What's a function?

A function is a named block of code you can call to do a task. It can optionally take parameters (inputs) and optionally return a value (output).

```
def name(parameters):
    """Docstring: what it does, args, returns."""
    # body
    return result
```

1. Functions with parameter

- **Meaning:** The function **takes inputs** (parameters/arguments) when you call it.
- **Purpose:** Used when you want the function to work on *different data* each time.

Example:

```
def greet(name):
    print(f"Hello, {name}!")

# Calling the function with different arguments
greet("Kavya")
```

OUTPUT: Hello Kavya

NOTE: Here, `name` is a **parameter**, and `"Kavya"` are **arguments** you pass while calling.

2.Functions without Parameters

- **Meaning:** The function **does not take any input** when called.
- **Purpose:** Used when the function always does the same job or uses data already available in the program.

EXAMPLE:

```
def greet():
    print("Hello, welcome to Python programming!")

# Calling the function
greet()
```

OUTPUT: Hello,welcome to Python programming!

NOTE: No matter how many times you call `greet()`, the message stays the same.

3) Parameters vs. Arguments (quick clarity)

- **Parameter**: the name in the function definition (`def f(x):` → `x` is a parameter).
- **Argument**: the actual value you pass (`f(10)` → `10` is an argument).

When to choose which?

Prefer *with parameters* when:

- You want reusability, testability, and composability.
- The task should work on different inputs.
- You aim for pure functions (no side effects).

Use *without parameters* when:

- The function is a simple, fixed action (e.g., print header, show menu).
- It wraps external I/O (prompt user, read time) that doesn't need inputs.