



COURS N° 1

Programmation en Python

Introduction générale

Chapitre 1 — Découverte du langage Python

```
print("Bonjour Python !")
```

Présentation du cours

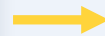


Un parcours complet pour apprendre Python pas à pas : de la découverte du langage jusqu'à la création d'interfaces graphiques interactives.



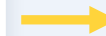
Découvrir

Les bases du langage :
installation, variables, types,
conditions et boucles.



Programmer

Organiser ses données et son
code : chaînes, listes, fonctions,
dictionnaires, fichiers.



Créer

Réaliser des fenêtres et des
formulaires interactifs avec
tkinter.

Sommaire du cours



1

Vous êtes ici



Découverte du langage Python

Installation, variables, types, conditions et boucles

2



Les chaînes de caractères

Indexation, découpage, méthodes et formatage

3



Les listes et les tuples

Stocker et organiser plusieurs valeurs

4



Les fonctions

Structurer, réutiliser et organiser le code

5



Les dictionnaires

Associer des clés à des valeurs

6



Les fichiers

Lire et écrire des données de façon permanente

7



Interfaces graphiques (tkinter)

Fenêtres et formulaires interactifs

CHAPITRE 1

Découverte du langage Python

Bases du langage et premières interactions



Installer

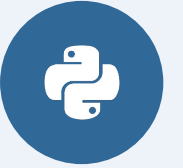
Variables

Types

Conditions

Boucles

1. Qu'est-ce que Python ?



Un langage interprété

Le code est exécuté directement, ligne par ligne, sans phase de compilation préalable.



Simple et lisible

Une syntaxe claire et intuitive, proche du langage naturel.



Puissant et très répandu

L'un des langages les plus utilisés au monde.



1991

Année de création

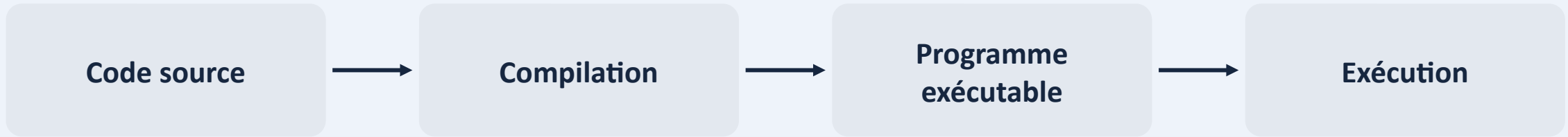
Guido van Rossum

Créateur de Python

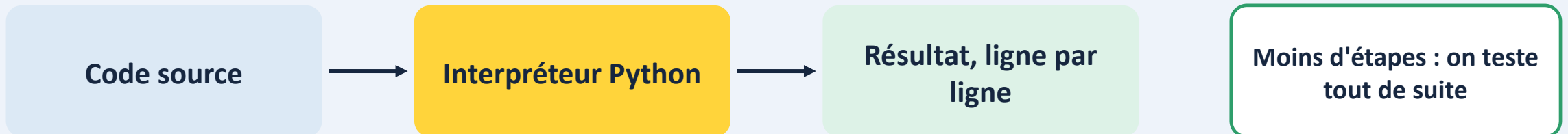
Interprété ou compilé ?



Langage compilé (exemple : C) : le programme est d'abord traduit



Python (interprété) : le code est lu et exécuté directement



On écrit, on exécute, on corrige : le développement est plus rapide.

Les atouts de Python



Syntaxe claire et intuitive

Un code facile à lire et à écrire, avec une structure proche du langage naturel.



Exécution directe

Le code s'exécute immédiatement : le développement et les tests sont plus rapides.



Mémoire gérée automatiquement

Python alloue et libère lui-même la mémoire : le programmeur n'a pas à s'en occuper.



Bibliothèques et modules riches

NumPy, Pandas, Matplotlib, TensorFlow, Django, Flask...



Communauté active

Documentation, tutoriels et forums d'entraide très nombreux.

Pourquoi apprendre Python ?



Apprendre la logique algorithmique

Raisonner étape par étape pour résoudre un problème.



Programmer rapidement

Développer très vite des programmes simples.



Se préparer au monde professionnel

Python ouvre la porte à des domaines variés.



Servir dans plusieurs disciplines

Informatique, sciences, automatisation, statistiques.

2. Domaines d'application



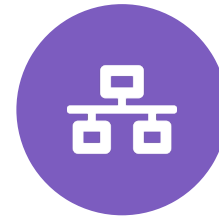
Développement informatique

- Applications de bureau
- Scripts d'automatisation
- Développement web (Django, Flask)



Sciences et données

- Analyse de données
- Intelligence artificielle
- Machine learning
- Data science



Réseaux et systèmes

- Automatisation de tâches
- Administration système
- Sécurité informatique



Éducation

- Enseignement de l'algorithmique
- Projets pédagogiques
- Simulations

3.1 Installer Python



1



Ouvrir python.org

Se rendre sur le site officiel de Python, rubrique Downloads.

2



Télécharger

Cliquer sur Downloads, puis sur Download Python 3.x.x.

3

IMPORTANT



Add Python to PATH

Cocher impérativement la case « Add Python to PATH ».

4



Installer

Cliquer sur Install Now.

TERMINAL

```
> python --version  
Python 3.x.x
```



Vérifier l'installation

Tapez cette commande dans l'invite de commande (CMD) ou dans le terminal : elle affiche le numéro de version installé.

3.2 Installer Visual Studio Code



1

Télécharger la version correspondant à son système d'exploitation.

2

Lancer l'installation en laissant les options par défaut.

3

Dans l'onglet Extensions, rechercher « Python » et installer l'extension officielle de Microsoft.

4

Créer un fichier test.py, puis choisir l'interpréteur Python en bas à droite de la fenêtre.



Un éditeur de code léger

très utilisé pour Python. Téléchargement :
code.visualstudio.com/Download



test.py

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

Python 3.x



4. Les variables en Python



Une variable est un espace mémoire qui permet de stocker une valeur.



Typage dynamique

Inutile de déclarer le type à l'avance : Python le détermine selon la valeur affectée.

```
age = 20
```

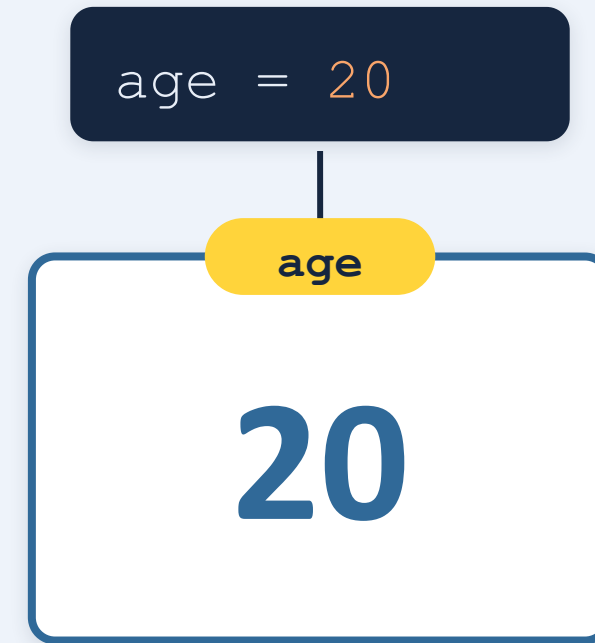
Python devine : **int** (entier)

```
prix = 12.5
```

Python devine : **float** (décimal)

```
nom = "Ali"
```

Python devine : **str** (texte)



5. Les types de données



int

Entier

Les nombres entiers

10

2026



float

Décimal

Les nombres à virgule

12.5

14.75



bool

Booléen

Vrai ou faux

True

False



str

Texte

Les chaînes de caractères

"Ali"

'Bonjour'

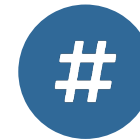
5.1 Les types int et float



PYTHON

```
# 5.1 Le type int (entier)
nombre = 10
annee = 2026

# 5.2 Le type float (nombre décimal)
prix = 12.5
moyenne = 14.75
```



int : nombres entiers

Exemples : 10, 2026



float : nombres décimaux

Exemples : 12.5, 14.75



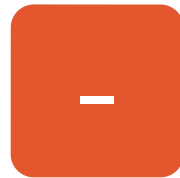
En Python, le séparateur décimal est le point (12.5), et non la virgule.

5.3 Les opérateurs mathématiques



Addition

$7 + 2 \rightarrow 9$



Soustraction

$7 - 2 \rightarrow 5$



Multiplication

$7 * 2 \rightarrow 14$



Division
classique
(résultat décimal)

$7 / 2 \rightarrow 3.5$



Division
entière
(partie entière)

$7 // 2 \rightarrow 3$



Modulo
(reste de la
division)

$7 \% 2 \rightarrow 1$



Puissance

$7 ** 2 \rightarrow 49$



Astuce

$n \% 2 == 0$ signifie que n est pair : le modulo sert à tester la parité.

5.4 Les types bool et str



Le type bool

Une valeur logique : True (vrai) ou False (faux).

True

False

```
est_valide = True  
est_absent = False
```



True et False s'écrivent avec une majuscule.



Le type str

Du texte (chaîne de caractères), entre guillemets.

"Ali"

'Bonjour'

```
nom = "Ali"  
message = 'Bonjour'
```



Guillemets simples ou doubles : le résultat est le même.

5.5 Concaténation et f-string



+

Concaténation : coller des textes avec +

```
prenom = "Ali"  
texte = "Bonjour " + prenom
```

TEXTE VAUT

Bonjour Ali

f

f-string : insérer des variables avec { }

```
age = 20  
message = f"Je m'appelle {prenom} et j'ai {age} ans"
```

MESSAGE VAUT

Je m'appelle Ali et j'ai
20 ans

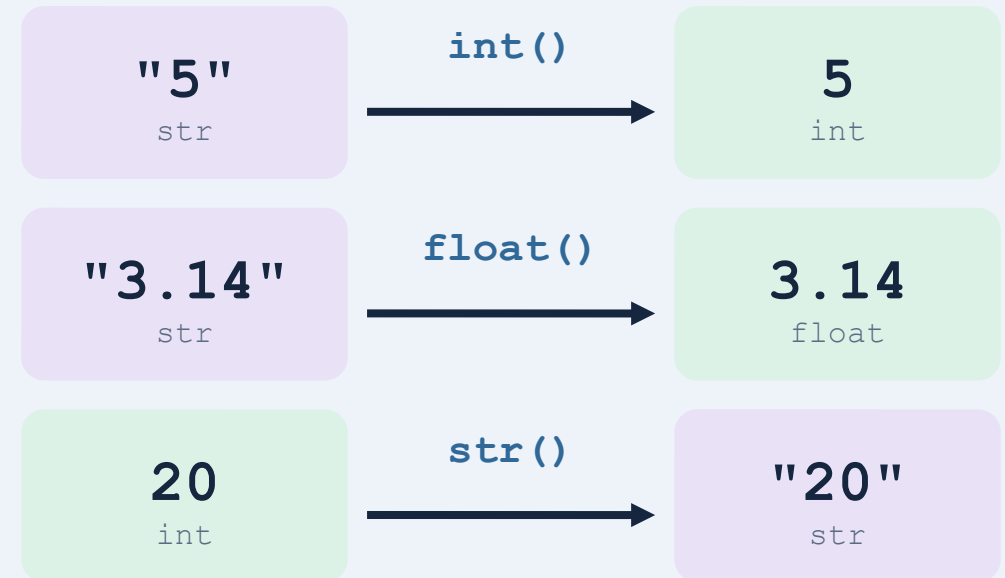
6. La conversion des types



PYTHON

```
x = int("5")           # conversion en entier
y = float("3.14")      # conversion en flottant

age = 20
texte = str(age)       # conversion en chaîne
```



Avec les f-strings, c'est plus simple

Inutile de convertir un nombre en texte avant de l'afficher.

```
note = 15
message = f"La note est {note}"
```

7.1 Afficher avec print()



PYTHON

```
print("Bonjour tout le monde")  
  
age = 20  
print(age)  
print("Age :", age)  
print(f"Votre âge est {age}")
```



RÉSULTAT

```
Bonjour tout le monde  
20  
Age : 20  
Votre âge est 20
```



print() affiche un texte ou la valeur d'une variable.

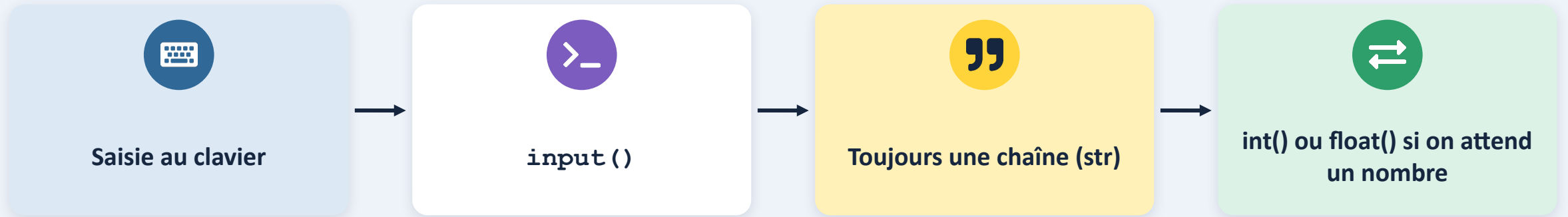


Plusieurs valeurs séparées par une virgule sont affichées avec un espace.



La f-string mélange texte et variables grâce aux accolades { }.

7.2 Saisir avec input()



PYTHON

```
nom = input("Entrez votre nom : ")
age = int(input("Entrez votre âge : "))
note = float(input("Entrez votre note : "))

print(f"{nom} a {age} ans et une note de {note}")
```



Attention

input() retourne toujours du texte, même si l'on tape 20. Il faut convertir avec int() ou float() avant de calculer.



8.1 La condition simple if

PYTHON

```
age = 20

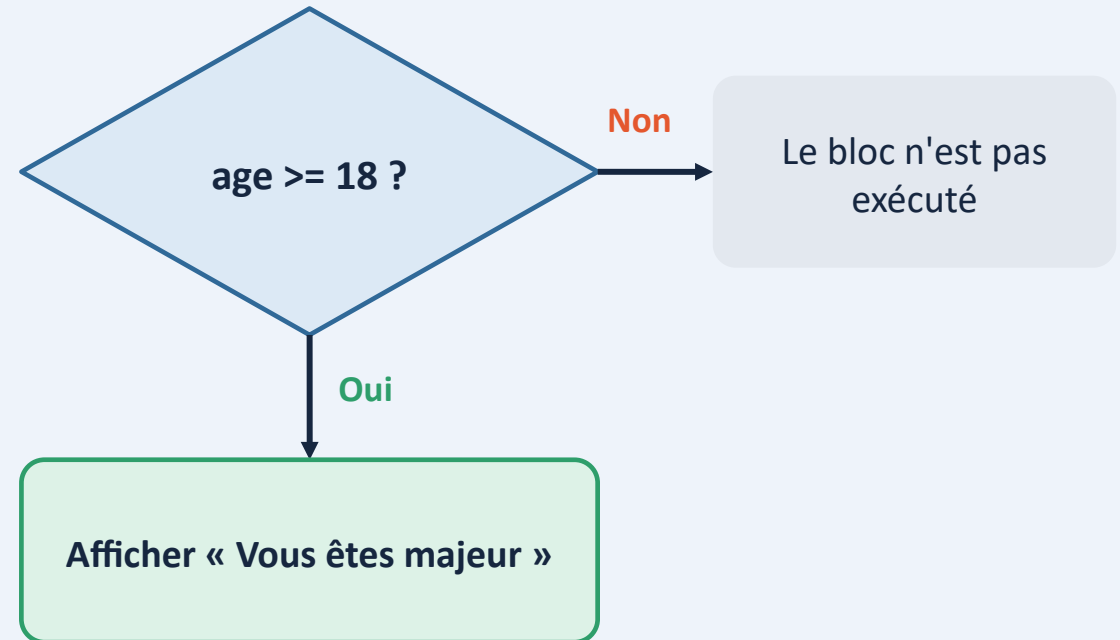
if age >= 18:
    print("Vous êtes majeur")
```



L'indentation compte !

Le bloc à exécuter est décalé de 4 espaces après les deux-points (:).

age = 20



8.2 La condition if ... else



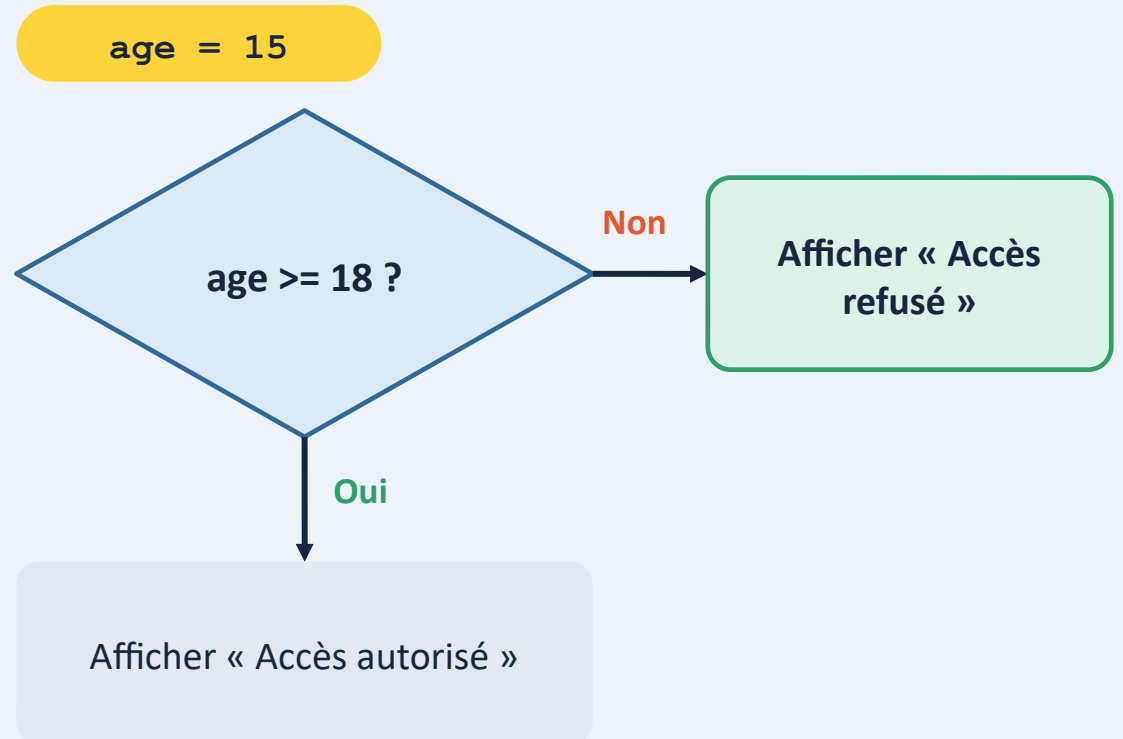
PYTHON

```
age = 15

if age >= 18:
    print("Accès autorisé")
else:
    print("Accès refusé")
```



else regroupe tous les autres cas : un seul des deux blocs est exécuté.



8.3 La condition if ... elif ... else

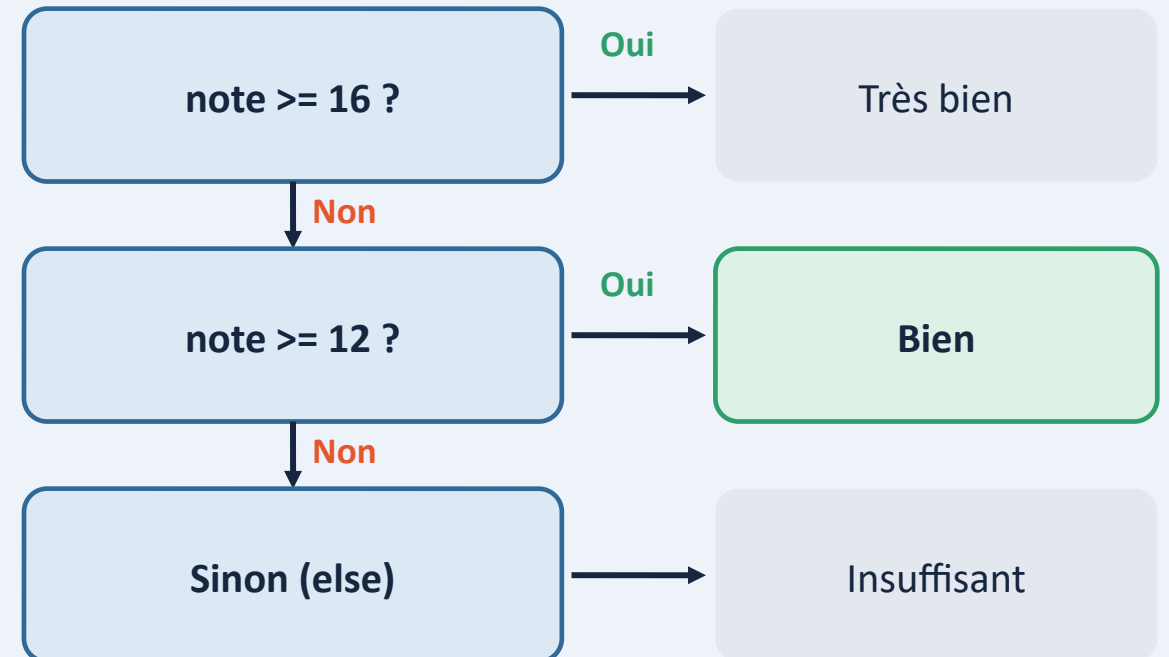


PYTHON

```
note = 14

if note >= 16:
    print("Très bien")
elif note >= 12:
    print("Bien")
else:
    print("Insuffisant")
```

note = 14



On peut enchaîner autant de elif que nécessaire : dès qu'une condition est vraie, les suivantes sont ignorées.

8.4 Les connecteurs and et or



ET

and : toutes les conditions doivent être vraies

```
age = 20
note = 12

if age >= 18 and note >= 10:
    print("Admis")
else:
    print("Non admis")
```

Vrai et **Vrai** → Admis

OU

or : au moins une condition doit être vraie

```
jour = "samedi"

if jour == "samedi" or jour == "dimanche":
    print("C'est le week-end")
```

Vrai ou **Faux** → C'est le week-end



Python est sensible à la casse : « casablanca » est différent de « Casablanca » lors d'une comparaison de chaînes.

8.5 L'instruction match case



PYTHON

```
jour = "lundi"

match jour:
    case "lundi":
        print("Jour 1")
    case "vendredi":
        print("Jour 5")
    case "samedi" | "dimanche":
        print("Week-end")
    case _:
        print("Jour inconnu")
```



Depuis Python 3.10

Il faut une version 3.10 ou plus récente.



Alternative aux longs if ... elif

Plus lisible ; comparable au switch ... case d'autres langages.



case "samedi" | "dimanche"

Le symbole | signifie « ou » : plusieurs valeurs pour un même cas.



case _

Cas par défaut : exécuté si aucune valeur ne correspond.

9. Les boucles



Un nombre précis de répétitions

ou en parcourant les éléments d'une collection (liste, chaîne...).

```
for i in range(8):
```



Tant qu'une condition est vraie

sans connaître à l'avance le nombre de répétitions.

```
while i < 5:
```



Les boucles répètent automatiquement des instructions, sans avoir à les réécrire.

9.1 La boucle for



PYTHON

```
for i in range(8):  
    print(i)
```



La **boucle for** répète un bloc de code un nombre précis de fois, ou en parcourant les éléments d'une collection.

i prend successivement les valeurs :

0

tour 1

1

tour 2

2

tour 3

3

tour 4

4

tour 5

5

tour 6

6

tour 7

7

tour 8



Le bloc indenté est exécuté 8 fois : de $i = 0$ jusqu'à $i = 7$ (la valeur 8 est exclue).

9.1 Les trois formes de range()



La valeur d'arrêt n'est jamais atteinte : range(5) s'arrête à 4.

range(5)

de 0 à 4

0

1

2

3

4

range(3, 8)

de 3 à 7

3

4

5

6

7

range(2, 10, 2)

2, 4, 6, 8 (pas de 2)

2

4

6

8

9.1 Parcourir une liste ou une chaîne



```
produits = ["livre", "pc", "disque dur"]  
for produit in produits:  
    print(produit)
```

produit prend successivement :

livre

tour 1

pc

tour 2

disque dur

tour 3

```
mot = "Python"  
for lettre in mot:  
    print(lettre)
```

lettre prend successivement :

P

tour 1

y

tour 2

t

tour 3

h

tour 4

o

tour 5

n

tour 6

9.1 Exemple : somme des impairs



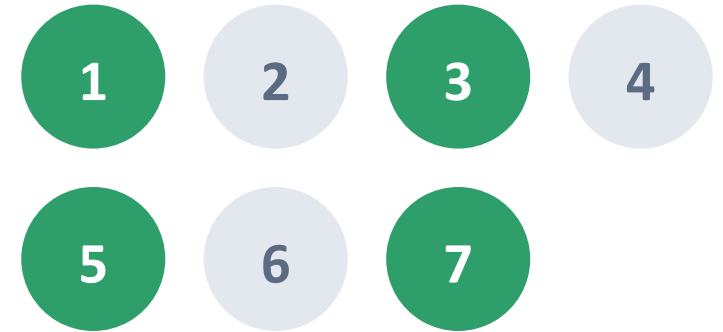
PYTHON

```
nombres = [1, 2, 3, 4, 5, 6, 7]
somme_impairs = 0

for n in nombres:
    if n % 2 != 0:
        somme_impairs += n

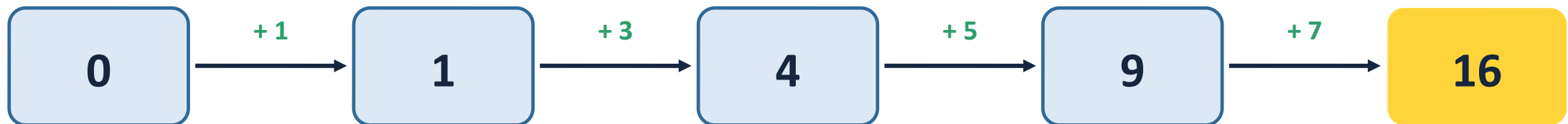
print("Somme des nombres impairs :", somme_impairs)
```

Parcours de la liste



Vert : impair, ajouté à la somme
Gris : pair, ignoré

somme_impairs au fil des tours :



9.2 La boucle while



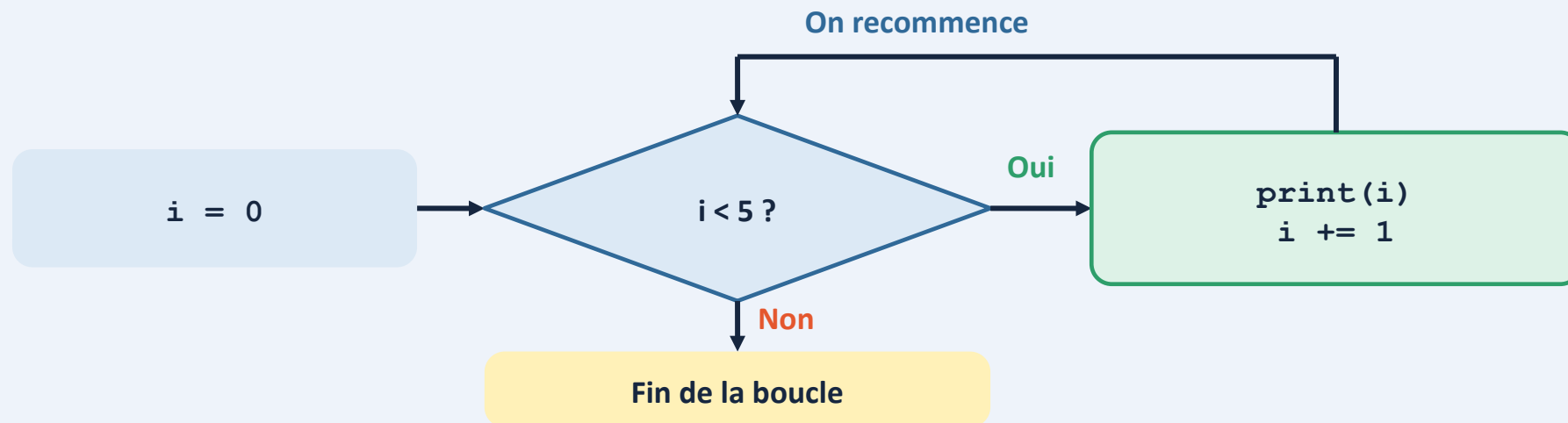
PYTHON

```
i = 0
while i < 5:
    print(i)
    i += 1
```

RÉSULTAT

```
0
1
2
3
4
```

La boucle **while** répète un bloc tant qu'une condition est vraie, sans connaître à l'avance le nombre de répétitions.



9.2 Exemple : saisie contrôlée



La boucle while est particulièrement utile pour répéter une saisie jusqu'à ce qu'elle soit correcte.

PYTHON

```
mot_de_passe = ""

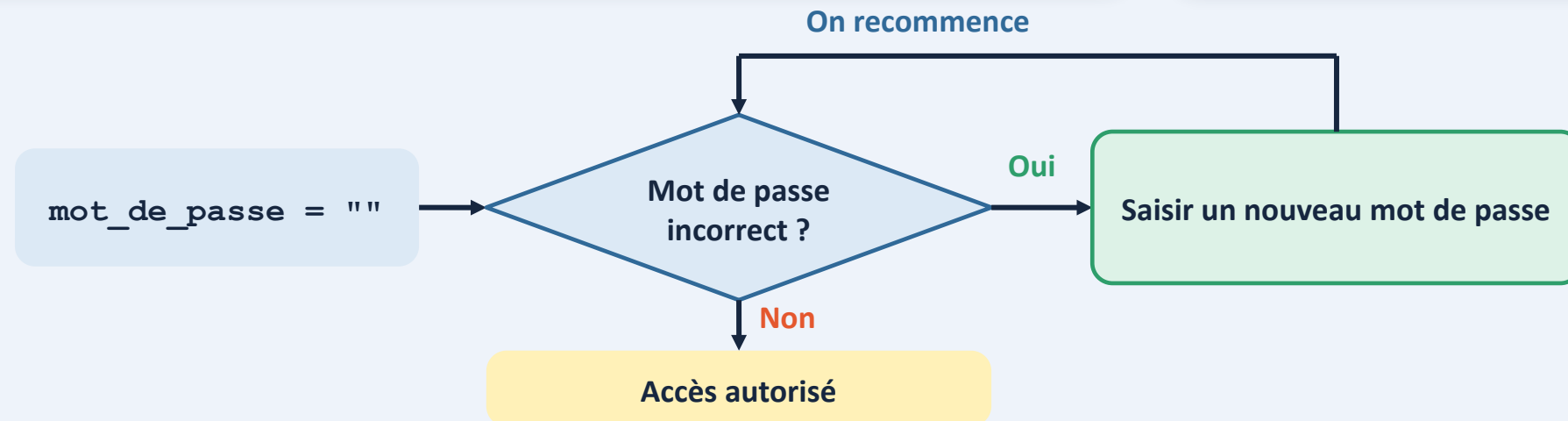
while mot_de_passe != "python123":
    mot_de_passe = input("Entrez le mot de passe : ")

print("Accès autorisé")
```



Boucle infinie ?

Prévoir toujours une condition d'arrêt atteignable, en modifiant la variable testée à chaque tour.

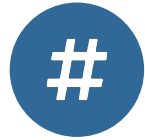


À retenir



Variables

Typage dynamique : Python devine le type selon la valeur.



Types de base

int, float, bool et str, avec les opérateurs
+ - * // % **.



Conversion

int(), float(), str() ; input() renvoie toujours du texte.



Affichage et saisie

print() affiche, input() lit le clavier, les f-strings mélangent texte et variables.



Conditions

if, elif, else, and / or et match case pour décider.



Boucles

for pour compter ou parcourir, while tant que la condition est vraie.