TP Python et PostgreSQL – Premier contact

— Rendez les programmes de différentes questions sur le elearning.
— Chaque question comporte une partie SQL pour extraire de l'information d'une base et une partie python pour communiquer avec la base et traiter cette information. N'hésitez pas à réfléchir à la partie SQL indépendamment du reste.

▸ **Exercice 1 :** Python et PostgreSQL - encore un peu d'usimag

Dans cet exercice, nous allons construire un programme Python qui se connecte à une base de données, lui envoie des requêtes et affiche les résultats.

Préparatifs

1. Nous allons utiliser la base de données usimag2 fournie pour le TP3. Chargez cette base de données dans votre base étudiant.
2. Vous allez avoir besoin de l'exemple donné dans le cours pour répondre aux questions. Assurez-vous d'avoir les slides du cours à portée de main.
3. Savez-vous encore lancer un programme python ? Pour les têtes en l'air, rappellons que pour éxécuter le fichier `blabla.py`, il faut vous placer dans le même bon répertoire et lancez la commande `python3 blabla.py`.
4. Savez-vous encore concaténer les chaines de caractères en python ? Cela se fait avec l'opérateur `+`. Par exemple `'machin'+'truc'` forme la chaîne `'machintruc'`
5. Un gentil chargé de TP vous a préparé un programme python à compléter. Téléchargez ce fichier appelé `python_postgresql.py` depuis la page du TP sur elearning.

C'est parti !

1. Remplissez maintenant les trous dans le fichier `python_mysql.py` comme cela est demandé dans les commentaires du fichier afin que le programme se connecte à votre base de données et affiche le nom des tables présentes.
2. En vous inspirant de la question précédente, proposez un programme Python que vous appellerez `couleur.py` qui demande à l'utilisateur d'entrer une couleur au clavier et affiche la liste des produits dont il existe une version de cette couleur. Le programme affichera une sortie de ce type :

```
entrez une couleur : rouge
Produit disponibles dans la couleur rouge
(u'tabouret',)
(u'ordinateur',)
(u'bottes',)
(u'table',)
```

Indice : Faites une copie du programme précédent et modifiez le. Cela vous fera gagner du temps.

3. Proposez maintenant un programme appelé `python_magasin.py` qui se connecte à la base de données et qui, pour chaque magasin, affiche son nom et la ville dans laquelle il est situé. Les lignes affichées par le programme doivent être de la forme suivante :

   ```
   Le magasin Stock10 est situe dans la ville Paris
   ```

4. Proposez enfin un programme `python_provenance.py` qui se connecte à la base de données et qui, pour chaque enregistrement de la table `provenance` affiche le contenu de toutes les colonnes.
   Ajustez votre requête pour que les enregistrements arrivent par ordre croissant de quantité. *Indice : voir la clause* `ORDER BY` *dans en cours lundi.*

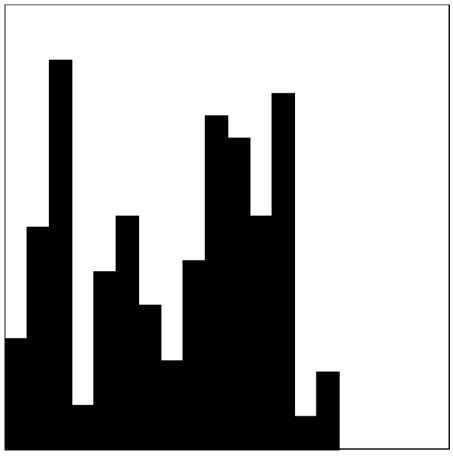
   L'affichage du programme doit être de la forme suivante :

   ```
   Produit : 20, Usine : 402, Magasin : 14, Quantite : 13
   Produit : 16, Usine : 189, Magasin : 20, Quantite : 18
   Produit : 16, Usine : 213, Magasin : 16, Quantite : 33
   Produit : 16, Usine : 302, Magasin : 14, Quantite : 49
   [...]
   Produit : 5, Usine : 302, Magasin : 20, Quantite : 858
   Produit : 4, Usine : 200, Magasin : 18, Quantite : 985
   Produit : 15, Usine : 189, Magasin : 20, Quantite : 1958
   Produit : 8, Usine : 302, Magasin : 16, Quantite : 2000
   ```

▸ **Exercice 2 :** Skyline pure python avec une pointe d'upem.tk

On va réaliser un programme python dessinant la sillouette d'un centre-ville plein de gratte-ciels. *(vous devriez avoir une sensation de déjà vu.)*

1. Téléchargez la bibliothèque `upemtk.py` et le fichier `dessin-init.py`. Lancez le programme, regardez le code du programme et modifiez le afin de (1) modifier la couleur du disque en rouge et (2) afficher le rectangle en bas à gauche.
2. Déclarez dans le programme une liste `hauteurs` des hauteurs des immeubles de la ville. Choisissez une quinzaine de valeurs comprises entre 30 et 400. Dans l'exemple : `hauteurs=[100,200,350,40,160,210,130,80,170,300,280,210,320,30,70]`
3. Modifiez enfin l'affichage pour afficher une suite de rectangles noirs de largeur fixe de 20 et dont les hauteurs sont données par la liste `hauteurs` :



▸ **Exercice 3 :** Skyline de SQL city

On veut maintenant dessiner la sillouette d'une ville dont les informations sont situées dans la base de donnée `batiments.sql` donnée sur la page du cours.

Préparatifs

1. Videz votre base de données de la base usimag2 en utilisant le script `drop-usimag.sql` donné au TP1 et chargez dans votre base les données du fichier `batiments.sql`
2. Explorez un peu cette nouvelle base pour découvrir ce qu'elle contient.
3. Gardez à portée de main les programmes `python_provenance.py` et `dessin-init.py` des exercices précédents. Ils seront une source d'inspiration bien utile.

A vous de jouer !

1. Créez un programme `python_sqlcity.py` qui extrait de la table `batiment` la liste des paires (référence de bâtiment, hauteur).
2. Modifiez ce programme pour qu'il n'affiche plus la liste des info mais qu'il dessine la sillouette des bâtiments de la même façon que dans l'exercice précédent c-a-d de couleur noire avec une largeur fixe de 20 pixels.
3. Proposez, en vous basant sur votre réponse à la question précédente, un programme Python `python_colorcity.py` qui utilise le champs `ref_type` de la table `batiment` ainsi que la table `type` pour obtenir la largeur et la couleur de chacun des bâtiments et dessine les bâtiments avec leur largeur et couleur respectives.
   Indice : Pensez à utiliser une clause `ORDER BY`*pour obtenir la liste des batiments par ordre croissant de* `ref_batiment`.
4. Faites une copie modifiée du programme précédent que vous appellerez `python_circlecity.py` qui affiche en plus un cercle rouge de diamètre 10 pixels au sommet des bâtiments de couleur noire.
5. Enfin modifiez ce programme pour que le diamètre du cercle affiché soit égal à la largeur du bâtiment sur lequel il est dessiné.