**PyQGIS Developer Cookbook:**<https://docs.qgis.org/testing/en/docs/pyqgis_developer_cookbook/>

**Notes Udemy Automating QGIS:**iface: Stands for interface and is a “hook”. This creates a connection between the Python Console and the open QGIS interface.   
  
*Map canvas*  
mapCanvas: mc = iface.mapCanvas()  
Is an object that allows interactions with layers and the map, such as mc.layers(), mc.zoomScale(), mc.zoomIn(), mc.zoomOut().   
lyr = mc.currentLayer()# returns the current layer.   
lyrs = mc.layers() # returns a list of all layers that are currently “clicked/shown” in the interface.  
  
*QgsProject*lyrs = QgsProject.instance().mapLayersByName(‘Country’) #access a layer within   
#Creates a list since a name can refere to more than on layer  
lyr = lyrs[0] #Will return the layer in question.  
  
*Layer*lyr.isValid() #   
lyr.sourceName() #Returns the name of the layer, only works for vector layers  
  
ftrs=lyr.getFeatures() #This returns an QgsFeatureIterator object, which is similar to a list. An important (!) difference is that as it is iterated through it empties the object. So in order to loop through it a second time you need to instance it again.   
  
*Features:*print(feat[‘Name’], feat[‘POP\_EST’])  
   
  
***Script for running a sequence|subset of features into any processing.run tool (in retarded lining)*** *layer = QgsProject.instance().mapLayersByName('GridMask')[0]  
for nr in range(1, len(list(layer.getFeatures()))):  
 layer.selectByExpression(f"\"id\" = {nr}")  
 new\_layer = layer.materialize(QgsFeatureRequest().setFilterFids(layer.selectedFeatureIds())) QgsProject.instance().addMapLayer(new\_layer)*

*processing.run("gdal:cliprasterbymasklayer", {'INPUT':'E:/GIS/Data/Raster/AllSkyddadeOmrd/Blekinge/BLEKINGEHav.tif','MASK':new\_layer,'SOURCE\_CRS':None,'TARGET\_CRS':None,'NODATA':None,\ 'ALPHA\_BAND':False,'CROP\_TO\_CUTLINE':True,'KEEP\_RESOLUTION':False,'SET\_RESOLUTION':False,'X\_RESOLUTION':None,'Y\_RESOLUTION':None,'MULTITHREADING':False,'OPTIONS':'','DATA\_TYPE':0,\ 'EXTRA':'','OUTPUT':f'E:/GIS/Data/Raster/AllSkyddadeOmrd/Blekinge/{nr}CutRaster.tif'})*  
  
  
  
  
***Symbology***  *fn = QgsProject.instance().homePath() #FileName   
#Access the homepath of the project   
  
#Create path to layer within geopackage*fn = '[D:/Project/Tutorials/Udemy/AutomatingWithQGIS/original/Data/QGIS\_scripting.gpkg](file:///D:/Project/Tutorials/Udemy/AutomatingWithQGIS/original/Data/QGIS_scripting.gpkg)'  
fn += "|layername=airports"  
lyr = iface.addVectorLayer(fn, "Airports", 'ogr') #Add a vector based on its path  
renderer = lyr.renderer() #Neccessary object to access symbology  
  
lyr=QgsProject.instance().mapLayersByName('Natura\_2000\_Dissolved')

#Remove Layer  
QgsProject.instance().removeMapLayer(dissLayer)  
  
renderer  
<qgis.\_core.QgsSingleSymbolRenderer object at 0x000002425F2541F8>  
symbol = renderer.symbol()  
symbol #This object only allows change of a few properties   
<qgis.\_core.QgsMarkerSymbol object at 0x000002425F254288>  
  
symbol.setSize(4) #Change the size of the MarkerSymbol   
lyr.triggerRepaint() # Cause the change to be vizualised within the map  
ltv = iface.layerTreeView() #Allows for interactions and update of the treeview   
ltv.refreshLayerSymbology(lyr.id())   
  
symlyr1 = symbol.symbolLayers()[0] #Allows for manipulation of more properties  
symlyr1  
<qgis.\_core.QgsSimpleMarkerSymbolLayer object at 0x000002425F254438>  
  
symlyr1.properties()  
{'angle': '0', 'color': '196,60,57,255', 'horizontal\_anchor\_point': '1', 'joinstyle': 'bevel', 'name': 'circle', 'offset': '0,0', 'offset\_map\_unit\_scale': '3x:0,0,0,0,0,0', 'offset\_unit': 'MM', 'outline\_color': '35,35,35,255', 'outline\_style': 'solid', 'outline\_width': '0', 'outline\_width\_map\_unit\_scale': '3x:0,0,0,0,0,0', 'outline\_width\_unit': 'MM', 'scale\_method': 'diameter', 'size': '4', 'size\_map\_unit\_scale': '3x:0,0,0,0,0,0', 'size\_unit': 'MM', 'vertical\_anchor\_point': '1'}  
  
symlyr1  
<qgis.\_core.QgsSimpleMarkerSymbolLayer object at 0x000002425F254438>  
symlyr1.setColor(QColor('red'))  
lyr.triggerRepaint()  
symnew = QgsMarkerSymbol.createSimple({'name':'square', 'color':'blue'})  
symnew  
<qgis.\_core.QgsMarkerSymbol object at 0x000002425F2540D8>  
ltv.refreshLayerSymbology(lyr.id())

***Basic User Input****parent = iface.mainWindow()  
mc = iface.mapCanvas()  
  
sStr, bOK = QInputDialog.getText(parent, "Title", "Promt", text="Default")  
sStr, bOK = QInputDialog.getText(parent, "Get Layer", "Please Enter layername: ", text=mc.currentLayer().sourceName())  
  
if bOK:  
 print(f"User Entered {sStr}")  
else:  
 print("User canceled")  
  
lSpecies = ['RTHA','SWHA','BTHA','HAHA'] #l refers to List in the variable name   
sStr, bOK = QInputDialog.getItem(parent, "Species of Hawk", "What species did you see", lSpecies, editable=False)*

*if bOK:  
 print(sStr)  
else:  
 print("Canceled!")*

***List of common tools to apply in processing run or run and load*** *"native:dissolve", {'INPUT':f'E:/GIS/Data/Raster/ScriptOutdir/Vektor/ClipRegionByProt/{lyrName}.shp','FIELD':[],'OUTPUT':'TEMPORARY\_OUTPUT'}*

*param={ 'ALPHA\_BAND' : False, 'CROP\_TO\_CUTLINE' : False, 'DATA\_TYPE' : 0, 'EXTRA' : '',\ 'INPUT' : f'E:/GIS/Data/Raster/Symphony/AnalysFold/250m/SymphonyByRegion/{regionName}.tif', 'KEEP\_RESOLUTION' : False,\'MASK' : f'E:/GIS/Data/Raster/ScriptOutdir/Vektor/ProtMaskNollBf/{prct}.shp', 'MULTITHREADING' : False,\*

*'NODATA' : None, 'OPTIONS' : 'COMPRESS=PACKBITS', 'OUTPUT' : f'E:/GIS/Data/Raster/Symphony/AnalysFold/250m/SymphonyRegionProt/{regionName}\_\_{prct}.tif', \*

*'SET\_RESOLUTION' : False, 'SOURCE\_CRS' : None, 'TARGET\_CRS' : None, 'X\_RESOLUTION' : None, 'Y\_RESOLUTION' : None }*

*processing.run("gdal:cliprasterbymasklayer", param)*

***List of common cmd GIS commands:****gdalwarp -s\_srs IGNF:ETRS89LAEA -t\_srs EPSG:3006 -tr 10.0 10.0 -r near -of GTiff -co COMPRESS=LZW C:/GIS/CurrentDirectory/AnalysFold/10m/{file} C:/GIS/CurrentDirectory/AnalysFold/10m/{file[:-4]}swrf99.tif"*