HAI103I
Exploration de dossiers, analyse de commandes
-
Pierre Pompidor

Exploration d'une arborescence de dossiers :

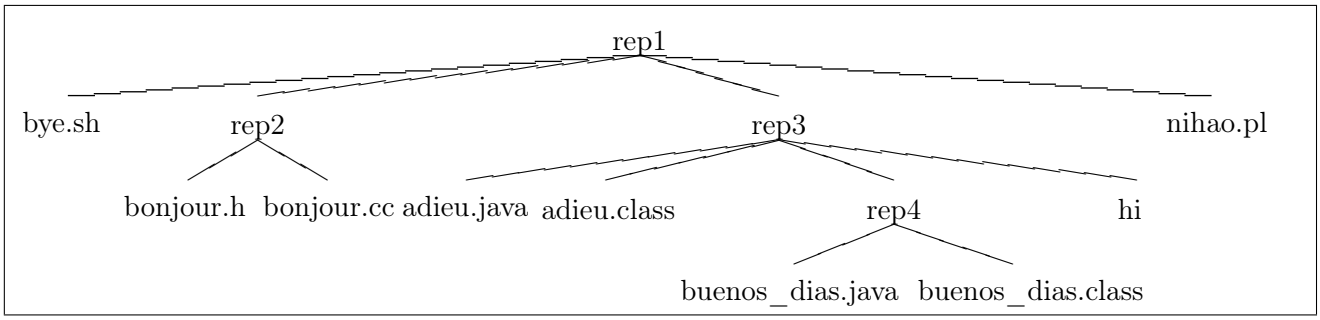
Ecrivez un script nommé **indexation.py**. qui lancé avec comme paramètre un nom de répertoire, indexe les fichiers qui se trouvent dans l'arborescence définie par ce répertoire suivant leurs suffixes.
Ce script comportera une fonction **récursive** (càd qui s'appelle elle-même) qui sera appelée sur chaque répertoire de l'arborescence. Je vous rappelle la structure générale du programme :

```
import os, sys
...
def parcours (rep) :
    print "Je suis dans "+rep
    liste = os.listdir(rep)
    for fichier in liste :
        if os.path.isdir(...) :
            ...
        else :
            ...

parcours (sys.argv[1])  # Appel initial de la fonction récursive
```

Exemple :

Soient les répertoires suivants :



l'appel suivant : **python3 indexation.py rep1** va afficher :

```
.sh : ['rep1/bye.sh']
.h : ['rep1/rep2/bonjour.h']
.cc : ['rep1/rep2/bonjour.cc']
.java : ['rep1/rep3/adieu.java', 'rep1/rep3/rep4/buenos_dias.java']
.class : ['rep1/rep3/adieu.class', 'rep1/rep3/rep4/buenos_dias.class']
.pl : ['rep1/nihao.pl']
pas de suffixe : [''rep1/rep3/hi']
TOTAL : 9
```

Complétez ce programme pour que l'utilisateur, tant qu'il le souhaite, puisse saisir une chaîne de caractères et un nombre arbitraire d'extensions et que le programme puisse afficher les lignes des fichiers (qui ayant une de ces extensions) contiennent la chaîne de caractères.

Bonus : Listing synthétique des jours de connexion de tous les utilisateurs

En utilisant la commande LAST écrire un script python qui liste les jours de connexion des utilisateurs qui ont fréquenté cette machine (en sachant que vous désirez surveiller les (mauvaises) fréquentations de celle-ci).

Imaginons que le résultat de la commande LAST soit :

```
pompidor pts/0        :0               Wed Oct 25 13:00 - still logged in
pompidor pts/0        193.52.27.135    Tue Oct 24 08:19 - down   (02:10)
meynard  :0                            Tue Oct 24 08:18 - 10:19  (02:01)
pompidor pts/2        :0               Mon Oct 23 12:27 - 13:09  (00:42)
pompidor pts/1        :0               Mon Oct 23 11:55 - 13:30  (01:35)
pompidor pts/0        :0               Mon Oct 23 11:49 - down   (01:56)
meynard  pts/0        :0               Mon Oct 23 11:37 - 11:39  (00:02)
pompidor pts/1        :0               Mon Oct 23 11:07 - down   (00:07)
```

Remarques:

- *pts* ou *tty* indique la création d'un terminal (exemple : **pts/1** ouverture d'un **second** terminal)
- un numéro IP à la place de *:0* indique une connexion à partir d'une machine distante

Le résultat de l'appel **python3 connexions.py** serait :

```
pompidor s'est connecté les : 25 Oct (1 fois), 24 Oct (1 fois), 23 Oct (4 fois)
meynard s'est connecté les : 24 Oct (1 fois), 23 Oct (1 fois)
```

Réalisez ce programme en quatre étapes (qui se complèteront progressivement) :

- Pour information, exécutez dans le script la commande *last* et affichez son résultat dans le terminal
- Récupérez maintenant le résultat de cette commande dans votre script. Mettez en place **progressivement** l'expression régulière qui extrait :
  - le login
  - le nom du mois et le numéro du jour dans le mois (la date)
  - le nombre d'heures
  - le nombre de minutes

  et affichez ces informations
- Affichez :
  - le nombre de connexions par login (via un dictionnaire)
  - le nombre de connexions et le temps de connexions (en minutes) par login (via un dictionnaire de listes)
  - puis par login et par date (via un dictionnaires de dictionnaires de listes)
- Suivant des paramètres indiquant un nombre maximal de connexions et/ou un temps de connexion cumulé maximal, affichez des alertes !