



Chapitre 2

Outils de contrôle de flux

Pour les étudiants en génie civil – M1 RIB

Dr. Hachimi Dahhaoui
Université de Tlemcen

mardi 15 octobre 2025



Introduction au contrôle de flux

Dans la vie quotidienne, **on prend des décisions** selon des conditions :

- ❑ S'il pleut, je prends un parapluie.
- ❑ Sinon, je sors sans parapluie.
- ➡ C'est exactement ce que fait un programme.





Introduction au contrôle de flux

Un programme **ne se lit pas toujours de haut en bas** :
il peut **choisir un chemin** différent selon une condition, ou **répéter une action plusieurs fois**.

Ces choix et répétitions sont ce qu'on appelle **le contrôle de flux**.

En Python, il existe trois grandes familles d'instructions de contrôle :

- ☐ Les conditions : **if, elif, else**
- ☐ Les boucles : **for, while**
- ☐ Les interrupteurs dans les boucles : **break, continue, pass, else**

1. L'instruction if

► Principe

Elle permet **d'exécuter un bloc d'instructions uniquement si une condition est vraie.**

python

 Copier le code

```
if condition:
```

```
    # bloc d'instructions
```

>

Exemple concret :

Si la température est supérieure à 30°, on affiche "Il fait chaud".

python

 Copier le code

```
temperature = 35
```

```
if temperature > 30:
```

```
    print("Il fait chaud")
```



Sortie :

nginx

 Copier le code

```
Il fait chaud
```

2. if – else

Quand la condition n'est **pas vraie**, on veut parfois exécuter **autre chose**.

python

 Copier le code

```
temperature = 15

if temperature > 30:
    print("Il fait chaud")
else:
    print("Il fait frais")
```

 Sortie :

nginx

 Copier le code

```
Il fait frais
```



3. if – elif – else

Le mot-clé **elif** signifie “sinon si”

Il permet de **tester plusieurs conditions** l’une après l’autre.

python

 Copier le code

```
temperature = 20

if temperature > 30:
    print("Il fait chaud")
elif temperature >= 20:
    print("La température est agréable")
else:
    print("Il fait froid")
```



Sortie :

nginx



Copier le code

```
La température est agréable
```



4. Les opérateurs logiques et de comparaison

Python permet de **comparer** des valeurs :

Type d'opération	Opérateurs	Exemple	Résultat
Égalité	==	5 == 5	True
Différence	!=	3 != 5	True
Supérieur / inférieur	> < >= <=	x > 10	False
Logiques	and, or, not	(x > 0 and y > 0)	True si les deux sont vrais



4. Les opérateurs logiques et de comparaison

Exemple :

python

 Copier le code

```
age = 25
nationalite = "algérienne"

if age >= 18 and nationalite == "algérienne":
    print("Vous pouvez voter")
```

5. Les boucles for et range()

► La boucle for



Elle permet de répéter une action un certain nombre de fois.

Exemple simple :

python

 Copier le code

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

5. Les boucles for et range()

Sortie :

```
nginx
```

 Copier le code

```
Bonjour 0
```

```
Bonjour 1
```

```
Bonjour 2
```

```
Bonjour 3
```

```
Bonjour 4
```

► Fonction range()

► Fonction `range()`

- `range(n)` → donne les nombres de 0 à n-1
- `range(1, 6)` → donne 1, 2, 3, 4, 5
- `range(0, 10, 2)` → pas de 2 en 2 → 0, 2, 4, 6, 8

5. Les boucles for et range()

Exemple de la vie réelle :

Afficher les 5 premiers jours de la semaine :

python

 Copier le code

```
jours = ["Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi"]

for jour in jours:
    print("Aujourd'hui, on est", jour)
```



6. La boucle while

Elle répète un bloc tant qu'une condition est vraie.

Exemple :

python

 Copier le code

```
compteur = 0
```

```
while compteur < 5:
```

```
    print("Compteur =", compteur)
```

```
    compteur += 1
```



6. La boucle while

Sortie :

```
ini
```

[Copier le code](#)

```
Compteur = 0
```

```
Compteur = 1
```

```
Compteur = 2
```

```
Compteur = 3
```

```
Compteur = 4
```

Différence avec for :

- ☐ for → on connaît le nombre d'itérations à l'avance.
- ☐ while → on répète jusqu'à ce qu'une condition change.

7. Les instructions break, continue, et else dans les boucles

◇ break : arrêter la boucle immédiatement

python

 Copier le code

```
for i in range(10):  
    if i == 5:  
        break  
    print(i)
```

💬 Affiche : 0 1 2 3 4

◇ continue : sauter une itération

python

 Copier le code

```
for i in range(6):  
    if i == 3:  
        continue  
    print(i)
```

💬 Affiche : 0 1 2 4 5

7. Les instructions break, continue, et else dans les boucles

◇ else avec les boucles

Le else s'exécute uniquement si la boucle **n'a pas été interrompue par un break.**

python

 Copier le code

```
for i in range(5):  
    print(i)  
else:  
    print("Boucle terminée sans interruption.")
```


♂ 8. L'instruction pass

Elle ne fait rien. On l'utilise quand Python attend une instruction, mais qu'on ne veut rien faire pour l'instant.

```
python📄 Copier le code  
  
for i in range(3):  
    pass # on écrira le code plus tard
```

Autre exemple dans une condition :

```
python📄 Copier le code  
  
x = 10  
if x > 0:  
    pass # à compléter plus tard  
else:  
    print("Négatif")
```



Conclusion

Élément	Rôle
if, elif, else	Faire des choix
for, while	Répéter des actions
break, continue, pass	Contrôler le déroulement des boucles
range()	Générer des séquences de nombres
else dans boucle	Exécuter un bloc si la boucle se termine normalement

Merci !
Des questions ?

Contact : hachimi.dahhaoui@univ-tlemcen.dz

Fin du Chapitre 1

*“Le meilleur moyen de prévoir l’avenir est de le créer.”
— Peter Drucker*

À très bientôt pour la suite !