Parsing JSON with a Python Application

Objectives

- Obtain a API Key.
- Import necessary modules.
- Create API request variables and construct a URL.
- Add user input functionality.
- Add a quit feature so that the user can end the application.
- Display trip information for time, distance, and fuel usage.
- Iterate through the JSON data to extract and output the directions.
- Display error messages for invalid user input.

Background / Scenario

In this lab, you will create three applications that retrieves JSON data from API, parses the data, and formats it for output to the user. You will use the GET Route request from the API. Review the GET API documentation here:

WheaterAPI.com

API-NBA

Rapid API

Free Public APIs for Developers

Required Resources

- Computer with Python installed according to the **Lab - PC Setup for Workshop**.
- Access to the internet.

Instructions

Step 1: Importing modules for the application.

To begin your script for parsing JSON data, you will need to import two modules from the Python library: **requests** and **urllib.parse**. The **request** module provides functions for retrieving JSON data from a URL. The **urllib.parse** module provides a variety of functions that will enable you to parse and manipulate the JSON data you receive from a request to a URL.

a. For each application, open a blank script file and save it **01_app_parse01.py**.

b. Import the **urllib.parse** and **requests** modules.

```
import urllib.parse
import requests
```

Step 2: Create variables for API request.

The first step in creating your API request is to construct the URL that your application will use to make the call. Initially, the URL will be the combination of the following variables:

- **main_api** – This is the main URL that you are accessing.
- **orig** – This is the parameter to specify your point of origin.
- **dest** – This is the parameter to specify your destination.
- **key** – This is the MapQuest API key you retrieved from the developer website.

a. Create variables to build the URL that will be sent in the request. Copy your key to the key variable.

```
main_api = "url API"
key = "your_api_key"
```

b. Combine the four variables **main_api** and **key** to format the requested URL. Use the **urlencode** method to properly format the address value. This function builds the parameters part of the URL and converts possible special characters in the address value (e.g. space into “+” and a comma into “%2C”).

```
url = main_api + urllib.parse.urlencode({"key": key })
```

c. Create a variable to hold the reply of the requested URL and print the returned JSON data. The **json_data** variable holds a Python’s Dictionary representation that is the **json** reply of the **get** method of the **requests** module. The **print** statement is used to check the returned data.

```
json_data = requests.get(url).json()
print(json_data)
```

Step 3: Test the URL request.

a. Run your **01_app_parse01.py** script and verify that it works. Troubleshoot your code, if necessary. Although your output might be slightly different, you should get a JSON response similar to the following:

b. Rerun the script to get different results.

Editar Selección Ver Ir Ejecutar Terminal Ayuda

Bienvenido | 01_app_parse01.py

home > blogch > Programacion-de-redes-2023 > R1 > 01_app_parse01.py > ...

```python
main_api = "https://www.mapquestapi.com/directions/v2/route"

# Parámetros para especificar el punto de origen y destino
orig = "Nueva+York,NY"
dest = "Los+Ángeles,CA"

# Clave de API de MapQuest
key = "Z76dPONA5SfqjzyOnRAIu26ZqPZ0c0Ht"

# Construir la URL completa
url = f"{main_api}?key={key}&from={orig}&to={dest}"

response = requests.get(url)
json_data = response.json()

# Imprimir la representación JSON de la respuesta
print("Datos JSON devueltos:")
print(json_data)
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS Python

```
nArea5Type': 'City', 'adminArea4': 'New York', 'adminArea4Type': 'County', 'adminArea3': 'NY', 'adminArea3Type': 'State', 'adminArea1': 'US', 'adminAre
a1Type': 'Country', 'postalCode': '', 'geocodeQualityCode': 'A5XAX', 'geocodeQuality': 'CITY', 'dragPoint': False, 'sideOfStreet': 'N', 'linkId': '0',
'unknownInput': '', 'type': 's', 'latLng': {'lat': 40.71453, 'lng': -74.00712}, 'displayLatLng': {'lat': 40.71453, 'lng': -74.00712}, 'mapUrl': ''}, {'
street': '', 'adminArea6': '', 'adminArea6Type': 'Neighborhood', 'adminArea5': 'Los Angeles', 'adminArea5Type': 'City', 'adminArea4': 'Los Angeles', 'a
dminArea4Type': 'County', 'adminArea3': 'CA', 'adminArea3Type': 'State', 'adminArea1': 'US', 'adminArea1Type': 'Country', 'postalCode': '', 'geocodeQua
lityCode': 'A5XAX', 'geocodeQuality': 'CITY', 'dragPoint': False, 'sideOfStreet': 'N', 'linkId': '0', 'unknownInput': '', 'type': 's', 'latLng': {'lat'
: 34.05357, 'lng': -118.24545}, 'displayLatLng': {'lat': 34.05357, 'lng': -118.24545}, 'mapUrl': ''}], 'locationSequence': [0, 1]}, 'info': {'statuscod
e': 0, 'copyright': {'text': '© 2023 MapQuest, Inc.', 'imageUrl': 'http://api.mqcdn.com/res/mqlogo.gif', 'imageAltText': '© 2023 MapQuest, Inc.'}, 'mes
sages': []}}
blogch@blogch:~/Documentos/Universidad/primera_API$
```

0 ⚠ 0 0 Lín. 5, col. 4 Espacios: 4 UTF-8 LF Python 3.10.12 64-bit

Step 4: Print the URL and check the status of the JSON request.

Now that you know the JSON request is working, you can add some more functionality to the application.

a. Save your script as **01_app_parse01.py.py**.

b. Delete the **print(json_data)** statement because you no longer need to test that the request is properly formatted.

```
# Imprimir la representación JSON de la respuesta
print("Datos JSON devueltos:")
```

c. Add the statements below, which will do the following:

- Print the constructed URL so that the user can see the exact request made by the application.

```
# Imprimir la URL construida
print("URL: " + url)
```

- Parse the JSON data to obtain the **statuscode** value.
- Start an **if** loop that checks for a successful call, which has a value of 0. Add a print statement to display the **statuscode** value and its meaning. The **\n** adds a blank line below the output.

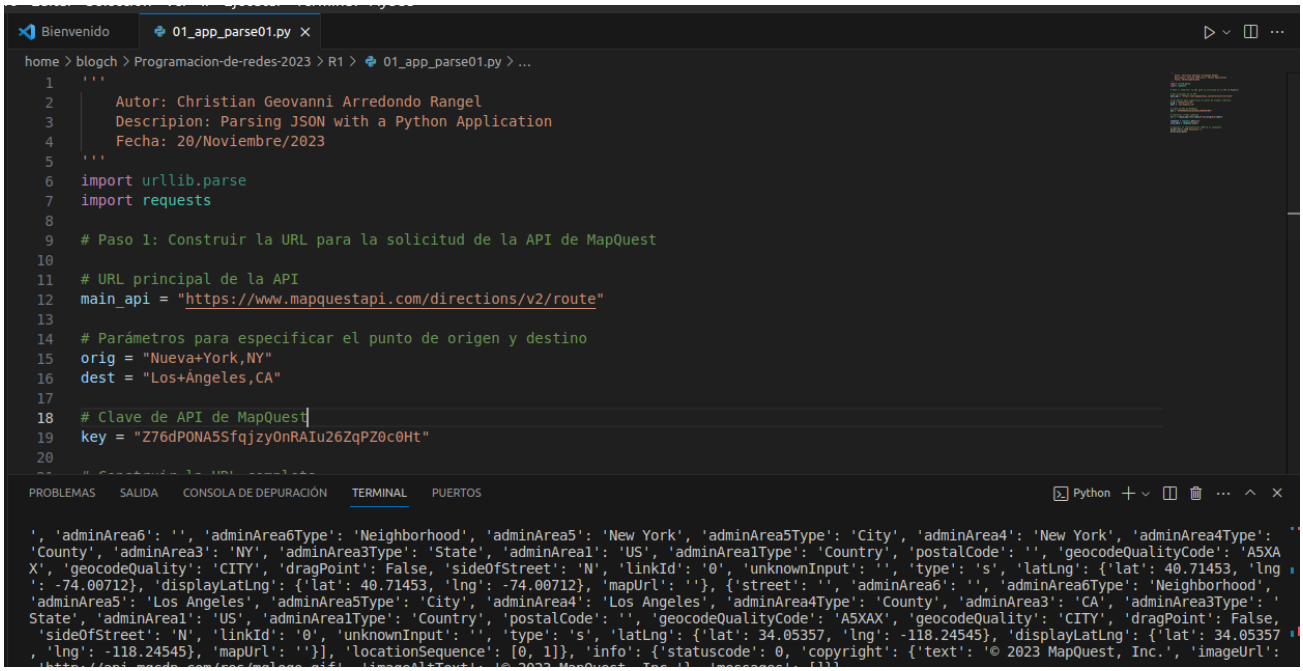
```
# Extraer el valor del código de estado de la respuesta JSON
json_status = json_data["info"]["statuscode"]

# Verificar si el código de estado es 0, que indica una llamada exitosa
if json_status == 0:
    # Imprimir un mensaje indicando que la llamada fue exitosa
    print("API Status: " + str(json_status) + " = A successful route call.\n")
```

```
print("URL: " + (url))

json_data = requests.get(url).json()
json_status = json_data["info"]["statuscode"]

if json_status == 0:
    print("API Status: " + str(json_status) + " = A successful route call.\n")
```

Later in this lab, you will add **elif** and **else** statements for different **statuscode** values.

Step 5: Test status and URL print commands.

Run your **01_app_parse01.py** script and verify that it works. Troubleshoot your code, if necessary.



Step 6: Add user input for starting location and destination.

Up to this point, you have used Washington and Baltimore as the static values for the location variables. However, the application requires that the user input these. Complete the following steps to update your application:

a. Delete the current **orig** and **dest** variables.

b. Rewrite the **orig** and **dest** to be within a **while** loop in which it requests user input for the starting location and destination. The **while** loop allows the user to continue to make requests for different directions.

c. Be sure all the remaining code is indented within the **while** loop.

```
while True:
    orig = input("Starting Location: ")
    dest = input("Destination: ")
    url = main_api + urllib.parse.urlencode({"key": key, "from":orig, "to":dest})
    print("URL: " + (url))

    json_data = requests.get(url).json()
    json_status = json_data["info"]["statuscode"]

    if json_status == 0:
        print("API Status: " + str(json_status) + " = A successful route call.\n")
```

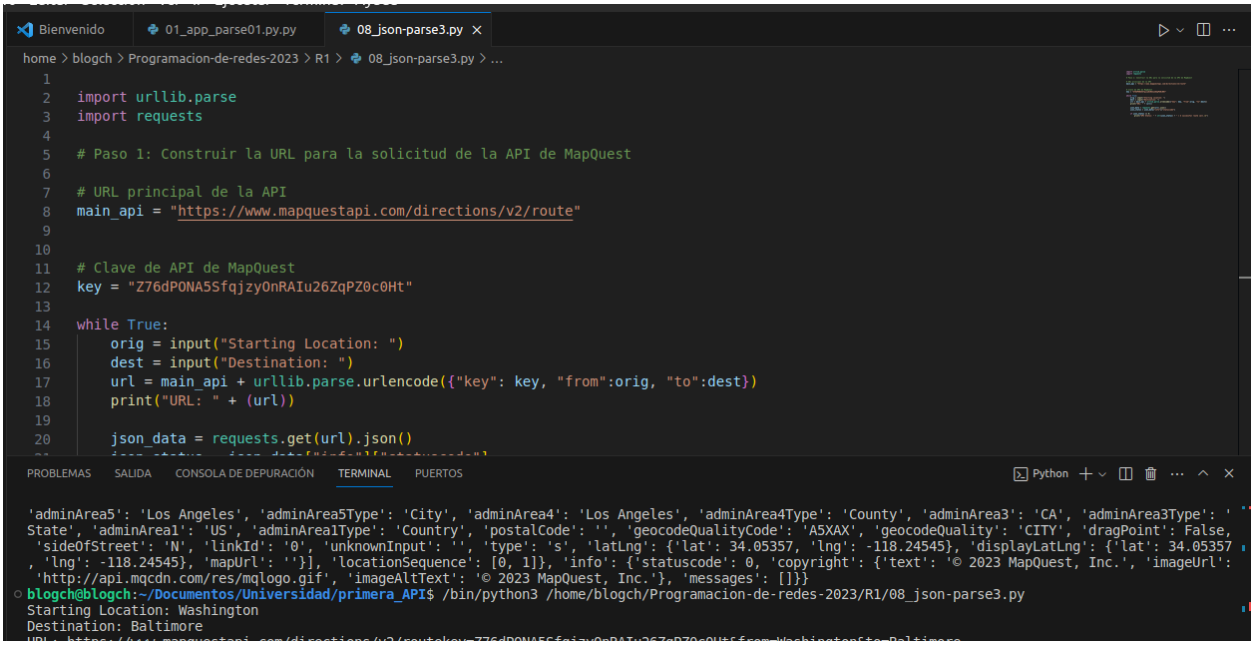
Step 7: Test user input functionality.

Run your **08_json-parse3.py** script and verify that it works. Troubleshoot your code, if necessary. You should get output similar to what is shown below. You will add quit functionality in the next step. For now, enter **Ctrl+C** to quit the application.

```
Starting Location: Washington
Destination: Baltimore
```



```
URL: https://www.mapquestapi.com/directions/v2/route?
key=your_api_key&from=Washington&to=Baltimore
API Status: 0 = A successful route call.

Starting Location: <Ctrl+C>
```



Step 8: Add the quit functionality to the application.

Instead of entering **Ctrl+C** to quit the application, you will add the ability for the user to enter **q** or **quit** as keywords to quit the application. Complete the following steps to update your application:

a. Save your script as **08_json-parse4.py**.

b. Add an **if** statement after each location variable to check if the user enters **q** or **quit**, as shown below:

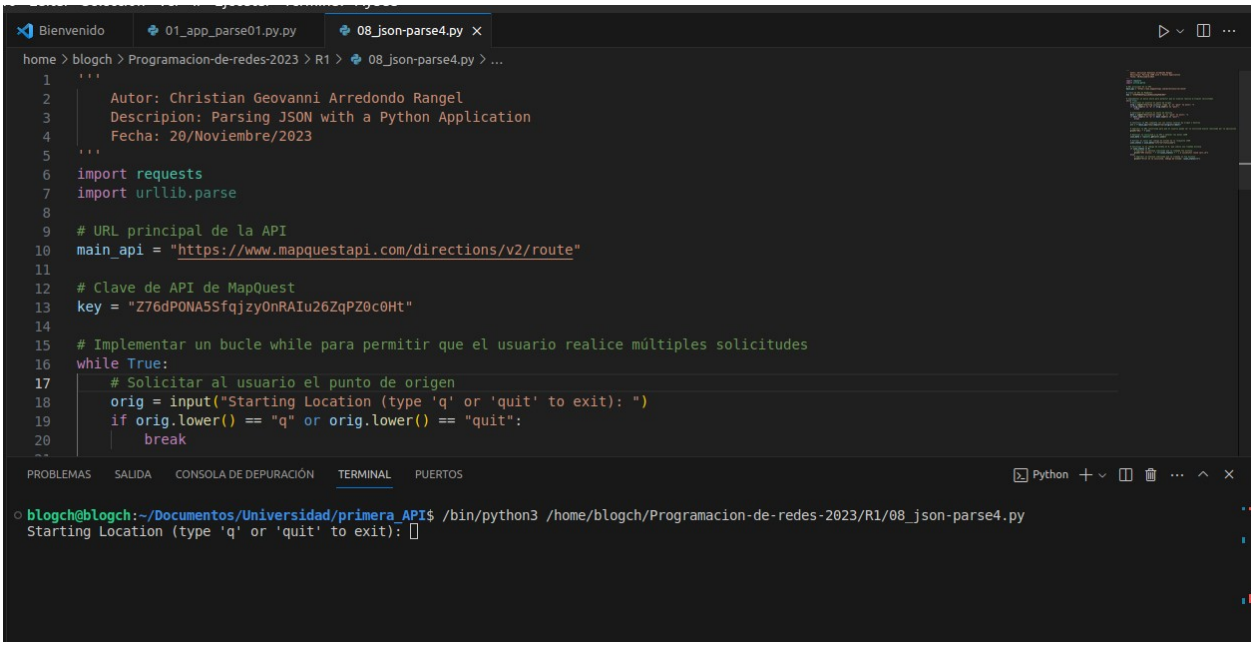
```
while True:
    orig = input("Starting Location: ")
    if orig == "quit" or orig == "q":
        break
    dest = input("Destination: ")
    if dest == "quit" or dest == "q":
        break
```

Step 9: Test the quit functionality.

Run your **08_json-parse4.py** script four times to test each location variable. Verify that both **quit** and **q** will end the application. Troubleshoot your code, if necessary. You should get output similar to the following:

```
Starting Location: q
>>>
Starting Location: quit
>>>
Starting Location: Washington
Destination: q
>>>
```

```
Starting Location: Washington
Destination: quit
>>>
```



Step 10: Parse and display some data about the trip.

a. Copy your URL into your web browser. If you collapse all the JSON data, you will see that there are two root dictionaries: **route and info**.

```
oute: {
  sessionId:
  "AI8A5wcAADgBAAAWAAAAEgAAAKcAAAB42mNYzsDAyMTAwMCekVqUapWcO1WcWRLIZdj5YlE3V_khC89Zx5ZHV0BpBiwApvF82lSwxv___9czbN_a6rlv4YfotUD6L5BmwAG-H-
  2LkWAEMgSbHBgDGDJbKgQTOV0cGBRdGAIYGgQYOsCqHFRYvBgAPSsqEONiK6o:car",
  realTime: 3928,
  distance: 44.963,
  time: 3225,
  formattedTime: "00:53:45",
  hasHighway: true,
  hasTollRoad: false,
  hasBridge: true,
  hasSeasonalClosure: false,
  hasTunnel: false,
  hasFerry: false,
  hasUnpaved: false,
  hasTimedRestriction: false,
  hasCountryCross: false,
▾ legs: [
    ▾ {
        index: 0,
        hasTollRoad: false,
        hasHighway: true,
        hasBridge: true,
        hasUnpaved: false,
        hasTunnel: false,
        hasSeasonalClosure: false,
        hasFerry: false,
        hasCountryCross: false,
        hasTimedRestriction: false,
        distance: 44.963,
        time: 3928,
        formattedTime: "01:05:28",
        origIndex: 0,
        origNarrative: "",
        destIndex: 0,
        destNarrative: "",
      ▾ maneuvers: [
```


b. Expand the **route** dictionary and investigate the rich data. There are values to indicate whether the route has toll roads, bridges, tunnels, highways, closures, or crosses into other countries. You should also see values for distance, the total time the trip will take, and fuel usage, as highlighted below. To parse and display this, specify the **route** dictionary and select key/value pair you want to print.


```
https://www.mapquestapi.com/directions/v2/

{
  - route: {
        hasTollRoad: false,
        hasBridge: true,
      + boundingBox: {…},
        distance: 38.089,
        hasTimedRestriction: false,
        hasTunnel: false,
        hasHighway: true,
        computedWaypoints: [ ],
      + routeError: {…},
        formattedTime: "00:49:19",
        sessionId: "5bc20e76-03aa-6750-02b4-1daf-0a1a4c2d1adc",
        hasAccessRestriction: false,
        realTime: 3309,
        hasSeasonalClosure: false,
        hasCountryCross: false,
        fuelUsed: 1.65,
      - legs: [
```


c. Save your script as **08_json-parse5.py**.

d. Below the API status print command, add print statements that display the from and to locations, as well as the **formattedTime**, **distance**, and **fuelUsed** keys.

e. Add a print statement that will display a double line before the next request for a starting location as shown below.

```
if json_status == 0:
    print("API Status: " + str(json_status) + " = A successful route call.\n")
    print("==============================================")
    print("Directions from " + (orig) + " to " + (dest))
    print("Trip Duration:   " + (json_data["route"]["formattedTime"]))
    print("Miles:           " + str(json_data["route"]["distance"]))
    print("Fuel Used (Gal): " + str(json_data["route"]["fuelUsed"]))
    print("==============================================")
```

f. Run **08_json-parse5.py** to see the following output:

```
Starting Location: Washington
Destination: Baltimore
URL: https://www.mapquestapi.com/directions/v2/route?
to=Baltimore&key=Your_api_key&from=Washington
API Status: 0 = A successful route call.

==============================================
Directions from Washington to Baltimore
Trip Duration:   00:49:19
Miles:           38.089
Fuel Used (Gal): 1.65
==============================================
Starting Location: q
>>>
```

g. MapQuest uses the imperial system and there is not a request parameter to change data to the metric system. Therefore, you should probably convert your application to display metric values, as shown below.

```
print("Kilometers:      " + str((json_data["route"]["distance"])*1.61))
print("Fuel Used (Ltr): " + str((json_data["route"]["fuelUsed"])*3.78))
```

h. Run the modified **08_json-parse5.py** script to see the following output:

```
Starting Location: Washington
Destination: Baltimore
URL: https://www.mapquestapi.com/directions/v2/route?
key=Your_api_key&to=Baltimore&from=Washington
API Status: 0 = A successful route call.

==============================================
Directions from c to Baltimore
```



```
Trip Duration:   00:49:19
Kilometers:      61.32329
Fuel Used (Ltr): 6.236999999999999
=============================================
Starting Location: q
>>>
```

```
blogch@blogch:~$ /bin/python3 /home/blogch/Programacion-de-redes-2023/R1/08_json-parse5.py
Starting Location (type 'q' or 'quit' to exit): Washington
Destination (type 'q' or 'quit' to exit): Baltimore
URL: https://www.mapquestapi.com/directions/v2/route?key=Z76dPONA5SfqjzyOnRAIu26ZqPZ0c0Ht&from=Washington&to=Baltimore
API Status: 0 = A successful route call.

=============================================
Directions from Washington to Baltimore
Trip Duration:   00:53:45
Miles:           44.963
Fuel Used information not available.
=============================================
Starting Location (type 'q' or 'quit' to exit):
```

i. Use the **"{:.2f}".format** argument to format the float values to 2 decimal places before converting them to string values, as shown below. Each statement should be on one line.

```
    print("Kilometers:      " + str("{:.2f}".format((json_data["route"]
["distance"])*1.61)))
    print("Fuel Used (Ltr): " + str("{:.2f}".format((json_data["route"]
["fuelUsed"])*3.78)))
```

Step 11: Test the parsing and formatting functionality.

Run your **08_json-parse5.py** script to verify that it works. Troubleshoot your code, if necessary. Make sure you have all the proper opening and closing parentheses. You should get output similar to the following:

```
Starting Location: Washington
Destination: Baltimore
URL: https://www.mapquestapi.com/directions/v2/route?
key=Your_api_key&to=Baltimore&from=Washington
API Status: 0 = A successful route call.

=============================================
Directions from Washington to Baltimore
Trip Duration:   00:49:19
Kilometers:      61.32
Fuel Used (Ltr): 6.24
=============================================
Starting Location: q
>>>
```



```
blogch@blogch:~$ /bin/python3 /home/blogch/Programacion-de-redes-2023/R1/08_json-parse5.py
.Starting Location (type 'q' or 'quit' to exit): Washington
Destination (type 'q' or 'quit' to exit): Baltimore
URL: https://www.mapquestapi.com/directions/v2/route?key=Z76dPONA5SfqjzyOnRAIu26ZqPZ0c0Ht&from=.Washington&to=Baltimore
API Status: 0 = A successful route call.

=============================================
Directions from .Washington to Baltimore
Trip Duration:   00:53:45
Miles:           44.963
Kilometers:      72.36
Fuel Used information not available.
=============================================
Starting Location (type 'q' or 'quit' to exit):
```

Step 12: Inspect the maneuvers JSON data.

a. Now you are ready to display the step-by-step directions from the starting location to the destination. Locate the **legs** list inside the route dictionary. The **legs** list includes one big dictionary with most of the JSON data.

b. Find the **maneuvers** list and collapse each of the seven dictionaries inside, as shown below.


```
https://www.mapquestapi.com/directions/v2/r
    - legs: [
        - {
              hasTollRoad: false,
              hasBridge: true,
              destNarrative: "Proceed to BALTIMORE, MD.",
              distance: 38.089,
              hasTimedRestriction: false,
              hasTunnel: false,
              hasHighway: true,
              index: 0,
              formattedTime: "00:49:19",
              origIndex: -1,
              hasAccessRestriction: false,
              hasSeasonalClosure: false,
              hasCountryCross: false,
            + roadGradeStrategy: […],
              destIndex: 3,
              time: 2959,
              hasUnpaved: false,
              origNarrative: "",
            - maneuvers: [
                + {…},
                + {…},
                + {…},
                + {…},
                + {…},
                + {…},
                + {…}
              ],
              hasFerry: false
          }
      ],
    - options: {
```

```
▾ maneuvers: [
  ▾ {
      index: 0,
      distance: 0.1939,
      narrative: "Head toward Jefferson Dr SW on 14th St NW (US-1). Go for 0.2 mi.",
      time: 46,
      direction: 4,
      directionName: "South",
      signs: [ ],
      maneuverNotes: [ ],
      formattedTime: "00:00:46",
      transportMode: "car",
    ▾ startPoint: {
          lat: 38.89037,
          lng: -77.03196
      },
      turnType: 0,
      attributes: 0,
      iconUrl: "",
    ▾ streets: [
          "US-1",
          "14th St NW",
```

c. Expand the first dictionary in the **maneuvers** list. Each dictionary contains a **narrative** key with a value, such as “Start out going north...”, as shown below. You need to parse the JSON data to extract the value for the **narrative** key to display inside your application.


```
https://www.mapquestapi.com/directions/v2/
time: 2959,
hasUnpaved: false,
origNarrative: "",
- maneuvers: [
  - {
      distance: 0.792,
    - streets: [
        "6th St",
        "US-50 E",
        "US-1 N"
      ],
      narrative: "Start out going north on 6
      turnType: 0,
    - startPoint: {
        lng: -77.019913,
        lat: 38.892063
      },
      index: 0,
      formattedTime: "00:02:05",
      directionName: "North",
      maneuverNotes: [ ],
      linkIds: [ ],
    - signs: [
      - {
```

```
{
  index: 1,
  distance: 1.528,
  narrative: "Turn left onto Independence Ave SW. Go for 1.5 mi.",
  time: 366,
  direction: 8,
  directionName: "East",
  signs: [ ],
  maneuverNotes: [ ],
  formattedTime: "00:06:06",
  transportMode: "car",
  startPoint: {
    lat: 38.88756999999999,
    lng: -77.03195
  },
  turnType: 6,
  attributes: 0,
  iconUrl: "",
  streets: [
    "Independence Ave SW",
    "Independence Ave SE"
  ],
  mapUrl: https://www.mapquestapi.com/staticmap/v5/map?
```


Step 13: Add a for loop to iterate through the maneuvers JSON data.

Complete the following steps to update your application:

a. Save your script as **08_json-parse6.py**.

b. Add a **for** loop below the second double line print statement. The **for** loop iterates through each **maneuvers** list and does the following:

   1) Prints the **narrative** value.

   2) Converts miles to kilometers with ***1.61**.

   3) Formats the kilometer value to print only two decimal places with the **"{:.2f}".format** function.

c. Add a print statement that will display a double line before the next request for a starting location, as shown below.

   Note: The second double line print statement is not indented within the **for** loop. Therefore, it is part of the previous **if** statement that checks the **statuscode** parameter.

```
print("Fuel Used (Ltr): " + str("{:.2f}".format((json_data["route"]["fuelUsed"])*3.78)))
print("==============================================")
for each in json_data["route"]["legs"][0]["maneuvers"]:
    print((each["narrative"]) + " (" + str("{:.2f}".format((each["distance"])*1.61) + " km)"))
print("==============================================\n")
```

Step 14: Activity - Test the JSON iteration.

Run your **08_json-parse6.py** script and verify that it works. Troubleshoot your code, if necessary. You should get an output similar to the following:

```
Starting Location: Washington
Destination: Baltimore
URL: https://www.mapquestapi.com/directions/v2/route?
key=Your_api_key&to=Baltimore&from=Washington
API Status: 0 = A successful route call.

Directions from Washington to Baltimore
Trip Duration:   00:49:19
Kilometers:      61.32
Fuel Used (Ltr): 6.24
==============================================
Start out going north on 6th St/US-50 E/US-1 N toward Pennsylvania Ave/US-1 Alt N.
(1.28 km)
Turn right onto New York Ave/US-50 E. Continue to follow US-50 E (Crossing into
Maryland). (7.51 km)
Take the Balt-Wash Parkway exit on the left toward Baltimore. (0.88 km)
Merge onto MD-295 N. (50.38 km)
Turn right onto W Pratt St. (0.86 km)
Turn left onto S Calvert St/MD-2. (0.43 km)
Welcome to BALTIMORE, MD. (0.00 km)
==============================================

Starting Location: q
>>>
```



```
PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    TERMINAL    PUERTOS                                    Python

URL: https://www.mapquestapi.com/directions/v2/route?key=Z76dPONA5SfqjzyOnRAIu26ZqPZ0c0Ht&from=Washington&to=Baltimore
API Status: 0 = A successful route call.

=============================================
Directions from Washington to Baltimore
Trip Duration:   00:53:45
Miles:           44.963
Kilometers:      72.36
Fuel Used information not available.
=============================================
Head toward Jefferson Dr SW on 14th St NW (US-1). Go for 0.2 mi. (0.31 km)
Turn left onto Independence Ave SW. Go for 1.5 mi. (2.46 km)
Continue on Pennsylvania Ave SE. Go for 246 ft. (0.08 km)
Turn left onto Independence Ave SE. Go for 0.4 mi. (0.62 km)
Keep right onto Independence Ave SE. Go for 1.5 mi. (2.47 km)
Continue on E Capitol St SE. Go for 0.3 mi. (0.45 km)
Take ramp onto DC-295 N (Kenilworth Ave NE) toward I-95 N/I-295 S. Go for 2.3 mi. (3.69 km)
Continue on MD-295 N. Go for 6.3 mi. (10.14 km)
Take the exit toward Baltimore onto I-95 N/I-495 N (Capital Beltway). Go for 4.2 mi. (6.84 km)
Keep right onto I-95 N toward Baltimore. Go for 25.7 mi. (41.39 km)
Take exit 53 toward Downtown/Inner Harbor onto I-395 N. Go for 1.5 mi. (2.42 km)
Keep right onto W Conway St toward Conway St. Go for 0.3 mi. (0.54 km)
Turn left onto Light St (MD-2 N). Go for 0.2 mi. (0.25 km)
Turn right onto E Pratt St. Go for 0.1 mi. (0.23 km)
Turn left onto Commerce St. Go for 430 ft. (0.13 km)
Continue on Commerce St. Go for 0.1 mi. (0.21 km)
Continue on N Holliday St. Go for 282 ft. (0.09 km)
Turn right onto E Fayette St. Go for 272 ft. (0.08 km)
Arrive at E Fayette St. (0.00 km)
=============================================

Starting Location (type 'q' or 'quit' to exit):
```

Step 15: Check for invalid user input.

Now you are ready to add one final feature to your application to report an error when the user enters invalid data. Recall that you started an **if** loop to make sure the returned **statuscode** equals 0 before parsing the JSON data:

```
json_status = json_data["info"]["statuscode"]

if json_status == 0:
    print("API Status: " + str(json_status) + " = A successful route call.\n")
```

a. But what happens if the **statuscode** is not equal to 0? For example, the user might enter an invalid location or might not enter one or more locations. If so, then the application displays the URL and asks for a new starting location. The user has no idea what happened. Try the following values in your application. You should see results similar to the following:

```
Starting Location: Washington
Destination: Beijing
URL: https://www.mapquestapi.com/directions/v2/route?
to=Beijing&key=your_api_key&from=Washington
Starting Location: Washington
Destination: Balt
URL: https://www.mapquestapi.com/directions/v2/route?
to=Balt&key=your_api_key&from=Washington
Starting Location: Washington
Destination:
URL: https://www.mapquestapi.com/directions/v2/route?
to=&key=your_api_key&from=Washington
Starting Location: q
```

b. Save your script as **08_jsont-parse7.py**.

c. To provide error information when this happens, add **elif** and **else** statements to your **if** loop. After the last double line print statement under the **if json_status == 0**, add the following **elif** and **else** statements:

```
        for each in json_data["route"]["legs"][0]["maneuvers"]:
            print((each["narrative"]) + " (" +
str("{:.2f}".format((each["distance"])*1.61) + " km)"))
        print("============================================\n")
    elif json_status == 402:
        print("***********************************************")
        print("Status Code: " + str(json_status) + "; Invalid user inputs for one or both
locations.")
        print("***********************************************\n")
    else:
        print("*********************************************************************")
        print("For Status Code: " + str(json_status) + "; Refer to:")
        print("https://developer.mapquest.com/documentation/directions-api/status-codes")
        print("*********************************************************************\n")
```

The **elif** statement prints if the **statuscode** value is 402 for an invalid location. The **else** statement prints for all other **statuscode** values, such as no entry for one or more locations. The **else** statement ends the **if**/**else** loop and returns the application to the **while** loop.

Step 16: Activity - Test full application functionality.

Run your **08_json-parse7.py** script and verify that it works. Troubleshoot your code, if necessary. Test all the features of the application. You should get output similar to the following:

```
Starting Location: Washington
Destination: Baltimore
URL: https://www.mapquestapi.com/directions/v2/route?
key=your_api_key&from=Washington&to=Baltimore
API Status: 0 = A successful route call.

Directions from Washington to Baltimore
Trip Duration:   00:49:19
Kilometers:      61.32
Fuel Used (Ltr): 6.24
============================================
Start out going north on 6th St/US-50 E/US-1 N toward Pennsylvania Ave/US-1 Alt N.
(1.28 km)
Turn right onto New York Ave/US-50 E. Continue to follow US-50 E (Crossing into
Maryland). (7.51 km)
Take the Balt-Wash Parkway exit on the left toward Baltimore. (0.88 km)
Merge onto MD-295 N. (50.38 km)
Turn right onto W Pratt St. (0.86 km)
Turn left onto S Calvert St/MD-2. (0.43 km)
Welcome to BALTIMORE, MD. (0.00 km)
============================================
```



```
blogch@blogch:~$ /bin/python3 /home/blogch/Programacion-de-redes-2023/R1/08_json-parse7.py
Starting Location (type 'q' or 'quit' to exit): Washington
Destination (type 'q' or 'quit' to exit): Baltimore
URL: https://www.mapquestapi.com/directions/v2/route?key=Z76dPONA5SfqjzyOnRAIu26ZqPZ0c0Ht&from=Washington&to=Baltimore
API Status: 0 = A successful route call.

=============================================
Directions from Washington to Baltimore
Trip Duration:   00:53:45
Miles:           44.963
Kilometers:      72.36
Fuel Used information not available.
=============================================
Head toward Jefferson Dr SW on 14th St NW (US-1). Go for 0.2 mi. (0.31 km)
Turn left onto Independence Ave SW. Go for 1.5 mi. (2.46 km)
Continue on Pennsylvania Ave SE. Go for 246 ft. (0.08 km)
Turn left onto Independence Ave SE. Go for 0.4 mi. (0.62 km)
Keep right onto Independence Ave SE. Go for 1.5 mi. (2.47 km)
Continue on E Capitol St SE. Go for 0.3 mi. (0.45 km)
Take ramp onto DC-295 N (Kenilworth Ave NE) toward I-95 N/I-295 S. Go for 2.3 mi. (3.69 km)
Continue on MD-295 N. Go for 6.3 mi. (10.14 km)
Take the exit toward Baltimore onto I-95 N/I-495 N (Capital Beltway). Go for 4.2 mi. (6.84 km)
Keep right onto I-95 N toward Baltimore. Go for 25.7 mi. (41.39 km)
Take exit 53 toward Downtown/Inner Harbor onto I-395 N. Go for 1.5 mi. (2.42 km)
Keep right onto W Conway St toward Conway St. Go for 0.3 mi. (0.54 km)
Turn left onto Light St (MD-2 N). Go for 0.2 mi. (0.25 km)
Turn right onto E Pratt St. Go for 0.1 mi. (0.23 km)
Turn left onto Commerce St. Go for 430 ft. (0.13 km)
Continue on Commerce St. Go for 0.1 mi. (0.21 km)
Continue on N Holliday St. Go for 282 ft. (0.09 km)
Turn right onto E Fayette St. Go for 272 ft. (0.08 km)
Arrive at E Fayette St. (0.00 km)
=============================================

Starting Location (type 'q' or 'quit' to exit):
```

```
Starting Location: Moscow
Destination: Beijing
URL: https://www.mapquestapi.com/directions/v2/route?
key=your_api_key&from=Moscow&to=Beijing
API Status: 0 = A successful route call.

Directions from Moscow to Beijing
Trip Duration:   84:31:10
Kilometers:      7826.83
Fuel Used (Ltr): 793.20
=============================================
Start out going west on Кремлёвская набережная/Kremlin Embankment. (0.37 km)
Turn slight right onto ramp. (0.15 km)
Turn slight right onto Боровицкая площадь. (0.23 km)
[output omitted]
Turn left onto 广场东侧路/E. Guangchang Rd. (0.82 km)
广场东侧路/E. Guangchang Rd becomes 东长安街/E. Chang'an Str. (0.19 km)
Welcome to BEIJING. (0.00 km)
=============================================
```



```
Starting Location (type 'q' or 'quit' to exit): Moscow
Destination (type 'q' or 'quit' to exit): Beijing
URL: https://www.mapquestapi.com/directions/v2/route?key=Z76dPONA5SfqjzyOnRAIu26ZqPZ0c0Ht&from=Moscow&to=Beijing
API Status: 0 = A successful route call.

=============================================
Directions from Moscow to Beijing
Trip Duration:   81:39:50
Miles:           4717.5619
Kilometers:      7592.16
Fuel Used information not available.
=============================================
Head toward Mokhovaya ulitsa on Tverskaya ulitsa. Go for 33 ft. (0.01 km)
Turn left onto ulitsa Okhotniy Ryad. Go for 0.2 mi. (0.30 km)
Continue on Teatral'niy proezd. Go for 0.3 mi. (0.42 km)
Turn left onto Lubyanskaya ploshchad' toward Bol'shaya Lubyanka ulitsa/Sretenka ulitsa. Go for 486 ft. (0.15 km)
Continue on ulitsa Bol'shaya Lubyanka. Go for 0.5 mi. (0.73 km)
Continue on ulitsa Sretenka toward M8. Go for 0.4 mi. (0.71 km)
Turn slightly right onto prospekt Mira. Go for 5.3 mi. (8.58 km)
Continue on Severyaninskiy puteprovod. Go for 0.4 mi. (0.62 km)
Continue on Yaroslavskoe shosse. Go for 14.7 mi. (23.59 km)
Keep left onto Yaroslavskoe shosse (M8). Go for 49.7 mi. (80.03 km)
Turn right onto Yaroslavskoe shosse (M8) toward Yaroslavl'/Yaroslavl'. Go for 12.6 mi. (20.32 km)
Turn left toward Pereslavl'-Zalesskiy. Go for 30 ft. (0.01 km)
Continue on 78K-0043. Go for 6.8 mi. (10.95 km)
Continue on Kholmogory (M8) toward Yaroslavl'/Yaroslavl/Arkhangel'sk. Go for 68.5 mi. (110.25 km)
Turn right onto Zolotoe kol'tso (R132) toward Aeroport Tunoshna/Airport Tunoshna/Kostroma/Kostroma. Go for 2.5 mi. (4.00 km)
Continue on Zolotoe kol'tso (R132). Go for 3.2 mi. (5.08 km)
Take the 2nd exit from roundabout onto Kostromskoe shosse (R132). Go for 4.8 mi. (7.73 km)
Continue on Zolotoe kol'tso (R132). Go for 37.1 mi. (59.65 km)
Continue on Kostromskoy Avtodorozhniy most toward Tsentr. Go for 0.8 mi. (1.27 km)
Continue on ulitsa Podlipaeva. Go for 0.4 mi. (0.57 km)
Continue on ulitsa Ivana Susanina. Go for 0.7 mi. (1.14 km)
Continue on Kalinovskaya ulitsa. Go for 0.2 mi. (0.32 km)
Turn right onto Galichskaya ulitsa. Go for 3.0 mi. (4.87 km)
```

Starting Location: **Washington**

Destination: **Beijing**

URL: https://www.mapquestapi.com/directions/v2/route?key=your_api_key&from=WashingtonTurn+right+onto+%E5%89%8D%E9%97%A8%E8%A5%BF%E5%A4%A7%E8%A1%97%2FQianmen+West+Street.+%281.01+km%29&to=Beijing

****

Status Code: 402; Invalid user inputs for one or both locations.

****

```
Arrive at West Chang'an Ave. (0.00 km)
=============================================

Starting Location (type 'q' or 'quit' to exit): Washington
Destination (type 'q' or 'quit' to exit): Beijing
URL: https://www.mapquestapi.com/directions/v2/route?key=Z76dPONA5SfqjzyOnRAIu26ZqPZ0c0Ht&from=Washington&to=Beijing
*********************************************
Status Code: 402; Invalid user inputs for one or both locations.
*********************************************

Starting Location (type 'q' or 'quit' to exit):
```

Starting Location: **Washington**

Destination: **Balt**

URL: https://www.mapquestapi.com/directions/v2/route?key=your_api_key&from=Washington&to=Balt

****

Status Code: 602; Refer to:

https://developer.mapquest.com/documentation/directions-api/status-codes

****

Starting Location: **Washington**

Destination:


```
URL: https://www.mapquestapi.com/directions/v2/route?
key=your_api_key&from=Washington&to=

**********************************************************************
Status Code: 611; Refer to:
https://developer.mapquest.com/documentation/directions-api/status-codes
**********************************************************************

Starting Location: q
>>>
```

