Python 3 : La boucle répétitive for

1) La liste range

La commande range(n) énumère les entiers de 0 à **n-1**
La commande range(m,n) énumère les entiers de m à n-1

Cette commande permet de délimiter notre boucle for, voir ci-dessous.

2) La boucle for

La boucle for permet de répéter plusieurs fois une opération qui se situe après l'instruction for k in range(...) :

Voir l'exemple :

```
x=0
for k in range(0,11):
        x=x+k
print(x))
```

Ici, la partie range(0,11) signifie que les lignes en vert vont s'exécuter 11 fois (car on rappelle que range(0,11) correspond aux entiers (0,1,2,...10)) en remplaçant à chaque exécution k par les valeurs de range : 0 pour la première exécution, 1 pour la seconde, jusqu'à 10 pour la onzième.
Concrètement, au départ x vaut 0, puis la boucle se lance, on affecte à x la valeur 0+0, donc x vaut 0, ensuite on affecte à x la valeur 0+1 donc x vaut 1, puis on affecte à x la valeur 1+2 donc vaut 3, etc. Une fois la boucle relancée jusqu'à la dernière valeur de k, on sort de la boucle et le programme continue, ici il affiche x.
Cela se modélise donc par le calcul de la somme $\sum_{k=0}^{10} k$.

Regardons un autre exemple :

```
x=0
for k in range(0,11):
        x=x+1
print(x)
```

Ici, le programme va afficher 11, car à chaque itération (càd chaque exécution de la boucle), on affecte à x la valeur x+1 (donc on lui ajoute 1).

Ou encore

```
x=0
for k in range(0,11):
        x=x+1
        print(x)
```

Ici le programme va afficher toutes les valeurs entre 0 et 10 (car le print est dans la boucle)

Notons que la commande x=+1 joue le même rôle que x=x+1.
En général la commande x+=b affecte à x la valeur x+b.

3) Méthodes usuelles

a) Les sommes

On peut donc calculer des sommes. Si on veut calculer S= $\sum_{k=p}^{n} a_k$ où a_k est une suite de nombres réels, il suffit d'écrire :

```
S=0                              # on n'oublie pas de déclarer S !
for k in range(p,n+1) : # tous les entiers de p à n+1-1=n
        S=S+ak          # le a_k est à adapter dans chaque cas !
```

Par exemple pour calculer S= $\sum_{k=1}^{n100} \ln(k)$, il suffit d'écrire :

```
S=0
for k in range(1,101) :
        S=S+log(k)
```

Attention, si vous écrivez seulement S=log(k) dans la boucle, alors à chaque opération, vous allez remplacer la valeur de S, donc à la fin de la boucle, votre S vaut ln(100) et non pas la somme.

b) Les termes d'une suite récurrente

Si on a une suite récurrente définie par :
Un premier terme u_0 et la formule de récurrence $u_{n+1} = f(u_n)$
Si on souhaite un programme qui calcule la valeur de u_n (où n est un entier, par exemple rentré par l'utilisateur), on peut écrire :

```
u=u_0                            #la valeur de u_0 est à adapter dans chaque cas !
for k in range(1,n+1):
        u=f(u)                   #la fonction f est à adapter dans chaque cas !
```

Par exemple si $u_0 = 3$ et $u_{n+1} = 2u_n + 4$ et sin on veut calculer et afficher la valeur de u_{100} alors on peut écrire

```
u=3
for k in range (1,101):
        u=2*u+4
print(u)
```

Si le premier terme est u_2, il faut bien sûr adapter un peu, par exemple écrire :

```
u=u_2
for k in range(3,n+1):
```

u=f(u)

Faites attention à bien indexer vos boucles for avec les bons indices dans le range !
Un bon moyen de savoir si on met n, n+1 ou n-1 est de tester dans sa tête avec n=1 et de voir si ça renvoie bien u_1.

4) Boucles imbriquées

On peut imbriquer des boucles for, pour calculer des sommes doubles par exemple.

Si on veut calculer $\sum_{k=1}^{n}\sum_{i=1}^{k}\ln(k+i)$, on peut faire :

```
S=0
for k in range(1,n+1)
        for i in range(1,k+1):          # on utilise une nouvelle lettre !
                S=S+log(k+i)
```