

# INITIATION À L'UTILISATION DU LANGAGE DE PROGRAMMATION PYTHON

Support de cours





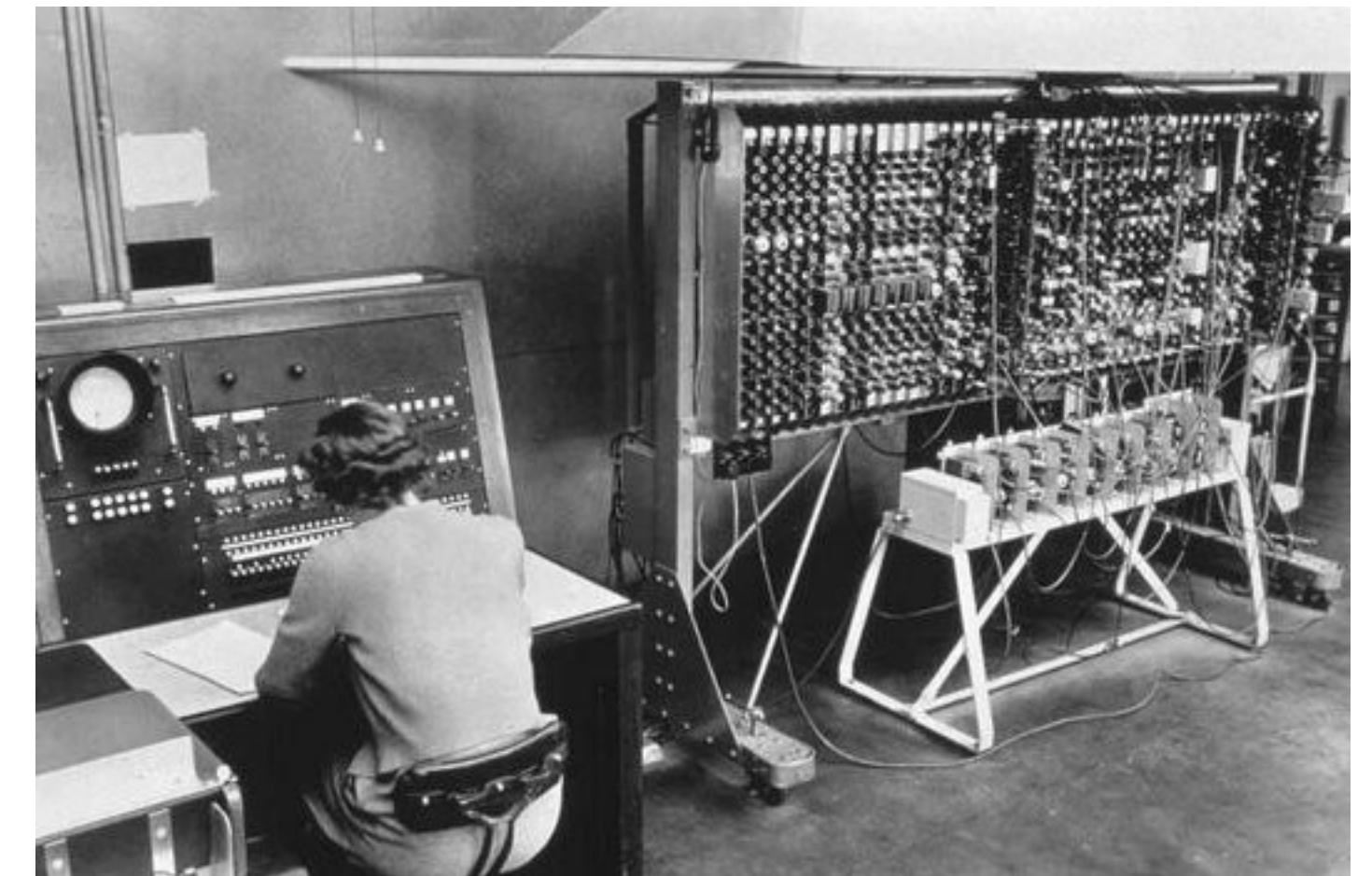
---

# Introduction générale

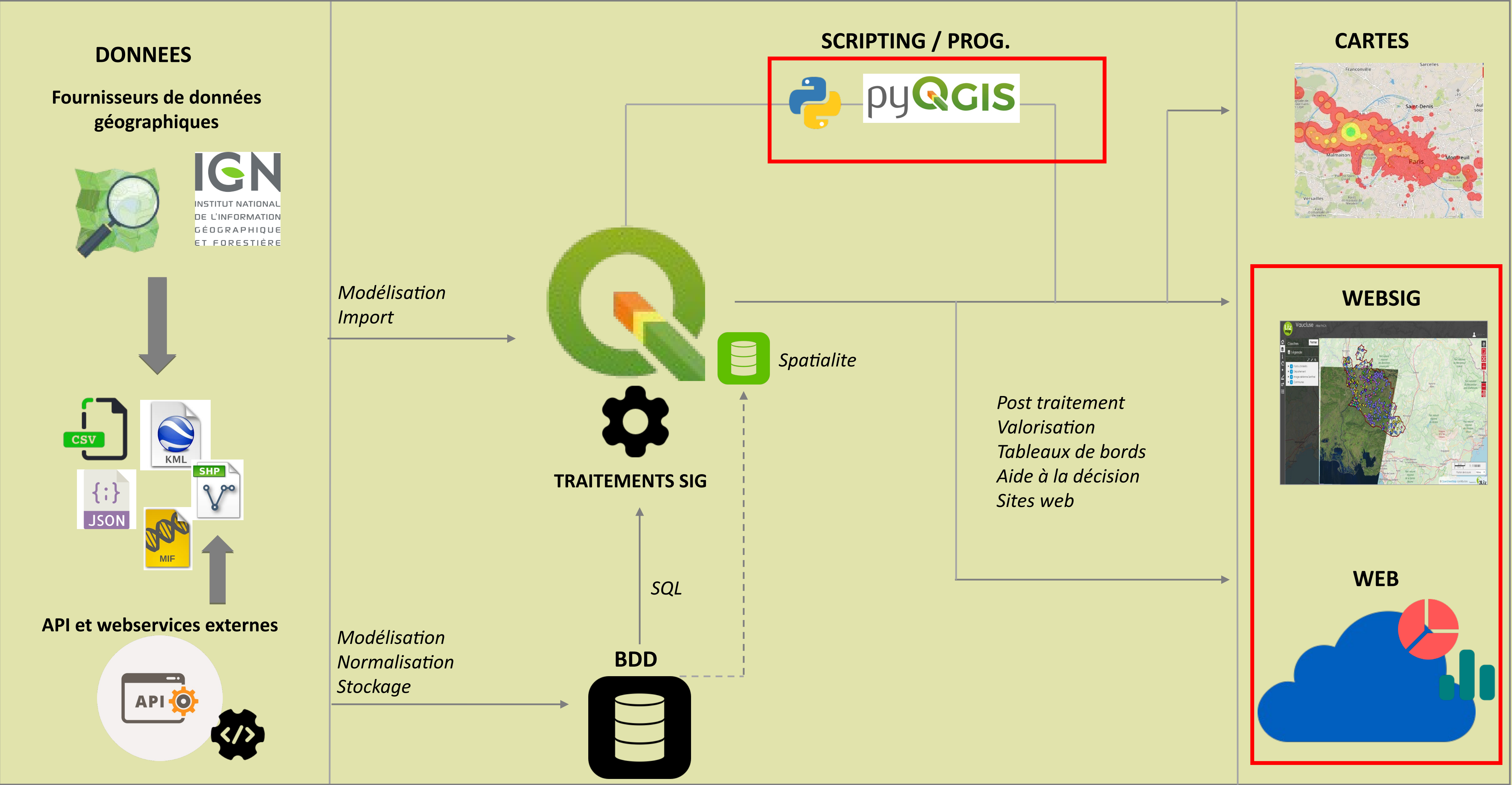
---

# </> Qu'est-ce que la programmation ?

- La programmation désigne l'ensemble des activités qui permettent **l'écriture de programmes informatiques** à l'aide d'un ou plusieurs langages.
- Le concept de l'ordinateur programmable date de **1937** : c'est la « machine de Turing ». Première utilisation pendant la 2<sup>ème</sup> guerre mondiale pour déchiffrer rapidement les messages allemands produits par « Enigma ».
- **Beaucoup d'usages variés et de possibilités** : mise en place de sites web, systèmes embarqués, domotique, développement d'applications web/mobiles, gestion de bases de données, administration de serveurs, automatisation de traitements SIG, cybersécurité/cryptographie, etc.
- Les premiers langages datent de la **fin des années 50** (Cobol & Fortran) ; explosion dans les années 90 avec le développement d'Internet



# COMPOSANTES D'UN SYSTÈME D'INFORMATION GEOGRAPHIQUE






# Objectifs du cours

## Python/PyQGIS

- A** Vous initier au langage de programmation Python
- B** Automatiser la manipulation de fichiers, l'analyse de données (graphiques), la consommation d'APIs, les liens avec les bases de données, etc
- C** Interagir avec QGIS via l'API et les fonctions PyQGIS

# Organisation du cours


6 demies-journées | 18h.

 Découverte des bases du langage Python (syntaxe, conditions, boucles, etc)

 Manipulation de données, traitements graphiques de base & APIs

- Utilisation de packages Python
- Développements de scripts automatisés

 QGIS & Python

-  Intégrer les algorithmes QGIS dans des scripts Python
- Découverte de l'API PyQGIS

 Développement d'un plugin QGIS

- Création de la maquette du plugin
- Association maquette / Algorithmes QGIS / PyQGIS

Cours 1 à 4

Cours 5 et 6

---

TD n°1

---

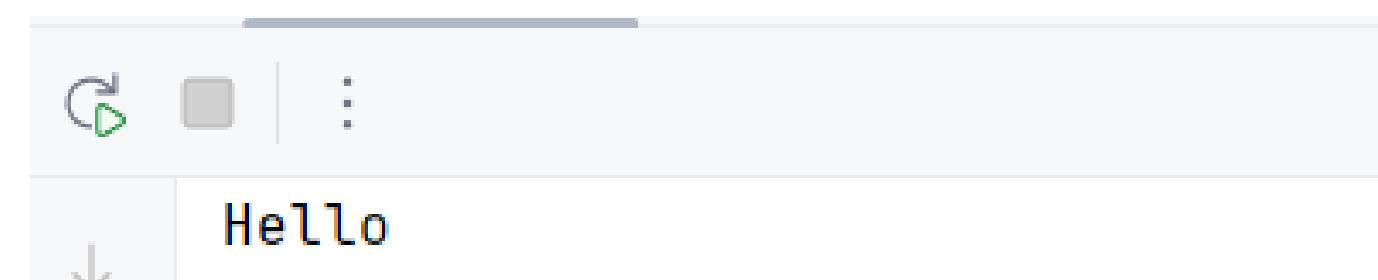
Découverte des bases du  
langage Python

# Qu'est-ce que Python ?

- Langage de programmation open-source conçu en [1991](#). Version 3.x depuis 2008
- Langage dit « de haut niveau » permettant d'exécuter un programme de façon indépendante des caractéristiques d'un ordinateur
- D'un langage de script « classique » l'utilisation du Python s'est progressivement diversifiée : traitement de données & calcul numérique, API web et logicielle, développement de jeux vidéos, IA, etc
- A également évolué vers la mise en place d'applications web au même titre que d'autres langages nativement orientés web (PHP, Javascript entre autres)
- Langage de programmation le plus populaire selon l'indice [TIOBE](#) depuis oct. 2021 (indicateur de la popularité des langages de programmation)
- Chaque fichier développé en langage Python possède l'extension .py (ou.pyc)

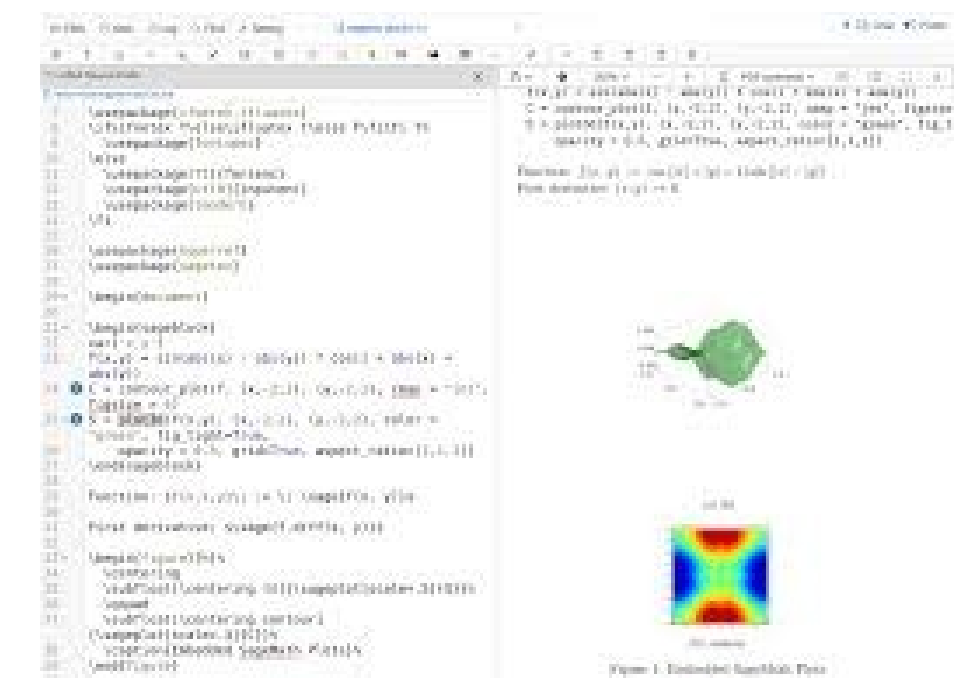
```
print("Hello");
```

Renvoie :





# Applications en Python



# Objectifs du TD

## n°1

- Installer votre environnement de développement (**I.D.E**) « PyCharm »
- Découvrir les concepts clés du langage Python (entre autres) :
  - Les variables
  - Les conditions
  - Les listes, tuples et dictionnaires
  - Les boucles
  - Les fonctions
  - La gestion des exceptions
  - Les packages
- Mobiliser vos nouvelles connaissances dans un exercice bilan





# Rappels (1/2)

- Utiliser des **points-virgules** pour chacune de vos instructions (excepté l'import des modules) :

```
monTuple = ('analyse spatiale', 'bdd', 'developpement', 'carto', 'web');  
print (monTuple);
```

- Indenter** votre code à l'intérieur de conditions et de boucles :

```
for i in sig:  
    print ('Le sujet sélectionné est : ' + i);
```

- Attention à la différence entre « = » et « == » :

➤ « = » : on affecte une valeur à une variable

```
motdepasse = 'qtV1$00l_a';
```

➤ « == » : on teste si une variable est égale à une valeur (condition)

```
if motdepasse == 'qtV1$00l_a':  
    print('Ok, vous pouvez entrer');
```

- Dès que cela est possible, utiliser **la gestion des exceptions** pour gérer les erreurs de votre code :

```
for i in sig:  
    try:  
        print (i);  
    except ValueError as err:  
        print (err);
```

# Rappels (2/2)

- Garder en tête l'intérêt des **listes**, **tuples** et **dictionnaires** : structures souvent utilisées en Python car elles permettent de stocker des données.

## Liste

	Valeur
0	analyse spatiale
1	bdd
2	developpement
3	carto
4	web

## Tuple

	Valeur
0	analyse spatiale
1	bdd
2	developpement
3	carto
4	web

## Dictionnaire simple

Clé	Valeur
nom	Avignon
departement	Vaucluse
population	92000
rang	48

↑ key      ↑ value

items

## Dictionnaire multi-dimensionnel

	Clé	Valeur
Avignon	departement	Vaucluse
	population	92000
	rang	48
Marseille	departement	Bouches-du-Rhône
	population	860000
	rang	2
Nice	departement	Alpes-Maritime
	population	343000
	rang	5

↑ key      ↑ value

keys

items

```
for i in multiVille.keys():  
    for key, value in multiVille[i].items():  
        print (key + " : " + str(value))
```



---

TD n°2

---

Automatiser la manipulation et  
le traitement de fichiers et  
APIs

# Objectifs du TD

## n°2

- 1) Créer des **scripts automatisés** pour :
  - Manipuler et restructurer des fichiers (SIG ou tabulaires) avec le package « Pandas »
  - Automatiser le traitement (graphique) de fichiers avec le package « Seaborn »
  - Consommer et exploiter des APIs : exemple de l'API Adresse ([data.gouv.fr](https://data.gouv.fr))
- 2) Mobiliser vos connaissances dans deux exercices bilan

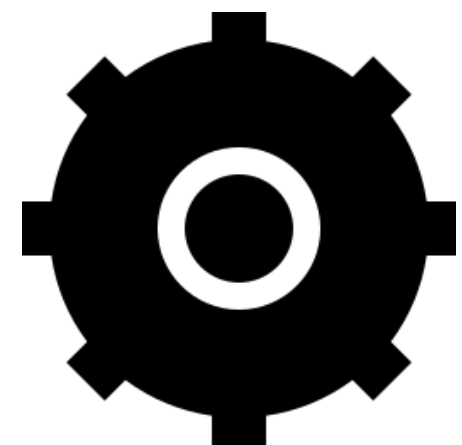




# Pandas & Seaborn

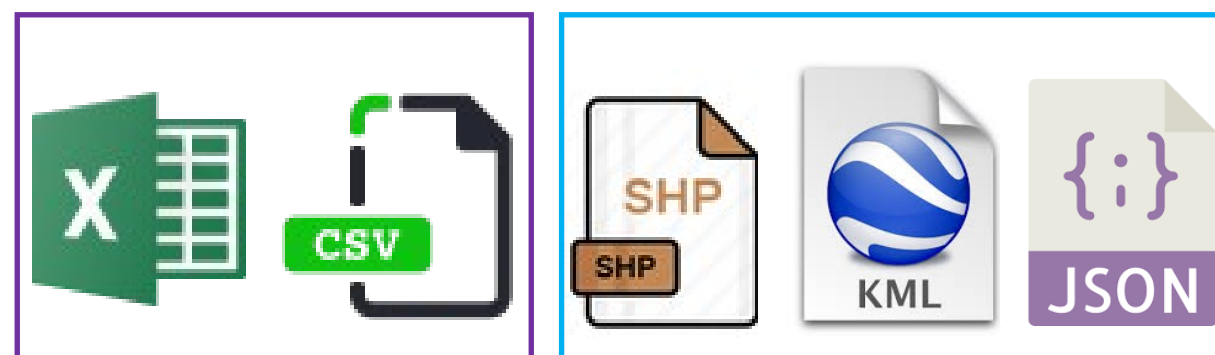


Pandas/  
Géopandas

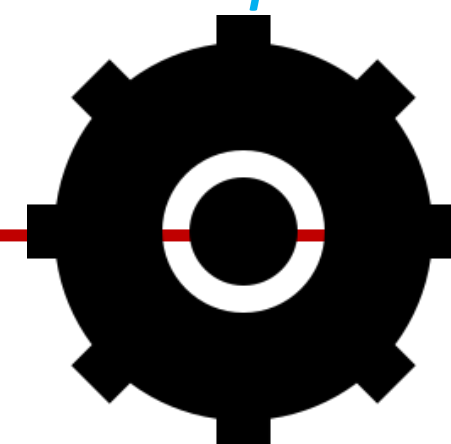


*script*

- Import fichier/API source (« input »)
- Affichage colonnes / lignes spécifiques
- Suppression colonnes
- Requêtage (texte, numérique, calculs)
- Extractions du fichier source
- Suppression de valeurs vides/nulles
- Jointures fichiers
- Export fichier destination (« output »)

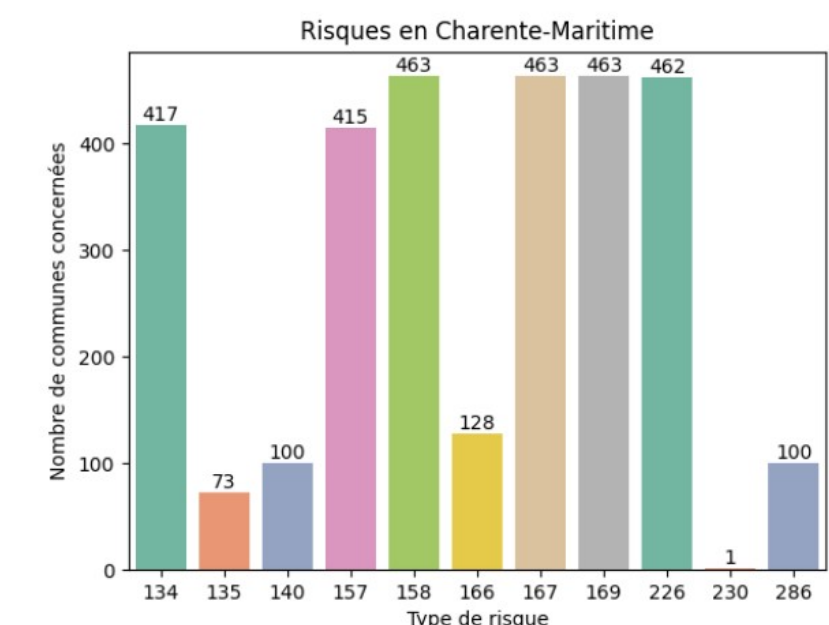
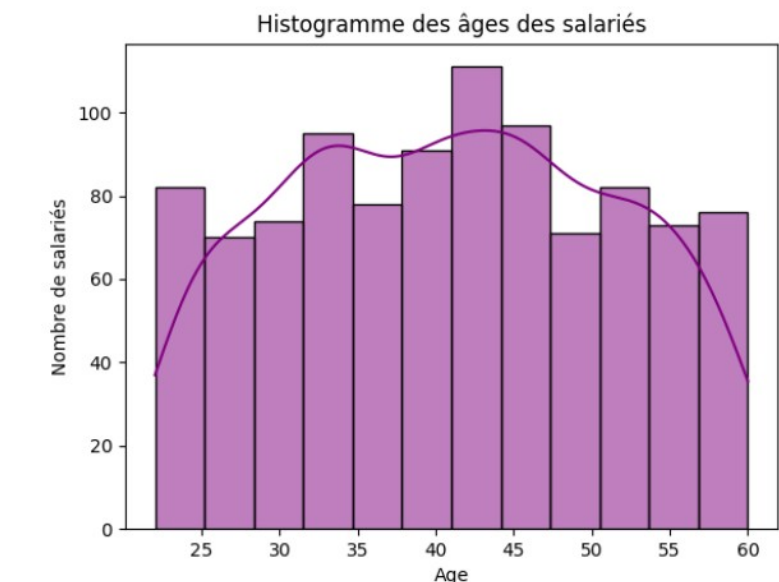


*script*



Seaborn

Mise en forme de graphiques  
(histogrammes, graphs circulaires, graphs  
bivariés, graphiques en barres, etc)



# API / webservice

- Une **API** définit les méthodes d'interaction d'un logiciel avec un autre. Elle implique généralement un appel de fonctions (ex : API QGIS « PyQGIS »)
- Lorsqu'une API est appelée par un protocole web (HTTP/HTTPS), on parlera de **webservice** (ou service web)
- Les webservices permettent l'automatisation d'échanges de données et d'informations entre systèmes informatiques, par l'exploitation d'URLs, dans des langages normés (XML, JSON, etc)
- Souvent les webservices sont intégrés dans la notion d'API

```
1  # Let's access the 'countries' layer
2  layer = QgsProject.instance().mapLayersByName('countries')[0]
3
4  # Let's filter for countries that begin with Z, then get their features
5  query = '"name" LIKE \'Z%\''
6  features = layer.getFeatures(QgsFeatureRequest().setFilterExpression(query))
7
8  # now loop through the features, perform geometry computation and print the results
9  for f in features:
10     geom = f.geometry()
11     name = f.attribute('NAME')
12     print(name)
13     print('Area: ', geom.area())
14     print('Perimeter: ', geom.length())
```



# Principe du webservice : exemple de Météo France

L'application interroge les serveurs de Météo France via un Webservice



**Vous souhaitez connaître le temps qu'il fera à Avignon, le 13 février : vous tapez « Avignon » dans la barre de recherche de l'application et sélectionnez le 13 février**

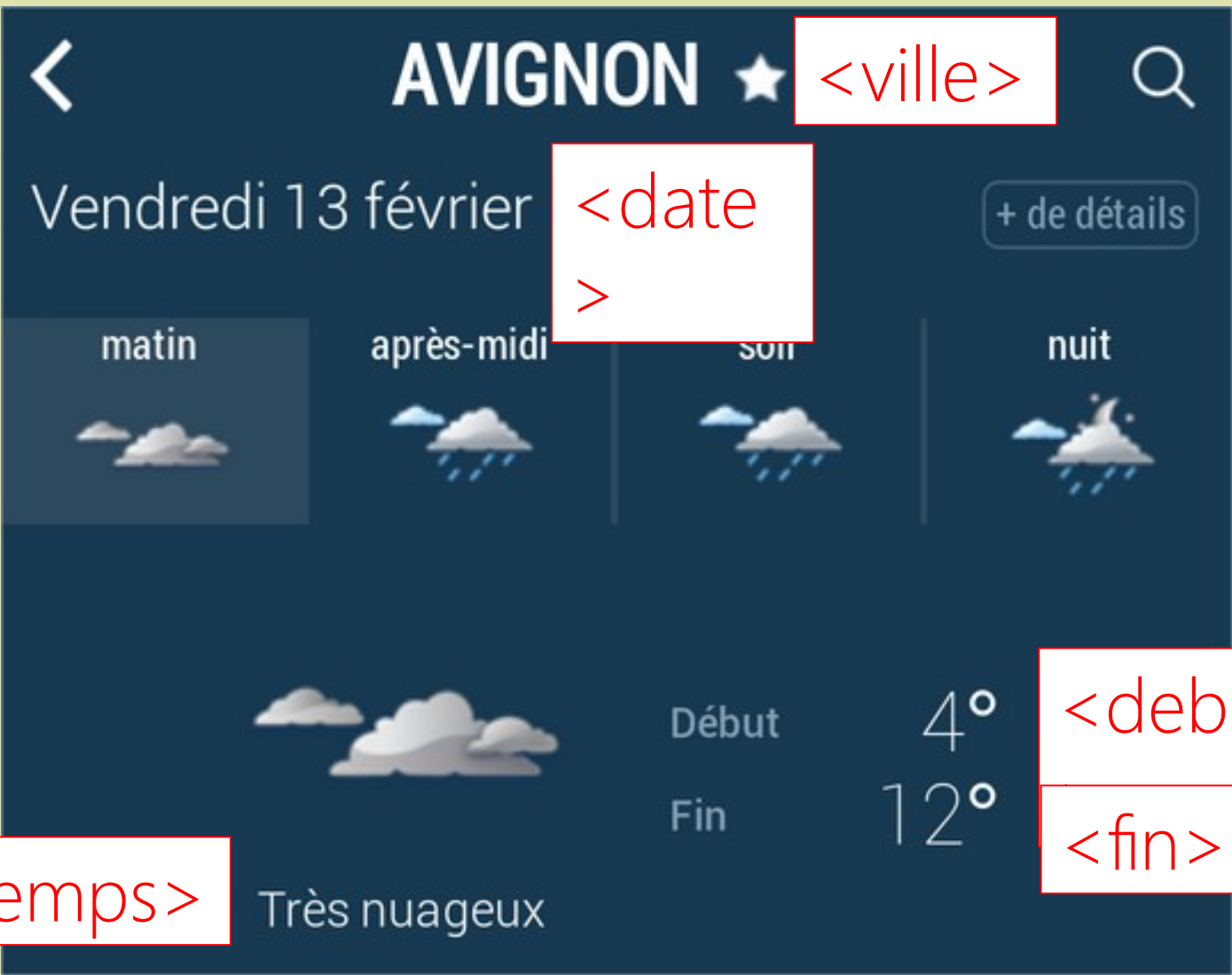


(1)

(2)

**L'application consomme le Webservice et renvoie les résultats sous la forme d'une interface moderne**

(3)



```
<ville> AVIGNON</ville>
<date>Vendredi 13
février</date>
<debut>4°</debut>
<fin>12°</fin>
<temps>Très nuageux</temps>

<ville> MARSEILLE</ville>
<date>Vendredi 13
février</date>
<debut>3°</debut>
<fin>17°</fin>
<temps>Nuageux</temps>
```

**Résultats de la requête en XML**

# APIs *gouv.fr* (= *webservices*)



api.gouv.fr

[? Une question ?](#)

[Rechercher une API du service public](#)

[Comprendre les API](#)

[À propos](#)

## Vous recherchez une API du service public ? Vous êtes au bon endroit !

Thématique

Toutes les thématiques



Modalité d'accès

Tous les accès



Recherchez un service, un ministère



168 résultats

### API Découpage Administratif - (API Geo)



Interrogez les référentiels géographiques plus facilement

**Produit par :** Direction Interministérielle du Numérique



Accès libre

● 100.00% actif / dernier mois

### API Particulier



Entités administratives, simplifiez les démarches des particuliers en récupérant pour eux leurs informations administratives (quotient familial CAF, composition familiale, statut demandeur d'emploi, étudiant et étudiant boursier).

**Produit par :** Direction Interministérielle du Numérique



Sous habilitation

● 99.99% actif / dernier mois

### API Bénéficiaires effectifs - Inpi | Bouquet API Entreprise



Entités administratives, parmi les données entreprises/associations distribuées par le bouquet API Entreprise, récupérez la liste Inpi des bénéficiaires effectifs d'une unité légale inscrite au RNE.

**Produit par :** Direction Interministérielle du Numérique



Sous habilitation

● 100.00% actif / dernier mois

### API Entreprise



Entités administratives, simplifiez les

### API Adresse (Base Adresse Nationale - BAN)



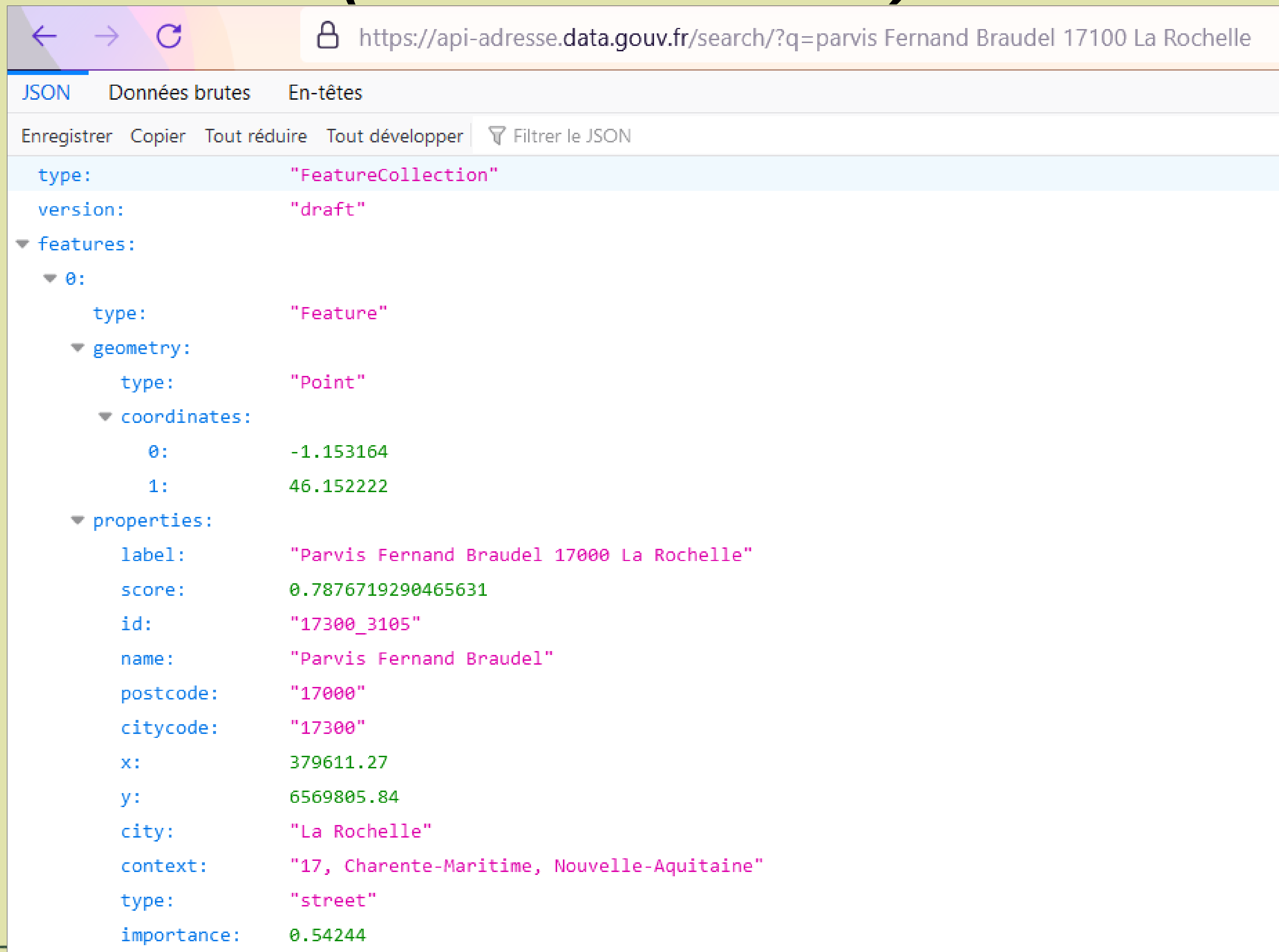
### API Recherche des personnes physiques (R2P)



[Une question ?](#)



## ***L' API Adresse (Base Adresse Nationale)***

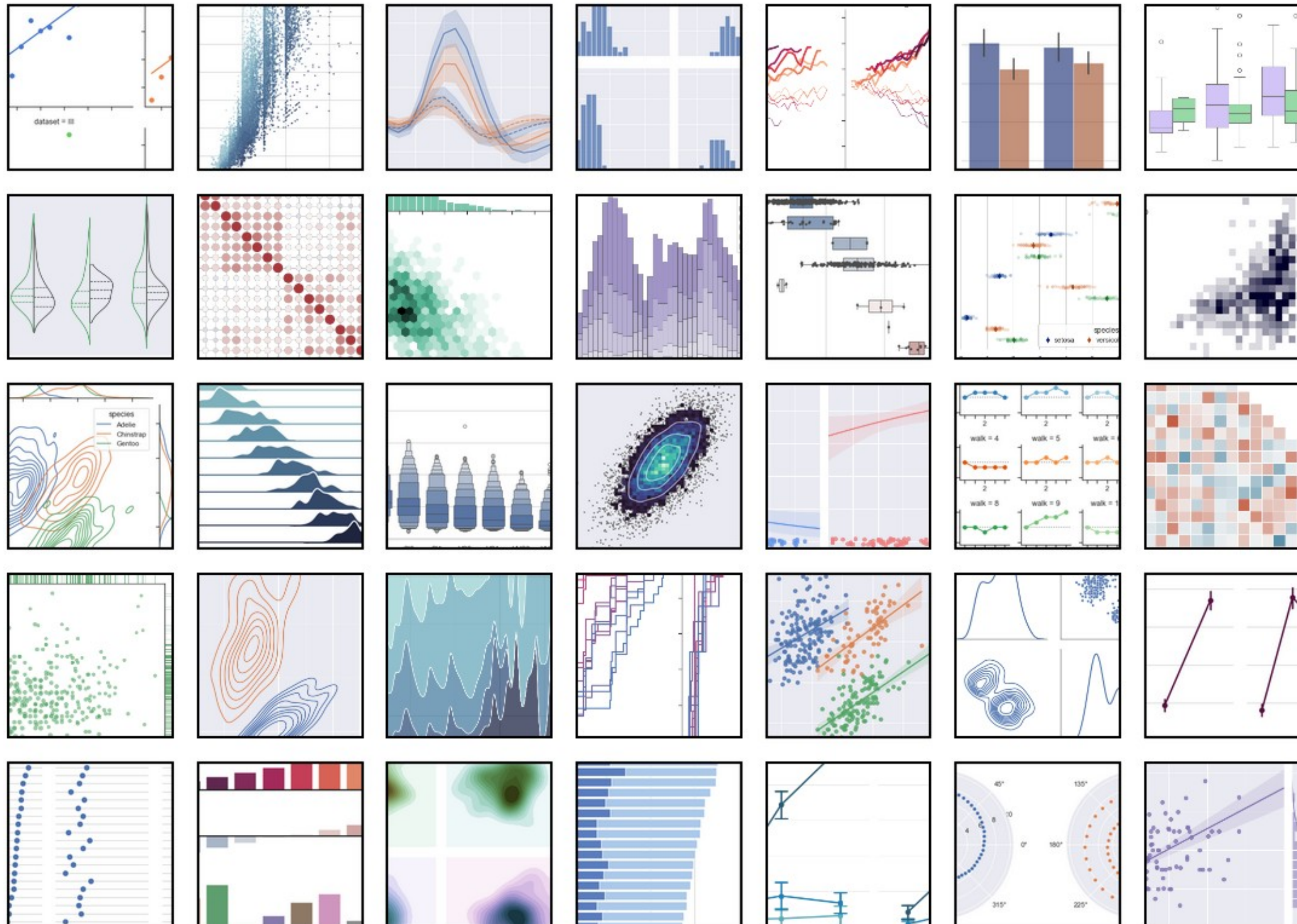


# Rappels

- La syntaxe Pandas :
  - `df['age'];` → Sélection d'une colonne
  - `df[['age', 'job title'];` → Sélection de n colonnes
  - `df.loc[5 : 10];` → Sélection d'un intervalle de lignes
  - `df.loc[5, ['age', 'job title'];` → Sélection de deux colonnes sur une ligne donnée
  - `df.where(df['age'] > 30);` → Requête numérique
  - `df_marketing = df.where(df['department'] == "Marketing")` → Requête textuelle
  - `df_marketing = df.where(df['department'] == "Marketing").dropna()[['name', 'job title'];` → Requête textuelle (bis)
  - `df[['age', 'department']].groupby(by='department').mean();` → Requête par calcul de regroupement

# Rappels

- De nombreuses possibilités avec Seaborn (<https://seaborn.pydata.org/examples>)





# Rappels

- Garder en tête l'importance des APIs :
  - De plus en plus de données mobilisées sous forme d'APIs dans le monde de la géomatique
  - Mode de consommation de donnée amené à se développer encore plus à l'avenir
  - Intégrable sans problème dans une chaîne de traitements codée en langage Python
  - 188 APIs open-source disponibles sur [api.gouv.fr](https://api.gouv.fr) mais pas que... API IGN, SIRENE (INSEE), Météo France, etc
  - Possibilité de créer et héberger ses propres APIs

Exemple : [https://data.pampas.univ-lr.fr/node/attributs/vue\\_cn\\_atherine\\_fa](https://data.pampas.univ-lr.fr/node/attributs/vue_cn_atherine_fa)

---

TD n°3

---

Développement de scripts  
Python avec QGIS

# Objectifs

- Utiliser les algorithmes QGIS dans des scripts Python
- Intégrer des actions d'interaction avec QGIS (affichage de données, de messages d'information, etc) en utilisant l'API PyQGIS
- Introduction à la mise en place de plugins QGIS (cf. TD n°5 et n°6)



# Principe (1/2)

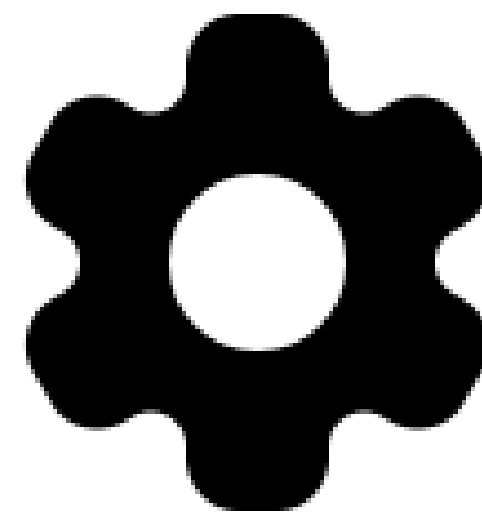
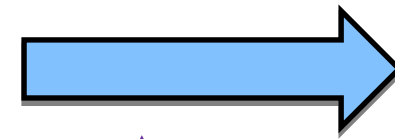
## Algos

Boîte à outils de traitement

Rechercher...

- Utilisé récemment
- Analyse de réseau
- Analyse de terrain raster
- Analyse raster
- Analyse vectorielle
- Cartographie
- Conversion de nuage de points
- Création d'un raster
- Création de vecteurs
- Database
- Extraction de nuage de points
- Géométrie vectorielle
- Gestion des données nuage de points
- GPS
- Interpolation
- Mesh
- Outils fichiers
- Outils généraux pour les couches
- Outils généraux pour les vecteurs
- Outils raster
- Points
- Recouvrement de vecteur
- Sélection dans un vecteur
- Table vecteur
- Tuiles 3D
- Tuiles vectorielles

Traitement



Interactivité



Avancé Exécuter comme processus de lot...

Algorithm Settings...

Copier en tant que commande Python

Copier en tant que commande qgis\_process



## API PyQGIS

addMapLayer(self, mapLayer: QgsMapLayer | None, addToLegend: bool = True) → QgsMapLayer | None

Add a layer to the map of loaded layers.

The layersAdded() and layerWasAdded() signals will always be emitted. The legendLayer adding multiple layers at once, you should use addMapLayers() instead.

Parameters:

- mapLayer (Optional[QgsMapLayer]) – A layer to add to the registry
- addToLegend (bool = True) – If True (by default), the layer will be added to the legend. If False, you can set this parameter to False to hide it.

Return type: Optional[QgsMapLayer]

Returns: None if unable to add layer, otherwise pointer to newly added layer

See also

- addMapLayers()

Note

As a side-effect QgsProject is made dirty.

Note

Use addMapLayers if adding more than one layer at a time

Note

takeOwnership is not available in the Python bindings - the registry will always take

See also

QGIS Python API

3.34

Search docs

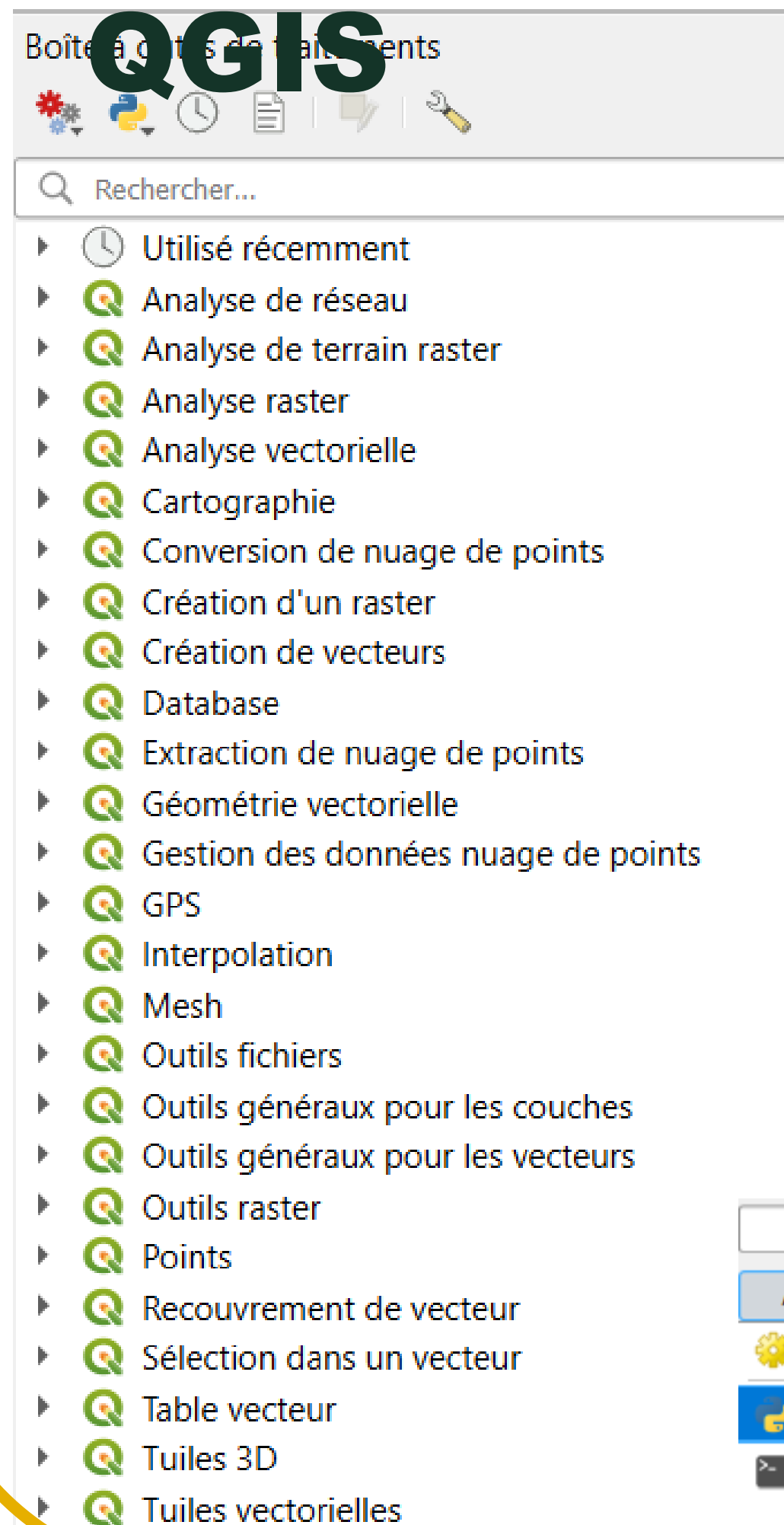
CONTENTS:

- core
- gui
- analysis
- server
- processing
- \_3d

QGIS Python API v: 3.34

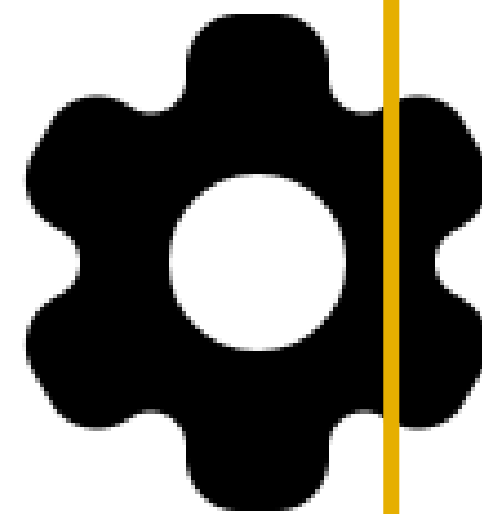
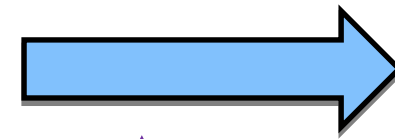
# Principe (2/2)

## Algos



## Prépa & Exécution PyCharm

Traitement



Interactivité



## Prépa PyCharm Exécution QGIS

## API PyQGIS

(depuis v. 3.0.0)

```
addMapLayer(self, mapLayer: QgsMapLayer | None, addToLegend: bool = True) → QgsMapLayer | None
```

Add a layer to the map of loaded layers.

The `layersAdded()` and `layerWasAdded()` signals will always be emitted. The `legendLayer` adding multiple layers at once, you should use `addMapLayers()` instead.

Parameters:

- `mapLayer` (Optional[`QgsMapLayer`]) – A layer to add to the registry
- `addToLegend` (bool = `True`) – If `True` (by default), the layer will be added to the legend. If `False`, you can set this parameter to `False` to hide it.

Return type: Optional[`QgsMapLayer`]

Returns: `None` if unable to add layer, otherwise pointer to newly added layer

### See also

`addMapLayers()`

### Note

As a side-effect `QgsProject` is made dirty.

### Note

Use `addMapLayers` if adding more than one layer at a time

### Note

`takeOwnership` is not available in the Python bindings - the registry will always take

### See also

# Application



## 1 – Configuration de PyCharm

## 2 – Utiliser les algorithmes QGIS dans un script Python

## 3 – Compléter par l'utilisation de l'API PyQGIS

## 4 - Compléments

- Import de l'ensemble des librairies dédiées au traitement QGIS dans PyCharm
- Objectif : se familiariser avec l'approche de l'utilisation des algorithmes présents dans la boîte à outils de QGIS
- Affichage ou suppression de couches, requêtes, connexions PostGIS, affichage de messages d'information ou d'erreurs, sauvegarder un nouveau projet, lister les couches affichées, les couches valides, affecter des symbologies
- Boucles, interaction avec bases de données, intégrer un script dans la boîte à outils de QGIS



# Bilan

## TD3

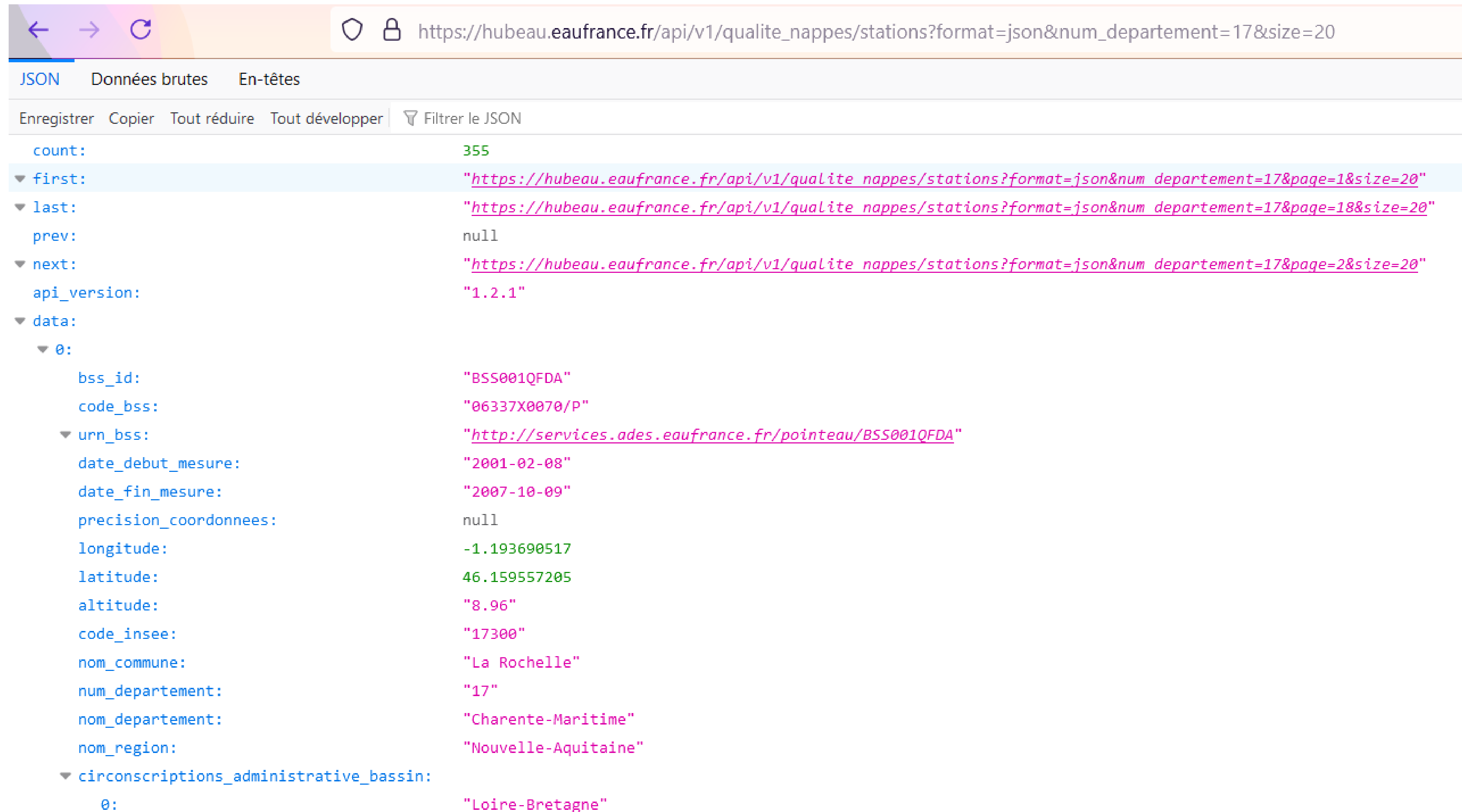
Retour sur les classes et fonctions utilisées dans l'API PyQGIS :

- **QgsVectorLayer** : featureCount(), getFeatures()
- **QgsProject** : addMapLayer(), featureCount(), write()
- **QgsMessageBar** : pushMessage()
- **QgsFeatureRequest** : setFilterExpression()
- **QgsDataSourceUri** : setConnection(), setDataSource()

Pour + d'infos :

<https://qgis.org/pyqgis/master> | <https://webgeodatavore.github.io/pyqgis-samples>

# Retour sur les APIs



---

TD n°4

---

Développement de plugins  
QGIS

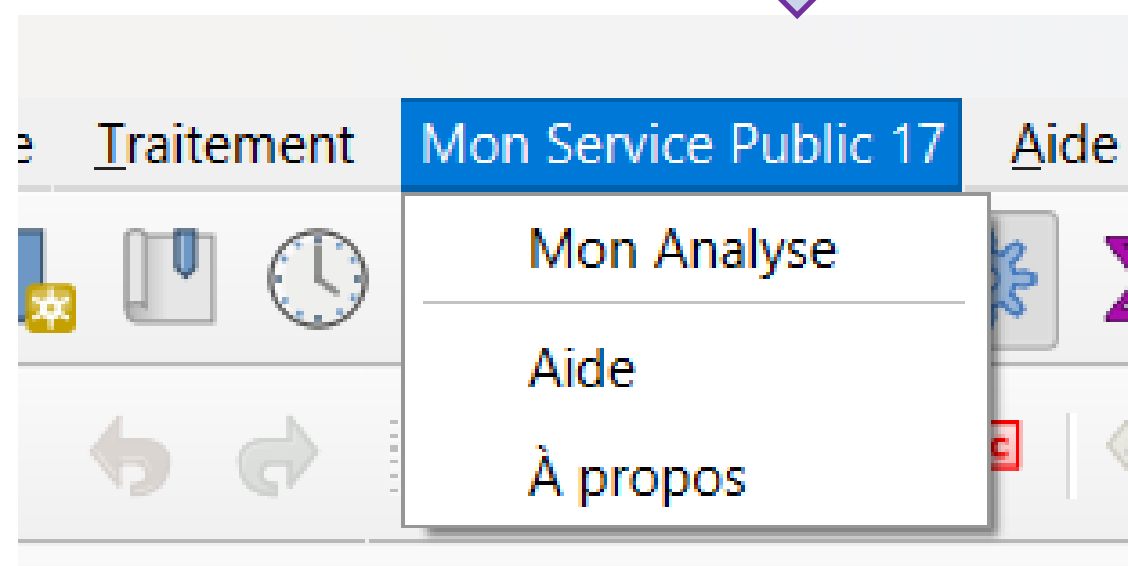


# Objectifs

- Vous initier à la **création et l'alimentation d'un plugin**, au sein même de l'interface de QGIS
- Remobiliser les compétences techniques vues **depuis le TD n°1** :
  - les algorithmes QGIS via le module « processing » :  
*processing.run(native:nom\_du\_script)*
  - l'API PyQGIS pour faire interagir le plugin avec QGIS (ajout de données, affichage de messages, etc)
  - les packages utilisés jusqu'à présent (glob2, pandas/geopandas, seaborn, sqlalchemy)

# Objectifs

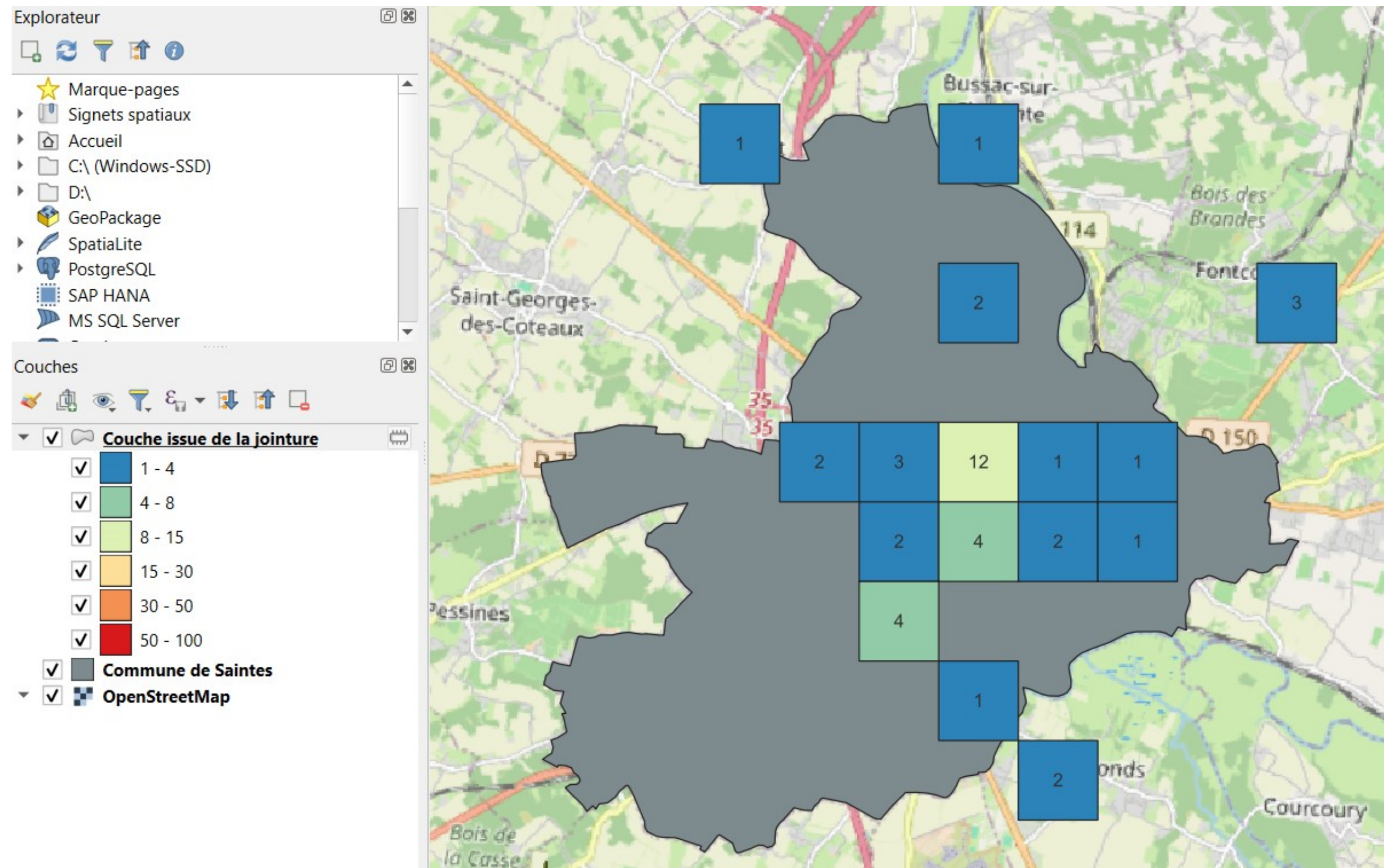
- Objectif du plugin : mise en place d'une carte de densités de services public par commune de Charente-Maritime





# Objectifs

Démarrer

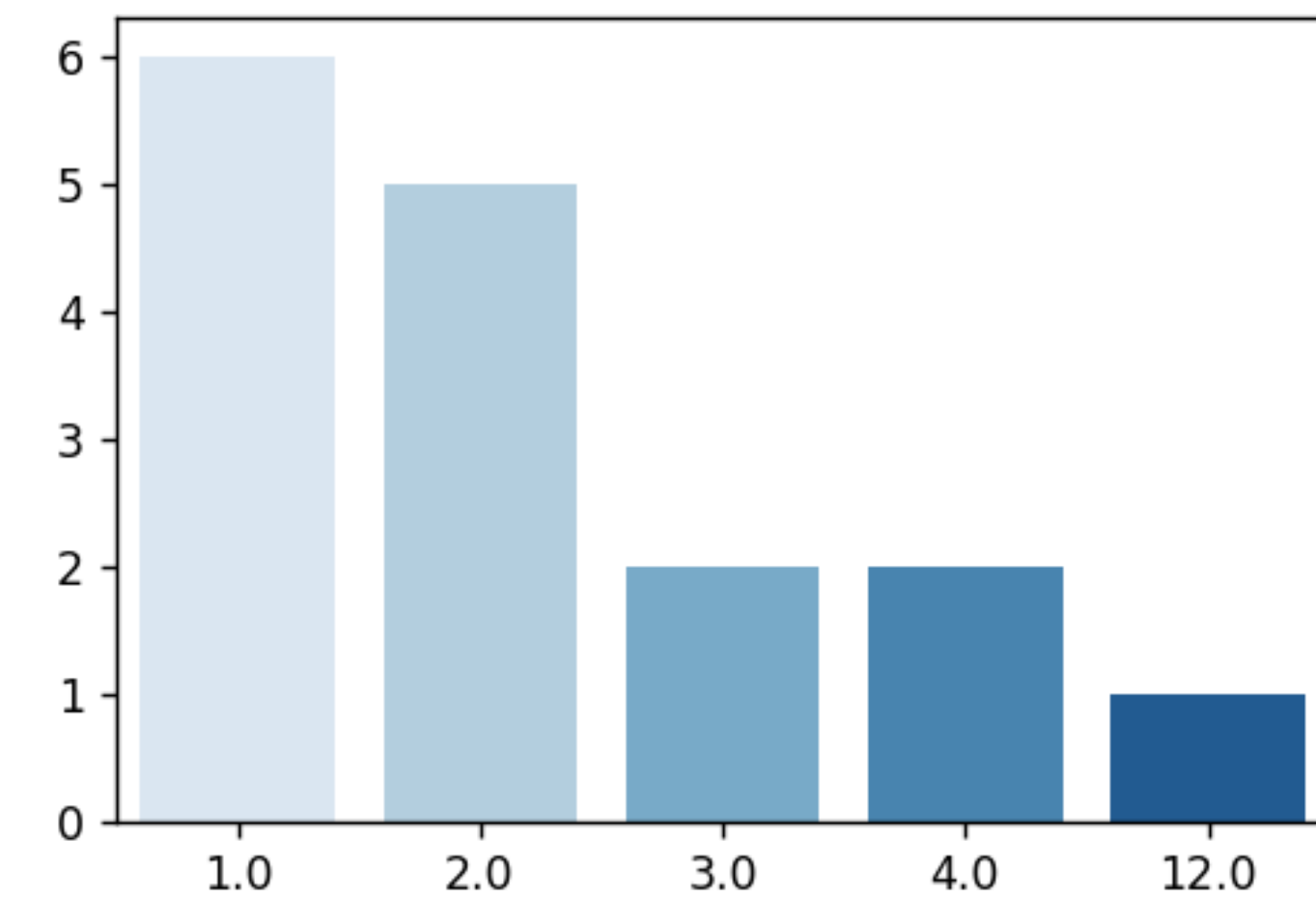


Visualiser

Mon Service Public

Niveau d'accès aux services public par maille

Commune de Saintes



# Application



## 1 – Mise en place du plugin

- Création du plugin avec « Plugin Builder »
- Création du menu parent du plugin

## 2 – Création de la fenêtre de traitements (UI)

- Utilisation de l'outil **Qt Designer**
- Mise en place sans code de l'interface de la fenêtre de traitements (« widgets »)

## 3 – Mise en place du code Python associé à la fenêtre principale

- Association de chaque widget à du code Python
- Contrôle de l'import des données *glob2 + os +*
- Mise en place des traitements *algos QGIS (processing) + geopandas + sqlalchemy*

## 4 – Création et développement de la fenêtre graphique

- Création de l'interface UI de la fenêtre
- Développement du graphique via Python *geopandas + seaborn*



---

# Conclusion

---

# Bilan

## **Vous avez désormais été initiés à :**

- Utiliser les fonctionnalités de base du langage Python (fonctions, listes, boucles, conditions)
- Utiliser quelques packages très utiles dans la programmation Python (glob, seaborn, pandas/geopandas, sqlalchemy, requests)
- Récupérer de gros volumes de données via APIs pour les exploiter en fonction de vos besoins (export, mise en forme graphique, alimentation d'une base de données, etc)
- Mobiliser les algorithmes SIG de QGIS dans des scripts de traitements automatisés
- Faire interagir Python et QGIS via l'API PyQGIS (ajout de données, affichage de messages, etc)
- Développer des plugins QGIS en mobilisant les points précédents et Qt Designer/PyQt

# Bilan

**Dans le cadre de vos travaux**, le langage Python est intéressant à utiliser...

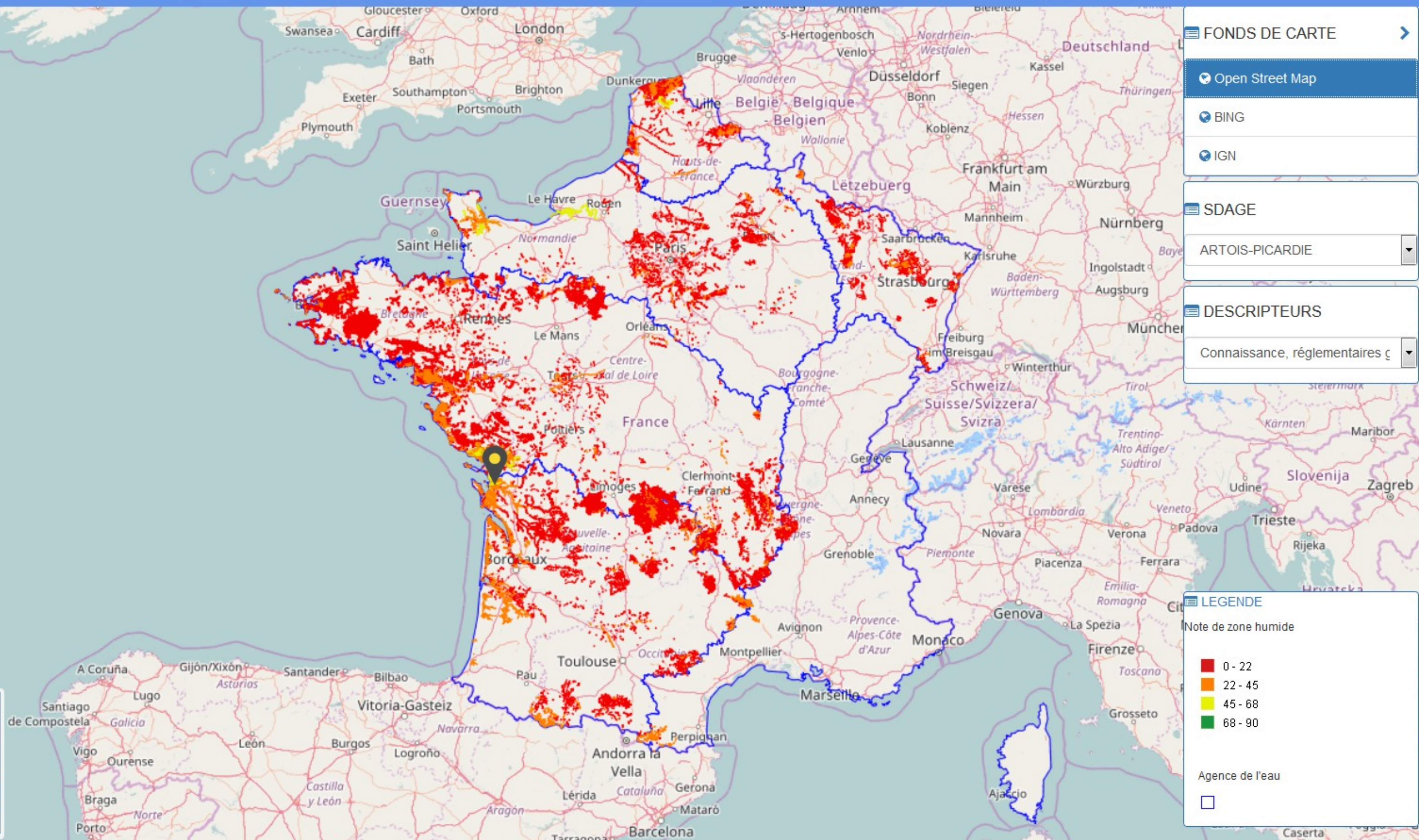
- Pour manipuler des données, SIG ou non (modifier la structure, extraire, joindre, exporter, supprimer, etc)
- Pour mobiliser des webservices, formats de données de plus en plus utilisés
- Pour mettre en place des graphiques de types divers qui résument vos données
- Pour manipuler les algorithmes de QGIS (équivalent du modeleur graphique mais avec plus de souplesse)
- Le tout en automatisant pour vous faire gagner du temps

# Conseils et perspectives

---

- Pratiquer, tester et travailler
- Commenter et rendre son code lisible autant que possible
- Se référer aux documentations en lignes
- Utiliser les outils IA avec parcimonie
- Avoir en tête l'intérêt de la programmation dans vos travaux SIG
- Utiliser des frameworks pour la mise en place d'interfaces web développées en Python







**Merci de votre  
attention !**