TDs Python

1/ Exercices basiques pour le cours Python

1.1/ Écrire un programme qui demande la saisie de deux entiers, les multiplie et affiche le résultat
Rq : pour convertir une chaine de caractère saisie au clavier en nombre entier on utilisera la fonction int() . Voir dans la documentation la définition de cette fonction : dans Library Reference - The Python Standard Library - Built-in Functions

1.2/ Idem mais on testera que les chaines saisies sont vraiment convertibles en entiers et le cas contraire on terminera le programme en indiquant un message d'erreur explicite.
Rq : on utilisera la fonction str.isdigit() disponible pour les chaines de caractères. Voir dans la documentation dans Library Reference - The Python Standard Library - Built-in Types

1.3/ Idem à 2/ mais on saisira deux nombres flottants (comme 1.5)

1.4/ Demander la saisie d'un nombre entier et afficher si le nombre est pair ou impair.

1.5/ Écrire un programme qui demande la saisie de 3 entiers et affiche la plus grand d'entre-eux

1.6/ Idem 1.5 mais avec un nombre quelconque d'entiers saisis au clavier.

1.7/ Saisir un nombre quelconque de nombres flottants, calculer et afficher leur écart-type.

2/ Exercices sur les chaines de caractères

2.1/ Saisir une phrases entières de plusieurs mots déparés par des blancs et afficher ses mots un par un à l'écran, en allant à la ligne à chaque mot affiché (algorithme sans utiliser de fonction, puis en utilisant des fonctions).

2.2/ Saisir une phrase puis un mot et faire dire au programme si le mot est contenu dans la phrase (d'abord sans utiliser de fonction, puis en utilisant des fonctions). On indiquera aussi la position du premier caractère du mot le cas échéant.

2.3/ Saisir une chaine, l'afficher et afficher la chaine inverse

3/ Exercices utilisant des collections

2.1/ Écrire un programme qui demande la saisie d'un nombre quelconque d'entiers, qui mémorise ces nombres dans une liste et qui affiche le plus grand entier contenu dans la liste.

2.2/ Saisir des entiers, les mémoriser dans une liste, afficher la liste, la classer, réafficher

4/ Exercice sur les fonctions

4.1/ Ecrire la fonction factorielle

4.2/ Ecrire la fonction factorielle de manière récursive

5/ Exercice sur les fichiers

5.1/ Grep : recherche d'un mot dans un fichier texte. Pour chaque occurrence du mot on affichera le numéro de ligne dans le fichier et la position du mot dans la ligne.

5,2/ Grep suite. A partir de 5,1/ créer un programme qui recherche, à partir d'une racine de répertoire, l'ensemble des occurrences d'un mot dans l'ensemble des fichiers contenus dans les osu-répertoires. On écrira un parcours récursif de la hiérarchie de sous répertoires (voir 6,3)

5,3/ Grep suite. On perfectionnera encore le programme grep précédent pour qu'il soit capable de rechercher non plus un simple mot dans des fichiers, mais une **expression régulière.**

5,4/ Lire un fichier texte et écrire un fichier texte où les mots du premier fichier sont extraits avec pour chaque mot sa position (numéro du mot dans le fichier ou offset).
Puis on peut regrouper dans le second fichier par mots en indiquant le nombre d'occurrence sur un entier et la liste des offset.
Cela consiste en fait à créer un fichier d'index automatique des mots d'un texte.

6/ Algorithmes

6.1/ Calculer la racine carrée d'un nombre en utilisant la méthode inventée par les babyloniens.
On poursuivra l'algorithme jusqu'à obtenir un résultat au dix-millième.

6.2/ Utilisation de l'algorithme de codage de messages de Jules César
Écrire le programme qui permet de saisir une phrase au clavier et de la coder. On affichera à l'écran la phrase codée.
Écrire le programme qui permet de lire un fichier codé par l'algorithme de César et d'écrire le message décodé dans un autre fichier.
Écrire le programme qui permet de décoder un message. On appliquera l'algorithme pour décoder le contenu d'un fichier.

6.3/ Parcourir un répertoire racine et sa hiérarchie de sous-répertoires en affichant des statistiques sur les fichiers / répertoires. (Ce programme pourra être écrit avantageusement avant 5.2, dont il pourra servir de base pour le parcours récursif du programme Grep)

6.4/ Suite 6,3. Écrire un parcours récursif d'une hiérarchie de répertoire à partir d'un répertoire racine. Sur chaque fichier on effectuera un certain traitement. Par exemple on mémorisera les noms des fichiers dans une liste triées par date et une autre liste trié par taille.