Cours

Python - Listes et fonctions

SNT

1) Listes

Les listes sont des **objets** Python. (*On approche ainsi la notion de programmation orientée objet*)
Une liste est écrite entre **crochets** et peut comporter des objets quelconques (entiers, chaînes, listes,..). On peut donc créer des listes de listes !
Les termes de la liste sont repérés par leur numéro **qui commence à 0**.
Exemple : maliste = [1, 'A',[1,2]] crée une liste nommée maliste qui contient 3 éléments :
maliste[0] contient l'entier 1,
maliste[1] contient la chaîne 'A',
maliste[2] contient la liste [1,2].
Il existe de nombreuses **fonctions** qui modifient les listes (en langage objet on parle de **méthodes**)
La plus utile est **append** :
Si liste = [1,2,4], l'instruction liste.append(5) transforme mon objet liste en [1,2,4,5].
(En langage objet la fonction s'applique **après l'objet** avec **un point de séparation**.)
D'autres méthodes utiles sont count, extend, insert, reverse, ….(*Voir la doc Python*)

La **boucle for** est aussi adaptée aux listes :
for element in liste : parcourt dans l'ordre les objets de la liste.

2) Les dictionnaires

Un dictionnaire permet de créer une liste de correspondances qui peuvent être très pratiques.
Un dictionnaire est écrit entre **accolades**.
Exemple : dico = {10 :'a',20 :'b',30 :'c'} crée un objet dictionnaire qui comporte 3 éléments.
Les entiers 10, 20, 30 sont les **clefs** correspondant aux **valeurs** 'a', 'b', 'c'.
La syntaxe est **{clef : valeur, clef : valeur, ...}**
dico[10] va donner alors la chaîne 'a'.
Penser à utiliser un dictionnaire lors de la conversion hexadécimal vers décimal par exemple.
Exemple : hex2ten={'A' :10,'B':11,.....}

3) Les fonctions

Il est souvent pratique d'utiliser des fonctions pour des parties qui servent plusieurs fois, pour la lisibilité d'un programme, ou pour utiliser une partie d'un programme dans un autre.
La syntaxe de création d'une fonction est :
def nomdelafonction(paramètres) :
Il peut y avoir un nombre quelquonque de paramètres (0,1,2,...).
S'il n'y en a aucun il faut mettre quand même les parenthèses.
Ensuite il y a la classique indentation (qui se termine avec la fonction).
Une fonction se termine souvent par **return(**objet**)** (pour récupérer qq chose!!!)
Exemple de fonction qui renvoie le nombre de diviseurs d'un entier naturel :

```
def nbdiv(n):
    nb=0
    for k in range(1,n+1):
        if n%k==0:
            nb=nb+1
    return(nb)
```

Dans le programme on utilise cette fonction de cette façon :

```
entier= int(input('nombre '))
print(nbdiv(entier))
```

Remarque importante :
Les variables utilisées dans les fonctions sont des **variables locales**, on peut reprendre un nom de variable utilisé ailleurs dans le programme, il n'y aura pas de conflit.

Si on veut modifier une variable externe à la fonction il faut utiliser l'instruction **global**.

Exemple 1

```
nombre=2
print(nombre)
def fonction_test():
    nombre=3
fonction_test()
print (nombre)
```

La variable nombre n'a pas changé !

Exemple 2

```
nombre=2
print(nombre)
def fonction_test():
    global nombre
    nombre=3
fonction_test()
print (nombre)
```

La variable nombre a changé !

4) Exercice
Dans un même programme :
Demander un entier naturel,
Recopier la fonction **nbdiv** (nombre de diviseurs) et écrire les fonctions :

listdiv (liste des diviseurs) qui renvoie la liste des diviseurs

premier (le nombre est premier?) qui renvoie une chaîne du style « Le nombre n'est pas premier » (les nombres premiers ont 2 diviseurs)

parfait (le nombre est parfait ?)
Un nombre est parfait lorsqu'il est égal à la somme de ses diviseurs propres (autres que lui même).

Exploiter vos fonctions :

Pour créer la liste des nombres premiers inférieurs à 1000,

Pour trouver des nombres parfaits,

Pour trouver le plus petit nombre qui possède un nombre de diviseurs donnés
(on peut sortir d'une boucle **for** par **break**)
Exemple 60 est le plus petit nombre qui possède 12 diviseurs. Et pour 54 diviseurs ?

Pour trouver des nombres amiables
(la somme des diviseurs de l'un est égale à la somme des diviseurs de l'autre, et cette somme de diviseurs vaut la somme des 2 nombres. Exemple 220 et 284)