Python

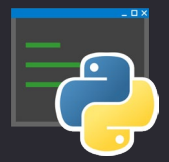
DEVELOPPER AVEC UN LANGAGE PROCEDURAL

Objectifs

- Comprendre les bases de la programmation procédurale (variables, boucles, conditions, fonctions).
- Utiliser un environnement de développement pour écrire et déboguer du code.
- Compiler et exécuter un programme Maitriser les outils de gestion de version

Environnement.






IDE

Interpreter Python

Documentation

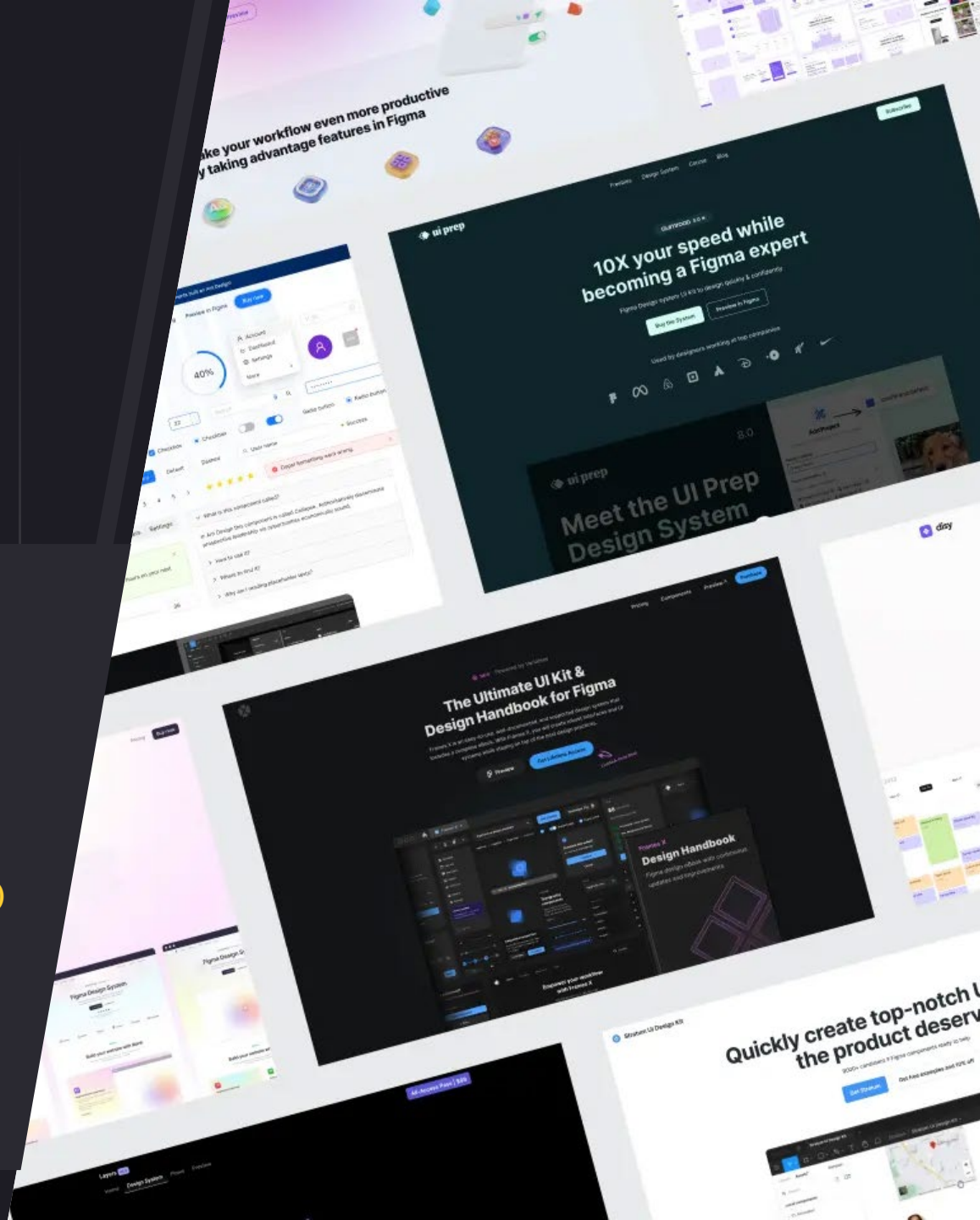
1. Python est un langage interprété

Inclus des gestionnaires de paquets (pour installer des bibliothèques)

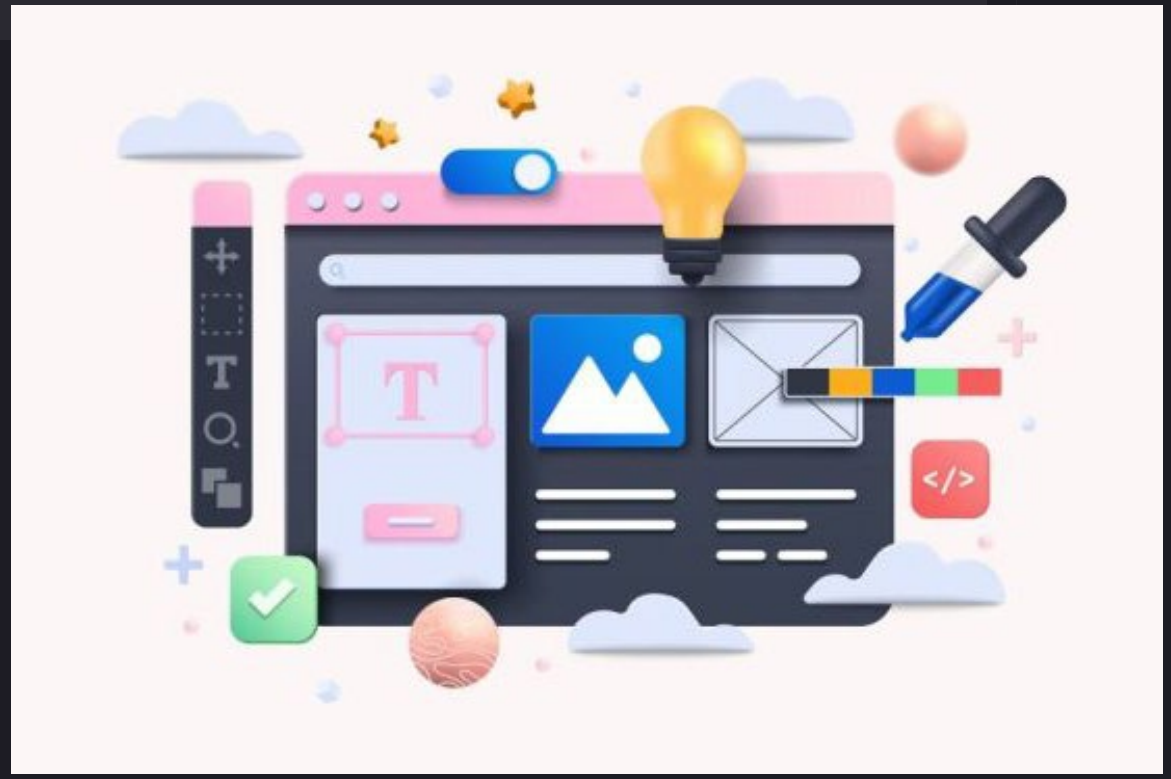
Pas d'interpréteur Python = pas de Python exécuté.

C'est lui qui traduit code Python en instructions que l'ordinateur comprend.

Sur Windows cocher la case « PATH »



Généralités



Commentaires avec le #

Print() pour afficher dans la console

Necessité de Run le code

```python
# Variable = A container for a value
#            A variable behaves as if

# Strings
first_name = "Bro"
food = "pizza"
email = "Bro123@fake.com"

# Integers
age = 25
quantity = 3
num_of_students = 30

# Float
price = 10.99
gpa = 3.2
distance = 5.5

# Boolean
is_student = True
for_sale = False
```

Déclarations variables

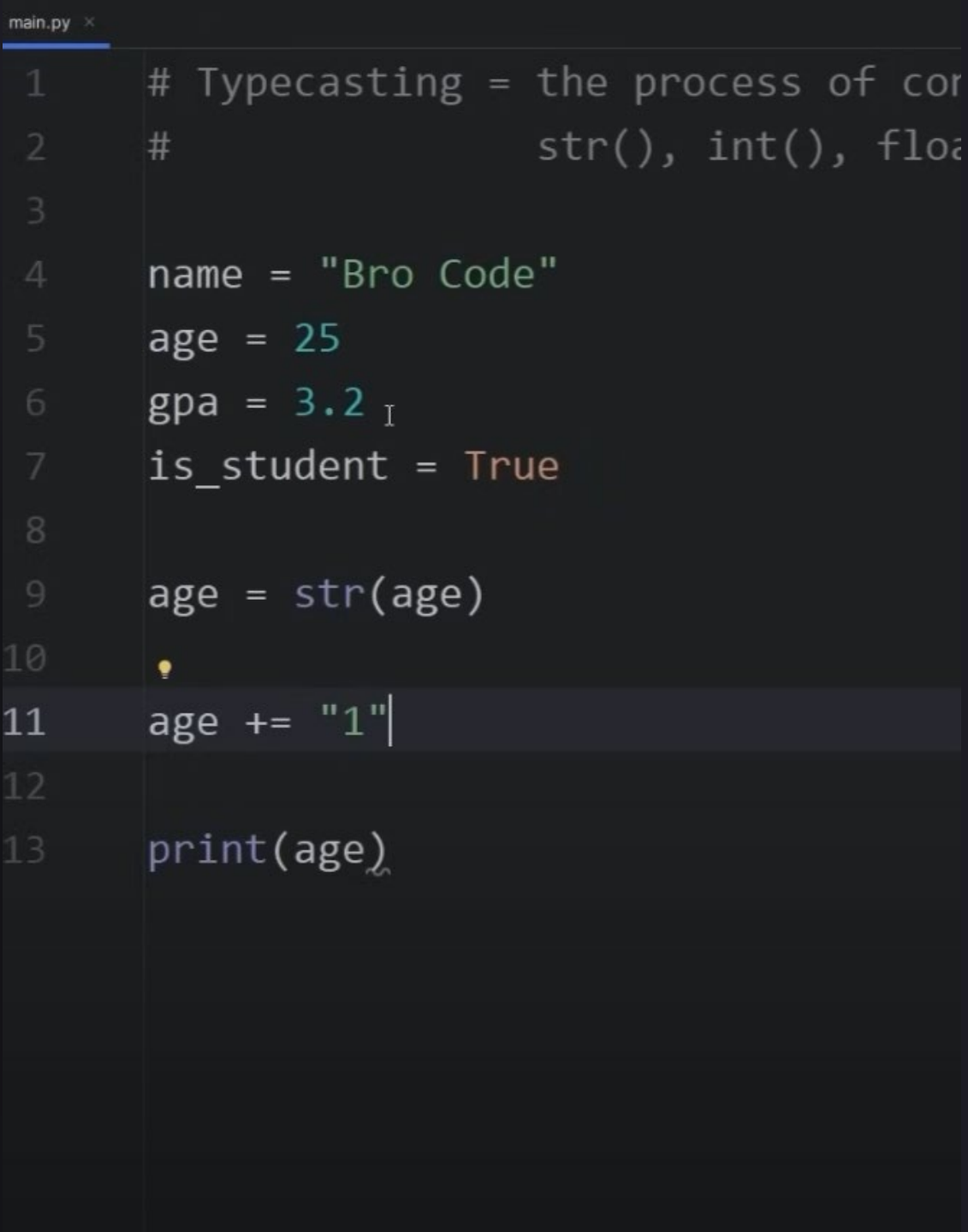
En Python, pas besoin de déclarer le type : le langage est dynamique.

📌 Règles de nommage

- Commence par une lettre ou _
- Pas d'espace, pas de caractères spéciaux
- Sensible à la casse (Age ≠ age)

Il est possible de faire des affectations multiples

```python
a, b, c = 1, 2, 3
x = y = 0  # Même valeur
```



TypeCasting

Le **typecasting** permet de convertir une variable d'un type à un autre.

Pour verifier le type d'une variable on utilisera **type(maVar)**

int()	int("10")	→ 10 (entier)
float	float("3.14")	→ 3.14 (flottant)
str()	str(25)	→ "25" (chaîne)
bool()	bool(0), bool("salut")	→ False, True

```python
# input() = A function that promp
#           Returns the entered d

name = input("What is your name?:
age = int(input("How old are you?

age = age + 1

print(f"Hello {name}!")
print("HAPPY BIRTHDAY!")
print(f"You are {age} years old")
```

Input

Pour demander à l'utilisateur **de saisir des informations** depuis le clavier.

- **input**() renvoie toujours **une chaîne de caractères (str).**
- Utiliser int() ou float() si besoin de conversion.

⚠️ Attention

```python
age = int(input("Ton âge ?"))
```

Si l'utilisateur tape du texte non numérique ❌ Erreur !

Exercice 1

```
/*
    1. Ecrire un programme qui demande la longueur puis la largeur d'un rectangle et calcule l'aire de ce rectangle puis l'affiche
*/


/*
    2. Ecrire un programme qui demande quel article acheter, son prix, puis la quantité. On affiche ensuite une phrase recapîtualitve du style « vous avez acheté 9 pizzas pour un total de 999€ »
*/
```

Correction 1

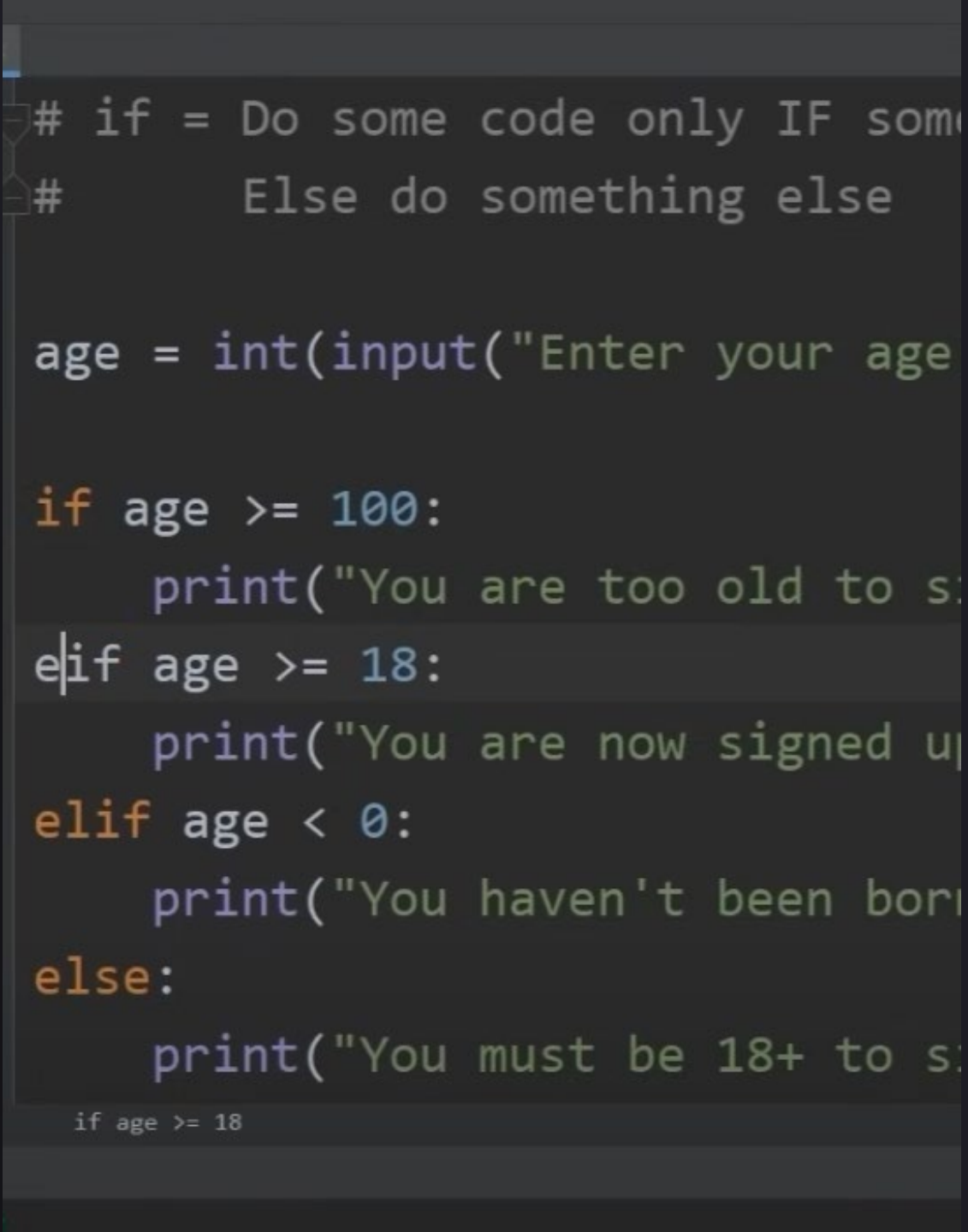
```python
# Exercise 1 Rectangle Area Calc

length = float(input("Enter the length: "))
width = float(input("Enter the width: "))
area = length * width

print(f"The area is: {area}cm²")
```

```python
# Exercise 2 Shopping Cart Program

item = input("What item would you like to buy?: ")
price = float(input("What is the price?: "))
quantity = int(input("How many would you like?: "))
total = price * quantity

print(f"You have bought {quantity} x {item}/s")
print(f"Your total is: ${total}")
```

```
# friends = friends + 1
# friends += 1
# friends = friends - 2
# friends -= 2
# friends = friends * 3
# friends *= 3
# friends = friends / 2
# friends /= 2
# friends = friends ** 2
# friends **= 2
# remainder = friends % 2
```

```
x = 3.14
y = 4
z = 5

# result = round(x)
# result = abs(y)
# result = pow(4, 3)
# result = max(x, y, z)
# result = min(x, y, z)

print(result)
```

```
import math

x = 9.9

# print(math.pi)
# print(math.e)
# result = math.sqrt(x)
# result = math.ceil(x)
result = math.floor(x)

print(result)
```



Condition if / else

Pour exécuter différents blocs de code selon des conditions logiques.

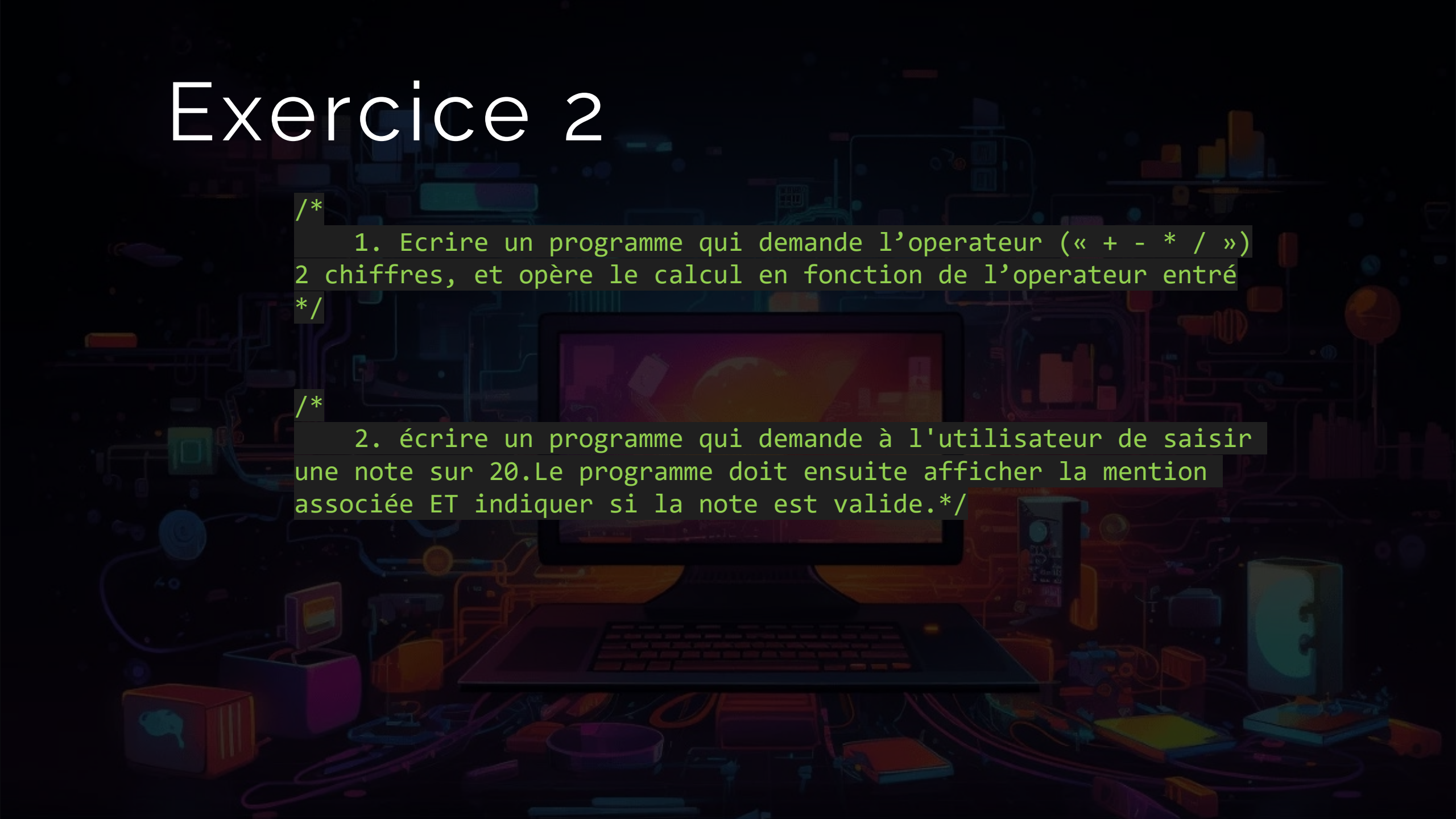
- **if :** teste une première condition.
- **elif :** (optionnel) teste d'autres conditions si la précédente est fausse
- **.else :** (optionnel) s'exécute si aucune condition n'est remplie.

⚠️ **Attention à l'indetation**
✅ Indenter le code (généralement avec **4 espaces** ou **Tab**)

Exercice 2

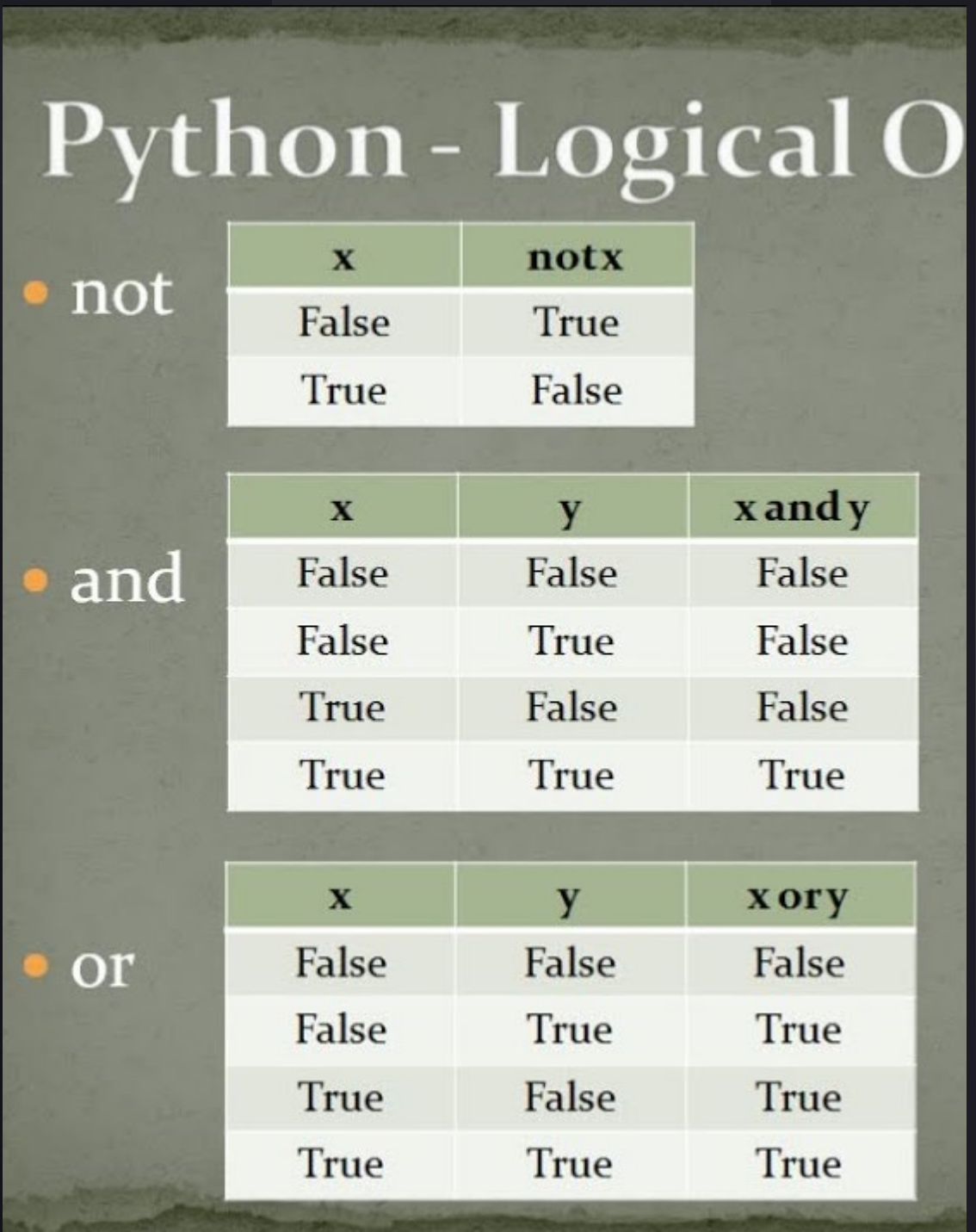
```
/*
    1. Ecrire un programme qui demande l’operateur (« + - * / ») 2 chiffres, et opère le calcul en fonction de l’operateur entré
*/
```

```
/*
    2. écrire un programme qui demande à l'utilisateur de saisir une note sur 20.Le programme doit ensuite afficher la mention associée ET indiquer si la note est valide.*/
```



Correction 2

```python
# Demander la note à l'utilisateur
note = float(input("Entrez votre note sur 20 : "))

# Vérification de la validité de la note
if note < 0 or note > 20:
    print("Erreur : note invalide (doit être entre 0 et 20)")
else:
    # Détermination de la mention selon la note
    if note >= 16:
        print("Mention : Très bien")
    elif note >= 14:
        print("Mention : Bien")
    elif note >= 12:
        print("Mention : Assez bien")
    elif note >= 10:
        print("Mention : Passable")
    else:
        print("Mention : Insuffisant")
```



Operateurs logiques

Les opérateurs logiques permettent de combiner ou inverser des conditions.

and	**Vrai si les 2 conditions sont vraies**
or	**Vrai si au moins une condition est vraie**
not	**Inverse la valeur d'une condition**

⚠️ **Eviter les if imbriqués**

✅ Utilise les opérateurs logiques pour simplifier les conditions longues au lieu de les imbriquer.

```python
age = 13
temperature = 20
user_role = "admin"

# print("Positive" if num > 0 else "Nega
# result = "EVEN" if num % 2 == 0 else "
# max_num = a if a > b else b
# min_num = a if a < b else b
# status = "Adult" if age >= 18 else "Ch
# weather = "HOT" if temperature > 20 el
access_level = "Full Access" if user_rol

print(access_level)
```

Condition ternaire

Permet d'écrire une **condition simple en une seule ligne.**

résultat = valeur_si_vrai **if condition else** valeur_si_faux

C'est une alternative plus compacte à if/else.Pratique pour les affectations rapides ou les affichages simples.

⚠️ Ne pas l'utiliser pour des conditions complexes

Methode string

Méthode	Description	Exemple
`len()`	Longueur de la chaîne	`len("Python")` → `6`
`find(s)`	Position de la **1ère occurrence** de `s`	`"bonjour".find("o")` → `1`
`rfind(s)`	Position de la **dernière occurrence** de `s`	`"bonjour".rfind("o")` → `4`
`capitalize()`	Met la **1re lettre en majuscule**	`"python".capitalize()` → `"Python"`
`upper()`	Transforme en **majuscules**	`"hello".upper()` → `"HELLO"`
`lower()`	Transforme en **minuscules**	`"HELLO".lower()` → `"hello"`
`isdigit()`	Vérifie si la chaîne ne contient que des **chiffres**	`"123".isdigit()` → `True`
`isalpha()`	Vérifie si la chaîne ne contient que des **lettres**	`"abc".isalpha()` → `True`
`count(s)`	Compte le nombre d'occurrences de `s`	`"salut".count("s")` → `1`
`replace(a, b)`	Remplace `a` par `b` dans la chaîne	`"pomme".replace("m", "n")` → `"ponne"`

Exercice 3

```
/*
    1. Ecrire un programme qui valide l’entrée d’un nom utilisateur. Il doit contenir moins de 12 caractères, pas d’espaces et pas de chiffres.
*/
```

Correction ex3

```
username = input("Enter a username: ")

if len(username) > 12:
    print("Your username can't be more than 12 characters")
elif not username.find(" ") == -1:
    print("Your username can't contain spaces")
elif not username.isalpha():
    print("Your username can't contain numbers")
else:
    print(f"Welcome {username}")
```

```python
# indexing = accessing elements of a
#            [start : end : step]

credit_number = "1234-5678-9012-3456"

last_digits = credit_number[-4:]
print(f"XXXX-XXXX-XXXX-{last_digits}"
```

Syntaxe	Résultat
`texte[0:4]`	`"Pyth"`
`texte[::2]`	`"Pto"` (1 sur 2)
`texte[1:]`	`"ython"` (à partir de l'index 1)
`texte[:3]`	`"Pyt"` (jusqu'à l'index 2 inclus)
`texte[::-1]`	`"nohtyP"` (chaîne inversée)

Indexation String

maVar **[début : fin : step]**

Les index commencent à 0

❌ Une erreur se produit si l'index dépasse la taille de la chaîne

- texte[a:b] inclut l'indice a, mais exclut b
- L'indexation négative permet de partir de la fin
- Le step (pas) est optionnel et sert à sauter des caractères

```python
# indexing = accessing elements of a
#           [start : end : step]

credit_number = "1234-5678-9012-3456"

last_digits = credit_number[-4:]
print(f"XXXX-XXXX-XXXX-{last_digits}"
```

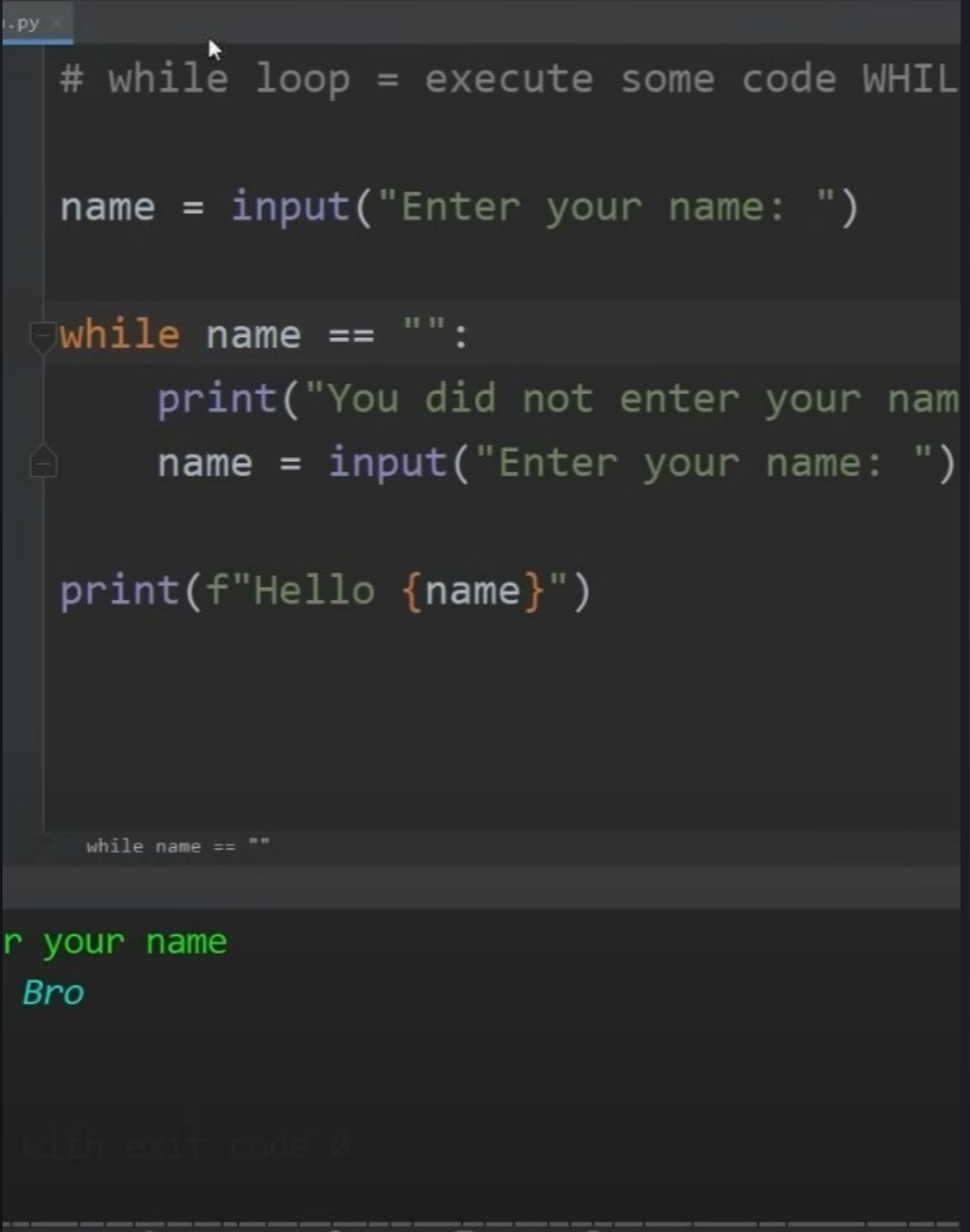
Syntaxe	Résultat
`texte[0:4]`	`"Pyth"`
`texte[::2]`	`"Pto"` (1 sur 2)
`texte[1:]`	`"ython"` (à partir de l'index 1)
`texte[:3]`	`"Pyt"` (jusqu'à l'index 2 inclus)
`texte[::-1]`	`"nohtyP"` (chaîne inversée)

formatage (flags)

Pour afficher proprement du texte, des nombres ou des variables avec un format lisible et contrôlé.

Syntaxe	Description	Exemple de sortie
f"{val:5}"	Largeur fixe (aligné à droite)	' 42'
f"{val:<5}"	Aligné à gauche	'42 '
f"{val:^5}"	Centré	' 42 '
f"{val:05}"	Zéros en padding	'00042'
f"{val:.2f}"	2 décimales (nombre flottant)	'3.14'
f"{pourcentage:.1%}"	Format pourcentage	0.25 → '25.0%'

- Le formatage f-string est lisible, moderne, et très utilisé.
- Combine les {} avec des options de mise en forme : alignement, décimales, zéros...



boucle while

Une boucle qui répète un bloc de code tant qu'une condition est vraie.

while condition:
    # bloc de code à répéter

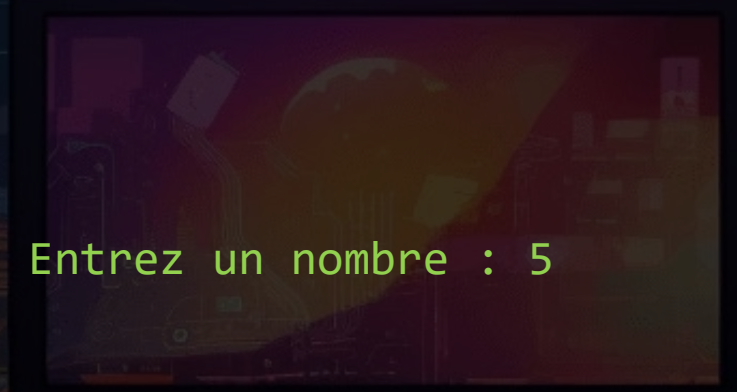
La condition est testée avant chaque itération.Si elle est fausse dès le départ, le code ne s'exécute pas.

⚠️ Attention aux boucles infinies

Attente d'une bonne saisie utilisateur
Répétition d'un calcul jusqu'à une condition atteinte

Exercice 4

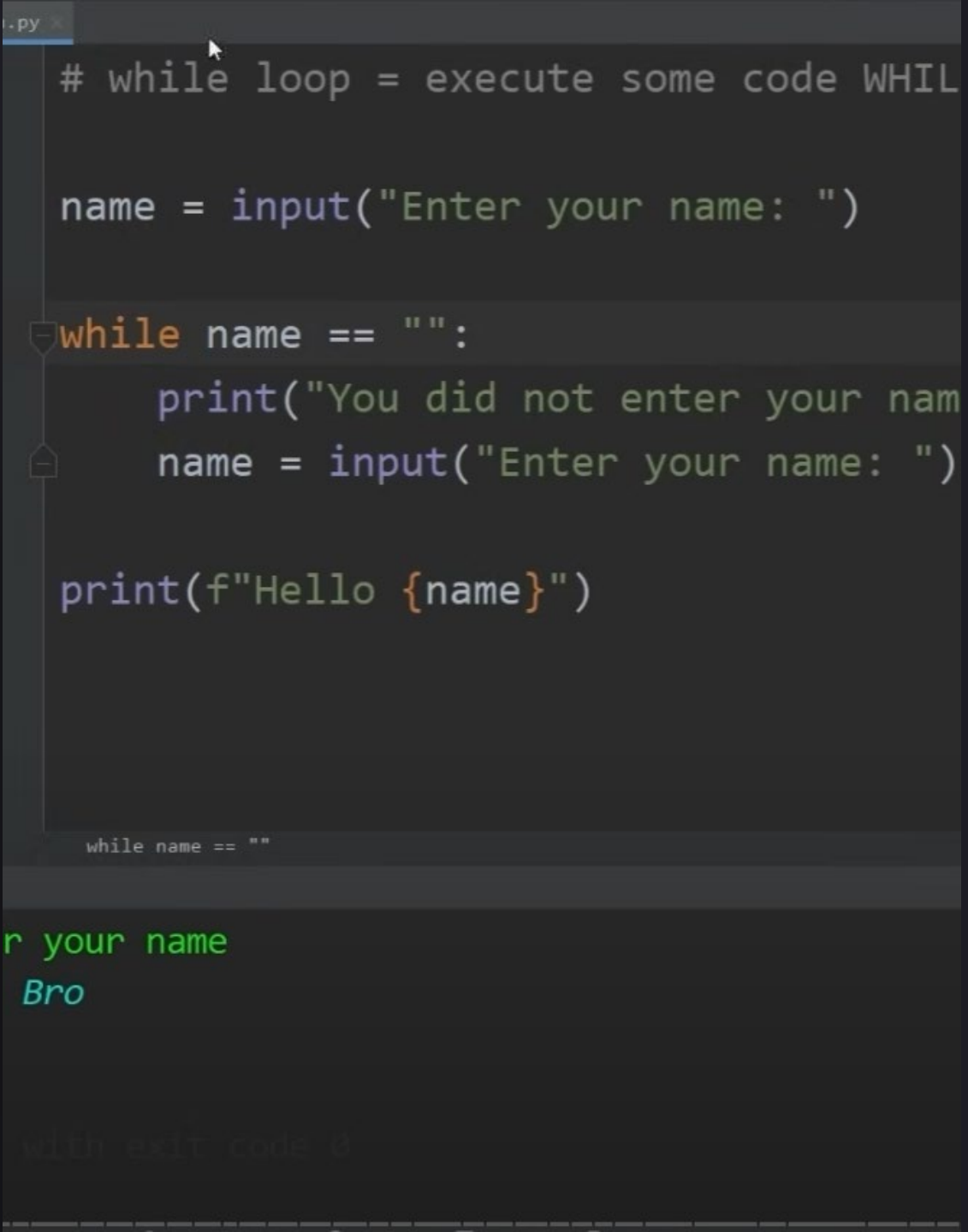
```
/*
    1. Créer un petit jeu où l'utilisateur doit deviner un nombre secret.Le programme utilise une boucle while pour répéter les tentatives, et des f-strings avec flags pour formater les messages.
*/
```



```
Tentative n°01 : Entrez un nombre : 5
Trop bas !

Tentative n°02 : Entrez un nombre : 10
Trop haut !

Tentative n°03 : Entrez un nombre : 7
Bravo ! Vous avez trouvé en 03 tentatives.
```

Correction ex4

```python
secret = 7
essai = 0
trouve = False

while not trouve:
    essai += 1
    guess = int(input(f"Tentative n°{essai:02} : Entrez un nombre : "))

    if guess < secret:
        print("Trop bas !")
    elif guess > secret:
        print("Trop haut !")
    else:
        print(f"Bravo ! Vous avez trouvé en {essai:02} tentatives.")
        trouve = True
```



boucle for

Une boucle qui permet de parcourir une séquence (liste, chaîne, plage de nombres, etc.)

for variable in séquence:
    # bloc de code à répéter

À chaque tour, la variable prend la valeur suivante de la séquence.

range(x) génère les nombres de 0 à x-1

Syntaxe	**Résultat généré**
range(n)	de 0 à n-1
range(a, b)	de a à b-1
range(a, b, step)	de a à b-1, par palier de step