Formation

Module III

Formation Python pour QGIS 3

2022

Didier LECLERC
Conseiller en Management des Systèmes d'Information Géographique

Département Relation Client

SG / SNUM / UNI / DRC

Cyril HOUISSE
Chef de projet usages et accompagnement décisionnel, datavisualisation, datascience Et intelligence artificielle

SG/SNUM/MSP/DS/GD3IA/PUAD

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

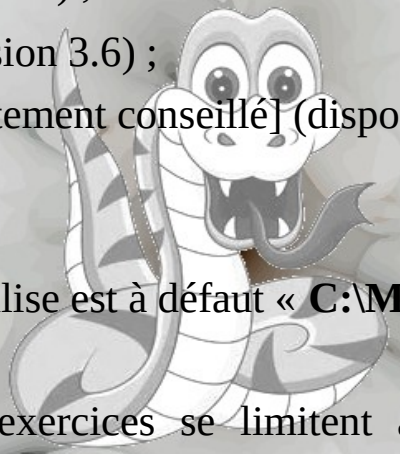
MINISTÈRE
DE LA TRANSITION
ÉCOLOGIQUE ET SOLIDAIRE
www.ecologique-solidaire.gouv.fr

MINISTÈRE
DE LA COHÉSION
DES TERRITOIRES
www.cohesion-territoires.gouv.fr

Pré-requis logiciel :

Pour dérouler correctement cette valise, vous devez au préalable avoir installé :

- **QGIS** [obligatoire] (version 3.x) ;
- **PYTHON** [facultatif] (version 3.6) ;
- **PLUGIN RELOADER** [fortement conseillé] (disponible dans la valise – Kit)



Le répertoire d'installation de la valise est à défaut « **C:\MOC_Q3_STAGE** ».

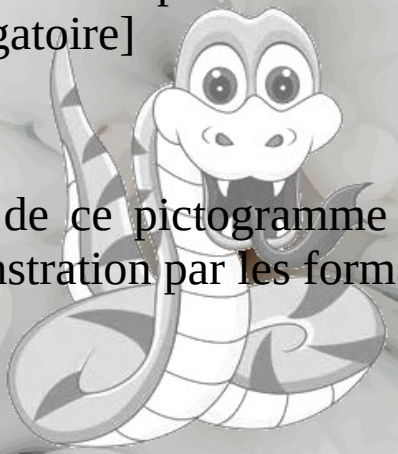
Les jeux de données pour les exercices se limitent à ceux contenus dans le projet « **test_QGIS3.qgs** » placé dans le répertoire « **Donnees** ».

La présente valise repose sur de multiples sources documentaires disponibles sur internet, et sur l'expérience capitalisée autour de développements d'extensions pour QGIS. La plupart des sources sont indiquées. Merci à l'ensemble des contributeurs volontaires ou involontaires pour le présent support.

Conventions sur la valise :

La présence de ce pictogramme indique qu'un exercice doit être réalisé (cf. le répertoire « **Exercices** » de la valise). « ***A faire*** » [Obligatoire]

La présence de ce pictogramme indique qu'un exercice sera fait en démonstration par les formateurs



```
Zone de code exemple
```

Les codes exemples

#Exemple

Les mots ou caractères importants sont en gras de couleur magenta.

Programme

Module I :

- Python, un peu d'histoire …
- Python, généralités …
- Python, les éditeurs ...
- Python, les caractéristiques du langage …
- Python, premiers contacts avec le langage …

Module II :

- Python, un langage orienté objet …
- Python, un langage puissant ...
- Python, gérer les erreurs ...
- Python, de multiples librairies et outils ...
- Python, créer des interfaces graphiques ...
- Python, aller plus loin ...

De Python à QGIS ... (*pont*)

Module III :

- QGIS, un peu d'histoire …
- QGIS, premiers contacts avec l'API …
- QGIS, les interfaces pour Python

Module IV :

- QGIS, créer une extension …
- QGIS, connecter des actions …
- QGIS, manipuler les objets des couches …
- QGIS, aller plus loin ...

, un peu d'histoire ...

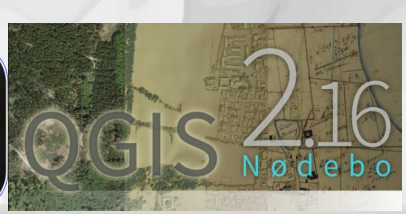
API

A l'origine Quantum GIS était destiné à n'être qu'un outil de visualisation des données de **GRASS** (**G**eographic **R**esources **A**nalysis **S**upport **S**ystem).

Aujourd'hui, ce SIG généraliste est capable de lire et de modifier des données géographiques, de faire des analyses thématiques simples et les mettre en page avec Map composer (logiciel de mise en page intégré).

Depuis la version 0.9, il possède un vrai moteur de scripts basé sur Python. Ceci permet tout à la fois de créer des modules plus simplement qu'en C++, mais aussi de construire de véritables applications (comme on pouvait le faire en C++). Cette possibilité passe par **PyQt**, le pont entre **Python** et la bibliothèque graphique **Qt5**.

Enfin, tout au long de son évolution, QGIS n'a cessé développé une API de plus en plus riche.



QGIS, les caractéristiques de l'API ...

API

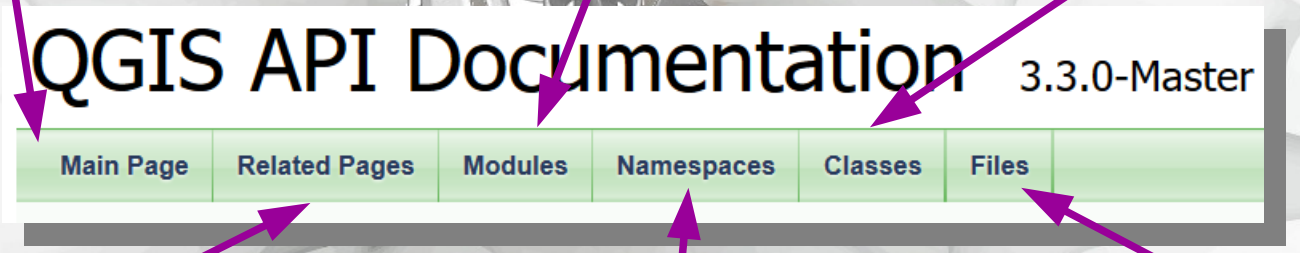
L'API Qgis dispose d'un site d'information propre : http://qgis.org/api/

Le menu principal permet d'accéder par des voies différentes aux informations de l'API (classes, méthodes, attributs, enum, ...)

Accueil,
Anciennes versions,
Bug reporting,...

Par modules

Liste des classes, Structures,
unions et interfaces
avec de brèves descriptions:



Chgts en arrière incompatibles
Modifications de l'utilisateur QGIS 3.0
Documentation rapide QGIS
Liste obsolète

Par espaces de noms
avec de brèves descriptions

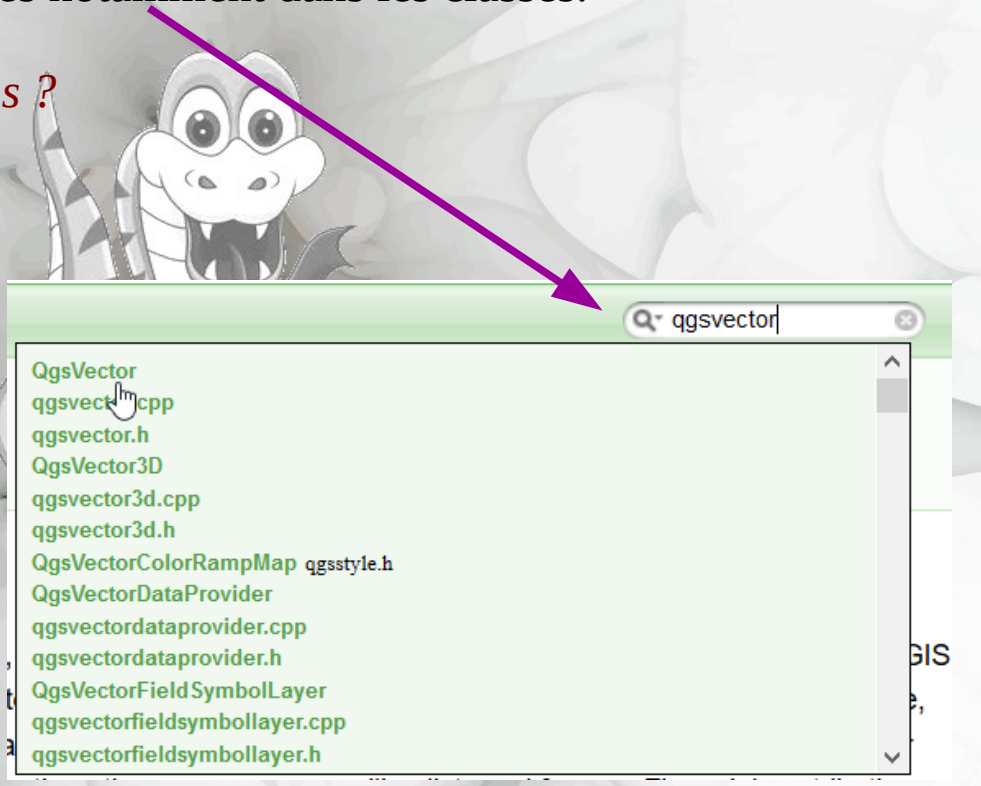
Liste des fichiers
avec de brèves descriptions

QGIS, les caractéristiques de l'API ...

API

Enfin, un moteur de recherche, situé en haut à droite, permet la saisie de mots clefs pour faciliter les entrées notamment dans les classes.

A privilègier ? Qu'en pensez-vous ?



QGIS, les caractéristiques de l'API ...

API

L'API de **QGIS** reste une matière vivante !

Reportez-vous aux différents sites d'information sur l'évolution de l'API :

Tous les changements dans cette dernière avec l'arrivée de la 3.x de QGIS :

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

Comment migrer vos outils (plugins) vers la 3.x :

https://github.com/qgis/QGIS/wiki/Plugin-migration-to-QGIS-3

QGIS, premiers contacts avec l'API ...

API

Affichage Historique Marque-pages Outils ?

QGIS API Documentation: Class Ind

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

Ludique Comptes SMQ = Qualité Outils quotidiens GéoIDE - GT Mélanie Web 2 Google Traduction Google Maps Gmail

QGIS API Documentation 3.3.0-Master (12c8f39)

Main Page | Related Pages | Modules | Namespaces | Classes | Files

Class Index

2 | 3 | _ | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | z

2

Qgs25DRenderer
Qgs25DRendererWidget

3

Qgs3DMapScene
Qgs3DMapSettings
Qgs3DRendererAbstractMetadata
Qgs3DRendererRegistry
Qgs3DUtils

_

_chain (pal)
_cHullBox (pal)
_CohenSutherland
_elementary_transformation (pal)

QgsDialog
QgsDirectoryItem
QgsDirectoryParamWidget
QgsDistanceArea
QgsDockWidget
QgsDoubleBoxScaleBarRenderer
QgsDoubleSpinBox
QgsDrawSourceEffect
QgsDrawSourceWidget
QgsDropShadowEffect
QgsDualView
QgsDummyConfigDlg
QgsVectorLayerUtils::QgsDuplicateFeatureContext
QgsDxfExport
QgsDxfLabelProvider
QgsDxfPaintDevice
QgsDxfPaintEngine

QgsLayerTreeModel
QgsLayerTreeModelLegendNode
QgsLayerTreeNode
QgsLayerTreeRegistryBridge
QgsLayerTreeUtils
QgsLayerTreeView
QgsLayerTreeViewDefaultActions
QgsLayerTreeViewIndicator
QgsLayerTreeViewMenuProvider
QgsLayout
QgsLayoutAligner
QgsLayoutAtlas
QgsLayoutEffect
QgsLayoutExporter
QgsLayoutFrame
QgsLayoutGridSettings
QgsLayoutGuide

QgsPaintEngineHack
QgsPainting
QgsPalettedRasterRenderer
QgsPalettedRendererWidget
QgsPalLabeling
QgsPalLayerSettings
QgsPanelWidget
QgsPanelWidgetStack
QgsPanelWidgetWrapper
QgsPasswordLineEdit
QgsPathResolver
QgsPenCapStyleComboBox
QgsPenJoinStyleComboBox
QgsPenStyleComboBox
QgsPhongMaterialSettings
QgsPieDiagram
QgsPixmapLabel

QGIS, premiers contacts avec l'API ...

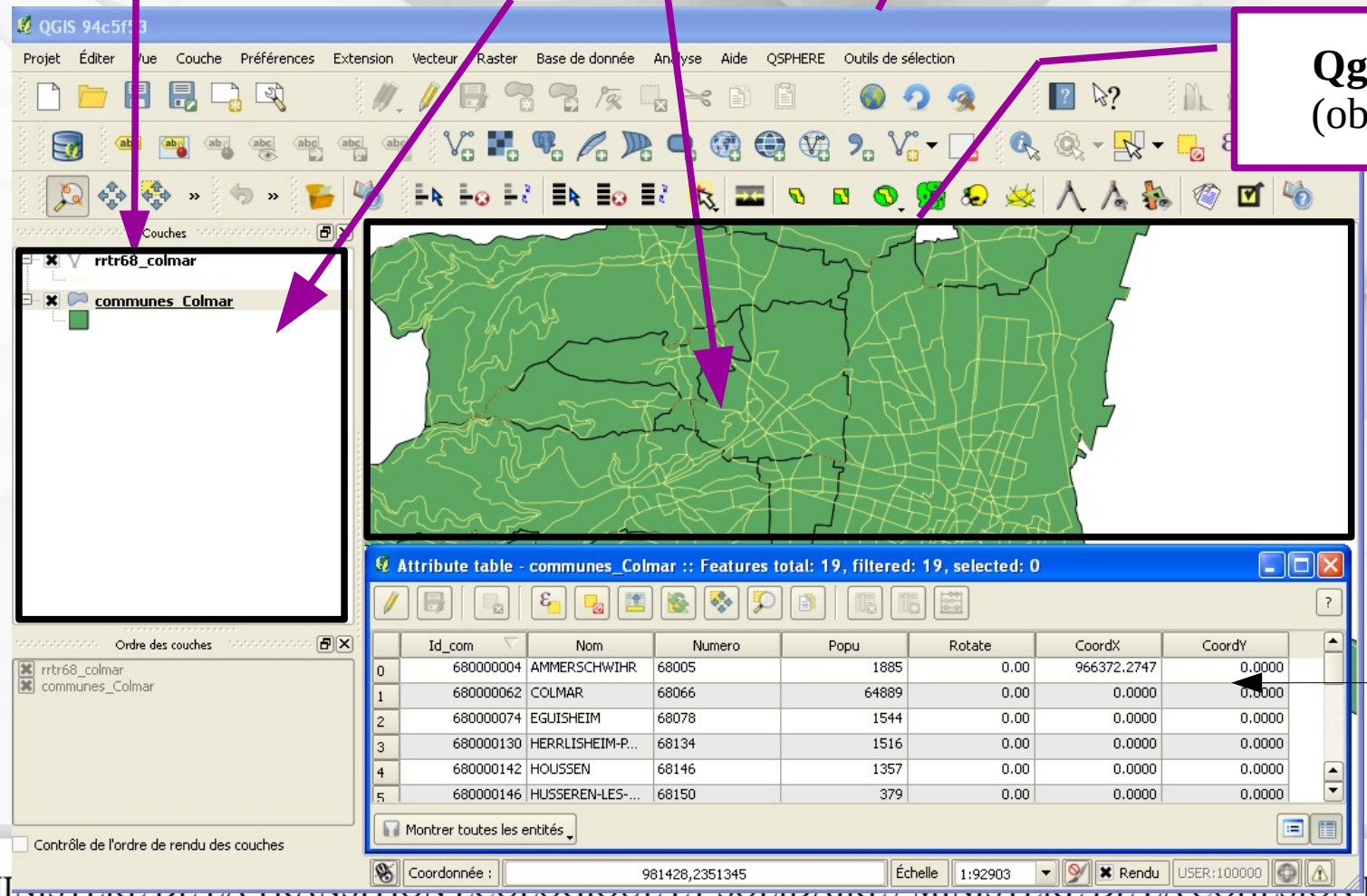
API

QgsLegendInterface
(objet conteneur)

QgsVectorLayer
QgsRasterLayer
....
(objets contenus)

QgisInterface
•showAttributeTable(vLayer)

QgsMapCanvas
(objet conteneur)



QGIS, premiers contacts avec l'API ...

Démonstration

Parcourir l'intégralité des fonctions ou des relations entre les classes pour l'ensemble de l'API de QGIS n'est pas possible.

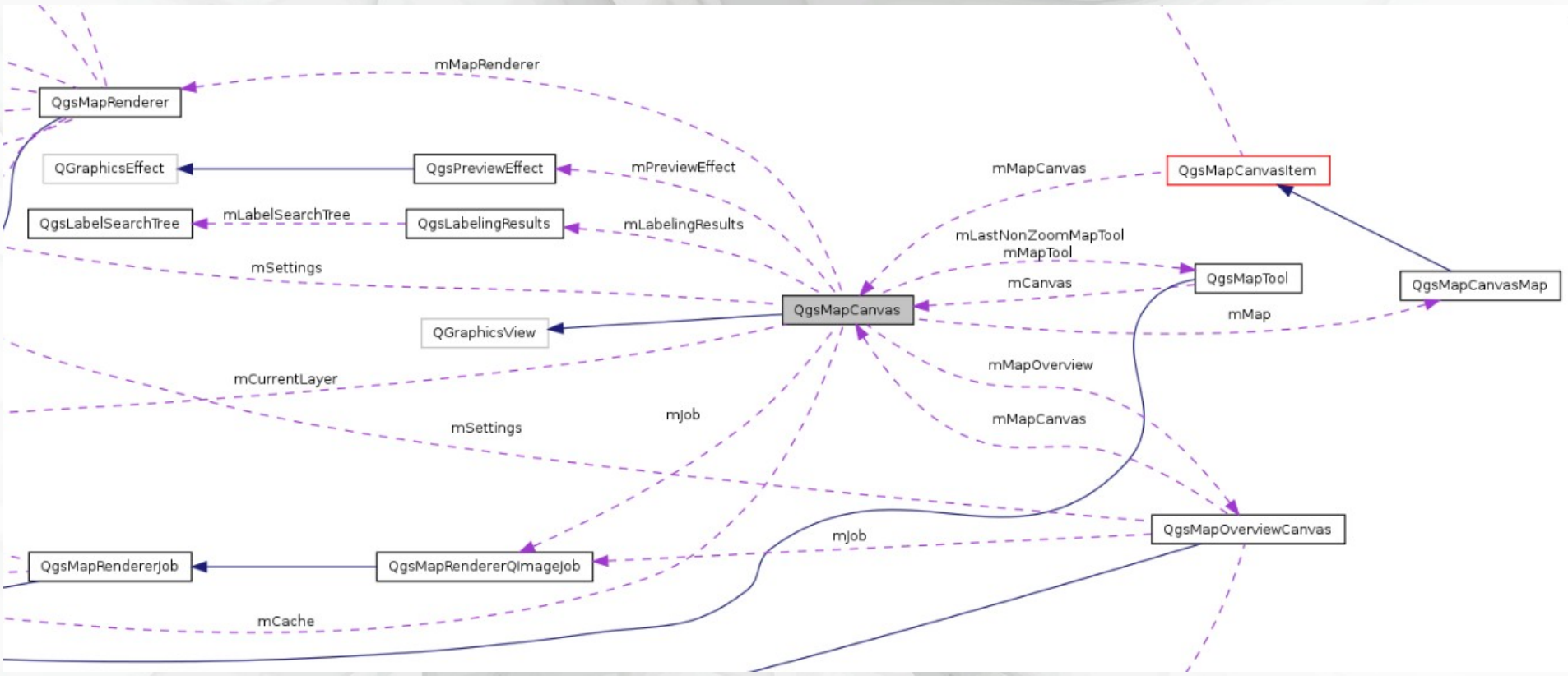
Prenons un peu de temps pour vous montrer comment naviguer au travers de l'interface de l'API.

Démonstration ….

QGIS, premiers contacts avec l'API ...

API

Une partie du diagramme de collaboration de la classe QgsMapCanvas
Api Version 2.4



Le diagramme de collaboration de la classe **QgsMapCanvas**

QGIS, premiers contacts avec l'API ...

Syntaxe

La majeure partie des **classes** présente un (à défaut) ou plusieurs **constructeurs.**
Exemples :

```
#Initialisation d'une instance de la classe géométrie de QGIS
zGeom = QgsGeometry()
#Initialisation d'une instance de la classe enregistrement (géométrie et attributs)
zFeature = QgsFeature ()

#Initialisation d'une instance de la classe pour le stockage des informations
#sur une projection
zCrs = QgsCoordinateReferenceSystem()

#Initialisation d'une instance de la classe pour le stockage des informations
#sur une projection EPSG 2154 (Lambert 93)
zCrs = QgsCoordinateReferenceSystem(2154)
```

Une fois l'instance initialisée, on peut accéder aux fonctions de la classe.

```
print(zCrs.description())
print(zCrs.toProj4())
```

QGIS, premiers contacts avec l'API ...

Syntaxe

zCrs = **QgsCoordinateReferenceSystem**(2154)

zCrs.**toProj4**()

zCrs.**description**()

```
>>> zCrs = QgsCoordinateReferenceSystem()
>>> zCrs
<qgis._core.QgsCoordinateReferenceSystem object at 0x000002D4D786EE58>
>>> zCrs.toProj4()
''
>>> zCrs = QgsCoordinateReferenceSystem(2154)
>>> zCrs.toProj4()
'+proj=lcc +lat_1=49 +lat_2=44 +lat_0=46.5 +lon_0=3 +x_0=700000 +y_0=6600000 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs'
>>> zCrs.description()
'RGF93 / Lambert-93'
```

QGIS, les interfaces pour Python ...

Syntaxe

Quelques exemples de lignes pour des scripts Python. QGIS depuis la console :

- qgis.utils.iface.activeLayer().name()
- qgis.utils.iface.mapCanvas().layers()

Exemple d'alternative :

```
ziface = qgis.utils.iface
zList = ziface.mapCanvas().layers()
print(zList)
```

Pour exploiter chaque élément de la liste, une boucle s'impose !

```
for i in range(len(zList)) :
    print(str(zList[i].name()))
```

Si la console affiche « **...** » pour le prompt, validez de nouveau (touche **entrée**) pour visualiser le résultat.

Utilisez les flèches haute et basse pour naviguer dans les instructions déjà passées dans la console.

QGIS, les interfaces pour Python ...

Exercices

A vous, faites vous plaisir,

premiers contacts avec la console Python de Qgis

```
ziface = qgis.utils.iface
zList = ziface.mapCanvas().layers()
print(zList)
```

Pour exploiter chaque membre de la liste, une boucle s'impose !

```
for i in range(len(zList)) :
    print(str(zList[i].name()))
```

N'oubliez pas de charger le projet
pré installé sur votre poste :

C:/.../PythonPourQGIS/Donnees/test_QGIS3.qgs

Console Python

```
1  Console Python
2  Utilisez iface pour accéder à l'interface de l'API QGIS ou tapez help(iface) po
   ur plus d'informations
3  >>> ziface = qgis.utils.iface
4  >>> zList = ziface.mapCanvas().layers()
5  >>> print(zList)
6  [<qgis._core.QgsVectorLayer object at 0x00000264D786ECA8>, <qgis._core.QgsVecto
   rLayer object at 0x00000264D786EEE8>, <qgis._core.QgsVectorLayer object at 0x00
   000264D786EDC8>, <qgis._core.QgsVectorLayer object at 0x00000264D786EF78>, <qgi
   s._core.QgsVectorLayer object at 0x000002648EFCB048>]
7  >>> for i in range(len(zList)) :
8  ...     print(str(zList[i].name()))
9  zone_troncon_hydrographie
10 zone_commune_densite
11 zone_commune_autres_ponctuels
12 zone_commune_autres
13 n_parcelle_travail_dgi_chasse_36131
14
```

QGIS, les interfaces pour Python ...

Syntaxe

Quelques exemples de lignes pour des scripts Python depuis la console de QGIS :

QgisInterface

- qgis.utils.iface.activeLayer().name()
- qgis.utils.iface.addProject(***PathOfTheProject ****)

(*) exemple : *'C:/.../PythonPourQGIS/Donnees/test_QGIS3.qgs'*

- qgis.utils.iface.addRasterLayer (***PathOfTheLayer***, ***NameForTheLayer***) ([1])
- qgis.utils.iface.addVectorLayer (***PathOfTheLayer***, ***NameForTheLayer, 'ogr'***) ([1])
- qgis.utils.iface.showAttributeTable(TheLayer *)

(*) exemple : *qgis.utils.iface.activeLayer()*

- qgis.utils.iface.showLayerProperties(TheLayer *)
- qgis.utils.iface.zoomFull()
- qgis.utils.iface.zoomToPrevious()
- ...
- qgis.utils.iface.mapCanvas()

QgsMapLayerRegistry.*instance().addMapLayer(zLayer)*
n'existe plus et est remplacé par
QgsProject.instance()*.addMapLayer(zLayer)*

QGIS, les interfaces pour Python ...

Syntaxe

Quelques exemples de lignes pour des scripts Python depuis la console de QGIS :

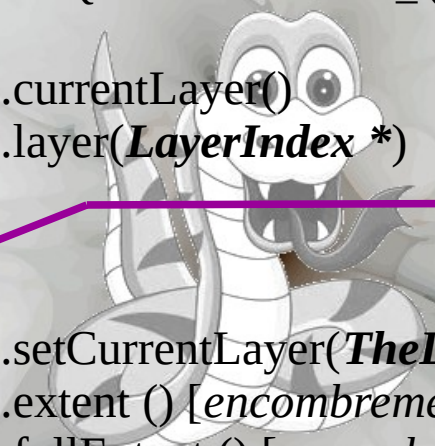
- qgis.utils.iface.mapCanvas().saveAsImage (***PathForImage*** *)

  (*) exemple : *'C:/.../PythonPourQGIS/Donnees/test_QGIS.png'*

- qgis.utils.iface.mapCanvas().currentLayer()
- qgis.utils.iface.mapCanvas().layer(***LayerIndex*** *)

  (*) exemple : *5*

- qgis.utils.iface.mapCanvas().setCurrentLayer(***TheLayer*** *)
- qgis.utils.iface.mapCanvas().extent () [*encombrement courant*]
- qgis.utils.iface.mapCanvas().fullExtent () [*encombrement général*]
- qgis.utils.iface.mapCanvas().zoomIn()
- qgis.utils.iface.mapCanvas().zoomOut()
- qgis.utils.iface.mapCanvas().zoomToFullExtent()
- ...
- qgis.utils.iface.mapCanvas().layerCount ()
- qgis.utils.iface.mapCanvas().layers()



QgsMapCanvas

QGIS, les interfaces pour Python ...

Syntaxe

Quelques **exemples** de lignes pour des scripts Python depuis la console QGIS :

Il faudra pointer sur une ressource vecteur.

- zlayer = qgis.utils.iface.addVectorLayer (***PathOfTheLayer ****, ***NameForTheLayer, 'ogr'***)

  (*) exemple : *'c:\repInstall\Donnees\couches\zone_commune_autres.shp', 'Communes', 'ogr'*

- zlayer.metadata()
- Zlayer.htmlMetadata()
- zlayer.capabilitiesString ()
- zlayer.dataProvider ()
- zlayer.featureCount ()
- zlayer.geometryType ()
- zlayer.invertSelection ()
- zlayer.isEditable ()
- zlayer.isReadOnly ()
- zlayer.maximumValue (***TheColumnIndex ****) (*) exemple : ***5***
- ...

QgsVectorLayer

QGIS, les interfaces pour Python ...

Exercices

A partir des commandes vues à la diapo précédente, récupérer les métadonnées de chaque couche de la fenêtre carte.

Copier le résultat de la console Python dans un fichier texte sauvegardé sous le nom « **resultat.html** »

zone_troncon_hydrographie :

Information du fournisseur

Original	zone_troncon_hydrographie
Nom	zone_troncon_hydrographie
Source	D:/CMSIG/3 - Formation/Python/Valise_Q3_2019/donnees/couches/zone_troncon_hydrographie.shp
Stockage	ESRI Shapefile
Commentaire	
Encodage	ISO-8859-1
Géométrie	Line (MultiLineString)
SCR	EPSG:2154 - RGF93 / Lambert-93 - Projeté
Emprise	462542.00000000000000000,6730962.0000000000000000 : 477934.00000000000000000,6749801.0000000000000000
Unité	mètres
Décompte d'entités	138

	Type	Longueur
	String	10
	String	12
	String	21
	String	24
	String	13
	String	8
	String	127
	String	15
	String	6

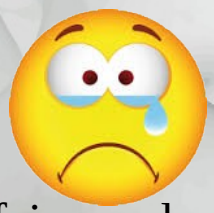
Contacts

Encore aucun contact.

QGIS, les interfaces pour Python ...

Démonstration

Super, mais un peu triste non ?



Est-ce que je ne pourrais pas essayer de faire quelque chose de plus ‘sexy’ ? Enfin, personnalisable.

Nom : zone_troncon_hydrographie

zone_troncon_hydrographie :

Information du fournisseur

Original	zone_troncon_hydrographie
Nom	zone_troncon_hy
Source	D:/CMSIG/3 - F
Stockage	ESRI Shapefile
Commentaire	
Encodage	ISO-8859-1
Géométrie	Line (MultiLineS
SCR	EPSG:2154 - RG
Emprise	462542.0000000
Unité	mètres
Décompte d'entités	138

Champs

Compte 9

id_bdcarto	String	1
etat	String	6
largeur		
nature		
navigable		
pos_sol		
toponyme		
sens	String	1
classe	String	6

Contacts

Encore aucun contact.

Information du fournisseur

Original	zone_troncon_hydrographie
Nom	zone_troncon_hydrographie
Source	D:/CMSIG/3 - Formation/Python/Valise_Q3_2019/Donnees/couches/zone_troncon_hydrographie.shp
Stockage	ESRI Shapefile
Commentaire	
Encodage	ISO-8859-1
Géométrie	Line (MultiLineString)
SCR	EPSG:2154 - RGF93 / Laml
Emprise	462542.0000000000000000,6
Unité	mètres
Décompte d'entités	138

Champs

Compte 9

Champ	Type	Longueur
id_bdcarto	String	10
etat	String	12
largeur	String	21
nature	String	24
navigable	String	13
pos_sol	String	8
toponyme	String	127
sens	String	15
classe	String	6

QGIS, les interfaces pour Python ...

Démonstration

Allez on remet les mains dans la machine à pythoner

Code source du fichier ‘result_prog_perso.html’

Nom : zone_troncon_hydrographie

```
<body>
<h1>Information du fournisseur</h1>
<hr>
<table class="list-view">
<tr><td class="highlight">Original</td><td>zone_troncon_hydrographie</td></tr>
<tr><td class="highlight">Nom</td><td>zone_troncon_hydrographie</td></tr>
<tr><td class="highlight">Sou
Formation/Python/Valise_Q3_
<tr><td class="highlight">Sto
<tr><td class="highlight">Cor
<tr><td class="highlight">Enc
<tr><td class="highlight">Géo
<tr><td class="highlight">SCl
<tr><td
class="highlight">Emprise</td
477934.0000000000000000,67
<tr><td class="highlight">Uni
<tr><td class="highlight">Déc
```

```
<br><table width="100%" class="tabular-view">
<tr><th>Champ</th><th>Type</th><th>Longueur</th><th>Précision</th><th>Commentaire</th></tr>
<tr ><td>id_bdcarto</td><td>String</td><td>10</td><td>0</td><td></td></tr>
<tr class="odd-row"><td>nom_comm</td><td>String</td><td>50</td><td>0</td><td></td></tr>
<tr ><td>insee_comm</td><td>String</td><td>5</td><td>0</td><td></td></tr>
<tr class="odd-row"><td>statut</td><td>String</td><td>20</td><td>0</td><td></td></tr>
<tr ><td>x_commune</td><td>String</td><td>10</td><td>0</td><td></td></tr>
<tr class="odd-row"><td>y_commune</td><td>String</td><td>10</td><td>0</td><td></td></tr>
<tr ><td>superficie</td><td>String</td><td>10</td><td>0</td><td></td></tr>
<tr class="odd-row"><td>population</td><td>String</td><td>10</td><td>0</td><td></td></tr>
<tr ><td>insee_cant</td><td>String</td><td>2</td><td>0</td><td></td></tr>
<tr class="odd-row"><td>insee_arr</td><td>String</td><td>1</td><td>0</td><td></td></tr>
<tr ><td>nom_dept</td><td>String</td><td>30</td><td>0</td><td></td></tr>
<tr class="odd-row"><td>insee_dept</td><td>String</td><td>2</td><td>0</td><td></td></tr>
<tr ><td>nom_region</td><td>String</td><td>30</td><td>0</td><td></td></tr>
<tr class="odd-row"><td>insee_reg</td><td>String</td><td>2</td><td>0</td><td></td></tr>
<tr ><td>densite</td><td>String</td><td>10</td><td>0</td><td></td></tr>
</table>
```

QGIS, les interfaces pour Python ...

Démonstration

Allez on remet les mains dans la machine à pythoner

La ruse …..

- Encapsuler le nom de la couche avec un style (Html / CSS)

Avec : str(MesCouches[i].name()) = zone_troncon_hydrographie

```
MonDebutStyle = "<html><body><meta http-equiv='Content-Type' content='text/html; charset=utf-8' />"
MonDebutStyle += "<font color='#0000FF' size=6 face = 'Comic sans serif'>"

MonFinStyle = "</font></body></html>"
```

```
R1 =        MonDebutStyle + " Nom : " + str(MesCouches[i].name()) + MonFinStyle)
```

Attention à l'encodage de votre page

- Recharger une nouvelle définition de la classe de style (**list-view** et **tabular-view**) de la balise <table>

Avec : MonListView = "<style>.list-view {font-weight: bold;color: #FF0000; background-color: #cccccc;padding: 4px; border: 2px solid #FF0000}</style>"

```
MonTabularView = "<style>.tabular-view {color: #547ca0; background-color: #cccccc;}</style>"
```

```
R2 =        MonListView + MonTabularView + MesCouches[i].htmlMetadata()
```

Il ne reste plus qu'à écrire : Monfichier.write(r1 + r2)

QGIS, les interfaces pour Python ...

A vous de jouer …..... la fonction qui va bien

Exercices

```
def Write_File_Meta(WithStyle,encodePage):

    myPath = "D:\\CMSIG\\3 - Formation\\Python\\Valise_Q3_2019\\Exercices\\Exo_Console_python_layers"
    myNomFileHtml = "result_prog_perso.html"

    MyFileOupout = open(myPath + "\\" + myNomFileHtml,'w')

    if WithStyle :
      MonDebutStyle = "<html><body><meta http-equiv='Content-Type' content='text/html; charset=" + encodePage + "' />"
      MonDebutStyle += "<font color='#0000FF' size=6 face = 'Comic sans serif'>"
      MonFinStyle = "</font></body></html>"
      Monlistview = "<style>.list-view {font-weight: bold;color: #FF0000; background-color: #cccccc;padding: 4px; border: 2px solid #FF0000}</style>"
      Montabularview = "<style>.tabular-view {color: #547ca0; background-color: #cccccc;}</style>"
    else :
      MonDebutStyle = ""
      MonFinStyle = ""
      Monlistview = ""
      Montabularview = ""

    for i in range(len(MesCouches)) :
      r1 = (MonDebutStyle + " Nom : " + str(MesCouches[i].name()) + MonFinStyle)
      r2 = (Monlistview + Montabularview + MesCouches[i].htmlMetadata())
      MyFileOupout.write(r1 + r2)

    MyFileOupout.close()
```

Lancer la fonction avec :

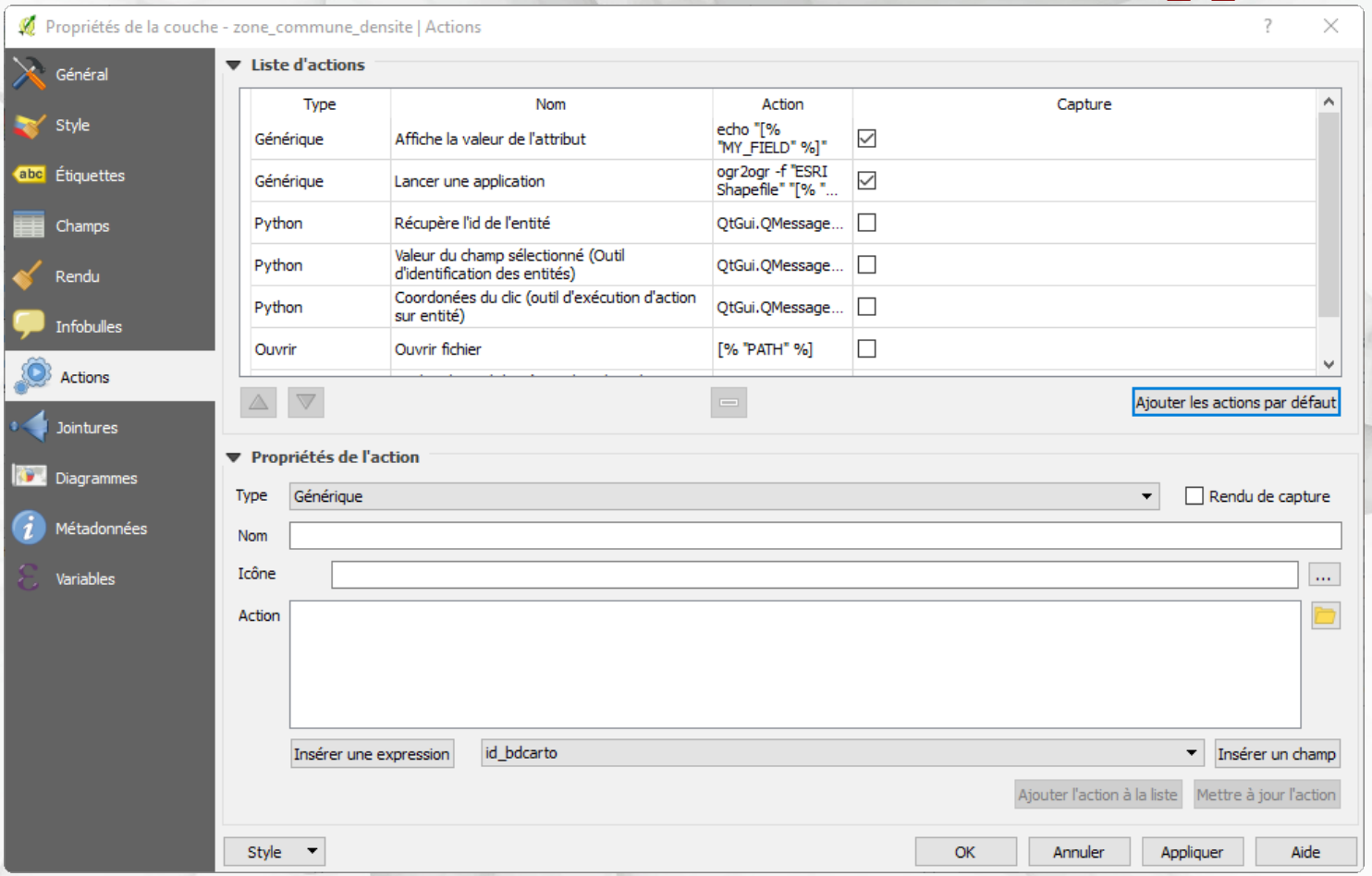
WithStyle = True
encodePage = "iso-8859-1"
Write_File_Meta(WithStyle,encodePage)

QGIS, les interfaces pour Python ...

Concepts

Il est également possible sous QGIS de paramétrer des actions de type Python dans l'onglet « actions » depuis les propriétés de la couche :

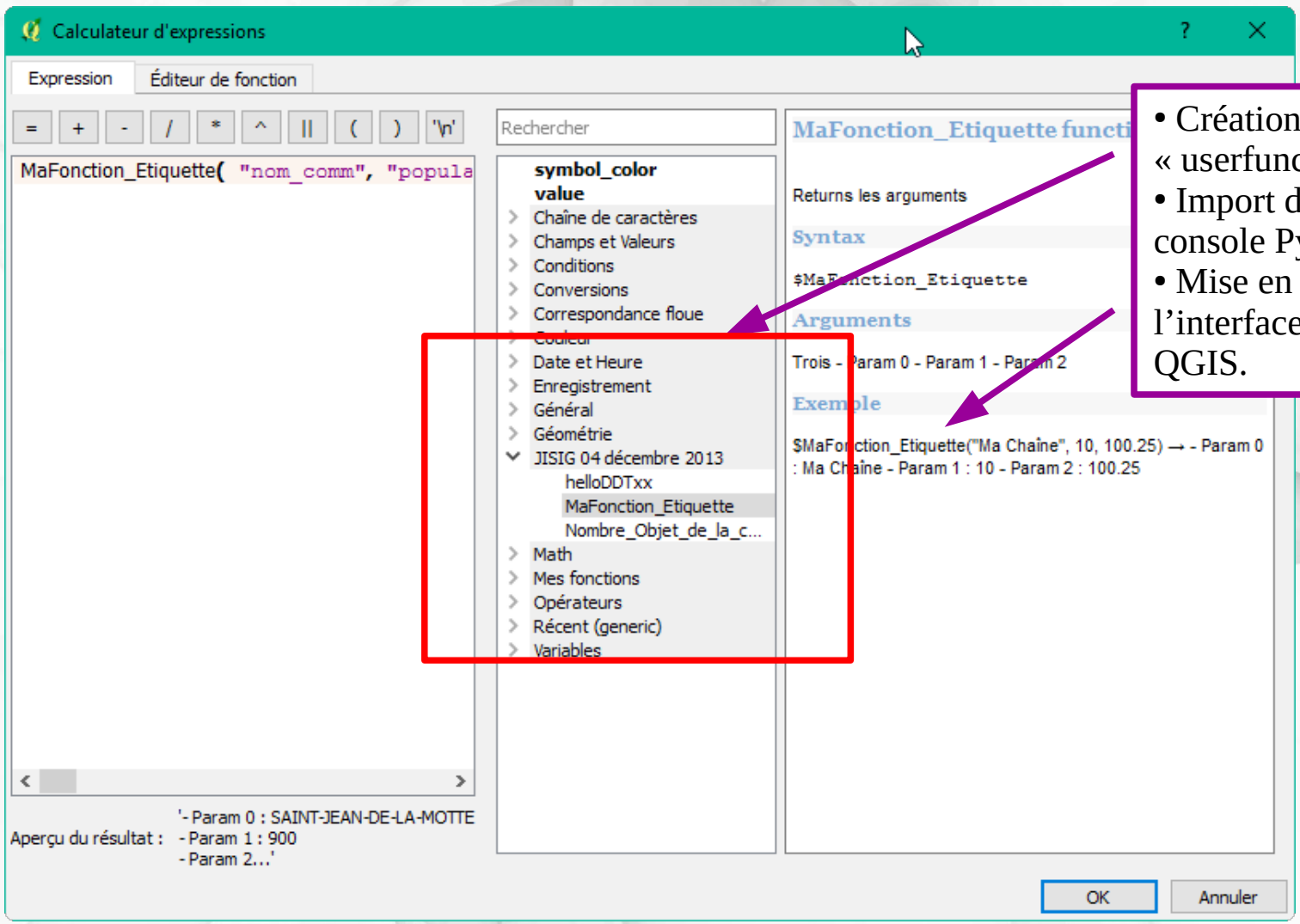
Rappel



QGIS, les interfaces pour Python ... *Rappel*

Concepts

Depuis la version 2.0, il est possible d'implémenter des fonctions en Python dans les expressions :



- Création d'un fichier « userfunctions.py » ;
- Import du module depuis la console Python ;
- Mise en œuvre dans l'interface Expression de QGIS.

Formation Python pour QGIS 3

Fin du module

Merci de votre attention !!!!