# 3.QGIS_Exploring PyQGIS and Native Algorithms for Spatial Analytics

## Data Browser:

**Question: which neighborhoods in San Francisco are less likely to have tree cover?**

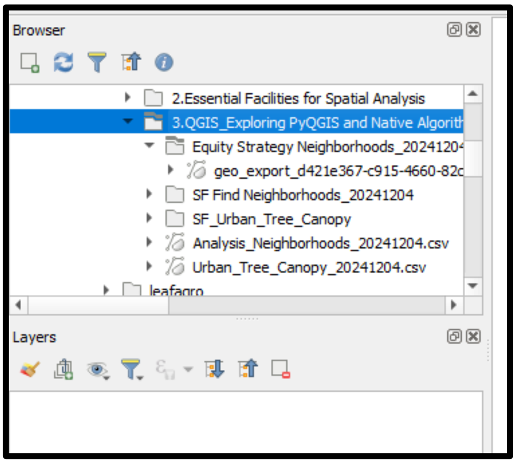
**Datasets:**

SF_Neighnorhood**:** SF_ Neighnorhood Data

Equality Strategy Neighborhood**:-** ESN Layer

SF_Trees Layer:- SF_Trees_Canopy

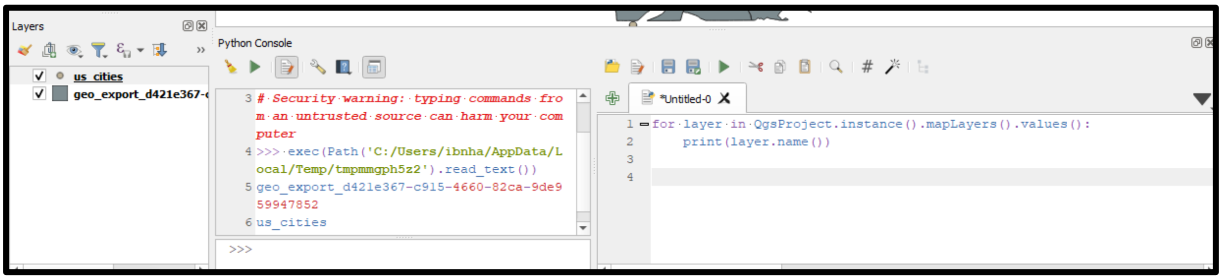
- Add data from Browser by double clicking, the data added in Layer Panel.


Browser
2.Essential Facilities for Spatial Analysis
3.QGIS_Exploring PyQGIS and Native Algorith
Equity Strategy Neighborhoods_2024120
geo_export_d421e367-c915-4660-82c
SF Find Neighborhoods_20241204
SF_Urban_Tree_Canopy
Analysis_Neighborhoods_20241204.csv
Urban_Tree_Canopy_20241204.csv
leafagro
Layers

- Print layers present in layer panel by Using Python Command in Python Console.

Python code:

```
for layer in QgsProject.instance().mapLayers().values():
        print(layer.name())
```



```
# Security warning: typing commands from an untrusted source can harm your computer
>>> exec(Path('C:/Users/ibnha/AppData/Local/Temp/tmpmmgph5z2').read_text())
geo_export_d421e367-c915-4660-82ca-9de959947852
us_cities
>>>
```

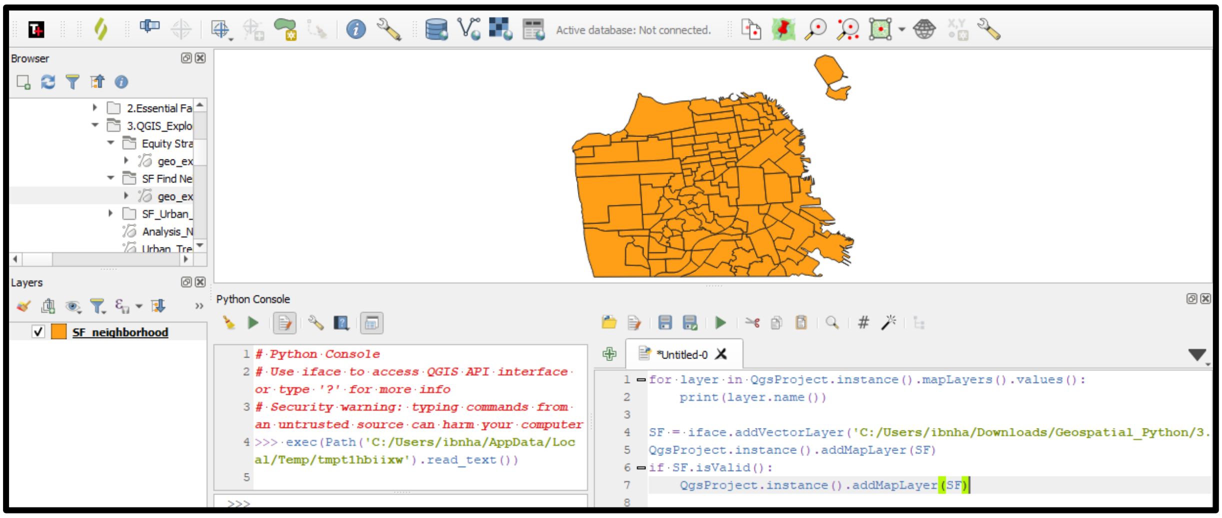
```
for layer in QgsProject.instance().mapLayers().values():
    print(layer.name())
```

## SF_Neighborhood:

- Add SF_Neighborhood using PyQGIS python Console.

**Python Code:**

```
SF = iface.addVectorLayer('ShpFile.shp', 'SF_neighborhood', 'ogr')

QgsProject.instance().addMapLayer(SF)

If SF.isValid():

        QgsProject.instance().addMapLayer(SF)
```



```
# Python Console
# Use iface to access QGIS API interface or type '?' for more info
# Security warning: typing commands from an untrusted source can harm your computer
>>> exec(Path('C:/Users/ibnha/AppData/Local/Temp/tmpt1hbiixw').read_text())
```

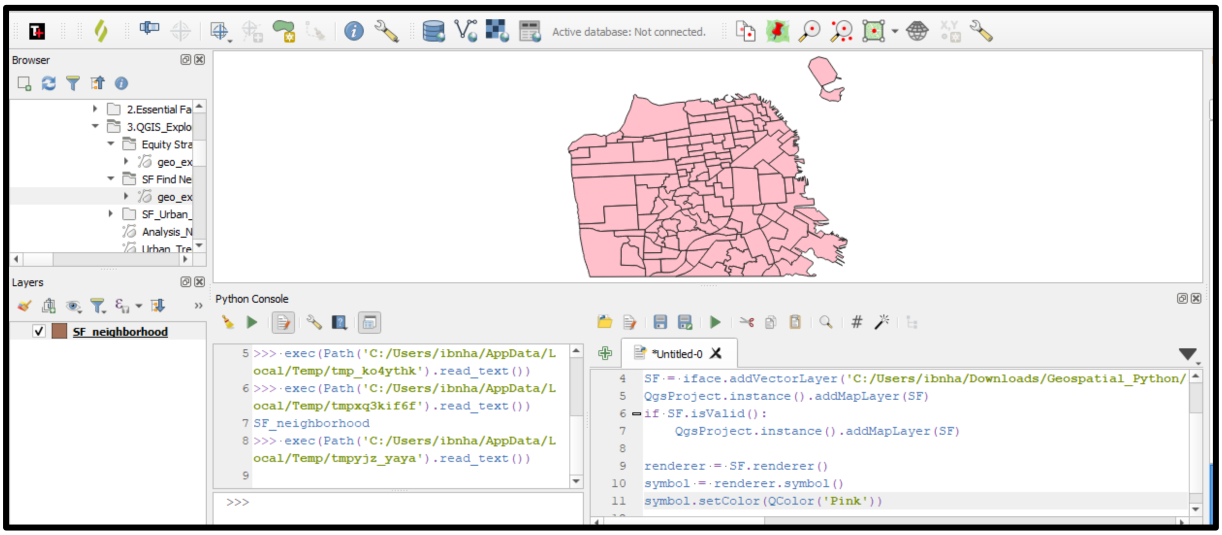
```
for layer in QgsProject.instance().mapLayers().values():
    print(layer.name())

SF = iface.addVectorLayer('C:/Users/ibnha/Downloads/Geospatial_Python/3.
QgsProject.instance().addMapLayer(SF)
if SF.isValid():
    QgsProject.instance().addMapLayer(SF)
```

- Set Color Using PyQGIS:
  - **Python Code:**

    **renderer = SF.renderer()**

    **symbol = renderer.symbol()**

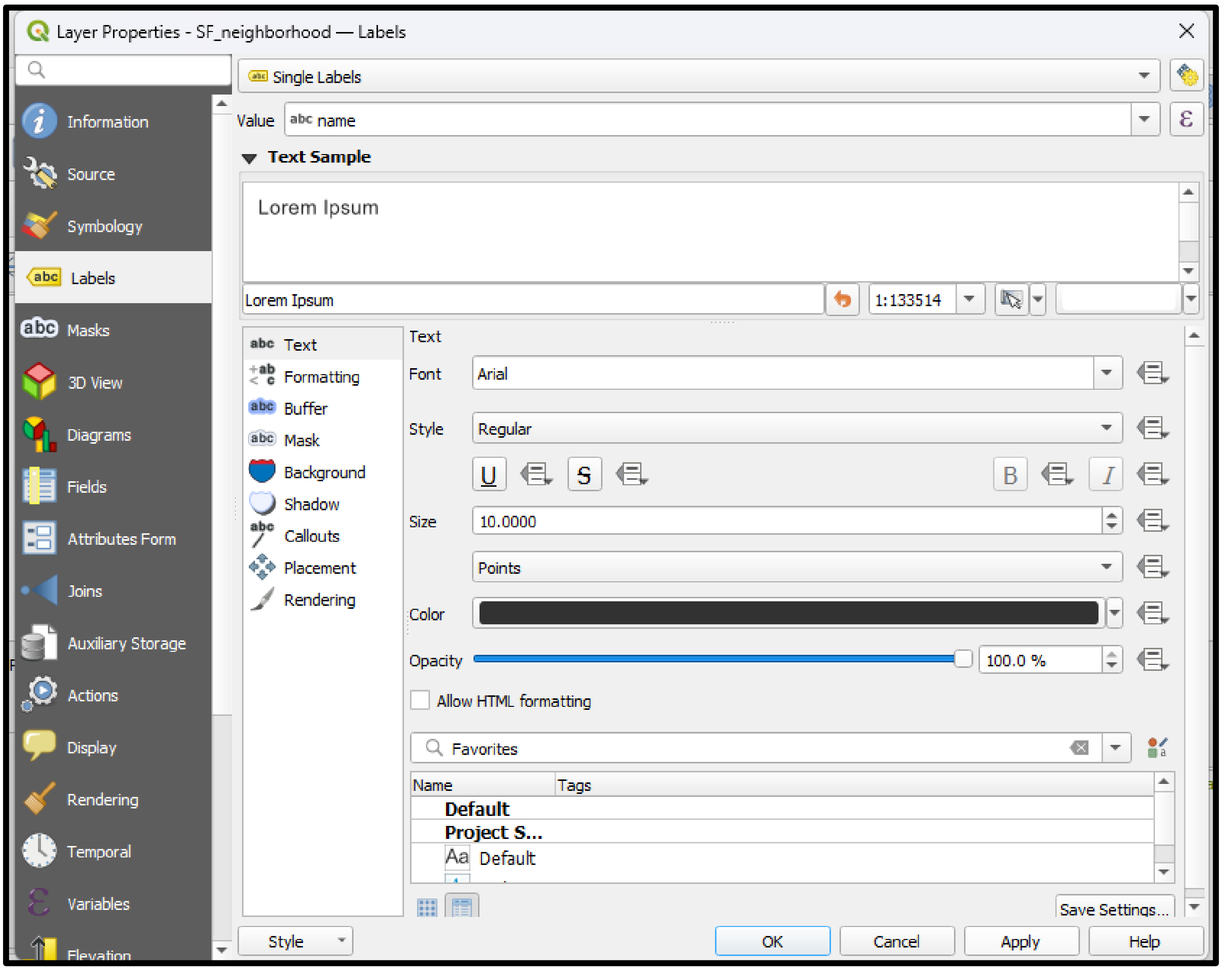
    **symbol.setColor(QColor(‘Pink’))**



```
>>> exec(Path('C:/Users/ibnha/AppData/Local/Temp/tmp_ko4ythk').read_text())
>>> exec(Path('C:/Users/ibnha/AppData/Local/Temp/tmpxq3kif6f').read_text())
SF_neighborhood
>>> exec(Path('C:/Users/ibnha/AppData/Local/Temp/tmpyjz_yaya').read_text())
```

```
SF = iface.addVectorLayer('C:/Users/ibnha/Downloads/Geospatial_Python/
QgsProject.instance().addMapLayer(SF)
if SF.isValid():
    QgsProject.instance().addMapLayer(SF)

renderer = SF.renderer()
symbol = renderer.symbol()
symbol.setColor(QColor('Pink'))
```

- Right Click the data -> Select Properties -> Go to Label, Select Single Labels -> Apply -> Ok



Layer Properties - SF_neighborhood — Labels

Single Labels

Value name

Text Sample

Lorem Ipsum

Lorem Ipsum 1:133514

Text

Font Arial

Style Regular

Size 10.0000

Points

Color

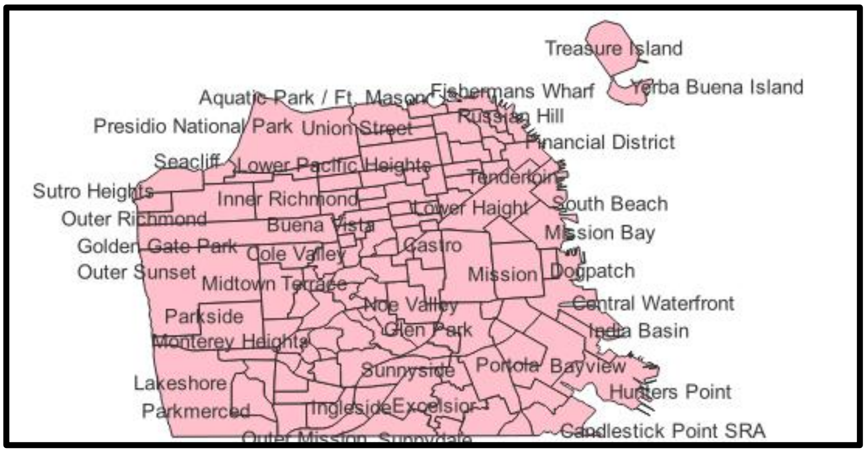
Opacity 100.0 %

□ Allow HTML formatting

Favorites

| Name | Tags |
|---|---|
| Default | |
| Project S... | |
| Aa Default | |

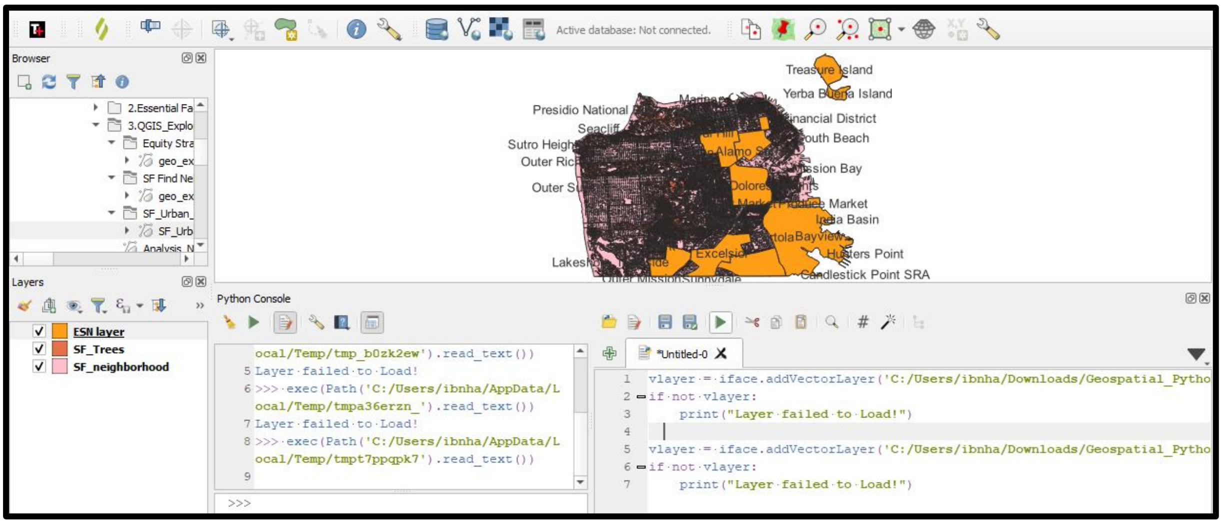
## Result of Labelled Data



- Add Balance data to the map SF_Urban_Tree and Equity Strategy Neighborhoods

```
vlayer = iface.addVectorLayer('C:/Users/ibnha/Downloads/Geospatial_Python/3.QGIS_Exploring PyQGIS and Native Algorithms for Spatial Analytics/SF_Urban_Tree_Canopy/SF_Urban_Tree_Canopy.shp','SF_Trees','ogr')
if not vlayer:
  print("Layer failed to Load!")

vlayer = iface.addVectorLayer('C:/Users/ibnha/Downloads/Geospatial_Python/3.QGIS_Exploring PyQGIS and Native Algorithms for Spatial Analytics/Equity Strategy Neighborhoods_20241204/geo_export_d421e367-c915-4660-82ca-9de959947852.shp','ESN layer','ogr')
```

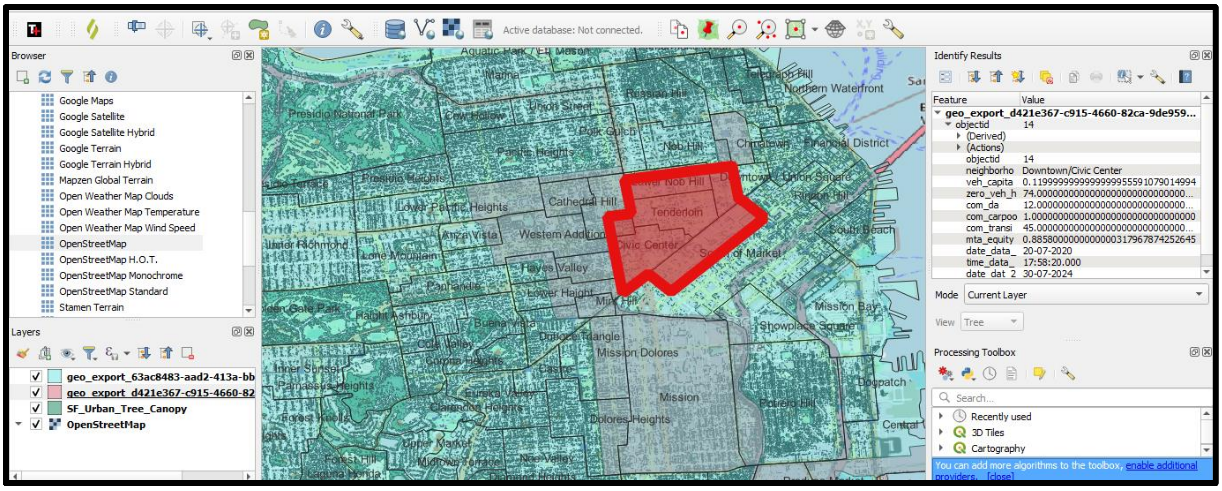
```
if not vlayer:

    print("Layer failed to Load!")
```



```
ocal/Temp/tmp_b0zk2ew').read_text())
Layer failed to Load!
>>> exec(Path('C:/Users/ibnha/AppData/L
ocal/Temp/tmpa36erzn_').read_text())
Layer failed to Load!
>>> exec(Path('C:/Users/ibnha/AppData/L
ocal/Temp/tmpt7ppqpk7').read_text())

>>>
```

```
vlayer = iface.addVectorLayer('C:/Users/ibnha/Downloads/Geospatial_Pytho
if not vlayer:
    print("Layer failed to Load!")

vlayer = iface.addVectorLayer('C:/Users/ibnha/Downloads/Geospatial_Pytho
if not vlayer:
    print("Layer failed to Load!")
```

**Addressing the Research Question: How does the tree-cover data line up with low-income neighborhoods? Can you get a sense of the answer by looking at the map?**

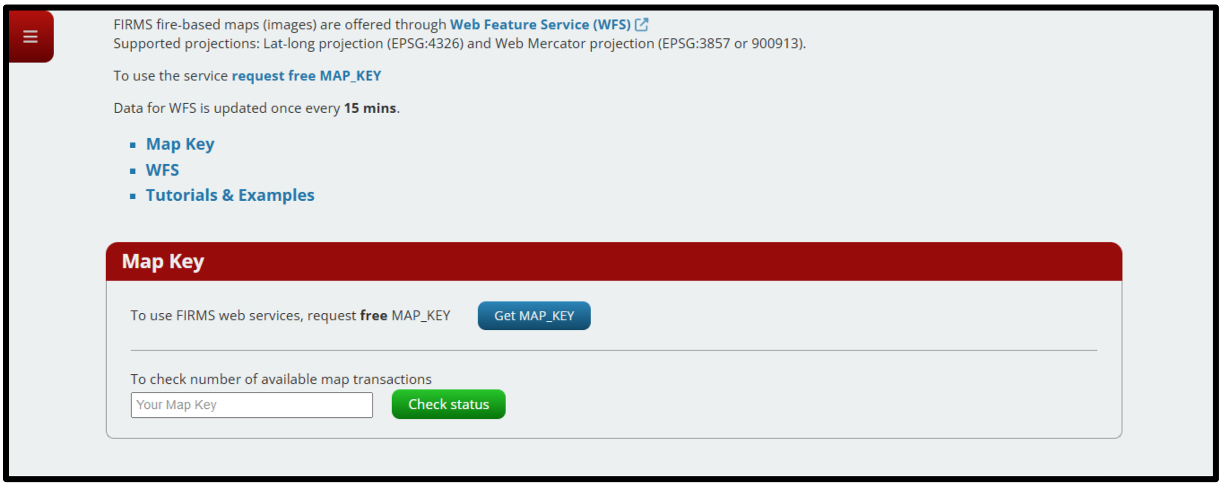


| Feature | Value |
|---|---|
| geo_export_d421e367-c915-4660-82ca-9de959... | |
| objectid | 14 |
| (Derived) | |
| (Actions) | |
| objectid | 14 |
| neighborho | Downtown/Civic Center |
| veh_capita | 0.119999999999999955591079014994 |
| zero_veh_h | 74.0000000000000000000000000... |
| com_da | 12.0000000000000000000000000... |
| com_carpoo | 1.00000000000000000000000000000 |
| com_transi | 45.0000000000000000000000000... |
| mta_equity | 0.8858000000000000317967874252645 |
| date_data_ | 20-07-2020 |
| time_data_ | 17:58:20.000 |
| date_dat_2 | 30-07-2024 |

**Working On Web Feature Service (QGIS)**

## Insert WFS Data in QGIS:

- **Get the FIRMS fire-based maps Map_Key by requesting Map_key.**



FIRMS fire-based maps (images) are offered through **Web Feature Service (WFS)**
Supported projections: Lat-long projection (EPSG:4326) and Web Mercator projection (EPSG:3857 or 900913).

To use the service **request free MAP_KEY**

Data for WFS is updated once every **15 mins**.

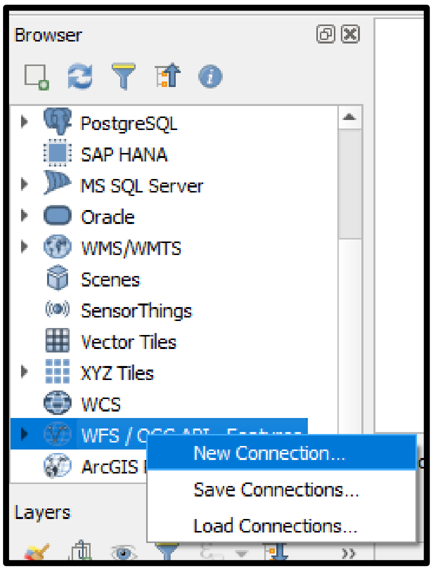
- **Map Key**
- **WFS**
- **Tutorials & Examples**

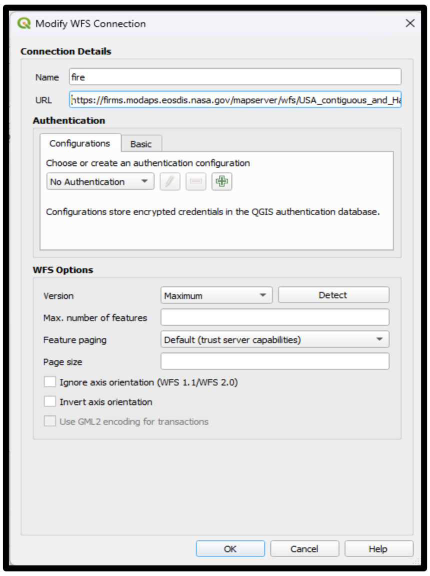
**Map Key**

To use FIRMS web services, request **free** MAP_KEY  Get MAP_KEY

To check number of available map transactions

Your Map Key  Check status





Modify WFS Connection

**Connection Details**

Name: fire

URL: https://firms.modaps.eosdis.nasa.gov/mapserver/wfs/USA_contiguous_and_H

**Authentication**

Configurations | Basic

Choose or create an authentication configuration

No Authentication

Configurations store encrypted credentials in the QGIS authentication database.

**WFS Options**

Version: Maximum  Detect

Max. number of features:

Feature paging: Default (trust server capabilities)

Page size:

□ Ignore axis orientation (WFS 1.1/WFS 2.0)

□ Invert axis orientation

□ Use GML2 encoding for transactions

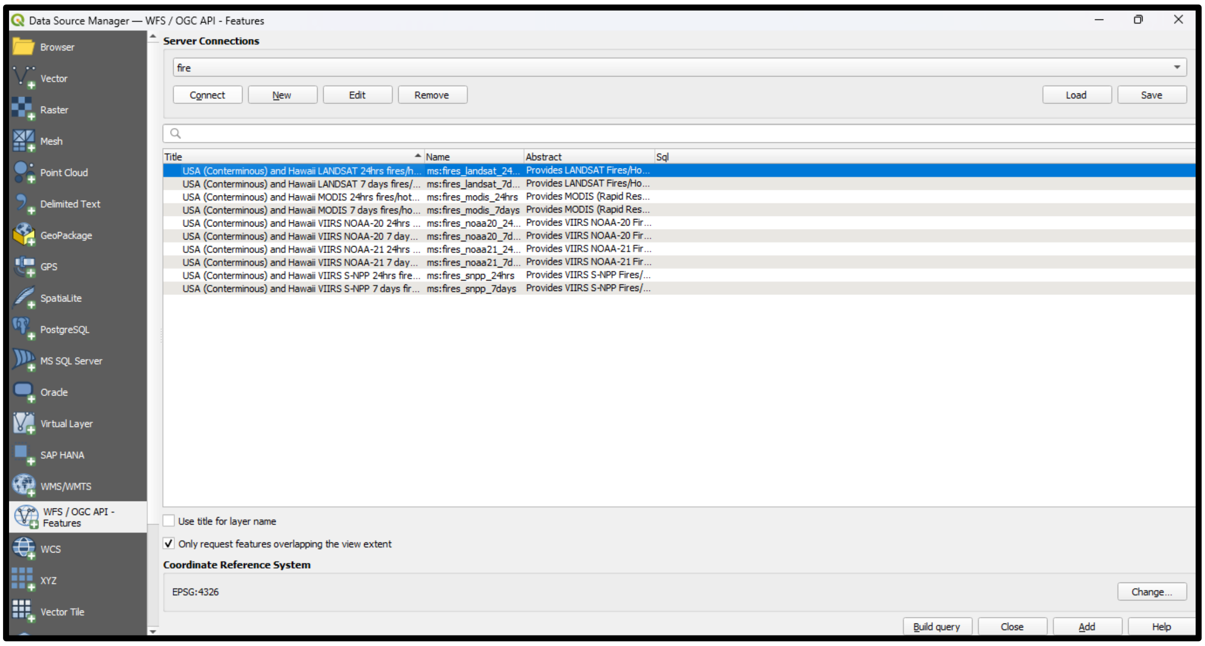
OK  Cancel  Help

- Give the desired name and URL to connect with FIRMS with Map_Key

**Format :**

**https://firms.modaps.eosdis.nasa.gov/mapserver/wfs/USA_contiguous_and_Hawaii/Map_key**

**Connect Web Feature Service Using FIRMS**



fire

| Title | Name | Abstract | Sql |
|---|---|---|---|
| USA (Conterminous) and Hawaii LANDSAT 24hrs fires/h... | ms:fires_landsat_24... | Provides LANDSAT Fires/Ho... | |
| USA (Conterminous) and Hawaii LANDSAT 7 days fires/... | ms:fires_landsat_7d... | Provides LANDSAT Fires/Ho... | |
| USA (Conterminous) and Hawaii MODIS 24hrs fires/hot... | ms:fires_modis_24hrs | Provides MODIS (Rapid Res... | |
| USA (Conterminous) and Hawaii MODIS 7 days fires/ho... | ms:fires_modis_7days | Provides MODIS (Rapid Res... | |
| USA (Conterminous) and Hawaii VIIRS NOAA-20 24hrs ... | ms:fires_noaa20_24... | Provides VIIRS NOAA-20 Fir... | |
| USA (Conterminous) and Hawaii VIIRS NOAA-20 7 day... | ms:fires_noaa20_7d... | Provides VIIRS NOAA-20 Fir... | |
| USA (Conterminous) and Hawaii VIIRS NOAA-21 24hrs ... | ms:fires_noaa21_24... | Provides VIIRS NOAA-21 Fir... | |
| USA (Conterminous) and Hawaii VIIRS NOAA-21 7 day... | ms:fires_noaa21_7d... | Provides VIIRS NOAA-21 Fir... | |
| USA (Conterminous) and Hawaii VIIRS S-NPP 24hrs fire... | ms:fires_snpp_24hrs | Provides VIIRS S-NPP Fires/... | |
| USA (Conterminous) and Hawaii VIIRS S-NPP 7 days fir... | ms:fires_snpp_7days | Provides VIIRS S-NPP Fires/... | |

□ Use title for layer name

☑ Only request features overlapping the view extent

Coordinate Reference System

EPSG:4326

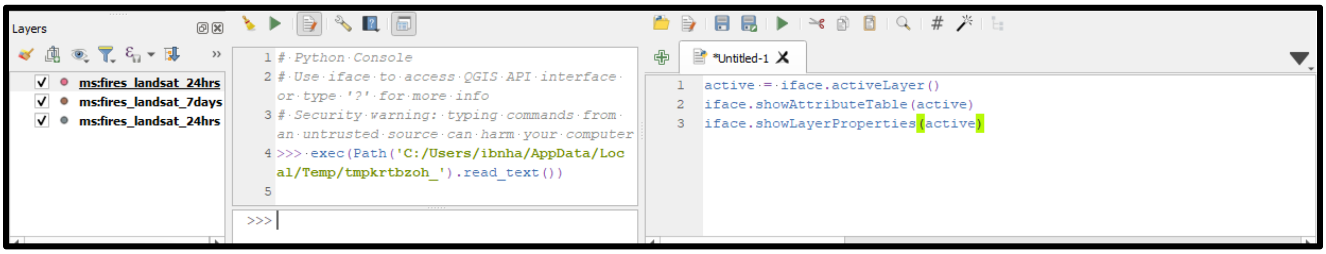
- Then select the layer and click add in Map layer.
- In PyQGIS Python Console, Get the Layer Attribute Table and Layer Properties

**Code:**

**active = iface.activeLayer()**

**iface.showAttributeTable(active)**

**iface.showLayerProperties(active)**

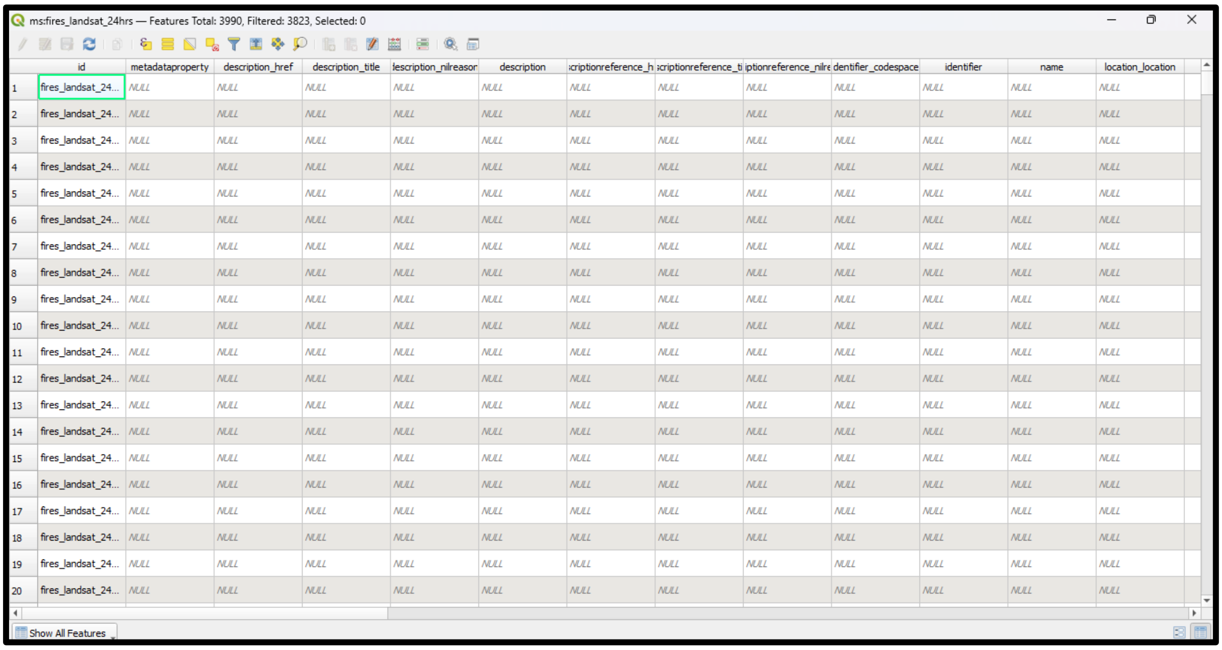


☑ ms:fires_landsat_24hrs
☑ ms:fires_landsat_7days
☑ ms:fires_landsat_24hrs

```
# Python Console
# Use iface to access QGIS API interface or type '?' for more info
# Security warning: typing commands from an untrusted source can harm your computer
>>> exec(Path('C:/Users/ibnha/AppData/Local/Temp/tmpkrtbzoh_').read_text())
```

```
>>>
```

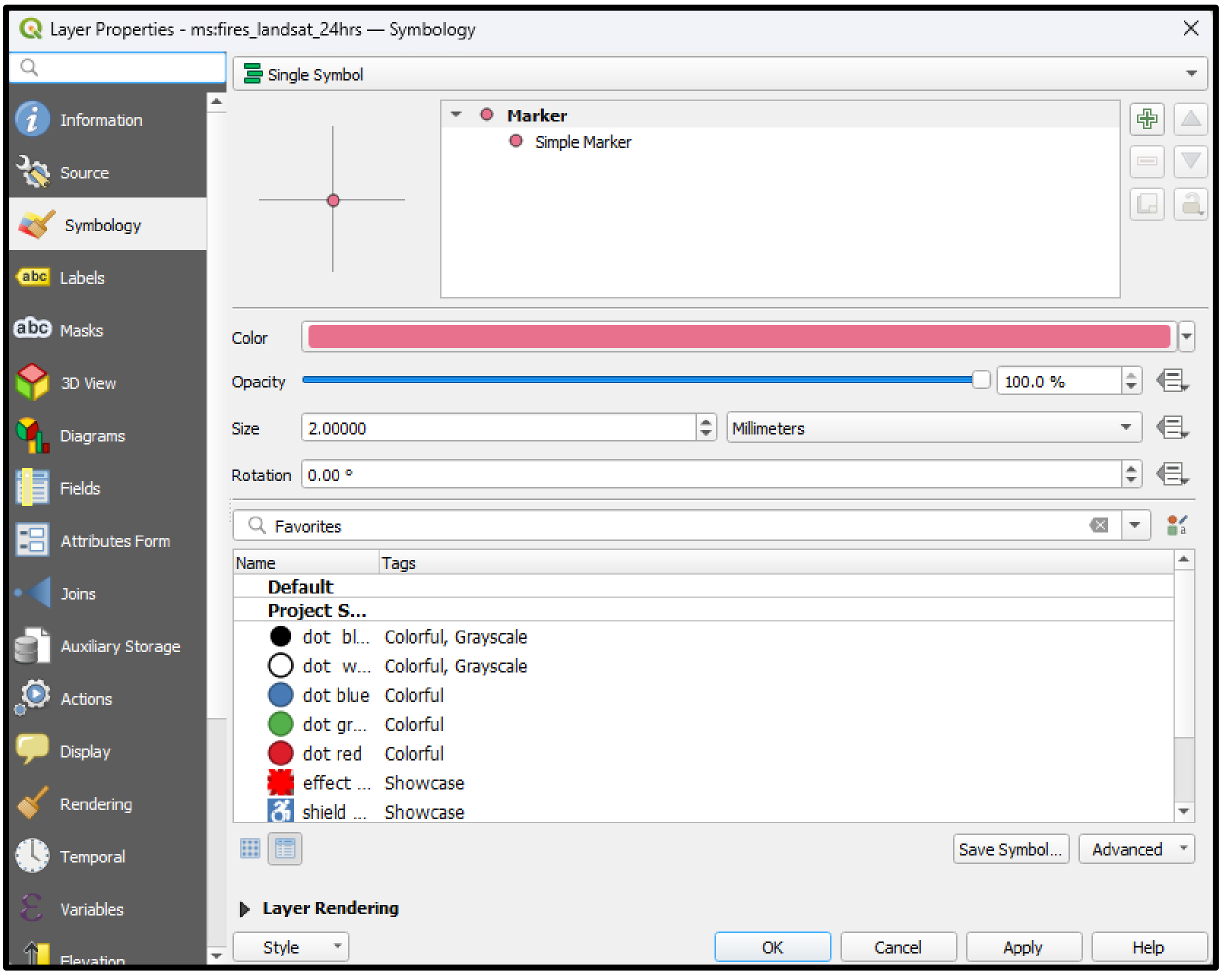
```
active = iface.activeLayer()
iface.showAttributeTable(active)
iface.showLayerProperties(active)
```



ms:fires_landsat_24hrs — Features Total: 3990, Filtered: 3823, Selected: 0

| | id | metadataproperty | description_href | description_title | lescription_nilreasor | description | :criptionreference_h | :criptionreference_ti | iptionreference_nilre | dentifier_codespace | identifier | name | location_location |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | fires_landsat_24... | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| 2 | fires_landsat_24... | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| 3 | fires_landsat_24... | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| 4 | fires_landsat_24... | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| 5 | fires_landsat_24... | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| 6 | fires_landsat_24... | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| 7 | fires_landsat_24... | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| 8 | fires_landsat_24... | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| 9 | fires_landsat_24... | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| 10 | fires_landsat_24... | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| 11 | fires_landsat_24... | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| 12 | fires_landsat_24... | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| 13 | fires_landsat_24... | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| 14 | fires_landsat_24... | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| 15 | fires_landsat_24... | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| 16 | fires_landsat_24... | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| 17 | fires_landsat_24... | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| 18 | fires_landsat_24... | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| 19 | fires_landsat_24... | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| 20 | fires_landsat_24... | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

**This is the Attribute Table**



Layer Properties - ms:fires_landsat_24hrs — Symbology

Single Symbol

Marker
Simple Marker

Color
Opacity 100.0 %
Size 2.00000 Millimeters
Rotation 0.00 °

Favorites

| Name | Tags |
|---|---|
| **Default** | |
| **Project S...** | |
| ● dot bl... | Colorful, Grayscale |
| ○ dot w... | Colorful, Grayscale |
| ● dot blue | Colorful |
| ● dot gr... | Colorful |
| ● dot red | Colorful |
| effect ... | Showcase |
| shield ... | Showcase |

**Layer Properties**

**Enter the Data in QGIS as per the image below**

The Data are taken from http://giswebservices.mass- gis.state.ma.us/geoserver/wfs as procedure mentioned above Using QGIS WFS.

But not Working for me so downloaded manually for Tutorial.

I have provided the dataset needed for this tutorial in Github and as link**.**

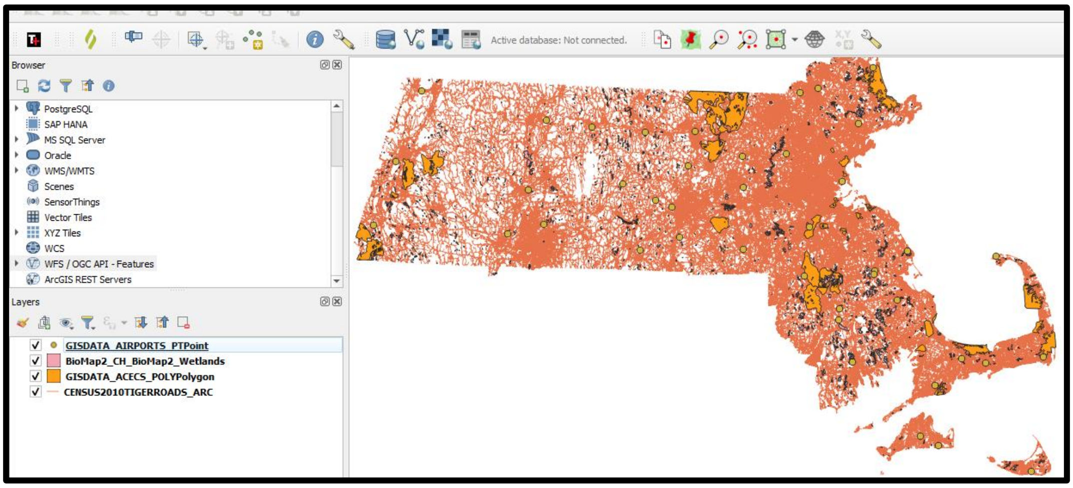
**Data are taken from this link :**

GIS Airport :- Airport Shape File

BioMap2_CH_BioMap2_Wetlands:- BioMap2

GISDATA_ACECS_Polygon :- ACECS  Polygon

CENSUS_2010_TIGER_ROAD:- CENSUS Data



- Iterating the attribute data from Airport data
    - **next() –** this function is to iterate the next attribute from attribute table in Python

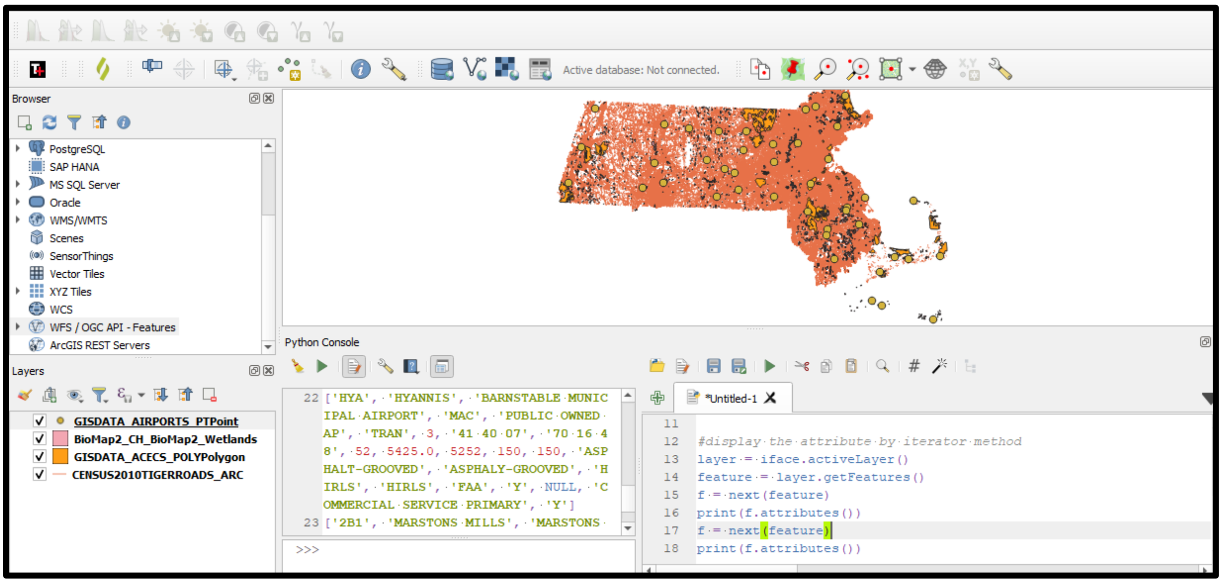
**Python Code:**

```
layer = iface.activeLayer()
features = layer.getFeatures()
f = next(features)
```

**print(f.attributes())**

**f = next(features)**

**print(f.attributes)**



```
['HYA', 'HYANNIS', 'BARNSTABLE MUNICIPAL AIRPORT', 'MAC', 'PUBLIC OWNED AP', 'TRAN', 3, '41 40 07', '70 16 48', 52, 5425.0, 5252, 150, 150, 'ASPHALT-GROOVED', 'ASPHALY-GROOVED', 'HIRLS', 'HIRLS', 'FAA', 'Y', NULL, 'COMMERCIAL SERVICE PRIMARY', 'Y']
['2B1', 'MARSTONS MILLS', 'MARSTONS
>>>
```

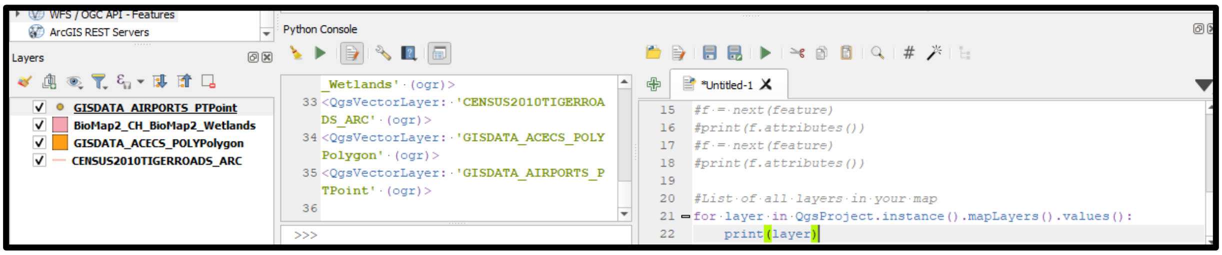
```

#display the attribute by iterator method
layer = iface.activeLayer()
feature = layer.getFeatures()
f = next(feature)
print(f.attributes())
f = next(feature)
print(f.attributes())
```

- To list all layers in map Using Python

Python Code:

**for layer in QgsProject.instance().mapLayers().values():**

**print(layer)**



```
_Wetlands' (ogr)>
<QgsVectorLayer: 'CENSUS2010TIGERROADS_ARC' (ogr)>
<QgsVectorLayer: 'GISDATA_ACECS_POLYPolygon' (ogr)>
<QgsVectorLayer: 'GISDATA_AIRPORTS_PTPoint' (ogr)>

>>>
```

```
#f = next(feature)
#print(f.attributes())
#f = next(feature)
#print(f.attributes())

#List of all layers in your map
for layer in QgsProject.instance().mapLayers().values():
    print(layer)
```

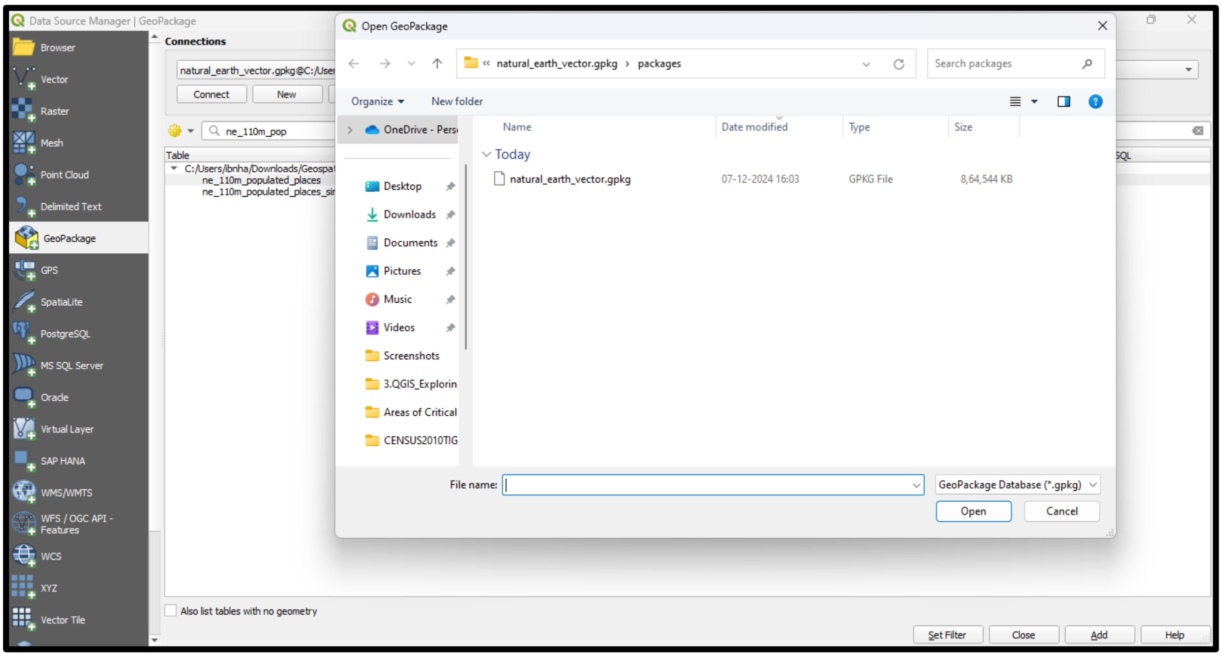
# Using Processing Algorithms in Python Console

**Question: which cities are located along the Amazonas River?**

**Dataset: Download [GeoPackage](GeoPackage) (Click link)**

**We are going to find the Answer using Processing Algorithm in Python (PyQGIS)**

- Download: **natural_earth_vector.gpkg**
- We are going to work on Processing Algorithms
    - This workflow will use three Processing Algorithms namely:
        - **native:extractbyexpression**
        - **native:buffer**
        - **native:extractbylocation**
- Open QGIS Go to -> Data Source -> Click on Geopackage and then -> Click new -> Choose the **natural_earth_vector.gpkg**
- Then Click on Connect (shows all vector Data).
- Select **ne_110m_populated_places** and **ne_10m_rivers_lake_centerlines**



natural_earth_vector.gpkg@C:/User

ne_110m_pop

C:/Users/ibnha/Downloads/Geospat
ne_110m_populated_places
ne_110m_populated_places_si

« natural_earth_vector.gpkg › packages

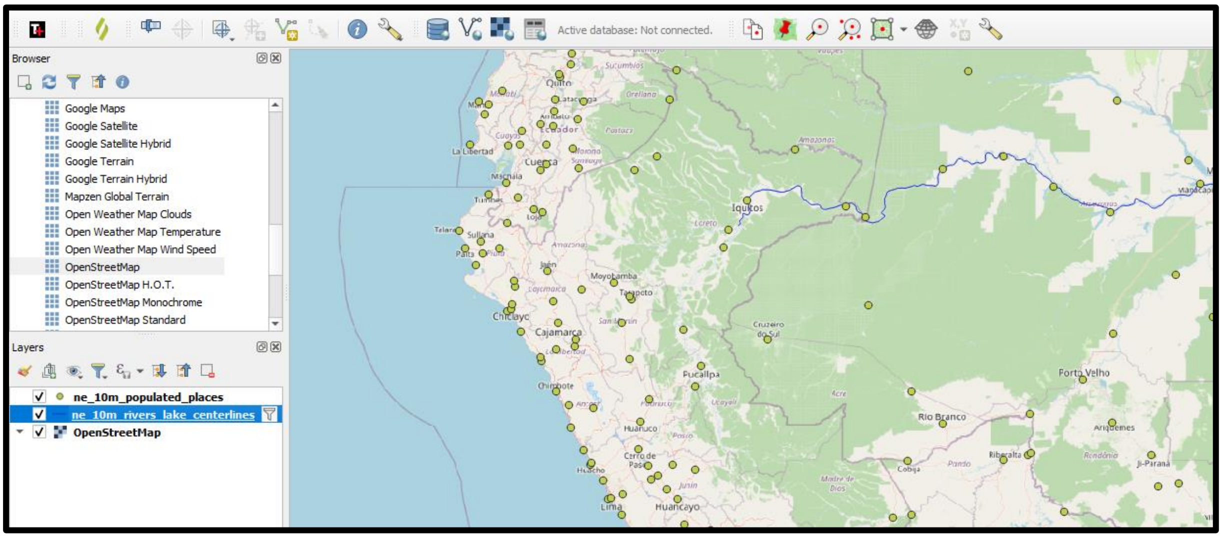
Search packages

| Name | Date modified | Type | Size |
|---|---|---|---|
| natural_earth_vector.gpkg | 07-12-2024 16:03 | GPKG File | 8,64,544 KB |

File name:

GeoPackage Database (*.gpkg)

□ Also list tables with no geometry



- Import the library to work on Processing Algorithms as below

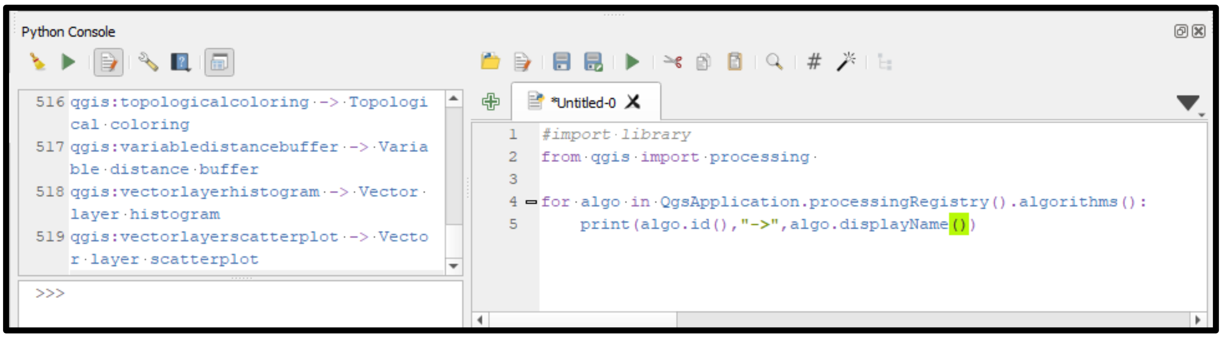
Python Code:

**from qgis import processing**

- While working in Processing Algorithms we need to call algorithms by name so, that they can execute reliably.
- We can list the algorithms by the Python console (QGIS) as follows:

Python Code:

```
for alg in QgsApplication.processingRegistry(). algorithms ():
    print (alg.id (), "->",alg.displayName())
```



```
qgis:topologicalcoloring -> Topological coloring
qgis:variabledistancebuffer -> Variable distance buffer
qgis:vectorlayerhistogram -> Vector layer histogram
qgis:vectorlayerscatterplot -> Vector layer scatterplot
>>>
```

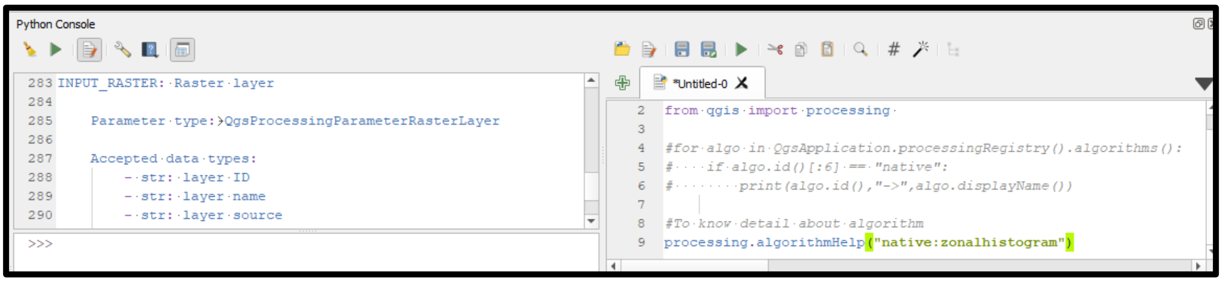
```
#import library
from qgis import processing

for algo in QgsApplication.processingRegistry().algorithms():
    print(algo.id(),"->",algo.displayName())
```

**Image displays the algorithms in qgis**

- If We need for detailed info on Algorithm then need to Use Syntax as below

  Python Code:

  **processing.algorithmHelp(“algorithm.id()”)**



```
INPUT_RASTER: Raster layer

    Parameter type:›QgsProcessingParameterRasterLayer

    Accepted data types:
        - str: layer ID
        - str: layer name
        - str: layer source
>>>
```

```
from qgis import processing

#for algo in QgsApplication.processingRegistry().algorithms():
#    if algo.id()[:6] == "native":
#        print(algo.id(),"->",algo.displayName())

#To know detail about algorithm
processing.algorithmHelp("native:zonalhistogram")
```

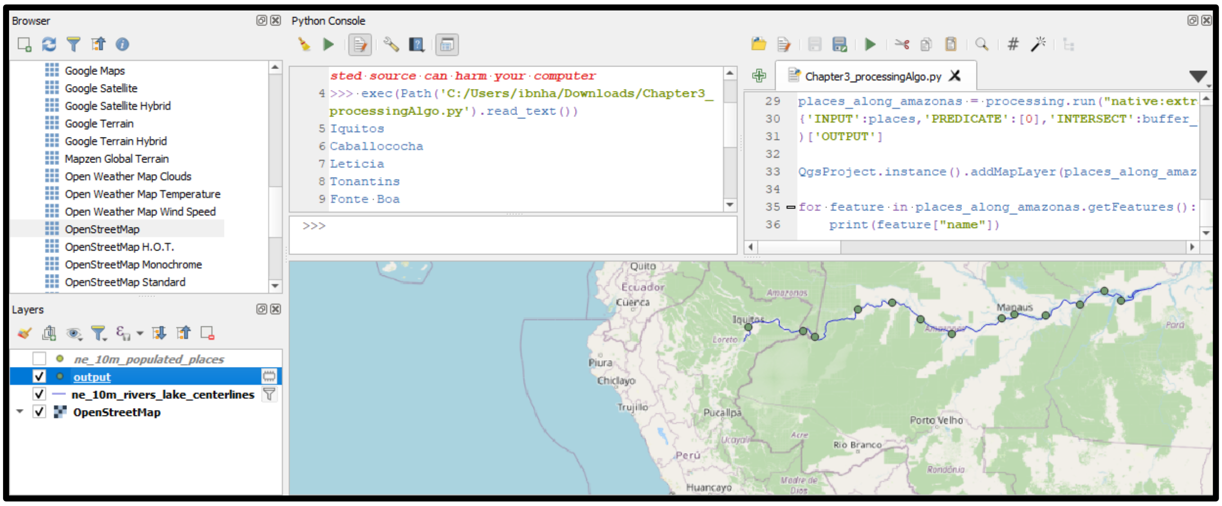
**A detailed input, Output and parameter are explained on the left panel of Python Console.**

## Algorithms Processing: Output the result that Cities are located near Amazonas River

**Python Code:**

```
#Data
my_pakg = "C:/Users/ibnha/Downloads/Geospatial_Python/3.QGIS_Exploring PyQGIS and Native Algorithms for Spatial Analytics/natural_earth_vector.gpkg/packages/natural_earth_vector.gpk"
rivers = '{}|layername=ne_10m_rivers_lake_centerlines'.format(my_pakg)
places ='{}|layername=ne_10m_populated_places'.format(my_pakg)

#native:extractbyexpression
expression = "name = 'Amazonas'"
```

```
amazonas = processing.run("native:extractbyexpression",
{'INPUT':rivers,'EXPRESSION':expression,'OUTPUT':'memory:'})['OUTPUT']

#native:buffer
buffer = 0.1 #degrees

buffer_amazonas = processing.run("native:buffer",
{'INPUT':amazonas,'DISTANCE':buffer,'SEGMENTS':5,'END_CAP_STYLE':0,
 'JOIN_STYLE':0,'MITER_LIMIT':2,'DISSOLVE':False,'OUTPUT':'memory:'}
 )['OUTPUT']

places_along_amazonas = processing.run("native:extractbylocation",
{'INPUT':places,'PREDICATE':[0],'INTERSECT':buffer_amazonas,'OUTPUT':'memory:'}
)['OUTPUT']

QgsProject.instance().addMapLayer(places_along_amazonas)

for feature in places_along_amazonas.getFeatures():
    print(feature["name"])
```



```
sted source can harm your computer
>>> exec(Path('C:/Users/ibnha/Downloads/Chapter3_processingAlgo.py').read_text())
Iquitos
Caballococha
Leticia
Tonantins
Fonte Boa
>>>
```

Chapter3_processingAlgo.py

```
places_along_amazonas = processing.run("native:extr
{'INPUT':places,'PREDICATE':[0],'INTERSECT':buffer_
)['OUTPUT']

QgsProject.instance().addMapLayer(places_along_amaz

for feature in places_along_amazonas.getFeatures():
    print(feature["name"])
```

**References:**

1. Data and Tutorials - https://github.com/Haseeb-oss-eng/Geospatial_Python.git
2. All Practical's is based on Book – Python for Geospatial Data Analysis (Author: Bonny P. McClain, Released October 2022, Publisher(s): O'Reilly Media, Inc. ISBN: 9781098104795
3. NASA FIRMS :- https://firms.modaps.eosdis.nasa.gov/mapserver/
4. GIS Web Service - http://giswebservices.mass- gis.state.ma.us/geoserver/wfs