(Notebook Capytale : 0fcb-77439)

5 — Boucle `for`

5.1 — Boucle bornée simple

5.1.1 — 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 connait 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")
```

5.1.2 — 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.")
```

5.2 — 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**.

5.3 – 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**.