

TP DU GROUPE AUYOT

Membres : Cyril HAYOT et Enzo AUDOUI

Objectif : mettre en pratique la théorie sur un cas concret

Sujet : Distribution optimale colis dans les communes de Guadeloupe.

- Modélisation d'un sujet par les graphes
- Utilisation des graphes pour résoudre un problème
- Utilisation pratique d'une bibliothèque avec Python

Partie 2 : Rechercher les bibliothèques de manipulation de graphes disponible en python et gratuite et justifier votre choix

Consigne additionnelle :

- A) Charger de manière extérieure un graphe de la Guadeloupe.
- B) Déterminer le nombre de nœuds.
- C) Le parcours du graphe soit possible.

Bibliothèques de Manipulation de Graphes en Python

1. **NetworkX**: C'est une bibliothèque Python très populaire pour la création, la manipulation et l'étude de la structure, de la dynamique et des fonctions des réseaux complexes. Elle est particulièrement adaptée pour les graphes de grande taille et fournit de nombreuses fonctions pour l'analyse de graphes.
2. **igraph**: Une autre bibliothèque très efficace pour la manipulation de graphes complexes et réseaux. Elle est connue pour sa rapidité et sa capacité à gérer des graphes de grande taille.
3. **Graph-tool**: C'est une bibliothèque Python qui excelle en termes de performance grâce à sa mise en œuvre en C++. Elle est appropriée pour les graphes complexes et de grande taille, mais peut être un peu plus difficile à installer

Bibliothèque de visualisation de données en Python

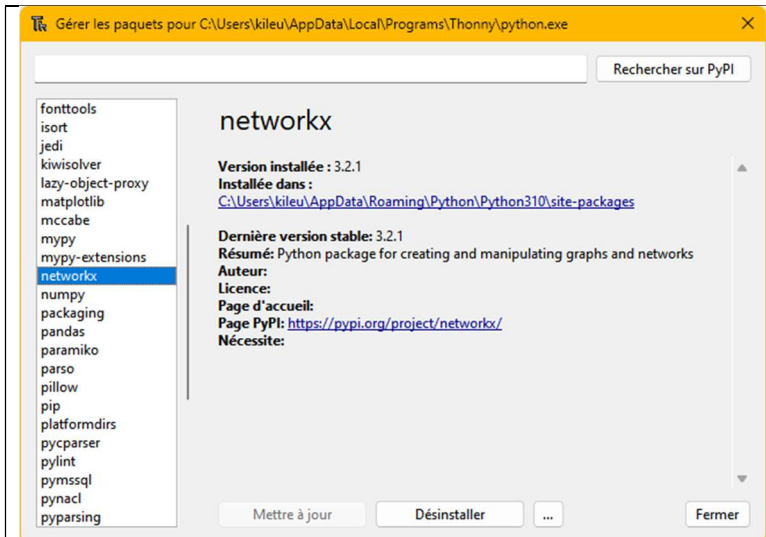
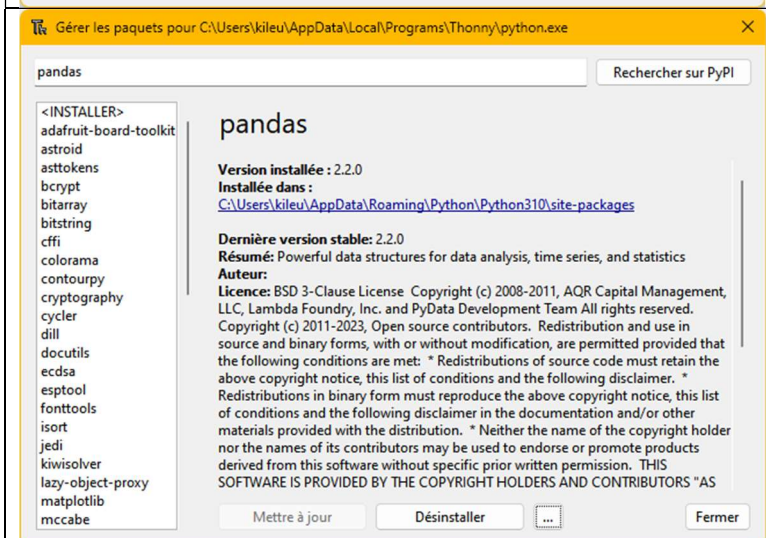
Il existe plusieurs bibliothèques de visualisation de données en Python, chacune offrant différentes fonctionnalités pour créer des graphiques statiques, animés et interactifs. Voici quelques-unes des plus populaires :

1. **Matplotlib** : C'est probablement la bibliothèque de visualisation de données la plus connue en Python. Elle est très versatile, permettant de créer une grande variété de graphiques statiques et animés. Matplotlib est idéal pour les tracés simples et les graphiques personnalisés.
2. **Seaborn** : Basé sur Matplotlib, Seaborn facilite la création de graphiques plus complexes et esthétiques avec moins de code. Il est particulièrement bon pour les visualisations statistiques et fonctionne bien avec les DataFrame de Pandas.
3. **Plotly** : Cette bibliothèque est excellente pour créer des visualisations interactives et des graphiques animés. Elle permet de créer des visualisations très sophistiquées qui peuvent être facilement partagées en ligne.

Bibliothèque d'extraction des données en Python

pandas fournit de nombreuses fonctions pour lire, filtrer, et modifier des ensembles de données. Elle permet de lire et écrire des données dans une variété de formats, y compris CSV, Excel, JSON, HTML, et SQL, entre autres.

Justification du Choix de Networkx, Matplotlib et Pandas

	<p>Pour ce projet, NetworkX semble être le meilleur choix. Elle est largement utilisée, bien documentée et relativement facile à prendre en main. NetworkX est particulièrement adaptée pour la manipulation de graphes où les données sur les nœuds et les arêtes sont importantes, ce qui est souvent le cas dans les problèmes de distribution.</p>
	<p>Matplotlib elle, est une bibliothèque de visualisation de données puissante et flexible. Elle nous permettra de créer des graphiques et des cartes qui rendent nos données immédiatement compréhensibles.</p>
	<p>Quand a Pandas, il nous offre des capacités de traitement des données qui vont bien au-delà des possibilités des structures de données standard de Python. C'est notre accoutumance a cet bibliothèque qui a motiver notre choix.</p>

En résumé, l'utilisation de **Pandas**, **NetworkX** et **Matplotlib** nous permet de gérer, analyser et visualiser des données complexes de manière efficace et approfondie. Ces bibliothèques constituent le socle technologique de notre projet, nous dotant des outils nécessaires pour conduire une étude complète et rigoureuse du réseau routier en Guadeloupe.

Code :

```
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt

# Chemin du fichier Excel contenant les données de liaisons routières
new_file_path = "C:\\Users\\zozo-\\OneDrive\\Documents\\distance route guadeloupe.xlsx"

# Étape 1 : Lire les données du fichier Excel
new_excel_data = pd.read_excel(new_file_path)

# Étape 2 : Créer un nouveau graphe vide avec des arêtes dirigées
G_bidirectional = nx.DiGraph() # Utilisation de DiGraph pour les arêtes dirigées

# Étape 3 : Ajouter des arêtes bidirectionnelles avec les distances du fichier Excel
for _, row in new_excel_data.iterrows():
    # Ajout d'une arête dans chaque direction
    G_bidirectional.add_edge(row['Noeud de Départ'], row['Noeud d\\'Arrivée'], weight=row['Distance (km)'])
    G_bidirectional.add_edge(row['Noeud d\\'Arrivée'], row['Noeud de Départ'], weight=row['Distance (km)'])

# Étape 4 : Calculer le nombre de nœuds dans le graphe
node_count = len(G_bidirectional.nodes)

# Étape 5 : Afficher le nombre de nœuds dans le graphe
print("Nombre de nœuds dans le graphe :", node_count)

# Étape 6 : Utiliser kamada_kawai_layout pour un meilleur espacement
pos_bidirectional = nx.kamada_kawai_layout(G_bidirectional)

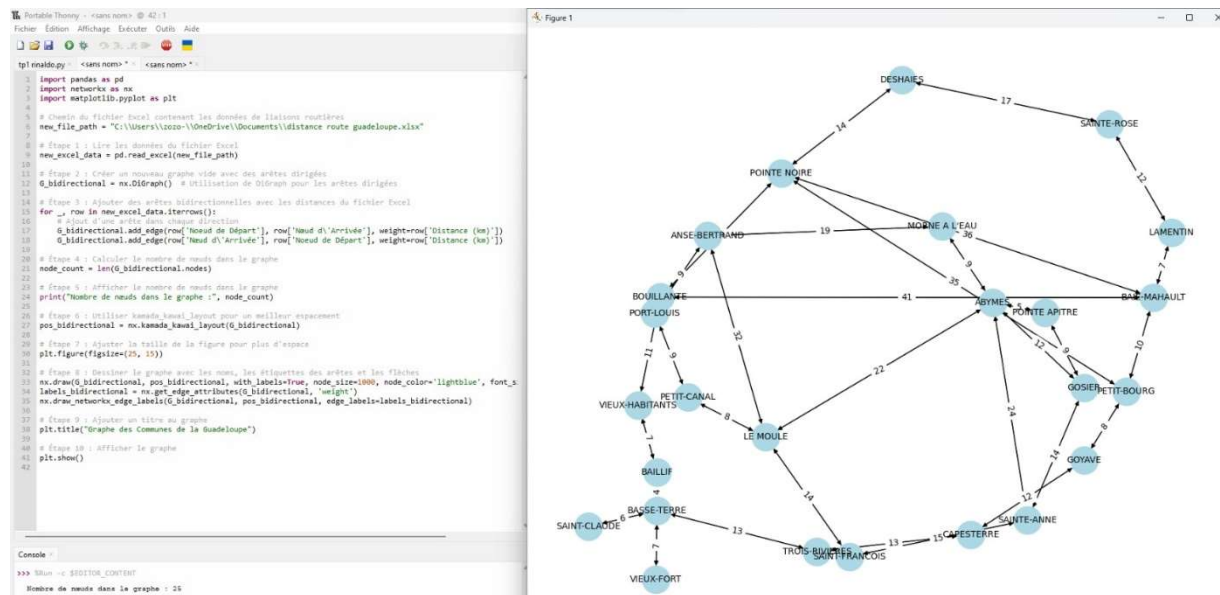
# Étape 7 : Ajuster la taille de la figure pour plus d'espace
plt.figure(figsize=(25, 15))

# Étape 8 : Dessiner le graphe avec les noms, les étiquettes des arêtes et les flèches
nx.draw(G_bidirectional, pos_bidirectional, with_labels=True,
        node_size=1000, node_color='lightblue', font_size=10, arrows=True)
labels_bidirectional = nx.get_edge_attributes(G_bidirectional, 'weight')
nx.draw_networkx_edge_labels(G_bidirectional, pos_bidirectional,
                             edge_labels=labels_bidirectional)

# Étape 9 : Ajouter un titre au graphe
plt.title("Graphe des Communes de la Guadeloupe")

# Étape 10 : Afficher le graphe
plt.show()
```

Exécution du Code:



Explication Du Code:

Il commence par spécifier le chemin d'accès au fichier Excel contenant les données de liaisons routières.

Ensuite, il lit les données à partir du fichier Excel (étape 1).

Un nouveau graphe vide est créé pour représenter les liaisons routières, avec des arêtes dirigées pour représenter les itinéraires bidirectionnels (étape 2).

Les arêtes bidirectionnelles sont ajoutées au graphe en utilisant les données du fichier Excel (étape 3).

Le code calcule le nombre de nœuds dans le graphe (étape 4).

Il affiche le nombre de nœuds dans le graphe (étape 5).

Le code utilise l'algorithme `kamada_kawai_layout` pour organiser les nœuds du graphe de manière à améliorer l'espacement (étape 6).

Il ajuste la taille de la figure pour une meilleure visualisation (étape 7).

Le graphe est dessiné avec les noms des nœuds, les étiquettes des arêtes et les flèches pour indiquer les directions (étape 8).

Un titre est ajouté au graphe (étape 9).

Enfin, le graphe est affiché à l'écran (étape 10).

Ce code permet de visualiser un graphe des communes de la Guadeloupe avec des liaisons routières bidirectionnelles entre elles, tout en affichant le nombre total de communes (nœuds) dans le graphe.

Parcours du graphe:

Nous avons choisi d'utiliser la bibliothèque networkx pour parcourir le graphe des communes de la Guadeloupe en utilisant des méthodes et des outils spécifiques. Voici pourquoi nous avons fait ces choix :

Utilisation de DiGraph (Graphe dirigé) : Nous avons opté pour DiGraph (Directed Graph) de networkx pour représenter notre graphe. Cette décision est justifiée par le fait que les liaisons routières entre les communes sont souvent bidirectionnelles. Les arêtes dirigées nous permettent de modéliser ces liaisons dans les deux sens, ce qui est essentiel pour notre analyse.

Parcours en largeur (BFS - Breadth-First Search) : Nous utilisons l'algorithme de parcours en largeur (BFS) fourni par networkx pour explorer le graphe à partir d'un nœud source donné. Le BFS est particulièrement adapté pour trouver le chemin le plus court entre deux nœuds et pour parcourir le graphe de manière systématique. Il nous permettra de répondre aux exigences de notre devoir.

Méthodes de visualisation intégrées : networkx propose des méthodes de visualisation intégrées qui nous permettent de générer des représentations graphiques de notre graphe. Cette fonctionnalité facilite la compréhension et la communication des résultats de manière visuelle.

Gestion des étiquettes (labels) : Nous utilisons les étiquettes (labels) pour annoter les arêtes de notre graphe avec les distances (en kilomètres) entre les communes. Cette fonctionnalité est disponible dans networkx et nous permet de fournir des informations détaillées sur les liaisons routières.

Facilité d'utilisation : networkx offre une interface conviviale qui nous permet de mettre en œuvre rapidement ces méthodes et outils. Sa syntaxe simple et sa documentation exhaustive facilitent la mise en place de notre projet.

En résumé, ces choix d'utilisation de méthodes telles que DiGraph, le parcours en largeur (BFS), les méthodes de visualisation, la gestion des étiquettes et la facilité d'utilisation de networkx sont justifiés pour atteindre les objectifs de notre devoir, notamment la création, l'analyse et la visualisation du graphe des communes de la Guadeloupe.