Nom :	Prénom :	Classe :

NSI 1re — Boucle bornée (boucle for)

Objectifs :
- Savoir identifier une boucle bornée
- Connaître l'utilité d'une boucle bornée
- Savoir utiliser une boucle bornée
- Savoir créer une boucle bornée

Intro

Dans la vie de tous les jours, nous effectuons certaines tâches répétitives, mécaniquement. Par exemple :

1. Je me lève
2. Je scroll sur mon tel
3. Je m'habille
4. Je déjeune

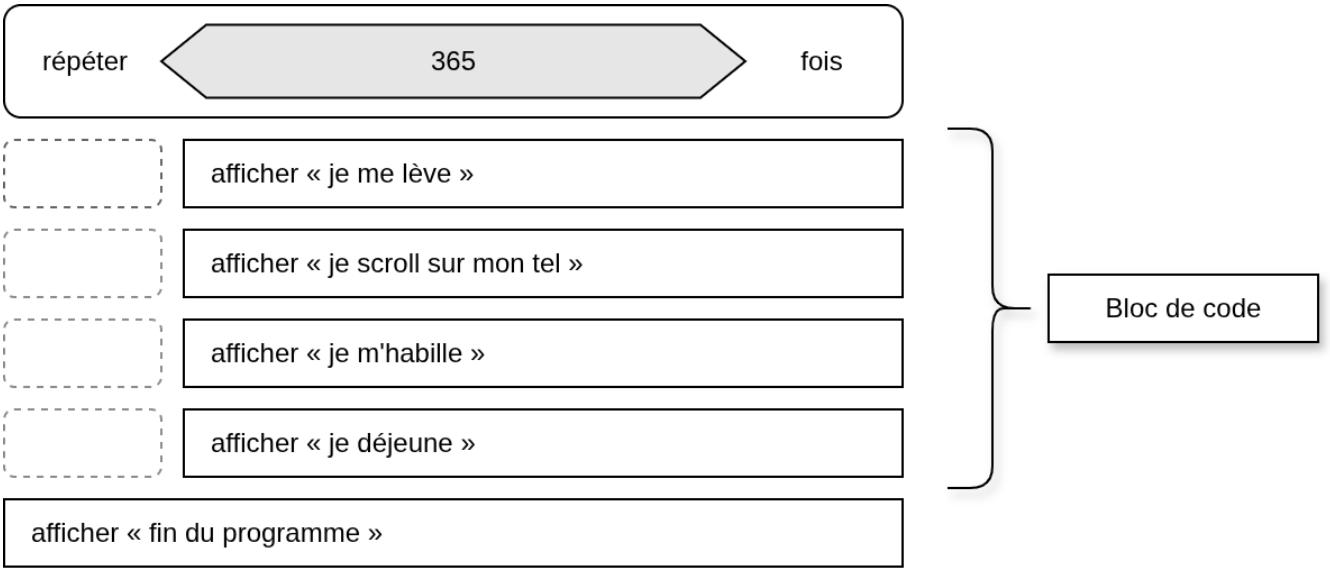
Dans cet exemple, nous avons une série d'instructions (composée de 4 instructions).
Cette série d'instructions est **répétée** chaque jour.

Il en va de même dans la programmation, où il est possible de créer des répétitions, appelées **boucles**.

Boucle bornée

Une **boucle bornée**, en programmation, est une structure qui permet de répéter une série d'instructions un nombre de fois déterminé à l'avance.

En reprenant l'exemple précédent, nous pouvons schématiser une boucle bornée de la façon suivante :

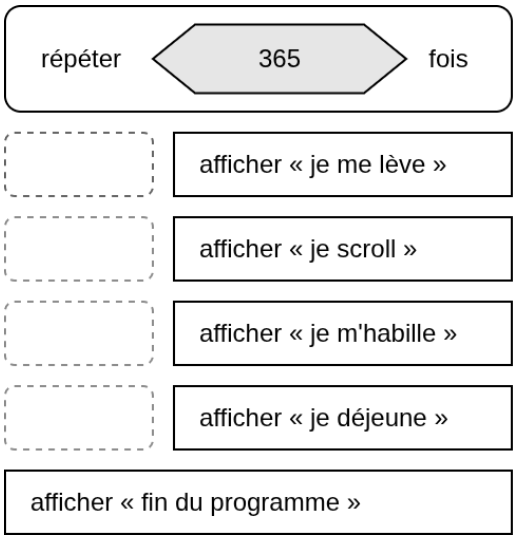


Dans ce schéma, nous avons créé une boucle bornée qui va répéter (**itérer**) un certain nombre de fois (365 fois) une série d'instructions (bloc de code).

Puis, lorsque les répétitions sont terminées, le programme affiche ici un message ("fin du programme").

Boucle bornée en Python

Poursuivons sur le même exemple de boucle, et observons sa transcription en Python :




```
1  for _ in range(365):
2      print("je me lève")
3      print("je scroll")
4      print("je m'habille")
5      print("je déjeune")
6  print("fin du programme")
```

Dans ce code Python, nous avons créé une boucle bornée à l'aide de l'instruction `for _ in range(365)` qui va répéter (**itérer**) un nombre *n* de fois (ici *n* vaut 365) une série d'instructions (bloc de code).

Puis, lorsque les répétitions sont terminées, le programme affiche un message ("fin du programme").

➤ Exercice 1 (à vous de jouer)

Combien de fois seront affichés les messages "je me lève", "je scroll", etc. ?

Combien de fois sera affiché le message "fin du programme" ?

➤ Exercice 2

Observez le nouvel exemple de code Python ci-dessous, puis **essayez de deviner** les messages affichés :

Code Python

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

Quels seront les messages affichés, **d'après vous** ? (à compléter)

Nous vérifierons votre prédiction un peu plus tard dans ce cours ;)

Traçage explicite

Pour comprendre le fonctionnement d'un programme, il est possible d'analyser son code, et de suivre l'évolution des valeurs des variables utilisées dans un tableau (c'est ce qu'on appelle un *traçage explicite*).

Prenons un autre exemple de code Python :

```
1  i = 2
2  for _ in range(4):
3      i = i + 1
4  print(i)
```

Et traçons la valeur de la variable `i`, au fur et à mesure du programme :

- Ligne 1, la variable `i` est déclarée, et on lui assigne la valeur `2`.
- Ligne 2, une boucle bornée est initialisée. Cette boucle va itérer `4` fois.
  - Lors de la 1ère itération, `i` est incrémentée de `1`. Donc `i` vaut `3`.
  - Lors de la 2e itération, `i` est incrémentée de `1`. Donc `i` vaut `4`.
  - Lors de la 3e itération, `i` est incrémentée de `1`. Donc `i` vaut `5`.
  - Lors de la 4e itération, `i` est incrémentée de `1`. Donc `i` vaut `6`.
- Ligne 4, la boucle est terminée. La valeur de `i` affichée est `6`.

Ce traçage peut également être représenté sous la forme d'un **tableau** :

Ligne	🔃 Itération	Valeur de `i`
1	-	2
3	1	3
3	2	4
3	3	5
3	4	6
4	-	6

➤ Exercice 3

Observez le code suivant :

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

Puis réalisez un traçage explicite de la valeur de `i`, en complétant ce tableau :

Ligne	🔃 Itération	Valeur de `i` (à compléter)
1	-	
3	1	
4	1	
3	2	
4	2	
3	3	
4	3	

Le code de cet exercice 3 est le même que celui de l'exercice 2 (page 2) !

Alors, une fois ce traçage explicite terminé, utilisez-le pour vérifier votre prédiction lors de l'exercice 2. Si vous vous étiez trompé, corrigez votre exercice 2 en utilisant une autre couleur ...

➤ Exercice 4

Nina aimerait **afficher** tous les **chiffres** allant de 0 à 10 inclus.
Elle a commencé à coder un petit programme, mais elle n'a pas eu le temps de finir…
Pourriez-vous l'aider, en complétant les zones grises :

```
1  i =                    # à compléter
2  for _ in range(      ):  # à compléter
3      i = i +            # à compléter
4      print(i)
```

N'hésitez pas à vous aider d'un traçage explicite sur un brouillon, à côté.

➤ Exercice 5

Une agence près de chez vous cherche à développer le programme Python suivant :

Description du programme	Code Python (à compléter)
- Déclarer une variable `age` valant `17` - Itérer 3 fois sur une incrémentation de `age` - À la fin, afficher la valeur de `age`	

➤ Exercice 6

Complétez le **tableau** de traçage explicite de la variable `z` du code Python ci-dessous :

```
1  z = -6
2  for _ in range(2):
3      z = z + 3
```

Ligne	🔃 Itération	Valeur de `z`
1		

Que peut-on coder dans le bloc de code d'une boucle ?

Tout ! Oui, nous pouvons coder **tout** ce qui nous passe par la tête dans le bloc de code d'une boucle.

Quelques exemples :

N°	Code Python	Description

1	`a = 0`
	`for _ in range(10):`
	`  a = a + 1`
	`  if a == 5:`
	`  print('5 ans, déjà')`
		Le bloc de code de cette boucle bornée contient une condition.
		
		Que va afficher ce programme ?
		Il affichera une seule fois "5 ans, déjà" (lorsque `a` vaudra 5)

2	`b = 0`
	`for _ in range(4):`
	`  if b == 0:`
	`  print('Rouge')`
	`  b = 1`
	`  else:`
	`  print('Vert')`
	`  b = 0`
		Le bloc de code de cette boucle bornée contient une condition if / else.
		
		Si **l'affirmation** est **vraie**, on affiche "Rouge", et on change la valeur de `b` à `1`.
		Si **l'affirmation** est **fausse**, on affiche "Vert", et on change la valeur de `b` à `0`.

Peut-on boucler dans une boucle ? #inception

Nous venons de voir qu'il est aussi possible de construire des conditions dans des boucles.
- Mais alors, est-il possible de **construire une boucle** ... *dans* une boucle ?

Pour le découvrir, essayons le petit exercice suivant.

➤ Exercice 6

Un ami américain (Dom Cobb[1]) a écrit le programme suivant :

```
1  a = 0
2  b = 0
3  for _ in range(2):       # itération
4      a = a + 1
5      for _ in range(3):   # sous-itération
6          b = b + 1
7  print(a, b)
```

Il vous met au défi et vous demande quelles valeurs seront affichées, à la fin de son programme ?

➤ Pour le savoir, réalisez le traçage explicite de `a` et de `b`, en complétant **toutes** les **cases vides** de ce tableau :

Ligne	🔃 Itération	🔃 Sous-itération	Valeur de `a`	Valeur de `b`
1	-	-	0	-
2	-	-	0	0
4				
6				
6				
6				
4				
6				
6				
6				
7				

🌶️ Pour aller plus loin

Pour aller plus loin, créez sur un brouillon un programme affichant les tables de multiplications de 1 à 10 ...

1 Dans le film « Inception », Dom Cobb (joué par Leonardo DiCaprio) est un voleur professionnel, spécialisé dans l'escroquerie, dont la spécialité est de dérober les secrets de ses victimes en s'infiltrant dans leurs rêves.