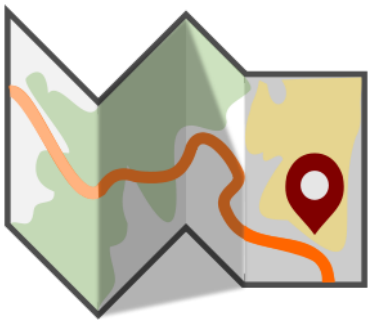
Găsirea poligoanelor învecinate dintr-un strat

QGIS Tutorials and Tips



Author

Ujaval Gandhi

http://google.com/+UjavalGandhi

Translations by

Sorin Călinică

Găsirea poligoanelor învecinate dintr-un strat

Există unele cazuri de utilizare, când se dorește găsirea tuturor poligoanelor învecinate, pentru fiecare dintre poligoanele dintr-un strat. Cu un mic script în Python, putem realiza acest lucru și multe altele, în QGIS. Aici este un exemplu de script, pe care îl puteți folosi pentru a găsi toate poligoane care au granița comună, pentru fiecare dintre poligoanele stratului și, de asemenea, să adăugați numele lor la tabela de atribute. De asemenea, ca un bonus, script-ul însumează un atribut, la alegere, din toate poligoanele învecinate.

Privire de ansamblu asupra activității

Pentru a demonstra modul în care funcționează script-ul, vom folosi un strat poligonal de țări și vom găsi țările care împart aceeași frontieră. De asemenea, ne dorim să calculăm totalul populației din țările învecinate.

Obținerea datelor

Vom folosi setul de date Admin 0 – Țări de la Natural Earth.

Vom folosi setul de date Admin 0 – fișierul shape al țărilor..

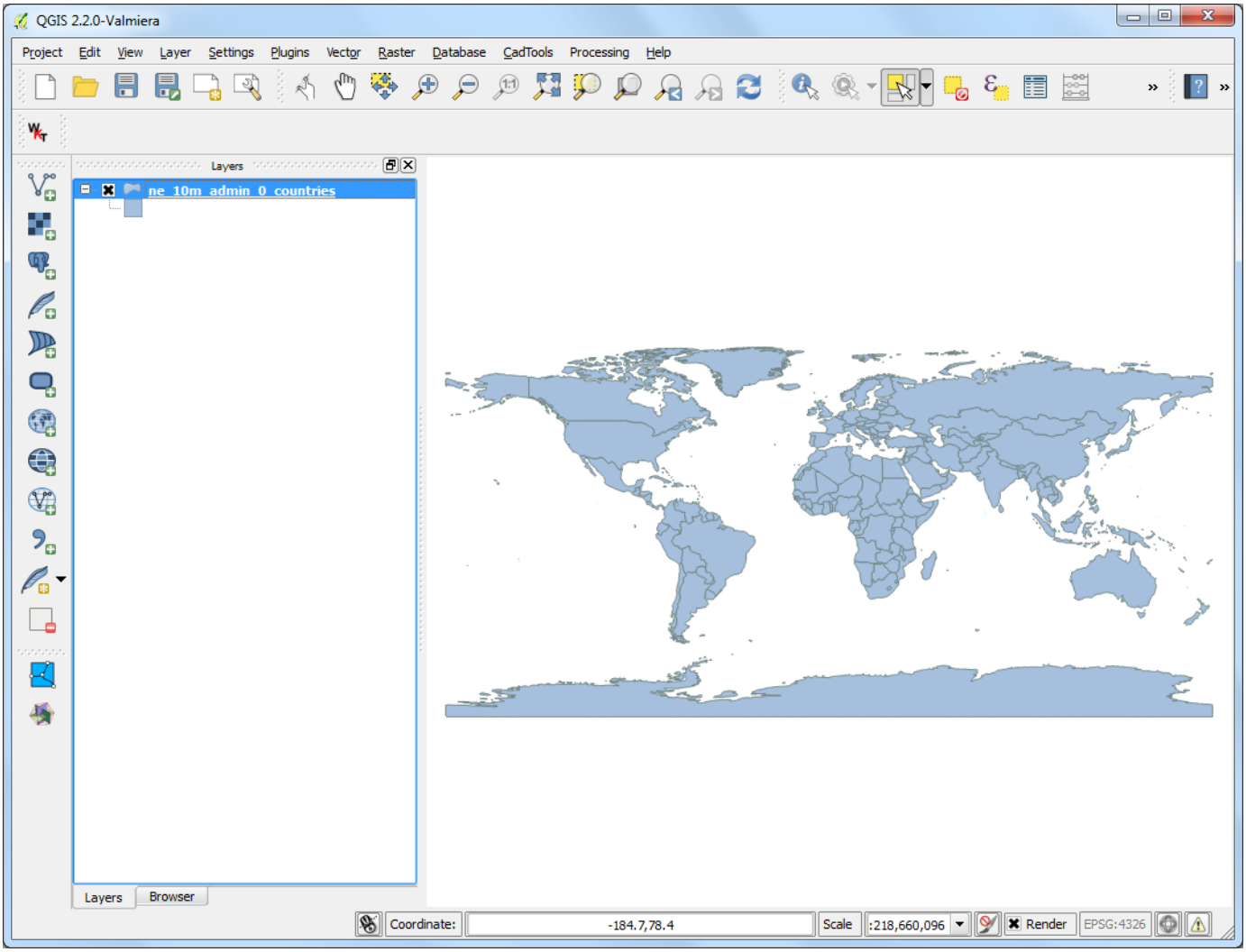
Sursa de date [NATURALEARTH]

Obținerea script-ului

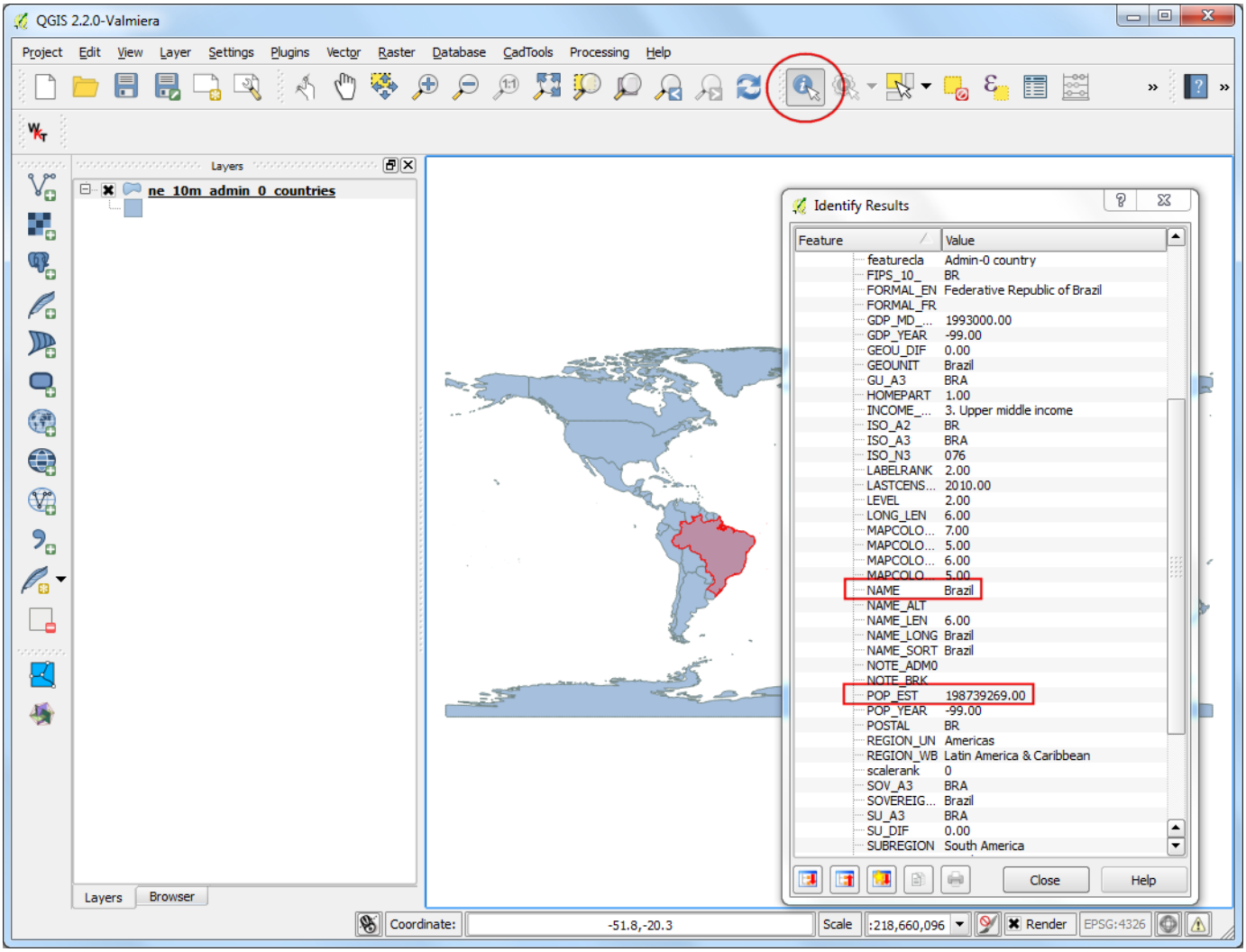
Descărcați script-ul ***neighbors.py*** și salvați-l pe discul dumneavoastră.

Procedura

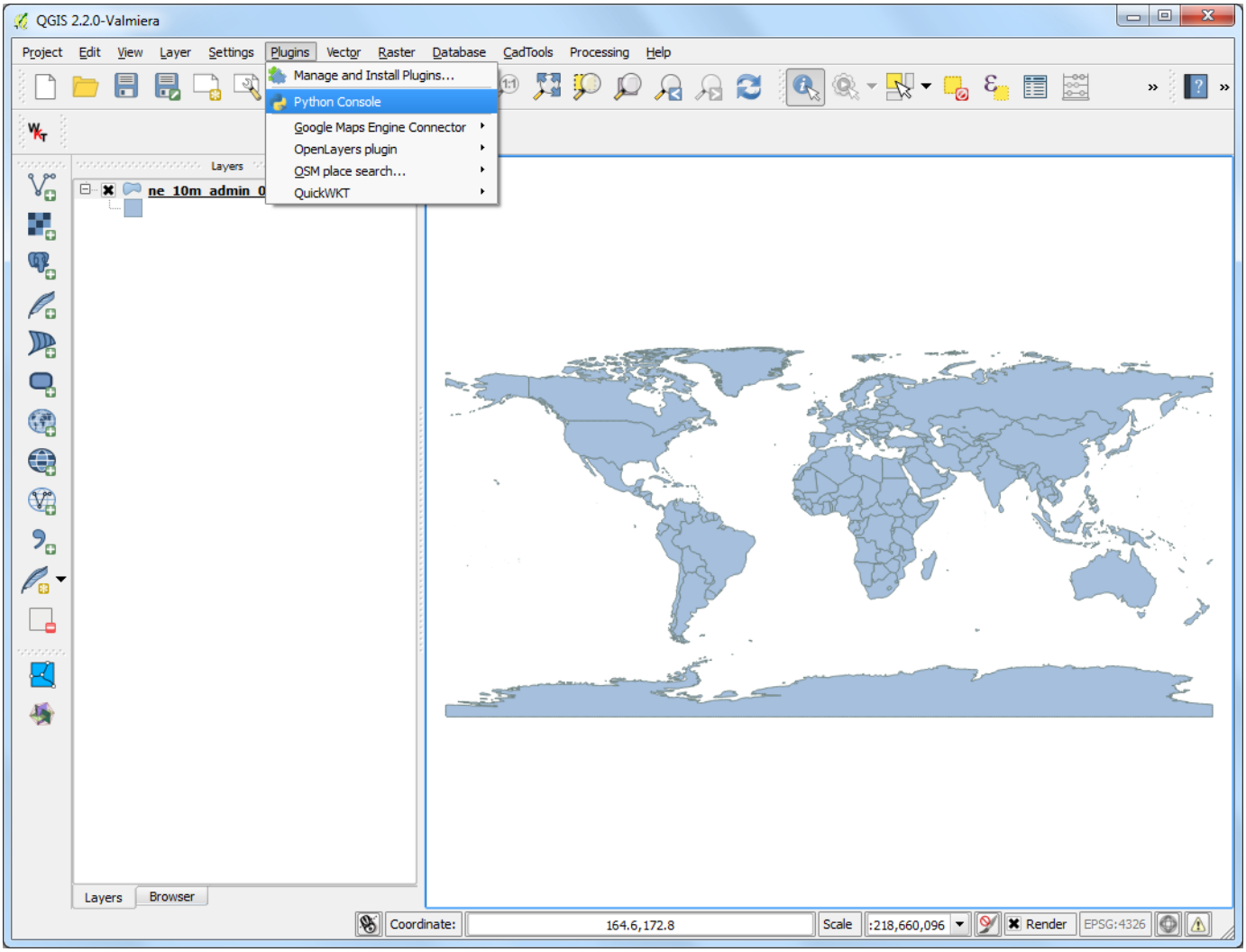
1. Încărcați stratul ***ne_10m_admin_0_countries*** mergând la Layer ‣ Add Vector Layer.



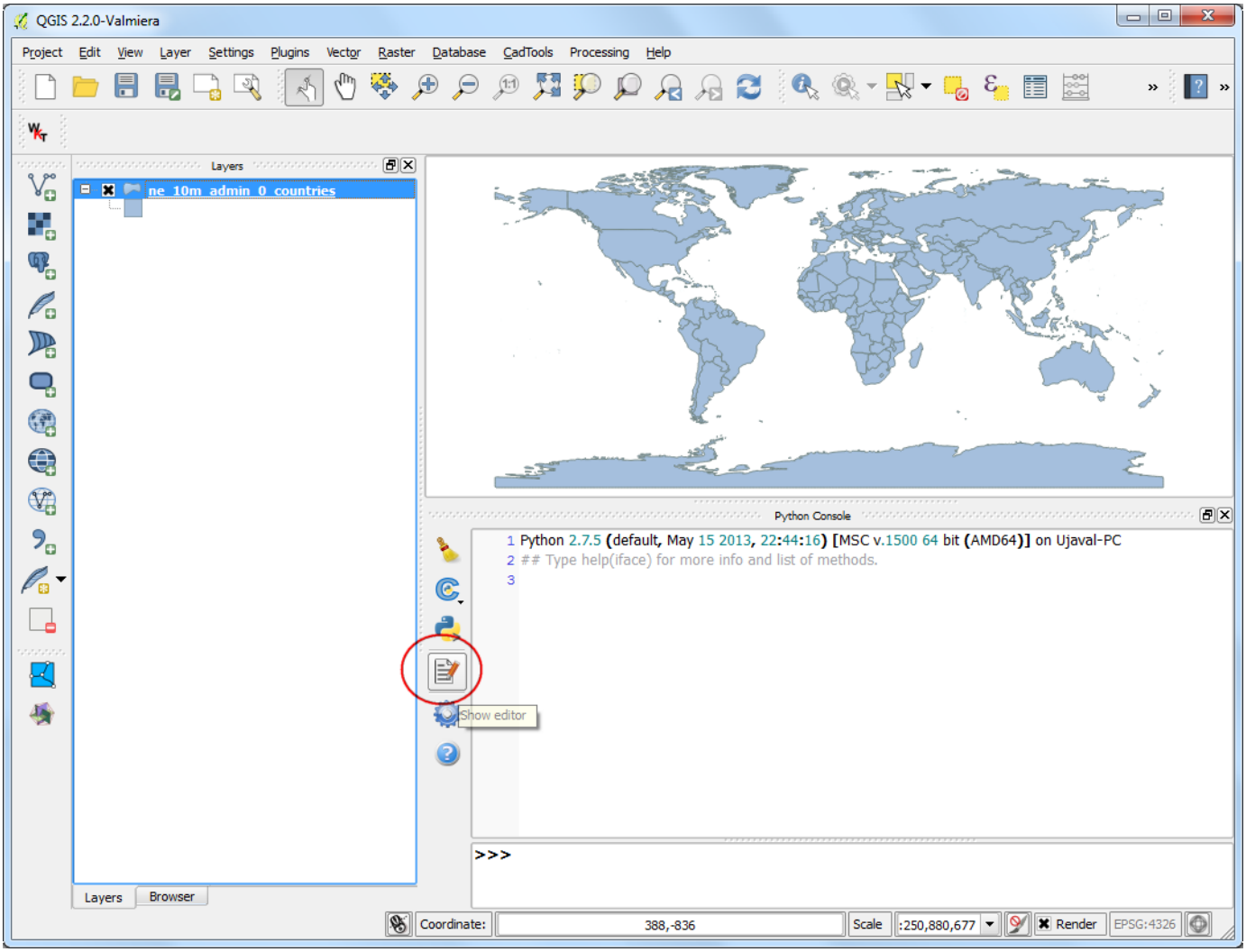
2. Script-ul folosește 2 câmpuri pentru a efectua acțiunea. Un câmp pentru nume și un câmp pe care doriți să-l însumați. Utilizați instrumentul Identify pentru a face clic pe orice element și să examinați atributele. În acest caz, câmpul pentru nume este NAME, și vrem să însumăm estimările de populație din câmpul POP_EST.



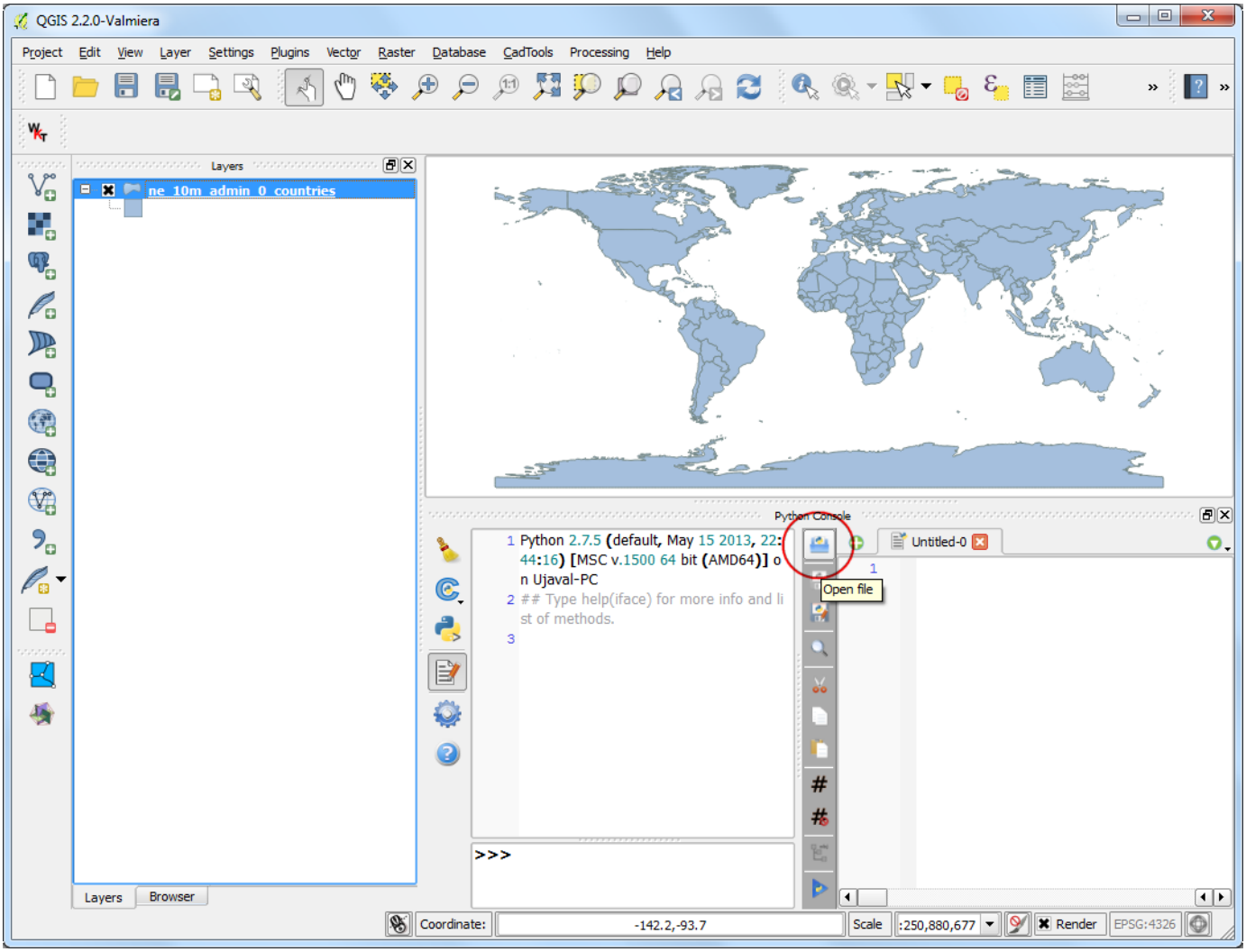
3. Mergeți la Plugins › Python Console.



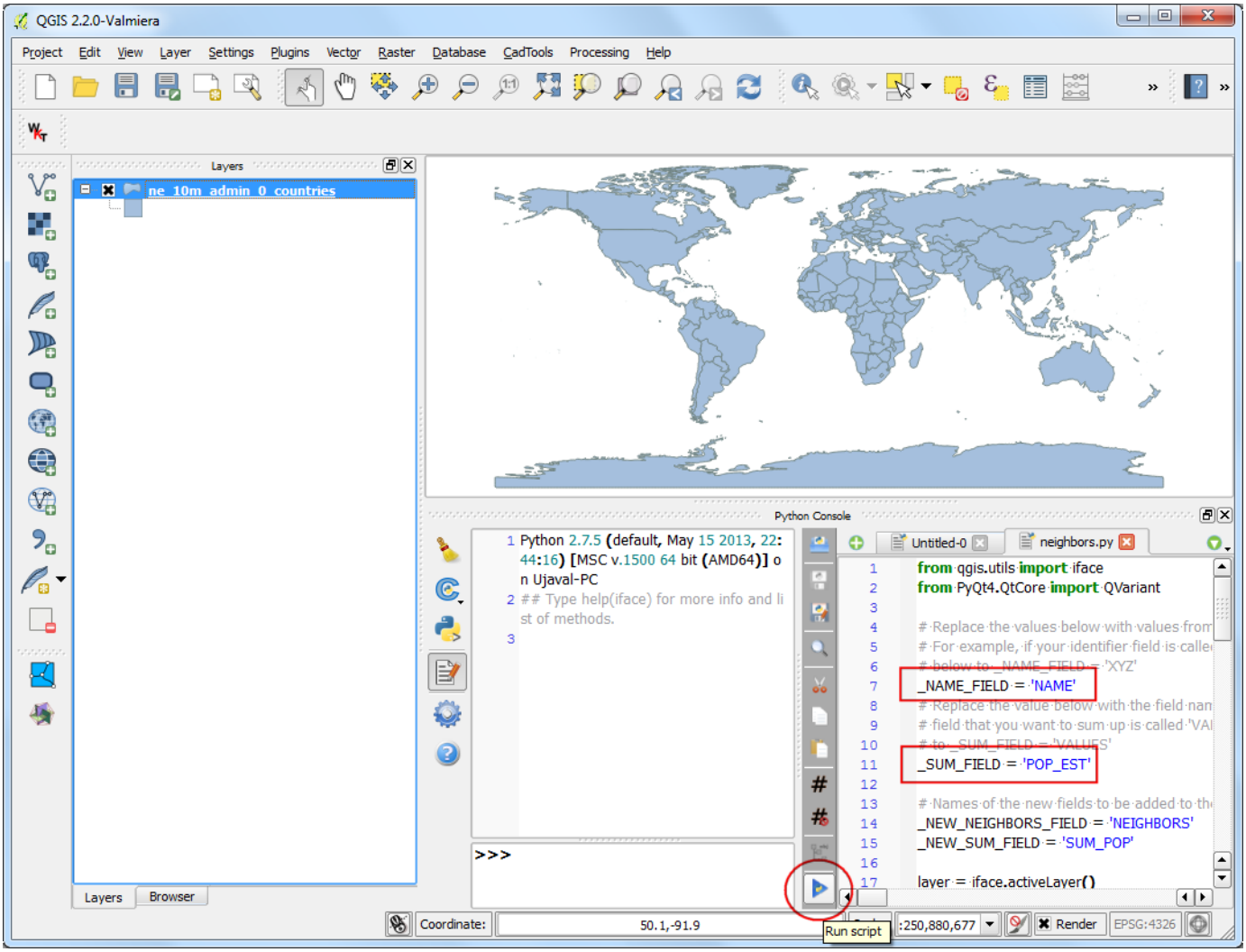
4. În fereastra Python Console, faceți clic pe butonul Show Editor.



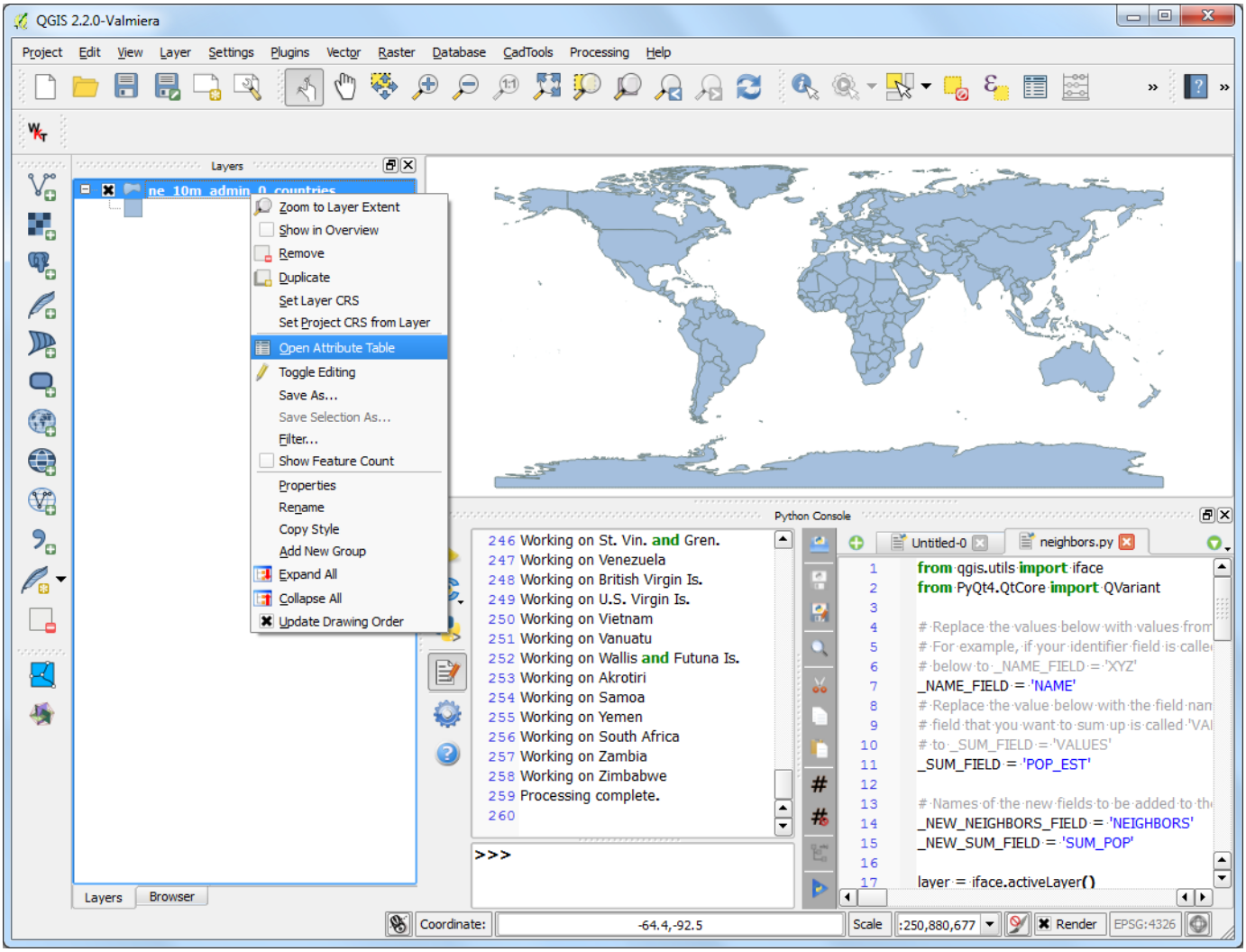
5. În panoul Editor, faceți clic pe butonul Open file, navigați la script-ul descărcat ***neighbors.py*** și apăsați Open.



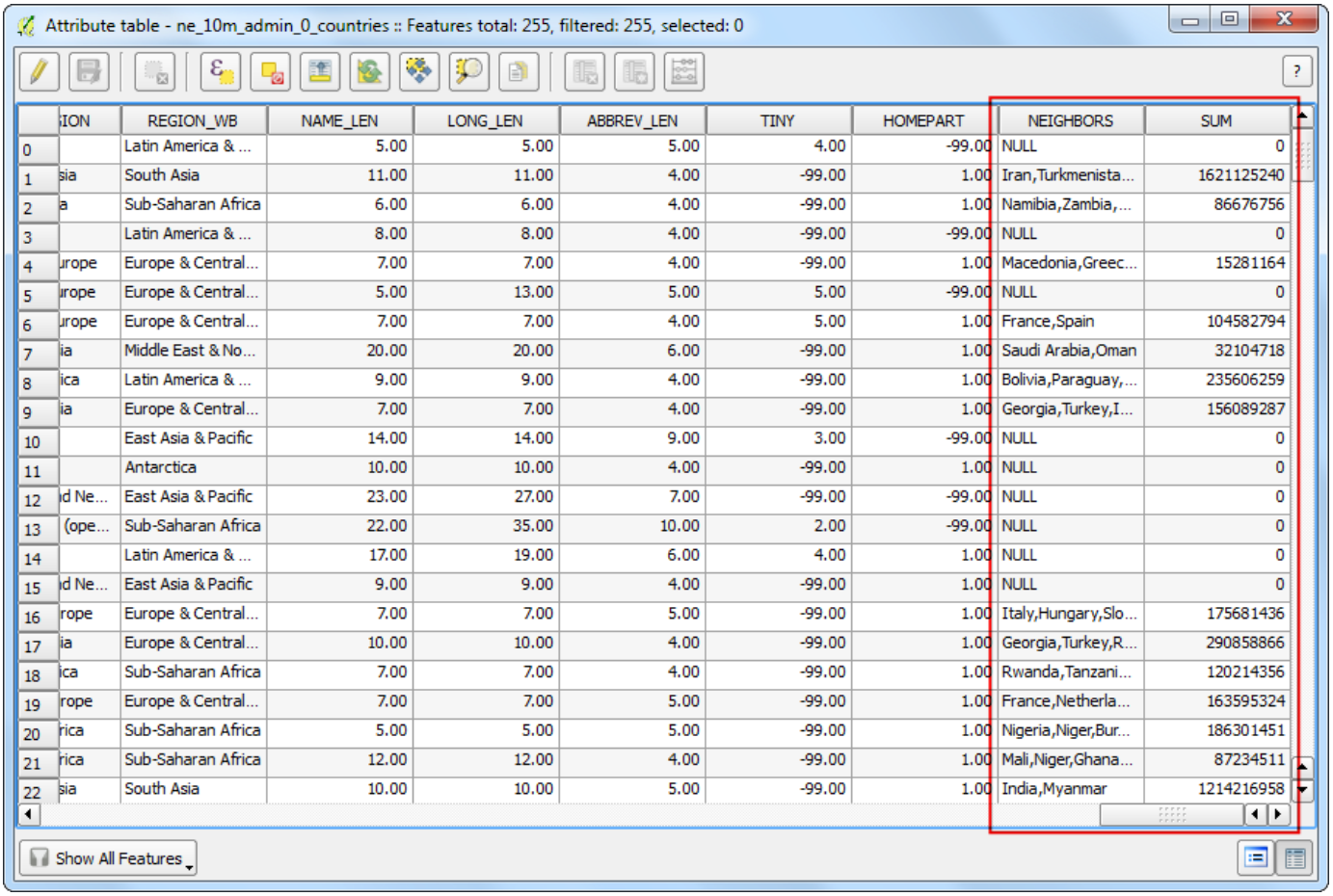
6. O dată ce script-ul este încărcat, este posibil să doriți să schimbați valorile ***_NAME_FIELD*** și ***_SUM_FIELD*** pentru a se potrivi cu atributele din propriul strat. Dacă lucrați cu stratul ***ne_10m_admin_0_countries***, le puteți lăsa așa cum sunt. Faceți clic pe butonul Save din panoul Editor dacă ați făcut vreo schimbare. Apoi apăsați butonul Run script pentru a rula script-ul.



7. O dată ce script-ul se încheie, faceți clic-dreapta pe stratul ***ne_10m_admin_0_countries*** și selectați Open Attribute Table.



8. Veți observa 2 noi atribute, numite ***NEIGHBORS*** și ***SUM***. Acestea au fost adăugate de către script.



	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	rica	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

Pentru referință, mai jos este script-ul complet. Îl puteți modifica pentru a se potrivi cerințelor dumneavoastră.

```
##############################################################################
# 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())
    # Initalize 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.'
```