UNIVERSITÉ Larbi Ben Mhidi

Faculté des Sciences de la Terre, de la Géographie et de l'Aménagement du Territoire

Département Géographie et aménagement du territoire

Matière : Programmation

Niveau : 1ère année Licence Géographie et aménagement du territoire — Semestre 2

Crédits : 2 | **Coefficient :** 1

Enseignant : Bezzaz Soumia

Année universitaire : 2025–2026

Les Fonctions

1. Introduction

1.1. Qu'est-ce qu'une fonction ?

Une **fonction** est un bloc de code réutilisable qui effectue une tâche spécifique. Les fonctions permettent de structurer le code, d'éviter les répétitions et de faciliter la maintenance des programmes.

Pourquoi utiliser des fonctions ?

• **Réutilisabilité** : écrire une fois, utiliser plusieurs fois

• **Lisibilité** : organiser le code en blocs logiques

• **Maintenabilité** : modifier un seul endroit plutôt que plusieurs

• **Modularité** : diviser un problème complexe en sous-problèmes

Exemple concret :

Au lieu de recalculer manuellement la distance entre deux coordonnées GPS chaque fois qu'on en a besoin, on crée une fonction calculer_distance() qu'on appelle quand nécessaire.

1.2. Les fonctions prédéfinies (built-in)

Python propose de nombreuses fonctions intégrées que nous avons déjà utilisées :

• **print()** : affiche du texte à l'écran

• **input()** : demande une saisie utilisateur

• **type()** : retourne le type d'une variable
• **len()** : retourne la longueur d'une liste ou d'une chaîne
• **int(), float(), str()** : convertissent les types

Exemple :

```
villes = ["Oum_Elbouaghi", "Oran", "Constantine"]
nombre_villes = len(villes)  # Retourne 3
print(nombre_villes)  # Affiche : 3
```

2.1. Syntaxe de base

Pour créer une fonction en Python, on utilise le mot-clé **def** :

```
def nom_fonction():

    instruction1
    instruction2
```

Exemple :

```
def saluer():
    print("Bonjour !")
    print("Bienvenue en programmation")

# Appel de la fonction
saluer()  # Exécute le code de la fonction
```

Sortie :

```
Bonjour !
Bienvenue en programmation
```

Remarque importante :

La définition d'une fonction (avec def) ne l'exécute pas. Il faut l'**appeler** explicitement avec nom_fonction().

2.2. Fonctions avec paramètres

Les paramètres (ou arguments) permettent de passer des valeurs à une fonction pour qu'elle les utilise.

```
def nom_fonction(parametre1, parametre2):
    instructions
```

Exemple :

```
def saluer_personne(nom):
    print(f"Bonjour {nom} !")

# Appels avec différents arguments
saluer_personne("Ahmed")
saluer_personne("Fatima")
```

Sortie :

```
Bonjour Ahmed !
Bonjour Fatima !
```

- **Fonction avec plusieurs paramètres**

```
def calculer_surface_rectangle(longueur, largeur):
    surface = longueur * largeur
    print(f"Surface : {surface} m²")

calculer_surface_rectangle(10, 5)  # Surface : 50 m²
```

2.3. L'instruction return (retourner une valeur)

Le mot-clé **return** permet à une fonction de renvoyer un résultat qui peut être stocké dans une variable ou utilisé dans une expression.

```
def nom_fonction(parametres):
    # Traitement
    return resultat
```

Exemple

```
def calculer_carre(nombre):
    resultat = nombre ** 2
    return resultat

# Utilisation du résultat retourné
x = calculer_carre(5)
print(x)  # Affiche : 25

# Ou directement dans un calcul
y = calculer_carre(3) + calculer_carre(4)  # 9 + 16 = 25
```

Exemple

```
def calculer_densite(population, surface):
    """Calcule la densité de population (hab/km²)"""
    densite = population / surface
    return densite

# Alger : 3 millions d'habitants, 809 km²
densite_alger = calculer_densite(3000000, 809)
print(f"Densité d'Alger : {densite_alger:.2f} hab/km²")
```

Sortie :

```
Densité d'Alger : 3708.03 hab/km²
```

2.4. Paramètres par défaut

On peut définir des valeurs par défaut pour les paramètres. Si l'appelant ne fournit pas de valeur, la valeur par défaut sera utilisée.

```
def afficher_ville(nom, pays="Algérie"):
    print(f"{nom}, {pays}")

afficher_ville("Alger")         # Utilise la valeur par défaut
afficher_ville("Paris", "France")  # Remplace la valeur par défaut
```

Sortie :

```
Alger, Algérie
Paris, France
```

3.1. Variables locales et globales

Variables locales :

Les variables définies à l'intérieur d'une fonction sont **locales** : elles n'existent que dans cette fonction et ne sont pas accessibles en dehors.

Variables globales :

Les variables définies en dehors de toute fonction sont **globales** : elles sont accessibles partout dans le programme.

```
temperature_globale = 20  # Variable globale

def afficher_temperature():
    temperature_locale = 15  # Variable locale
    print(f"Locale : {temperature_locale}")
    print(f"Globale : {temperature_globale}")  # Accessible

afficher_temperature()
# print(temperature_locale)  # ERREUR ! Variable locale inaccessible ici
```

EXEMPLES

4.1. Conversion de températures

```
def celsius_vers_fahrenheit(celsius):
    """Convertit Celsius en Fahrenheit"""
    fahrenheit = (celsius * 9/5) + 32
    return fahrenheit

temp_c = 25
temp_f = celsius_vers_fahrenheit(temp_c)
print(f"{temp_c}°C = {temp_f}°F")  # 25°C = 77.0°F
```

4.2. Calcul de distance entre deux points

```
def calculer_distance(x1, y1, x2, y2):
    """Calcule la distance euclidienne entre deux points"""
    distance = ((x2 - x1)**2 + (y2 - y1)**2)**0.5
    return distance

d = calculer_distance(0, 0, 3, 4)
print(f"Distance : {d}")  # Distance : 5.0
```

4.3. Classification climatique

```
def classifier_climat(temperature_moyenne, precipitation):
    """Classification simplifiée du climat"""
    if temperature_moyenne > 25 and precipitation < 250:
        return "Climat aride"
    elif temperature_moyenne > 20 and precipitation > 1000:
        return "Climat tropical"
    elif temperature_moyenne < 10:
```

```
        return "Climat froid"
    else:
        return "Climat tempéré"

climat = classifier_climat(28, 200)
print(climat)  # Climat aride
```

EXERCICES

Exercice 1

Créer une fonction afficher_message() qui affiche « Bienvenue dans le cours de programmation ».

Solution

```
def afficher_message():
    print("Bienvenue dans le cours de programmation")

afficher_message()
```

Exercice 2

Créer une fonction calculer_double(n) qui affiche le double d'un nombre.

Solution

```
def calculer_double(n):
    resultat = n * 2
    print(f"Le double de {n} est {resultat}")

calculer_double(7)  # Le double de 7 est 14
```

Exercice 3

Créer une fonction calculer_aire_cercle(rayon) qui retourne l'aire d'un cercle.
Formule : $\pi \times r^2$ (utiliser 3.14 pour π).

Solution

```
def calculer_aire_cercle(rayon):
    aire = 3.14 * rayon ** 2
    return aire
```

```
a = calculer_aire_cercle(5)
print(f"Aire : {a}")  # Aire : 78.5
```

Exercice 4

Créer une fonction est_ville_grande(population) qui retourne True si la population est supérieure à 1 million, sinon False.

Solution

```
def est_ville_grande(population):
    if population > 1000000:
        return True
    else:
        return False


print(est_ville_grande(3000000))  # True
print(est_ville_grande(500000))   # False
```

Exercice 5

Créer une fonction analyser_pluviometrie(mm) qui :

- Retourne « Sec » si mm < 100
- Retourne « Modéré » si 100 ≤ mm < 500
- Retourne « Humide » si mm ≥ 500

Solution

```
def analyser_pluviometrie(mm):
    if mm < 100:
        return "Sec"
    elif mm < 500:
        return "Modéré"
    else:
        return "Humide"


print(analyser_pluviometrie(75))   # Sec
print(analyser_pluviometrie(300))  # Modéré
print(analyser_pluviometrie(600))  # Humide
```

Conclusion

Les fonctions sont essentielles en programmation. Elles permettent de :

• **Réutiliser du code** sans le réécrire
• **Organiser le programme** en blocs logiques
• **Faciliter la maintenance** et la lecture du code
• **Créer des outils réutilisables** pour des calculs spécifiques