Marc Jachym, Lurpa, ENS de Cachan

TDs Python

Exercices basiques pour le cours Python

1/ Travail sur les types de valeurs

1.1/ Placez deux chaines de caractères, un prénom et un nom, dans deux variables puis trois valeurs de type entier jour, mois, année dans trois variables et affichez une phrase comme :
Je m'appelle Steve McQueen et je suis né le 24/03/1930

1.2/ Affectez les variables temps et distance par des valeurs de type flottant et calculez et affichez la valeur de la vitesse. Affichez en imposant deux chiffres après le point décimal.

2/ Les Entrée / Sortie simples (lecture au clavier et écriture formatée à l'écran)

2.1/ 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

2.2/ Idem mais on testera que les chaines saisies sont vraiment convertibles en entier et le cas contraire on terminera le programme en indiquant un message d'erreur explicite, ou mieux on redemandera la saisie à l'utilisateur tant que celle-ci n'est pas convertible.

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

2.3/ Idem à 2.2/ mais on saisira deux nombres flottants.

Rq : le contrôle des chaines saisies, pour savoir si elles sont convertibles en flottant, est ici plus complexe car il n'y a pas de fonction toute faite comme isdigit(). On utilisera directement la fonction de conversion float() avec le mécanisme des exceptions (voir aussi dans le cours, partie 2, II.2.11) qui permet d'intercepter les erreurs. Cela permet de redemander la saisie tant qu'il y a erreur.

3/ Manipulations simples des structures if, for et while

3.1/ Demandez la aisissez un flottant. S'il est positif ou nul, affichez sa racine, sinon affichez un message d'erreur.

Rq : le module math de la librairie standard de Python contient la fonction sqrt() qui renvoie la racine carrée d'une valeur donnée en argument. Pour l'utiliser, omportez d'abord de module par
import math
puis l'appel se fait par
math.sqrt()

3.2/ Initialisez deux entiers : a à 0 et b à 10.
Écrivez une boucle affichant et incrémentant la valeur de a tant qu'elle reste inférieure à celle de b.
Écrivez une seconde boucle décrémentant la valeur de b et affichant sa valeur si elle est impaire. Bouclez tant que b n'est pas nul.

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

3,4/ Idem 3,3 mais avec un nombre quelconque d'entiers saisis au clavier.

4/ Exercices sur les chaines de caractères

4,1/ Saisir une chaine de caractères , l'afficher et afficher la chaine inverse

4.2/ 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 les fonctions de l'objet string, puis en utilisant des fonctions).

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

4.4/ Utilisation de l'algorithme de codage cod13 (simplissime, on ne peut pas se baser dessus à des fins de sécurité !)
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.

4,5/ Grâce au module des expressions régulières, écrivez une commande qui prend en entrée une chaine de caractère (comme argument sur la ligne de commande) et indique si cette chaine représente une adresse email.

5/ Exercices d'utilisation des collections

5.1/ 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.

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

6/ Exercice sur les fonctions

6.1/ Écrire la fonction factorielle

6.2/ Écrire la fonction factorielle de manière récursive

7/ Algorithmes

7.1/ 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.

7,2/ Caculd du pgcd de 2 nombres suivant la méthode d'Euclide

7.3/ Faites **s**aisir un nombre quelconque de nombres flottants, qui seront mémorisés dans une liste, puis calculez et affichez leur écart-type.

7.4/ 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)

7.5/ Suite 7.4. 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 triée par date et une autre liste triée par taille.

8/ Exercice sur les fichiers

8.1/ Reprendre l'algorithme de calcul d'écart-type fait en 7,3 mais cette fois, les valeurs flottantes seront lues dans un fichier. Le fichier sera un fichier texte avec des nombres flottants séparés par des virgules.

8.2/ fonction grep. Écrivez un programme qui recherche un mot dans un fichier texte (le fichier et le mot seront donnés sur la ligne de commande lançant le programme). Pour chaque occurrence du mot dans le fichier, on affichera le numéro de ligne dans le fichier et la position du mot dans la ligne.

8,3/ fontion grep suite. A partir de 8,1/, créez 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 sous-répertoires. On écrira un parcours récursif de la hiérarchie de sous répertoires (voir 6,3)

8,4/ fontion 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.**