Le but de ce TD est d'extraire des données d'une base de données avec python, et de tracer un graphique grâce à ces données.

I Base de données

Vous disposez d'une base de données nommée **ipt.sqlite3** contenant des données fictives de notes d'élèves. Cette base ne contient qu'une seule table

notes_details

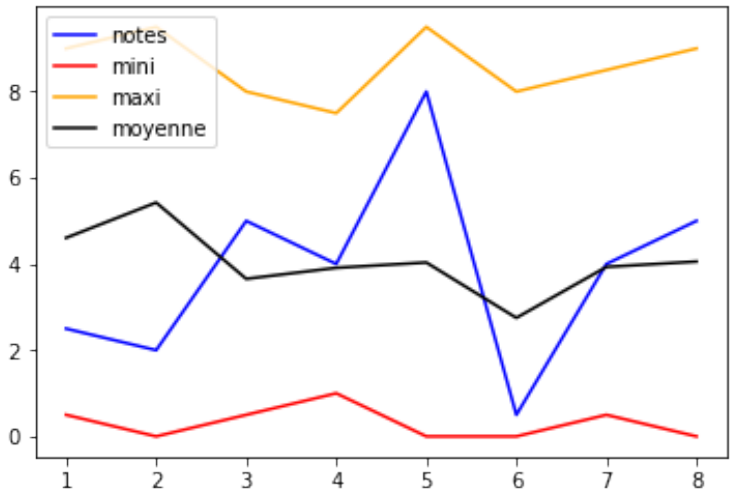
champs	type	description
id	entier	clé primaire
eleve_id	entier	identifiant de l'élève, pour l'anonymat
note	flottant	note obtenue, ou NULL si absent
devoir	texte	nom complet du devoir
semestre	entier	numéro du semestre
type_devoir	texte	type du devoir ('Interro', ou 'DS')

Exercice 1
Dans cet exercice, on pratique seulement l'exécution de requêtes, grâce au logiciel de gestion de base de données.

1. Trouver la liste des élèves et notes pour le devoir "Interro1".
2. Donner la note moyenne des devoir de type "Interro" pour le premier semestre, sans compter les absents (ajouter la condition "note is not NULL")
3. Donner le nombre d'élèves ayant eu une note supérieur à la moyenne pour le devoir "DS1".

II Extraction des données en python

Le but de cette partie est d'obtenir une graphique comme :



où l'on représente l'évolution des notes d'un élève choisi arbitrairement (ici celui d'id égale à 1), pour un type de devoir particulier (ici les interros)
Pou cela, dans la feuille de script `td7.py`, nous avons une fonction `select` qui prend comme argument une chaîne de caractère contenant une requête et retourne la liste des lignes correspondantes. Chaque ligne est un tuple (l'équivalent d'une liste, mais non modifiable)

```
1  select('SELECT * FROM notes_details WHERE id=1')
```

Pour que l'exemple précédent fonctionne, il faut absolument avoir placé la base de données et la feuille de script dans un même dossier, et avoir exécuté toute la feuille de script `td7.py` via F5 (avant de recopier l'exemple dans la console).
Comparer le résultat en python au résultat obtenu via votre logiciel de base de données.

2.1 Construction des données en python

Exercice 2 (Première requête)
Extraire la liste des notes de l'élève choisi (par son id, par exemple l'élève d'id valant 1) et pour le type de devoir choisi. Chaque ligne devra contenir la note et de devoir correspondant (dans cet ordre). La requête devra être ordonnée selon la colonne "devoir" pour trier par ordre alphabétique, et donc par numéro de devoir.
Vous pouvez stocker cette liste dans une variable que nous nommerons `notes_eleve` dans la suite.

Exercice 3
Chaque élément de la liste `notes_eleve` est de la forme (note, devoir). On suppose que les devoirs sont numérotés à partir de 1.
Créer deux listes, `notes` et `numeros` contenant respectivement les notes de l'élèves (courbe en bleu) et les numéros des devoirs (liste des abscisses).

Exercice 4
Pour chaque note de notre élève, il nous faut calcule la note minimale, la note maximale ainsi que la moyenne.
Compléter la fonction `statistiques` qui prend comme argument une liste de nombres et retourne la liste `[mini, maxi, moyenne]`. Vous ne devez pas utiliser les fonction `min, max, sum` de python, mais calculer toutes les valeurs voulues en une seule boucle.

Exercice 5 (Requêtes à foison)
Une première idée pour créer les 3 listes de données qui nous manque est la suivante :

- Pour chaque devoir contenu dans `notes_eleve`, on effectue une requête qui récupère toutes les notes (non NULL) associées à ce devoir.
- La table obtenue contient seulement des lignes de longueur 1 (chaque ligne est un tuple), on construit donc la liste des notes.
- Calculer les éléments statistiques voulus et les stocker dans chacune des 3 listes en construction

Créer suivant cette idée les 3 listes `Lmini`, `Lmaxi`, `Lmoy`

Exercice 6
Afficher à l'écran le graphique voulu. Voici quelques indications pour la mise en forme.

- il suffit d'exécuter plusieurs `plt.plot` à la suite pour afficher sur un même graphique (par exemple dans une fonction, ou en exécutant toute une feuille de script)
- le choix de la couleur est à vous. On utilise `plt.plot(X, Y, color='une chaine contenant un nom de couleur, en anglais')`
- Pour préparer la légende, on utilise `plt.plot(X, Y, color='une chaine contenant un nom de couleur, en anglais', label='le titre de la courbe')`
  Ensuite, pour placer la légende `plt.legend(loc='demander à votre moteur de recherche comment placer une légende en matplotlib')`

2.2 Utilisation de SQL

On peut rédiger des requêtes plus compliquées pour éviter le traitement statistique en python

Exercice 7
Trouver, à la main, le nom du premier devoir contenu dans `notes_eleve` et rédiger puis exécuter une requête qui retourne (en python) la note minimale, la note maximale et la moyenne pour ce devoir.
En déduire une autre méthode pour construire des listes notées `Lmini2`, `Lmaxi2`, `Lmoy2`

Exercice 8
Le problème fondamental des deux méthodes proposées jusqu'à maintenant, est qu'il faut effectuer une requête par devoir.
On peut contourner ce problème en utilisant une clause `GROUP BY`. Écrire la requête (d'abord dans le logiciel de base de données, peut-être) permettant d'obtenir la note minimal, la note maximale et la moyenne par devoir de type "Interro" et pour l'élève sélectionné.
Il ne manque que la note de notre élève dans la table ainsi retournée.

Exercice 9 (La bonne requête)
Pour obtenir la note de notre élève en plus, on utilise une sous-requête. Ajouter, dans la clause `SELECT` de l'exercice précédent,

```
(SELECT note FROM notes_details WHERE eleve_id=1 and devoir=nd.devoir) as note
```

Cette donnée (scalaire) sera ajoutée à chaque ligne, à condition que la clause `FROM` donne le nom `nd` comme alias à la table `notes_details`, sous la forme

```
FROM notes_details nd
```

La requête obtenue retourne une table contenant toutes les données nécessaires à la création du graphique voulu.
En python, à partir de cette table, créer les listes nécessaires et afficher le graphique correspondant.

Exercice 10
Notre approche manque un peu de généralité, dans le sens où nous avons choisi arbitrairement un élève et un type de devoir dès le début du TD.
Pour remédier à ceci, nous allons construire dans une fonction la chaîne de caractère contenant la bonne requête à exécuter, en utilisant la concaténation de chaînes. Pour rappel, si `s1` et `s2` sont des chaînes de caractères, alors la concaténation de ces chaînes s'obtient par

```
1  s1 + s2
```

Attention, ceci crée seulement la chaîne concaténée, sans la stocker.
Compléter la fonction `requete` puis la fonction `evolution`