APIs and RESTful APIs

Application Programming Interface (API)

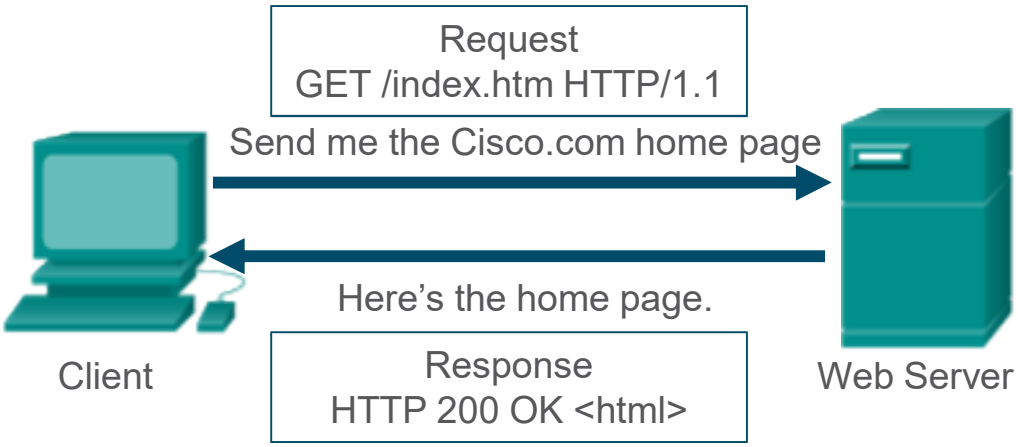
- An API allows one piece of software talk to another.
- An API is analogous to a power outlet.
- Without a power outlet, what would you have to do to power your laptop?
  - Open the wall
  - Unsheath wires
  - Splice wires together
  - Understand all the wires in the wall
- An API defines how a programmer can write a piece of software to extend an existing application's features or even build entirely new applications.

API Example

- Restaurant Recommendation App
  - Returns a list of relevant restaurants in the area
  - Integrates a third-party API to provide map functionality
  - The map API enforces a specification of an interface

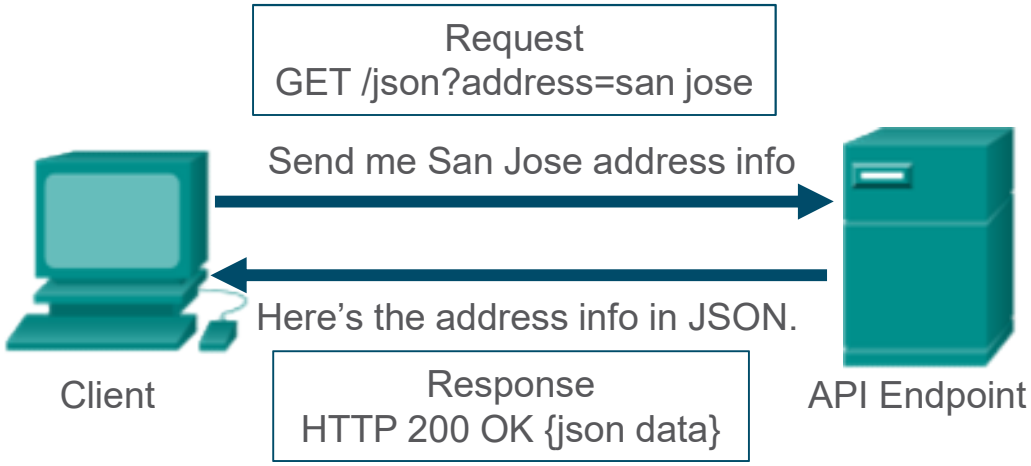
Web Services Interface using HTTP

- Web browsers use Hypertext Transfer Protocol (HTTP) to request (GET) a web page.
- If successfully requested (HTTP status code 200), web servers respond to GET requests with a Hypertext Markup Language (HTML) coded web page.



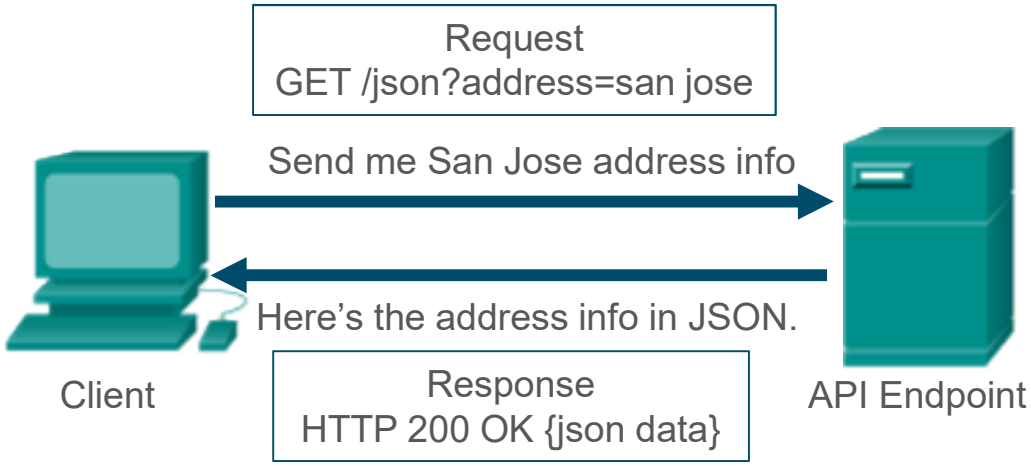
RESTful API using HTTP

- Representation State Transfer (REST) APIs use HTTP to interface with RESTful services.
- The HTTP request asks for JavaScript Object Notation (JSON) formatted data.
- If successfully formatted according to the API documentation, the server will respond with JSON data.



RESTful API using HTTP

- Representation State Transfer (REST) APIs use HTTP to interface with RESTful services.
- The HTTP request asks for JavaScript Object Notation (JSON) formatted data.
- If successfully formatted according to the API documentation, the server will respond with JSON data.



Anatomy of a RESTful Request

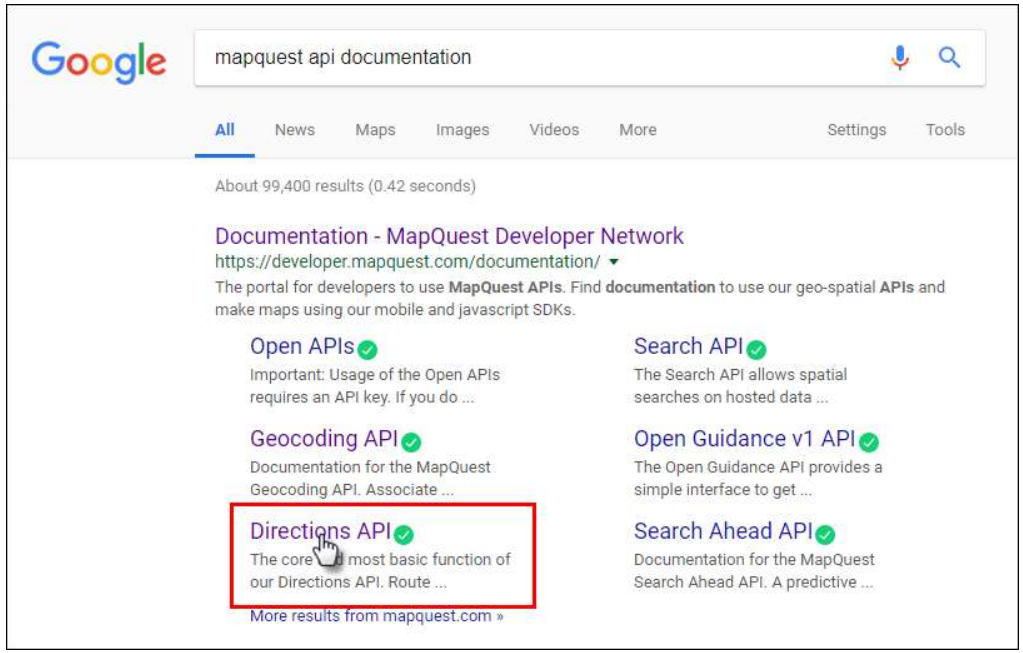
https://www.mapquestapi.com/directions/v2/route?outFormat=json&key=KEY&...

API Server | Resources | Format | Parameters

- API Server: The URL for the server that answers REST requests
- Resources: Specifies the API that is being requested.
- Format: Usually JSON or XML
- Parameters: Specifies what data is being requested

API Documentation

- Use an Internet search to find documentation for an API.



API Documentation

- The API documentation will specify...
  - The request format (JSON, XML, or text)
  - The request parameters
  - The response fields

Directions API

GET Route

Resource URL

```
http://www.mapquestapi.com/directions/v2/route
```

Resource Information

Response Formats	JSON, XML
Authentication	Yes (Requires Key)
Rate Limited	Yes

Request Parameters

Request Parameter	Description	Required?
key String	The API Key, which is needed to make requests to MapQuest services.	Yes

JSON and XML

JSON Response Data

http://api.open-notify.org/iss/v1/?lat=30.26715&lon=-97.74306

- Enter the above URL in your browser to get a JSON response.
- JSON data looks a lot like a Python dictionary.
- The third item in the dictionary is the response element.
- Inside the list are five dictionaries.

Not secure | api.open-notify.org/iss/v1


```
{
    message: "success",
  - request: {
        altitude: 100,
        datetime: 1539354878,
        latitude: 30.26715,
        longitude: -97.74306,
        passes: 5
    },
  - response: [
      - {
            duration: 254,
            risetime: 1539357421
        },
      - {
            duration: 638,
            risetime: 1539362981
        },
      - {
            duration: 537,
            risetime: 1539368837
        },
      - {
            duration: 108,
            risetime: 1539374926
        },
      - {
            duration: 450,
            risetime: 1539386568
        }
    ]
}
```

JSON Response Data

http://api.open-notify.org/iss/v1/?lat=30.26715&lon=-97.74306

- Collapse components of the JSON to better view its structure.
- In the example, JSON returns three root elements: message, request, and response
- The message root returns the HTTP status code.
- The request root shows default parameters and the parameters entered in the request.
- The response root typically returns an array with relevant responses.

XML Response Data

https://www.mapquestapi.com/directions/v2/route?outFormat=xml&key=KEY...

- Extensible Markup Language (XML) extends the functionality of HTML allowing web programmers to construct custom tags.
- To get XML data instead of JSON from the MapQuest API, add the outFormat=xml to the URL structure.
- You can see in the XML response that the embedded dictionaries have the same basic structure as JSON.

https://www.mapquestapi.com/directions/v2/route?outFormat=xml

This XML file does not appear to have any style information associated with it. The docum


```
▼<response>
  ▼<info>
    <statusCode>0</statusCode>
    <messages/>
  ▼<copyright>
      <imageUrl>http://api.mqcdn.com/res/mqlogo.gif</imageUrl>
      <imageAltText>© 2018 MapQuest, Inc.</imageAltText>
      <text>© 2018 MapQuest, Inc.</text>
    </copyright>
  </info>
 ▼<routeError>
    <errorCode>-400</errorCode>
    <message/>
  </routeError>
 ▼<route>
    <sessionId>5bc0dd98-036d-6750-02b4-1c24-0a30311e07a8</sessionId>
  ▼<options>
      <shapeFormat>raw</shapeFormat>
      <generalize>-1.0</generalize>
      <maxLinkId>0</maxLinkId>
      <narrativeType>text</narrativeType>
      <stateBoundaryDisplay>true</stateBoundaryDisplay>
      <countryBoundaryDisplay>true</countryBoundaryDisplay>
      <sideOfStreetDisplay>true</sideOfStreetDisplay>
      <destinationManeuverDisplay>true</destinationManeuverDisplay>
      <avoidTimedConditions>false</avoidTimedConditions>
      <enhancedNarrative>false</enhancedNarrative>
      <returnLinkDirections>false</returnLinkDirections>
      <timeType>0</timeType>
      <routeType>FASTEST</routeType>
      <locale>en_US</locale>
      <unit>M</unit>
```


Parsing JSON with Python

Demonstration - MapQuest API Application

08_parse-json_sol.py - /home/allan/Dropbox/ETW-Google_to_MapQuest/08_parse-json_sol.py (3.5.2)

File Edit Format Run Options Window Help

```python
#Replace "your_api_key" with your MapQuest API key

import urllib.parse
import requests

main_api = "https://www.mapquestapi.com/directions/v2/route?"
key = "your_api_key"

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

    dest = input("Destination: ")
    if dest == "quit" or dest == "q":
        break

    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")
        print("Directions from " + (orig) + " to " + (dest))
        print("Trip Duration:   " + str(json_data["route"]["formattedTime"]))
        print("Kilometers:      " + str("{:.2f}".format((json_data["route"]["distance"])*1.61)))
        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")
    elif json_status == 402:
        print("\n*****************************************************************")
        print("Staus Code: " + str(json_status) + "; Invalid user inputs for one or both locations.")
        print("*****************************************************************\n")
    else:
        print("\n*****************************************************************************")
        print("Staus Code: " + str(json_status) + "; Refer to:")
        print("https://developer.mapquest.com/documentation/directions-api/status-codes")
        print("*****************************************************************************\n")
```

Ln: 7 Col: 39

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
== RESTART: /home/allan/Dropbox/ETW-Google_to_MapQuest/08_parse-json_sol.py ==
Starting Location: Washington
Destination: Baltimore
URL: https://www.mapquestapi.com/directions/v2/route?to=Baltimore&from=Washingto
n&key=your_api_key
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
>>>
```

Ln: 26 Col: 4

MapQuest API Application Objectives

To build this application, you will complete the following objectives:

- Obtain a MapQuest 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.

Authenticating a RESTful Request

https://www.mapquestapi.com/directions/v2/route?key=your_api_key&to=Baltimore&...

API Server | Resources | Token | Parameters

- **None**: The API resource is public and anybody can place the request.
- **Basic HTTP**: The username and password are passed to the server in an encoded string.
- **Token**: A secret key generally retrieved from the Web API developer portal.
- **Open Authorization (OAuth)**: An open standard for retrieving an access token from an Identity Provider. The token is then passed with each API call.

Activity - Get Your MapQuest API Key

1. Go to: https://developer.mapquest.com/.
2. Click **Sign Up** at the top of the page.
3. Fill out the form to create a new account. For **Company**, enter **Cisco Networking Academy Student**.
4. After clicking **Sign Me Up**, you are redirected to the **Manage Keys** page.
5. Click **Approve All Keys** and expand **My Application**.
6. Copy your **Consumer Key** to Notepad for future use.

Importing Modules

- Open a blank script file and save it as **08_parse-json1.py**.
- Import modules

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

Create Variables for API Request

Add the following variables to your script:

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

Combine the variables into the **url** variable using the **urlencode** method to properly format the address variable

```
main_api = "https://www.mapquestapi.com/directions/v2/route?"
orig = "Washington"
dest = "Baltimaore"
key = "your_api_key"
url = main_api + urllib.parse.urlencode({"key": key, "from":orig, "to":dest})
```

Create the JSON Request

- Create a variable that uses the **get** method to request JSON data from the submitted URL.
- Print the results to verify the request was successful.

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

Activity - Test the URL Request

1. Run your **08_json-parse1.py** script and verify it works.
2. Troubleshoot your code, if necessary.
3. You should get a similar JSON response to what is shown here.

```
========== RESTART: /home/user/08_parse-json1.py ==========
{'route': {'distance': 38.089, 'hasHighway': True,
'hasUnpaved': False, 'hasAccessRestriction': False,
'options': {'mustAvoidLinkIds': [], 'maxWalkingDistance': -
1, 'manmaps': 'true', 'urbanAvoidFactor': -1,
'stateBoundaryDisplay': True, 'cyclingRoadFactor': 1,
'routeType': 'FASTEST', 'countryBoundaryDisplay': True,
'drivingStyle': 2, 'highwayEfficiency': 22,
'narrativeType': 'text', 'routeNumber': 0,
'tryAvoidLinkIds': [], 'generalize': -1,
'returnLinkDirections': False, 'doReverseGeocode': True,
'avoidTripIds': [], 'timeType': 0, 'sideOfStreetDisplay':
True, 'filterZoneFactor': -1, 'walkingSpeed': -1,
'useTraffic': False, 'unit': 'M', 'tr
<output omitted>
>>>
```

Print the URL and Check the Status of the JSON Request

1. Save your script as **08_json-parse2.py**.
2. Print the URL
3. Store the request status
4. Create an if loop that prints request status if **statuscode** is 0.

```
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")
```

Activity - Test Status and URL Print Commands

1. Run your **08_json-parse2.py** script and verify it works.
2. Troubleshoot your code, if necessary.
3. You should get a similar output to what is shown here.

```
========== RESTART: /home/user/08_parse-json2.py ==========
URL:
https://www.mapquestapi.com/directions/v2/route?key=your_ap
i_key&from=Washington&to=Baltimore
API Status: 0 = A successful route call.

>>>
```

Add User Input for Address

1. Save your script as **08_json-parse3.py**.
2. Delete the current **orig** and **dest** variables.
3. Rewrite the **orig** and **dest** to be within a while loop in which it requests user input for the starting location and destination.
4. 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))
```

Activity - Test User Input

1. Run your **08_json-parse3.py** script and verify it works.
2. Troubleshoot your code, if necessary.
3. You should get a similar output to what is shown here.

```
========== RESTART: /home/user/08_parse-json3.py ==========
Starting Location: Washington
Destination: Baltimore
URL:
https://www.mapquestapi.com/directions/v2/route?key=your_ap
i_key&from=Washington&to=Baltimore
API Status: 0 = A successful route call.

Starting Location: <Ctrl+C>
>>>
```

Add Quit Functionality

1. Save your script as **08_json-parse4.py**.
2. Add an if statement after the address variable to check if the user enters **q** or **quit**.

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

Activity - Test Quit Functionality

1. Run your **08_json-parse4.py** script and verify it works.
2. Troubleshoot your code, if necessary.
3. You should get a similar output to what is shown here.

```
========== RESTART: /home/user/08_parse-json4.py ==========
Starting Location: q
>>>
========== RESTART: /home/user/08_parse-json4.py ==========
Starting Location: quit
>>>
========== RESTART: /home/user/08_parse-json4.py ==========
Starting Location: Washington
Destination: q
>>>
========== RESTART: /home/user/08_parse-json4.py ==========
Starting Location: Washington
Destination: quit
>>>
```

Displaying Trip Data

The final application prints the values for the **distance**, **formattedTime**, and **fuelUsed** keys from the route dictionary.

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

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

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: [
```

Parse and Display Trip Data

1. Save your script as **08_json-parse5.py**.
2. 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.
3. 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("Directions from " + (orig) + " to " + (dest))
    print("Trip Duration:   " + str(json_data["route"]["formattedTime"]))
    print("Miles:           " + str(json_data["route"]["distance"]))
    print("Fuel(Gal):       " + str(json_data["route"]["fuelUsed"]))
    print("==============================================")
```

Activity - Test Trip Data Display

1. Run your **08_json-parse5.py** script and verify it works.
2. Troubleshoot your code, if necessary.
3. You should get a similar output to what is shown here.

```
========== RESTART: /home/user/08_parse-json5.py ==========
Starting Location: Washington
Destination: Baltimore
URL:
https://www.mapquestapi.com/directions/v2/route?to=Baltimor
e&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
>>>
```

Convert Imperial to Metric

Convert the **distance** and **fuelUsed** values to the metric system.

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

Activity - Test Metric Conversion

1. Run your **08_json-parse5.py** script and verify it works.
2. Troubleshoot your code, if necessary.
3. You should get a similar output to what is shown here.

```
========== RESTART: /home/user/08_parse-json5.py ==========
Starting Location: Washington
Destination: Baltimore
URL:
https://www.mapquestapi.com/directions/v2/route?to=Baltimor
e&key=Your_api_key&from=Washington
API Status: 0 = A successful route call.

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

Format to 2 Decimal Places

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)))
```

Activity - Test Formatting Decimal Places

1. Run your **08_json-parse5.py** script and verify it works.
2. Troubleshoot your code, if necessary.
3. You should get a similar output to what is shown here.

```
========== RESTART: /home/user/08_parse-json5.py ==========
Starting Location: Washington
Destination: Baltimore
URL:
https://www.mapquestapi.com/directions/v2/route?key=your_ap
i_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
>>>
```

Inspect the Directions JSON Data

1. Locate the **legs** list inside the **route** dictionary.
2. The legs list includes one big dictionary with most of the JSON data.
3. Find the **maneuvers** list and collapse each of the seven dictionaries inside.


```
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: {
```


Inspect the Maneuvers JSON Data

1. Expand the first dictionary in the maneuvers list. Each dictionary contains a **narrative** key with a value, such as “Start out going north...”
2. You will parse the JSON data to extract the values for narrative and distance keys to display inside your application.

Iterate Through the Directions Data

1. Save your script as **08_json-parse6.py**.
2. Add an if statement below the double line print statement to check the **json_status**.
3. The for loop iterates through each **maneuvers** list, prints the narrative, and prints the distance in kilometers formatting for 2 decimal places.
4. Add a print statement that will display a double line at the end of the list of maneuvers.

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

Activity - Test Iteration

1. Run your **08_json-parse6.py** script and verify it works.
2. Troubleshoot your code, if necessary.
3. You should get a similar output to what is shown here.

```
========= RESTART: /home/user/08_parse-json6.py ==========
Starting Location: Washington
Destination: Baltimore
URL:
https://www.mapquestapi.com/directions/v2/route?key=Your_a
pi_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
>>>
```


Check for Invalid User Input

1. Save your script as **08_json-parse7.py**.
2. The final step is to finish the if loop to respond to the user when the **json_status** does not equal 0.
3. The most common error will most likely be for invalid location entries.
4. Create a **elif** function that checks for **json_status = 402** and display a message, as shown here.

```
if json_status == 0:
    .
    .
    .
elif json_status == 402:
    print("***********************************************")
    print("For Staus Code: " + str(json_status) + "; Invalid user inputs for one or both locations.")
    print("***********************************************\n")
    print("===============================================")
```

Check for Other Error Types

Invalid user inputs is only one error type.

You could write code to display error messages for other common errors.

For this application, add an **else** statement that ends the if loop and covers all other **json_status** values, as shown here.

```
    elif json_status == 402:
        print("**********************************************")
        print("For Staus Code: " + str(json_status) + "; Invalid user inputs for one or both locations.")
        print("**********************************************\n")
        print("==============================================")
    else:
        print("*********************************************************************")
        print("For Staus Code: " + str(json_status) + ", Refer to:")
        print("https://developer.mapquest.com/documentation/directions-api/status-codes")
        print("*********************************************************************\n")
```

Activity - Test Full Application Functionality

1. Run your **08_json-parse7.py** script and verify it works.
2. Troubleshoot your code, if necessary.
3. For errors, you should get a similar output as shown on the next slide.

```
========= RESTART: /home/user/08_parse-json7.py ==========
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.
****************************************************************

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: [no user input]
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
*********************************************************************
```

CISCO