Google Maps/Directions API
Python Debugging

Nifty tools for projects!
Using Google Maps

- The “API” (Application Programmer Interface) is a way of submitting queries and getting answers from Google.
- You need an “API key” to do this
- There are limits on the number of queries you can make
Getting started

• Sign in to Google (e.g., read your Gmail)
• Then load
  https://code.google.com/apis/console
  to get an API key
Terms of Service
By using this API, you consent to be bound by the Google APIs Terms of Service ("API ToS") at https://developers.google.com/terms.
Make sure you’re logged in; click “Services”
Get a long list of services; Click on Geocoding API; accept terms of service
Check that all’s well

• Click on “Active” and see a list like this:
Getting the magic key

- Click on “API Access”
Get the API key

• Look for “Simple API Access” on that page, and copy the key:
Relatively easy from here

- [https://developers.google.com/maps/documentation/geocoding/?csw=1](https://developers.google.com/maps/documentation/geocoding/?csw=1)
Example code for three tasks

• 1. What are the lat/lon coordinates for this place?
• 2. What’s the time-zone for this place?
• 3. Is this lat/lon position in the US, and if so, what state is this lat/lon position in?
Find Lat/Lon for Boston

http://maps.googleapis.com/maps/api/geocode/json?address=Boston, MA
import urllib
import json

def getCoordinates(placeName):
    '''Takes in a string query Returns a (long, lat) tuple, both floats.'''
    googleGeocodeUrl = 'http://maps.googleapis.com/maps/api/geocode/json?'
    query = placeName.encode('utf-8')
    params = {
        'address': query
    }
    url = googleGeocodeUrl + urllib.urlencode(params)
    json_response = urllib.urlopen(url)
    response = json.loads(json_response.read())

    if response['results']:
        location = response['results'][0]['geometry']['location']
        latitude, longitude = location['lat'], location['lng']
    else:
        latitude, longitude = None, None
    return longitude, latitude
Notes

• If the location cannot be found, the results look like this:

{ "results" : [], "status" : "ZERO_RESULTS" }

So the "if response['results']": " turns out false, and we return None
What’s the time-zone for this place?

• Surprisingly tricky
• The time zone depends on the date!
  – Eastern Standard vs Eastern Daylight
• So we have to pass in the date and the lat/lon
  `def getTimeZone(dateString, latlonString)`
• Typical Use
  `def getTimeZone('11/23/2014', '42.3584308,-71.0597732')`
def getTimeZone(dateString, latlonString):
    googleTimeZoneURL = ".../maps/api/timezone...
    # Get Greenwich mean time at midnight on given date.
    timestamp = calendar.timegm(time.strptime(dateString, '%d/%m/%Y'))
    location = latlonString.encode('utf-8')
    params = {
        'location': location,
        'timestamp': timestamp
    }
    url = googleTimeZoneURL + urllib.urlencode(params)
    json_response = urllib.urlopen(url)
    response = json.loads(json_response.read())
    if response['timeZoneName']:
        return response['timeZoneName']
    else:
        return None
Is this lat/lon position in the US, and if so, what state is it in?

• Once again, use the “geocode” API
• This time, because we want to restrict to certain parts of the response, we have to include the “API key” from earlier
• Simply paste “&key=AIzaSyD...” onto the end of the URL
def getUSState(latlonString):
    googleMapsURL = "https://maps.googleapis.com/maps/... "
    params = "result_type=administrative_area_level_1&latlng="
    keyString = "&key=AIzaSy... "
    location = latlonString.encode('utf-8')

    url = googleMapsURL + params + location + keyString
    json_response = urllib.urlopen(url)
    response = json.loads(json_response.read())
    info = response['results'][0]['address_components']

    if (response['status'] != 'OK') :
        print("No response to query")
        return None

    ...
inUS = False
for u in info :
    if 'country' in u['types'] :
        inUS = (u['short_name'] == 'US')

if not inUS :
    print("Location not in US!")
    return None

for u in info :
    if 'administrative_area_level_1' in u['types'] :
        return u['short_name']

return None
PDB: A Python Debugger
import pdb

...
code
pdb.set_trace()
more code
Using the debugger

• During program execution, when the program reaches the set_trace() function, control returns to IDLE
• There’s a “(Pdb) ” prompt
• You can tell “the debugger” to do various things.
Pdb commands

• c : continue
• p x : print x
• s : take a single step (perhaps into a function-call)
• n : “next”; move on to the next line (by doing everything on this line without interruption)
• l : list a few lines around the current point of execution
• l 22: list a few lines starting at line 22
• q : quit (stop both PDB and your program)
Example Program

```python
import pdb

def update(mat):
    for i in range(0, 3):
        mat[i][i] = 0
    return mat

A = [1, 2, 3]
B = [A, A, A]

C = update(B)
print(C)
```

• What result do you expect?
import pdb

def update(mat):
    for i in range(0, 3):
        mat[i][i] = 0
    return mat

A = [1, 2, 3]
B = [A, A, A]

C = update(B)
print(C)

>>> ================== RESTART=====================
>>>

>>> [[0, 0, 0], [0, 0, 0], [0, 0, 0]]
import pdb

def update(mat):
    pdb.set_trace()
    for i in range(0, 3):
        mat[i][i] = 0
    return mat

A = [1, 2, 3]
B = [A, A, A]

C = update(B)
print(C)
>>> ============= RESTART ================
>>> 
> c:\users\jfh\desktop\lsa\pdbtest.py(5)update()
-> for i in range(0, 3):
  (Pdb) 1
  1     import pdb
  2
  3     def update(mat) :
  4         pdb.set_trace()
  5 ->     for i in range(0, 3):
  6           mat[i][i] = 0
  7         return mat
  8
  9     A = [1, 2, 3]
 10    B = [A, A, A]
 11
(Pdb) p mat
[[1, 2, 3], [1, 2, 3], [1, 2, 3]]
(Pdb)
def update(mat):
    pdb.set_trace()
    for i in range(0, 3):
        mat[i][i] = 0
    return mat

A = [1, 2, 3]
B = [A, A, A]
(Pdb) s
> c:\users\jfh\desktop\lsa\pdbtest.py(5) update()
-> for i in range(0, 3):
(Pdb) mat
[['0, 2, 3], [0, 2, 3], [0, 2, 3]]
(Pdb)
Aha!

- Somehow when we change mat[0][0], mat[1][0] and mat[2][0] are changing too!
- Box-and-pointers explain this: the arrows for mat[0], mat[1], and mat[2] all point to the same place
- When you change the first element of that pointed-to list, the results are what we saw!
Activity

• Open “geocodeExamples.py” and uncomment the stuff at the end, run the program, and look at test.kml.

• Then try to make the things in the Exercise at the bottom work. Try using PDB to help if you have troubles!