Documentation_0.0.5

February 10, 2019

1 geoClassy

The geoClassy package was developed from the need of using GeoJSON files with OpenStreetMap data to classify points which GPS coordinates are known. Using the GeoJSON format as input, the package can also be used with files from sources different from OpenStreetMap.

Examples of use:

- find the neighborhood of the city with most AirBnB beds;
- classification of customers by key account area of competence;
- filter of road accidents in a certain province;
- assessment of the air quality for a specific Municipality;
- list of cities affected by the passage of an hurricane.

2 Contact

If you wish to contact the author of this module you can just send an e-mail to *support@simboli.eu*. If you want to read the lastest news about this package you can stay connected with the Simboli.EU blog.

3 Changelog

3.1 Version 0.0.5 (beta)

- Small code diet;
- Added project documentation.

3.2 Version 0.0.4 (beta)

Fist downloadable version: * Project added to PyPI

3.3 Versions <= 0.0.3

Test versions used for debugs and primary tests.

4 Data sources

The module was designed to accept as input the GeoJSON files from the OpenStreetMap database. This data can be extracted in several ways, some of my favourites are the following:

- GeoJSON page downloaded from Simboli.eu;
- Overpass-Turbo;
- Polygons OpenStreetMap FR.

4.1 GeoJSON page on this web site

That's the easiest way to download a complete and ready to use file geoJSON to feed the geoClassy package. All available geoJSON files can be found on this page. If you need some specific files just mail me and I'll try to help you.

4.2 Overpass-Turbo

This is a powerful tool and I can say that it is one of the best tool you can learn if you want to work in the data science using geographical data. The platform can be found on this page where there is also a useful Query Wizard for those who are using the tool for their first times. Query should be written in Overpass XML or Overpass QL, a getting started guide and examples cookbook are provided from OpenStreetMap.

4.3 Polygons OpenStreetMap FR

If you are a data nerd (and probably you are if you are using this package) you could think about including in your projects this direct link to execute Overpass queries. Please note that the linked tutorial shows how to extract data in JSON, you also have to convert it in geoJSON if you want to use with geoClassy.

5 Requirements

The module, to work properly, requires that the Shapely module and the Json module have also been installed.

If don't have them, you have to install them before continue. Json is a build-in package so your Python installation should already include it, Shapely could be downloaded by its Pypi page.

6 Installation

As most of the Python modules, geoClassy can be installated with the easy PIP installer, just using the following: `pip install geoClassy`

7 Import

The best way to import this module is the following

```
In [1]: from geoClassy import single as gCs
```

8 Usage ot the geoClassy package

8.1 requisites()

This methods does not accept any parameters and try to import Shapely and JSON package as they are needed to use geoClassy module.

```
In [2]: gCs.requisites()
```

```
Json module correctly imported
Shapely Geometry module correctly imported
```

8.2 loadFile(fname)

This methods is the first to use in the program as it read the geoJSON file (the path should be stored into a variable fname or write directly between the two braches) and build n polygons for every closed area defined in the file.

If the GeoJSON file is in the same folder where the Python script is saved you can use only the filename:

```
In [3]: GEOJSON_PATH='USA New York City neighborhoods 20190127.geojson'
        gCs.loadFile(GEOJSON_PATH)
```

otherwise you can specify the complete path:

```
In [4]: GEOJSON_PATH='/home/Paul/geoClassy_documentation/USA New York City neightborhoods 201901
        gCs.loadFile(GEOJSON_PATH)
```

8.3 numPoly()

It simply shows the number of polygons that are being imported into our systems.

```
In [5]: gCs.numPoly()
```

```
55  polygons loaded
```

8.4 checkPoly()

This methods print the list of polygons loaded in the LoadFile step and print if they are valid or not.

```
In [6]: gCs.checkPoly()
```

```
Polygon 0 : Marble Hill
ok
Polygon 1 : Times Square
ok
Polygon 2 : Flatiron District
```

```
ok
Polygon 3 : Inwood
ok
Polygon 4 : Fort George
ok
Polygon 5 : Hudson Heights
ok
Polygon 6 : Washington Heights
ok
Polygon 7 : Sugar Hill
ok
Polygon 8 : Hamilton Heights
ok
Polygon 9 : Manhattanville
ok
Polygon 10 : Morningside Heights
ok
Polygon 11 : Manhattan Valley
ok
Polygon 12 : Harlem
ok
Polygon 13 : East Harlem
ok
Polygon 14 : Carnegie Hill
ok
Polygon 15 : Yorkville
ok
Polygon 16 : Upper West Side
ok
Polygon 17 : Lincoln Square
ok
Polygon 18 : Columbus Circle
ok
Polygon 19 : Upper East Side
ok
Polygon 20 : Lenox Hill
ok
Polygon 21 : Midtown
ok
Polygon 22 : Midtown East
ok
Polygon 23 : Turtle Bay
ok
Polygon 24 : Tudor City
ok
Polygon 25 : Murray Hill
ok
Polygon 26 : Hudson Yards
```

```
ok
Polygon 27 : Hell's Kitchen
ok
Polygon 28 : Garment District
ok
Polygon 29 : Herald Square
ok
Polygon 30 : Koreatown
ok
Polygon 31 : Midtown South
ok
Polygon 32 : NoMad
ok
Polygon 33 : Rose Hill
ok
Polygon 34 : Kips Bay
ok
Polygon 35 : Stuy Town
ok
Polygon 36 : Gramercy
ok
Polygon 37 : Union Square
ok
Polygon 38 : Chelsea
ok
Polygon 39 : Meatpacking District
ok
Polygon 40 : Alphabet City
ok
Polygon 41 : East Village
ok
Polygon 42 : NoHo
ok
Polygon 43 : Greenwich Village
ok
Polygon 44 : West Village
ok
Polygon 45 : Hudson Square
ok
Polygon 46 : SoHo
ok
Polygon 47 : Little Italy
ok
Polygon 48 : Lower East Side
ok
Polygon 49 : Chinatown
ok
Polygon 50 : TriBeCa
```

```
ok
Polygon 51 : Civic Center
ok
Polygon 52 : Two Bridges
ok
Polygon 53 : Financial District
ok
Polygon 54 : Battery Park City
ok
```

8.5 polyList()

This method print all the polygons that are stored in the GeoJSON file.

```
In [7]: gCs.polyList()
```

```
Out[7]: ['Marble Hill',
         'Times Square',
         'Flatiron District',
         'Inwood',
         'Fort George',
         'Hudson Heights',
         'Washington Heights',
         'Sugar Hill',
         'Hamilton Heights',
         'Manhattanville',
         'Morningside Heights',
         'Manhattan Valley',
         'Harlem',
         'East Harlem',
         'Carnegie Hill',
         'Yorkville',
         'Upper West Side',
         'Lincoln Square',
         'Columbus Circle',
         'Upper East Side',
         'Lenox Hill',
         'Midtown',
         'Midtown East',
         'Turtle Bay',
         'Tudor City',
         'Murray Hill',
         'Hudson Yards',
         "Hell's Kitchen",
         'Garment District',
         'Herald Square',
         'Koreatown',
         'Midtown South',
         'NoMad',
         'Rose Hill',
         'Kips Bay',
         'Stuy Town',
         'Gramercy',
         'Union Square',
         'Chelsea',
         'Meatpacking District',
         'Alphabet City',
         'East Village',
         'NoHo',
         'Greenwich Village',
         'West Village',
         'Hudson Square',
         'SoHo',
         'Little Italy',
         'Lower East Side',
         'Chinatown',
         'TriBeCa',
         'Civic Center',
         'Two Bridges',
         'Financial District',
         'Battery Park City']
```

You can also use to make a list of the polygons:

```
In [8]: Names=gCs.polyList()
        print(Names)
```

```
['Marble Hill', 'Times Square', 'Flatiron District', 'Inwood', 'Fort George', 'Hudson Heights',
```

8.6 getNames(lat, lon)

It shows the name of the polygons which contains the point (latitude, longitude) is. If the point is outside every polygon unknown is printed.

```
In [9]: # Empire State Building
        gCs.getNames(40.748417,-73.985833)
```

```
Out[9]: 'Midtown South'
```

This method could be used also with Pandas Dataframe:

```
In [10]: import pandas as pd
         import numpy as np
         df = pd.DataFrame(np.array([['Ghostbusters Firehouse',40.719646, -74.006297], ['New Yor
         df
```

```
Out[10]:                 description    latitude    longitude
        0   Ghostbusters Firehouse   40.719646   -74.006297
        1  New York Public Library   40.752042   -73.993447
        2              White house  38.8976998  -77.0365535
```

```
In [11]: df['zone'] = df.apply(lambda row: gCs.getNames(pd.to_numeric(row.latitude), pd.to_numer
         df
```

```
Out[11]:                 description    latitude    longitude     zone
        0   Ghostbusters Firehouse   40.719646   -74.006297  TriBeCa
        1  New York Public Library   40.752042   -73.993447  Chelsea
        2              White house  38.8976998  -77.0365535  unknown
```

last revision of this document: 2019-02-10