TP 12 - Python avancé

Exercice 1. Exceptions

Dans cet exercice, on va écrire un programme permettant de chercher les occurrences d'une expression régulière dans un fichier.

1. Écrire une fonction `afficheRegexp(chaine,regexp)` qui prend une chaîne `chaine` et une expression régulière `regexp` et affiche tous les "morceaux" de `str` correspondants. Pour cela, vous utiliserez la fonction `findall` du module `re` de Python.
   Attention, si l'expression est mal formée vous devez afficher le message suivant et renvoyer l'exception provoquée : `"L'expression reguliere ... est incorrecte"`
   Par exemple, l'appel :

   ```
   afficheRegexp("(totot-1)*(2+3)","\([^)]*\)")
   ```

   donne : `(totot-1)` et `(2+3)` et l'appel `afficheRegexp("((()()))","\())")` donne :

   ```
   L'expression reguliere \()) est incorrecte
   Traceback (most recent call last):
   ...
   sre_constants.error: unbalanced parenthesis
   ```

2. Faire en sorte de ne pas afficher la partie `"Traceback"` pour l'appel précédent.
3. Écrire une fonction `essaieOuvrir()` qui demande à l'utilisateur un nom de fichier et essaie de l'ouvrir. S'il l'ouverture ne pose pas de problème, la fonction renvoie le résultat de `open`. Malheureusement, il peut y avoir des problèmes lors de l'ouverture... Nous allons en traiter 2 :
   (a) Si le fichier n'existe pas, la fonction redemande un nom de fichier à l'utilisateur.
   (b) Si le fichier n'a pas les bonnes permissions, on renvoie une `NameError("Permission")`

   Dans la doc, vous trouverez comment distinguer ces 2 cas en utilisant la variable `errno`.
4. Nous pouvons maintenant écrire le programme principal. Utiliser `essaieOuvrir()` pour demander un nom de fichier à l'utilisateur. En cas de problème de permissions, le programme s'arrête en disant : "Changez les permissions de votre fichier !". Si le fichier n'existe pas le programme redemande un nom de fichier.
   Ensuite, le programme demande une expression régulière à l'utilisateur et utilise la fonction `afficheRegexp` pour la chercher dans le fichier. Si elle est mal formée, le programme s'arrête en disant : "L'expression reguliere ... est incorrecte".

Exercice 2. En une ligne...

1. ... afficher l'alphabet : ['a', 'b', 'c', ...].
2. ... afficher la liste des triplets croissants d'entiers < 10 : [(1, 2, 3), (1, 2, 4), ..., (1, 3, 4), ..., (2, 3, 5), ..., (7, 8, 9)].
3. ... afficher la liste [('a', 0), ('b', 1), ('a', 2), ('b', 3), ..., ('a', 18), ('b', 19)].
4. ... afficher **l'ensemble** des nombres premiers inférieurs à 100.
5. ... afficher **la liste** du 10e au 20e nombre premier :[31, 37, 41, 43, 47, 53, 59, 61, 67, 71].

Exercice 3. Modules pour les dictionnaires français et anglais

Dans cet exercice, vous allez manipuler des modules pour gérer des dictionnaires dans des langues différentes. Les modules sont simplement des fichiers indépendants auxquels vous pouvez accéder par la commande `import`.

1. Dans un module nommé `fonctionsDico.py`, écrire les 2 fonctions suivantes :
   (a) la fonction `fabriqueDicoSet(fichier)` qui prend un fichier de mots (1 par ligne) et renvoie l'ensemble de tous les mots de ce fichier.
   (b) la fonction `prefixeDico(mot, dico)` qui renvoie `True` si le mot a un préfixe dans le dictionnaire et `False sinon`. Par exemple, en français, "sympa" n'a pas de préfixe dans le dictionnaire alors que "ballon" en a ("bal").
2. Vous pouvez ensuite récupérer les modules `dicoFR.py` et `dicoAN.py` et les fichiers de dictionnaire associés sur la page web. Ces modules permettent de créer des dictionnaires en français et en anglais.
   En utilisant les dictionnaires (`dico`) qui se trouvent dans ces modules, créer l'ensemble de tous les mots communs de l'anglais et du français (il y en a environ 12 000). Est-il possible d'appeler cet ensemble `dico` ?
3. Afficher la liste des 100 premiers mots communs dans l'ordre alphabétique : ['a', 'abandon', 'abattoir', 'abbatial', ...]
4. Afficher la liste de mots qui commencent par un 'x' et finissent par un 'e' dans le dictionnaire français : ['xyste', 'xerographie', 'xylocope', ...]
5. En une ligne, calculer le pourcentage de mots qui n'ont pas de préfixes dans le dictionnaire anglais (et comparer avec le français).
   Quel est le problème ? Corriger les modules fournis pour que cela fonctionne.

Exercice 4. Parcours

Un `plateau` est un tableau à deux dimensions qui contient des Booléens. Si `plateau[i][j]` vaut `True`, il y a un mur et sinon la case est libre.

```
1  plateau = [[True, False, False, False],
2             [False, True, True, False]]
```

Le but de l'exercice est d'écrire une fonction `chemin` prenant un plateau et les coordonnées de deux cases `deb` et `fin` et renvoie `True` si l'on peut aller de la case `deb` à la case `fin` en se déplacant horizontalement et verticalement.

Dans notre exemple, `chemin(plateau,(1,3),(1,0))` renverra `False` et `chemin(plateau,(1,3),(0,1))` renverra `True`.

1. Ecrire une fonction `voisinsCase` qui prend un plateau et une case et renvoie l'ensemble de ces voisins immédiats horizontaux ou verticaux qui sont sur le plateau et qui sont libres.
2. Ecrire une fonction `voisinsCases` qui prend un plateau et un ensemble de cases et renvoie l'ensemble de tous les voisins de ces cases.
3. Ecrire une fonction `accessibles` qui prend un plateau et une case et renvoie l'ensemble des cases que l'on peut atteindre depuis cette case en se déplacant horizontalement et verticalement.
4. Ecrire la fonction `chemin`.