Πολυτεχνείο Κρήτης

Γεωγραφικά Συστήματα Πληροφοριών

Διδάσκων: Παναγιώτης Παρτσινέβελος, Αν. Καθηγητής

Εργαστήριο: Γεώργιος Πετράκης, Υπ. Διδάκτωρ

Python & GIS

Τι θα διδαχθείτε στο εργαστήριο Python & GIS:

- Γιατί να μάθω Python ;
- Οδηγίες φόρτωσης προγραμματιστικού περιβάλλοντος
- Συνοπτική εισαγωγή στη Python
- GIS Python Βιβλιοθήκες
- Επεξεργασία γεωχωρικών δεδομένων με Python

Περιγραφή εργαστηρίου Python & GIS

Το εργαστήριο αυτό δημιουργήθηκε για να σας εισάγει στη γλώσσα Python με προσανατολισμό τις GIS εφαρμογές, τα χωρικά προβλήματα και τις γεω-επιστήμες γενικότερα.

Γιατί να μάθω Python ;

- Είναι η πιο δημοφιλής γλώσσα προγραμματισμού στις μέρες μας
- Είναι εύκολη στην εκμάθηση της
- Είναι ανοιχτού κώδικα
- Έχει μεγάλη και ενεργή κοινότητα
- Υπάρχει μεγάλη πληθώρα βιβλιοθηκών και τεχνολογιών βασισμένες στην Python
- Καλύπτει μεγάλο εύρος εφαρμογών
- Χρησιμοποιείται ιδιαίτερα στις γεω-επιστήμες
- Αποτελεί ιδανική εναλλακτική για εφαρμογή στον επιστημονικό προγραμματισμό

Οδηγίες φόρτωσης προγραμματιστικού περιβάλλοντος

Για να πειραματιστείτε με την ύλη που θα διδαχθείτε στο εργαστήριο της Python & GIS, θα πρέπει να φορτώσετε ένα Python περιβάλλον στο οποίο θα εγκατασταθούν όλα όσα θα χρησιμοποιήσετε. Η κύρια πλατφόρμα που θα χρησιμοποιήσετε για το εργαστήριο αυτό είναι το Jupyter notebook (https://jupyter.org/), μία πλατφόρμα προγραμματισμού που προσφέρει πολλές δυνατότητες στους προγραμματιστές. Για να ενεργοποιηθεί το περιβάλλον αυτό, υπάρχουν δύο τρόποι που μπορείτε να ακολουθήσετε:

1ος τρόπος: Cloud setup

Ανοίξτε έναν browser (οποιονδήποτε εκτός από τον Internet Explorer) και εισάγετε την παρακάτω διεύθυνση: https://mybinder.org/v2/gh/SenseLabTUC/pythonGISCourse/master

Στη συνέχεια περιμένετε μερικά λεπτά όσο θα φορτώνετε το περιβάλλον. Όταν τελειώσει θα σας εισάγει αυτόματα στο περιβάλλον του μαθήματος.

Προσοχή !

Ο χώρος αυτός είναι ένας προσωρινός χώρος που δημιουργείτε για κάθε χρήστη στο cloud. Πριν βγείτε από το περιβάλλον αυτό (ή κλείσετε το browser που δουλεύετε) πρέπει να κατεβάσετε το αρχείο αυτό επιλέγοντας από το menu: File --> Download as --> Notebook (.ipynb). Διαφορετικά ότι έχετε κάνει θα χαθεί !
Όταν θέλετε να συνεχίσετε την εργασία θα εισάγετε ξανά το παραπάνω link και θα ανεβάσετε επιλέγοντας το πλήκτρο "upload", το αρχείο .ipynb που κατεβάσατε στο προηγούμενο βήμα.

2ος τρόπος: Local setup

Στο local setup θα χρειαστεί να ακολουθήσετε κάποια βήματα στον εικονικό υπολογιστή του μηχανογραφικού με το λειτουργικό Ubuntu linux. Ο τρόπος αυτός είναι πιο χρονοβόρος από τον 1ο, αλλά θα σας βοηθήσει να εξεικοιωθείτε με το Ubuntu περιβάλλον στο οποίο θα γίνουν τα επόμενα εργαστήρια για το QGIS. Για να φορτωθεί το περιβάλλον τοπικά στον εικονικό σας υπολογιστή, ανοίξτε το terminal πληκτρολογώντας Ctrl - Alt - t και εισάγετε τις παρακάτω εντολές:

1. git clone https://github.com/SenseLabTUC/pythonGISCourse
2. cd pythonGISCourse
3. sh pyenvSetup1.sh
4. conda init bash
5. source ~/.bashrc
6. conda activate pythonGIS
7. sh pyenvSetup2.sh

Προσοχή !

Η παραπάνω διαδικασία θα χρειαστεί να επαναλαμβάνετε ΚΑΘΕ φορά που μπαίνετε στον εικονικό σας υπολογιστή (Ubuntu) καθώς λόγω της πολιτικής του Μηχανογραφικού κέντρου οτιδήποτε εγκαθιστάτε, σβήνετε μετά το logout.
\ Επισημαίνεται επίσης ότι όλα τα αρχεία που θέλετε να παραμείνουν στον υπολογιστή σας μετά το logout, πρέπει να βρίσκονται στο φάκελο Documents. Διαφορετικά θα χαθούν !

In [3]:
```
import this
```
```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Τα βασικά της Python

Αριθμητικές πράξεις, Μεταβλητές και Συναρτήσεις

Η Python, μπορεί να χρησιμοποιηθεί σαν απλή αριθμητική μηχανή (calculator):

In [4]:
```
1 + 1
```
Out[4]: 2

In [5]:
```
4 * 4
```
Out[5]: 16

Συναρτήσεις

Για πιο προχωρημένες μαθηματικές πράξεις μπορείτε να χρησιμοποιήσετε συναρτήσεις (functions). Οι συναρτήσεις είναι τμήματα κώδικα που εκτελούν μια συγκεκριμένη ενέργεια (πχ εμφάνιση του output στην οθόνη). Για τη χρήση των ενσωματωμένων μαθηματικών συναρτήσεων πρέπει να εισάγεται στην αρχή του προγράμματος την βιβλιοθήκη "math" με την εντολή "import".

In [6]:
```
import math

math.sin(3)
```
Out[6]: 0.1411200080598672

In [7]:
```
math.sqrt(4)
```
Out[7]: 2.0

In [8]:
```
math.pi
```
Out[8]: 3.141592653589793

Η συνάρτηση print()

Για να εμφανίσετε αποτελέσματα του προγράμματος σας στην οθόνη χρησιμοποιείστε την συνάρτηση print().

In [ ]:
```
print("My name is George")
```

In [ ]:
```
print('The square root of 5 is', math.sqrt(5))
```

Μεταβλητές

Οι μεταβλητή (variable), είναι ένα συμβολικό όνομα μιας περιοχής της μνήμης στην οποία μπορούμε να αποθηκεύσουμε και να ανακτήσουμε δεδομένα μέσω του συμβολικού αυτού ονόματος.

In [1]:
```
elevation = 10
print(elevation)
```
```
10
```

Τύποι δεδομένων

Data type name	Data type	Example
int	Whole integer values	4
float	Decimal values	3.1415
str	Character strings	'Hot'
bool	True/false values	True

Ο τύπος δεδομένων μπορεί να βρεθεί χρησιμοποιώντας τη συνάρτηση `type()`

In [1]:
```
x = 5.2
type(x)
```
Out[1]: float

Είσοδος χρήστη

Με την συνάρτηση `input()`, είναι δυνατή η είσοδος του χρήστη:

In [4]:
```
name = input('What is your name ? ')
print('Nice to meet you', name)
```
```
What is your name ? John
Nice to meet you John
```

Συλλογές Δεδομένων

- List
- Tuple
- Set
- Dictionary

In [13]:
```
# A list
fruits = ["apple", "banana", "orange", "watermelon"]

print(fruits)

# Access items
print(fruits[2])
print(fruits[-1])

# Range of indexes
print(fruits[1:3])
print(fruits[1:])
print(fruits[:4])

fruits[1] = 'cherry'
print(fruits)

fruits.append('pear')
print(fruits)
```
```
['apple', 'banana', 'orange', 'watermelon']
orange
watermelon
['banana', 'orange']
['banana', 'orange', 'watermelon']
['apple', 'banana', 'orange', 'watermelon']
['apple', 'cherry', 'orange', 'watermelon']
['apple', 'cherry', 'orange', 'watermelon', 'pear']
```

In [2]:
```
# A tuple

fruits = ("apple", "banana", "orange", "watermelon")
print(fruits)

# Access items
print(fruits[2])
print(fruits[-1])

# Range of indexes
print(fruits[1:3])
print(fruits[1:])
print(fruits[:4])

# The following is supported by tuples
fruits[1] = 'cherry'
print(fruits)
```
```
('apple', 'banana', 'orange', 'watermelon')
orange
watermelon
('banana', 'orange')
('banana', 'orange', 'watermelon')
('apple', 'banana', 'orange', 'watermelon')
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-2-73a23ed8cdd7> in <module>
     14
     15 # The following is supported by tuples
---> 16 fruits[1] = 'cherry'
     17 print(fruits)

TypeError: 'tuple' object does not support item assignment
```

In [39]:
```
# A set
fruits = {"apple", "banana", "orange", "watermelon"}

for i in fruits:
    print(i)
```
```
apple
banana
orange
watermelon
```

In [34]:
```
# A dictionary
car =   {
  "brand": "Toyota",
  "model": "yaris",
  "year": 2020
}
print(car)
x = car["model"]
print(x)
```
```
{'brand': 'Toyota', 'model': 'yaris', 'year': 2020}
yaris
```

if / if-else statement

In [43]:
```
x = 20
y = 10
if x > y:
    print("x is greater than y")
elif x < y:
    print("y is greater than x")
else:
    print("x is equal with y")
```
```
x is greater than y
```

While loops

In [45]:
```
i = 1
while i < 10:
    print(i)
    if i == 4:
        break
    i += 1
```
```
1
2
3
4
```

For loops

In [53]:
```
fruits = ['apple', 'banana', 'orange', 'watermelon']
for x in fruits:
    print(x)
```
```
apple
banana
orange
watermelon
```

Συναρτήσεις

In [56]:
```
def eq(a, x, b):
    y = a*x + b
    return y

solution = eq(3, 4, 2)
print(solution)
```
```
14
```

Κλάσεις - Αντικείμενα

In [1]:
```
# Class definition
class Dog():
    def __init__(self, name, color, age):
        self.name = name
        self.color = color
        self.age = age

    def speak(self):
        print("Hello my name is " + self.name + ", I have "
              + self.color + " color and I'm " + str(self.age) + " years old")

# Object definition
d1 = Dog("Jack", "black", 4)
print("My dog is " + str(d1.age) + " years old")

d1.speak()

d2 = Dog("Lucy", "white", 2)
print("It has " + d2.color + "color")

d2.speak()
```
```
My dog is 4 years old
Hello my name is Jack, I have black color and I'm 4 years old
It has whitecolor
Hello my name is Lucy, I have white color and I'm 2 years old
```

GIS Python βιβλιοθήκες

Data analysis & visualization:

- Numpy –> Fundamental package for scientific computing with Python
- Pandas –> High-performance, easy-to-use data structures and data analysis tools
- Scipy –> A collection of numerical algorithms and domain-specific toolboxes, including signal processing, * optimization and statistics
- Matplotlib –> Basic plotting library for Python

GIS:

- GDAL / OGR –> Fundamental package for processing vector and raster data formats (many modules below depend on this). Used for raster processing.
- Geopandas –> Working with geospatial data in Python made easier, combines the capabilities of pandas and shapely.
- Shapely –> Python package for manipulation and analysis of planar geometric objects (based on widely deployed GEOS).
- Pyproj –> Performs cartographic transformations and geodetic computations (based on PROJ.4).
- Rasterio –> Clean and fast and geospatial raster I/O for Python.

Στο εργαστήριο αυτό θα χρησιμοποιήσουμε τις βιβλιοθήκες Geopandas, Shapely και Pyproj.

- https://geopandas.org/
- https://shapely.readthedocs.io/en/stable/manual.html
- https://pyproj4.github.io/pyproj/stable/

Βασική γνώση απαραίτητη για τις ασκήσεις

Shapefile:

Shapefile ειναι το format που θα χρησιμοποιήσουμε για τα διανυσματικά δεδομένα. Για την σωστή χρήση του αρχείου shapefile, πρέπει να υπάρχουν στον ίδιο φάκελο τα παρακάτω υπο-αρχεία:

- .shp
- .dbf
- .shx
- .prj

Κωδικοί EPSG:

Οι EPSG κωδικοί αντιπροσωπεύουν συστήματα αναφοράς και χρησιμοποιούνται στα προγράμματα και τα πακέτα λογισμικού GIS για τον ορισμό / μετασχηματισμό / διαχείριση συστημάτων αναφοράς.

Γεωμετρικά αντικείμενα

- Point
- Line
- Polygon

- MultiPoint
- MultiLineString
- MultiPolygon

Point

In [1]:
```
from shapely.geometry import Point, LineString, Polygon

# Create a point
point1 = Point(2.3, 4.2)
point2 = Point(7.3, -35.1)
point3 = Point(8.26, -3.456)
point3D = Point(10.26, -2.456, 0.57)
```

In [2]:
```
point1
```
Out[2]:

In [3]:
```
# Calculate the distance between point1 and point2
point_dist = point1.distance(point2)

print("Distance between the points is {0:.2f} meters".format(point_dist))
```
```
Distance between the points is 39.62 meters
```

LineString

In [15]:
```
# Create a LineString from our Point objects
line = LineString([point1, point2, point3])

# It is also possible to produce the same outcome using coordinate tuples
line2 = LineString([(2.3, 4.2), (7.3, -35.1), (8.26, -3.456)])
```

In [16]:
```
line
```
Out[16]:

In [17]:
```
# Get the lenght of the line
l_length = line.length

# Get the centroid of the line
l_centroid = line.centroid

# Print the outputs
print("Length of our line: {0:.2f}".format(l_length))
print("Centroid of our line: ", l_centroid)
print("Type of the centroid:", type(l_centroid))
```
```
Length of our line: 71.28
Centroid of our line:  POINT (6.123634431860776 -17.15029282052452)
Type of the centroid: <class 'shapely.geometry.point.Point'>
```

Polygon

In [18]:
```
# Create a Polygon from the coordinates
poly = Polygon([(2.3, 4.2), (7.3, -35.1), (8.26, -3.456)])

# It is also possible to produce the same outcome using a list of lists which contain
# We can do this using the point objects we created before and a list comprehension:
# --> here, we pass a list of lists as input when creating the Polygon (the linst comp
poly2 = Polygon([[p.x, p.y] for p in [point1, point2, point3]])
```

In [18]:
```
poly
```
Out[18]:

In [19]:
```
print('poly:', poly)
print('poly2:', poly2)
```
```
poly: POLYGON ((2.3 4.2, 7.3 -35.1, 8.26 -3.456, 2.3 4.2))
poly2: POLYGON ((2.3 4.2, 7.3 -35.1, 8.26 -3.456, 2.3 4.2))
```

In [21]:
```
# Get the centroid of the Polygon
poly_centroid = poly.centroid

# Get the area of the Polygon
poly_area = poly.area

poly_bbox = poly.bounds

# Print the outputs
print("Poly centroid: ", poly_centroid)
print("Poly Area: ", poly_area)
print("Poly Bounding Box: ", poly_bbox)
```
```
Poly centroid:  POINT (5.953333333333333 -11.452)
Poly Area:  97.97400000000002
Poly Bounding Box:  (2.3, -35.1, 8.26, 4.2)
```

Διανυσματικά δεδομένα

Με τη χρήση της βιβλιοθήκης `Geopandas` μπορούμε να διαχειριστούμε γεωχωρικά δεδομένα. Το `Geopandas` αποτελεί επέκταση της βιβλιοθήκης για data analysis `Pandas` έτσι ώστε να είναι δυνατή η ανάλυση γεωχωρικών δεδομένων.

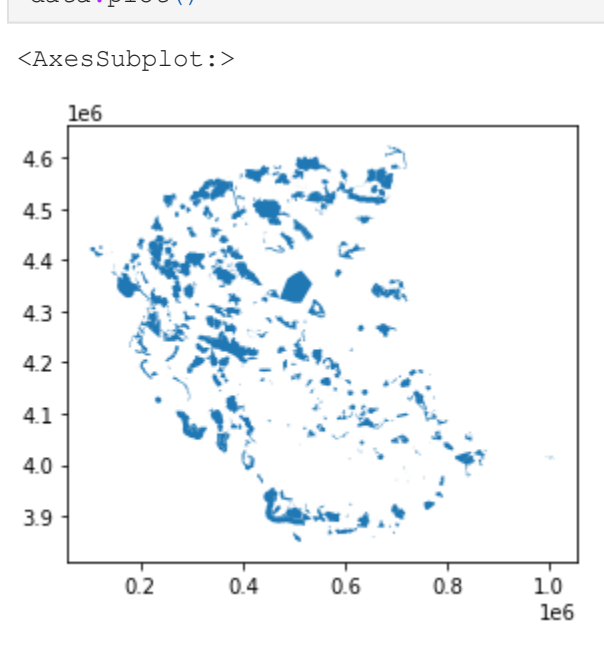
Εισαγωγή και ανάγνωση ενός αρχείου Shapefile

In [1]:
```
import geopandas as gpd

# Read file from Shapefile
fp = "data/natura/gr_natura_v30.shp"
data = gpd.read_file(fp)

type(data)
```
Out[1]: geopandas.geodataframe.GeoDataFrame

In [2]:
```
print(data.columns)
```
```
Index(['OBJECTID', 'SITECODE', 'SITETYPE', 'Shape_Leng', 'Shape_Area',
       'geometry'],
      dtype='object')
```

Οπτικοποίηση του Shapefile αρχείου

In [6]:
```
%matplotlib inline
data.plot()
```
Out[6]: <AxesSubplot:>



Υπολογισμός του εμβαδού των πρώτων πέντε οντοτήτων

In [7]:
```
# Iterate over rows and print the area of a Polygon
for index, row in data[0:5].iterrows():

    # Get the area from the shapely-object stored in the geometry-column
    poly_area = row['geometry'].area

    # Print info
    print("Polygon area at index {index} is: {area:.2f} m^2".format(index=index, area=
```
```
Polygon area at index 0 is: 99677459.45 m^2
Polygon area at index 1 is: 163291185.32 m^2
Polygon area at index 2 is: 432992897.94 m^2
Polygon area at index 3 is: 96346547.92 m^2
Polygon area at index 4 is: 757562820.21 m^2
```

Επαλήθευση του αποτελέσματος εξάγωντας τις αντίστοιχες τιμές από την περιγραφική πληροφορία του αρχείου

In [8]:
```
# It is possible to get a specific column by specifying the column name within square
print(data['Shape_Area'][0:5])
```
```
0    9.967746e+07
1    1.632912e+08
2    4.329929e+08
3    9.634655e+07
4    7.575628e+08
Name: Shape_Area, dtype: float64
```

Μετασχηματισμός προβολικού συστήματος

In [9]:
```
# EPSG information
print(data.crs)

print(data['geometry'].head())
```
```
epsg:2100
0    POLYGON ((675261.287 4553517.408, 675261.165 4...
1    POLYGON ((639643.343 4484885.199, 639819.039 4...
2    MULTIPOLYGON (((688920.416 4569026.849, 688938...
3    POLYGON ((684870.160 4527377.228, 684940.275 4...
4    POLYGON ((615054.366 4530886.252, 615135.133 4...
Name: geometry, dtype: geometry
```

In [10]:
```
# Let's make a backup copy of our data
data_wgs84 = data.copy()

# Reproject the data
data = data.to_crs(epsg=4326)

print(data.crs)

print(data['geometry'].head())
```
```
epsg:2100
0    POLYGON ((675261.287 4553517.408, 675261.165 4...
1    POLYGON ((639643.343 4484885.199, 639819.039 4...
2    MULTIPOLYGON (((688920.416 4569026.849, 688938...
3    POLYGON ((684870.160 4527377.228, 684940.275 4...
4    POLYGON ((615054.366 4530886.252, 615135.133 4...
Name: geometry, dtype: geometry
```

Εγγραφή των νέων δεδομένων στο δίσκο

In [11]:
```
natura2000_ggrs87 = 'data/natura/natura_ggrs87.shp'
data.to_file(natura2000_ggrs87)
```

Τοπολογικές σχέσεις

Σημείο εντός ή εκτός πολυγώνου

Για να ελέγξουμε αν ένα σημείο είναι εντός ή εκτός πολυγώνου μπορούμε να χρησιμοποιήσουμε τις συναρτήσεις της βιβλιοθήκης shapely:

- .within()
- .contains()

In [64]:
```
from shapely.geometry import Point, Polygon

# Create Point objects
p1 = Point(24.0217, 35.3031)
p2 = Point(24.1468, 35.3215)

# Create a Polygon object
coords = [(23.9308, 35.3094), (24.0591, 353459), (24.1432, 35.2873)]
poly = Polygon(coords)

print(p1)
print(p2)
print(poly)
```
```
POINT (24.0217 35.3031)
POINT (24.1468 35.3215)
POLYGON ((23.9308 35.3094, 24.0591 353459, 24.1432 35.2873, 23.9308 35.3094))
```

In [65]:
```
# Check if p1 is within the polygon using the within function
p1.within(poly)
```
Out[65]: True

In [66]:
```
# Check if p2 is within the polygon
p2.within(poly)
```
Out[66]: False

In [67]:
```
# Does polygon contain p1?
poly.contains(p1)
```
Out[67]: True

In [68]:
```
# Does polygon contain p2?
poly.contains(p2)
```
Out[68]: False

Τομή

Για να ελέγξουμε αν ένα αντικείμενο τέμνεται με κάποιο άλλο χρησιμοποιούμε τις συναρτήσεις:

- intersects()

In [74]:
```
from shapely.geometry import LineString, MultiLineString

# Create two lines
line_a = LineString([(0, 0), (1, 1)])
line_b = LineString([(1, 1), (0, 2)])

line_a.intersects(line_b)
```
Out[74]: True