1 Introduction

Ce TP vous invite à manipuler des bases de données directement depuis Python.

Vous devez disposer dans votre répertoire de travail du fichier `dots.sqlite`[1]. Il s'agit d'un fichier sqlite, que vous pourriez explorer avec sqliteman ou sqlitemanager. Ce fichier est téléchargeable sur le site. Sa structure est détaillée plus bas.

Pour importer le fichier avez pyzo, vous devez taper les lignes suivantes :

- `import sqlite3 as lite` (charge le module)
- `dbh=lite.connect('dots.db')` (ouvre le fichier de base de donnée).
- `cur=dbh.cursor()` (`cur` est alors l'outil de manipulation de la base de donnée, il « pointe » dessus, d'où le nom de curseur).

On execute une requête SQL avec :
`cur.execute("SELECT (...) FROM (...);")`
(la partie entre guillemets est une requête SQL standard).

Puis on récupère les résultats avec :
`result=cur.fetchall()`
Après cette ligne, `result` est une liste de tuples, c'est à dire qu'on peut travailler avec une boucle `for i in result` classique.

2 Manipulation basique de la base de donnée

Nous allons pour le moment utiliser deux tables de la base : `depts`, qui décrit les départements de France métropolitaine, et dont les champs utiles sont :

- `id` (entier)
- `code` (chaine de caractères)
- `nom` (chaine de caractères)
- `chef_lieu` (entier, id du chef lieu dans la table des communes)

et `pointsdep` qui en décrit les contours. Les contours d'un département sont décrits par un ou plusieurs polygones (s'il y a une ile, ou une enclave). Les polygones sont décrits par les coordonnées des sommets. Ainsi, les champs sont :

- `id` (entier)
- `iddept` (entier, id du département)
- `poly` (entier, numéro du polygone)
- `ordre` (entier, numéro du point)
- `x` et `y` (flottants, longitude et latitude du point)

1. Produit à partir des données publiées librement sur `https://www.data.gouv.fr`.

▶ **Question 1** *Tester par exemple la requête* `SELECT id,nom FROM depts;` *et afficher la réponse. Quel est le département d'ID 20 ? Combien y a-t-il de départements au total dans la base ?*

▶ **Question 2** *Écrire une fonction* `min_lat(id)` *qui renvoie la latitude minimale du département indexé par* `id`*.*

```
>>> min_lat(5)
44.18648337183157
```

▶ **Question 3** *Afficher la liste des noms de départements triés par ordre décroissant du nombre de points dans leur contour (le max est le Finistère avec près de 900 points).*

▶ **Question 4** *Écrire une fonction* `points(dept,poly)` *qui renvoie la liste des points du département* `dept` *et du polygone* `poly`*, triés par numéro d'ordre.*

▶ **Question 5** *Donner la liste des départements ayant des enclaves (extérieures ou intérieures). Il y en a 26.*

3 Avec `matplotlib`

Dans cette partie, on utilise le package matplotlib, qu'on chargera avec
`import matplotlib.pyplot as plt`

On pourra également vérifier qu'on peut tracer un polygone avec `plt.plot(x,y)`[2], où `x` et `y` sont les listes des coordonnées du polyone. Le premier point doit être répété à la fin (mais c'est déjà le cas dans la base de donnée). On effacera un dessin en fermant simplement la fenêtre graphique.

▶ **Question 6** *Écrire une fonction* `polyreg(n)` *qui affiche un polygone régulier à* `n` *côtés. On pourra utiliser* `plt.axis("equal")` *pour imposer des axes orthonormés.*

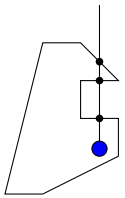
▶ **Question 7** *Afficher le contour du département du Rhône (on peut commencer par en chercher l'id, puis par regarder le nombre de polygônes).*

▶ **Question 8** *Afficher la carte de France départementale.*

Notez que cette carte est beaucoup plus jolie si on fait : `plt.axes().set_aspect(1.5)`*. Expliquer pourquoi.*

2. Puis éventuellement `plt.show()`.

4 Un peu de géométrie algorithmique



Note : pour déterminer si un point est dans un polygone, nous allons tracer un rayon partant de ce point dans n'importe quelle direction (nous choisirons le nord ici) et on compte combien de segments on coupe. Si c'est un nombre impair, alors le point est dans le polygone. On négligera les cas trop pathologiques de segments exactement alignés.

▶ **Question 9** *Écrire une fonction* `intersection(x,y,xA,yA,xB,yB)` *qui détermine si la demi-droite partant vers le haut depuis $M(x, y)$ coupe le segment $[AB]$. On suggère de faire intervenir les déterminants pour trouver une expression simple.*

▶ **Question 10** *Écrire une fonction* `est_dans(x,y,iddept)` *qui détermine si le point $M(x, y)$ se situe dans le département* `iddept`*.*

▶ **Question 11** *Écrire une fonction* `departement(x,y)` *qui renvoie l'id du département contenant le point (x,y), ou -1 si le point n'est dans aucun département.*

5 Avec les communes

Les tables `communes` et `pointscomm` suivent la même structure que les deux précédentes.

▶ **Question 12** *Adapter la fonction* `departement(x,y)` *pour en faire une fonction* `commune(x,y)` *qui détermine dans quelle commune se situe le point $M(x, y)$. Quelle est la complexité de cette fonction ?*

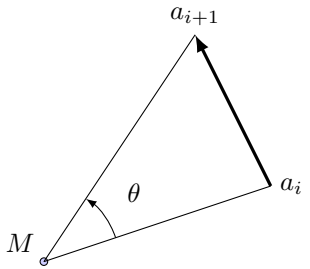
▶ **Question 13** *Pour accélérer les choses, on a l'idée de pré-générer, pour chaque commune, un encadrement des abscisses et ordonnées. Écrire une fonction qui retourne la liste des quintuplets de la forme* `(code,xmin,xmax,ymin,ymax)` *pour chaque commune. On pourra utiliser une requête SQL pour générer directement ces valeurs.*

On appellera `limites` le tableau contenant ces quintuplés.

▶ **Question 14** *Réécrire la fonction* `commune(x,y)` *pour qu'elle utilise la table* `limites` *et qu'elle soit plus rapide. Comment pourrait-on décrire la complexité de cette fonction ?*

6 Winding number

Une autre manière de déterminer si un point est à l'intérieur d'un polygone ou non consiste à compter le nombre de fois où le polygone tourne autour du point. Une première manière de réaliser cela est de calculer l'angle algébrique (cf figure ci-dessous) pour chaque segment.



▶ **Question 15** *Trouver une formule permettant de calculer l'angle θ à partir des coordonnées de M, de a_i et de a_{i+1}.*

Cette méthode est en réalité assez peu efficace parce qu'elle nécessite des calculs d'arccosinus. On peut cependant l'accélérer de la manière suivante :

— Choisir une direction arbitraire.

— Compter le nombre de fois où un segment passe cette direction arbitraire (+1 si on le passe dans le sens trigonométrique, −1 sinon).

— Conclure à partir de ce résultat (on peut choisir d'interpréter si un point ayant un résultat de 2 se trouve dans ou hors du polygone).

▶ **Question 16** *Programmer cette fonction* `winding_number(x,y,id)` *pour un département ou une commune.*