### 🔹 Why Functions?

Functions help you group reusable code. Instead of writing the same logic again and again, you write it once inside a function and call it whenever needed.

### Simple Example:

def greet(name):

return f"Hello, {name}!"

print(greet("Vijay"))

print(greet("Aroha"))

Output:

Hello, Vijay!

Hello, Aroha!

### 🔹 Real-Time Problem Statement:

You are working in a banking project.  
You need a function that calculates Simple Interest given principal, rate, and time.

Formula:

Simple Interest = (Principal \* Rate \* Time) / 100

### Example Code:

def simple\_interest(principal, rate, time):

return (principal \* rate \* time) / 100

print(simple\_interest(10000, 5, 2)) # 1000

Here’s the roadmap for Functions:

1. Defining and Calling Functions
2. Function Parameters
   * Positional Arguments
   * Keyword Arguments
   * Default Arguments
   * Variable-Length Arguments (\*args, \*\*kwargs)
3. Return Statement
4. Scope (Local & Global Variables)
5. Nested Functions
6. Lambda (Anonymous Functions)
7. Recursion
8. Built-in Higher Order Functions (map, filter, reduce)
9. Decorators (Advanced)

### 🔹 Step 1: Defining and Calling a Function

A function is defined using the def keyword.

Example:

def greet():

print("Hello, Welcome to Python Functions!")

# Calling the function

greet()

Output:

Hello, Welcome to Python Functions!

👉 This is the simplest function:

* No parameters
* No return value

### 🔹 2.1 Positional Arguments

Arguments are passed in the same order as parameters.

Example:

def add(a, b):

print("Sum:", a + b)

add(10, 20) # a=10, b=20

Output:

Sum: 30

### 🔹 2.2 Keyword Arguments

We specify parameters by name, so order doesn’t matter.

Example:

def introduce(name, age):

print(f"My name is {name} and I am {age} years old.")

introduce(age=25, name="Vijay")

Output:

My name is Vijay and I am 25 years old.

### 🔹 2.3 Default Arguments

We give a default value if no argument is passed.

Example:

def greet(name="Guest"):

print(f"Hello, {name}!")

greet("Vijay")

greet()

Output:

Hello, Vijay!

Hello, Guest!

### 🔹 2.4 Variable-Length Arguments

When we don’t know how many values will be passed, we use:

#### (a) \*args → accepts multiple values as a **tuple**

def total\_sum(\*numbers):

print("Numbers:", numbers)

print("Sum:", sum(numbers))

total\_sum(10, 20, 30, 40)

Output:

Numbers: (10, 20, 30, 40)

Sum: 100

#### (b) \*\*kwargs → accepts multiple values as a **dictionary**

def person\_info(\*\*details):

print("Details:", details)

person\_info(name="Vijay", age=25, city="Nellore")

Output:

Details: {'name': 'Vijay', 'age': 25, 'city': 'Nellore'}

Summary of Step 2:

* Positional → based on order
* Keyword → based on name
* Default → fallback value
* \*args → multiple positional values (tuple)
* \*\*kwargs → multiple keyword values (dictionary)