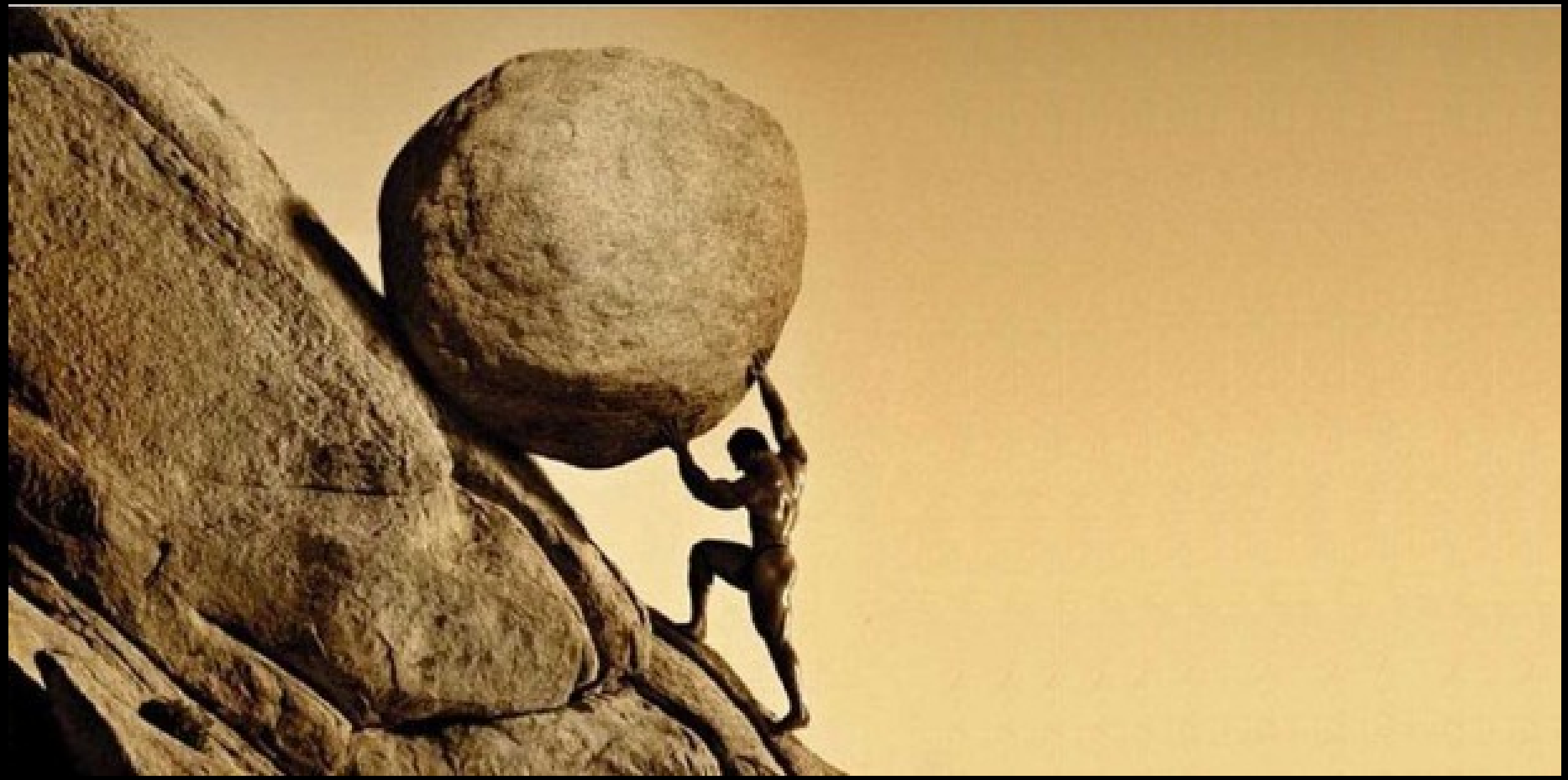
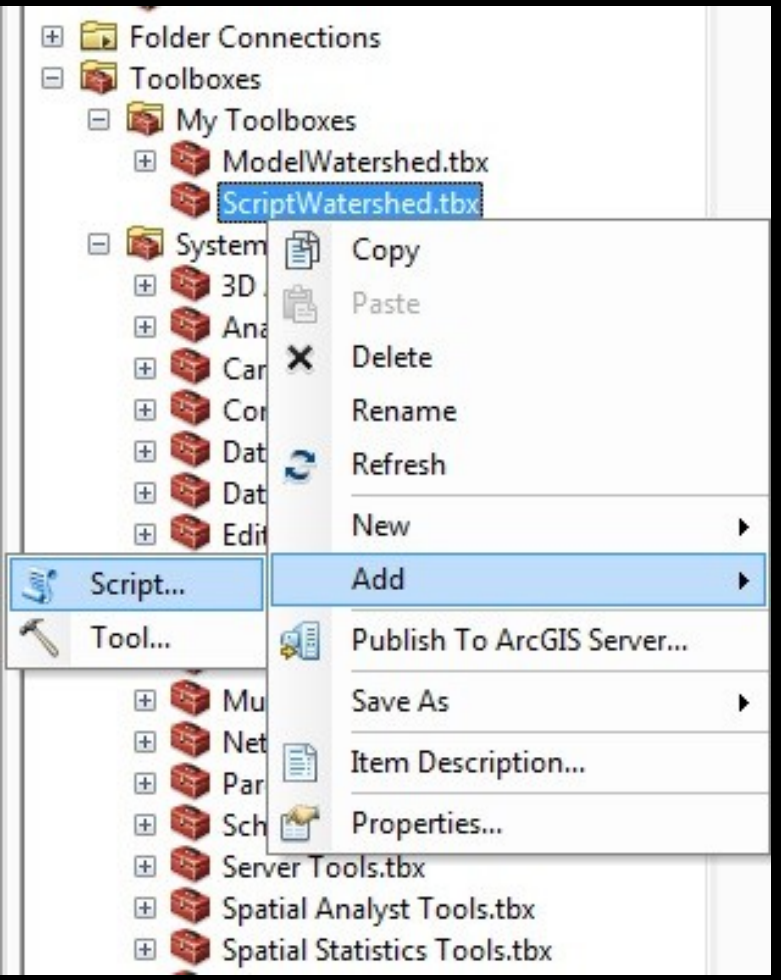
Python processing scripts for Qgis 3.X

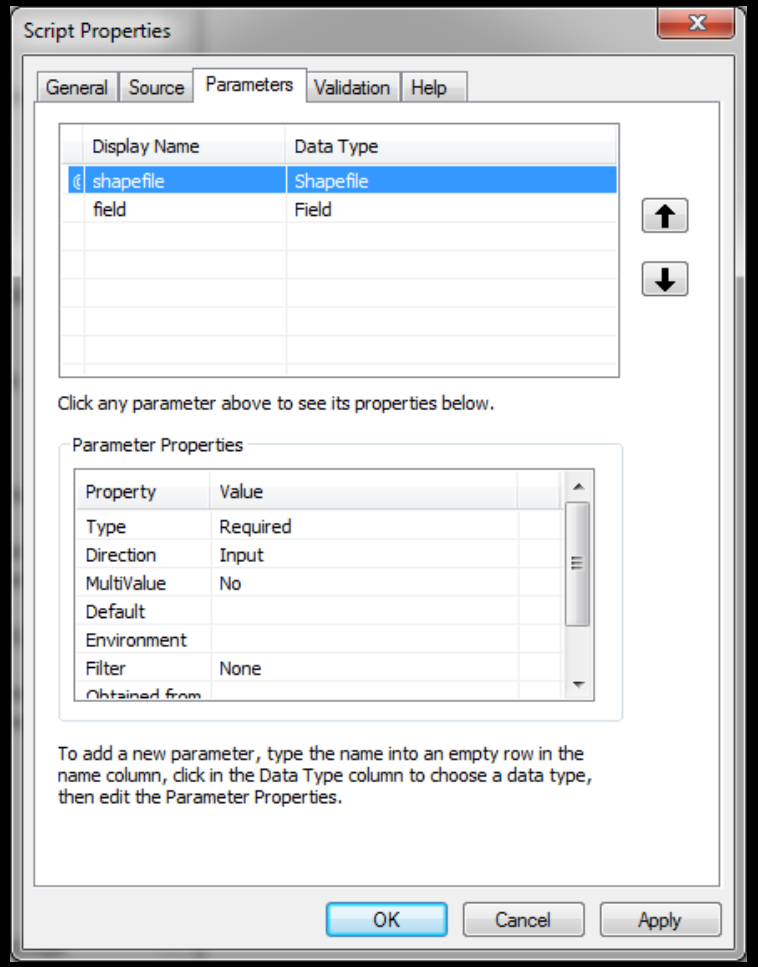
Yehuda Horn

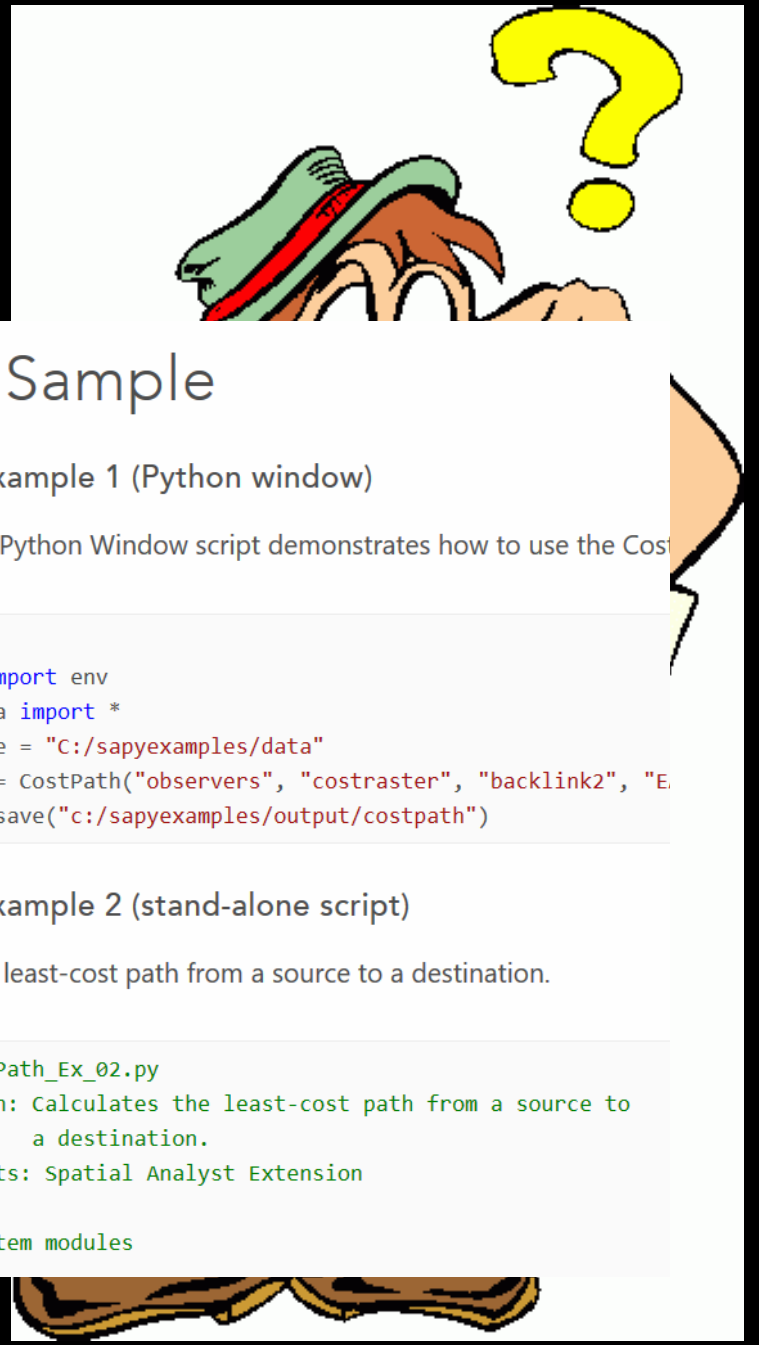
December 2018



From Arcpy to Qgis







Code Sample

CostPath example 1 (Python window)

The following Python Window script demonstrates how to use the Cost

```
import arcpy
from arcpy import env
from arcpy.sa import *
env.workspace = "C:/sapyexamples/data"
outCostPath = CostPath("observers", "costraster", "backlink2", "E
outCostPath.save("c:/sapyexamples/output/costpath")
```

CostPath example 2 (stand-alone script)

Calculates the least-cost path from a source to a destination.

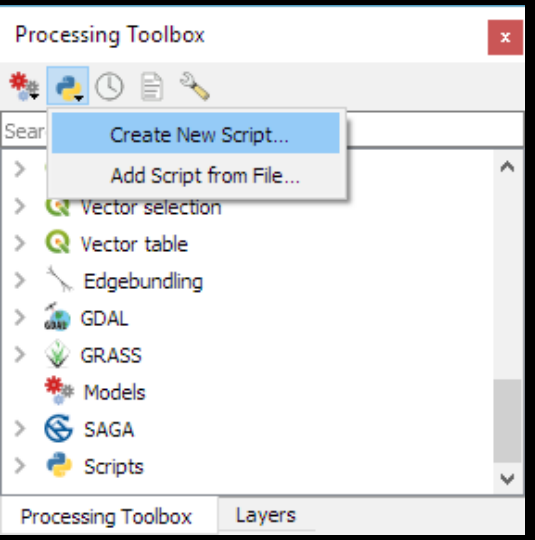
```
# Name: CostPath_Ex_02.py
# Description: Calculates the least-cost path from a source to
#              a destination.
# Requirements: Spatial Analyst Extension

# Import system modules
import arcpy
```

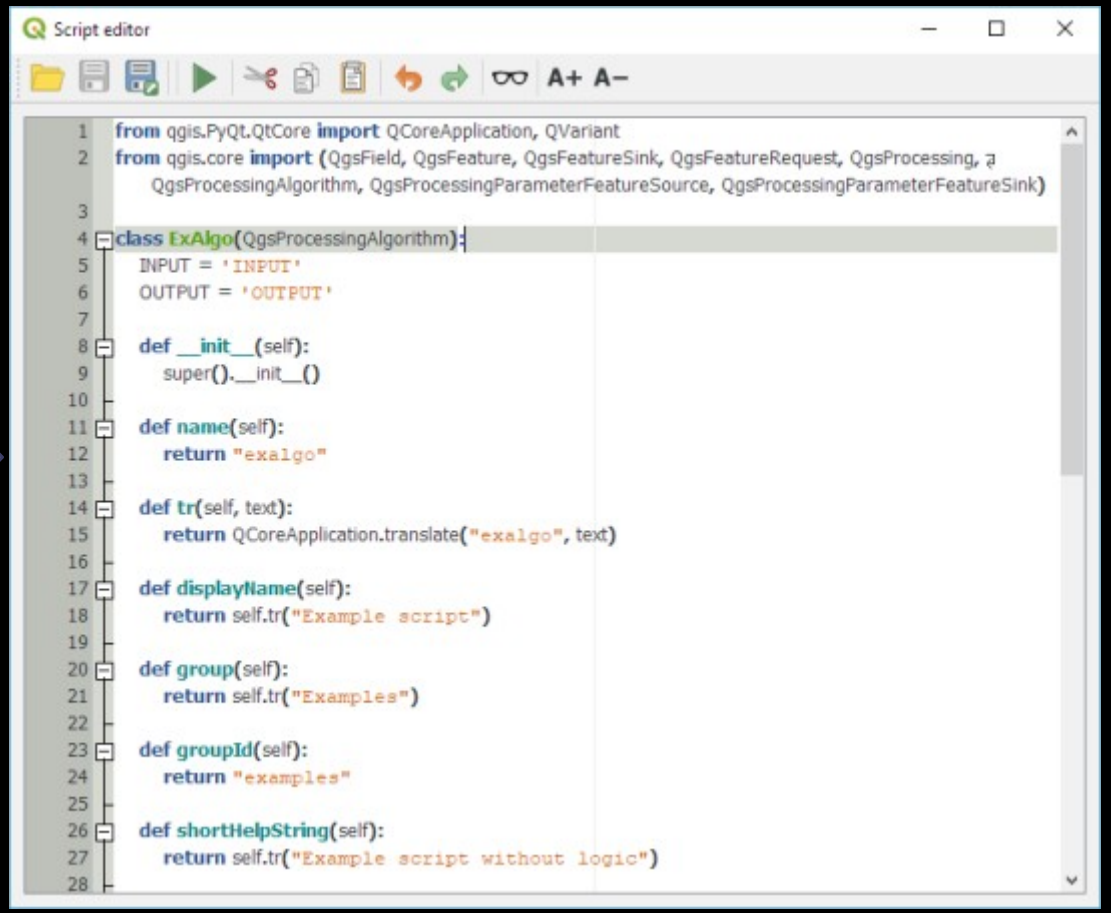
QGIS2.x to QGIS3.x

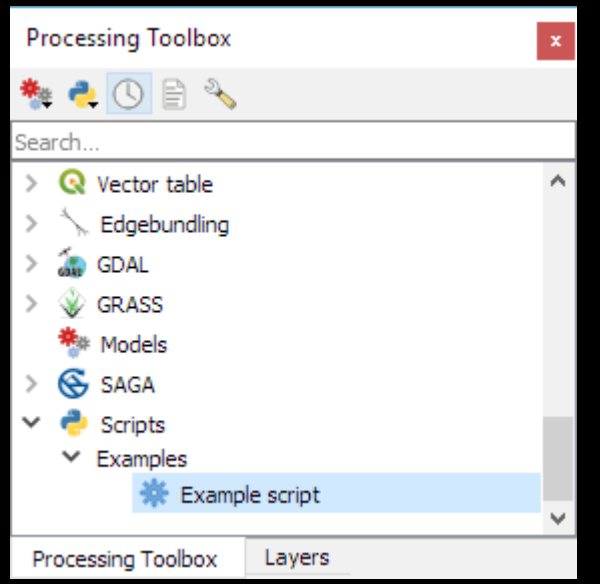
```
##Layer1=raster
##Layer2=raster
##myDouble=Double
##OutLayer1=output raster
##OutLayer2=output raster
```

PyQt 4 TO PyQt 5

Processing Flow


Processing Toolbox
Create New Script...
Add Script from File...
Vector selection
Vector table
Edgebundling
GDAL
GRASS
Models
SAGA
Scripts
Processing Toolbox
Layers


Script editor
A+ A−
from qgis.PyQt.QtCore import QCoreApplication, QVariant
from qgis.core import (QgsField, QgsFeature, QgsFeatureSink, QgsFeatureRequest, QgsProcessing,
QgsProcessingAlgorithm, QgsProcessingParameterFeatureSource, QgsProcessingParameterFeatureSink)
class ExAlgo(QgsProcessingAlgorithm):
INPUT = 'INPUT'
OUTPUT = 'OUTPUT'
def __init__(self):
super().__init__()
def name(self):
return "exalgo"
def tr(self, text):
return QCoreApplication.translate("exalgo", text)
def displayName(self):
return self.tr("Example script")
def group(self):
return self.tr("Examples")
def groupId(self):
return "examples"
def shortHelpString(self):
return self.tr("Example script without logic")


Processing Toolbox
Search...
Vector table
Edgebundling
GDAL
GRASS
Models
SAGA
Scripts
Examples
Example script
Processing Toolbox
Layers

Syntax

Import all necessary classes

```
from PyQt5.QtCore import QCoreApplication
from qgis.core import
    (QgsField, QgsFeature,
    QgsFeatureSink,
    QgsFeatureRequest,
    QgsProcessing,
   QgsProcessingParameterFile)
Import  json
Import pandas as pd
```

Define the algorithm as a class inheriting from QgsProcessingAlgorithm

```
class ExampleProcessingAlgorithm(QgsProcessingAlgorithm):
```

Python Classes/Objects:
Python is an object oriented programming language.
Almost everything in Python is an object, with its properties and methods.
A Class is like an object constructor, or a "blueprint" for creating objects.

Declare the names of the Input and Output parameters

```
class SpatialJoin(QgsProcessingAlgorithm):
    INPUT = "INPUT"
    JOIN = "JOIN"
    PREDICATE = "PREDICATE"
    JOIN_FIELDS = "JOIN_FIELDS"
    METHOD = "METHOD"
    PREFIX = "PREFIX"
    OUTPUT = "OUTPUT"
    NON_MATCHING = "NON_MATCHING"
```

GUI , Help , Etc

```
def tr(self, string):
  return QCoreApplication.translate('Processing', string)
def createInstance(self):
  return ExampleProcessingAlgorithm()
def name(self):
  return ' test'
def displayName(self):
  return self.tr(‘Test')
```

GUI , Help , Etc

```
def group(self):
    return self.tr('Qgis Meetup')
def groupId(self):
    return 'qgis_meetup'
def shortHelpString(self):
    return self.tr("Example algorithm short description")
```

Define the parameters of the processing framework

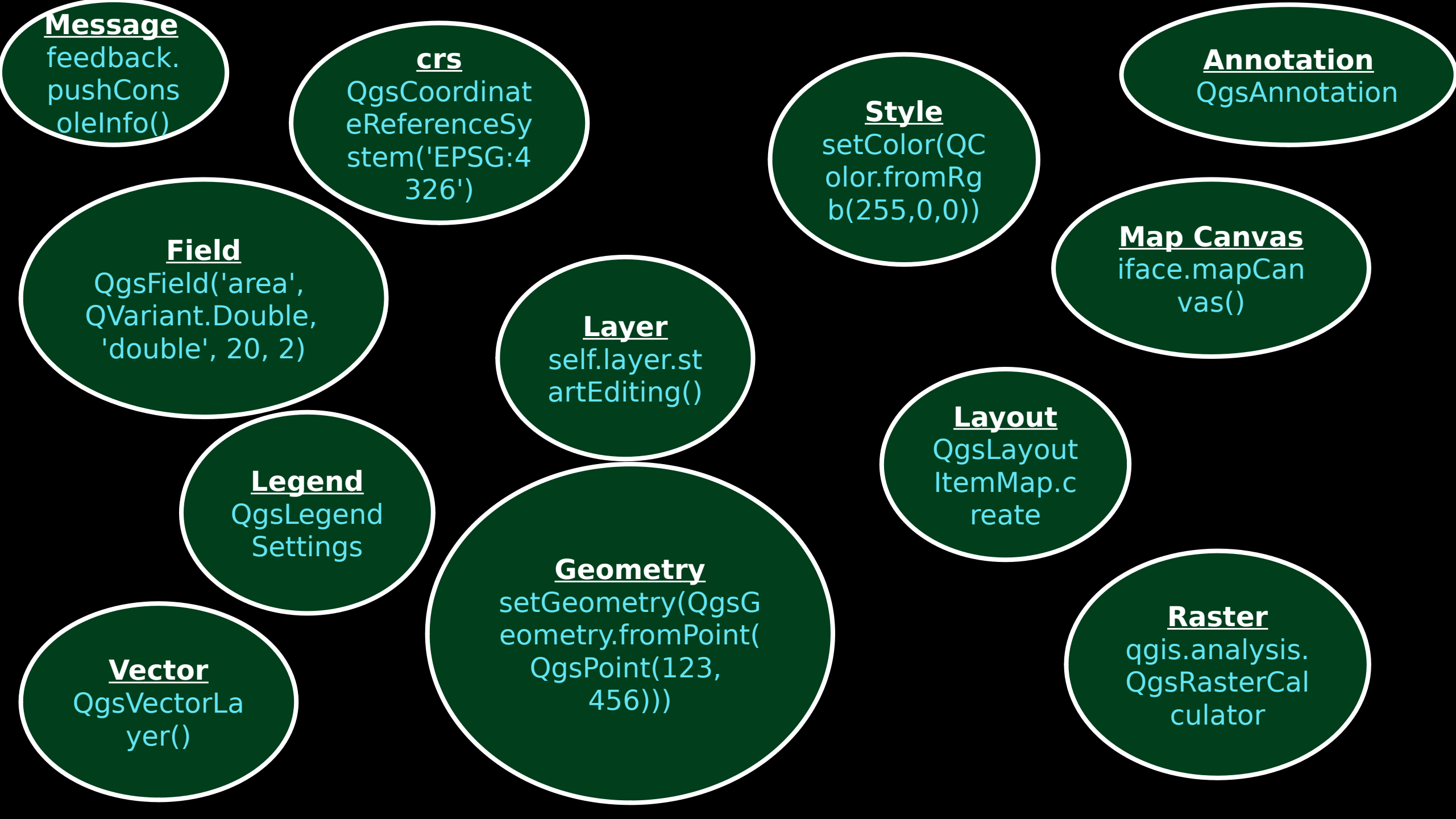
```
def initAlgorithm(self, config=None):
    self.addParameter(
        QgsProcessingParameterFeatureSource(
            self.INPUT,
            self.tr('Input layer'),
            [QgsProcessing.TypeVectorAnyGeometry] ) )
        self.addParameter(
                QgsProcessingParameterFeatureSink(
                        self.OUTPUT,
                self.tr('Output layer') )
        )
```

processAlgorithm function

```
def processAlgorithm(self, parameters, context, feedback):
    source = self.parameterAsSource(parameters, self.INPUT, context)
```

```
    results = {}
    results[self.OUTPUT_RASTER_A] = output_path_raster_a
    results[self.OUTPUT_RASTER_B] = output_path_raster_b
    return results
```

Elements


Message
feedback.pushConsoleInfo()
crs
QgsCoordinateReferenceSystem('EPSG:4326')
Style
setColor(QColor.fromRgb(255,0,0))
Annotation
QgsAnnotation
Field
QgsField('area', QVariant.Double, 'double', 20, 2)
Layer
self.layer.startEditing()
Map Canvas
iface.mapCanvas()
Legend
QgsLegendSettings
Layout
QgsLayoutItemMap.create
Geometry
setGeometry(QgsGeometry.fromPoint(QgsPoint(123, 456)))
Vector
QgsVectorLayer()
Raster
qgis.analysis.QgsRasterCalculator

Examples

Links

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

https://qgis.org/pyqgis/3.0/index.html

https://qgis.org/api/index.html

https://gis.stackexchange.com/questions/282773/writing-a-python-processing-script-with-qgis-3-0

https://github.com/qgis/QGIS/blob/master/doc/porting_processing.dox

http://www.geoint.co.il/%D7%94%D7%A8%D7%A6%D7%AA-%D7%A1%D7%A7%D7%A8%D7%99%D7%A4%D7%98%D7%99%D7%9D-%D7%91-qgis3/