Langages de scripts (LS4)

Examen de 1ère session – vendredi 4 mai 2007

Attention :

- ***Une feuille de réponses est jointe à cet énoncé, pour l'exercice 3. Vous devez l'insérer dans votre copie et inscrire le code marqué dessus sur toutes vos copies.***
- ***Seuls les exercices 4 et 5 sont à faire par les étudiants suivant la matière en module ouvert*[1]*.***
- ***Le seul document autorisé est une feuille au format A4 (qui devra rester* personnelle *!).***
- ***L'utilisation de téléphones ou ordinateurs portables est interdite.***

Exercice 1 – *grep*

Cet exercice n'est pas à faire par les étudiants suivant la matière en module ouvert.

Donnez ce qu'affiche la dernière ligne sur la sortie standard.

```
/tmp % cat liste.txt
foobar
foobr
---foobar
foobbr
foor
bar
fbar
fbaz
/tmp % grep '^f[o]*b\+a\?r' liste.txt
```

Exercice 2 – *processus*

Cet exercice n'est pas à faire par les étudiants suivant la matière en module ouvert.

Ecrire un script renvoyant le nombre de processus en cours d'exécution et appartenant à l'utilisateur courant. Ce script affichera de plus un message d'avertissement si le nombre de processus est supérieur à 10. On rappelle que :

- la variable d'environnement `USER` désigne le nom de login de l'utilisateur courant,
- `wc -l` permet de compter le nombre de lignes écrites sur la sortie standard,
- `ps aux` affiche l'ensemble des processus exécutés.

[1] Si vous avez un doute, c'est que vous ne suivez pas la matière en module ouvert.

Exercice 3 – *Mutation de listes*

Cet exercice n'est pas à faire par les étudiants suivant la matière en module ouvert.

Sur la feuille de réponses jointe à l'énoncé, indiquez les valeurs des listes obtenues.

Exercice 4 – *`split` étendu*

Ecrire une fonction `mysplit` qui prend en argument deux chaînes de caractères et renvoie une liste des facteurs de la première séparés par les caractères de la deuxième.

exemple : `mysplit('abracadabra', 'a')` → `['', 'br', 'c', 'd', 'br', '']`
`mysplit('abracadabra', 'ab')` → `['', '', 'r', 'c', 'd', '', 'r', '']`
`mysplit('abracadabra', 'bc')` → `['a', 'ra', 'ada', 'ra']`

On rappelle que la méthode `split` des chaînes de caractères peut prendre un argument.

Exercice 5 – *Gestion d'un annuaire*

Note : Les questions de cet exercice sont indépendantes ! Rien ne vous interdit de sauter des questions en supposant que les fonctions précédentes sont bien implémentées.

On souhaite écrire un module contenant des outils pour manipuler des annuaires. Ces annuaires seront stockés dans des fichiers textes au format CSV (Comma Separated Values) : chaque ligne représentera une entrée de l'annuaire, avec 5 champs séparés par des virgules, représentant les 5 caractéristiques du personnage concerné : nom, prénom, profession, adresse et numéro de téléphone. Ainsi, par exemple :

Haddock,Archibald,Capitaine,Chateau de Moulinsart,421

Ce module devra définir deux classes, `Annuaire` et `Personnage`. Chaque instance de la classe `Annuaire` contiendra une liste d'instances de la classe `Personnage`, qui elles-mêmes auront pour attributs les 5 caractéristiques précédemment citées.

1. Écrire un constructeur pour la classe `Personnage`, prenant en paramètre une chaîne de caractères au format CSV.
2. On rappelle qu'il est possible de définir deux méthodes spéciales pour la représentation d'un objet par une chaîne de caractères, `__str__` et `__repr__`. Lorsque ces deux méthodes sont définies, `print` et `str` utilisent `__str__` tandis que `repr` utilise `__repr__`. Définir ces deux méthodes dans la classe `Personnage` pour permettre soit un affichage agréable pour l'utilisateur, soit la repésentation au format CSV.
3. Écrire un constructeur pour la classe `Annuaire`, prenant en paramètre le nom d'un fichier dont chaque ligne contient un personnage décrit au format CSV.
4. Écrire une méthode `copie` dans la classe `Annuaire` prenant en paramètre un nom de fichier et y copiant les représentations des entrées de l'annuaire au format CSV, triées par ordre alphabétique. Pour cela, on pourra créer un dictionnaire et en parcourir la liste des clés triée.
5. Écrire un script `tri_annuaire.py` (appelé en ligne de commande) qui prend en argument le nom d'un fichier d'annuaire, en trie les entrées alphabétiquement selon le nom de famille, et écrit le résultat dans le fichier d'origine.

6. Écrire une méthode `liste_pages_jaunes` dans la classe `Annuaire` prenant en paramètre une chaîne de caractères et renvoyant la liste des entrées de l'annuaire ayant la profession correspondante. Écrire une méthode `cherche_pages_jaunes` affichant le nombre et la liste des réponses obtenues.
7. Écrire une méthode `liste_pages_blanches` dans la classe `Annuaire` prenant en paramètre une chaîne de caractères et retournant la liste des personnes dont le nom ou le prénom contient cette chaîne. Écrire la méthode `cherche_pages_blanches` correspondante.
8. Écrire une méthode `modifier` dans la classe `Annuaire` prenant un nom en paramètre et demandant à l'utilisateur d'entrer les nouvelles coordonnées de la personne correspondante. Pour cela, on pourra écrire une méthode `selectionne` qui vérifie que le nom apparaît dans l'annuaire et, s'il apparaît plusieurs fois, demande à l'utilisateur de choisir parmi les personnes correspondantes.
9. Ajouter des méthodes `inserer` et `supprimer` dans la classe `Annuaire`.