Langage et Programmation	Chap 03	**Boucle `for`**

3.1 Pourquoi des boucles ?

Éviter les répétitions inutiles

Jusqu'à présent, notre code ne nous permettait pas d'être très efficace : si l'on voulait afficher dix fois un message ou additionner plusieurs nombres, il fallait écrire plusieurs lignes presque identiques.

Prenons un exemple simple :

```
print("Bonjour")
print("Bonjour")
print("Bonjour")
print("Bonjour")
print("Bonjour")
print("Bonjour")
```

Ce code fonctionne, mais il n'est ni élégant ni pratique. Et si demain on voulait l'afficher 1000 fois ? Il faudrait recopier les lignes encore et encore...

C'est précisément pour cela que les boucles existent : elles permettent de **répéter automatiquement** un bloc d'instructions plusieurs fois, sans le réécrire.
Avec les **boucles**, pour afficher 1000 fois "Bonjour" nous avons juste à écrire :

```
for i in range(1000):
    print("Bonjour")
```

3.2 La boucle `for`

Principe général

Une boucle `for` permet de parcourir une suite de valeurs, qu'on appelle une **séquence** (par exemple une chaîne de caractères ou un intervalle de nombres).
Sa forme générale est la suivante :

```
for valeur in sequence:
    # bloc d'instructions à répéter
```

À chaque tour de boucle, la variable **valeur** prend une nouvelle valeur issue de la séquence, puis les

instructions du bloc sont exécutées. Quand il n'y a plus de valeur, la boucle s'arrête.

Exemple 1 — Premiers pas avec `range()`

En Python, la fonction `range()` permet de créer une séquence de nombres. C'est souvent la première manière d'utiliser une boucle.

```
for i in range(5):
    print("Ligne numéro", i)
```

Ici, la variable `i` prend successivement les valeurs 0, 1, 2, 3 et 4. Le programme affiche donc cinq lignes, sans qu'on ait eu besoin d'écrire cinq fois la commande `print()`.

Affichage :

Les paramètres de `range()`

La fonction `range()` peut être utilisée de plusieurs manières :

- `range(n)` : génère les entiers de 0 à `n-1`.
- `range(debut, fin)` : va de `debut` à `fin-1`.
- `range(debut, fin, pas)` : ajoute un pas (positif ou négatif).

```
for k in range(2, 6):
    print(k)
```

Affichage :

```python
for t in range(6, 0, -2):
    print(t)
```

Affichage :

Les paramètres de `range()`

La fonction `range()` peut être utilisée de plusieurs manières :

— `range(n)` : génère les entiers de 0 à `n-1`.

— `range(debut, fin)` : va de `debut` à `fin-1`.

— `range(debut, fin, pas)` : ajoute un pas (positif ou négatif).

```python
for k in range(2, 6):
    print(k)
```

Affichage :

Attention

Attention : la borne de fin n'est jamais incluse. Pour aller jusqu'à 10 inclus, il faut écrire `range(0, 11)`.

3.3 Parcourir d'autres séquences

Parcourir une chaîne

Une boucle `for` ne sert pas qu'à compter. On peut aussi l'utiliser pour parcourir directement une chaîne de caractères, c'est-à-dire "visiter" chaque caractère l'un après l'autre.

```python
mot = "patate"
for lettre in mot:
    print("La lettre vaut", lettre)
```

Affichage :

```
---------------------------------------

---------------------------------------

---------------------------------------

---------------------------------------

---------------------------------------

---------------------------------------
```

3.4 Les erreurs classiques

Ce qu'il faut éviter

- Oublier les deux points : après un `for` ou un `if`.
- Mauvaise indentation : le bloc à répéter doit être décalé vers la droite.
- Penser que la borne de fin de `range()` est incluse (alors qu'elle ne l'est pas).

3.5 À retenir

Astuce

- Une boucle `for` permet de répéter un ensemble d'instructions automatiquement.
- Elle parcourt une séquence (nombres, texte, liste, etc.) sans qu'on ait à compter soi-même.
- Elle devient très utile quand on la combine avec des conditions pour créer des comportements variés.

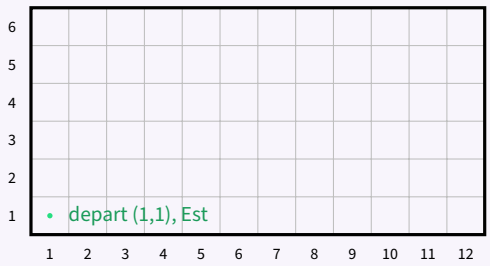
En résumé : une boucle permet d'écrire **moins de code** pour faire **plus d'actions**.

3.6 Robot dessinateur sur quadrillage

Exercice 1 — Principe general

Vous pilotez un petit robot sur une grille. Il commence en case (1,1), oriente vers l'Est. Votre objectif est d'ecrire, sur papier, des algorithmes en style Python pour tracer des motifs.

Important : les missions sont graduees. Utiliser uniquement les **instructions autorisees au niveau** de la mission. Les blocs `python` ci-dessous servent de zone de brouillon pour ecrire vos algorithmes.



Grille (12x6)

Jeu d'instructions par niveau

Niveau 1 (missions M1 et M2) :

- Boucle for
- `avancer()` : avancer d'une case dans la direction actuelle.
- `marquer()` : colorier la case courante en noir.
- `tourner_gauche()` et `tourner_droite()` : pivoter de 90 degres.

Niveau 2 (missions M3 et M4) : les memes instructions que niveau 1, plus :

- `est_devant_libre()` : renvoie True si la case devant est hors limite.
- Conditions autorisees (`if`, `elif`, `else`, `and`, `or`).

Niveau Bonus (M5) : Il faut être astucieux

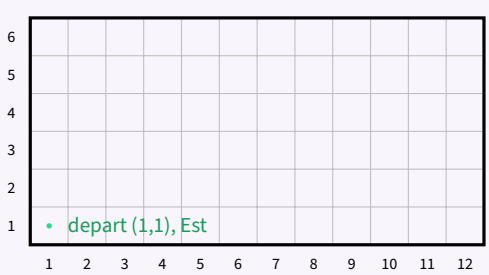
Exercice 2 — M1 — Frise simple (Niveau 1)

Objectif. Tracer des points espaces regulierement sur la premiere ligne. Le robot doit marquer, puis avancer de deux cases, et recommencer, 6 fois de suite.

Consigne. Utiliser uniquement `marquer()`, `avancer()`

```
# Depart suppose en (1,1), oriente Est.
# Ecrire une boucle for qui repete 6 fois :
#   marquer()
#   avancer()
#   avancer()
```

Tracez votre resultat sur la grille ci-dessous.



Grille M1 (12x6)

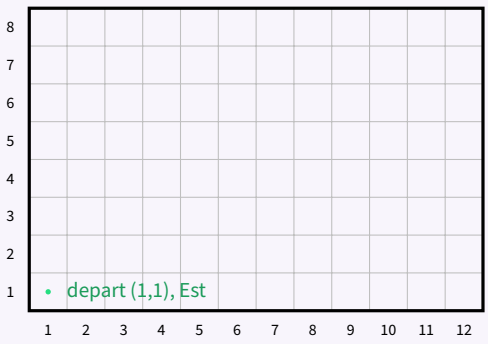
Exercice 3 — M2 — Petit zigzag guide (Niveau 1)

Objectif. Dessiner un zigzag sur deux lignes : avancer en marquant vers la droite, monter d'une case, puis repartir vers la droite, et ainsi de suite, pour 4 repetitions.

Idées. Commencez déjà par dessiner un petit zigzag !

```
1
2
3
4
5
6
7
8
9
10
11
12
```

Tracez votre resultat sur la grille ci-dessous.



Grille M2 (12x8)

Exercice 4 — M3 — Damier 4x4 (Niveau 2)

Objectif. Colorier un damier 4x4 (une case sur deux) en partant d'un coin. Vous pouvez maintenant utiliser `if` et une variable pour alterner.

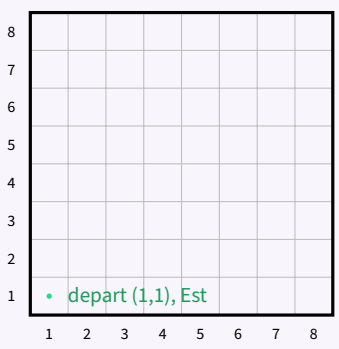
Pistes.

— On peut utiliser la parité de la ligne pour savoir ce qu'on doit faire.

— On peut aussi utiliser une variable de test

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

Tracez votre resultat sur la grille ci-dessous.



Grille M3 (8x8, zone utile 4x4)

Exercice 5 — M4 — Bordure de la grille (Niveau 2)

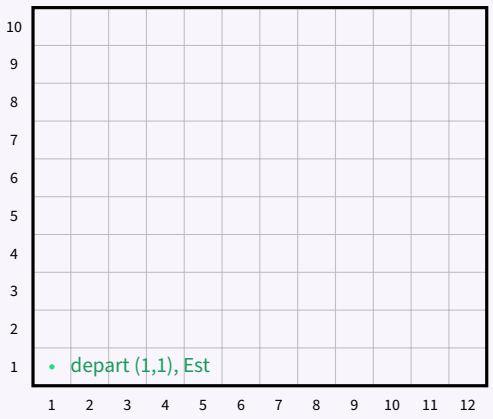
Objectif. Tracer uniquement la bordure de la grille, vous devez utiliser la fonction `est_vide_devant()` pour progresser.

Pistes.

— Commencer déjà par écrire le code qui permet de dessiner un rectangle fixe

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

Tracez votre resultat sur la grille ci-dessous.



Grille M4 (12x10)

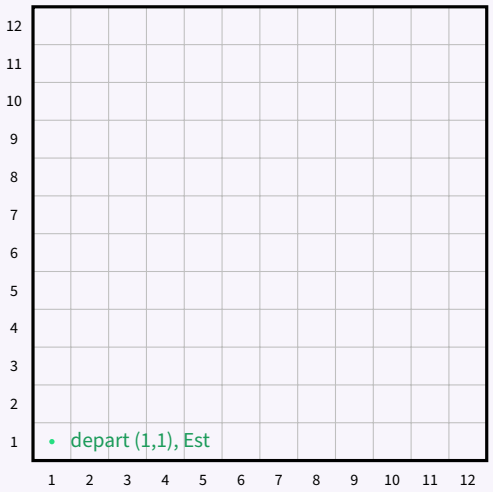
Exercice 6 — M5 — Spirale carree (Bonus)

Objectif. Realiser une spirale avec des longueurs decroissantes : 6,6,5,5,4,4,3,3,2,2,1,1.

Piste. Utiliser une boucle `for` sur la liste des longueurs. Pour chaque longueur `L`, repeter `marquer()+avancer()` L fois, puis tourner a droite.

```
# for L in [6,6,5,5,4,4,3,3,2,2,1,1]:
#     for i in range(L):
#         marquer(); avancer()
#     tourner_droite()
```

Tracez votre resultat sur la grille ci-dessous.



Grille M5 (12x12)