

Find Neighbor Polygons in a Layer

QGIS Tutorials and Tips



Author

Ujaval Gandhi

<http://google.com/+UjavalGandhi>

Translations by

Marina Pavlova Ilya Trofimov Fayçal Fatihi

Introduction to QGIS and Python

QGIS is a free and open source GIS software. It is a powerful tool for working with spatial data. Python is a programming language that can be used to automate tasks in QGIS. This tutorial will show you how to use Python in QGIS. We will start by installing the Python environment. Then we will learn how to use the QGIS Python console. Finally, we will see how to use Python to automate some common QGIS tasks.

Installing the Python Environment

Before we can use Python in QGIS, we need to install the Python environment. This is a simple process that can be done in a few minutes. We will use the Anaconda distribution of Python. Anaconda is a popular choice for data science and GIS applications. It includes a package manager called Conda, which makes it easy to install and manage the libraries we need for QGIS.

Setting up the QGIS Python Environment

Once we have installed Anaconda, we need to set up the QGIS Python environment. This involves installing the QGIS Python bindings. We can do this by running the following command in the terminal:

```
conda install -c conda-forge qgis
```

Using the QGIS Python Console

Now that we have set up the QGIS Python environment, we can use the QGIS Python console. The console is a tool that allows us to run Python code directly within QGIS. It is located in the QGIS interface under the 'Tools' menu. We can use the console to test our code and to automate tasks.

Automating Tasks with Python

1. Open the QGIS Python console. In the console, type the following code to load the `ne_10m_admin_0_countries` layer into the QGIS map canvas:



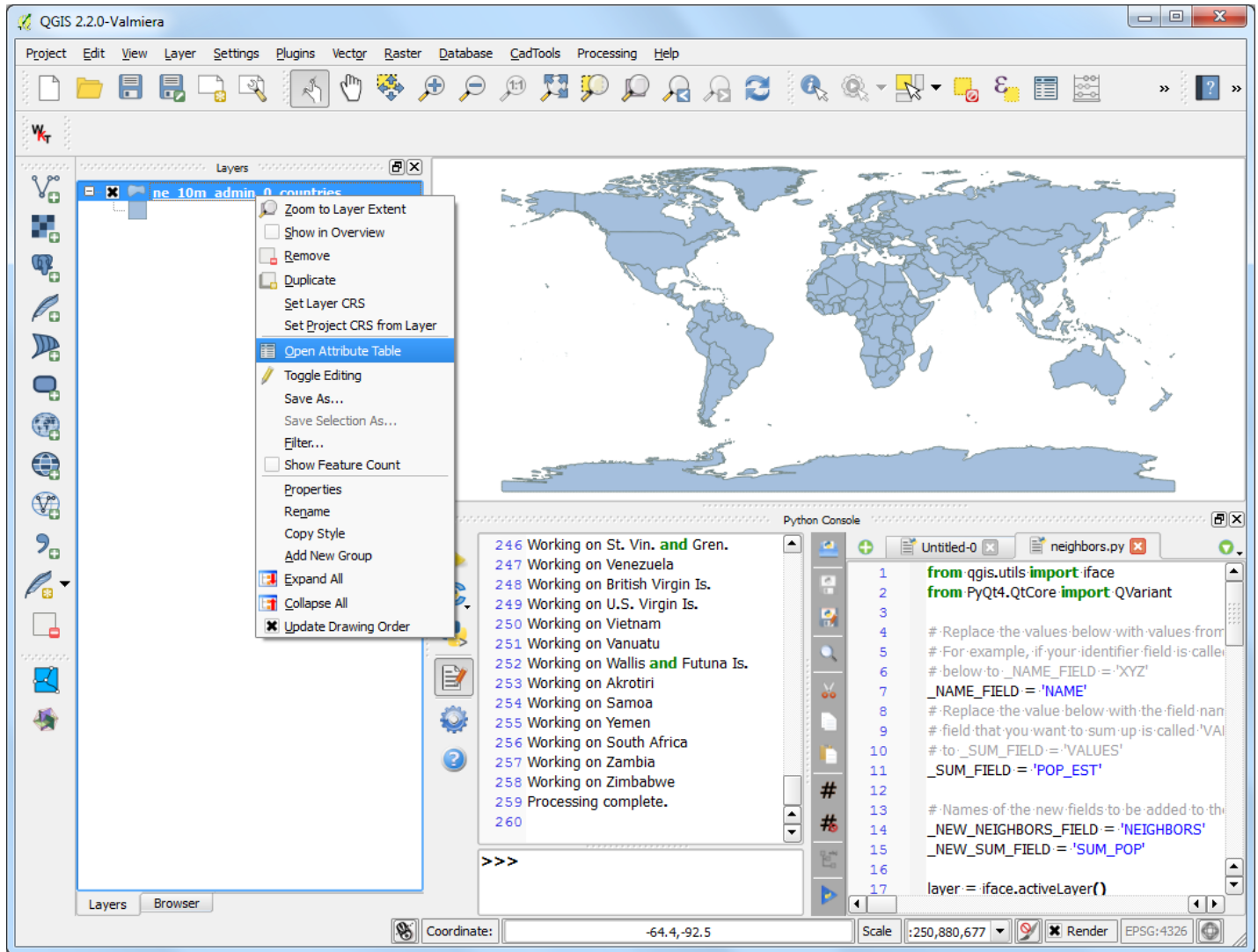
3. ■■■■■■■■■■ ■■■■■ Plugins ■ Python Console.



6. `ne_10m_admin_0_countries`, `ne_10m_admin_0_countries`, `ne_10m_admin_0_countries`
`_NAME_FIELD` `_SUM_FIELD` `ne_10m_admin_0_countries`
`ne_10m_admin_0_countries`, `ne_10m_admin_0_countries`, `ne_10m_admin_0_countries`
`ne_10m_admin_0_countries`: `guilabel: save` `ne_10m_admin_0_countries`: `guilabel: Editor`, `ne_10m_admin_0_countries`
`ne_10m_admin_0_countries`: `guilabel: Run script`, `ne_10m_admin_0_countries`



7. **ne_10m_admin_0_countries** **Open** **Table**.



8. `NEIGHBORS` `SUM.`

Attribute table - ne_10m_admin_0_countries :: Features total: 255, filtered: 255, selected: 0

	ION	REGION_WB	NAME_LEN	LONG_LEN	ABBREV_LEN	TINY	HOMEPART	NEIGHBORS	SUM
0		Latin America & ...	5.00	5.00	5.00	4.00	-99.00	NULL	0
1	sia	South Asia	11.00	11.00	4.00	-99.00	1.00	Iran,Turkmenista...	1621125240
2	a	Sub-Saharan Africa	6.00	6.00	4.00	-99.00	1.00	Namibia,Zambia,...	86676756
3		Latin America & ...	8.00	8.00	4.00	-99.00	-99.00	NULL	0
4	urope	Europe & Central...	7.00	7.00	4.00	-99.00	1.00	Macedonia,Greec...	15281164
5	urope	Europe & Central...	5.00	13.00	5.00	5.00	-99.00	NULL	0
6	urope	Europe & Central...	7.00	7.00	4.00	5.00	1.00	France,Spain	104582794
7	ia	Middle East & No...	20.00	20.00	6.00	-99.00	1.00	Saudi Arabia,Oman	32104718
8	ica	Latin America & ...	9.00	9.00	4.00	-99.00	1.00	Bolivia,Paraguay,...	235606259
9	ia	Europe & Central...	7.00	7.00	4.00	-99.00	1.00	Georgia,Turkey,I...	156089287
10		East Asia & Pacific	14.00	14.00	9.00	3.00	-99.00	NULL	0
11		Antarctica	10.00	10.00	4.00	-99.00	1.00	NULL	0
12	d Ne...	East Asia & Pacific	23.00	27.00	7.00	-99.00	-99.00	NULL	0
13	ope...	Sub-Saharan Africa	22.00	35.00	10.00	2.00	-99.00	NULL	0
14		Latin America & ...	17.00	19.00	6.00	4.00	1.00	NULL	0
15	d Ne...	East Asia & Pacific	9.00	9.00	4.00	-99.00	1.00	NULL	0
16	rope	Europe & Central...	7.00	7.00	5.00	-99.00	1.00	Italy,Hungary,Slo...	175681436
17	ia	Europe & Central...	10.00	10.00	4.00	-99.00	1.00	Georgia,Turkey,R...	290858866
18	ica	Sub-Saharan Africa	7.00	7.00	4.00	-99.00	1.00	Rwanda,Tanzani...	120214356
19	rope	Europe & Central...	7.00	7.00	5.00	-99.00	1.00	France,Netherla...	163595324
20	rica	Sub-Saharan Africa	5.00	5.00	5.00	-99.00	1.00	Nigeria,Niger,Bur...	186301451
21	ica	Sub-Saharan Africa	12.00	12.00	4.00	-99.00	1.00	Mali,Niger,Ghana...	87234511
22	sia	South Asia	10.00	10.00	5.00	-99.00	1.00	India,Myanmar	1214216958

Show All Features

#####

```
#####
# Copyright 2014 Ujaval Gandhi
#
#This program is free software; you can redistribute it and/or
#modify it under the terms of the GNU General Public License
#as published by the Free Software Foundation; either version 2
#of the License, or (at your option) any later version.
#
#This program is distributed in the hope that it will be useful,
#but WITHOUT ANY WARRANTY; without even the implied warranty of
#MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
#GNU General Public License for more details.
#
#You should have received a copy of the GNU General Public License
#along with this program; if not, write to the Free Software
#Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.
#
#####
from qgis.utils import iface
from PyQt4.QtCore import QVariant

# Replace the values below with values from your layer.
# For example, if your identifier field is called 'XYZ', then change the line
```

```

# below to _NAME_FIELD = 'XYZ'
_NAME_FIELD = 'NAME'
# Replace the value below with the field name that you want to sum up.
# For example, if the # field that you want to sum up is called 'VALUES', then
# change the line below to _SUM_FIELD = 'VALUES'
_SUM_FIELD = 'POP_EST'

# Names of the new fields to be added to the layer
_NEW_NEIGHBORS_FIELD = 'NEIGHBORS'
_NEW_SUM_FIELD = 'SUM'

layer = iface.activeLayer()

# Create 2 new fields in the layer that will hold the list of neighbors and sum
# of the chosen field.
layer.startEditing()
layer.dataProvider().addAttributes(
    [QgsField(_NEW_NEIGHBORS_FIELD, QVariant.String),
     QgsField(_NEW_SUM_FIELD, QVariant.Int)])
layer.updateFields()
# Create a dictionary of all features
feature_dict = {f.id(): f for f in layer.getFeatures()}

# Build a spatial index
index = QgsSpatialIndex()
for f in feature_dict.values():
    index.insertFeature(f)

# Loop through all features and find features that touch each feature
for f in feature_dict.values():
    print 'Working on %s' % f[_NAME_FIELD]
    geom = f.geometry()
    # Find all features that intersect the bounding box of the current feature.
    # We use spatial index to find the features intersecting the bounding box
    # of the current feature. This will narrow down the features that we need
    # to check neighboring features.
    intersecting_ids = index.intersects(geom.boundingBox())
    # Initialize neighbors list and sum
    neighbors = []
    neighbors_sum = 0
    for intersecting_id in intersecting_ids:
        # Look up the feature from the dictionary
        intersecting_f = feature_dict[intersecting_id]

        # For our purpose we consider a feature as 'neighbor' if it touches or
        # intersects a feature. We use the 'disjoint' predicate to satisfy
        # these conditions. So if a feature is not disjoint, it is a neighbor.
        if (f != intersecting_f and
            not intersecting_f.geometry().disjoint(geom)):
            neighbors.append(intersecting_f[_NAME_FIELD])
            neighbors_sum += intersecting_f[_SUM_FIELD]
    f[_NEW_NEIGHBORS_FIELD] = ','.join(neighbors)
    f[_NEW_SUM_FIELD] = neighbors_sum
    # Update the layer with new attribute values.
    layer.updateFeature(f)

```

```
layer.commitChanges()  
print 'Processing complete.'
```