Boucles *for*

◇ Une boucle *for* repète une liste d'intructions un certain nombre de fois. Une telle boucle est décrite par 4 éléments :

- Une *valeur de départ* a ;
- Une *valeur d'arrivée* b ;
- Un *compteur* k qui prend toutes les valeurs de a jusqu'à b ;
- Une *liste d'intructions* qui sont exécutées pour chaque valeur de k.

◇ Dans un algorithme, on écrit :

---

pour k **allant de** a **à** b
    Liste d'instructions
fin pour

Exemple Python. Afficher les carrés des entiers 1,2,3,4,5 et calculer la somme de ces carrés :

```
s = 0
for k in range(1,6):
    print(k**2) # Rappel : k**2 = k puissance 2
    s = s+k**2
print("Le total :",s)
```

```
1 4 9 16 25 Le total : 55
```

⚠*Remarques.*

- La commande **`range(a,b)`** désigne l'ensemble des entiers k tels que $a \leqslant k < b$ (on peut retenir que a est le première valeur prise par k et b est la première valeur qui n'est pas prise par k). Ici, **`range(1,6)`** représente l'ensemble des entiers de 1 à 5.
- Les intructions à répéter sont repérées par leur indentation. On notera que la dernière instruction de l'exemple ci-dessus ne fait par conséquent pas partie de la boucle (elle n'est exécutée qu'une seule fois).
- Les boucles *for* peuvent prendre des formes plus générales qui seront en partie vues plus tard. □

Application 1 : calcul des termes d'une suite $u_{n+1} = f(u_n)$

◇ Considérons à titre d'exemple la suite (u_n) définie par :

$$u_0 = 0.9; \qquad \forall n \in \mathbb{N},\ u_{n+1} = u_n - 0.3u_n^2$$

En pratique pour calculer u_n, on calcule u_1 à partir de u_0, u_2 à partir de u_1, etc. jusqu'à u_n. Avec PYTHON, on va utiliser une même variable u qui va contenir successivement toutes les valeurs. On pourra retenir le schéma suivant qui permet de passer du calcul « à la main » à l'algorithme de calcul de u_n :

$$\left.\begin{array}{l} u_0 = 0.9 \\ u_1 = u_0 - 0.3u_0^2 \\ u_2 = u_1 - 0.3u_1^2 \\ \quad\vdots \\ u_{n-1} = u_{n-2} - 0.3u_{n-2}^2 \\ u_n = u_{n-1} - 0.3u_{n-1}^2 \end{array}\right\} n \text{ étapes} \longrightarrow \left[\begin{array}{l} u=0.9 \\ u=u-0.3u^2 \\ u=u-0.3u^2 \\ \quad\vdots \\ u=u-0.3u^2 \\ u=u-0.3u^2 \end{array}\right] \longrightarrow$$

$u \longleftarrow 0.9$
pour k **allant de** 0 **à** $n-1$
    $u \longleftarrow u - 0.3u^2$
fin pour

On doit donc répéter n fois l'instruction $u = u - 0.3u^2$, ce qui est fait au moyen d'une boucle *for* :

```
def u(n):
    U = 0.9
    for i in range(0,n): # i va de 0 à n-1 (inclus)
        U = U-0.3*U**2
    return U
```

On peut alors calculer u_5 par exemple :

```
print(u(5))
```

```
0.3404393095450089
```

⚠*Remarque.* On doit faire n étapes de calcul, donc il faut bien écrire **`range(0,n)`** et pas **`range(0,n-1)`**. On peut abréger **`range(0,n)`** en **`range(n)`**. □

Remarque. On verra en TP des situations plus complexes ($u_{n+1} = f(n,u_n)$ ou $u_{n+1} = f(u_n,u_{n-1})$, on peut même imaginer que u_{n+1} s'écrit en fonction de $u_0,\ldots,u_n$). □

Application 2 : calcul d'une somme

◇ On veut écrire une fonction **`somme_carres(n)`** qui calcule et renvoie la valeur de la somme :

$$1+2^2+3^2+\cdots+(n-1)^2+n^2 = \sum_{k=1}^{n} k^2$$

On utilise une variable **`s`** qui vaut initialement 0 et on lui ajoute successivement tous les termes de la somme à l'aide d'une boucle *for*.

```
def somme_carres(n):
    s = 0
    for k in range(1,n+1):
        # k va de 1 à n
        s = s+k**2
    return s

print(somme_carres(3))
```

Exercices pour la semaine prochaine

Question 1. Écrire un programme (de moins de 10 lignes) qui affiche tous les nombres de 2 à 100 (inclus).

Question 2. Écrire un programme (de moins de 10 lignes) qui calcule la somme de tous les nombres de 2 à 100 (inclus).

Question 3. On considère la suite $(u_n)_{n\geqslant 0}$ telle que :

$$u_0 = 1 \quad \text{et} \quad \forall n \in \mathbb{N},\ u_{n+1} = \frac{1}{2}u_n + 1$$

- Écrire une fonction **`u(n)`** qui prend en paramètre l'entier n et qui renvoie la valeur de u_n.
- Utiliser cette fonction pour faire afficher la valeur de u_{10}.

Question 4.

- Écrire une fonction **`somme_geom(n,q)`** qui calcule et renvoie la valeur de la somme :

$$1 + q + q^2 + \cdots + q^{n-1}$$

- Utiliser cette fonction pour faire afficher la valeur de la somme $1 + 2 + 2^2 + \cdots + 2^{10}$.

Corrections

Q 1.

```
for k in range(2,101):
    # k va de 2 à 100
    print(k)
```

Q 2.

```
s = 0
for k in range(2,101):
    # k va de 2 à 100
    s = s+k
```

La variable **`s`** contient alors la somme de tous les nombres de 2 à 100.

Q 3.

```
def u(n):
    U = 1
    for k in range(0,n):
        # k va de 0 à n-1
        U = 1/2*U+1
    # Fin de la boucle
    return U

print(u(10))
```

```
1.9990234375
```

Q 4.

```
def somme_geom(n,q):
    s = 0
    for k in range(0,n):
        # k va de 0 à n-1
        s = s+q**k
    # Fin de la boucle
    return s

print(somme_geom(11,2))
```

```
2047
```