Programmation python

I. Découverte des fonctions de base

1) Premier programme : « Hello World »

Pour afficher quelque chose à l'écran, on utilise la fonction python « **print** ». Comme tout bon apprenti en informatique, notre premier programme consiste à écrire à l'écran « Hello World! ».

En python :

```
print ( " Hello world! " )
```

Notes :

- le message affiché à l'utilisateur doit être entre guillemets.
- on peut mettre autant d'espace que l'on veut entre les symboles, mais **pas en début de ligne !**

```
print    (   " Hello world! "      )
```

2) Variables

On veux faire le calcul suivant : $6^2 - 4 \times 3 \times (-5)$. Mais on ne voudrait faire qu'une opération à la fois…
On commence par calculer 6^2. Il faut garder le résultat en mémoire pour l'utiliser après. On le stocke donc dans une **variable**.
La deuxième opération à faire, d'après les priorités de calcul, est la multiplication.
Puis on utilise les résultats précédents, stockés dans les variables, pour faire la soustraction.
Et on peut afficher le résultat.

```
carre = 6**2
multi = 4 * 3 * -5
res = carre - multi
print ( res )
```

Notes :

- on essaie de donner des **noms significatifs** aux variables, pour comprendre le code que l'on écrit !
- avec la fonction **print**, on peut afficher un message avant d'afficher la valeur d'une variable. Il suffit de séparer les messages et les variables par des virgules.

```
print ( " Le résultat est " , res )
```

message (entre "") *variable*

3) Saisie

On va demander le nom de l'utilisateur pour le saluer. Pour cela, on utilise la fonction python « **input** ».
On garde le nom de l'utilisateur dans une variable (ici, « *nom* »), que l'on utilise pour afficher notre message de bienvenue.

```
nom = input ( " Comment t'appelles-tu ? " )
print ( " Bonjour " , nom)
```

On demande maintenant deux nombres à l'utilisateur pour faire la somme.
Mais attention, si l'utilisateur saisit « 3 », l'ordinateur considère que c'est le caractère du clavier '3', et non le nombre trois. Pour **convertir** des caractères numériques en nombres, il faut utiliser la fonction « **int** » pour des nombres entiers, ou « **float** » pour des nombres décimaux.

```
a = int ( input ( " Saisir le 1er nombre entier :" ) )
b = int ( input ( " Saisir le 2e nombre entier :" ) )
res = a + b
print ( " La somme vaut ", res )
```

Exercice 1 : afficher le résultat de la multiplication de deux nombres choisis par l'utilisateur.
Exercice 2 : afficher les nombres choisis par l'utilisateur pour faire le calcul (ex. *« Le produit de 3 et 4 vaut 12 »*).
Exercice 3 : faire la somme de deux nombres décimaux : dans le programme précédent, remplacer « **int** » par « **float** ». Exécuter le programme et tester avec des nombres décimaux *(par exemple 0.2 et 1.3)* .

Note : attention, les calculs des nombres décimaux sont parfois mal gérés en informatique. Testez votre programme avec les nombres 0,7 et 0,2 par exemple !

4) Comparaison : « si »

On veut maintenant **obtenir un résultat différent** selon les valeurs choisies par l'utilisateur. Par exemple, on va afficher si le nombre choisi est plus grand ou plus petit que 10. Pour cela, il faut **comparer** le nombre saisi ; **s'il** est plus grand, on affiche « le nombre est plus grand », **sinon** on affiche « le nombre est plus petit ».

```
a = int ( input ("Quel nombre ? ") )
if a > 10 :
    print ("Ce nombre est plus grand que 10.")
else :
    print ("Ce nombre est plus petit que 10.")
```

Notes :

- ne pas oublier les « **:** » à la fin des lignes « if » et « else », ni les **indentations** (espaces) en début de **bloc** !
- on peut écrire des instructions « if » dans des **sous-blocs** (on aura plusieurs blocs imbriqués) :
- on peut ajouter des **commentaires** dans le code, en commençant la ligne avec le caractère #. Ces lignes ne sont **pas des instructions**, elles servent à expliquer le programme.

```
if a >  b :
    # a plus grand que b
    if b > c :
        print ...
    else :
        ...
else :
    # a plus petit que b
```

Exercice 4 : demander deux nombres et afficher le maximum.
Exercice 4.1 * : demander trois nombres et afficher le maximum.
Exercice 4.2 ** : demander trois nombres et les afficher dans l'ordre croissant.

Exercice 5 : demander deux nombres et l'opération que veut faire l'utilisateur :

Note : on peut enchaîner les tests avec l'instruction **elif**. C'est une contraction de **else** et **if**.

```
...
op = input("Quelle opération ? ")
if op == '+' :
    # on fait une somme
    res = a + b
elif op == '-' :
    # on fait une soustraction
    ...
...
# quel que soit le choix, on écrit le résultat :
print ("le resultat est", res)
```

5) Boucle : « pour »

On veut afficher les quatre premiers nombres entiers à l'écran.
On pourrait écrire quatre lignes :

```
print ("1")
print ("2")
print ("3")
print ("4")
```

Mais si on veut afficher *cent* nombres, cela devient un peu long…
On va plutôt **répéter** autant de fois que l'on veut les mêmes instructions.

```
# on utilise la variable "nombre" pour stocker le nombre
# à afficher (on commence par afficher "1")
nombre = 1
for i in range ( 4 ) :
    print ( nombre )
    # on incrémente la variable pour le prochain affichage
    # (pour afficher "2", puis "3"... )
    nombre = nombre + 1
# une fois les instructions répétées,
# on revient dans le bloc principal
print ( " sont les quatre premiers nombres entiers. " )
```

Notes :

- on pourrait utiliser directement la variable i, qui varie de 0 à 3.
- on peut utiliser la fonction « **range** » pour faire varier i de 1 à 4 : « **range** (1, 5) ».
- il existe un autre type de boucle : la **boucle conditionnelle « while »**. Il faut alors donner la condition d'arrêt, comme on donne une condition avec le « **if** ». Par exemple, `while i < 4 :`
  *Attention ! si la variable i n'est pas incrémentée dans le bloc, on ne sort jamais de la boucle. Le programme exécute alors une **boucle infinie** et finit par **crasher** !*

```
i = 1
while i < 4 :
    print ( i )
    i = i + 1
```

Exercice 6 : écrire un programme qui affiche les cinq premiers nombres entiers pairs.
Exercice 7 : écrire un programme qui affiche la somme des six premiers nombres entiers.
Exercice 8 : compléter le programme de l'exercice 5 pour pouvoir faire l'opération *factorielle*.