Download a preselected metro area from Map Zen

https://mapzen.com/metro-extracts/ walt-disney-world_florida.osm.bz2

1. Problems Encountered in the map. (Modified audit.py to fix this)

a. Noticed that the street address drive and city were not separated by "," in the full address ' Drive Windermere, FL 34786'*

```
import pymongo
postcodes = ["34786"]
db.waltdosm.find_one({"address.postcode" : { "$in": postcodes } })
```

```
{u'_id': ObjectId('54f2928e18a7741f049233a5'),
 u'address': {u'city': u'Windermere',
  u'country': u'US',
  u'county': u'Orange',
  u'full': u'8879 Newmarket Drive Windermere, FL 34786',
  u'housenumber': u'8879',
  u'postcode': u'34786',
  u'street': u'Newmarket Drive'},
```

In the function is_street_name added the following to deal with the addr:full information.
def is_street_name(elem):
# Should also include the addr:full to be fixed
return (elem.attrib['k'] == "addr:street") or (elem.attrib['k'] == "addr:full")

Modified the mapping variable to include the below values

mapping = {

"Ave ":"Avenue",

"St.": "Street",

"Rd." : "Road",

"N.":"North",

"St " : "Street",

"Blvd" : "Boulevard",

"Ln" : "Lane",

"Drive Windermere" : "Drive, Windermere" }

After cleanup it looks as

```
import pymongo
postcodes = ["34786"]
db.waltdosm.find_one({"address.postcode" : { "$in": postcodes } })
```

```
{u'_id': ObjectId('54f66cb818a7740cd0b6a318'),
 u'address': {u'city': u'Windermere',
  u'country': u'US',
  u'county': u'Orange',
  u'full': u'8879 Newmarket Drive, Windermere, FL 34786',
  u'housenumber': u'8879',
  u'postcode': u'34786',
  u'street': u'Newmarket Drive'},
```

b. "5730" was being displayed as a street address instead of house number. (line 265113 of the input file)

```
 <node id="3268334984" version="1" timestamp="2015-01-04T03:09:41Z" uid="1408522" user="Omnific" changeset="27901969" lat="28.3319787" lon="-81.5141469">
  <tag k="name" v="Supermarket"/>
  <tag k="addr:street" v="5730"/>
</node>
```

kmapping = {"addr:street":"addr:housenumber"}

Added the following to handle house number assignment

```
  if (tag.attrib['v'] == "5730"):
              tag.attrib['k'] = update_name(tag.attrib['k'],kmapping)
```

Below is the output after running the file through audit.py and inserting into the waltdosm collection.

```
import pymongo
hnum = ["5730"]
db.waltdosm.find_one({"address.housenumber" : { "$in": hnum } })
```

```
{u'_id': ObjectId('54f66cb218a7740cd0b668ca'),
 u'address': {u'housenumber': u'5730'},
 u'created': {u'changeset': u'27901969',
  u'timestamp': u'2015-01-04T03:09:41Z',
  u'uid': u'1408522',
  u'user': u'Omnific',
  u'version': u'1'},
 u'id': u'3268334984',
 u'name': u'Supermarket',
 u'pos': [28.3319787, -81.5141469],
 u'type': u'node'}
```

2. Data Overview

This section contains basic statistics about the dataset and the mongodb queries used to gather them.

# File sizes:

```
%load filesize.py
```

```
"""
Created on Sun Mar 01 16:02:26 2015

@author: Prema iyer
"""

import os
import pprint
OSM_FILE ="walt-disney-world_florida.osm"
JSON_FILE = "walt-disney-world_florida_audit.osm.json"

def getSize(OSM_FILE):
    st = os.stat(OSM_FILE)
    return st.st_size

def test():

    fsize = getSize(OSM_FILE)
    pprint.pprint("The size of the osm file:" + str(fsize/1000000) + ' MB')
    jsize = getSize(JSON_FILE)
    pprint.pprint("The size of the json file:" + str(jsize/1000000) + ' MB')

if __name__ == "__main__":
    test()
```

```
'The size of the osm file:52 MB'
'The size of the json file:59 MB'
```

# Number of documents:

```
total = db.waltdosm.find().count()
print "Total number of elements inserted into the database:" + str(total)
```

```
Total number of elements inserted into the database:271247
```

# Number of nodes and ways:

```
TotNodes = db.waltdosm.find({"type": "node"}).count()
print "Nodes Inserted: " + str(TotNodes)
TotWays = db.waltdosm.find({"type": "way"}).count()
print "Ways Inserted: " + str(TotWays)
```

```
Nodes Inserted: 248984
Ways Inserted: 22063
```

# The other element types inserted:

```
EleTyp = db.waltdosm.distinct( "type" )
print "List of element types:" + str(EleTyp)
print db.waltdosm.aggregate({"$group": {"_id" : "$type", "count":{"$sum":1}} } )
```

```
List of element types:[u'way', u'node', u'1', u'palm', u'conifer', u'automatic', u'broad_leaved', u'gas', u'water', u'multipolygon', u'
route']
{u'ok': 1.0, u'result': [{u'count': 2, u'_id': u'route'}, {u'count': 2, u'_id': u'multipolygon'}, {u'count': 2, u'_id': u'water'}, {u'c
ount': 126, u'_id': u'broad_leaved'}, {u'count': 1, u'_id': u'automatic'}, {u'count': 4, u'_id': u'conifer'}, {u'count': 52, u'_id': u'
palm'}, {u'count': 1, u'_id': u'1'}, {u'count': 248984, u'_id': u'node'}, {u'count': 10, u'_id': u'gas'}, {u'count': 22063, u'_id': u'w
ay'}]}
```

# Number of unique users:

```
print "Total number of contributors: " + str(len(db.waltdosm.distinct("created.uid")))
```

```
Total number of contributors: 164
```

# Top 1 contributing user:

```
db.waltdosm.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}}, {"$sort":{"count": -1}},{"$limit":1}])
```

```
{u'ok': 1.0, u'result': [{u'_id': u'NE2', u'count': 57430}]}
```

3. Additional Ideas

Analysis of data using mongodb queries:

Railway Station in the area along with lat and lon coordinates →

```
pipeline = [
            {'$match': {'railway':'station',
                        'name':{'$exists':1}}},
            {'$project':{'_id':'$name',
                         'coordinates':'$pos'}}
]
result  = db.waltdosm.aggregate(pipeline)['result']
pprint.pprint(result)
```

```
[{u'_id': u'Express Line: Magic Kingdom',
  u'coordinates': [28.4160297, -81.5823135]},
 {u'_id': u'Resort Line: Magic Kingdom',
  u'coordinates': [28.4158955, -81.5823535]},
 {u'_id': u'Resort Line: The Grand Floridian Resort',
  u'coordinates': [28.4108524, -81.5881486]},
 {u'_id': u'Resort Line: The Polynesian Village',
  u'coordinates': [28.4050124, -81.5851846]},
 {u'_id': u'Resort Line: Ticket and Transportation Center',
  u'coordinates': [28.4057505, -81.5796685]},
 {u'_id': u'Express Line: Ticket and Transportation Center',
  u'coordinates': [28.4056865, -81.5795519]},
 {u'_id': u'Epcot Line: Ticket and Transportation Center',
  u'coordinates': [28.4055624, -81.5793257]},
 {u'_id': u'Resort Line: The Contemporary Resort',
  u'coordinates': [28.4147901, -81.5745762]},
 {u'_id': u'Main Street Station', u'coordinates': [28.4163896, -81.5811922]},
 {u'_id': u'Frontierland Station', u'coordinates': [28.4196868, -81.5851419]},
 {u'_id': u'Harambe Station', u'coordinates': [28.3597539, -81.5913027]},
 {u'_id': u'Conservation Station', u'coordinates': [28.3635848, -81.589942]},
 {u'_id': u'Epcot Monorail Station', u'coordinates': [28.376797, -81.549616]},
 {u'_id': u'Carlwood Park Station', u'coordinates': [28.4211279, -81.5782691]},
 {u'_id': u'Ticket and Transportation Center'}]
```

Restaurants available →

```
pipeline = [
            {'$match': {'amenity':'restaurant',
                        'name':{'$exists':1}}},
            {'$project':{'_id':'$name',
                         'cuisine':'$cuisine'}
             }
]
result  = db.waltdosm.aggregate(pipeline)['result']
pprint.pprint(result)
```

```
[{u'_id': u"Chili's"},
 {u'_id': u"Logan's Roadhouse"},
 {u'_id': u'IHOP'},
 {u'_id': u"Logan's Roadhouse"},
 {u'_id': u'Olive Garden'},
 {u'_id': u"Cinderella's Royal Table"},
 {u'_id': u'Biergarten Restaurant', u'cuisine': u'german'},
 {u'_id': u'Nine Dragons Restaurant', u'cuisine': u'chinese'},
```

(Created the helper function autolabel based on the idea from the link http://stackoverflow.com/questions/7423445/how-can-i-display-text-over-columns-in-a-bar-chart-in-matplotlib*).*

```
import numpy as np
from matplotlib import pyplot as plt
```

```
def autolabel(rects,ax):
    for rect in rects:
        height = rect.get_height()
        ax.text(rect.get_x()+rect.get_width()/2., 1.05*height, height,
                ha='center', va='bottom')
    return ax
```

Created the "plt_graph" helper function to be used in the drawing of the bar chat.

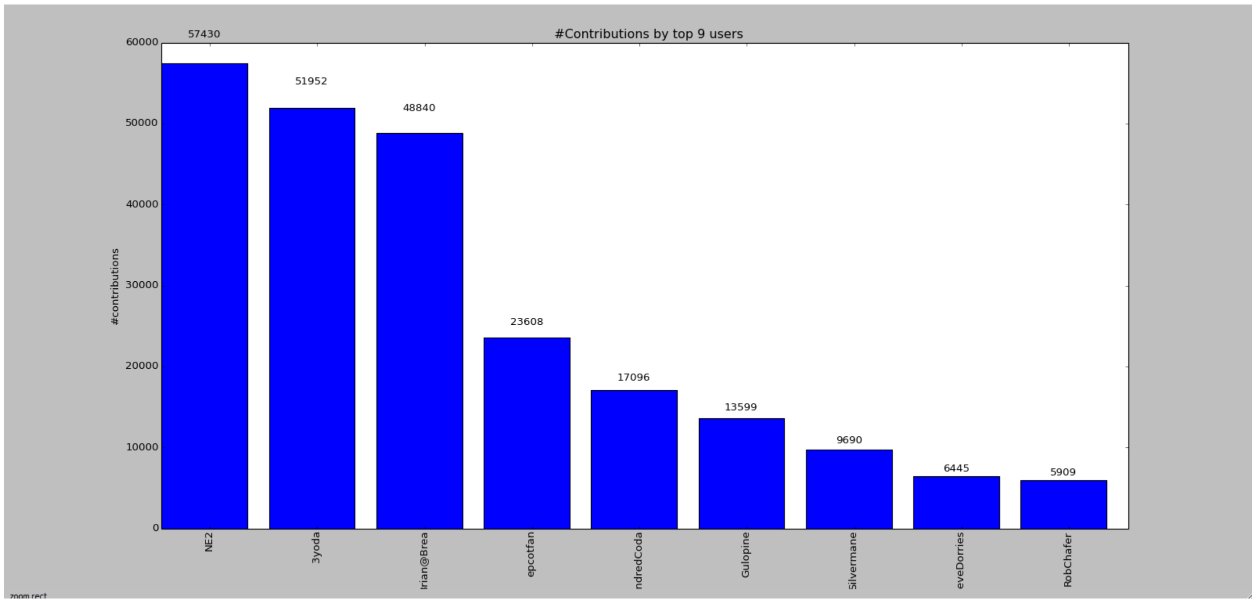
```
def plt_graph(x, y, title='', xlabel='', ylabel=''):

    fig = plt.figure()
    ax = plt.subplot(111)
    width = 0.9
    barlist = ax.bar(range(len(y)), y)
    ax = autolabel(barlist,ax)
    ax.set_xticks(np.arange(len(x)) + width/2)
    ax.set_xticklabels(x, rotation=90)
    ax.set_title("  " + title)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    return plt.show()
```

The above two helper functions are used in rendering the (i) and (ii) graphs.

(i)Below is the graph of the top 9 contributor's contributions.

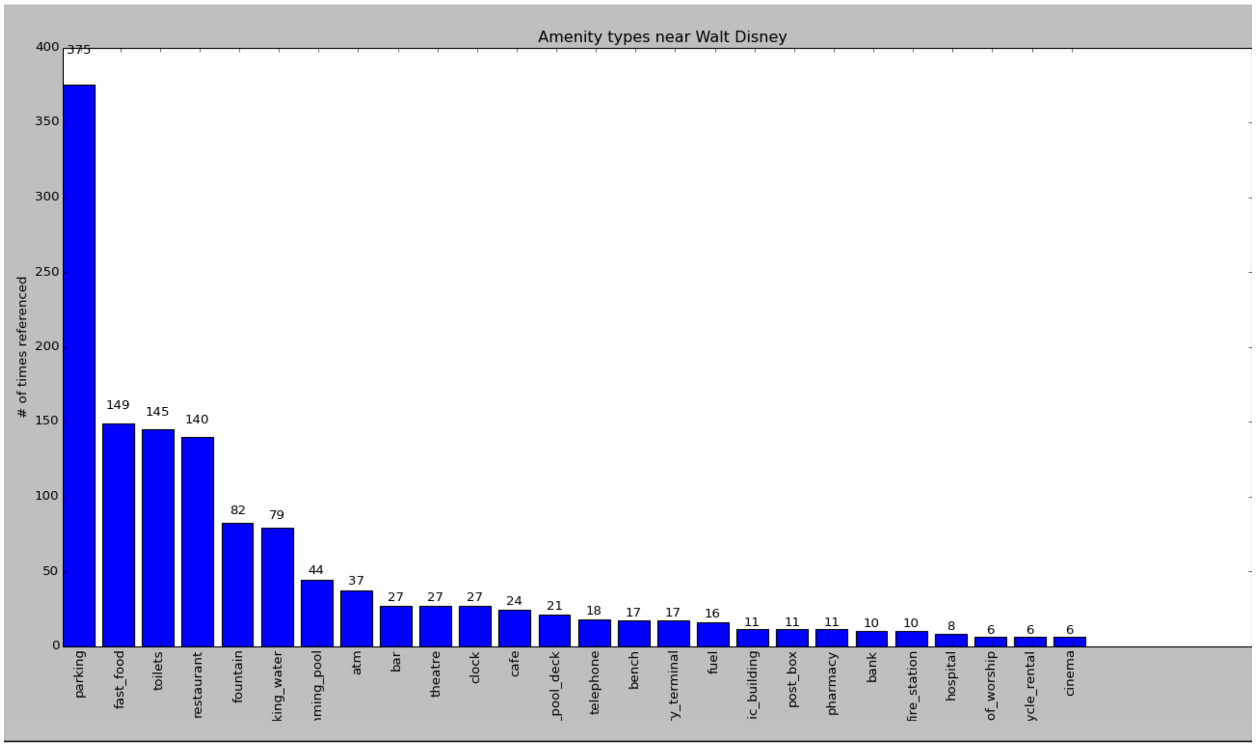
```
pipeline = [
            { "$group": { "_id": "$created.uid",
                          "name": { "$first": "$created.user"},
                          "count": { "$sum": 1 } } },
            { "$sort": { "count" : -1 } }
           ]
users = db.waltdosm.aggregate(pipeline)['result']
```

```
top_users = users[:9] #select the top 9 contributors
user_names = [ user['name'].encode('ascii', 'ignore') for user in top_users]
contributions = [ user['count'] for user in top_users]
plt_graph(user_names, contributions, title="#Contributions by top 9 users", xlabel='user names', ylabel="#contributions")
```



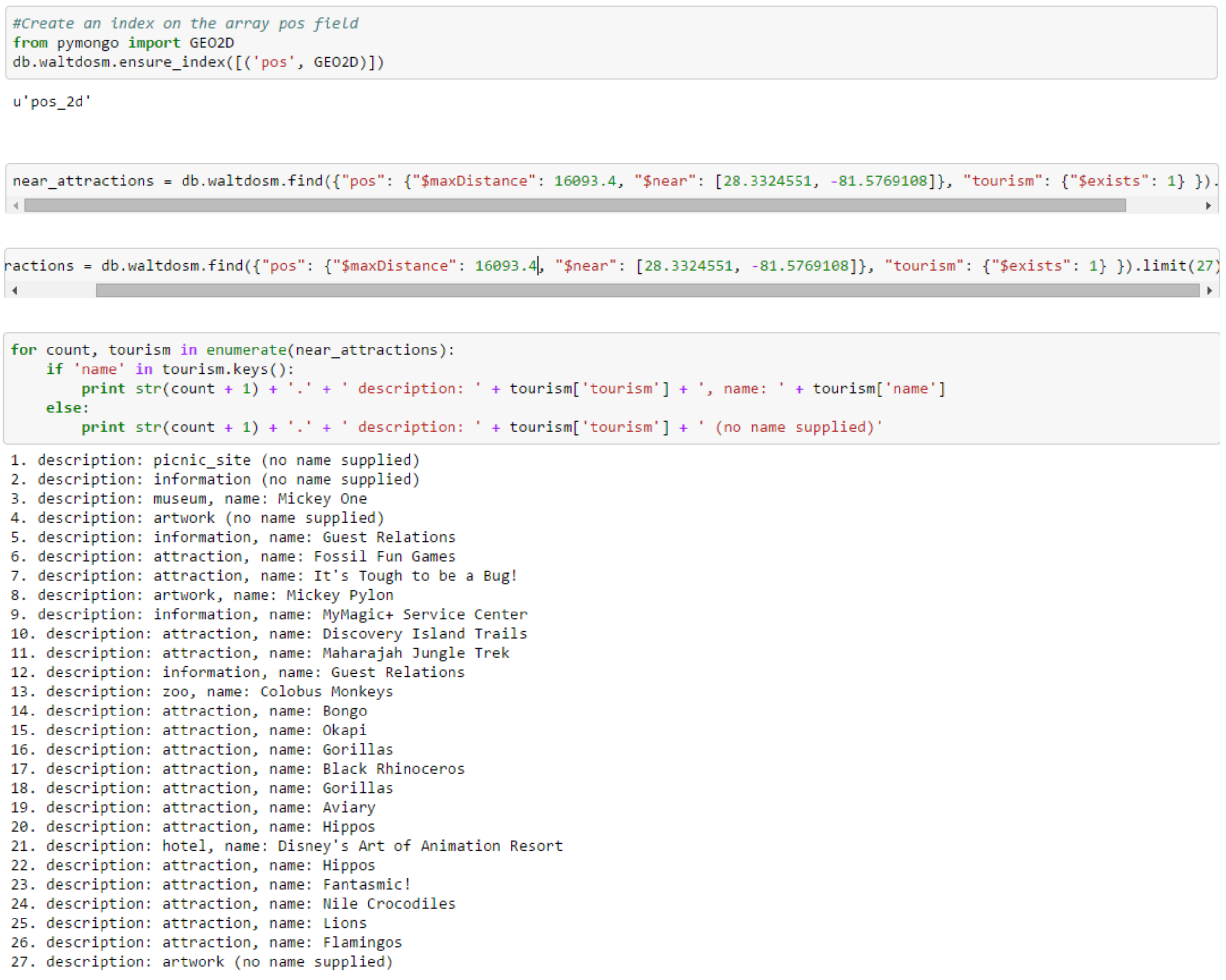
(ii) Below is the graph of the Amenities available in the Walt Disney Area:

```
TopAmen = amenities[1:27] #select top 27. 0 entry in array is 'None' (elements without amenity field)
amenity_names = [ amenity['_id'].encode('ascii', 'ignore') for amenity in TopAmen]
Amcount = [ amenity['count'] for amenity in TopAmen ]
plt_graph(amenity_names, Amcount, title="Amenity types near Walt Disney", xlabel='amenity types', ylabel="# of times referenced")
```



Places of interest in a 10 mile radius:

(http://api.mongodb.org/python/2.0.1/examples/geo.html)



```
#Create an index on the array pos field
from pymongo import GEO2D
db.waltdosm.ensure_index([('pos', GEO2D)])
```

```
u'pos_2d'
```

```
near_attractions = db.waltdosm.find({"pos": {"$maxDistance": 16093.4, "$near": [28.3324551, -81.5769108]}, "tourism": {"$exists": 1} }).
```

```
ractions = db.waltdosm.find({"pos": {"$maxDistance": 16093.4, "$near": [28.3324551, -81.5769108]}, "tourism": {"$exists": 1} }).limit(27)
```

```
for count, tourism in enumerate(near_attractions):
    if 'name' in tourism.keys():
        print str(count + 1) + '.' + ' description: ' + tourism['tourism'] + ', name: ' + tourism['name']
    else:
        print str(count + 1) + '.' + ' description: ' + tourism['tourism'] + ' (no name supplied)'
```

```
1. description: picnic_site (no name supplied)
2. description: information (no name supplied)
3. description: museum, name: Mickey One
4. description: artwork (no name supplied)
5. description: information, name: Guest Relations
6. description: attraction, name: Fossil Fun Games
7. description: attraction, name: It's Tough to be a Bug!
8. description: artwork, name: Mickey Pylon
9. description: information, name: MyMagic+ Service Center
10. description: attraction, name: Discovery Island Trails
11. description: attraction, name: Maharajah Jungle Trek
12. description: information, name: Guest Relations
13. description: zoo, name: Colobus Monkeys
14. description: attraction, name: Bongo
15. description: attraction, name: Okapi
16. description: attraction, name: Gorillas
17. description: attraction, name: Black Rhinoceros
18. description: attraction, name: Gorillas
19. description: attraction, name: Aviary
20. description: attraction, name: Hippos
21. description: hotel, name: Disney's Art of Animation Resort
22. description: attraction, name: Hippos
23. description: attraction, name: Fantasmic!
24. description: attraction, name: Nile Crocodiles
25. description: attraction, name: Lions
26. description: attraction, name: Flamingos
27. description: artwork (no name supplied)
```