UAA2 Programmation Python

Table des matières

Introduction à la programmation

Qu'est-ce qu'un programme ?

Un **programme informatique** est une suite d'instructions que l'on donne à un ordinateur pour qu'il exécute une tâche précise.
➡️ On peut comparer un programme à une **recette de cuisine** : chaque étape doit être suivie dans le bon ordre pour obtenir le résultat attendu.

Exemples simples de programmes :

- Une calculatrice qui additionne deux nombres.
- Une application météo qui affiche la température du jour.
- Un robot qui avance, tourne, s'arrête selon des commandes.

Exemples concrets dans la vie quotidienne

Les programmes sont partout autour de nous :

- **Smartphone** : chaque application (WhatsApp, TikTok, Spotify) est un programme.
- **Jeux vidéo** : ils sont constitués de milliers de lignes de code.
- **Sites web** : comme YouTube ou Amazon, ce sont des programmes accessibles via internet.
- **Robotique** : un aspirateur robot ou un bras mécanique suivent un programme pour agir.

Pourquoi Python et quels outils utiliser ?

Pourquoi apprendre Python ?

- **Lisible** : sa syntaxe est simple et proche du langage naturel.
- **Puissant** : utilisé par Google, YouTube, Netflix, Tesla…
- **Polyvalent** : il sert pour l'intelligence artificielle, le web, les jeux vidéo, la robotique, l'automatisation.
- **Idéal pour débuter** : il est largement utilisé dans les écoles et universités comme premier langage de programmation.

Outils nécessaires pour ce cours

- **Thonny (recommandé)**

Un logiciel gratuit, spécialement conçu pour apprendre Python.

Installation :

1. Aller sur le site officiel https://thonny.org.
2. Télécharger la version adaptée (Windows, Mac ou Linux).
3. Lancer l'installation (tout par défaut suffit).

Avantages de Thonny :

- Interface simple et claire.
- Bouton ▶ (*Run*) pour exécuter directement le programme.
- Messages d'erreurs compréhensibles.
- Mode *pas-à-pas* pour observer l'exécution ligne par ligne.

- **Compilateurs en ligne (si Thonny n'est pas disponible)**

Si tu n'as pas Thonny installé, tu peux tester ton code Python directement dans ton navigateur, sans installation :

- Programiz Online Python Compiler
- Replit Python3

Avantages :

- Rapide et pratique.
- Accessible sur n'importe quel ordinateur connecté à Internet.
- Permet de partager son code avec d'autres.

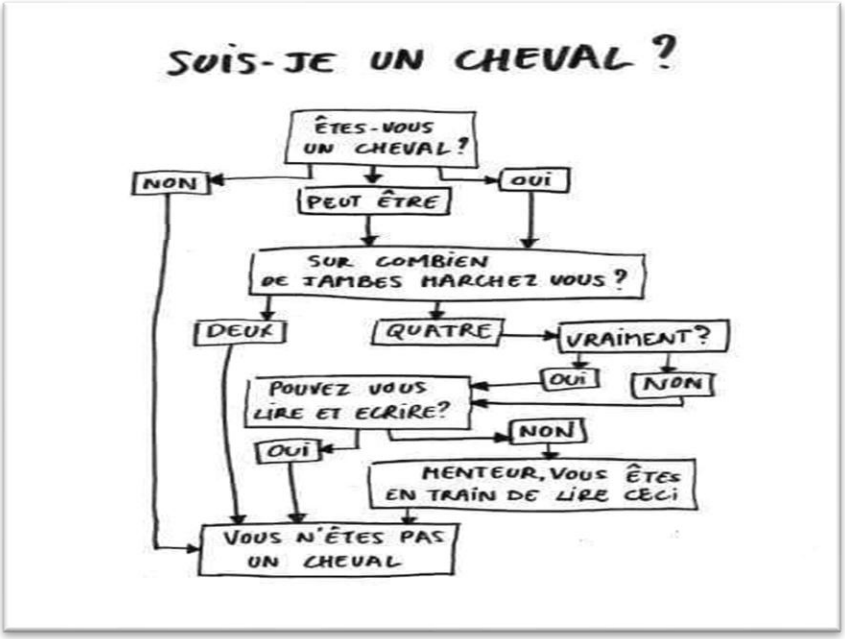
⚠️ Limite : certains compilateurs en ligne gèrent mal input() (saisie utilisateur). Pour les exercices de ce cours, **Thonny reste le meilleur choix**.

Chapitre 1 : Les logigrammes

Qu'est-ce qu'un logigramme ?

Un **logigramme** (ou ordinogramme) est un schéma qui décrit étape par étape la logique d'un programme.
C'est comme un **plan** que l'on fait avant d'écrire le code.
Il aide à :

- **Visualiser** le déroulement d'un programme.
- **Prévoir** les choix possibles.
- **Comprendre** les étapes nécessaires avant d'écrire en Python.



Symboles normalisés

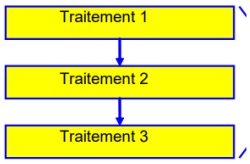
- ⬭ **Ovale** : début ou fin du programme.
- ▭ **Rectangle** : action (ex. : afficher un message, calculer une somme).
- ◆ **Losange** : décision (oui/non, vrai/faux).
- → **Flèche** : sens de lecture et ordre des étapes.

	Premier étape et dernière étape
	Autres étapes
	Un choix, une décision répond toujours par Oui ou Non
	Document lié à une étape
	Lien entre 2 activités

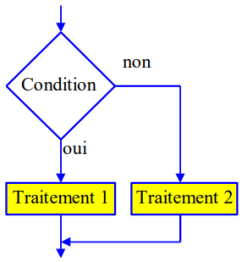
---

Structures fondamentales

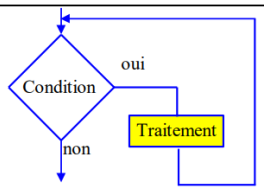
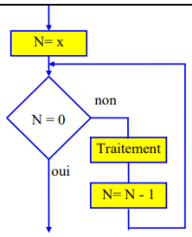
1. **Séquentielle/Linéaire** : les instructions s'exécutent les unes après les autres.
   Exemple : *Se lever → S'habiller → Déjeuner*.

Structure	Algorigramme	Algorithme
Structure linéaire La structure linéaire se caractérise par une suite d'actions à exécuter successivement dans l'ordre de leur énoncé.		**FAIRE « traitement 1 »** **FAIRE « traitement 2 »** **FAIRE « traitement 3 »**

2. **Alternative (choix)** : une condition oriente le déroulement.
   Exemple : *S'il pleut → prendre un parapluie, sinon → sortir sans parapluie*.

Structure alternative ou conditionnelle Une structure alternative n'offre que deux issues possibles s'excluant mutuellement. Une condition est testée et en fonction du résultat du test soit le traitement 1, soit le traitement 2 est réalisé.		**SI « condition » vraie** **ALORS FAIRE** **« traitement 1 »** **SINON FAIRE** **« traitement 2 »** **FIN SI**

3. **Répétitive (boucle)** : une action se répète tant qu'une condition est vraie.
   Exemple : *Répéter "faire ses devoirs" tant que la pile d'exercices n'est pas vide*.

Structure répétitive ou itérative (boucle avec pré-test) Dans cette structure on commence par tester la condition, si elle est vraie alors le traitement est exécuté.		**TANT QUE « condition » vraie** **FAIRE « traitement »** **FIN TANT QUE**
Boucle avec comptage On initialise la variable N avec une valeur x. On teste si N est égal à 0, si ce n'est pas le cas, on exécute le traitement et on décrémente la variable N puis on teste à nouveau la variable N, et ainsi de suite jusqu'à ce que N=0.		**POUR N = x A 0 REPETER** **« traitement »** **FIN POUR**

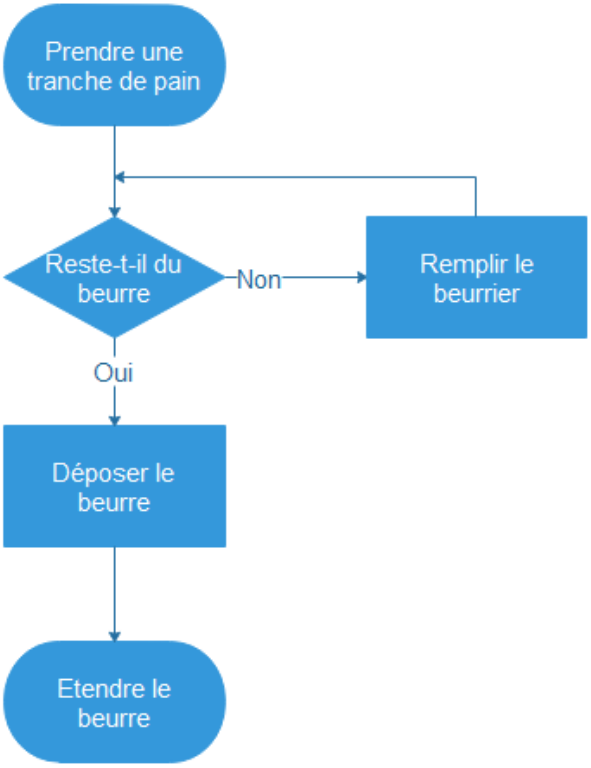
Exemple 1 : Préparer une tartine de beurre

Logigramme simplifié :

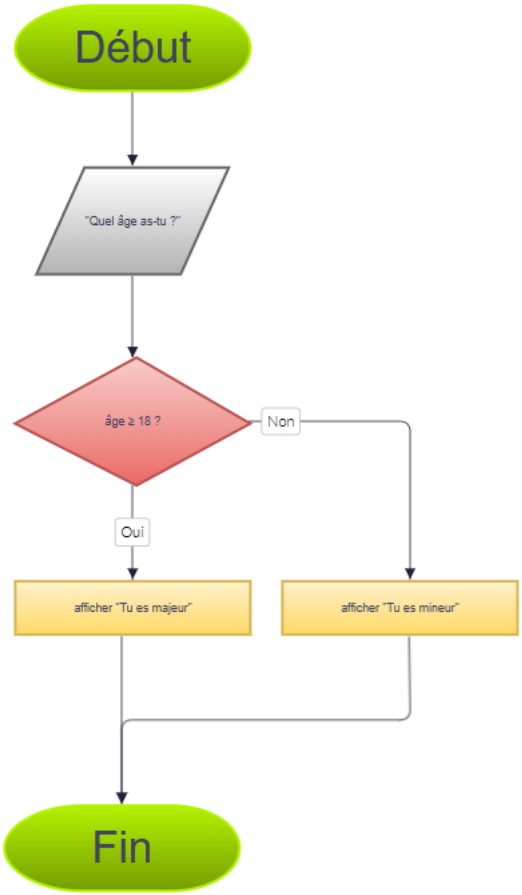
Début → Prendre une tranche de pain → Prendre du beurre → Étaler le beurre → Fin

Logigramme alternatif :

Début → Prendre une tranche de pain → S'il reste du beurre -> Prendre du beurre, sinon → Remplir le beurrier -> Déposer le beurre → Étendre le beurre -> Fin



Exemple 2 : Vérifier la majorité



Les logigrammes (exercice)

🖍️ **Exercice élève**

Dessine un logigramme qui représente les étapes pour :
👉 *Se lever le matin et aller à l'école.*

✅ **Actions proposée**

- Se réveiller
- Se lever
- S'habiller
- Prendre son cartable
- Aller à l'école

Chapitre 2 : Afficher du texte

Explication

En Python, la fonction **print()** permet d'afficher un message à l'écran.

- Le texte à afficher doit être écrit **entre guillemets** " " ou ' '.
- Chaque instruction est exécutée **dans l'ordre, de haut en bas**.
- C'est la première étape pour communiquer avec l'utilisateur.

Exemple simple

```
print("Bonjour le monde !")
print("Bienvenue en informatique !")
```

Résultat affiché :

```
Bonjour le monde !
Bienvenue en informatique !
```

Contrôler la fin de ligne avec **end**

Par défaut, print() ajoute un **retour à la ligne** à la fin de chaque affichage.
On peut modifier ce comportement avec le paramètre **end**.

Exemple

```
print("Bonjour", end=" ")
print("les élèves !")
```

Résultat :

```
Bonjour les élèves !
```

Concaténation et séparation avec des virgules

- **Concaténation (+)** : permet de coller plusieurs chaînes de texte.
- **Virgule (,)** : permet d'afficher plusieurs éléments, séparés par un espace automatique.

Exemple

```
nom = "Alice"
age = 15

print("Bonjour " + nom + ", tu as " + str(age) + " ans.")  # Concaténation
```

```
print("Bonjour", nom, ", tu as", age, "ans.")          # Virgules
```

Résultat :

```
Bonjour Alice, tu as 15 ans.
Bonjour Alice , tu as 15 ans.
```

Conversion de type (str())

⚠️ Quand on veut mélanger du texte et un nombre avec +, il faut convertir le nombre en **chaîne de caractères** avec str().
Sinon, Python renverra une **erreur**.

Exemple

```
x = 10
print("Le résultat est : " + str(x))  # Conversion obligatoire
```

🖍️ **Exercice**

1. Écris un programme qui affiche ton prénom et ton âge sur une seule ligne en utilisant **end=""**.
2. Affiche ensuite :

Bonjour [Prénom], tu as [Âge] ans.

- une fois avec la **concaténation (+)**
- une autre fois avec les **virgules (,)**

✅ **Solution**

```
nom = "Anthony"
age = 15

# 1. Utiliser end=""
print("Mon prénom est", end=" ")
print(nom, "et j'ai", age, "ans.")

# 2. Concaténation
print("Bonjour " + nom + ", tu as " + str(age) + " ans.")

# 3. Virgules
print("Bonjour", nom, ", tu as", age, "ans.")
```

Chapitre 3 : Les commentaires

Explication

Un **commentaire** est une note écrite dans le programme mais **ignorée par Python** lors de l'exécution.
Il sert uniquement pour l'humain (programmeur, professeur, ...).

On écrit un commentaire en commençant la ligne par le symbole **#**.

Pourquoi utiliser des commentaires ?

- Pour **expliquer** ce que fait le code.
- Pour **rendre le programme plus lisible**.
- Pour **désactiver temporairement** une ligne de code sans la supprimer.

Un bon programme contient toujours des commentaires pour être compris par quelqu'un d'autre (ou par soi-même quelques mois plus tard !).

Exemple

```
# Programme qui affiche un message de bienvenue

print("Bonjour")  # Affiche un message
```

Résultat affiché :

```
Bonjour
```

⚠️ Remarque : le texte après # n'apparaît jamais à l'écran.

🖍️ **Exercice**

Écris un programme qui affiche ton prénom et ajoute un commentaire en haut du fichier expliquant ce que fait ton programme.

✅ **Solution**

```
# Programme qui affiche mon prénom

print("Anthony")
```

Chapitre 4 : Les variables et les types

Explication

Une **variable** est comme une **boîte** dans laquelle on peut ranger une valeur.

- Le **nom de la variable** est l'étiquette de la boîte.
- La **valeur de la variable** est le contenu de la boîte.

En Python, on crée une variable en écrivant :

```
nom_de_la_variable = valeur
```

Exemple simple

```
age = 15
nom = "Alice"


print("Bonjour", nom)
print("Tu as", age, "ans.")
```

Résultat :

```
Bonjour Alice
Tu as 15 ans.
```

Modifier une variable

On peut changer le contenu de la boîte à tout moment.

Exemple

```
x = 10
print("Valeur initiale :", x)


x = x + 5
print("Nouvelle valeur :", x)
```

Résultat :

```
Valeur initiale : 10
Nouvelle valeur : 15
```

Les types de variables

En Python, il existe plusieurs types de données :

- **int** : nombres entiers → 1, 42, -7
- **float** : nombres décimaux → 3.14, -0.5
- **str** : chaînes de caractères (texte entre guillemets) → "Bonjour", "Python"
- **bool** : valeurs logiques (vrai ou faux) → True, False

Conversion de type

Parfois, il faut transformer une valeur d'un type à un autre.
Exemple : si on demande un nombre à l'utilisateur avec input(), Python le lit comme **texte** (str).

- int() → transforme en entier
- float() → transforme en nombre décimal
- str() → transforme en texte

Exemple

```
x = "10"      # texte
y = int(x)    # entier
z = float(x)  # décimal

print(x, type(x))
print(y, type(y))
print(z, type(z))
```

Résultat :

```
10 <class 'str'>
10 <class 'int'>
10.0 <class 'float'>
```

🖍️ Exercices

1. Crée une variable nom qui contient ton prénom et affiche :

Bonjour [nom], bienvenue !

2. Crée une variable x = 100. Ajoute 25 à cette variable et affiche le résultat.
3. Crée une variable x = "20" (texte). Transforme-la en entier et additionne-la avec 5.

✅ Solutions

Exercice 1

```
nom = "Anthony"
print("Bonjour", nom, ", bienvenue !")
```

Exercice 2

```
x = 100
x = x + 25
print("La nouvelle valeur est :", x)
```

Exercice 3

```
x = "20"
x = int(x)      # conversion en entier
print(x + 5)
```

Résultat :

```
25
```

Chapitre 5 : Les opérateurs

Explication

Un **opérateur** est un symbole qui permet de réaliser une opération sur des valeurs (nombres, variables...).
On distingue plusieurs familles :

1. **Opérateurs arithmétiques** → calculs
2. **Opérateurs de comparaison** → comparer des valeurs
3. **Opérateurs logiques** → combiner des conditions

---

1. Les opérateurs arithmétiques

Ils permettent de faire des calculs comme en mathématiques.

Opérateur	Signification	Exemple	Résultat
+	Addition	5 + 3	8
-	Soustraction	10 - 4	6
*	Multiplication	7 * 2	14
/	Division	10 / 3	3.333...
//	Division entière	10 // 3	3
%	Modulo (reste)	10 % 3	1
**	Puissance	2 ** 3	8

Exemple

```
a = 10
b = 3


print(a / b)   # 3.333...
print(a // b)  # 3
print(a % b)   # 1
```

---

2. Les opérateurs de comparaison

Ils servent à **tester une condition**. Le résultat est toujours True (vrai) ou False (faux).

Opérateur	Signification	Exemple	Résultat
==	égal à	5 == 5	True
!=	différent de	5 != 3	True
>	supérieur à	10 > 3	True
<	inférieur à	2 < 5	True
>=	supérieur ou égal à	4 >= 4	True
<=	inférieur ou égal à	3 <= 2	False

Exemple

```
x = 7
y = 10

print(x < y)   # True
print(x == y)  # False
```

---

3. Les opérateurs logiques

Ils servent à combiner plusieurs conditions.

Opérateur	Signification	Exemple	Résultat
and	ET logique	(5 > 3) and (10 > 2)	True
or	OU logique	(5 > 3) or (10 > 20)	True
not	NON logique	not(5 > 3)	False

Exemple

```
age = 16
ville = "Namur"

print(age > 15 and ville == "Namur")  # True
print(age > 18 or ville == "Namur")   # True
print(not(age > 18))              # True
```

🖍️ Exercices

1. Calcule le reste de la division de 47 par 6.
2. Vérifie si 15 est plus grand ou égal à 20.
3. Écris un programme qui affiche True si un nombre x est compris entre 10 et 20 (inclus), et False sinon.

✅ Solutions

Exercice 1

```
print(47 % 6)
# Résultat : 5
```

Exercice 2

```
print(15 >= 20)
# Résultat : False
```

Exercice 3

```
x = 15
print(x >= 10 and x <= 20)
# Résultat : True
```

Chapitre 6 : Les entrées utilisateur

Explication

Jusqu'ici, nos programmes affichaient uniquement des messages fixes.
Avec la fonction **input()**, on peut demander une valeur à l'utilisateur.

- input("Texte") affiche un message et attend que l'utilisateur tape quelque chose au clavier.
- Par défaut, la valeur récupérée est une **chaîne de caractères** (str).
- Si on veut un **nombre**, il faut convertir avec int() ou float().

Exemple simple

```
nom = input("Quel est ton prénom ? ")
print("Bonjour", nom, "!")
```

Exécution possible :

```
Quel est ton prénom ? Alice
Bonjour Alice !
```

Conversion pour les nombres

```
age = int(input("Quel est ton âge ? "))
print("L'année prochaine tu auras", age + 1, "ans.")
```

Exécution possible :

```
Quel est ton âge ? 15
L'année prochaine tu auras 16 ans.
```

🖍️ **Exercices**

1. Demande à l'utilisateur son prénom et affiche :

Bonjour [Prénom], bienvenue en Python !

2. Demande un nombre à l'utilisateur, ajoute 10 à ce nombre et affiche le résultat.

✅ Solutions

Exercice 1

```
prenom = input("Entrez votre prénom : ")
print("Bonjour", prenom, ", bienvenue en Python !")
```

Exercice 2

```
x = int(input("Entrez un nombre : "))
x = x + 10
print("Le résultat est :", x)
```

Chapitre 7 : Les conditions

Explication

Une **condition** permet de prendre une décision dans un programme.
En Python, on utilise :

- **if** → exécute un bloc de code si la condition est vraie.
- **else** → exécute un autre bloc si la condition est fausse.
- **elif** → permet de tester plusieurs conditions successives.

⚠️ En Python, les **blocs de code** doivent être **indentés** (décalés avec la touche Tab).

Exemple 1 : if simple

```
age = 18


if age >= 18:
    print("Tu es majeur.")
```

Résultat :

```
Tu es majeur.
```

Exemple 2 : if + else

```
age = 15


if age >= 18:
    print("Tu es majeur.")
else:
    print("Tu es mineur.")
```

Résultat :

```
Tu es mineur.
```

Exemple 3 : if + elif + else

```
note = 12


if note >= 16:
   print("Très bien")
elif note >= 14:
   print("Bien")
elif note >= 10:
   print("Assez bien")
else:
   print("Recalé")
```

Résultat :

```
Assez bien
```

🖍️ Exercices

1. Demande l'âge de l'utilisateur et affiche :
    - "Tu es majeur." si âge ≥ 18
    - "Tu es mineur." sinon.
2. Demande une note à l'utilisateur et affiche :
    - "Excellent" si note ≥ 16
    - "Bien" si note ≥ 12
    - "Passable" si note ≥ 10
    - "Échec" sinon.

✅ Solutions

Exercice 1

```
age = int(input("Quel est ton âge ? "))


if age >= 18:
   print("Tu es majeur.")
else:
```

```
    print("Tu es mineur.")
```

Exercice 2

```
note = int(input("Entrez votre note : "))


if note >= 16:
    print("Excellent")
elif note >= 12:
    print("Bien")
elif note >= 10:
    print("Passable")
else:
    print("Échec")
```

Chapitre 8 : Les boucles

Explication

Une **boucle** permet de répéter un bloc d'instructions plusieurs fois sans réécrire le code.
En Python, il existe deux types principaux de boucles :

- **for** : on connaît à l'avance le nombre de répétitions.
- **while** : on répète tant qu'une condition est vraie.

---

1. La boucle for

Elle utilise la fonction range() qui génère une suite de nombres.

Exemple : répéter 5 fois

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

Résultat :

```
Bonjour
Bonjour
Bonjour
Bonjour
Bonjour
```

Exemple : compter de 1 à 10

```
for i in range(1, 11):
    print(i)
```

Résultat :

```
1
2
3
...
10
```

---

2. La boucle while

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

Exemple : compteur

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

Résultat :

```
0
1
2
3
4
```

⚠️ Attention : si la condition reste toujours vraie, la boucle devient **infinie** et ne s'arrête jamais.

---

3. Boucles imbriquées

On peut mettre une boucle à l'intérieur d'une autre.

Exemple : afficher un carré de X

```
for i in range(5):
    for j in range(10):
        print("X", end="")
    print()
```

Résultat :

```
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
```

🖍️ Exercices

1. Écris une boucle for qui affiche les nombres de 1 à 20.
2. Écris une boucle while qui demande un nombre à l'utilisateur et s'arrête seulement quand l'utilisateur entre 0.
3. Affiche la table de multiplication de 7 avec une boucle for.

✅ Solutions

Exercice 1

```
for i in range(1, 21):
    print(i)
```

Exercice 2

```
x = -1
while x != 0:
    x = int(input("Entrez un nombre (0 pour arrêter) : "))
```

Exercice 3

```
for i in range(1, 11):
    print(i, "x 7 =", i * 7)
```

Chapitre 9 : Les fonctions

Explication

Une **fonction** est un bloc de code qui exécute une tâche précise.

- Elle permet de **réutiliser du code** sans le recopier.
- Elle rend le programme **plus clair et plus organisé**.
- On peut lui donner des **paramètres** (des valeurs à utiliser).
- Elle peut **renvoyer un résultat** avec return.

En Python, on définit une fonction avec le mot-clé **def**.

Exemple 1 : fonction simple sans paramètre

```
def bonjour():
    print("Bonjour à tous !")


bonjour()   # Appel de la fonction
```

Résultat :

```
Bonjour à tous !
```

Exemple 2 : fonction avec paramètres

```
def saluer(prenom):
    print("Bonjour", prenom, "!")


saluer("Alice")
saluer("Marc")
```

Résultat :

```
Bonjour Alice !
Bonjour Marc !
```

Exemple 3 : fonction qui renvoie une valeur

```
def carre(x):
    return x * x


resultat = carre(5)
```

```
print("Le carré de 5 est", resultat)
```

Résultat :

```
Le carré de 5 est 25
```

🖍️ Exercices

1. Écris une fonction dire_bonjour() qui affiche :

Bonjour, bienvenue en Python !

2. Écris une fonction addition(a, b) qui calcule et affiche la somme de deux nombres.
3. Écris une fonction maximum(a, b) qui renvoie le plus grand des deux nombres.

✅ Solutions

Exercice 1

```
def dire_bonjour():
    print("Bonjour, bienvenue en Python !")


dire_bonjour()
```

Exercice 2

```
def addition(a, b):
    print("La somme est :", a + b)


addition(5, 3)
```

Exercice 3

```
def maximum(a, b):
    if a > b:
        return a
    else:
        return b


print("Le plus grand est :", maximum(7, 12))
```

Chapitre 10 : Exercices récapitulatifs

Projet 1 : Mini-calculatrice

Consignes

Écris un programme qui :

1. Demande deux nombres à l'utilisateur.
2. Demande ensuite une opération (+, -, *, /).
3. Affiche le résultat de l'opération choisie.

✅ **Solution**

```
a = float(input("Entrez le premier nombre : "))
b = float(input("Entrez le deuxième nombre : "))
op = input("Choisissez une opération (+, -, *, /) : ")

if op == "+":
    print("Résultat :", a + b)
elif op == "-":
    print("Résultat :", a - b)
elif op == "*":
    print("Résultat :", a * b)
elif op == "/":
    if b != 0:
        print("Résultat :", a / b)
    else:
        print("Erreur : division par zéro !")
else:
    print("Opération non reconnue")
```

Projet 2 : Jeu de devinette

Consignes

1. L'ordinateur choisit un nombre secret (par exemple 7).

2. L’utilisateur doit le deviner.
3. Le programme indique si la proposition est trop petite, trop grande, ou correcte.

✅ **Solution**

```
secret = 7
trouve = False

while not trouve:
    guess = int(input("Devine le nombre : "))

    if guess < secret:
        print("Trop petit !")
    elif guess > secret:
        print("Trop grand !")
    else:
        print("Bravo, tu as trouvé !")
        trouve = True
```

Projet 3 : Fonction de conversion

Consignes

Crée une fonction celsius_to_fahrenheit(c) qui convertit une température en °C vers °F avec la formule :

°F = (°C × 9/5) + 32

Puis teste la fonction avec une valeur donnée par l’utilisateur.

✅ **Solution**

```
def celsius_to_fahrenheit(c):
    return (c * 9/5) + 32

temp = float(input("Entrez une température en °C : "))
print("En Fahrenheit :", celsius_to_fahrenheit(temp))
```

Projet 4 : Questionnaire simple

Consignes

Écris un programme qui :

1. Pose trois questions à l'utilisateur (réponses "oui" ou "non").
2. Donne un score final sur 3.

✅ **Solution**

```
score = 0


q1 = input("Aimes-tu Python ? (oui/non) : ")
if q1 == "oui":
   score += 1


q2 = input("Sais-tu utiliser les boucles ? (oui/non) : ")
if q2 == "oui":
   score += 1


q3 = input("As-tu compris les conditions if ? (oui/non) : ")
if q3 == "oui":
   score += 1


print("Ton score est :", score, "/ 3")
```