

(Notebook Capytale : 0fcb-77439)

4 – Boucle `for`

4.1 – Boucle bornée simple

4.1.1 – Répétition d'une instruction

Pour exécuter plusieurs fois la même instruction, il suffisait de recopier plusieurs fois la même instruction :

```
print("A")
print("A")
print("A")
```

Cette solution n'est pas raisonnable et elle n'est pas envisageable si on ne connaît pas le nombre de répétitions.

Python (et de nombreux langages de programmation) propose une instruction, appelée **la boucle** `for` permettant de gérer les **répétitions**.

Exemple

Forme la plus simple :

```
for _ in range(3): print("A")
```

Le nombre de tours de boucles peut dépendre d'une variable :

```
n = int(input())
for _ in range(2*n): print("A")
```

4.1.2 – Répétition d'un bloc d'instruction

La répétition n'est pas limitée à une instruction.

Exemple

```
[ ]: from random import randint

a = randint(1,6)
b = randint(1,6)
print("somme des deux dés : ",a+b)
a = randint(1,6)
b = randint(1,6)
print("somme des deux dés : ",a+b)
a = randint(1,6)
b = randint(1,6)
print("somme des deux dés : ",a+b)
a = randint(1,6)
b = randint(1,6)
print("somme des deux dés : ",a+b)
```

Pour cela les instructions formant le *corps de la boucle* doivent être regroupés en un **bloc** : une suite de lignes en retraits du même nombre d'espace.

Exemple

```
[ ]: from random import randint

for _ in range(4):
    a = randint(1,6)
    b = randint(1,6)
    print("somme des deux dés : ",a+b)
```

Une instruction supplémentaire qui n'est plus alignée avec le corps de boucle ne sera exécutée qu'une seule fois et après tous les tours de boucle :

Exemple

```
[ ]: from random import randint

print("Effectuons 4 expériences avec 2 dés :")
for _ in range(4):
    a = randint(1,6)
    b = randint(1,6)
    print("somme des deux dés : ",a+b)
print("Les 4 tirages sont effectués.")
```

4.2 – Utilisation de l'indice de boucle

Dans une boucle bornée, il est possible d'introduire une **nouvelle variable**

- accessible à l'intérieur du corps de boucle
- dont la valeur donne le numéro du tour de boucle
- appelée *indice de boucle* ou *compteur de boucle*

Exemple

```
[ ]: for i in range(10): print(i)
```

Le premier tour est 0, le deuxième 1, etc. Le numéro du dernier tour est donc égal à **un de moins que le nombre total de tours**.

4.3 – Utilisation des accumulateurs

On peut utiliser dans une boucle un *accumulateur* : une variable dont la valeur progresse à chaque tour de boucle.

```
[ ]: a = 1
for _ in range(4):
    a = a + 2
print(a)
```

Ce qui est équivalent au code suivant et affiche 9 après avoir donné à la variable *a* les valeurs 1, 3, 5, 7 et 9.

```
[ ]: a = 1
a = a + 2
a = a + 2
a = a + 2
a = a + 2
print(a)
```

Une boucle `for` permet de répéter une suite d'instructions regroupées en un **bloc**. **Le nombre de tours est prédéfini** et chacun des tours est associé à un **indice**.

4.4 Activités



ACTIVITÉ 1

(Capytale : 0fcb-77439) Écrire un programme qui demande un entier n à l'utilisateur, puis calcule et affiche le résultat de la multiplication

$$\underbrace{2 \times 2 \times 2 \times \dots \times 2}_{n \text{ fois}}$$



ACTIVITÉ 2

Écrire un programme qui calcule et affiche $1 \times 2 \times \dots \times 100$.



ACTIVITÉ 3

Écrire un programme qui demande un entier n à l'utilisateur puis calcule et affiche

1. $1 + 2 + \dots + n$
2. le nombre entier $n * (n + 1) // 2$



ACTIVITÉ 4

(Capytale : 1983-77502) Écrire un programme qui demande à l'utilisateur

- une somme d'argent initiale s déposée sur un livret,
- un taux d'intérêt annuel t exprimé en pourcents

– un nombre d'année n

et qui affiche les intérêts perçus chaque année ainsi que le montant total présent sur le livret après n années.

*(chaque année, il faut ajouter à s la quantité $s*t/100$)*



ACTIVITÉ 5

(Capitale : e7b7-77504)

1. Écrire un programme qui demande à l'utilisateur un nombre de chiffres n puis n chiffres, et qui calcule et affiche le nombre formé avec les n chiffres fournis dans l'ordre.
2. Écrire une variante du programme précédent dans lequel les chiffres sont donnés dans l'ordre inverse.



ACTIVITÉ 6

(Capitale : a7bc-77505) Écrire un programme qui demande à l'utilisateur un nombre entier n et un nombre de chiffres k , et qui affiche successivement les k derniers chiffres de n , en commençant par les unités.

Si n contient moins de k chiffres, il suffira d'afficher des zéros à la fin.

(on rappelle que $n \% 10$ renvoie le chiffre des unités de n)

**ACTIVITÉ 7**

(Capytale : 2294-77506) La suite de Fibonacci est la suite d'entiers (F_n) définie par $F_0 = 0, F_1 = 1$ et, pour tout entier n à partir de 2, $F_n = F_{n-1} + F_{n-2}$.

Écrire un programme qui demande à l'utilisateur un entier n qu'on supposera supérieur ou égal à 1, et qui affiche la valeur de F_n .

(on utilisera deux variables pour mémoriser F_n et F_{n-1} ainsi qu'une variable temporaire)