paso de ISS.

# 10. Proporcione los valores clave ISS del tiempo de ascenso y duración.

# 11. Convierta el valor del tiempo epoch de subida en una fecha y hora legibles por humanos.

# 12. Complete el código para formatear el mensaje de respuesta.

# 13. Complete el código para publicar el mensaje en la sala de Webex Teams.

###############################################################

# 1. Importar bibliotecas para solicitudes, JSON y tiempo.

<!!!REEMPLAZAR con código para librerías>

# 2. Complete la declaración if para solicitar al usuario el token de acceso de Webex Teams.

choice = input ("¿Desea usar el token de Webex codificado? (y/n) ")

<!!!REEMPLAZAR con instrucciones if para pedir al usuario el token de acceso de WebexTeams!!!>

else:

AccessToken = "Bearer<!!!REEMPLAZAR con token codificado!!!>"

# 3. Proporcione la URL de la API de sala de Webex.

r = requests.get ("<!!!!REEMPLAZAR URL!!!>",

headers = {"Authorization": accessToken}

).

#####################################################################################

# NO EDITAR NINGÚN BLOCK CON r.status\_code

if not r.status\_code == 200:

raise Exception ("Respuesta incorrecta de la API de Webex Teams. Status code: {} Text: {}" .format (r.status\_code, r.text))

######################################################################################

# 4. Cree un bucle para imprimir el tipo y el título de cada sala.

print ("Lista de salas:")

rooms = r.json () ["items"]

for room in rooms:

<!!!REEMPLAZAR con código de impresión para terminar el bucle >

#######################################################################################

# BUSCAR SALA DE EQUIPOS DE WEBEX PARA MONITOREAR

# - Busca el nombre de sala proporcionado por el usuario.

# - Si se encuentra, imprima el mensaje "found", de lo contrario imprime el error.

# - Almacena valores para su uso posterior por bot.

# NO EDITAR CÓDIGO EN ESTE BLOQUE

#######################################################################################

while True:

RoomNameToSearch = input ("¿Qué sala debe ser monitoreada para mensajes /location? ")

RoomIDToGetMessages = None

for room in rooms:

if(room["title"].find(roomNameToSearch) != -1):

print ("Found rooms with the word " + roomNameToSearch)

print (sala ["title"])

RoomidTogetMessages = room ["id"]

RoomTitleTogetMessages = room ["title"]

print ("sala encontrada:" + RoomTitleToGetMessages)

break

if (RoomIDToGetMessages == None):

print ("Lo siento, no encontré ninguna sala con" + RoomNameToSearch +".")

print ("Inténtelo de nuevo...")

else:

break

######################################################################################

# CÓDIGO BOT DE WEBEX TEAMS

# Inicia el bot de Webex para escuchar y responder a los mensajes /location.

######################################################################################

while True:

time.sleep (1)

GetParameters = {

"roomId": RoomIDToGetMessages,

"max": 1

}

# 5. Proporcione la URL de la API de mensajes de Webex.

r = requests.get ("<!!!!REEMPLAZAR URL!!!>",

params = getParameters,

headers = {"Authorization": accessToken}

).

if not r.status\_code == 200:

raise Exception ("Respuesta incorrecta de la API de Webex Teams. Status code: {} Texto: {}" .format (r.status\_code, r.text))

json\_data = r.json ()

if len (json\_data ["items"]) == 0:

raise Exception ("No hay mensajes en la sala.")

messages = json\_data ["items"]

message = messages [0] ["text"]

print("Received message: " + message)

if message.find ("/") == 0:

location = message [1:]

# 6. Proporcione la clave de consumidor de la API de MapQuest.

MapsaPigetParameters = {

"location": location,

"clave": "<!!!REEMPLAZAR con su clave de API de MapQuest!!!>"

}

# 7. Proporcione la URL de la API de direcciones de MapQuest.

r = requests.get("<!!!REEMPLAZAR URL!!!>",

params = MapsaPigetParameters

).

json\_data = r.json()

if not json\_data ["info"] ["statuscode"] == 0:

raise Exception ("Respuesta incorrecta de MapQuest API. Status code: {}" .format (r.statuscode))

locationResults = json\_data ["results"] [0] ["ProvidedLocation"] ["location"]

print ("Ubicación:" + LocationResults)

# 8. Proporcione los valores clave de MapQuest para obtener la latitud y la longitud.

LocationLat = json\_data ["<!!!!REEMPLAZAR!!!> con la ruta a la tecla de latitud!!!> "]

locationLng = json\_data ["<!!!!REEMPLAZAR!!!> con la ruta a la clave de longitud!!!> "]

print ("Localización coordenadas GPS:" + str (LocationLat) + "," + str (LocationLng))

IssaPigetParameters = {

"lat": locationLat,

"lon": locationLng

}

# 9. Proporcione la URL de la API de tiempos de paso de ISS.

r = requests.get("<!!!REPLACE with URL!!!>",

params = IssaPigetParameters

).

json\_data = r.json()

if not"response" en json\_data:

raise Exception ("Respuesta incorrecta de la API open-notify.org. Status code: {} Texto: {}" .format (r.status\_code, r.text))

# 10. Proporcione los valores clave ISS del tiempo de espera y duración.

risetimeinEpochSeconds = json\_data ["<!!!!REEMPLAZAR!!!> con la ruta a la clave de tiempo de ascenso!!!> "]

durationInSeconds = json\_data ["<!!!!REEMPLAZAR!!!> con la ruta a la clave de duración!!!> "]

# 11. Convierta el valor de risetime epoch en una fecha y hora legible para humanos.

risetimeInFormattedString = <!!!REEMPLAZAR con código de conversión!!!>

# 12. Complete el código para formatear el mensaje de respuesta.

# Ejemplo de resultado de un mensaje de respuesta: En Austin, Texas, la ISS sobrevolará el jue Jun 18 18:42:36 2020 durante 242 segundos.

responseMessage = "In {} the ISS will fly over on {} for {} seconds.".format(<!!!REPLACE with required variables!!!>)

print ("Envío a Webex:" +ResponseSage)

# 13. Complete el código para publicar el mensaje en la sala de Webex.

HttpHeaders = {

"Authorization": <!!!REEMPLAZAR!!!>,

"Content-Type": "application/json"

}

PostData = {

"RoomID": <!!!REEMPLAZAR!!!>,

"text": <!!!REEMPLAZAR!!!>

}

r = requests.post ("<!!!!REEMPLAZAR URL!!!>",

data = json.dumps (<!!!REEMPLAZAR!!!>),

headers= <!!!REEMPLAZAR!!!>

).

if not r.status\_code == 200:

raise Exception ("Respuesta incorrecta de la API de Webex. Status code: {} Text: {}" .format (r.status\_code, r.text))