Concours Blanc : épreuve d'Informatique

PTSI A et B Lycée Eiffel

28 mai 2015

Durée : 2H.

Ce sujet s'intéresse à différents aspects de la gestion d'une bibliothèque. Il est contitué de quatre parties indépendantes, qui peuvent être traitées dans l'ordre souhaité par le candidat. Toutes les fonctions que l'énoncé demande de programmer en précisant leur comportement peuvent être réutilisées dans les questions ultérieures, même si le candidat n'a pas réussi à programmer la fonction lui-même.

Première partie : Code ISBN d'un ouvrage.

Le code **ISBN** (**I**nternational **S**tandard **B**ook **N**umber) est un code international permettant de désigner de façon unique tous les livres édités. Il en existe deux versions : l'ISBN13 est constitué de 13 chiffres, tandis que l'ISBN10 (plus ancien) est contitué de 10 caractères, les neuf premiers étant des chiffres mais le dixième étant soit un chiffre, soit la lettre X. Les neuf premiers chiffres du code ISBN10 d'un ouvrage sont des identifiants de pays, d'éditeur, et d'ouvrage, alors que le dixième caractère est une clé de vérification obtenue à partir des neuf premiers : si on note c_1, c_2, ..., c_9 les neuf premiers caractères du code, on calcule $n = \sum_{i=1}^{9} i \times c_i$, et le caractère c_{10} est le reste de la division euclidienne de n par 11. Dans le cas où ce reste est égal à 10, le caractère c_{10} sera la lettre X. Le code ISBN13 sera décrit en détail un peu plus loin dans l'énoncé. On rappelle que le reste de la division euclidienne de n par p peut s'obtenir en Python grâce à l'opération $n\%p$.

1. Combien d'octets sont nécessaires pour stocker en mémoire un code ISBN10 (on détaillera la réponse) ?
2. Les neuf premiers chiffres d'un code ISBN10 sont les suivants : 226611156. Déterminer le dixième caractère de ce code.
3. Écrire une fonction **listechiffres** qui prend comme argument un nombre entier n, et ressort une liste constituée des chiffres de l'écriture décimale de n (dans l'ordre de lecture).

4. En déduire une fonction **clécode** qui prend comme argument un nombre de neuf chiffres, et qui calcule la clé de vérification correspondant à ces neuf chiffres. Écrire ensuite une fonction **completecode** qui prend toujours comme argument un nombre de neuf chiffres, et qui ressort le code ISBN10 à 10 caractères obtenu en ajoutant à ce nombre la clé de vérification correspondante.
5. Écrire une fonction **verifiecode** qui prend comme argument une chaine de dix caractères, et qui détermine s'il s'agit d'un code ISBN10 valide ou non (on supposera que les dix caractères sont tous des chiffres, sauf le dernier qui pourra être un X).
6. Le code ISBN13 d'un ouvrage est obtenu en ajoutant devant les neuf premiers chiffres du code ISBN10 les trois chiffres 978, et le treizième chiffre est une clé de vérification obtenue à partir des douze premiers de la façon suivante : en notant $n = \sum_{i=0}^{5} c_{2i+1} + \sum_{i=1}^{6} c_{2i}$, et $f(n)$ le dernier chiffre du nombre n, alors $c_{13} = f(10 - f(n))$.
7. Un code ISBN10 est erroné, mais on sait que l'erreur est due à la mauvaise transcription de l'un des neuf premiers chiffres, qui a été augmenté accidentellement d'une unité. Expliquer comment corriger cette erreur à l'aide de la clé de vérification. Pourrait-on corriger une erreur sans savoir que le chiffre incorrect a été augmenté d'une seule unité ? Et sans savoir qu'il n'y a qu'un seul chiffre incorrect sur les neuf premiers ?
   (a) Calculer le code ISBN13 correspondant au code ISBN10 dont les neuf premiers chiffres sont 226611156.
   (b) Écrire un programme **conversioncode** qui prend comme un argument un code ISBN10 à dix caractères, et qui calcule le code ISBN13 correspondant.
   (c) Écrire une fonction **livrejaponais** qui prend comme argument un code ISBN13 et qui détermine s'il s'agit d'un livre édité au Japon (tous les livres édites au Japon ont un ISBN10 dont le premier chiffre est un 4).

Deuxième partie : Utilisation d'un tri par table de hachage.

On souhaite trier tous les livres de la bibliothèque en utilisant la méthode suivante : on associe à chaque titre un code (pas nécessairement unique) et on crée une liste contenant autant d'indices que le nombre de codes possibles. On insère ensuite chaque titre dans la case de la liste correspondant à son code. Chaque case de la liste peut ainsi contenir plusieurs titres, et sera donc elle-même une liste. Un exemple concret mais peu performant : on prend comme code le rang alphabétique de la première lettre du titre. La première case de notre liste contiendra alors tous les titres commençant par la lettre 'A', la deuxième tous ceux qui commencent par 'B' etc.

1. Rappeler ce que représente le Standard **ASCII**.
2. La commande Python **ord** permet d'obtenir le code ASCII d'un caractère. Les caractères minuscules de l'alphabet latin correspondent aux codes ASCII compris entre 97 (pour le a) et 122 (pour le z). La commande **chr** effectue la conversion réciproque : par exemple, chr(99) renvoit le caractère 'c'. Écrire une commande Python qui prend comme argument un caractère alphabétique en minuscule ou en majuscule, et qui renvoit son rang alphabétique.
3. On code les titres de la façon suivante : on ne considère que les deux premiers caractères du titre, on note r et s le rang alphabétique de ces deux caractères, et on calcule $26 * (r - 1) + s$. Si l'un des deux caractères n'est pas alphabétique (code ASCII qui n'est pas compris entre 97 et 122), on associe au titre le code 0. Donner le code obtenu pour le livre « Python pour les Nuls ».
4. Écrire une fonction Python **code** prenant comme argument une chaine de caractères et calculant le code du titre correspondant.

5. Écrire une fonction Python **trihachage** prenant comme argument une liste de titres (chaînes de caractères) et créant la liste des titres triés suivant le code précédent (comme décrit en préambule à cette deuxième partie).
6. Estimer la complexité de cette procédure de tri. Comparer aux tris classiques, qui demandent un nombre d'opérations proportionnel à $n\ln(n)$, où n est le nombre d'objets à trier. Quels inconvénients possède cette procédure de tri ?
7. On dispose de la liste triée des ouvrages de la bibliothèque, et on souhaite retrouver dans cette liste un livre dont on connait le titre. Comment procèdera-t-on (on ne demande pas de programme Python pour cette question) ?

Troisième partie : Exploitation du fichier de prêts.

La liste des livres empruntés par des lecteurs de la bibliothèque est stockée dans un fichier texte, nommé **prets.txt**, qui contient les données sous la forme suivante : chaque ligne concerne un emprunt, et contient successivement le code ISBN10 du livre emprunté, le nom de la personne ayant emprunté le livre, son prénom, la date d'emprunt, et la date de retour prévue. Les deux dates sont au format aaaammjj (sans caractère de séparation, par exemple 20150528 pour la date d'aujourd'hui), et on suppose que les différentes données sont séparées à chaque fois par une virgule. Ainsi, la première ligne du fichier pourra ressembler à ceci :

0553103547,Lafon,Guillaume,20150428,20150728

On rappelle que la méthode **split**, appliquée à une chaine de caractères avec comme argument un caractère donné, permet de créer une liste de sous-listes de la liste initiale, en utilisant le caractère spécifié comme séparateur pour la création de ces sous-listes. Ainsi, la commande :

split('abracadabra','a')

renverra la liste ['br','c','d','br'] (on a introduit une césure pour chaque apparition du caractère 'a', et accessoirement supprimé les a).

1. Rappeler quels sont les différents modes d'ouverture d'un fichier texte en Python, et pour quel type de traitement ils sont adaptés.
2. Écrire un programme Python permettant d'accéder au fichier **prets.txt**, et créant une liste **emprunteurs** contenant les noms de toutes les personnes ayant emprunté (au moins) un livre à la bibliothèque (on ne se préocuppera pas du fait qu'un même emprunteur puisse apparaitre à plusieurs reprises).
3. Modifier le programme précédent pour qu'il crée une liste constituée uniquement des personnes ayant un livre à rendre en retard (autrement dit, ayant un emprunt dont la date de retour est antérieure au 28 mai 2015).
4. Créer une liste contenant tous les noms d'emprunteurs ainsi que le nombre de livres empruntés par chacun d'entre eux (on pourra créer une liste simple contenant alternativement des chaines de caractères et des entiers ; on rappelle que la méthode **index** permet de trouver l'emplacement d'un élément dans une liste, et que la méthode **count** permet de savoir le nombre d'occurrences d'un élément dans une liste).

Quatrième partie : Base de données.

On souhaite constituer une base de données pour notre bibliothèque, qui devra contenir les relations suivantes :

- une relation **livre**, avec les attributs n°livre, titre, auteur
- une relation **lecteurs**, avec les attributs n°lecteur, nom, prénom, datedenaissance
- une relation **emprunt**, permettant de relier les livres et les lecteurs, avec comme attributs supplémentaires la date d'emprunt, et la date de retour.

1. Que faut-il choisir comme clé primaire de la table **emprunt** ?
2. Écrire la commande SQL permettant de créer la table **lecteurs** (on donnera des types cohérents pour les différentes données).
3. Écrire une commande de l'algèbre relationnelle permettant d'obtenir la liste des livres empruntés par le lecteur dont le nom est **Knuth**. Traduire cette requête en SQL.
4. Écrire une commande en SQL permettant d'ajouter à la table une livre une colonne **etat**, permettant d'indiquer l'état du livre (qui sera une chaine de caractères).
5. Écrire une commande en SQL permettant d'afficher la liste des livres écrits par un auteur dont le nom commence par un G et empruntés par un lecteur dont le nom commence par un L.