In Austin, Texas the ISS will fly over on Sat Jun 20 20:18:29 2020 for 645 seconds.

e. In your terminal window, enter **Ctrl+C** to exit the program.

Student Script

Below is the content of the **devasc-sa.py** script. However, we recommend that you use the **devasc-sa.py** file you obtain from your instructor. Copying and pasting the following script from a PDF or Word document can be problematic.

WARNING: You are responsible for correcting the code formatting errors if you choose to copy and paste the following

```
#############################################################
# This program:
# - Asks the user to enter an access token or use the hard coded access token.
# - Lists the user's Webex Teams rooms.
# - Asks the user which Webex Teams room to monitor for "/location" requests.
# - Monitors the selected Webex Team room every second for "/location" messages.
# - Discovers GPS coordinates for the "location" using MapQuest API.
# - Discovers the date and time of the next ISS flyover over the "location" using the
ISS API
# - Formats and sends the results back to the Webex Team room.
#
# The student will:
# 1. Import libraries for requests, json, and time.
# 2. Complete the if statement to ask the user for the Webex access token.
# 3. Provide the URL to the Webex Teams room API.
# 4. Create a loop to print the type and title of each room.
# 5. Provide the URL to the Webex Teams messages API.
# 6. Provide your MapQuest API consumer key.
# 7. Provide the URL to the MapQuest address API.
# 8. Provide the MapQuest key values for latitude and longitude.
# 9. Provide the URL to the ISS pass times API.
# 10. Provide the ISS key values risetime and duration.
# 11. Convert the risetime epoch value to a human readable date and time.
# 12. Complete the code to format the response message.
# 13. Complete the code to post the message to the Webex Teams room.
#############################################################

# 1. Import libraries for requests, json, and time.

import requests
import json
import time

# 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 = "Bearer " + input("Input token: ")
else:
```

© Enter copyright date here - 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

www.netacad.com

```
        accessToken = "Bearer <!!!REPLACEME with hard-coded token!!!>"

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

################################################################################
# DO NOT EDIT ANY BLOCKS WITH r.status_code
if not r.status_code == 200:
    raise Exception("Incorrect reply from Webex Teams API. Status code: {}. Text:
{}".format(r.status_code, r.text))
################################################################################

# 4. Create a loop to print the type and title of each room.
print("List of rooms:")
rooms = r.json()["items"]
for room in rooms:
    print(room)

################################################################################
#
# SEARCH FOR WEBEX TEAMS ROOM TO MONITOR
#  - Searches for user-supplied room name.
#  - If found, print "found" message, else prints error.
#  - Stores values for later use by bot.
# DO NOT EDIT CODE IN THIS BLOCK
################################################################################
#

while True:
    roomNameToSearch = input("Which room should be monitored for /location messages?
")
    roomIdToGetMessages = None

    for room in rooms:
        if(room["title"].find(roomNameToSearch) != -1):
            print ("Found rooms with the word " + roomNameToSearch)
            print(room["title"])
            roomIdToGetMessages = room["id"]
            roomTitleToGetMessages = room["title"]
            print("Found room : " + roomTitleToGetMessages)
            break

    if(roomIdToGetMessages == None):
        print("Sorry, I didn't find any room with " + roomNameToSearch + " in it.")
        print("Please try again...")
    else:
        break
```

© Enter copyright date here - 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

www.netacad.com

```
################################################################################
# WEBEX TEAMS BOT CODE
#  Starts Webex bot to listen for and respond to /location messages.
################################################################################

while True:
    time.sleep(1)
    GetParameters = {
                            "roomId": roomIdToGetMessages,
                            "max": 1
                    }
# 5. Provide the URL to the Webex Teams messages API.
    r = requests.get("https://webexapis.com/v1/messages",
                         params = GetParameters,
                         headers = {"Authorization": accessToken}
                    )

    if not r.status_code == 200:
        raise Exception( "Incorrect reply from Webex Teams API. Status code: {}. Text:
{}".format(r.status_code, r.text))

    json_data = r.json()
    if len(json_data["items"]) == 0:
        raise Exception("There are no messages in the room.")

    messages = json_data["items"]
    message = messages[0]["text"]
    print("Received message: " + message)

    if message.find("/") == 0:
        location = message[1:]
# 6. Provide your MapQuest API consumer key.
        mapsAPIGetParameters = {
                                 "location": location,
                                 "key": "<!!!REPLACEME with your MapQuest API Key!!!>"
                                }
# 7. Provide the URL to the MapQuest address API.
        r = requests.get("https://www.mapquestapi.com/geocoding/v1/address?",
                              params = mapsAPIGetParameters
                         )
        json_data = r.json()

        if not json_data["info"]["statuscode"] == 0:
            raise Exception("Incorrect reply from MapQuest API. Status code:
{}".format(r.statuscode))

        locationResults = json_data["results"][0]["providedLocation"]["location"]
        print("Location: " + locationResults)

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

© Enter copyright date here - 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

www.netacad.com

```
        locationLat = json_data["results"][0]["locations"][0]["displayLatLng"]["lat"]
        locationLng = json_data["results"][0]["locations"][0]["displayLatLng"]{"lng"]
        print("Location GPS coordinates: " + str(locationLat) + ", " +
str(locationLng))

        issAPIGetParameters = {
                                "lat": locationLat,
                                "lon": locationLng
                              }
# 9. Provide the URL to the ISS pass times API.
        r = requests.get("hhtp://api.open.notify.org/iss-pass.json?",
                             params = issAPIGetParameters
                         )

        json_data = r.json()

        if not "response" in json_data:
            raise Exception("Incorrect reply from open-notify.org API. Status code:
{}. Text: {}".format(r.status_code, r.text))

# 10. Provide the ISS key values risetime and duration.
        risetimeInEpochSeconds = json_data["response"][0]["risetime"]
        durationInSeconds      = json_data["response"][0]["duration"]

# 11. Convert the risetime epoch value to a human readable date and time.
        risetimeInFormattedString = time.ctime(risetimeInEpochSeconds)

# 12. Complete the code to format the response message.
#     Example responseMessage result: In Austin, Texas the ISS will fly over on Thu
Jun 18 18:42:36 2020 for 242 seconds.
        responseMessage = "In {} the ISS will fly over on {} for {}
seconds.".format(location, risetimeInFormattedString, durationInSeconds)

        print("Sending to Webex Teams: " +responseMessage)

# 13. Complete the code to post the message to the Webex Teams room.
        HTTPHeaders = {
                         "Authorization": accessToken,
                         "Content-Type": "application/json"
                       }
        PostData = {
                        "roomId": roomIdToGetMessages,
                        "text": responseMessage
                    }

        r = requests.post( "https://webexapis.com/v1/messages",
                           data = json.dumps(PostData),
                           headers = HTTPHeaders
                         )
        if not r.status_code == 200:
```

© Enter copyright date here - 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

```
            raise Exception("Incorrect reply from Webex Teams API. Status code: {}.
Text: {}".format(r.status_code, r.text)
```

© Enter copyright date here - 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public