Python & Cartographie – Eléments de corrigé

Visualisation de villes françaises avec coordonnées GPS et Folium

Partie 1 — Questions de compréhension

Question 1 — import pandas as pd

Question : Quel est le rôle de la ligne import pandas as pd ? Pourquoi utilise-t-on l'alias pd ?

> La ligne import pandas as pd charge la bibliothèque pandas, qui permet de lire et manipuler des données tabulaires (fichiers CSV, Excel…) sous forme de DataFrame. L'alias pd est une convention universelle qui permet d'écrire pd.read_csv() au lieu de pandas.read_csv(), rendant le code plus court et plus lisible.

Question 2 — sep=';'

Question : À quoi sert le paramètre sep=';' dans pd.read_csv() ?

> Par défaut, read_csv() suppose que les colonnes sont séparées par des virgules. Le fichier villes_coo.csv utilise le point-virgule (;) comme séparateur. Le paramètre sep=';' indique à pandas quel caractère délimite les colonnes afin d'interpréter correctement le fichier.

Question 3 — Valeur de retour si ville non trouvée

Question : Que retourne la fonction obtenir_coordonnees_de_csv() si la ville n'est pas trouvée ?

> La fonction retourne le tuple (None, None). Le test if not ville_data.empty vérifie si le DataFrame filtré contient au moins une ligne. S'il est vide, le bloc else affiche un message et retourne (None, None). Dans main(), la condition if latitude is not None empêche d'utiliser des coordonnées invalides.

Question 4 — popup vs tooltip

Question : Quelle est la différence entre popup et tooltip dans Folium ?

> popup : affiche une fenêtre contextuelle uniquement lorsque l'utilisateur clique sur le marqueur.
> tooltip : affiche un texte au survol de la souris, sans clic nécessaire.
> Dans le script, les deux sont utilisés simultanément pour maximiser l'accessibilité de l'information sur la carte.

Question 5 — if __name__ == '__main__':

Question : Pourquoi utilise-t-on le test if __name__ == '__main__': à la fin du script ?

> Cette condition garantit que le code dans main() ne s'exécute que lorsque le fichier est lancé directement (python villes_coo.py). Si le fichier est importé comme module dans un autre script, __name__ vaut le nom du module et non '__main__', donc main() n'est pas appelée automatiquement. C'est une bonne pratique de structuration en Python.

Partie 2 — Exploration du fichier CSV

2.1 — Structure du fichier villes_coo.csv

Information	Valeur observée
Nombre de lignes (hors en-tête)	39 146 villes
Noms des colonnes	ville, latitude, longitude
Séparateur utilisé	; (point-virgule)
Exemple de ville (ligne 2)	Ville Du Pont
Latitude	46.999873398
Longitude	6.498147193

2.2 — Script de manipulation pandas

Corrigé complet

```
import pandas as pd

# Lecture du fichier
data = pd.read_csv('./villes_coo.csv', sep=';')

# Afficher les 5 premières lignes
print(data.head())

# Afficher le nombre total de villes
print('Nombre de villes :', len(data))
# Equivalent :
print('Nombre de villes :', data.shape[0])
```

Explication : data.head() affiche par défaut les 5 premières lignes du DataFrame. len(data) ou data.shape[0] retournent le nombre de lignes, soit 39 146 villes.

Partie 3 — Exercices de programmation

Exercice 1 — Recherche insensible à la casse

Corrigé

```
def obtenir_coordonnees_de_csv(nom_ville, fichier_csv):
    data = pd.read_csv(fichier_csv, sep=';')
    # Comparaison insensible à la casse
    ville_data = data[data['ville'].str.lower() == nom_ville.lower()]
    if not ville_data.empty:
        latitude = ville_data['latitude'].values[0]
        longitude = ville_data['longitude'].values[0]
        return latitude, longitude
    else:
        print(f'Coordonnées non trouvées pour {nom_ville}.')
        return None, None
```

Explication : str.lower() convertit toutes les valeurs de la colonne 'ville' en minuscules. nom_ville.lower() fait de même pour la saisie. La comparaison == fonctionne alors quelle que soit la casse saisie par l'utilisateur ('angers', 'ANGERS', 'Angers' donnent le même résultat).

Exercice 2 — Afficher plusieurs villes

Corrigé complet

```
def main():
    fichier_csv = './villes_coo.csv'
    saisie = input('Entrez des villes séparées par des virgules : ')
    villes = [v.strip() for v in saisie.split(',')]

    # Carte centrée sur la France
    carte = folium.Map(location=[46.5, 2.3], zoom_start=6)

    for nom_ville in villes:
        lat, lon = obtenir_coordonnees_de_csv(nom_ville, fichier_csv)
        if lat is not None:
            folium.Marker(
                [lat, lon],
                popup=nom_ville,
                tooltip=nom_ville
            ).add_to(carte)

    carte.save('carte_multi.html')
    print('Carte enregistrée sous carte_multi.html')
```

Explication : split(',') découpe la chaîne à chaque virgule. La compréhension [v.strip() for v in ...] supprime les espaces superflus. La boucle for ajoute un marqueur par ville trouvée. La carte est centrée sur la France (46.5, 2.3) avec zoom_start=6 pour voir tout le territoire.

Exercice 3 — Marqueurs colorés

Corrigé

```
couleurs = ['red', 'blue', 'green', 'orange', 'purple']

for i, nom_ville in enumerate(villes):
    lat, lon = obtenir_coordonnees_de_csv(nom_ville, fichier_csv)
    if lat is not None:
        couleur = couleurs[i % len(couleurs)]
        folium.Marker(
            [lat, lon],
            popup=nom_ville,
            tooltip=nom_ville,
            icon=folium.Icon(color=couleur, icon='info-sign')
        ).add_to(carte)
```

Explication : enumerate(villes) retourne un couple (indice i, valeur) à chaque itération. L'opérateur modulo % permet de cycler dans la liste : avec 5 couleurs et 7 villes, les indices sont 0,1,2,3,4,0,1. Ainsi on ne dépasse jamais les bornes de la liste couleurs.

Partie 4 — Pour aller plus loin

Exercice 4 — Toutes les communes d'un département

Corrigé complet

```
import pandas as pd
import folium

def afficher_communes_departement():
    fichier_csv = './villes_coordonnees.csv'
    data = pd.read_csv(fichier_csv, sep=';', low_memory=False)

    departement = input('Entrez le nom du département (ex : Ain) : ').strip()

    # Filtrer par département
    communes = data[data['DEP_NOM'].str.lower() == departement.lower()]

    if communes.empty:
        print(f'Aucune commune trouvée pour : {departement}')
        return

    print(f'{len(communes)} communes trouvées dans {departement}.')
    carte = folium.Map(location=[46.5, 2.3], zoom_start=8)

    for _, row in communes.iterrows():
        geo = str(row['geolocalisation'])
        if ',' in geo:
            parts = geo.split(',')
            try:
                lat = float(parts[0].strip())
                lon = float(parts[1].strip())
                folium.CircleMarker(
                    location=[lat, lon],
                    radius=4,
                    color='steelblue',
                    fill=True,
                    fill_opacity=0.7,
                    tooltip=row['COM_NOM']
                ).add_to(carte)
            except ValueError:
                pass  # Ignorer les coordonnées invalides

    nom_fichier = f"carte_{departement.replace(' ', '_')}.html"
    carte.save(nom_fichier)
    print(f'Carte enregistrée : {nom_fichier}')

if __name__ == '__main__':
    afficher_communes_departement()
```

Points clés : low_memory=False évite les avertissements de pandas sur les types mixtes. La colonne 'geolocalisation' contient '46.15, 4.92' (lat et lon dans une seule chaîne) : on la découpe avec split(',') puis on convertit en float. CircleMarker est préféré à Marker classique pour les performances avec de nombreux points.

Exercice 5 — Questions de réflexion

Question A — Limites du script

Question : Citez deux limites du script actuel liées à la qualité des données.

Corrigé

- Sensibilité à la casse et aux accents : 'beziers' échoue si le fichier contient 'Béziers'. Il faut normaliser les chaînes (minuscules + suppression des accents via unicodedata.normalize).
- Homonymes : certaines villes partagent le même nom dans des départements différents. Le script retourne toujours le premier résultat sans avertir l'utilisateur. Il faudrait afficher toutes les correspondances et laisser l'utilisateur choisir.
- Performance : le fichier CSV est relu à chaque appel de la fonction. Il vaudrait mieux charger le DataFrame une seule fois dans main() et le passer en paramètre.

Question B — Alternatives à Folium

Question : Quelle bibliothèque alternative à Folium pourrait-on utiliser pour créer des cartes en Python ?

Corrigé

- Plotly Express : cartes interactives intégrables dans des dashboards (Dash).
- GeoPandas + Matplotlib : cartes statiques de haute qualité avec des données géospatiales (shapefiles, GeoJSON).
- Bokeh : visualisation interactive incluant des outils cartographiques.
- Kepler.gl : visualisation de grandes quantités de données géospatiales.

Question C — Format HTML

Question : Dans quel format est enregistrée la carte ? Quels sont les avantages de ce format ?

Corrigé

La carte est enregistrée au format HTML. Ce fichier contient du JavaScript (bibliothèque Leaflet.js) et des données intégrées.

Avantages :

- Portable : ouvrable directement dans tout navigateur sans installation de logiciel.
- Interactif : zoom, déplacement, clic sur les marqueurs grâce au JavaScript embarqué.
- Partageable : envoyable par email ou hébergeable sur un serveur web.
- Autonome : toutes les dépendances sont incluses ou chargées depuis un CDN.
