Práctica de Laboratorio 4b - Integrar una API REST en una aplicación Python

Versión en inglés:

https://itexamanswers.net/4-9-2-lab-integrate-a-rest-api-in-a-python-application-answers.html

Objetivos

Parte 1: Iniciar la máquina virtual de DEVASC.

Parte 2: Demostrar la Aplicación de Direcciones de MapQuest

Parte 3: Obtener una clave API de MapQuest

Parte 4: Crear la aplicación de Dirección Básica de MapQuest

Parte 5: Actualizar la Aplicación de Dirección de MapQuest con más características

Parte 6: Probar la funcionalidad completa de la aplicación

Aspectos básicos/Situación

En este laboratorio, creará una aplicación en Visual Studio Code (VS Code) que recupera datos JSON de la API de Direcciones de MapQuest, analiza los datos y les da formato para la salida al usuario. Utilizará la solicitud GET Route de la API de direcciones de MapQuest. Revise la documentación de GET Route Directions API aquí:

https://developer.mapquest.com/documentation/directions-api/route/get/

Nota: Si el enlace anterior ya no funciona, busque por “Documentación de la API de MapQuest”.

Recursos necesarios

- Una computadora con el sistema operativo de su elección.
- VirtualBox o VMware.
- Máquina virtual (Virtual Machine) DEVASC.

Instrucciones

Parte 1: Inicie la máquina virtual de DEVASC.

Si aún no ha completado el **Laboratorio - Instalar el entorno laboratorio de la máquina virtual**, Hágalo ahora. Si ya ha completado ese laboratorio, ejecute la máquina virtual (Virtual Machine) de DEVASC ahora.

Parte 2: Demostrar la Aplicación de Direcciones de MapQuest

La aplicación solicitará una ubicación inicial y un destino. A continuación, solicitará datos JSON de la API de Direcciones de MapQuest, los analiza y muestra información útil.

```
>>>
Lugar de inicio: Washington
Destino: Baltimore
URL: https://www.mapquestapi.com/directions/v2/route?key
=your_api_key&from=Washington&to=Baltimore
Estado de la API: 0 = Una llamada de ruta exitosa.

Cómo llegar desde Washington a Baltimore
Duración del viaje: 00:49:19
Kilómetros: 61.32
Combustible utilizado (Litros): 6.24
=============================================
Salga hacia el norte por la 6ª ST/US-50 E/US-1 N hacia Pennsylvania Ave/US-1 Alt N.
(1,28 km)
Gire a la derecha en New York Ave/US-50 E. Y continúe por US-50 E (cruce hacia
Maryland). (7.51 km)
Tome la salida Balt-Wash Parkway a la izquierda hacia Baltimore. (0.88 km)
Incorpórese en MD-295 N. (50.38 km)
Gire a la derecha hacia W Pratt St. (0,86 km)
Gire a la izquierda hacia S Calvert ST/MD-2. (0.43 km)
Bienvenido a BALTIMORE, MD. (0.00 km)
=============================================

Ubicación inicial: salir
>>>
```

Nota: Para ver el JSON para la salida anterior, puede copiar la URL en una pestaña del navegador. Sin embargo, deberá reemplazar **your_api_key por** la clave API de MapQuest que obtenga en la Parte 3.

Nota del instructor: A continuación, se muestra el script de respuesta para la aplicación Direcciones de MapQuest (MapQuest Directions). Es posible que desee mostrar el script a los estudiantes y mostrar su funcionamiento. Sin embargo, le recomendamos que no dé a los estudiantes el guion. Lo construirán durante el curso de este laboratorio.

```
#Replace "your_api_key" con su clave API de MapQuest

import urllib.parse
solicitudes de importación

main_api = "https://www.mapquestapi.com/directions/v2/route?"
clave = "tu_api_key"


while True:
    orig = input ("Ubicación inicial: ")
    if orig == "quit" u orig == "q":
        break
    dest = entrada ("Destino: ")
    if dest == "quit" o dest == "q":
        break
    url = main_api + urllib.parse.urlencode ({"key" :key, "from" :orig, "to" :dest})
    json_data = requests.get (url) .json ()
```

```
    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) + "= Una llamada de ruta exitosa.\
n")
        print("=============================================")
        print("Directions from " + (orig) + " to " + (dest))
        print("Trip Duration: " + (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("=============================================")
        para cada uno en json_data ["ruta"] ["piernas"] [0] ["maniobras"]:
            print (((each ["narrative"]) + "(" + str ("{:.2f}" .format (((each
["distancia"]) *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")
    elif json_status == 611:
        print("*********************************************")
        print("Status Code: " + str(json_status) + "; Missing an entry for one or both
locations.")
        print ("*********************************************\ n")
    else:

print("************************************************************************")
        print("For Staus Code: " + str(json_status) + "; Refer to:")
        print (" https://developer.mapquest.com/documentation/directions-api/status-
codes ")
        print
("************************************************************************\ n")
```

Parte 3: Obtener una clave de API de MapQuest

Antes de crear la aplicación, debe completar los siguientes pasos para obtener una clave de API de MapQuest.

a. Vaya a: https://developer.mapquest.com/.

b. Haga clic en **Registrarse** arriba de la página

c. Rellene el formulario para crear una nueva cuenta. Para **compañía**, ingrese **Estudiante de Cisco Networking Academy**.

d. Después de hacer clic en **Registrarme**, se le redirigirá a la página **Gestionar claves**. Si se le redirige a otro lugar, haga clic en **Manage Keys** en la lista de opciones de la izquierda.

e. Haga clic en **Approve All Keys**.

f. Expanda **My Application**.

g. Copie su **clave de consumidor** en un archivo de texto para su uso futuro. Esta será la clave que usted usará para el resto de este laboratorio.

Nota: MapQuest puede cambiar el proceso para obtener una clave. Si los pasos anteriores ya no son válidos, busque en Internet "pasos para generar el clave api de mapquest".

Parte 4: Crear la aplicación de dirección básica de MapQuest

En esta parte, creará una secuencia de comandos Python para enviar una solicitud de URL a la API de direcciones de MapQuest. A continuación, va a probar su llamada a la API. A lo largo del resto de este laboratorio, creará el script en partes, guardando el archivo con un nombre nuevo cada vez. Esto ayudará con el aprendizaje de las piezas de la aplicación, así como le proporcionará una serie de scripts a los que puede volver si tiene algún problema en la versión actual de su aplicación.

Paso 1: Cree un nuevo archivo en Visual Studio Code (VS Code).

Puede usar cualquier herramienta que desee introducir en los comandos Python y ejecutar el código Python. Sin embargo, este laboratorio demostrará la creación de la aplicación en VS Code.

a. Abrir **VS Code.** Hay un acceso directo en el **escritorio**, para su comodidad.

b. Seleccione Archivo > Abrir carpeta...

c. Vaya al directorio **~/labs/devnet-src/mapquest** y haga clic en **Aceptar**. Este directorio está actualmente vacío y es donde almacenará cada iteración de su aplicación.

d. Seleccionar **File** > **New File**.

e. Seleccionar **File Guardar como...,** nombrar el archivo **mapquest_parse-json_1.py** y haga clic en **Save**.

Paso 2: Importación de módulos para la aplicación.

Para comenzar su script para analizar datos JSON, necesitará importar dos módulos de la biblioteca de Python: **requests** y **urllib.parse**. El módulo de **peticiones** proporciona funciones para recuperar datos JSON de una URL. El módulo **urllib.parse** proporciona una variedad de funciones que le permitirán analizar y manipular los datos JSON que recibe de una solicitud a una URL.

a. Agregue las siguientes instrucciones de importación en la parte superior del script.

```
import urllib.parse
solicitudes de importación
```

b. Seleccione **Terminal > New Terminal** para abrir una Terminal dentro del VS Code.

c. Guardar y ejectuar el script. No debería recibir errores. Debe guardar y ejecutar los scripts a menudo para probar la funcionalidad del código.

```
devasc @labvm: ~/labs/devnet-src/mapquest$ python3 mapquest_parse-json_1.py
devasc @labvm: ~/labs/devnet-src/mapquest$
```

Paso 3: Crea la URL de la solicitud a la API de direcciones de MapQuest.

El primer paso para crear su solicitud de API es construir la URL que su aplicación utilizará para realizar la llamada. Inicialmente, la URL será la combinación de las siguientes variables:

- **main_api** - la URL principal a la que está accediendo
- **orig** - el parámetro para especificar su punto de origen
- **dest** - el parámetro para especificar su destino
- **key** : la clave API de MapQuest que ha recuperado del sitio web del desarrollador

a. Cree variables para generar la URL que se enviará en la solicitud. En el código siguiente, reemplace **your_api_key por la clave** de consumidor que copió de su cuenta de desarrollador de MapQuest.

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

```
orig = "Washington, D.C."
dest = "Baltimore, Md"
clave = "tu_api_key"
```

b. Combine las cuatro variables **main_api**, **orig**, **dest**y **key** para dar formato a la URL solicitada. Utilice el método **urlencode** para formatear correctamente el valor de la dirección. Esta función construye la parte de parámetros de la URL y convierte posibles caracteres especiales en el valor de la dirección en caracteres aceptables (por ejemplo, espacio en "+" y una coma en "%2C").

```
url = main_api + urllib.parse.urlencode ({"key" :key, "from" :orig, "to" :dest})
```

c. Crear una variable para contener la respuesta de la URL solicitada e imprimir los datos JSON devueltos. La variable **json_data** contiene una representación del diccionario de Python de la respuesta **json** del método **get** del módulo de **peticiones**. El **requests.get** realizará la llamada API a la API de MapQuest. La declaración de **print** se utilizará temporalmente para comprobar los datos devueltos. Reemplazará esta declaración de impresión con opciones de visualización más sofisticadas más adelante en el laboratorio.

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

d. Su código final deberá verse así, pero con un valor diferente para la clave.

```
import urllib.parse
solicitudes de importación

main_api = "https://www.mapquestapi.com/directions/v2/route?"
orig = "Washington, D.C."
dest = "Baltimore, Md"
key = "FzadaFoy22VieeeEMZCBFFXL5VJSXIPPZ" #Replace con la clave de MapQuest

url = main_api + urllib.parse.urlencode ({"key" :key, "from" :orig, "to" :dest})

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

Paso 4: Pruebe la solicitud de URL.

a. Guarde y ejecute su script **mapquest_parse-json_1.py** y verifique que funcione.

b. Solucione el problema de su código, si es necesario. Aunque su salida puede ser ligeramente diferente, debería obtener una respuesta JSON similar a la siguiente. Observe que la salida es un diccionario con dos pares clave/valor. El valor de la clave **route** es otro diccionario que incluye diccionarios y listas adicionales. La clave **información** incluye el par de clave/valor de **código de estado que usará** más adelante en el laboratorio.

```
devasc @labvm: ~/labs/devnet-src/mapquest$ python3 mapquest_parse-json_1.py
{'route': {'HastLRoad': False, 'hasBridge': True, 'boundingBox': {'lr': {'lng'
□76.612137, 'lat': 38.892063}, 'ul': {'lng' □77.019913, 'lat': 39.290443}},
'distancia': 38.089, 'TimedRestriction': False, 'HaStunnel': False, 'hasHighway':
True, 'ComputedWayPoints': [], 'RouteError': {'ErrorCode': -400, 'message': "},
'formattedTime': '00:49:29', 'sessionId': '5eadfc17-00ee-5f21-02b4-1a24-0647e6e69816',
'hasAccessRestriction': False, 'RealTime': 2915, 'hasSeasonalClosure': False, ':
False,' HasCountryCross ': False,': Falso, ': Falso Used': 1.65, 'legs':
[{'HastLRoad': False, 'hasBridge': True, 'destNarrative': 'Proceder a BALTIMORE,
MD.',' distance ': 38.089,' hasTimedRestriction ': False,' HaStunnel ': False,'
hasHighway ': True,' index ': 0,' FormattedTime ':' 00:49:29 ',' originDeX '□1,'
hasAccessRestriction ': False,' hasSeasonalClosure ': False': False, ':':
'HasCountryCross': False ': Falso,': Falso, ': Falso,' HasCountryCross ': False':
False [[]], 'destinDeX': 3, 'tiempo': 2969, 'hasUnpavimentado': False,
```

```
'Orignarrative': ", 'maniobras': [{'distancia': 0.792, 'calles': ['6th St', 'US-50 E',
'US-1 N'], 'narrativa': 'Comienza hacia el norte por 6ª ST/US-50 E/US-1 N hacia
Pennsylvania Ave/US-1 Alt N. ',' TurnType ': 0,': 0, 'StartPoint': {'lng' □77.019913,
'lat': 38.892063}, 'index': 0, 'formattedTime': '00:02:06', 'DirectionName': 'Norte',
'manieuverNotes': [], 'linKids': [], 'signos': [{'extratText': ", 'texto': '50',
'tipo': 2, 'url': 'http://icons.mqcdn.com/icons/rs2.png?n=50&d=EAST', 'dirección': 8},
{'extratext': ", 'texto': '1', 'tipo': ':' url ':': ':': ':': '':
'http://icons.mqcdn.com/icons/rs2.png?n=1&d=NORTH',
""< >>>>>
      output omitted
""< >>>>>
'GeocodeQuality': 'CITY', 'AdminArea1Type': 'País', 'AdminArea3Type': 'Estado',
'LatLng': {'lng' □76.61233, 'lat': 39.29044}}], 'tiempo': 2969, 'hasUnpaved': False,
'LocationSequence': [0, 1], 'hasFerry': False}, 'info': {'statuscode': 0, 'copyright':
{'ImageAltText': '© 2019 MapQuest, Inc. ',' ImageURL ':'
http://api.mqcdn.com/res/mqlogo.gif ',' text ': '© 2019 MapQuest, Inc.'}, 'messages':
[]}}
devasc@labvm:~/labs/devnet-src/mapquest$
```

c. Cambie las variables **orig** y **dest**. Vuelva a ejecutar el script para obtener resultados diferentes. Para garantizar los resultados que desea, lo mejor es incluir tanto la ciudad como el estado para las ciudades de los Estados Unidos. Al referirse a ciudades de otros países, normalmente puede usar el nombre en inglés de la ciudad y el país o el nombre nativo. Por ejemplo:

```
orig = "Roma, Italia"
dest = "Frascati, Italia"
```

o

```
orig = "Roma, Italia"
dest = "Frascati, Italia"
```

Paso 5: Imprima la URL y compruebe el estado de la solicitud JSON.

Ahora que sabe que la solicitud JSON está funcionando, puede agregar algo más de funcionalidad a la aplicación.

a. Guarde su script como **mapquest_parse-json_2.py**.

b. Elimine la instrucción **print (json_data)** ya que no necesita probar que la solicitud tiene el formato correcto.

c. Agregue las declaraciones a continuación, que hará lo siguiente:

- Imprima la URL construida para que el usuario pueda ver la solicitud exacta realizada por la aplicación.
- Analizar los datos JSON para obtener el valor de **código de estado**.
- Inicie un bucle **if** que comprueba si hay una llamada exitosa, que se indica con un valor devuelto de 0. Agregue una instrucción print para mostrar el valor de **código de estado** y su significado. El **\ n** agrega una línea en blanco debajo de la salida.

Más adelante en este laboratorio, agregará instrucciones **elif** y **else** para diferentes valores de **código de estado**.

```
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")
```

Paso 6: Comandos de impresión de estado e URL de prueba.

a. El ejemplo aquí utiliza los siguientes parámetros.

```
orig = "Washington, D.C."
dest = "Baltimore, Md"
```

b. Guarde y ejecute su script **mapquest_parse-json_2.py** y verifique que funcione. Solucione el problema de su código, si fuera necesario. Usted debería de obtener un resultado similar al siguiente. Observe que su clave está incrustada en la solicitud de URL.

```
devasc@labvm:~/labs/devnet-src/mapquest$ python3 mapquest_parse-json_2.py
URL: https://www.mapquestapi.com/directions/v2/route?key
=FZADAFoy22VieeeEMZCBFXL5VJSXIPPZ&from=Washington%2C+D.C.&to=Baltimore%2C+MD
Estado de la API: 0 = Una llamada de ruta exitosa.

devasc@labvm:~/labs/devnet-src/mapquest$
```

Paso 7: Añadir entrada de usuario para la ubicación inicial y el destino.

Ha utilizado valores estáticos para las variables de ubicación. Sin embargo, la aplicación requiere que el usuario los introduzca. Complete los siguientes pasos para actualizar su aplicación:

a. Guarde su script como **mapquest_parse-json_3.py**.

b. Elimine las variables **orig** y **dest** actuales.

c. Vuelva a escribir el **orig** y **dest** para estar dentro de un bucle while en el que solicita la entrada del usuario para la ubicación inicial y el destino. El bucle while permite al usuario continuar realizando solicitudes para diferentes direcciones. Agregue el siguiente código, resaltado a continuación, después del parámetro clave. Asegúrese de que todo el código restante esté sangrado correctamente dentro del bucle while.

```
import urllib.parse
solicitudes de importación

main_api = "https://www.mapquestapi.com/directions/v2/route?"
key = "fZadaFOY22VIEEemZcBFfxl5vjSXIPpZ"

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) + "= Una llamada de ruta exitosa.\ n")
```

Paso 8: Probar la funcionalidad de entrada del usuario.

Ejecute su script **mapquest_parse-json_3.py** y verifique que funcione. Solucione el problema de su código, si es necesario. Debería obtener una salida similar a la que se muestra a continuación. Para finalizar el programa, escriba **Ctrl+C**. Obtendrá un error de **KeyboardInterrupt** como se muestra en la salida a continuación. Para detener la aplicación con más gracia en ella, agregará la funcionalidad de salir en el siguiente paso.

```
devasc@labvm:~/labs/devnet-src/mapquest$ python3 mapquest_parse-json_3.py
Lugar de inicio: Washington, D.C.
```

```
Destino: Baltimore, Md
URL: https://www.mapquestapi.com/directions/v2/route?key
=FZADAFoy22VieeeEMZCBFXL5VJSXIPPZ&from=Washington%2C+D.C.&to=Baltimore%2C+MD
Estado de la API: 0 = Una llamada de ruta exitosa.

Ubicación inicial: ^CTraceback (última llamada):
  Archivo(File) "mapquest_parse-json_3.py", línea 9, en <module>
    orig = input("Starting Location: ")
KeyboardInterrupt

devasc@labvm:~/labs/devnet-src/mapquest$
```

Paso 9: Agregue la funcionalidad para salir a la aplicación.

En lugar de forzar a la aplicación a salir con una interrupción del teclado, agregará la capacidad para que el usuario escriba **q** o **quit** como palabras clave para salir de la aplicación. Complete los siguientes pasos para actualizar su aplicación:

a. Guarde su script como **mapquest_parse-json_4.py**.

b. Agregue una instrucción if después de cada variable de ubicación para comprobar si el usuario ingresa **q** o **sale**, como se muestra a continuación.

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

Paso 10: Pruebe la funcionalidad de salir.

Ejecute el script **mapquest_parse-json_4.py** cuatro veces para probar cada variable de ubicación. Verifique que tanto quit y q terminen la aplicación.Solucione el problema de su código, si fuera necesario. Usted debería de obtener un resultado similar al siguiente.

```
devasc@labvm:~/labs/devnet-src/mapquest$ python3 mapquest_parse-json_4.py
Ubicación de inicio: q
devasc@labvm:~/labs/devnet-src/mapquest$ python3 mapquest_parse-json_4.py
Locación inicial: quit
devasc@labvm:~/labs/devnet-src/mapquest$ python3 mapquest_parse-json_4.py
Lugar de inicio: Washington, D.C.
Destino: q
devasc@labvm:~/labs/devnet-src/mapquest$ python3 mapquest_parse-json_4.py
Localización inicial: Washington, D.C.
Destino: salir
devasc@labvm:~/labs/devnet-src/mapquest$
```

Paso 11: Mostrar los datos JSON en JSONView.

El navegador Chromium de la máquina virtual DEVASC incluye la extensión JsonView. Puede usar esto para ver un objeto JSON en un formato legible, coloreado y plegable.

a. Ejecute su **mapquest_parse-json_4.py** de nuevo y copie el código devuelto para la URL. No utilice el siguiente código. Su resultado incluirá su clave API.

```
devasc@labvm:~/labs/devnet-src/mapquest$ python3 mapquest_parse-json_4.py
Lugar de inicio: Washington, D.C.
Destino: Baltimore, Md
URL:
https://www.mapquestapi.com/directions/v2/route?key=fZadaFOY22VIEEemZcBFfxl5vjSXIPpZ&from=Washington%2C+D.C.&to=Baltimore%2C+Md
Estado de la API: 0 = Una llamada de ruta exitosa.

Ubicación inicial: salir
devasc@labvm:~/labs/devnet-src/mapquest$
```

b. Pegue la URL en el campo de direcciones del navegador Chromium

c. Contraer los datos de JsonView seleccionando el guión "**-**" antes de la **ruta**, verá que hay dos diccionarios raíz: **ruta** e **información**.

```
{
 - ruta:{
      HastollRoad: false,
      hasBridge: true,
 <output omitted>
```

Verá que hay dos diccionarios root: **rout** e **info**. Observe que la **información** tiene la clave/valor de **código de estado emparejado** utilizada en su código.

```
{
 + ruta: {},
 - info: {
       código de estado: 0,
     - derechos de autor: {
           ImageAltText: "© 2019 MapQuest, Inc.",
           ImageURL: "http://api.mqcdn.com/res/mqlogo.gif ",
           text: "© 2019 MapQuest, Inc."
       },
       mensajes: [ ]
   }
}
```

d. Expanda el diccionario de **ruta** (haga clic en el signo más "**+**" antes de la **route**) e investigue los datos enriquecidos. Hay valores que indican si la ruta tiene carreteras de peaje, puentes, túneles, autopistas, cierres o cruces hacia otros países. También deberías ver los valores de distancia, el tiempo total que tardará el viaje y el consumo de combustible. Para analizar y mostrar estos datos en su aplicación, debe especificar el diccionario de **route** y, a continuación, seleccionar el par key/value que desea imprimir. Hará un análisis del diccionario de ruta en la siguiente parte del laboratorio.

Parte 5: Actualice la aplicación Direcciones de MapQuest (MapQuest Directions) con más características

En esta parte, agregará características adicionales a la aplicación MapQuest Directions para proporcionar más información al usuario. Incluirá información de resumen del viaje y luego una lista de las direcciones analizadas

desde el diccionario. Como paso final, agregará algunas comprobaciones básicas de errores para validar la entrada del usuario.

Paso 1: Muestra información de resumen de viaje para incluir la duración, la distancia y el combustible utilizado.

a. Guarde su script como **mapquest_parse-json_5.py**.

b. Debajo del comando API status **print**, agregue varias instrucciones de **print** que muestren las ubicaciones de origen y destino, así como las claves **FormattedTime**, **distance**y **fuelUsed**.

Las instrucciones adicionales también incluyen instrucciones de impresión que mostrarán una línea doble antes de la siguiente solicitud de una ubicación inicial. Asegúrese de que estas instrucciones están incrustadas en la función while True.

```
    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 ("Combustible usado (Gal):" + str (json_data ["route"] ["FuelUsed"]))
        print("=============================================")
```

c. Guarde y ejecute **mapquest_parse-json_5.py** para ver la siguiente salida.

```
devasc@labvm:~/labs/devnet-src/mapquest$ python3 mapquest_parse-json_5.py
Lugar de inicio: Washington, D.C.
Destino: Baltimore, Md
URL: https://www.mapquestapi.com/directions/v2/route?key
=FZADAFoy22VieeeEMZCBFXL5VJSXIPPZ&from=Washington%2C+D.C.&to=Baltimore%2C+MD
Estado de la API: 0 = Una llamada de ruta exitosa.

=============================================
Cómo llegar desde Washington, D.C. a Baltimore, Md
Duración del viaje: 00:49:29
Millas: 38.089
Combustible utilizado (Gal): 1.65
=============================================
Localización Inicial: q
devasc@labvm:~/labs/devnet-src/mapquest$
```

d. Por defecto, MapQuest utiliza el sistema imperial y no hay un parámetro de solicitud para cambiar los datos al sistema métrico. Por lo tanto, probablemente debería convertir su aplicación para mostrar los valores de métrica, como se muestra a continuación.

```
    print("Kilometers: " + str((json_data["route"]["distance"])*1.61))
    print ("Combustible usado (Ltr):" + str ((json_data ["route"] ["FuelUsed"])*3.78))
```

e. Ejecute el script **mapquest_parse-json_5.py** modificado para ver el siguiente resultado:

```
devasc@labvm:~/labs/devnet-src/mapquest$ python3 mapquest_parse-json_5.py
Lugar de inicio: Washington, D.C.
Destino: Baltimore, Md
URL: https://www.mapquestapi.com/directions/v2/route?key
=FZADAFoy22VieeeEMZCBFXL5VJSXIPPZ&from=Washington%2C+D.C.&to=Baltimore%2C+MD
Estado de la API: 0 = Una llamada de ruta exitosa.
```

```
=============================================
Direcciones desde Washington, D.C. a Baltimore, Md
Duración del viaje: 00:49:29
Kilómetros: 61.32329
Combustible utilizado (Ltr): 6.2369999999999
=============================================
Localización Inicial: q
devasc@labvm:~/labs/devnet-src/mapquest$
```

f. Las posiciones decimales adicionales para Kilómetros y Combustible Usado no son útiles. Utilice el argumento **de formato "{: .2f}"**. Para formatear los valores flotantes a 2 lugares decimales antes de convertirlos en valores de cadena, como se muestra a continuación. Cada declaración debe estar en una línea.

```
    print ("Kilómetros:" + str ("{:.2f}" .format((json_data ["route"] ["distance"])
*1.61)))
    print ("Combustible usado (Ltr):" + str ("{:.2f}" .format((json_data ["route"]
["FuelUsed"]) *3.78)))
```

Paso 2: Pruebe la funcionalidad de análisis y formato.

Guarde y ejecute el script **mapquest_parse-json_5.py** para verificar que funciona. Solucione el problema de su código, si es necesario. Asegúrese de tener todos los paréntesis de apertura y cierre adecuados. Usted debería de obtener un resultado similar al siguiente.

```
devasc@labvm:~/labs/devnet-src/mapquest$ python3 mapquest_parse-json_5.py
Lugar de inicio: Washington, D.C.
Destino: Baltimore, Md
URL: https://www.mapquestapi.com/directions/v2/route?key
=FZADAFoy22VieeeEMZCBFXL5VJSXIPPZ&from=Washington%2C+D.C.&to=Baltimore%2C+MD
Estado de la API: 0 = Una llamada de ruta exitosa.

=============================================
Direcciones desde Washington, D.C. a Baltimore, Md
Duración del viaje: 00:49:29
Kilómetros: 61.32
Combustible utilizado (Ltr): 6.24
=============================================
Localización Inicial: q
devasc@labvm:~/labs/devnet-src/mapquest$
```

Paso 3: Inspeccione la lista de maniobras en los datos JSON.

a. Ahora está listo para mostrar las indicaciones paso a paso desde la ubicación inicial hasta el destino. Vuelva al navegador de Chromium donde vio anteriormente la salida en JsonView. Si usted cerró el navegador, copie la URL de la última vez que ejecutó el programa y péguela en la barra de direcciones del navegador.

b. Dentro del diccionario de **ruta**, localice la lista de **tramos**. La lista de **reglas** incluye un diccionario grande con la mayoría de los datos JSON. Encuentre la lista de **maniobras** y colapse cada uno de los siete diccionarios dentro, como se muestra a continuación (haga clic en el **signo menos**"**-** "para cambiarlo a un **signo "+" más**). Si está utilizando diferentes ubicaciones, probablemente tendrá un número diferente de diccionarios de maniobra.

```
- instrucciones: [
    - {
 HastollRoad: false,
 hasBridge: true,
 " Proceder a BALTIMORE, MD.",
 distancia: 38.089,
 hasTimedRestriction: false,
 HaStunnel: false,
 HasHighway: true,
 índice: 0,
 FormatedTime: "00:49:29",
 OriginDeX: -1,
 hasAccessRestriction: false,
 hasSeasonalClosure: false,
 HasCountryCross: false,
       - Estrategia RoadGradeStrategy: [
            [ ]
         ],
 DestinDeX: 3,
 tiempo: 2969,
 hasUnpavimentado: false,
 Orignarrative: "",
      - maneuvers: [
          + {…},
          + {…},
          + {…},
          + {…},
          + {…},
          + {…},
          + {…}
         ],
          HasFerry: false
     }
        ],
      - options: {
```

c. Expanda el primer diccionario de la lista de **maniobras**. Cada diccionario contiene una clave **narrativa** con un valor, como "Empezar hacia el norte...", como se muestra a continuación. Debe analizar los datos JSON para extraer el valor para que la clave **narrativa** se muestre dentro de su aplicación.

```
- instrucciones: [
    - {
 HastollRoad: false,
 hasBridge: true,
 " Proceder a BALTIMORE, MD.",
 distance: 38.089,
 hasTimedRestriction: false,
 HaStunnel: false,
 HasHighway: true,
 index: 0,
 FormatedTime: "00:49:29",
```

```
 OriginDeX: -1,
 hasAccessRestriction: false,
 hasSeasonalClosure: false,
 HasCountryCross: false,
     - roadGradeStrategy: [
           [ ]
       ],
 DestinDeX: 3,
 time: 2969,
 hasUnpavimentado: false,
 Orignarrative: "",
    - maneuvers: [
        - {
 distance: 0.792,
           - streets: [
                 "6° St",
                 "US-50 E",
                 "US-1 N"
             ],
  narrativa: "Comienza hacia el norte por la 6ª ST/US-50E/US-1 N hacia Pennsylvania
Ave/US-1 Alt N.",
 TurnType: 0,
           - StartPoint: {
 lng: -77.019913,
                lat: 38.892063
             },
 índex: 0,
 FormatedTime: "00:02:06",
 DirectionName: "Norte",
             Notas de maniobra: [],
             LinKids: [],
           - signs: [
               - {
 ExtraText: "",
 ext: "50",
 type: 2,
<output omitted>
```

Nota: Se agregó el ajuste de Word para el valor de la narrativa a efectos de visualización.

Paso 4: Agregue un bucle for para iterar a través de los datos JSON maniobras.

Complete los siguientes pasos para actualizar la aplicación y mostrar el valor de la clave **narrativa**. Lo hará creando un bucle for para iterar a través de la lista de maniobras, mostrando el valor narrativo de cada maniobra desde la ubicación inicial hasta el destino.

a. Guarde su script como **mapquest_parse-json_6.py**.

b. Agregue un bucle "for", resaltado a continuación, después de la segunda declaración de impresión de doble línea. El bucle for itera a través de cada lista de **maniobras** y hace lo siguiente:

   1) Imprima el valor **narrativo**.

   2) Convierta las millas en kilómetros con ***1.61**.

3) Formatea el valor del kilómetro para imprimir sólo dos posiciones decimales con la función **"{:.2f}" .format**.

c. Agregue una instrucción de **impresión** que mostrará una línea doble antes de que la aplicación solicite otra ubicación inicial, como se muestra a continuación.

Nota: La sentencia de impresión de línea doble no está sangrada dentro del bucle for. Por lo tanto, es parte de la instrucción if anterior que comprueba el parámetro statuscode.

```
print ("Combustible usado (Ltr):" + str ("{:.2f}" .format ((json_data ["route"] ["FuelUsed"]) *3.78)))
print("=============================================")
para cada uno en json_data ["ruta"] ["piernas"] [0] ["maniobras"]:
    print (((each ["narrative"]) + "(" + str ("{:.2f}" .format (((each ["distancia"]) *1.61) + "km)"))
print ("=============================================\ n")
```

Paso 5: Actividad: pruebe la iteración de JSON.

Guarde y ejecute el script **mapquest_parse-json_6.py** para verificar que funciona. Solucione el problema de su código, si es necesario Usted debería de tener un resultado similar al siguiente

```
devasc@labvm:~/labs/devnet-src/mapquest$ python3 mapquest_parse-json_6.py
Localización inicial: Washington, D.C.
Destino: Baltimore, Md
URL: https://www.mapquestapi.com/directions/v2/route?key
=FZADAFoy22VieeeEMZCBFXL5VJSXIPPZ&from=Washington%2C+D.C.&to=Baltimore%2C+MD
Estado de la API: 0 = Una llamada de ruta exitosa.

=============================================
Direcciones desde Washington, D.C. a Baltimore, Md
Duración del viaje: 00:49:29
Kilómetros: 61.32
Combustible utilizado (Litosr): 6.24
=============================================
Salga hacia el norte por la 6ª ST/US-50 E/US-1 N hacia Pennsylvania Ave/US-1 Alt N.
(1,28 km)
Gire a la derecha en New York Ave/US-50 E. Y continúe por US-50 E (cruce hacia
Maryland). (7.51 km)
Tome la salida Balt-Wash Parkway a la izquierda hacia Baltimore. (0.88 km)
Incorpórese en MD-295 N. (50.38 km)
Gire a la derecha hacia W Pratt St. (0,86 km)
Gire a la izquierda hacia S Calvert ST/MD-2. (0.43 km)
Bienvenido a BALTIMORE, MD. (0.00 km)
=============================================

Ubicación de inicio: q
devasc@labvm:~/labs/devnet-src/mapquest$
```

Paso 6: Compruebe la entrada de usuario no válida.

Ahora está listo para agregar una característica final a su aplicación para informar de un error cuando el usuario introduce datos no válidos. Recuerde que inició un bucle if para asegurarse de que el **código de estado** devuelto es igual a 0 antes de analizar los datos JSON:

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

```
    if json_status == 0:
        print ("API Status:" + str (json_status) + "= Una llamada de ruta exitosa.\
n")
```

Pero, ¿qué sucede si el **código de estado** no es igual a 0? Por ejemplo, es posible que el usuario introduzca una ubicación no válida o que no especifique una o varias ubicaciones. Si es así, la aplicación muestra la URL y solicita una nueva ubicación de inicio. El usuario no tiene idea de lo que pasó.

a. Para hacer que la aplicación falle sin notificación del usuario, pruebe los siguientes valores en la aplicación. Deberías ver resultados similares.

```
devasc@labvm:~/labs/devnet-src/mapquest$ python3 mapquest_parse-json_6.py
Localización inicial: Washington, D.C.
Destino: Pekín, China
URL:
https://www.mapquestapi.com/directions/v2/route?key=fZadaFOY22VIEEemZcBFfxl5vjSXIPpZ&f
rom=Washington%2C+D.C.&to=Beijing%2C+China
Localización inicial: Washington, D.C.
Destino: Bal
URL:
https://www.mapquestapi.com/directions/v2/route?key=fZadaFOY22VIEEemZcBFfxl5vjSXIPpZ&f
rom=Washington%2C+D.C.&to=Bal
Ubicación de inicio: q
devasc@labvm:~/labs/devnet-src/mapquest$
```

b. Copie una de las URL en una pestaña del navegador Chromium. Observe que la única entrada en el diccionario de **route** es un diccionario **RouteError** con **ErrorCode**2. En el diccionario info, el **código de estado** es **402**. Por lo tanto, su bucle if nunca ejecutó el código para cuando el **código de estado** es **0**.

c. Guarde su script como **mapquest_parse-json_7.py**.

d. Para proporcionar información de error cuando el **código de estado** es igual a **402**, **611** u otro valor, agregue dos instrucciones **elif** y una sentencia **else** a su bucle if. Sus declaraciones **elif** y **else** deben alinearse con la declaración **if** anterior. Después de la última declaración de impresión de doble línea bajo **if json_status == 0**, agregue las siguientes instrucciones **elif** y **else** :

```
    if json_status == 0:
       <statements omitted>
       para cada uno en json_data ["ruta"] ["piernas"] [0] ["maniobras"]:
           print((each["narrative"]) + " (" +
str("{:.2f}".format((each["distance"])*1.61) + " km)"))
        print ("=============================================\ n")
    elif json_status == 402:
        print("********************************************")
        print ("Código de estado:" + str (json_status) + "; Entradas de usuario no válidas
para una o ambas ubicaciones.")
        print ("********************************************\ n")
    elif json_status == 611:
        print("********************************************")
        print ("Código de estado:" + str (json_status) + "; Falta una entrada para una o
ambas ubicaciones.")
        print ("********************************************\ n")
    else:
        print("************************************************************")
        print ("For Staus Code:" + str (json_status) + "; Consulte:")
```

```
        print (" https://developer.mapquest.com/documentation/directions-api/status-codes ")
        print
("********************************************************************\ n")
```

La primera instrucción **elif** se imprime si el valor **de código de estado** es **402** para una ubicación no válida. La segunda instrucción **elif** se imprime si el valor de **código de estado** es **611** porque el usuario no proporcionó una entrada para una o ambas ubicaciones. La instrucción **else** se imprime para todos los demás valores de **código de estado**, como cuando el sitio de MapQuest devuelve un error. La sentencia **else** termina el bucle **if/else** y devuelve la aplicación al bucle while.

Parte 6: Pruebe la funcionalidad completa de la aplicación

Ejecute su script **mapquest_parse-json_7.py** y verifique que funcione. Solucione el problema de su código, si es necesario Pruebe todas las características de la aplicación. Usted debería de obtener un resultado similar al siguiente.

```
devasc @labvm: ~/labs/devnet-src/mapquest$ python3 mapquest_parse-json_7.py
Localización inicial: Washington, D.C.
Destino: Baltimore, Md
URL: https://www.mapquestapi.com/directions/v2/route?key
=FZADAFoy22VieeeEMZCBFXL5VJSXIPPZ&from=Washington%2C+D.C.&to=Baltimore%2C+MD
Estado de la API: 0 = Una llamada de ruta exitosa.

=============================================
Direcciones desde Washington, D.C. a Baltimore, Md
Duración del viaje: 00:49:29
Kilómetros: 61.32
Combustible utilizado (Litosr): 6.24
=============================================
Salga hacia el norte por la 6ª ST/US-50 E/US-1 N hacia Pennsylvania Ave/US-1 Alt N.
(1,28 km)
Gire a la derecha en New York Ave/US-50 E. Y continúe por US-50 E (cruce hacia
Maryland). (7.51 km)
Tome la salida Balt-Wash Parkway a la izquierda hacia Baltimore. (0.88 km)
Incorpórese en MD-295 N. (50.38 km)
Gire a la derecha hacia W Pratt St. (0,86 km)
Gire a la izquierda hacia S Calvert ST/MD-2. (0.43 km)
Bienvenido a BALTIMORE, MD. (0.00 km)
=============================================


Lugar de inicio: Moscú, Rusia
Destino: Pekín, China
URL:
https://www.mapquestapi.com/directions/v2/route?key=fZadaFOY22VIEEemZcBFfxl5vjSXIPpZ&f
rom=Moscow%2C+Russia&to=Beijing%2C+China
Estado de la API: 0 = Una llamada de ruta exitosa.

=============================================
Cómo llegar desde Moscú, Rusia a Pekín, China
Duración del viaje: 84:31:10
Kilómetros: 7826,83
Combustible utilizado (Ltr): 793.20
=============================================
```

```
Comienza hacia el oeste por Кремлёвская Набережная/terraplén del Kremlin. (0.37 km)
Gire ligeramente a la derecha en la rampa. (0.15 km)
Gire ligeramente a la derecha hacia Боровицкая площадь. (0.23 km)
<output omitted>
Gire ligeramente a la izquierda hacia la calle/Qianmen East Street. (0.31 km)
Gira a la izquierda hacia/E. Guangchang Rd. (0.82 km)
/E. Guangchang Rd se convierte en/E. Chang'an Str. (0.19 km)
Bienvenido a BEIJING. (0.00 km)
==========================================

Localización inicial: Washington, D.C.
Destino: Pekín, China
URL:
https://www.mapquestapi.com/directions/v2/route?key=fZadaFOY22VIEEemZcBFfxl5vjSXIPpZ&f
rom=Washington%2C+D.C.&to=Beijing%2C+China
******************************************
Código de estado: 402; Entradas de usuario no válidas para una o ambas ubicaciones.
******************************************

Localización inicial: Washington, D.C.
Destino: Bal
URL:
https://www.mapquestapi.com/directions/v2/route?key=fZadaFOY22VIEEemZcBFfxl5vjSXIPpZ&f
rom=Washington%2C+D.C.&to=Bal
******************************************
Código de estado: 402; Entradas de usuario no válidas para una o ambas ubicaciones.
******************************************

Localización inicial: Washington, D.C.
Destino:
URL:
https://www.mapquestapi.com/directions/v2/route?key=ANUqwkHlgDv1vlsyBPtVrFeX8
uu6agjA&from=Washington%2C+D.C.&to =
******************************************
Código de estado: 611; Falta una entrada para una o ambas ubicaciones.
******************************************

Locación inicial: Salir
devasc@labvm:~/labs/devnet-src/mapquest$
```