UTILISATION APY PYTHON QGIS

> **Utile**
>
> PyGIS Cookbook & Official cheat sheet
> Documentation
> Console : `Extensions > Console Python` puis lancer le script :

```
exec(open('/home/[...]/script.py'.encode('utf-8')).read())
```

MISE EN ROUTE

Prérequis : avoir téléchargé QGIS (version utilisée ici : 3.22)

La façon la plus simple d'utiliser Python dans QGIS est via la console. On l'ouvre : `Plugins > Python console` ou `Ctrl + Alt + P`
Vérifions que tout fonctionne avec une commande simple :

```
print("hello world")
```

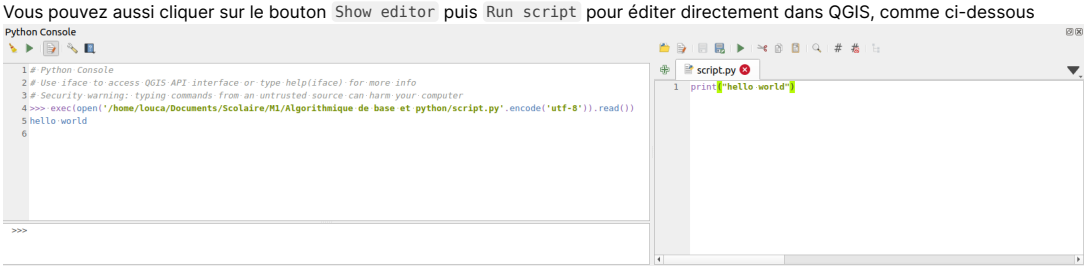
La console doit renvoyer le texte demandé :

```
# Python Console
# Use iface to access QGIS API interface or type help(iface) for more info
# Security warning: typing commands from an untrusted source can harm your computer
>>> print("hello world")
hello world
```

On peut aussi charger un script externe
Créons un script `script.py` avec la même instruction que précédemment dans un répertoire, puis lancez le script :

```
exec(open('/home/[...]/script.py'.encode('utf-8')).read())
```

Vous pouvez aussi cliquer sur le bouton `Show editor` puis `Run script` pour éditer directement dans QGIS, comme ci-dessous



Si la console renvoie le même texte, tout fonctionne. On peut maintenant coder via un éditeur externe et lancer des scripts, ou exécuter des instructions directement depuis la console.

MANIPULATION DE COUCHES

Téléchargez les données d'entrainement fournies par QGIS sur github. Dézippez le fichier ailleurs que dans votre répertoire de travail, puis copiez-collez XX et XX dans un dossier data, dans le même répertoire que votre script.

AJOUT DE COUCHES

On va d'abord ajouter la couche vecteur `regions` à notre projet pour l'afficher.
On crée d'abord un object de type `QgsVectorLayer` à partir du chemin vers la couche comme ceci : `layer = QgsVectorLayer(path, name, provider)` ; puis on vérifie que la couche chargée est valide, et si oui on l'ajoute à la carte.
On peut alternativement la méthode `addVectorLayer, plus concise`

Note : pour la manipulation de chemins, j'utilise `os`. Vous pouvez entrer les chemins manuellement si c'est plus simple

```
import os
wdir = "/home/louca/Documents/Scolaire/M1/Algorithmique de base et python/pygis/data/"
path_regions = os.path.join(wdir, "shapefiles/regions.shp")

# Méthode 1
layer = QgsVectorLayer(reg, "Regions", "ogr")
if not layer.isValid():
        print("Error in layer loading")
else:
        QgsProject.instance().addMapLayer(layer)
```

```
# Méthode 2
layer = iface.addVectorLayer(reg, "Regions", "ogr")
if not layer:
  print("Error in layer loading")
```

De la même façon, on peut aussi charger la couche `trees`.

Au final, on a :

```
# exec(open('/home/louca/Documents/Scolaire/M1/Algorithmique de base et python/pygis/script.py'.encode('utf-8')).read())
import os
wdir = "/home/louca/Documents/Scolaire/M1/Algorithmique de base et python/pygis/data/"

layers = [
        QgsVectorLayer(os.path.join(wdir, "shapefiles/regions.shp"), "Regions", "ogr"),
        QgsVectorLayer(os.path.join(wdir, "shapefiles/trees.shp"), "Trees", "ogr")
]
for layer in layers:
        print(layer)
        if not layer.isValid():
                print("Error in loading layer", layer)
        else:
                QgsProject.instance().addMapLayer(layer)

print("Layers loaded")
```

Vous pouvez supprimer toutes les couches chargées, puis lancer votre script. Si tous fonctionne, les couches regions et trees doivent être visibles dans l'interface et dans la liste des couches.
Si vous voulez que cette suppression se fasse automatiquement à chaque exécution de script, vous pouvez ajouter cette ligne au début du script :
`QgsProject.instance().removeAllMapLayers()`

INFORMATIONS SUR LES COUCHES

On peut afficher des informations de base sur les couches dans la console.
Pour récupérer un dictionnaire identifiant unique de couche:objet couche : `toutes_couches = QgsProject.instance().mapLayers()`
Un object `QgsMapLayer` (ou objet héritant de cette classe) a notamment les propriétés suivantes :

```
nom = layer.name()
emprise = layer.extent()
type_couche = layer.type() #si vecteur = QgsMapLayer.VectorLayer
nbr_obj = layer.featureCount() #nombres de feature (entités)
liste_attr = layer.fields() #éventuellement, .names().
```

On peut donc afficher des informations sur les couches, par exemple comme ceci :

```
for l in QgsProject.instance().mapLayers().values():
        print("Couche", l.name())
        print("Couche de type", l.type(), ", d'emprise ", l.extent())
        print("Contient ", l.featureCount(), " entités")
        print("Liste des attributs :", l.fields().names())
        print("Liste des attributs :", [f.name() for f in l.fields()])
        print("--------")
```

Python Console

```
# Python Console
# Use iface to access QGIS API interface or type help(iface) for more info
# Security warning: typing commands from an untrusted source can harm your computer
>>> exec(open('/home/louca/Documents/Scolaire/M1/Algorithmique de base et python/pygis/script.py'.encode('utf-8')).read())
Layers loaded
Couche Regions
Couche de type QgsMapLayerType.VectorLayer , d'emprise  <QgsRectangle: -7117451.88276480510830879 1357479.18457806529477239, 18764433.087876
60673260689 9961531.59820250608026981>
Contient  26  entités
Liste des attributs : ['ID', 'NAME_2', 'TYPE_2']
Liste des attributs : ['ID', 'NAME_2', 'TYPE_2']
--------
Couche Trees
Couche de type QgsMapLayerType.VectorLayer , d'emprise  <QgsRectangle: -2175230.04128031991422176 1899842.62445650668814778, 4895522.3876856
3885241747 6909266.538424224196895957>
Contient  444  entités
Liste des attributs : ['cat', 'VEGDESC', 'VEG_ID', 'F_CODEDESC', 'F_CODE', 'AREA_KM2']
Liste des attributs : ['cat', 'VEGDESC', 'VEG_ID', 'F_CODEDESC', 'F_CODE', 'AREA_KM2']
--------

```

MANIPULATION D'ENTITÉS ET D'ATTRIBUTS

L'attribut `VEGDESC` de la couche `trees` contient une description du type de végétation de l'entité. Affichons chaque description différente existante.