Nom :	Prénom :	Classe :

SNT — Boucles bornées

Objectifs :
- Savoir identifier une boucle bornée.
- Comprendre le rôle d'une boucle bornée.
- Savoir modifier une boucle bornée.
- Savoir écrire une boucle bornée.
- Savoir tracer une valeur à travers des itérations.

Introduction

En programmation, il arrive parfois que l'on répète une même instruction un certain nombre de fois.

Par exemple, si nous souhaitons faire avancer un robot 10 fois, nous pouvons lui faire exécuter :

```
1  avance()
2  avance()
3  avance()
4  avance()
5  avance()
6  avance()
7  avance()
8  avance()
9  avance()
10 avance()
```

Et que se passerait-t-il si nous devions le faire avancer non pas 10 mais 1000 fois ?
Faudrait-il écrire 1000 fois de suite la même instruction `avance()` ?

Heureusement, les femmes et hommes qui ont inventés la programmation nous proposent un nouvel outil : la **boucle bornée**. Une boucle bornée, est une structure qui permet de **répéter** un **bloc d'instructions** un **nombre de fois** déterminé à l'avance.

En reprenant notre exemple de robot à faire avancer **10** fois, cela donnerait :

```
1  répète 10 fois:
2      avance()
```

Et pour le faire avancer **1000** fois, il suffirait alors d'écrire :

```
1  répète 1000 fois:
2      avance()
```

Voilà qui est plus simple !

Boucle bornée

Une boucle bornée **répète** une *ou plusieurs* instructions en sachant dès le départ combien de fois cette répétition aura lieu (par exemple 10 fois).

En Python, une boucle bornée s'écrit de la façon suivante :

```
for _ in range(n):
    instruction
```

Ici, l'instruction sera exécutée *n* fois.

Exemple Python

Prenons un exemple, nous souhaitons afficher 10 fois de suite le texte `Bonjour`.
Voici comment nous pourrions le faire :

```
1  for _ in range(10):
2      print("Bonjour")
```

➤ Exercice 1 — Prédire

Lisez le programme suivant :

```
1  for _ in range(2):
2      print('oui')
3      print('non')
```

Puis essayez de **deviner** ce que ce programme affichera. Écrivez votre prédiction dans l'espace ci-dessous :

➤ Exercice 2 — Vérifier

Exécutez le programme de l'exercice 1, en **scannant** le QR code ci-dessous :

Alors, votre prédiction était-elle juste ?

➤ **Exercice 3** — Enquêter

Pour rappel[1], une instruction `print()` en Python permet d'afficher un message et/ou le contenu d'une variable. À chaque fois que Python exécute cette instruction `print()`, il affiche une **nouvelle ligne**.

Par exemple, le programme suivant affiche deux lignes :

```
1  prenom = "Anna"
2  print(prenom)
3  print(prenom)
```

Dans les programmes ci-dessous, déterminez le **nombre de lignes** affichées :

N°	Programme Python	Nombre de lignes affichées (à compléter)
1	`for _ in range(3):` `  print('Salut')`	

2	`print('One')`
	`for _ in range(3):`
	`  print('Two')`	

3	`age = 16`
	`for _ in range(age):`
	`  print('Happy birthday')`	

4	`a = 0`
	`for _ in range(3):`
	`  a = a + 1`
	`  print(a)`	

5	`for _ in range(2):`
	`  for _ in range(2):`
	`  print('Hello')`	

➤ **Exercice 4** — Modifier

Le programme ci-dessous à gauche affiche un certain nombre de fois le texte "Pizza".
Proposez-en une nouvelle version, à droite, en **utilisant une boucle bornée** :

Un programme Python	Votre programme Python (à compléter)
`print('Pizza')` `print('Pizza')` `print('Pizza')` `print('Pizza')`	

1 Voir le cours « Premiers pas en Python : calculs, commentaire, variable ».

➤ **Exercice 5** — Créer

Créez le programme Python correspondant à la description suivante :
Affichez une fois le texte `Bonjour`, puis affichez 10 fois le texte `ça va ?` :

Itération

En programmation, une **itération** est une **répétition unique** d'un **bloc d'instructions** à l'intérieur d'une boucle. Ou, plus concrètement, quand une boucle s'exécute plusieurs fois, chaque passage dans le bloc de code est une itération.

Par exemple, dans le programme ci-dessous le bloc de code sera exécuté 5 fois. On dit alors qu'il y a 5 itérations.

```
1  for _ in range(5):
2      print("Python")
3      print("C'est cool, non ?")
```

➤ **Exercice 6** — Enquêter

Déterminez le nombre d'itérations dans les programmes ci-dessous :

N°	Programme Python	Nombre d'itérations (à compléter)
1	`for _ in range(3):` `  print('Salut')`	
2	`for _ in range(10):` `  print(20)`	

3	`for _ in range(10):`
	`  print('oui')`
	`  print('non')`
	`  print('ok')`	

Tracer des valeurs, au fur et à mesure des itérations

Il est possible de suivre (on peut dire aussi *tracer*) des valeurs, au fur et à mesure des itérations.

Prenons par exemple le programme suivant :

```
1  a = 0
2  for _ in range(4):
3      a = a + 10
```

Une question pourrait être :
Quelle est la valeur stockée dans la variable `a`*, à la fin du programme ?*

À première vue, difficile de répondre.
Une astuce peut nous permettre d'y voir plus clair : dresser un tableau permettant de suivre la valeur de `a` :

Ligne du programme	Itération	Valeur stockée dans `a`
1		`0`
2		`0`
3	1	`10`
3	2	`20`
3	3	`30`
3	4	`40`

Voici comment lire ce tableau :

- À la ligne 1 du programme, la valeur `0` est stockée dans la variable `a`.
- Ligne 2, on créé une boucle bornée qui s'apprête à itérer 4 fois.
  Mais cette ligne ne provoque aucun changement de valeur pour `a`.
- Ligne 3, on est *dans l'itération de la boucle*.
  Cette ligne 3 être exécutée 4 fois (car la boucle itère quatre fois).
  - Itération 1 : on ajoute `10` à `a` qui vaut `0`, donc `a` vaut désormais `10`.
  - Itération 2 : on ajoute `10` à `a` qui vaut `10`, donc `a` vaut désormais `20`.
  - Itération 3 : on ajoute `10` à `a` qui vaut `20`, donc `a` vaut désormais `30`.
  - Itération 4 : on ajoute `10` à `a` qui vaut `30`, donc `a` vaut désormais `40`.

Puis, une fois la quatrième itération effectuée, la boucle s'arrête.
Le programme est terminé. À la fin du programme, la valeur de `a` est donc `40`.

➤ **Exercice 7** — Enquêter

Tracez la valeur de `a` dans chacun des programmes ci-dessous (complétez les cases marquées d'un ...)

<table>
<tr><th>N°</th><th>Programme Python</th><th>Traçage de la valeur de <code>a</code> (à compléter)</th></tr>
<tr><td>1</td><td><pre>a = 0
for _ in range(2):
    a = a + 2</pre></td><td>
<table>
<tr><th>Ligne</th><th>Itération</th><th>Valeur de <code>a</code></th></tr>
<tr><td>1</td><td></td><td>...</td></tr>
<tr><td>2</td><td></td><td>...</td></tr>
<tr><td>3</td><td>1</td><td>...</td></tr>
<tr><td>3</td><td>2</td><td>...</td></tr>
</table>
</td></tr>
<tr><td>2</td><td><pre>a = -2
for _ in range(3):
    a = a + 2</pre></td><td>
<table>
<tr><th>Ligne</th><th>Itération</th><th>Valeur de <code>a</code></th></tr>
<tr><td>1</td><td></td><td>...</td></tr>
<tr><td>2</td><td></td><td>...</td></tr>
<tr><td>3</td><td>...</td><td>...</td></tr>
<tr><td>3</td><td>...</td><td>...</td></tr>
<tr><td>3</td><td>...</td><td>...</td></tr>
</table>
</td></tr>
<tr><td>3</td><td><pre>a = 2
for _ in range(2):
    a = a + 10</pre></td><td>
<table>
<tr><th>Ligne</th><th>Itération</th><th>Valeur de <code>a</code></th></tr>
<tr><td>...</td><td></td><td>...</td></tr>
<tr><td>...</td><td></td><td>...</td></tr>
<tr><td>...</td><td>...</td><td>...</td></tr>
<tr><td>...</td><td>...</td><td>...</td></tr>
</table>
</td></tr>
</table>