CONCOURS
CPGE
EPITA - IPSA - ESME
1 SEUL ET UNIQUE CONCOURS
POUR VOUS PRÉPARER
À UNE MULTITUDE
DE MÉTIERS

Samedi 8 avril 2023

OPTION : SCIENCES DU NUMÉRIQUE

MP - PC - PSI - PT - TSI

Durée : 2 heures

Conditions particulières

Calculatrice interdite
Indiquez uniquement votre code candidat SCEI sur le QCM à insérer dans votre copie d'examen

Concours CPGE EPITA-IPSA-ESME 2023
Option Sciences du Numérique
Filières MP/PC/PSI/PT/TSI

Consignes Python

Tout code doit être écrit dans le langage Python, à l'exception des requêtes SQL présentes dans la section I.

- Veuillez indenter votre code correctement.
- Tout ce dont vous avez besoin (fonctions, méthodes) est indiqué ci-dessous.
- Vous pouvez écrire vos propres fonctions, dans ce cas elles doivent être documentées (on doit savoir ce qu'elles font). Dans tous les cas, la dernière fonction écrite doit être celle qui répond à la question.

Fonctions et méthodes autorisées

- Itérateurs fournis par la fonction `range`, itérateurs sur une liste.
- Slicing (extraction de portions) sur les listes.
- n-uplets.
- La fonction `len` donne la taille d'une liste ou d'une chaîne de caractères.
- Les méthodes utilisables sur les listes sont `append` et `pop`, tout autre méthode (`insert`, `sort`...) est exclue.
- Pour `s` une chaîne de caractères et `c` un caractère, `s.split(c)` s'évalue en une liste de chaînes de caractères obtenues par séparation de `s` autour de `c`. Par exemple, `'a bb ccc'.split(' ')` s'évalue en `['aa', 'bb', 'ccc']` et `'a,b,c,d'.split(',')` en `['a', 'b', 'c', 'd']`.
- Fonctions pour les fichiers :
  - `f=open(fichier, mode)` ouvre un « wrapper » vers un fichier. Le mode peut-être `'r'` (lecture), `'w'` (écriture) ou `'a'` (ajout).
  - `f.close()` ferme le « wrapper ».
  - `f.readlines()` : pour `f` un fichier ouvert en lecture, renvoie une liste de chaînes de caractères, chacune correspondant à une ligne du fichier. Le saut de ligne `'\n'` est présent en fin de chacune des lignes.
  - `+` : concaténation de chaînes de caractères.
- conversion de types : `int`, `float`, `str`, etc...

Introduction

Le sujet de l'option informatique contient trois sections, reliées par un thème commun. Elles sont indépendantes.

Partons faire de la randonnée !

I. SQL

Donald souhaite partir en randonnée. Tout d'abord, il doit choisir sa randonnée ! Il a récupéré une base de données contenant les différentes randonnées de son département, qui se présente comme suit.

- la table `rando` décrit les randonnées possibles. Les attributs sont les suivants :
  - `id` : une clé primaire entière ;
  - `nom` : le nom de la randonnée (chaîne de caractères) ;
- la table `pp` décrit des points de passage des différentes randonnées dans le département. Les attributs sont :
  - `id` : une clé primaire entière ;
  - `x`, `y`, `h` : trois attributs entiers donnant les coordonnées du point de passage dans un repère cartésien, en mètres.
- enfin, la table `appartient` possède trois attributs :
  - `rid` : une clé étrangère vers l'attribut `id` de `rando`.
  - `pid` : une clé étrangère vers l'attribut `id` de `pp`.
  - `num` : un numéro entier.

  La table `appartient` est construite de sorte que pour une randonnée comportant n points de passage, n lignes soient présentes et les numéros `num` associés sont exactement les entiers de 0 à $n-1$. Par exemple, si la randonnée numéro 2 démarre au point de passage 213, poursuit vers le point 478, puis vers le point 142, pour revenir au point de départ, la table `appartient` comportera les quatre lignes `(2, 213, 0)`, `(2, 478, 1)`, `(2, 142, 2)` et `(2, 213, 3)`. Un même point de passage peut appartenir à plusieurs randonnées.

Question 1. Rappeler la définition d'une clé primaire. Une clé primaire peut-elle comporter plusieurs attributs ? Donner une clé primaire de la table `appartient`.

Question 2. Dans cette question, on demande de répondre à chaque question posée avec une unique reqête SQL.

1. Donner le nombre d'entrées de la table `rando`.
2. Donner les abscisses et ordonnées minimales et maximales des points de passage de la table `pp`.
3. Donner les identifiants des points de passage de la randonnée « Le mont chauve ». On suppose que ce nom est bien présent dans la table `rando`, une unique fois.
4. Écrire une requête fournissant pour chaque identifiant de randonnée le nombre de points de passage. On ne gardera que les randonnées ayant au moins 10 points de passage.
5. Donner les couples d'identifiants de randonnées différentes qui partagent au moins un point de passage.

II. Calculer le temps de trajet

Donald a choisi sa randonnée. Les coordonnées des points de passage successifs sont inscrits sous la forme d'un fichier `rando.csv` dont les premières lignes sont :

```
Randonnee 42
x;y;z
123;-245;88
98;-259;92
125;-287;67
```

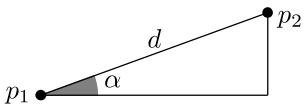
La dernière ligne termine par un saut de ligne également. Comme on le voit, à partir de la troisième ligne, les lignes sont formées de trois chaînes de caractères représentant un entier relatif, séparées par le caractère `';'`. Ces trois entiers représentent l'abscisse, l'ordonnée, et la hauteur du point de passage, en mètres.

Question 3. Le fichier `rando.csv` est dans le répertoire courant. Écrire un script Python permettant de récupérer les coordonnées des points de passage sous la forme d'une liste `Lrando` constituée de triplets d'entiers.

On assimile une randonnée à la succession des segments de droite entre les points de passage. Pour un tel segment, on appelle :

— *distance* entre les deux points de passage p_1 et p_2 la distance euclidienne d entre les points dans l'espace ;

— *pente* la tangente de l'angle algébrique α formé par la projection du segment sur le plan (Oxy) et le segment lui-même. Cet angle est dans l'intervalle $]-\pi/2, \pi/2[$.



Question 4. On suppose toutes les fonctions du module `math` importées, dont `sqrt` qui donne la racine carrée. Écrire une fonction `calcule_d_pente(L)` prenant en entrée une liste similaire à `Lrando` et renvoyant une liste constituée de couples (d, p) où d est la distance entre deux points de passage successifs et p est la pente. Si `L` contient n points de passage, la liste retournée par votre fonction contiendra $n-1$ éléments.

Donald, en temps que randonneur expert, a mesuré sa vitesse de marche dans un maximum de conditions possibles. Il a stocké cette information sous la forme de deux listes `Pente` et `Vitesse` de même taille, de sorte que `Vitesse[i]` est sa vitesse moyenne de marche sur un segment de pente `Pente[i]`. La liste `Pente` est strictement croissante. Les deux listes `Pente` et `Vitesse` sont définies globalement.

Question 5. Écrire une fonction `indice_pente(p)`, prenant en entrée une pente donnée `p`, et renvoyant l'unique indice `i` tel que `Pente[i]` $\leq p <$ `Pente[i+1]`. On supposera que `Pente[0]` $\leq p <$ `Pente[len(Pente)-1]`. On fera usage d'une méthode dichotomique pour assurer une complexité logarithmique en la taille de `Pente`.

Question 6. On note p_i et p_{i+1} deux pentes successives de la liste `Pente`, et v_i, v_{i+1} les vitesses associées. Pour $p \in [p_i, p_{i+1}[$, proposer une formule donnant une estimation cohérente de la vitesse à laquelle Donald peut espérer progresser sur un segment de pente p.

Question 7. Déduire des questions précédentes une fonction `temps_rando(L)`, prenant en entrée une liste similaire à `Lrando`, et renvoyant une estimation du temps que Donald mettra à faire sa randonnée.

III. Faire son sac...

La randonnée qu'a choisie Donald dure plusieurs jours. Bien qu'il soit un randonneur aguérri, il a la fâcheuse habitude de trop se charger et ne pas arriver à terminer ses randonnées pour cause de fatigue. Ce coup-ci, il veut être sûr d'emporter un maximum de choses nécessaires, sans dépasser le poids maximal qu'il s'est fixé.

Dans la suite, on travaillera avec une liste contenant des triplets comme `('telephone', 100, 50)` ou bien `('eau', 5000, 200)` ou encore `('barbecue', 20000, 80)`. Un tel triplet contient la description de l'objet (chaîne de caractères), son poids en grammes, et un entier qui est un score affectif que Donald a attribué à l'objet. Comme on le voit, Donald préfère son barbecue à son téléphone, mais son téléphone pèse beaucoup moins lourd. Il a aussi attribué un gros score à 5 litres d'eau, dont il a besoin pour survivre.

Donald s'est fixé un entier C qui est le poids maximal qu'il peut emporter avec lui, en grammes. Le problème qu'il doit résoudre est donc le suivant :

Maximiser la somme des scores affectifs des objets qu'il met dans son sac à dos, sans excéder le poids C.

III. 1. Un algorithme glouton pour le remplissage du sac

Une première idée pour résoudre le problème est de trier l'ensemble des objets par ratio $\frac{\text{score}}{\text{poids}}$ décroissant, puis considérer les objets un à un. Chaque objet est ajouté au sac si son poids aditionné aux objets déja choisis n'excède pas le poids autorisé C. Dans cette sous-section, on va implémenter cet algorithme.

Question 8. Dans cette question, on trie simplement une liste de nombres, on n'utilisera pas la fonction de tri de Python mais on demande de réécrire un tri à la main. Écrire une fonction `tri_selection(L)` prenant en entrée une liste de nombres et la modifiant de sorte qu'elle soit triée dans l'ordre croissant. On rappelle le principe du tri par sélection :

— chercher le plus petit élément de la liste et l'amener à l'indice 0 par un échange ;

— chercher le plus petit élément de la liste à partir de l'indice 1 inclus et l'amener à l'indice 1 par un échange ;

— etc...