DS 1

Informatique de tronc commun, classe de PC

Julien Reichert

Ce devoir consiste en une partie de programmation et deux parties sur des bases de données totalement indépendantes. Les requêtes à écrire ne sont pas forcément de difficulté croissante.

Partie 1 : Programmation

Exercice 1.1 : Écrire une fonction qui renvoie le minimum d'une liste de nombres.

Exercice 1.2 : Même exercice mais en supposant que la liste soit d'abord strictement décroissante puis strictement croissante. Il faut alors procéder par dichotomie et avoir une complexité logarithmique en la taille de la liste pour que la réponse soit prise en compte.

Exercice 1.3 : Écrire une fonction prenant en argument deux listes croissantes et déterminant si elles ont au moins un élément en commun. Calculer (avec détails) la complexité de la fonction.

Exercice 1.4 : Écrire une fonction prenant en entrée une liste de nombres et déterminant le plus grand produit de deux éléments à des indices distincts dans la liste. On admettra que la taille de la liste est au moins de deux.
Non, ce n'est pas aussi simple que cela en a l'air !

Exercice 1.5 : Écrire une fonction établissant un classement général en tenant compte de possibles égalités. La fonction aura pour paramètre une liste de couples (chaîne de caractères, valeur) et imprimera (oui, c'est exceptionnel mais on se servira de `print`) les chaînes de caractères dans l'ordre décroissant des valeurs associées à raison d'une par ligne en commençant les lignes par le classement.

Exemple de résultat attendu :

```
>>> classement([('a', 42), ('b', 57), ('c', 3), ('d', 20), ('e', 3)])

1 b
2 a
3 d
4 c
4 e
```

Partie 2 : Base de données « enseignement »

La base de données pour la deuxième partie est celle du cours :

— Table `ETUDIANTS`, avec les attributs `id`, `nom`, `prenom` et `classe`.
— Table `EXAMENS`, avec les attributs `id`, `date`, `matiere` et `coeff`.
— Table `NOTES`, avec les attributs `etudiant`, `examen` et `note`.

Exercice 2.1 : Rappeler les clés (dont les clés étrangères) dans cette base de données.

Exercice 2.2 : Écrire une requête permettant d'obtenir la liste des dates d'examens (on peut admettre qu'elles sont toutes différentes).

Exercice 2.3 : Écrire une requête permettant d'obtenir la liste des matières pour lesquelles il y a eu au moins une note égale à `20`.

Exercice 2.4 : Écrire une requête permettant d'obtenir la liste des matières pour lesquelles au moins un étudiant de `PCSI1` a eu un examen.

Exercice 2.5 : Écrire une requête permettant d'obtenir le plus grand nombre de prénoms différents dans une même classe.

Exercice 2.6 : Écrire une requête permettant d'obtenir le prénom ayant le plus grand nombre d'occurrences. En cas d'égalité, on acceptera de ne garder qu'un résultat.

Exercice 2.7 : Écrire une requête permettant d'obtenir la classe ayant le plus grand nombre d'étudiants prénommés `Théo`. En cas d'égalité, on acceptera de ne garder qu'un résultat.

Exercice 2.8 : Écrire une requête permettant d'obtenir le prénom ayant la plus grande moyenne à l'examen numéro `1` (la moyenne se calculant parmi les porteurs d'un même prénom). En cas d'égalité, on acceptera de ne garder qu'un résultat.

Exercice 2.9 : Écrire une requête permettant d'obtenir les cinq plus mauvaises notes à l'examen de maths ayant eu lieu à la rentrée (disons le 9 septembre, la représentation de la date étant libre dans la limite du raisonnable).

Exercice 2.10 : Écrire une requête permettant d'obtenir l'histogramme des notes : pour chaque note, combien de personnes l'ont eue sur l'ensemble des étudiants et des examens.

Exercice 2.11 : Supposons qu'il existe trois variables globales `ETUDIANTS`, `EXAMENS` et `NOTES` en Python, dont chaque élément est une liste donnée dans l'ordre du schéma des tables éponymes. Écrire une fonction sans argument renvoyant le résultat correspondant à l'exercice 2.2 sous une forme adéquate (liste de chaînes de caractères).

Exercice 2.12 : Idem en reprenant l'exercice 2.5 (on pourra renvoyer un entier).

Exercice 2.13 : Idem en reprenant l'exercice 2.7 (on pourra renvoyer une chaîne de caractères).

Exercice 2.14 : Idem en reprenant l'exercice 2.10 (on pourra renvoyer une liste de couples formés de la note et du nombre d'occurrences). Il est permis de supposer que les notes sont nécessairement des entiers de `0` à `20`.

Partie 3 : Base de données « correction »

Depuis des années, ma correction se fait sur un tableur. Or, comme dit en cours, une feuille de tableur matérialise bien une table dans une base de données. Il s'agit donc de simuler en mieux ce tableur à l'aide de la structure suivante :

- Table `ETUD`, dont les attributs sont `id_etudiant` (entier), `nom` (chaîne de caractères), `prenom` (chaîne de caractères) et `classe` (chaîne de caractères).
- Table `DS`, dont les attributs sont `id_ds` (entier), `date` (chaîne de caractères) ; `titre` (chaîne de caractères) et `coeff` (entier).
- Table `QUESTION`, dont les attributs sont `id_ds` (entier), `code_question` (chaîne de caractères), `nom_partie` (chaîne de caractères) et `points` (flottant).
- Table `NOTE`, dont les attributs sont `id_ds` (entier), `id_etudiant` (entier), `code_question` (chaîne de caractères), et `pourcentage` (flottant).

Les deux premières tables sont intuitives.

En ce qui concerne la troisième, le code de chaque question sera une valeur du type `"Question 1.1"`, `"Exercice 4"`, etc. et pourra être utilisé dans différents DS, bien qu'il soit unique au sein d'un même DS. Le nom de la partie (par exemple `"Cours"`, `"Exercice 1"`...) permet de rassembler différents codes de question au sein d'un DS et sera lui aussi potentiellement réutilisé dans un autre DS. L'attribut `points` correspond, on s'en doutait, au nombre de points alloué à la question dans le barème prévu.

Dans la table `NOTE`, la copie d'un étudiant à un devoir est évaluée. Pour chaque question traitée, le pourcentage du total de points est précisé. Ainsi, si la question de code correspondant au DS correspondant a son attribut `points` à `5.0` et l'attribut `pourcentage` vaut `0.6` pour l'étudiant considéré, cela donne trois points en vue du DS. **On distinguera une question non traitée, ne donnant pas lieu à un enregistrement dans cette table, et une question si mal traitée que le pourcentage est nul.**

Exercice 3.1 : Écrire une requête permettant d'obtenir le plus grand nombre de points alloués à une question du DS d'identifiant `1`.

Exercice 3.2 : Écrire une requête permettant d'obtenir le nombre de questions totalement réussies par l'étudiant d'identifiant `1` sur l'ensemble des DS.

Attention, cela devient difficile à présent, mieux vaut avoir fait au moins une fois des sous-requêtes et des calculs sur des attributs, typiquement...

Exercice 3.3 : Écrire une requête permettant d'obtenir le nombre de questions qu'au moins un étudiant a traitées dans le DS d'identifiant `1`. Écrire une requête permettant d'obtenir (éventuellement en se servant de ce qui précède) le nombre de questions que personne n'a abordées dans le DS d'identifiant `1`.

Exercice 3.4 : Écrire une requête permettant d'obtenir le plus grand nombre de points marqués par un étudiant sur une question, tous DS, tous étudiants et toutes questions confondues.

Exercice 3.5 : Écrire une requête permettant d'obtenir la note de l'étudiant d'identifiant `1` au DS d'identifiant `1`.

Exercice 3.6 : Écrire une requête permettant d'obtenir le nom de la partie dans laquelle l'étudiant d'identifiant `1` a marqué le plus de points au DS d'identifiant `1`. En cas d'égalité, on acceptera de ne garder qu'un résultat.

Exercice 3.7 : Écrire une requête permettant d'obtenir le classement des étudiants (nom et prénom suffisent) au DS d'identifiant `1`.

Exercice 3.8 : Écrire une requête permettant d'obtenir le classement **par efficacité décroissante** des étudiants (nom et prénom suffisent) au DS d'identifiant `1`.

L'efficacité d'un étudiant se calcule en divisant le nombre de points effectivement obtenus par le nombre de points que les questions effectivement traitées pouvaient permettre d'obtenir.