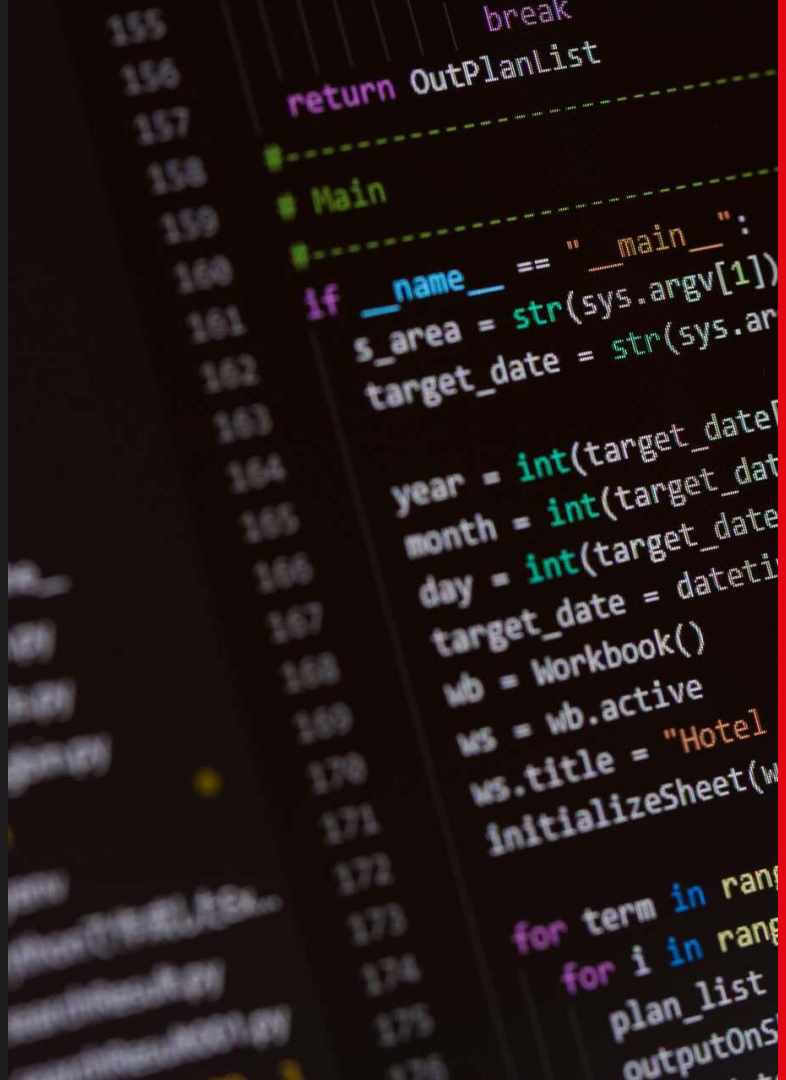

# SIG – Systèmes d'Information Géographiques

Introduction Python - 2

Gabriel Kathari

Stéphane Joost



# Les bases de Python pour le cours de SIG

Les dictionnaires

Les classes

Les librairies

# Les dictionnaires - Déclaration

- Permet de stocker des données complexes
- Déclaration par ajout de paires clé-valeur :

```
# Création d'un dictionnaire vide
mon_dictionnaire = {}

# Ajout de paires clé-valeur
mon_dictionnaire["cle_1"] = "valeur1"
mon_dictionnaire["cle_2"] = "valeur2"
mon_dictionnaire["cle_3"] = "valeur3"
```

- Déclaration avec duo clé-valeur directement dans le dictionnaire :

```
mon_dictionnaire = {
    "cle_1": "valeur1",
    "cle_2": "valeur2",
    "cle_3": "valeur3"
    }
```

# Les dictionnaires – Accès

- Accéder à la valeur associée à une clé :

```
print(mon_dictionnaire["cle_2"])
```

```
valeur2
```

- Accéder à la valeur associée à une clé avec get()

*Valeur par défaut si "cle_2" n'est pas accessible dans Le dictionnaire*

```
valeur_1 = mon_dictionnaire.get("cle_2", "valeur_par_defaut")
print(valeur_1)
valeur_2 = mon_dictionnaire.get("cle_inexistante", "valeur_par_defaut")
print(valeur_2)
```

```
valeur2
```

```
valeur_par_defaut
```

# Les dictionnaires – Accès dans une boucle



- Accéder aux éléments d'un dictionnaire dans une boucle:

```
mon_dictionnaire = {
    "cle_1": "valeur1",
    "cle_2": "valeur2",
    "cle_3": "valeur3"
    }
```

```
for cle, valeur in mon_dictionnaire.items():
    print(f"Clé: {cle} - Valeur: {valeur}")
```

```
Clé: cle_1 - Valeur: valeur1
Clé: cle_2 - Valeur: valeur2
Clé: cle_3 - Valeur: valeur3
```

*On fait une boucle sur chaque cle et sa valeur associée grâce à mon_dictionnaire.items()*
*Cette méthode va récupérer chaque cle et chaque valeur associée pour chacun des éléments du dictionnaire.*

# Les classes – Déclaration

- Une classe en Python est un modèle pour la création d'objets. Elle définit à la fois les données et les méthodes qui agissent sur ces données.

class : définit la creation d'une classe

Nom de la classe (une majuscule au début de chaque mot composant le nom)

__init__() est une méthode spéciale en Python appelée constructeur. Elle est appelée automatiquement lorsqu'un nouvel objet est créé. Elle initialise les attributs de l'objet.

```
class RappeurFrancais:
    def __init__(self, nom, age, album):
        self.nom = nom
        self.age = age
        self.album = album

    def description(self):
        return f"{self.nom}, {self.age} ans, a sorti l'album '{self.album}'"
```

self est une référence à l'objet lui-même et est utilisé pour accéder aux attributs et méthodes de l'objet.

Méthode propre à la classe RappeurFrancais

# Les classes – Utilité

- **Encapsulation** :

  Les classes permettent d'encapsuler les données (les attributs) et les fonctionnalités (les méthodes) liées à un même concept. Cela favorise la modularité et la réutilisabilité du code.

- **Abstraction** :

  Les classes permettent de représenter des concepts abstraits et des entités du monde réel. Elles cachent les détails d'implémentation et ne montrent que les fonctionnalités essentielles à l'utilisateur.

- **Héritage** :

  Une classe peut hériter des propriété d’une autre classe.

# Les classes - Exemple

- Classe : RappeurFrancais

```python
class RappeurFrancais:
    def __init__(self, nom, age, album):
        self.nom = nom
        self.age = age
        self.album = album

    def description(self):
        return f"{self.nom}, {self.age} ans, a sorti l'album '{self.album}'"
```

```python
booba = RappeurFrancais("Booba", 45, "Trône")
```

self.nom self.age self.album

# Les classes - Exemple

- **Classe : RappeurFrancais**

Objets de la classe RappeurFranais

```
booba = RappeurFrancais("Booba", 45, "Trône")
kaaris = RappeurFrancais("Kaaris", 42, "Or Noir")
```

```
print(booba.description())
print(kaaris.description())
```

```
Booba, 45 ans, a sorti l'album 'Trône'
Kaaris, 42 ans, a sorti l'album 'Or Noir'
```

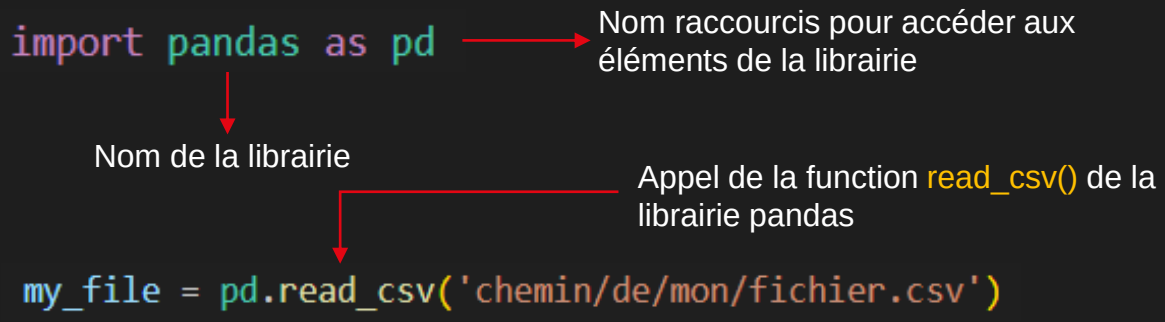
Appel de la fonction description() de l'objet de classe RappeurFrancais

```
class RappeurFrancais:
    def __init__(self, nom, age, album):
        self.nom = nom
        self.age = age
        self.album = album

    def description(self):
        return f"{self.nom}, {self.age} ans, a sorti l'album '{self.album}'"
```

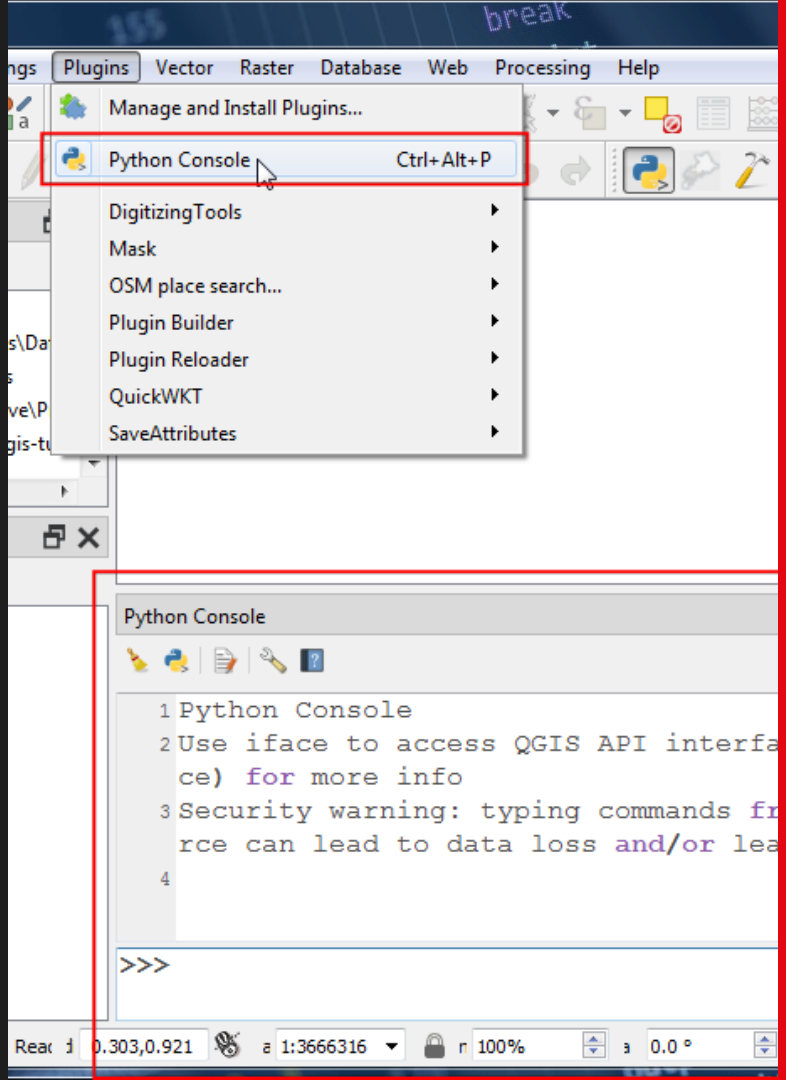
# Les librairies

- **Une librairie est un ensemble de classes et de fonctions accessibles en ligne.**
- **Pour pouvoir utiliser les classes et fonctions d'une librairie, il faut importer cette dernière avec import**




**Pour importer une librairie, il est nécéssaire de l'avoir téléchargée sur dans son environnement virtuel**

→ https://docs.python.org/fr/3/tutorial/venv.html



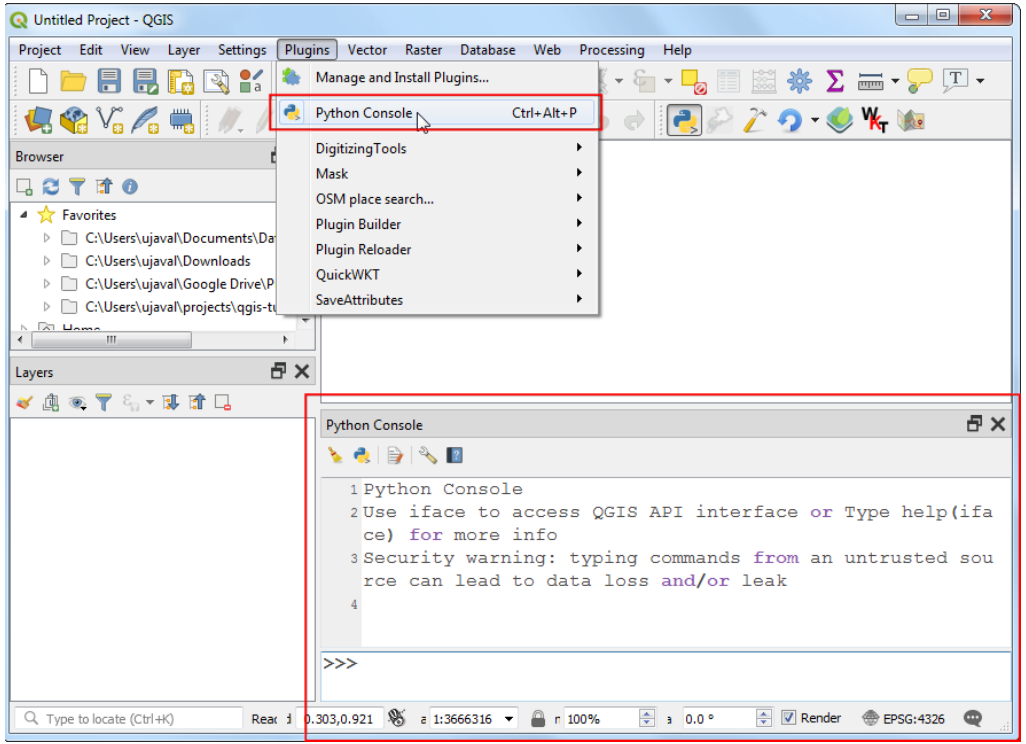

# Les possibilités offertes par Python pour les systèmes d'information Géographique

Console Python dans QGIS

La librairie PyQT

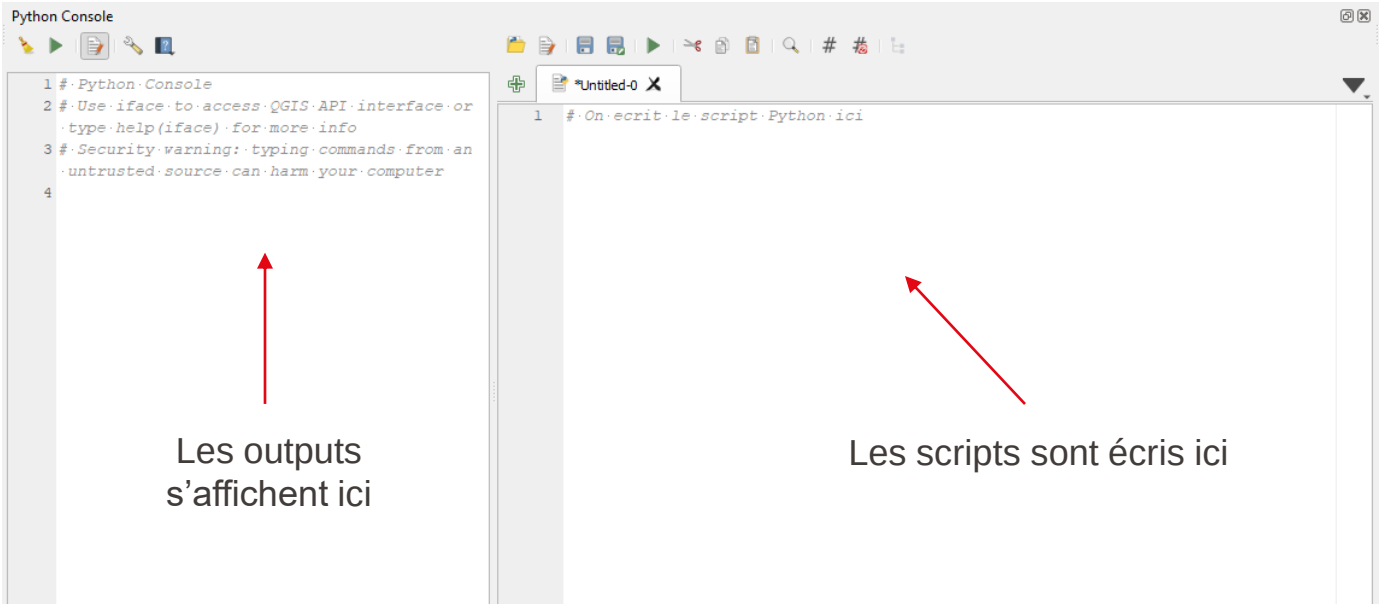
# Console python dans QGIS

- La console Python dans QGIS se trouve dans Plugins > Python console (Ctrl + Alt + P)



# Console python dans QGIS

- La console est composée de 2 parties, une partie pour écrire les scripts (code python) et une partie pour afficher les outputs :



# Console python dans QGIS

- Elle permet de créer des scripts en python pour automatiser certaines taches sur les couches et utiliser les fonctions de la toolbox.
- Voici un exemple de code python permettant d'afficher le nom de tous les layers (couches) présents dans mon projet :

Appel de la librairie qgis.core, spécifique a QGIS

```
from qgis.core import QgsProject

# Récupération de l'instance du projet courant
project_instance = QgsProject.instance()

# Boucle sur tous les layers présents dans le projet
# et affichage de leur nom
for layer in project_instance.mapLayers().values():
    print(layer.name())
```

```
Réseau Hydrographique
Réseau Routier
Surfaces primaires
Valais_central
orthophoto
```

# Console python dans QGIS

- Voici un exemple de code qui indique le CRS de toutes les couches présentes dans mon projet :

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('C:/Users/Gabriel/AppData/Local/Temp/tmp_cu8aek3.py'.encode('utf-8')).read())
Nom de la couche : Réseau Hydrographique
CRS de la couche : EPSG:2056
Nom de la couche : Réseau Routier
CRS de la couche : EPSG:2056
Nom de la couche : Surfaces primaires
CRS de la couche : EPSG:2056
Nom de la couche : Valais_central
CRS de la couche : EPSG:2056
Nom de la couche : orthophoto
CRS de la couche : EPSG:2056
```

*Untitled-0

```
# Obtenir toutes les couches du projet
layers = QgsProject.instance().mapLayers().values()

# Boucle à travers toutes les couches
for layer in layers:
    # Obtenir le nom de la couche
    layer_name = layer.name()

    # Obtenir le système de référence de coordonnées (CRS) de la couche
    layer_crs = layer.crs().authid()

    # Afficher le nom de la couche et le CRS dans la console Python de QGIS
    print("Nom de la couche : ", layer_name)
    print("CRS de la couche : ", layer_crs)
```

# Console python dans QGIS

Les fonctions spécifiques a la librairie QGIS peuvent être trouvées sur le lien suivant.

https://docs.qgis.org/3.22/fr/docs/pyqgis_developer_cookbook/intro.html

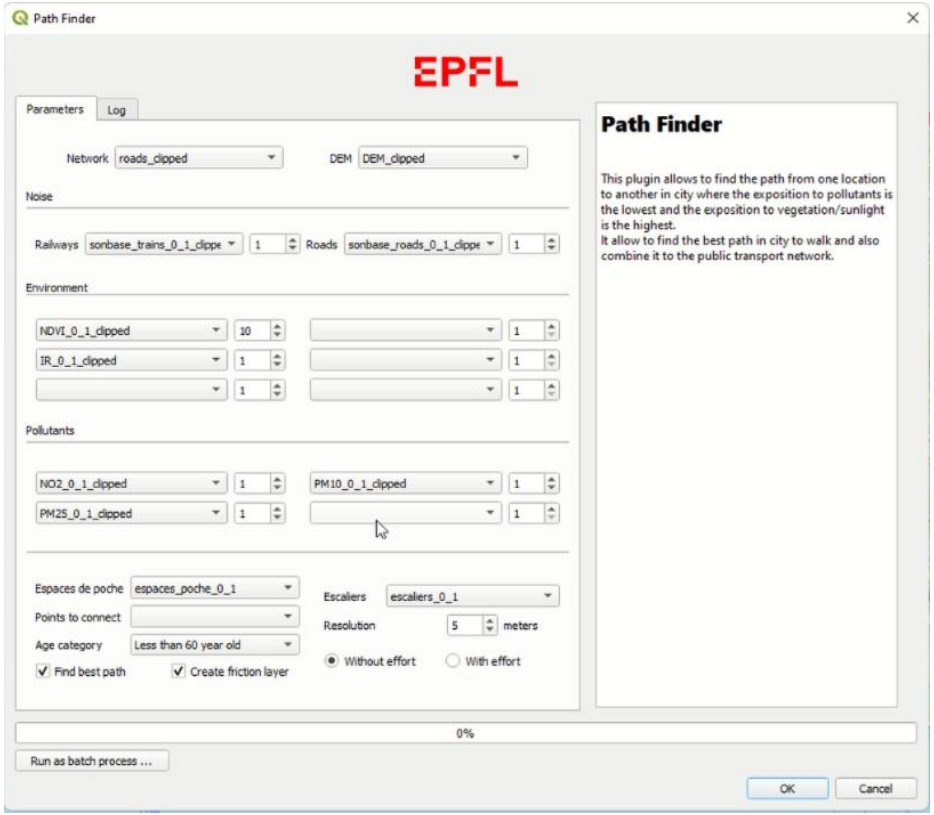
Lorsque l'on travaille avec une librairie spécifique, il y à plusieurs étapes à respecter pour gagner du temps de vie précieux :

- Lisez la documentation attentivement avant de vous lancer.
- Stack Overflow et autres forums sont vos amis.
- Lorsque vous rencontrez une erreur, il est fort probable que quelqu'un d'autre y ai déjà fait face, internet saura probablement vous répondre !
- Lorsque vous faites une recherche sur internet, soyez le plus précis et explicite possible dans votre recherche.
- Pratiquer le plus régulièrement possible afin de développer des automatismes.

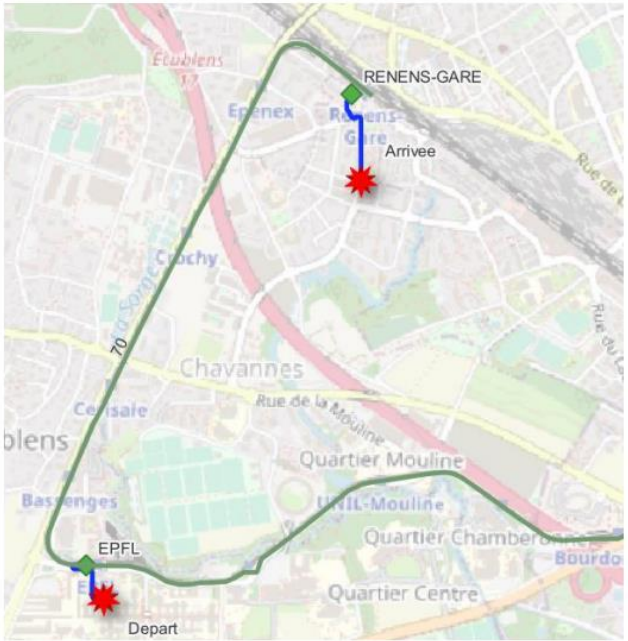
# Projet hybride : Plugin Python (TL - cheminements mixtes )

- **Objectif** : Créer des cheminements mixtes entre transport public et marche en ville de Lausanne maximisant l'exposition à la végétation, l'ensoleillement et diminuant l'exposition à la pollution atmosphérique et au bruit par tranche d'âge de la population.
- **Methode** : Création d'une couche de friction prenant en compte les variables environnementales, la pente, les espaces de séjour, les escaliers etc.
- **Résultat** : Meilleur cheminement en fonction du temps de parcours, du cout de friction et de la tranche d'age.

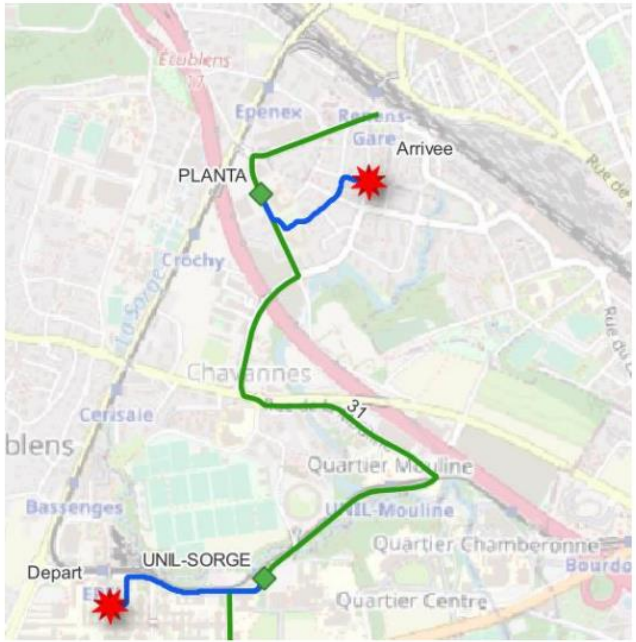
# Projet hybride : Plugin Python (TL - cheminements mixtes )



# Projet hybride : Plugin Python (TL - cheminements mixtes )




**Paramètre** : temps de trajet
**Temps de parcours** : 7 minutes
**Ratio** : 1.9




**Paramètre** : ratio coût / distance
**Temps de parcours** : 15 minutes
**Ratio** : 1.4

# Merci pour votre attention et a la semaine prochaine !