

# LSIN200A - Cours 3

Coline Gianfrotta

coline.gianfrotta@ens.uvsq.fr

19 février 2026

# Première évaluation

- Le premier contrôle sera sur papier et aura lieu **jeudi 12 mars**

## Les structures itératives (boucles)

**Structure de contrôle qui répète certaines lignes de code jusqu'à ce qu'une condition déterminée soit satisfaite.**

*Exemple:* On veut afficher *Programmer c'est cool!* 100 fois. Comment doit-on procéder ?

Il serait fastidieux de taper l'instruction 100 fois:

```
print("Programmer c'est cool!")
print("Programmer, c'est cool!")
...
print("Programmer, c'est cool!")
```

# L'itération

En programmation, une boucle a plusieurs composantes:

- Une **variable de contrôle**: initialisée à une certaine valeur, elle est le point de départ dans la boucle
- Une **condition de sortie**: l'arrêt de l'itération est déclenché par une condition sur l'état des variables (en particulier la variable de contrôle)
- Un **itérateur**: incrémente généralement la variable de contrôle petit à petit à chaque boucle successive, jusqu'à ce que celui-ci remplisse la condition de sortie.

## Deux structures itératives

`for`

`while`

# L'instruction for

- Structure itérative qui permet d'exécuter un bloc de code, un nombre **prédéterminé** de fois
- en Python, on itère sur les objets d'une séquence d'objets (liste d'entiers, chaîne de caractères par exemple)

## Syntaxe

```
for nom_variable in sequence_objets:  
    instruction 1  
    instruction 2  
    ...  
    instruction n
```

- La **variable de contrôle** (`nom_variable`) prend une à une les valeurs dans la séquence d'objets.
- Les instructions **indentées** dans le corps de la boucle sont exécutées une fois pour chaque valeur de la variable de contrôle

## L'instruction for avec range()

- La fonction **range(...)** retourne une séquence de nombres, commençant par une **valeur\_initiale** et se terminant par une **valeur\_de\_fin - 1**.
- La variable de contrôle prend les valeurs définies par **range(valeur\_initiale, valeur\_de\_fin)**
- Les valeurs de la fonction **range()** doivent être de type **entier**.
  - Par exemple, **range(1.5, 8.5)** est incorrect.

### Exemple

```
print("Afficher les valeurs dans l'intervalle:")
for i in range(1, 11):
    print("i=", i)
print("Terminé")
```

# La fonction range()

## Trois variantes pour range()

```
range(valeur_initiale, valeur_de_fin)
```

```
range(valeur_de_fin)
```

```
range(valeur_initiale, valeur_de_fin, increment)
```

- La fonction `range(valeur_de_fin)` est équivalente à la fonction `range(0, valeur_de_fin)`.
- Dans la fonction `range(valeur_initiale, valeur_de_fin, increment)`, le paramètre *increment* est utilisé comme pas d'avancement.
  - il est facultatif. S'il est omis, le pas sera pris par défaut égal à 1.

## Exemple

```
print("Afficher les valeurs dans l'intervalle")
for i in range(3, 9, 2) :
    print("i =", i)
print("Terminé")
```

# Variable de contrôle

## Exemple

```
for i in range(1, 11, 3)
    print("Avant addition i =", i)
    i += 1
    print("Après addition i =", i)
print("Dernière valeur, i = ", i)
print("Terminé")
```

- En modifiant la valeur de `i` dans le corps de la boucle, la valeur modifiée sera utilisée pour le reste de cette itération
- Cependant, lorsque cette itération se termine, la valeur précédente de `i` est restaurée.

## Exemple

Que fait le programme suivant ?

```
n = int(input("Entrez un nombre:"))
result = 0
for i in range(1, n+1):
    result += i
print("La somme vaut:", result)
```

## La boucle for: autres variantes

```
liste_de_nombres = [23, 43, 56, 22, 109, 34]
sum = 0
for nb in liste_de_nombres:
    sum += nb
moyenne = sum / len(liste_de_nombres)
print("La moyenne vaut", moyenne)
```

- La boucle for est utilisée quand on veut faire un **traitement sur un ensemble de données regroupées dans une séquence**.

# La boucle while

Utilisée lorsqu'**on ne connaît pas à l'avance le nombre de fois que le traitement sera répété.**

## Syntaxe

```
while condition:  
    instruction 1  
    instruction 2  
    ...  
    instruction n
```

- Une **expression de contrôle** (condition), expression booléenne, contrôle l'exécution de la boucle.
- Cette expression est testée avant chaque itération
- Les instructions dans le corps de la boucle sont exécutées une fois par itération si l'expression de contrôle reste satisfaite.

# La boucle while

## Exemple

```
i = 1
while i < 10:
    print("i =", i)
    i = i + 1
print("Terminé")
```

- La variable `i` est initialisée à 1.
- Puisque la condition (`i < 10`) est **True**, le programme entre dans la boucle et affiche la valeur de `i` (première itération)
- La valeur de `i` s'incrémente par 1 (ligne 4), jusqu'à 10, tant que la condition est **True**
- Lorsque `i` vaut 10 et lors de la vérification de la condition de contrôle, la condition (`i < 10`) vaut **False** et donc la boucle se termine.

## Exemple

Quel est le résultat du code suivant ?

```
sum = 0
i = 1
while i < 10:
    sum = sum + i
    i = i + 1
print(sum)
```

Remarques:

- il faut inclure un code dans la boucle qui permet de **modifier l'état de l'expression de contrôle** (sinon boucle infinie)
- il faut faire attention à l'indentation des instructions dans la boucle

## Un programme plus complexe

On veut écrire un programme qui demande à l'utilisateur de deviner un nombre aléatoire. Ce jeu se poursuit jusqu'à la bonne réponse de l'utilisateur.

```
import random
nombre_a_deviner = random.randint(1,100)
reponse_utilisateur = 0
while not reponse_utilisateur == nombre_a_deviner:
    reponse_utilisateur = int(input("Entrez un nombre"))
    if reponse_utilisateur > nombre_a_deviner:
        print("Trop grand")
    elif reponse_utilisateur < nombre_a_deviner:
        print("Trop petit")
    else:
        print("Bravo")
```

La fonction `random.randint(min, max)` renvoie un entier aléatoire supérieur ou égal à min et inférieur ou égal à max.

# Boucle for ou boucle while

Toute boucle for peut être écrite sous la forme d'une boucle while

Prenons un exemple:

Avec for

```
for i in range(1, 6):
    print("Programmer c'est cool!")
```

Avec while

```
i = 0
while (i < 5):
    i += 1
    print("Programmer c'est cool!")
```

# Choisir entre for et while

## Exemple

```
for a in "hello world":  
    print(a)
```

Que fait ce programme ?

Donner la variante avec while de ce bloc.

## Boucles imbriquées

Le joueur décide à l'avance combien de fois il veut répéter le jeu.

```
import random
nb_jeu = int(input("Combien de fois voulez-vous jouer?"))
for i in range(0, nb_jeu):
    print("Le jeu démarre")
    nombre_a_deviner = random.randint(1,100)
    reponse_utilisateur = 0
    while not reponse_utilisateur == nombre_a_deviner:
        reponse_utilisateur = int(input("Entrez un
nombre"))
        if reponse_utilisateur > nombre_a_deviner:
            print("Trop grand")
        elif reponse_utilisateur < nombre_a_deviner:
            print("Trop petit")
        else:
            print("Bravo")
```

## D'autres outils de contrôle: continue

Le mot-clé `continue` permet de passer prématulement à l'itération suivante de la boucle

### Exemple

```
for i in range(1,21):
    if i % 2 == 0:
        continue
        print("résultat?")
    print(i, "est impair")
print("Terminé")
```

## D'autres outils de contrôle: break

Le mot-clé `break` permet de *casser* (d'arrêter) l'exécution d'une boucle et de passer à l'instruction suivante

### Exemple

```
for i in range(1,21):
    if i % 2 == 0:
        break
    print(i, "est impair")
print("Terminé")
```