Python Object-Oriented Program with Libraries

Unit 3: Web Programming

CHAPTER 5: MAPQUEST API PROJECT

DR. ERIC CHOU

IEEE SENIOR MEMBER

Objectives

- Use of the API on the server side to perform calculation.
- Demonstrate simple geodetic program that use MapQuest API

Backgrounds

LECTURE 1

Background

- We saw in the previous project that our Python programs are capable of connecting to the "outside world" around them – to other programs running on the same machine, or even to other programs running on different machines in faraway places.
- This is a powerful thing for a program to be able to do, because it is no longer limited to taking its input from a user or from a file stored locally; its input is now potentially anything that's accessible via the Internet, making it possible to solve a vast arrya of new problems and process a much broader collection of information.
- Once you have the ability to connect your programs to others, a whole new world opens up. Suddenly, the idea that you should be able to write a program that combines, say Google Search queries, the Internet Movie Database, and your favorite social network to find people who like movies similar to the ones that you loke doesn't seem so far-fetched.

Motivation for Web API

- But we also saw that getting programs to share information is tricky, for two reasons. Firstly, there's a software engineering problem:
  - A protocol has to be designed that both programs can use to have their conversation.
  - Secondly, there's a social problem: If the same person isn't writing both programs, it's necessary for them to agree on the protocol ahead of time, then to implement it.
- This second problem has a potentially catastrophic effect on our ability to make things work – how could you ever convince Google to agree to use your protocol just to communicate with you?

Standardization

- In practice, both of these problems are largely solved by the presence of standards, such as those defined by the **World Wide Web Consortium** and **Internet Engineering Task Force**.
- Standards help by providing detailed communication protocols whose details have already been hammered out, with the intention of handling the most common set of needs that will arise in programs. This eliminates the need to design one's own protocol and allows programs to be combined in arbitrary ways; as long as they support the protocol, they've taken a big step toward being able to interoperate with each other.
- What's more, standard protocols often have standard implementations, so that you won't have to code up the details yourself. For example, python has built-in support for a number of standard internet protocols, including HTTP among others.

HTTP Support

- Since HTTP support is built directly into Python, we can write programs that use these web services without having to handle low-level details of the protocol, though there are some details that you will need to be familiar with if you want to use the provided implementation effectively. We'll discussing some of these details.

Case Study Purpose

- This project gives you the opportunities to explore a small part of the vast sear of possibilities presented by web APIs and web services. You'll likely find that you spend a lot of your time in this project understanding the web API – being able to navigate technical documentation and gradually build an understanding of another system is a vital skill in building real software – and that the amount of code you need isn't nearly what you might expect when you first read the project write-up.
- As always, work incrementally rather than trying to work on the entire project all at once. When you're done, you'll have taken a valuable step toward being able to build Python programs that interact with web services, which opens up your ability to write programs for yourself that are real and useful.

Problem Description

LECTURE 1

Project Statement

- In your work on this project, you'll write a program that is capable of displaying information about a trip from one location to another.
- For example, you might display turn-by-turn directions, an estimate of how long it might take to get from one location to another, and so on. You'll use real-world map data – real cities, real streets – to solve your problem. So, when you're done, your program will be very simple navigation system, not entirely unlike the ones you see on some smartphones or in some cars. If you want to know how to drive from Bren Hall at UCI to Staples Center in Los Angeles, your program will be able to tell you.

MapQuest API

LECTURE 1

The MapQuest Open Data APIs

- MaqQuest is a company that is in the business of providing online services for displaying maps, providing directions, reporting on current traffic conditions, and other related services. While they're a for-profit company, some of their services are provided free for non-commercial use.
- As long as you're not building a product from which you'll be trying to make money, you cn ause MapQuest's free services, with the one additional caveat that you have to follow the rules laid out in their license.

MapQuest API

- For our work on this project, you'll be concerned with two parts of the Open MapQuest API, provided by a company called MapQuest; we'll need both the Open Directions Service and the Open Elevation Service. Both of these are web-based APIs, similar to the one we used before in which we downloaded and displayed information about YouTube Videos. Like YouTube's API, then Open MapQuest API used HTTP, with queries described using a URL, and with responses returned in JSON format.
- Fortunately, all of these technologies are supported in Python's standard library, so most of the details are things that will be handled for us automatically, but as we saw in the YouTube example, there are some things we need to get right, and not all of these details will be the same in the MapQuest example as they were in the YouTube example, so it'll be vital for you to understand why we did the things we did in the example, so you can know whether and where the same techniques apply.

MapQuest API

MapQuest API:

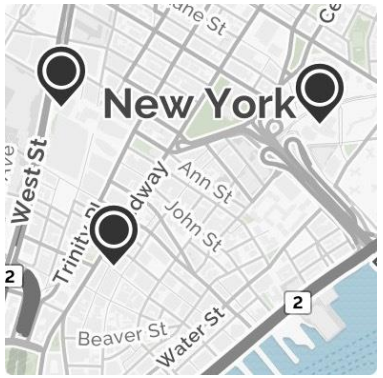
- MapQuest Developer

Two MapQuest API:

- MapQuest Open Directions API Service Documentation
- MapQuest Open Elevation API Service Documentation

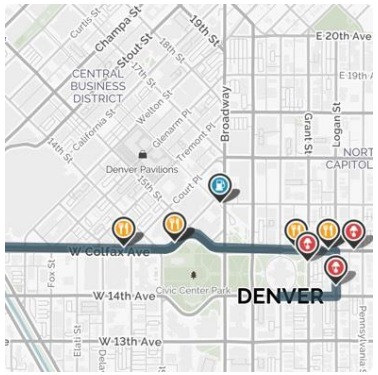
Web Services

Mapping



Geocoding

Directions



Search

Search Ahead

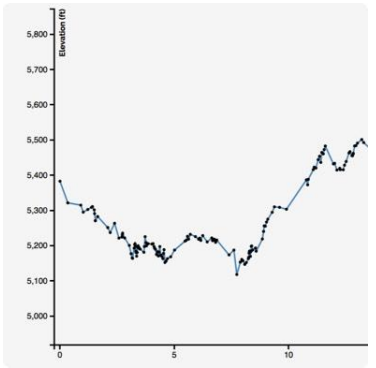
Static Map

Traffic

```
{
    "clientId": "CLIENT_ID",
    "password": "REGISTRY_PASSWORD",
    "tableName": "mqap.TABLENAME",
    "friendlyName": "friendly_name",
    "notes": "notes",
    "columns": [
        {
            "name": "NAME",
            "type": "TYPE",
            "isPublic": false
        }
        ...
    ],
    "isPublic": false
}
```

Data Manager

Nominatim



Elevation

API Key

- Visit the MapQuest Developer site in your browser.
- Part of the way down the page, you will see a button that says **Get You Free API Key**. Click the button.
- A form will be displayed, in which you can choose a username, a password, and so on. Fill the necessary information, and be sure to sue an email address that you have access to; you will need to receive emails from MapQuest along the way.
- Once you've created your account, you'll be logged in and presented with some choices, one of which is **Manage Keys,** click that.

API Key

- You'll then see a button that says **Create a New Key**. Click that, because you're going to be needing an API Key that allows you to use the Open MapQuest APIs.
- When asked, supply an **App Name**. The **Callback URL** is not important for us, since we're not building a web application, so you can specify the URL of the Project 3 Web Page: http://www.ics.ued.edu/~thornthon/ics32/ProjectGuide/Project3/
- Now that your key has been created withing your MapQuest Developer profile, you will be able to obtain your API key. Click the name of your application, which you should now see on the page, which will reveal more information about it. Make a note of both **Consumer Key** and **Consumer Secret** somewhere; you'll need these later.
- Notice, too, that there is a limit on the number of times you can use the API each month – currently 15000 transactions per month – and you can see in this same area of the MapQuest Developer web site how many of these transactions you've used at any given time. The limit should be plenty for our use, but you may nonetheless want to keep an eye on it.

Testing your API Key

- Wait for a little while after your API Key, then it's time to test that it's working. Open your favorite web browser; enter a URL in the following format into the browser's address bar and press Enter, replacing APIKEY with the Consumer Key part of API key that you created in the previous step.

http://open.mapquestapi.com/directions/v2/route?key=**APIKEY**&from-Irvine%2CCA&to=Los+Angeles%2CCA

Testing your API Key


```
{"route":{
            "routeError":{"errorCode":211,
                          "message":""
                         }
          },
  "info":{
          "statuscode":611,
          "copyright":{
                       "imageAltText":"© 2021 MapQuest, Inc.",
                       "imageUrl":"http://api.mqcdn.com/res/mqlogo.gif",
                       "text":"© 2021 MapQuest, Inc."
                      },
          "messages":["At least two locations must be provided."]
         }
}
```

Case Study

LECTURE 1

The Program

- Your program will describe a trip taken between a sequence of locations, the goal being to travel from the first location to the second, then from the second location to the third, and so on, until reading the last location. Based on the user's input, it will show different information about the trip, such as turn-by-turn directions, distances and times, etc.

Input

- Your program will take input in the following format. It should not prompt the user in any way; it should simply read whatever input is typed into the console, and you should assume that your user knows the precise input format.
  - An integer whose value is at least 2, alone on a line, that specifies how many locations the trip consists of.
  - If here are n locations, the next n lines of input will each describe on location. Each location can be a city such as Irvine, CA, and address such as **4545 Campus Dr., Irvine, CA**, or anything that the Open MapQuest API will accept as a location.
  - A Positive integer alone on a line, that specifies how many outputs will need to be generated.

Input

- If there are m outputs, the next m lines will each describe one output. Each output can be one of the following:
  - **STEPS** for step-by-step directions, meaning a brief description of each maneuver you would have to make to drive from one location to another
  - **TOTALDISTANCE** for the total distance traveled if completing the entire trip
  - **TOTALTIME** for the total estimated time to complete the entire trip
  - **LATLONG** for the latitude and longitue of each of the locations specified in the input
  - **ELEVATION** for the elevation, in feet, of each of the locations specified in the input.

The output

When a route was found by MapQuest

- After reading the input and processing it – downloading information from MapQuest API, etc. – your program will generate the specified outputs in the forms described below.
- Each output must be preceded by a blank line , to set each one off from the others. The outputs must be written in the order that they were specified in the input.

The output

When a route was found by MapQuest

- **STEPS** Output: The output should begin with the word **DIRECTIONS** alone on a line, followed by one line of output for each maneuver that needs to be made along the path from the first location to the last.
- **TOTALDISTANCE** Output: The output should be the words **TOTAL DISTANCE**, followed by a colon and a space, followed by the total distance of the entire trip.
- **TOTALTIME** output: The output should be the words TOTAL TIME, followed by a colon and a space, followed by the total time that would be required to take the entire trip.

The output

When a route was found by MapQuest

- **LATLONG** output: The output should be the word LATLONGS alone on a line, followed by a latitude and longitude, one of each per line, for each of the locations specified in the input, in the order specified in the input. The latitude should come first, followed by a space, followed by the longitude.
  - The latitude's format is a number of degrees followed by either **N** for the north or **S** for south.
  - The longitude's format is a number of degrees followed by either **W** for the west or **E** for east.
- **ELEVATION** output: The output should be the word ELEVATIONS alone on a line, followed by an integer number of feet of elevation, one per line, for each of the locations specified in the input, in the order specified in the input. If MapQuest, reports the elevation with a decimal part, round to the nearest integer.

The output

When a route was not found by MapQuest

- When no route was found by MapQuest – e.g., if you look for driving directions from Irvine, CA to Lisbon, Portugal, you won't find any – the program should simply output a blank line, followed by NO ROUTE FOUND alone on a line. This also include the scenario where one or more of the locations was not valid

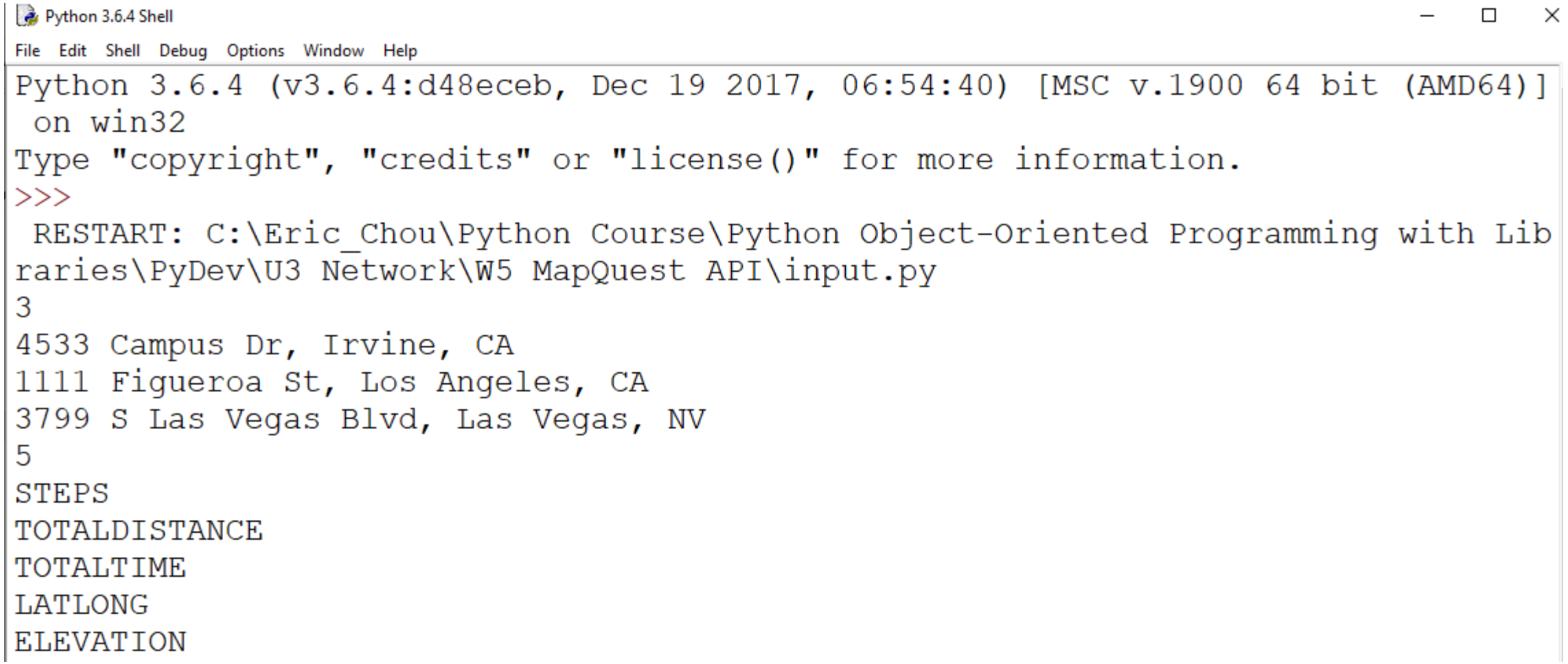
Demonstration

LECTURE 1

With Valid Route

- Run input.py and the input a valid route.

```
4533 Campus Dr, Irvine, CA
1111 Figueroa St, Los Angeles, CA
3799 S Las Vegas Blvd, Las Vegas, NV
```


Python 3.6.4 Shell
File Edit Shell Debug Options Window Help
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:54:40) [MSC v.1900 64 bit (AMD64)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Eric_Chou\Python Course\Python Object-Oriented Programming with Libraries\PyDev\U3 Network\W5 MapQuest API\input.py
3
4533 Campus Dr, Irvine, CA
1111 Figueroa St, Los Angeles, CA
3799 S Las Vegas Blvd, Las Vegas, NV
5
STEPS
TOTALDISTANCE
TOTALTIME
LATLONG
ELEVATION

```
DIRECTIONS
Start out going northwest on Campus Dr toward Carlson Ave.
Turn right onto Jamboree Rd.
Merge onto I-405/San Diego Fwy.
Merge onto I-110/Harbor Fwy via EXIT 37A.
Take the Adams Boulevard exit, EXIT 20C.
Turn left onto W Adams Blvd.
Turn right onto ramp.
Merge onto Figueroa St.
FIGUEROA STREET.
Start out going northeast on Figueroa St toward 9th St.
Turn left onto 8th St.
Merge onto CA-110/Pasadena Fwy.
Take the US-101/I-5 exit, EXIT 24A.
Merge onto US-101/Santa Ana Fwy.
Stay straight to go onto San Bernardino Fwy.
Merge onto I-15 E via EXIT 58A toward Barstow/Las Vegas (Crossing into Nevada).
Merge onto NV-159 via EXIT 41A toward Charleston Blvd East.
Turn left onto S Las Vegas Blvd.
S Las Vegas Blvd becomes Las Vegas Blvd N.
SOUTH LAS VEGAS BOULEVARD.

TOTAL DISTANCE:  317 miles

TOTAL TIME: 291 minutes

LATLONGS
33.66N -117.85W
34.05N -118.26W
36.16N -115.14W

ELEVATIONS
10
80
620

Directions Courtesy of MapQuest; Map Data Copyright OpenStreetMap Contributors.
>>>
```

With Valid Route 2

- Run input.py and the input a valid route.

```
5 Avenue Anatole France, 75007 Paris, France
6 Parvis Notre-Dame - Pl. Jean-Paul II,
75004 Paris, France
```

```
>>>
 RESTART: C:\Eric_Chou\Python Course\Python Object-Oriented Programming with Lib
raries\PyDev\U3 Network\W5 MapQuest API\input.py
2
5 Avenue Anatole France, 75007 Paris, France
6 Parvis Notre-Dame - Pl. Jean-Paul II, 75004 Paris, France
5
STEPS
TOTALDISTANCE
TOTALTIME
LATLONG
ELEVATION
```

DIRECTIONS
Start out going northeast on Avenue Gustave Eiffel toward Allée Adrienne Lecouvreur.
Avenue Gustave Eiffel becomes Avenue Silvestre de Sacy.
Turn left onto Avenue de la Bourdonnais.
Turn right onto Quai Branly.
Turn slight left onto Quai d'Orsay.
Quai d'Orsay becomes Boulevard Saint-Germain.
Keep left at the fork to continue on Boulevard Saint-Germain.
Turn left onto Rue des Bernardins.
Rue des Bernardins becomes Pont de l'Archevêché.
Pont de l'Archevêché becomes Quai de l'Archevêché.
Quai de l'Archevêché becomes Quai aux Fleurs.
Turn left onto Rue d'Arcole.
Turn left onto Rue du Cloître Notre-Dame.
6 PARVIS NOTRE-DAME - PLACE JEAN-PAUL II.

TOTAL DISTANCE: 4 miles

TOTAL TIME: 10 minutes

LATLONGS
48.86N 2.29E
48.85N 2.35E

ELEVATIONS
51
43

With Invalid Route

- Run input.py and the input an invalid route.

```
Los Angeles, CA, USA
Beijing, China
Tokyo, Japan
```

```
 RESTART: C:\Eric_Chou\Python Course\Python Object-Oriented Programming with Libraries\PyD
ev\U3 Network\W5 MapQuest API\input.py
3
Los Angeles, CA, USA
Beijing, China
Tokyo, Japan

NO ROUTE FOUND
```

Rewrite the Program

LECTURE 1

Programming Assignment

- Use only the mapquest.py file.
- Write your own input.py and output.py.
- Use input.py as the main program

mapquest.py

```python
#mapquest.py

import json
import urllib.parse
import urllib.request

API_KEY = 'iA97npZzYByGoYRoS77wuLxertjTJmC9' # Use your own key
BASE_MAPQUEST_URL = 'http://open.mapquestapi.com/directions/v2/route'
BASE_ELEVATION_URL = 'http://open.mapquestapi.com/elevation/v1/'

def build_route_url(locations:list) -> str:
    '''Creates the url for mapquest route api'''
    route_parameters = [
        ('key', API_KEY), ('from', locations[0])]
    for i in range(1, len(locations)):
        route_parameters.append(('to', locations[i]))

    return BASE_MAPQUEST_URL + '?' + urllib.parse.urlencode(route_parameters)
```

mapquest.py

```
def build_elevation_url(lat_long: str)-> str:
    '''Creates the url for the elevation route api'''
    elevation_parameters = [
        ('key', API_KEY), ('latLngCollection', lat_long)
    ]
    return BASE_ELEVATION_URL + 'profile?'+ urllib.parse.urlencode(elevation_parameters
)

def get_result(url: str) -> dict:
    '''Takes the url and creates the dict'''
    response = None
    try:
        response = urllib.request.urlopen(url)
        return json.load(response)
    finally:
        if response != None:
            response.close()
```