# QGIS Cheatsheet

from pyqgis\_scripting\_ext.core import \*

## Create Geometries

Point = HPoint(30.0,10.0)

Print(point.asWkt())

coords = [[20,30],[30,40],[10,5],[30,20]]

line = HLineString.fromCoord(coords)

coords = [[20,30],[30,40],[10,5],[30,20],[20,30]]

polygon = HPolygon.fromCoords(coords)

extPoints = [[20,30],[30,40],[10,5],[30,20],[20,30]]

holePoints = [[21,29],[...]

polygonWithHole = HPolygon.fomCoords(extPoints)

holleRing = HLineString.fromCoords(holePoints)

polygonWithHole.add\_interior\_ring(hollering)

## Multigeometries

coords = [[20,30],[30,40],[10,5],[30,20]]  
MultiPoints = HMultiPoint.fromCoords(cords)

coords1 = [[20,30],[30,40],[10,5],[30,20]]

coords2 = [[21,29],[...]]

multiline = HMultiLineString.fromCoords([coords2],[coords2])

…

Multipolygon = HMultiPolygon.fromCoords([coords1, coords2])

### Subgeometries and Coordinates

subGeometries = multiPolygon.geometries()

for I in range(len(subGeometries)):

child = subGeometries[i]

print(f”polygon at position {i} = {child.asWkt()}”)

## Access Coordinates

for I, coordinate in enumerate(polygon.coordinates()):

print(f”coord {i} x ={coordinate[0]}, y={ coordinate[1]}”)

## Check Type of Geometry

Print(type(geometry))

Functions on Geometries

geom.length()

geom.area()

Point.distance(point2)

G1.intersects(g2)

G1.touches(g2)

G1.contains(g2)

G1.intersection(g2) 🡪 cuts out

G1.union(g2) 🡪 combination

G1.difference(g2). 🡪 returns g1 minus g2

G1.symdifference(g2). 🡪 portions not shared

G1.buffer(1.0)

G1.buffer(1.0,2). 🡪 buffer with few points

## Convex Hull

Collection = HGeometryCollection([g1,g2,g3,g4])

Convexhull = collection.convex\_hull()

## Read File

with open (file, 'r') as file:

lines = file.readlines()

## CRS TRANSFORM

crsHelper = HCrs()

crsHelper.from\_srid(4326)

crsHelper.to\_srid(3857)

newpoint = crsHelper.transform(point)

## CANVAS

canvas = HMapCanvas.new()

canvas.set\_extent(countryGeom.bbox())

canvas.add\_geometry(polygon, “colour”, 2)

canvas.show()

## OSM

osm = HMap.get\_osm\_layer()

canvas.set\_layers([osm])

## Remove Layer

HMap.remove\_layers\_by\_name(['OpenStreetMap'])

## Load Layers to QGIS Map

layer = HVectorLayer.open(gpg\_path, layer\_name)

HMap.add\_layer(layer)

## Some Functions

layer.field\_index(“…”) 🡪 finds the index of the str “…”

layerFeatures = layer.features() 🡪 work with features of attribute table

for feature in layerFeatures:

name = feature.attributes[Index]

countuies.append(name)

## Reading GPKG

geopackagePath = folder + "natural\_earth\_vector.gpkg"

countriesName = "ne\_50m\_admin\_0\_countries"

countriesLayer = HVectorLayer.open(geopackagePath, countriesName)

print("Schema (first 4 fields):")  
counter = 0  
for name, type in countriesLayer.fields.items():

counter = counter + 1 if counter < 5:

print("\t", name, "of type", type)

## Get more infos about gpkg layer:

crs = countriesLayer.prjcode

print("Projection: ", crs)

print("Spatial extent: ", countriesLayer.bbox())

print("Feature count: ", countriesLayer.size())

geom = feature.geometry

## FILTERS

https://docs.qgis.org/3.28/en/docs/user\_manual/expressions/functions\_list.html

expression = "NAME like 'I%' and POP\_EST > 30000000" features = countriesLayer.features(expression) count = 0  
for feature in features:

print(feature.attributes[nameIndex])

count+=1

print("Feature count with filter: ", count)

## BBOX Filters

lon = 11.119982

lat = 46.080428

point = HPoint(lon, lat)

buffer = point.buffer(2)

citiesLayer = HVectorLayer.open(geopackagePath, citiesName)

HMap.add\_layer(citiesLayer)

citiyNameIndex = citiesLayer.field\_index("NAME")

print("\napply bbox filter on features")

aoi = buffer.bbox()

count = 0

for feature in citiesLayer.features(bbox=aoi):

print(feature.attributes[citiyNameIndex]) count += 1

print("Count =", count)

## exact Geometry Filter

(no function for that 🡪1. Bbox filter, 2. Intersection)

print("\napply geometry filter on features")  
coutn = 0  
for feature in citiesLayer.features(geometryfilter=buffer):

print(feature.attributes[citiyNameIndex])

count += 1

print("Count =", count)

## CREATE AN IN-MEMORY VECTOR LAYER

fields = { 🡪 creare a schema with fields and datatypes

"id": "Integer",

"name": "String",

}

just2citiesLayer = HVectorLayer.new("test", "Point", "EPSG:4326", ) 🡪 schema and, apart of a name, the needed geometry type and crs:

just2citiesLayer.add\_feature(HPoint(-122.42, 37.78), [1, "San Francisco"])

just2citiesLayer.add\_feature(HPoint(-73.98, 40.47), [2, "New York"]) 🡪features can be created adding the geometry and the attributes, which need to be in the creation order of the schema:

## CREATE a NEW GPKG:

path = folder + "test.gpkg"

error = just2citiesLayer.dump\_to\_gpkg(path, overwrite=True) #overwirte true --> will start gpkg from scratch, if you want to insert a layer to the same gpkg overwrite = false

if error: #error is NONE if there is no error

print(error)

#when dumped to gpkg its not a temporary file anymore

HMap.add\_layer(just2citiesLayer) ---> just a temporary layer

More fields:

fields ={

"name": "String",

"population":"Integer",

"lat": "Double",

"lon": "Double"

}

oneCityMoreAttributes = HVectorLayer.new("test2", "Point", "EPSG:4326", fields) 🡪#"name of layer", "geometrytype","CRS", "schema"

oneCityMoreAttributes.add\_feature(HPoint(-73.98, 40.47), ["New York", 19040000, 40.47, -73.98])

#dumping it into tghe same gpkg, but to the "test" gpkg. to see it you need to drag it to the layers panel and then you can choose between test and test2

error = oneCityMoreAttributes.dump\_to\_gpkg(path, overwrite=False)

if error:

print(error)

## STYLE

citiesLayer.subset\_filter("SOV0NAME = 'Italy'") #just show those features that are obeying to the filter

**Marker‚¥**

pointStyle = HMarker("square",5,45) + HFill("red") + HStroke("black", 1) #3 argumnets: geom, size, rotation, 0,255,0 (RGB) is green (you can look it up on intscape, 128 is transparency, must be between 0 and 255 in this cas32

field = "NAME"

pointStyle += HLabel(field, yoffset =-15, xoffset = 15) + HHalo("white",1) #field, yoffset,... halo = buffer

field = "if(POP\_MAX > 1000000, concat(NAME, ' (',round(POP\_MAX/1000000,1),')'), NAME)" #if(expression,iftrue,else), concat takes several strings and takes them together

labelProperties = {

"font": "Arial",

"color": "black",

"size": 10,

"field": field,

"xoffset": 0,

"yoffset": -10

}

pointStyle += HLabel(\*\*labelProperties) + HHalo("white",1) # \*\* take the dictonary and expand it so that it is equivalet to HLabel(font='Arial',color='black',...)

citiesLayer.set\_style(pointStyle)

**for Polygon Layers**

countriesLayer = HVectorLayer.open(geopackagePath, countriesNames)

countriesLayer.subset\_filter("NAME = 'Italy'")

italyGeometry = countriesLayer.features()[0].geometry

polygonStyle = HFill("0,255,0,100") + HStroke("green",2)

countriesLayer.set\_style(polygonStyle)

**for Line Layers**

riversLayer = HVectorLayer.open(geopackagePath, riversName)

riversLayerItaly = riversLayer.sub\_layer(italyGeometry, "rivers\_italy", ['scalerank','name']) #qgis doesmz allow sub set with geometries. but you can do so and get a new layer. boundary, new layer name, s

**THEMATIC STYLING**

#list of lists for ranges. later on we will make a list of styles for these ranges

ranges = [

[0,0],

[1,5],

[6,7],

[8,9],

[10,11],

]

styles = [

HStroke("blue",7),

HStroke("blue",5),

HStroke("blue", 2),

HStroke("darkblue", 2),

HStroke("darkblue",1)

]

labelProperties = {

"font": "Arial",

"color": "blue",

"size": 14,

"field": 'name',

"along\_line": True,

"bold": True,

"italic": True

}

labelStyle = HLabel(\*\*labelProperties) +HHalo("white",1)

riversLayerItaly.set\_graduated\_style('scalerank', ranges, styles, labelStyle )

HMap.add\_layer(countriesLayer)

HMap.add\_layer(citiesLayer)

HMap.add\_layer(riversLayerItaly)