

## Introduction

Cette conversation montre comment un élève de seconde en SNT peut utiliser Python pour calculer des itinéraires à pied à partir de cartes OpenStreetMap (OSM), en adoptant une approche socratique pour comprendre étape par étape.

---

## 1. Bibliothèques Python classiques pour le routage OSM

### Question socratique :

Si je veux trouver un itinéraire sur une carte, quelles sont les étapes nécessaires ? - Récupérer les données de la carte - Transformer les routes en graphe - Calculer le plus court chemin - Afficher le résultat

### Librairies :

- **OSMnx** : Télécharge les données OSM, transforme en graphe et calcule les itinéraires.
  - **NetworkX** : Manipulation de graphes et calcul du plus court chemin.
  - **Pyrosm** : Lecture rapide des données OSM, utilisé avec NetworkX.
  - **GraphHopper / OSRM** : moteurs externes de routage via API.
- 

## 2. Exemple : Itinéraire Notre-Dame → Grande Mosquée de Paris

### Coordonnées :

- Notre-Dame : (48.8530, 2.3499)
- Grande Mosquée : (48.8422, 2.3553)

### Code Python :

```
import osmnx as ox
import networkx as nx
import matplotlib.pyplot as plt

notre_dame = (48.8530, 2.3499)
mosquee = (48.8422, 2.3553)

G = ox.graph_from_point(notre_dame, dist=2000, network_type="walk")
start_node = ox.distance.nearest_nodes(G, notre_dame[1], notre_dame[0])
end_node = ox.distance.nearest_nodes(G, mosquee[1], mosquee[0])
route = nx.shortest_path(G, start_node, end_node, weight='length')
fig, ax = ox.plot_graph_route(G, route, route_linewidth=3, node_size=0)
```

## Compréhension :

- Graphe piéton → `network_type='walk'`
- Plus court chemin → `weight='length'`
- Nœud le plus proche → car le graphe est discret

## 3. Itinéraire Notre-Dame → Sacré-Cœur avec fond blanc et itinéraire rouge

### Coordonnées :

- Notre-Dame : (48.8530, 2.3499)
- Sacré-Cœur : (48.8867, 2.3431)

### Code Python :

```
import osmnx as ox
import networkx as nx
import matplotlib.pyplot as plt

notre_dame = (48.8530, 2.3499)
sacre_coeur = (48.8867, 2.3431)

G = ox.graph_from_point(notre_dame, dist=4000, network_type="walk")
start_node = ox.distance.nearest_nodes(G, notre_dame[1], notre_dame[0])
end_node = ox.distance.nearest_nodes(G, sacre_coeur[1], sacre_coeur[0])
route = nx.shortest_path(G, start_node, end_node, weight="length")

fig, ax = ox.plot_graph_route(
    G,
    route,
    route_color="red",
    route_linewidth=4,
    node_size=0,
    bgcolor="white",
    edge_color="lightgray",
    show=False,
    close=False
)

ax.text(
    0.98, 0.98,
    "Itinéraire vers le Sacré Cœur",
    transform=ax.transAxes,
    ha="right",
    va="top",
    fontsize=12,
    color="black",
    bbox=dict(facecolor="white", edgecolor="none", alpha=0.8)
```

```
)  
  
plt.show()
```

## 4. Itinéraire Notre-Dame → Jardin du Luxembourg avec fond blanc et itinéraire bleu

### Coordonnées :

- Notre-Dame : (48.8530, 2.3499)
- Entrée Jardin du Luxembourg (rue de Vaugirard) : (48.8483, 2.3372)

### Code Python :

```
import osmnx as ox  
import networkx as nx  
import matplotlib.pyplot as plt  
  
notre_dame = (48.8530, 2.3499)  
luxembourg = (48.8483, 2.3372)  
  
G = ox.graph_from_point(notre_dame, dist=2500, network_type="walk")  
start_node = ox.distance.nearest_nodes(G, notre_dame[1], notre_dame[0])  
end_node = ox.distance.nearest_nodes(G, luxembourg[1], luxembourg[0])  
route = nx.shortest_path(G, start_node, end_node, weight="length")  
  
fig, ax = ox.plot_graph_route(  
    G,  
    route,  
    route_color="blue",  
    route_linewidth=4,  
    node_size=0,  
    bgcolor="white",  
    edge_color="lightgray",  
    show=False,  
    close=False  
)  
  
ax.text(  
    0.02, 0.02,  
    "Itinéraire vers le Luxembourg",  
    transform=ax.transAxes,  
    ha="left",  
    va="bottom",  
    fontsize=12,  
    color="black",  
    bbox=dict(facecolor="white", edgecolor="none", alpha=0.8)  
)
```

```
plt.show()
```

---

## Conclusion

- **OSMnx + NetworkX** sont suffisants pour récupérer des données OSM, construire des graphes et calculer des itinéraires.
  - On peut modifier la **couleur du tracé**, le **fond**, et ajouter des **annotations** avec Matplotlib.
  - L'approche socratique aide à **comprendre chaque étape** avant de coder.
- 

**Fin du mini-cours SNT**