Algorithmique et Python: fonctions

BTS SIO

Exercice 1

1. Qu'affiche le programme suivant :

```
def Stars(n) :
    print(n*"*")

Stars(3)
Stars(10)
```

2. En utilisant cette fonction dans une boucle, produirte l'affichage suivant :

```
*
**
***
****
*****
```

Exercice 2

1. Qu'affiche le programme suivant :

```
def MaFonction(a,b) :
    y=a+b
    return y

a=2 ; b=5
c=MaFonction(a,b)
print("c = ", c)
```

2. Combien la fonction nommée MaFonction a-t'elle d'arguments en entrée ? en sortie ?

3. Écrire une fonction, nommée Moy, avec trois arguments qui retourne la moyenne des trois valeurs.

4. Écrire une fonction, nommée *Mention*, qui a un argument en entrée : une note entre 0 et 20, et en sortie renvoie la mention correspondante : P, AB, B, TB.

5. Écrire une troisième fonction, nommée *Apprecie*, avec trois arguments en entrée et qui renvoie la mention correspondante à la moyenne des trois valeurs entrées.
Cette fontion sera écrite en appelant les deux fonctions précédentes : *Moy* et *Mention*.

Exercice 3

1. Qu'affiche le programme suivant ?

```
def f(x) :
    return 100*x-4*x**2+2

print(f(2))
```

2. Écrire un programme qui affiche les valeurs *f(x)* pour tous les nombres entiers x entre 0 et 20.

3. Modifier le programme précédent pour qu'il trouve et affiche le maximum des valeurs *f(x)*, lorsque x est entre 0 et 20.

Exercice 4 Fonction factorielle

Écrire une fonction *Fact* d'un nombre entier n et qui calcule et affiche la factorielle de n.

On rappelle que la factorielle de n, notée en mathématiques n !, est le produit de tous les entiers positifs inférieurs à n. Par exemple la factorielle de 4 est $4! = 4 \times 3 \times 2 \times 1 = 24$.

On pourra donc utiliser que $1! = 1$ et la relation $n! = n \times (n-1)!$

Exercice 5

```
def u(n) :
  if n==0 :
    return 2
  else :
    return 3*u(n-1)-2

print(u(0))
print(u(1))
print(u(2))
```

1. Qu'affiche le programme ci-contre ?
   La fonction définie et utilisée ici s'appelle une fonctions **récursive** : c'est une fonction qui s'appelle elle-même...
   Modifier le programme précédent pour qu'il calcule, après $u_0 = 2$, les termes de la suite (u_n) définie par l'expression

$$u_n = \frac{2u_{n-1}^2 - 1}{u_{n-1}^2 + 2}$$

   Afficher en particulier les termes u_{10}, u_{20}.

Expliquer pourquoi le calcul des termes suivants devient aussi long. Proposer alors une modification de ce programme pour éviter ce problème et calculer u_{30}, u_{100} et u_{1000}.

Exercice 6 Réduction à un chiffre

```
n=412
s=str(n)

print(s[2])
print(s[0]+s[1])
print(int(s[0])+int(s[1]))
```

Considérer un nombre comme une chaine de caractère permet de le décomposer simplement chiffre par chiffre. Par exemple dans le code ci-contre.
a) Écrire une fonction *SommeChiffres* qui à un nombre entier donné en entrée, retourne la somme de ses chiffres.
Par exemple, *SommeChiffres(123)* retournera 6 (=1+2+3).

b) On réduit un nombre à un chiffre en réitérant la procédure précédente. Par exemple le nombre 8432 se réduit à 9+5+5+9=28 puis à 2+8=10 puis finalement au chiffre 1+0=1.

Écrire une fonction *Reduce2One* qui permet de réduire à un chiffre n'importe quel nombre.