Projet : Module 5

Flux de Contrôle & Automatisation

M2 GER
Université de Nantes -- IGARUN

Objectifs du Module 5

Nous avons nos données (Liste de Dictionnaires) et notre moteur (Fonction). Il faut maintenant appuyer sur l'accélérateur.

PROGRAMME

- **Théorie :** Le concept d'itération (boucle `for`).
- **Robustesse :** La gestion des erreurs (`try/except`) pour éviter les crashs en série.
- **Pratique :** Lancement du traitement sur Nantes, Rennes et Brest en une seule fois.

1. Théorie : L'Itération

La Boucle For

La boucle `for` permet de répéter une action pour chaque élément d'une collection.
C'est le concept de **Batch Processing** (Traitement par lots).

```
for element in liste:
    traiter(element)
```

Ici, `element` est une variable temporaire qui change de valeur à chaque tour.

Itérer sur notre Configuration

Rappel de notre structure de données (Module 2) :

```
config = [
    {"nom": "Nantes", ...},  # Index 0
    {"nom": "Rennes", ...},  # Index 1
    {"nom": "Brest", ...}    # Index 2
]
```

Quand on fait `for ville in config:`, la variable `ville` devient successivement le dictionnaire de Nantes, puis Rennes, puis Brest.

2. Robustesse du Code

Le Problème du "Crash"

> **Scénario catastrophe :** Vous lancez un script sur 50 villes avant de partir en pause déjeuner. La 3ème ville (Brest) a un fichier corrompu. Le script plante. Résultat : Les 47 villes suivantes ne sont pas traitées.

Nous devons dire à Python : *"Si ça rate, note l'erreur et passe à la suivante"*.

Try / Except

Le bloc `try/except` permet de capturer les erreurs.

```
for ville in config:
    try:
        # Tente d'exécuter ce code
        analyser_ville(ville["nom"], ... )
        print("Succès !")

    except Exception as e:
        # Si une erreur survient, exécute ceci au lieu de planter
        print(f"Erreur sur {ville['nom']} : {e}")
```

3. Mise en Pratique

Exercice 1 : La Boucle Simple

CONSIGNE

Écrivez une boucle simple qui parcourt votre liste `config` et qui affiche juste : *"Je vais traiter la ville de : [Nom de la ville]"*

C'est la première étape pour vérifier que votre itérateur fonctionne.

Correction Exercice 1

Solution

```
for item in config:
    nom_ville = item["nom"]
    print(f"Je vais traiter la ville de : {nom_ville}")
```

Exercice 2 : Intégration de la Fonction

CONSIGNE

Intégrez l'appel à votre fonction `analyser_ville` (créée au Module 3) dans la boucle.
N'oubliez pas de passer les bons arguments du dictionnaire (url_q, url_p).

Correction Exercice 2


```
# Liste pour stocker les résultats en mémoire
resultats_globaux = []

for item in config:
    try:
        # Appel de la fonction
        gdf_res = analyser_ville(
            item["nom"],
            item["fichier_quartier"],
            item["fichier_parcs"]
        )
        # Stockage dans la liste
        resultats_globaux.append(gdf_res)

    except Exception as e:
        print(f"Erreur critique sur {item['nom']} : {e}")
```

4. Conclusion & Suite

Résultat

Si vous avez bien suivi :

1. Le script a tourné 3 fois.
2. Il a généré 3 fichiers GeoPackage (`Resultat_Nantes.gpkg`, etc.).
3. Il a stocké les 3 GeoDataFrames dans la liste `resultats_globaux`.

Vous avez officiellement automatisé un processus SIG.

Prochain Module

MODULE 6 : APIS & MONDE EXTÉRIEUR

Pour l'instant, nous avons copié-collé les liens manuellement dans notre config.
Peut-on demander à Python de trouver les coordonnées GPS d'une ville juste avec son nom ? (Géocodage).
Peut-on télécharger les données automatiquement ?

Projet : Module 6

APIs & Données Distantes

M2 GER
Université de Nantes -- IGARUN

Objectifs du Module 6

Dans notre boucle d'automatisation, nous utilisons des fichiers locaux. Mais dans la vraie vie, les données sont sur Internet.

PROGRAMME

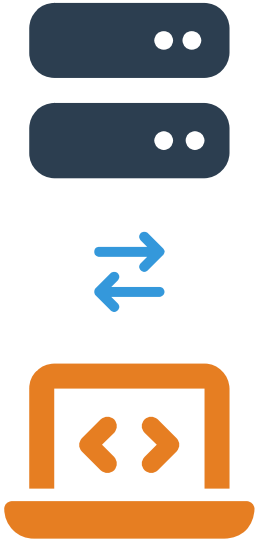
- **Théorie :** Qu'est-ce qu'une API ? (Le serveur comme source de données).
- **Outil :** La librairie geopy pour le géocodage.
- **Pratique :** Récupérer automatiquement les coordonnées d'une ville par son nom.

1. Théorie : Les APIs

Le Concept d'API

API = Application Programming Interface.
C'est un guichet automatique pour les programmes.

- **Requête :** "Donne-moi la météo à Nantes".
- **Réponse :** Un fichier JSON (texte structuré).



Client (Python) $\leftrightarrow$ Serveur

Le Format JSON

Les APIs répondent souvent en **JSON** (JavaScript Object Notation). C'est exactement la structure d'un **Dictionnaire Python** !

```
// Réponse d'une API de géocodage
{
    "ville": "Nantes",
    "lat": 47.218,
    "lon": -1.553
}
```

2. Géocodage avec Geopy

Pourquoi le Géocodage ?

Dans notre script GEE (Module 7), nous devons centrer la carte sur la ville.
Actuellement, on doit chercher "Lat/Lon Nantes" sur Google. C'est manuel.
Solution : Un script qui transforme "Nantes" en `[47.21, -1.55]`.

La Librairie Geopy

geopy est une librairie Python qui simplifie l'accès aux services de géocodage (comme OpenStreetMap Nominatim).

```
# Installation (si besoin)
!pip install geopy
```

```
from geopy.geocoders import Nominatim

# Initialisation (User Agent obligatoire !)
geolocator = Nominatim(user_agent="cours_m2_ger_nantes")
```

Exercice 1 : Test Simple

CONSIGNE

1. Importez `Nominatim`. 2. Cherchez la localisation de "IGARUN, Nantes". 3. Affichez l'adresse complète trouvée et les coordonnées (latitude, longitude).

```
location = geolocator.geocode("IGARUN, Nantes")
print(location.address)
print(location.latitude, location.longitude)
```

3. Intégration dans le Projet

Création d'une Fonction Utilitaire

Pour notre projet, nous avons besoin d'une fonction robuste.

```
def get_city_geometry(city_name):
    try:
        loc = geolocator.geocode(city_name)
        if loc:
            return [loc.longitude, loc.latitude]
    except Exception as e:
        print(f"Erreur de connexion : {e}")
    return None
```

Mise à jour de la Configuration

Nous n'avons plus besoin de stocker les coordonnées manuellement dans notre dictionnaire `config` !
Le script pourra les trouver tout seul.

> **VISION**
>
> Input : `["Nantes", "Rennes", "Brest"]`
> $\downarrow$
> Géocodage Automatique
> $\downarrow$
> Extraction GEE (Module 7)

Exercice 2 : Boucle de Géocodage

CONSIGNE

1. Créez une liste de noms : `villes = ["Nantes", "Rennes", "Brest", "Angers"]`. 2. Faites une boucle qui affiche les coordonnées de chaque ville.

Correction Exercice 2

Solution

```
villes = ["Nantes", "Rennes", "Brest", "Angers"]

for v in villes:
    coords = get_city_geometry(v)
    if coords:
        print(f"{v} : {coords}")
    else:
        print(f"{v} : Non trouvé")
```

4. Conclusion & Suite

Bilan

- Nous savons interroger une API (Nominatim) via une librairie (Geopy).
- Nous savons gérer les réponses (JSON / Objets).
- Nous avons automatisé la localisation spatiale.

> **PROCHAIN MODULE**
>
> Maintenant que nous avons les coordonnées (le "Où"), nous allons demander les images satellites (le "Quoi") à **Google Earth Engine**.