

Python and (Q)GIS

A very brief overview of the possibilities of using python scripting in GIS:

QGIS/Python ecosystem

- Written in C++
- In 2007 work began to add python as scripting language
- QGIS have 400 core C++ classes of which 75% are python enabled through SIP
- QGIS use Qt (C++) for windows and buttons etc. PyQt binds this with python

Python and (Q)GIS

cont.

Recommend reading (in course literature list):

- The PyQGIS Programmers Guide, Gary Sherman (QGIS founder)
- PyQGIS developer cookbook

https://docs.qgis.org/testing/en/docs/pyqgis_developer_cookbook/

recent transition period between QGIS2 and QGIS3

QGIS2 is no more [2022]

Package management

pip – a install program for python packages

`pip list`

`pip install your_favorite_module`

`pip uninstall your_not_so_favorite_module`

Windows users:

Use the setup file in the start menu

Python and (Q)GIS

There are four main modules that has most of all the python classes for QGIS

```
from PyQt4.QtCore import * (qgis.PyQt.QtCore)
```

```
from PyQt4.QtGui import * (qgis.PyQt.QtGui)
```

```
from qgis.core import *
```

```
from qgis.gui import *
```

(Not good coding practice to use *)

In parentheses used in QGIS3. qgis.analysis is still available in QGIS3

Python and (Q)GIS

Let us try to open a vector layer with only code

Open python console in QGIS

```
>>>vlayer = QgsVectorLayer('c:/temp/drone_pos.gml',  
'point', 'ogr')
```

```
>>>vlayer.isValid()
```

```
>>>QgsProject.instance().addMapLayer(vlayer)
```

Zoom in to one point

```
>>>iface.zoomFull()
```

All GUI functionalities are accessible through code

https://docs.qgis.org/3.16/en/docs/pyqgis_developer_cookbook/loadlayer.html

Python and (Q)GIS

Let us try to open a raster layer with only code

```
>>>fileName = 'c:/temp/DSM_LondonCity_1m.tif'

>>>fileInfo = QFileInfo(fileName)

>>>from qgis.PyQt.QtCore import *

>>>baseName = fileInfo.baseName()

>>>print(baseName)

>>>rlayer = QgsRasterLayer(fileName, baseName)

>>>QgsProject.instance().addMapLayer(rlayer)
```

https://docs.qgis.org/3.16/en/docs/pyqgis_developer_cookbook/loadlayer.html

An alternative way to open raster in order to get matrix and make into a numpy array

```
>>>import numpy as np
```

```
>>>from osgeo import gdal
```

```
>>>data = gdal.Open(fileName)
```

```
>>>mat = np.array(data.ReadAsArray())
```

```
>>>mat
```

The numpy array variable mat has many methods that could be used now

```
>>>mat.shape
```

```
>>>mat.mean()
```

qgis.core – qgis.gui

The CORE library contains all basic GIS functionality

The GUI library is build on top of the CORE library and adds reusable GUI widgets

```
>>>iface.messageBar().pushMessage("Ops", "Lots of red  
here", level=Qgis.Critical)
```

qgis.pyqt

Bindings between Qt and Python

```
>>>from qgis.PyQt.QtWidgets import QMessageBox
```

```
>>>QMessageBox.about(None, "About MyPlugin" ,"No animals  
were harmed in the development of this Plugin")
```


Getting started with an IDE (VSCode)

To use QGIS outside you need to add the following lines:

```
# Starting a QGIS application
```

```
qgishome = 'C:/OSGeo4W/apps/qgis/'
```

```
QgsApplication.setPrefixPath(qgishome, True)
```

```
app = QgsApplication([], False)
```

```
app.initQgis()
```

```
[
```

```
code
```

```
]
```

```
app.exitQgis()
```

Access attributes by a loop

Try to find out how to access vector attributes and write a loop that prints an attribute column from the drone vector file.

```
>>>idx = vlayer.dataProvider().fieldNameIndex('Id')  
  
>>>for f in vlayer.getFeatures():  
  
>>>    print(f.attributes()[idx])
```

See **PyQGIS Developer Cookbook** on how to edit attribute by code

Try to debug your loop

Accessing other tools

The PyQGIS libraries also have access to external tools

One solution is to use a system call to e.g. access a C++ function:

```
>>>import subprocess
```

```
>>>rSquare = 100
```

```
>>>x = 283935
```

```
>>>y = 5711504
```

```
>>>gdalclipdsm = 'gdalwarp -dstnodata -9999 -q -overwrite -  
te ' + str(x - rSquare) + ' ' + str(y - rSquare) + ' ' +  
str(x + rSquare) + ' ' + str(y + rSquare) + ' -of GTiff ' +  
'c:/temp/DSM_LondonCity_1m.tif c:/temp/clipdsm.tif'
```

```
>>>subprocess.call(gdalclipdsm)
```

Accessing other tools

gdal also part of core functionality in QGIS via Python:

```
>>>from osgeo import gdal
```

```
>>>r = 200
```

```
>>>x = 283935
```

```
>>>y = 5711504
```

```
>>>filepath_tempdsm = 'C:/temp/clipdsm.tif'
```

```
>>>bigraster = gdal.Open(fileName)
```

```
>>>bbox = (x - r, y + r, x + r, y - r)
```

```
>>>gdal.Translate(filepath_tempdsm, bigraster, projWin=bbox)
```

```
>>>bigraster = None
```

Accessing other tools

One other solution is to access the tools in the processing toolbox (VSCode). Here you can access function from GRASS, SAGA etc.:

```
# Prepare processing framework
```

```
sys.path.append('C:/OSGeo4W/apps/qgis/python/plugins')
```

```
import processing
```

```
from processing.core.Processing import Processing
```

```
Processing.initialize()
```

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

```
processing.algorithmHelp("native:buffer")
```

Accessing other tools

One other solution is to access the tools in the processing toolbox (VSCode). Here you can access function from GRASS, SAGA etc.:

```
processing.run("native:buffer", {'INPUT':  
'c:/temp/DroneExercise/dronephotos.shp', 'DISTANCE':  
100.0, 'SEGMENTS': 10, 'DISSOLVE': True, 'END_CAP_STYLE': 0,  
'JOIN_STYLE': 0, 'MITER_LIMIT': 10, 'OUTPUT':  
'c:/temp/DroneExercise/dronephotos_buff.shp'})
```

To access the syntax of each tool, first run it in the processing toolbox in QGIS and then look at the log

Making a python script

- A drone path is planned over the central area on London and we would like to retrieve height information for an area around each point
- Now we will examine the mean and (maximum height) for a 200 meter square around all points based on the DSM
- This will be accomplished by making a loop through each vector point
- To make it easier and also to easily change the settings, we will produce a script

Task 1

- Write out the result from previous script to a text file. The text file should consist of a header, and three column (ID, mean height and max height)

Hints:

- Use **numpy.zeros()** function to create an empty matrix to fill in your numbers
- Use **numpy.savetxt()** function to save to file

Task 2

Perform a moving 3x3 kernel filtering (both smoothing and edge detecting) on a raster by using a nested looping process and saving the raster to disk.

Use one cutout (500 x 500 pixels) (any position) from the raster data used in first task.

`np.array()` can be used to create a kernel matrix

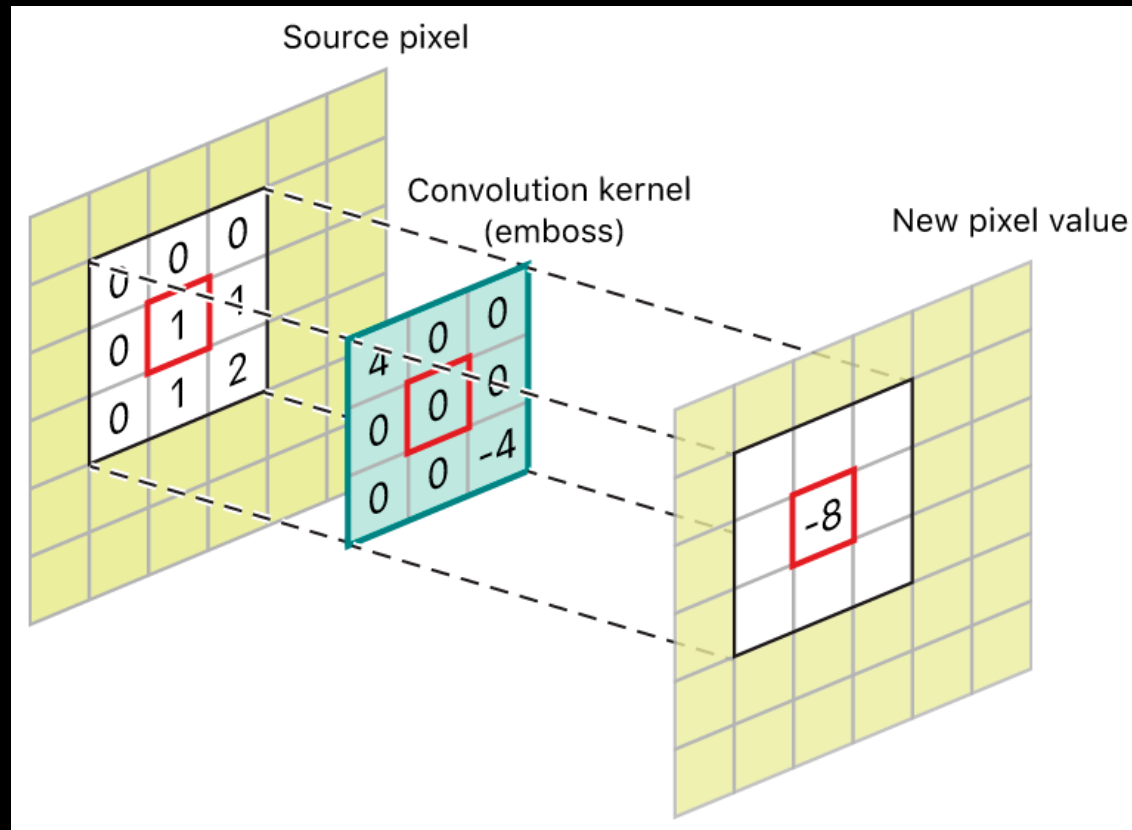
Beware of the raster edges!!!

`np.where()` can be used to extract values from the raster using the kernel

Make use of the **saveraster** function in **misc.py** found on Canvas to save your filtered raster.

Task 2

A kernel filter, convolution matrix, or mask is a small matrix. It is used for blurring, sharpening, embossing, edge detection, and more. This is accomplished by doing a convolution between a kernel and an image.



Example code for plotting two raster:

```
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(12, 6))

ax1 = plt.subplot(1,2,1)

im1 = ax1.imshow(dsm, clim=[dsm.min(), dsm.max()])

plt.colorbar(im1)

plt.title('Original DSM')

ax2 = plt.subplot(1,3,2)

im2 = ax2.imshow(result, clim=[dsm.min(), dsm.max()])

plt.colorbar(im2)

plt.title('Mean filter')

plt.tight_layout()

plt.show()
```

Task 3

Extend your first script from task 1 to include a new column in your text file including average sky view factor calculated using a processing algorithm, either from UMEP (a QGIS third party plugin) or SAGA GIS

Make use of instructions given earlier. For third party plugins (such as UMEP), talk to teachers or look at next slide for more help

No hand in

Add third party plugin

```
sys.path.append(r'C:\Users\xlinfr\AppData\Roaming\QGIS\QGIS3\profiles\default\python\plugins')

import processing

from processing_umep.processing_umep_provider import ProcessingUMEPPProvider

umep_provider = ProcessingUMEPPProvider()

QgsApplication.processingRegistry().addProvider(umep_provider)

from processing.core.Processing import Processing

Processing.initialize()
```