

# Find Neighbor Polygons in a Layer

QGIS Tutorials and Tips



Author

Ujaval Gandhi

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

Translations by

SongHyun Choi

□□ □□□ □□

□□□□ □ □□□ □□ □□□ □□ □□ □□ □□ □□ □□. QGIS□□ □□ □□ □□□□ □□□□ □ □□ □ □□ □ □□ □□ □□ □ □□□. □□ □□ □□□□ □□□□ □ □□□ □□ □□□□ □□ □□□ □□ □ □□ □□ □□ □□ □□ □□□□ □□ □ □□□. □□□□ □□□□ □□ □□ □□ □□□□□□ □□ □□ □□□□.

□□ □□

□□ □□□□ □□□□ □□□ □□ □□ □□ □□□ □□□ □□ □□ □□ □□ □□□. □□ □□ □□ □□ □□□ □□□□ □□.

□□□ □□

Natural Earth□ [Admin 0 – Countries](#) □□□□ □□ □□□.

Download the [Admin 0 – countries shapefile](#)..

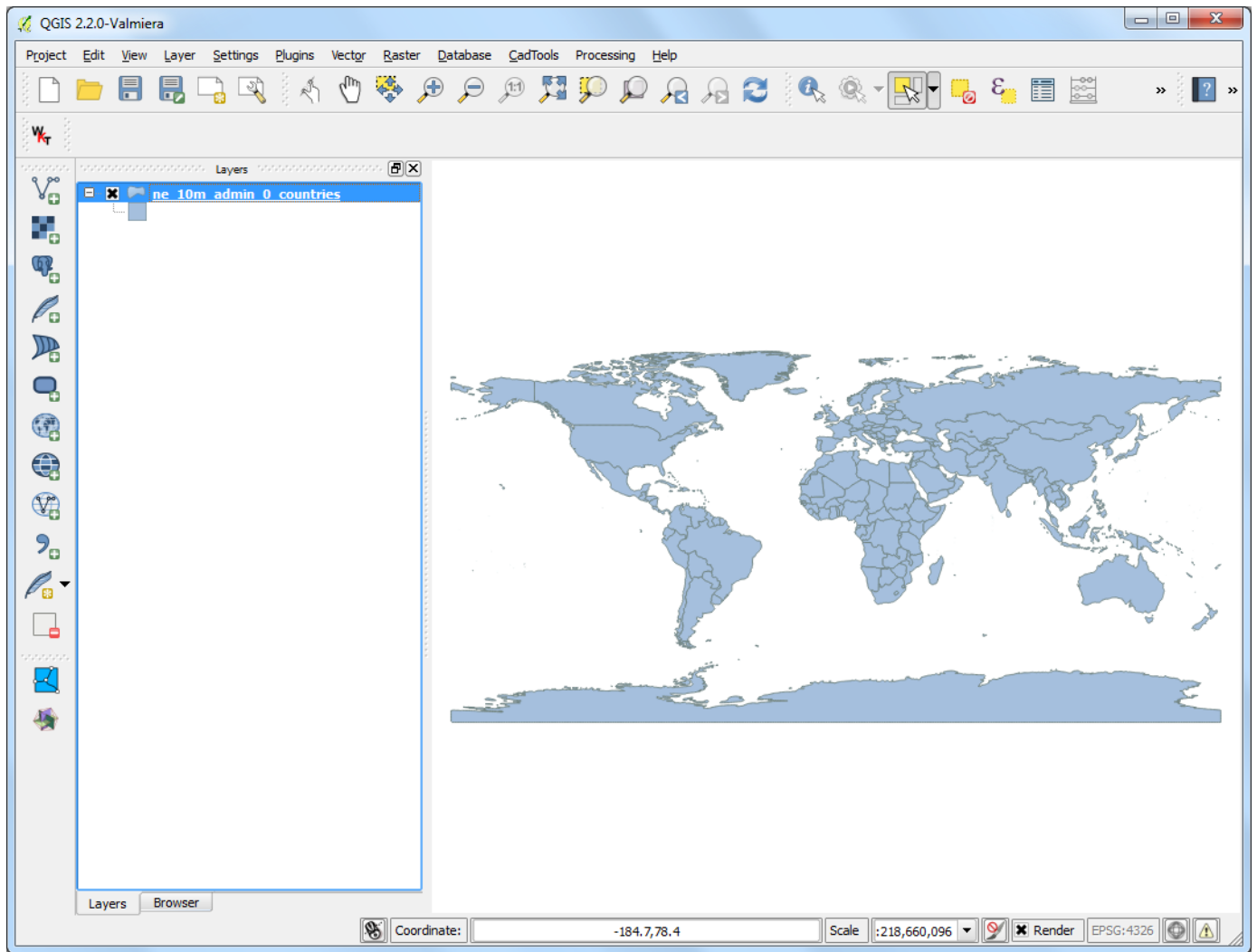
□□□ □□: [NATURALEARTH]

□□□□ □□

Download the *neighbors.py script* and save it to your disk.

□□

1. □□ □□□ --> □□ □□□ □□ `ne_10m_admin_0_countries` `` □□ □□  
`ne_10m_admin_0_countries` `` □□□□ □□□□.



2.       2      . name      .   Identify  
         .      .   name   **\*\*NAME\*\***   
          **\*\*POP\_EST\*\***  .



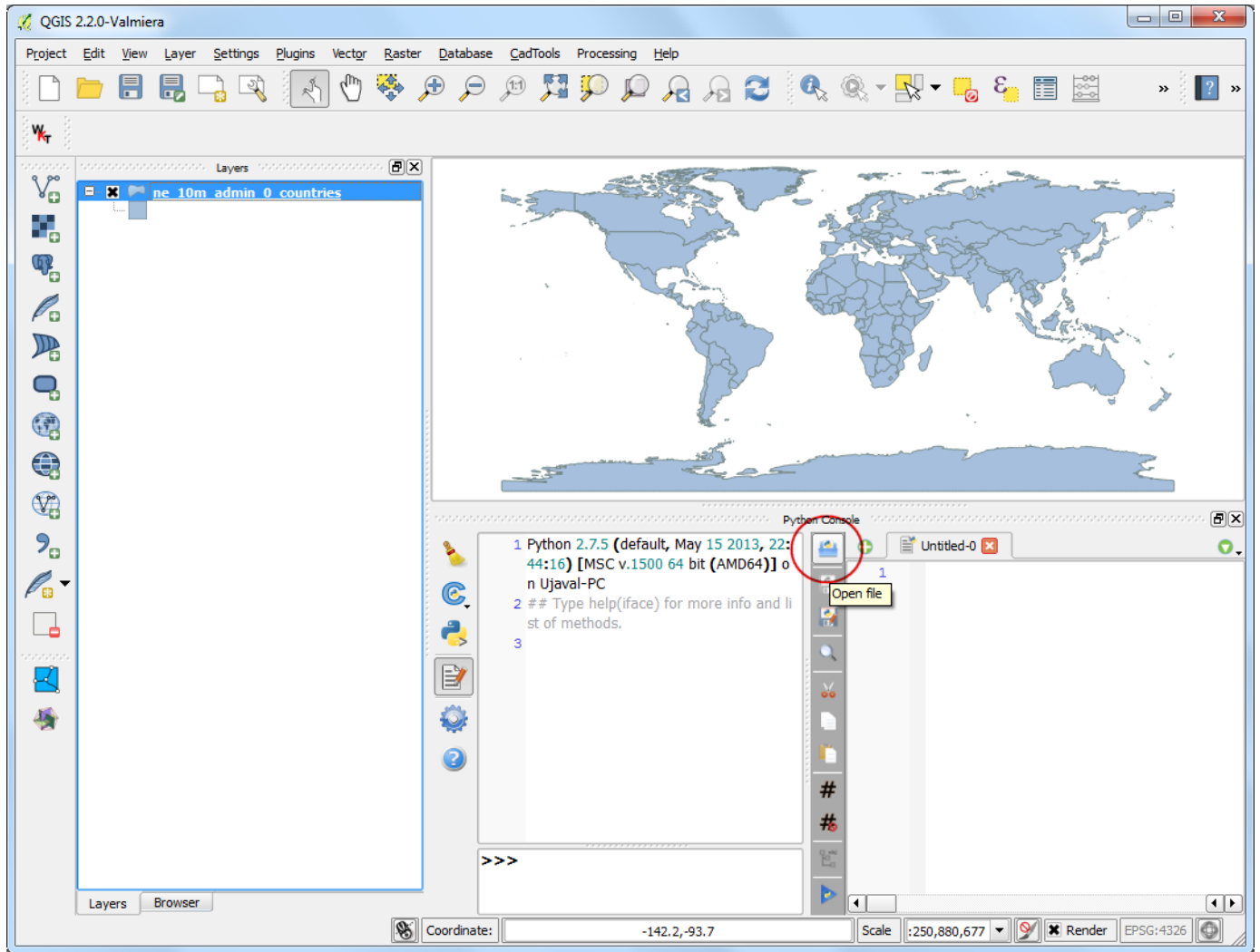
3. □□ □□□□ --> Phtyon □□ :menuselection: `Plugins --> Python Console` □ □□□□.



4. Phthon □□ □□□ □□□ □□□ Show Editor □□□ □□□□.



5. `Editor`` `Open file` `neighbors.py`  
`Open``



6. Open the file explorer, click on the file named `ne_10m_admin_0_countries`. In the Python Console, click on the file explorer icon (circled in red in the screenshot) to open the file. In the Python Console, click on the 'Run script' button (labeled 'Run script' in the screenshot).



7. `ne_10m_admin_0_countries` `:guilabel: 'Open Attribute Table'`.





8. You will notice 2 new attributes called **NEIGHBORS** and **SUM**. These were added by the script.

Attribute table - ne\_10m\_admin\_0\_countries :: Features total: 255, filtered: 255, selected: 0

ID	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	Asia	11.00	11.00	4.00	-99.00	1.00	Iran,Turkmenista...	1621125240
2	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	Europe & Central...	7.00	7.00	4.00	-99.00	1.00	Macedonia,Greec...	15281164
5	Europe & Central...	5.00	13.00	5.00	5.00	-99.00	NULL	0
6	Europe & Central...	7.00	7.00	4.00	5.00	1.00	France,Spain	104582794
7	Middle East & No...	20.00	20.00	6.00	-99.00	1.00	Saudi Arabia,Oman	32104718
8	Latin America & ...	9.00	9.00	4.00	-99.00	1.00	Bolivia,Paraguay,...	235606259
9	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	East Asia & Pacific	23.00	27.00	7.00	-99.00	-99.00	NULL	0
13	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	East Asia & Pacific	9.00	9.00	4.00	-99.00	1.00	NULL	0
16	Europe & Central...	7.00	7.00	5.00	-99.00	1.00	Italy,Hungary,Slo...	175681436
17	Europe & Central...	10.00	10.00	4.00	-99.00	1.00	Georgia,Turkey,R...	290858866
18	Sub-Saharan Africa	7.00	7.00	4.00	-99.00	1.00	Rwanda,Tanzani...	120214356
19	Europe & Central...	7.00	7.00	5.00	-99.00	1.00	France,Netherla...	163595324
20	Sub-Saharan Africa	5.00	5.00	5.00	-99.00	1.00	Nigeria,Niger,Bur...	186301451
21	Sub-Saharan Africa	12.00	12.00	4.00	-99.00	1.00	Mali,Niger,Ghana...	87234511
22	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.'

```