Functions

🧠 Functions

Let's go slowly and intuitively, so you *understand* functions — not just use them.

🔷 What is a Function?

Think of a **function** as a **mini machine** inside your program.

You **give it input**, it **does some work**, and it **returns an output**.

You've already used functions without realizing it!

Examples:

```
print("Hello")
type(5)
len("Ameya")
```

Here, `print()`, `type()`, and `len()` are **built-in functions**.

They take some data (called *arguments*) and perform tasks for you.

🔷 Why Functions Are Needed

Imagine you're building a game, and you need to check who won — 20 times.

Without functions, you'd have to copy the same code again and again.

With functions, you can **write the logic once**, and **call it whenever you want**.

Functions help you:

1. Avoid repeating code (DRY – *Don't Repeat Yourself*)
2. Organize your logic
3. Make code reusable and readable

🔷 Creating (Defining) a Function

In Python, we use the keyword `def` to define a function.

Syntax:

```
def function_name():
    # code block
```

Example:

```
def greet():
    print("Hello, AMK!")
```

Now to use (or “call”) it:

```
greet()
```

Output:

```
Hello, AMK!
```

🔷 Functions with Parameters (Inputs)

You can pass **data into a function** — just like putting ingredients into a blender.

Example:

```
def greet(name):
    print("Hello,", name)
```

Now call it with different values:

```
greet("AMK")
greet("HRS")
```

```
```

Output:

```
Hello, AMK
Hello, HRS
```

So `name` is a **parameter**, and `"AMK"` or `"HRS"` are **arguments**.

🔷 Functions that Return a Value (Outputs)

Sometimes you want a function to **give back a result**, not just print something.

For that, you use the `return` keyword.

Example:

```
def add(a, b):
    return a + b
```

Now:

```
result = add(5, 3)
print(result)
```

Output:

```
8
```

The `return` sends the answer back to the line that called the function.

🔷 Putting it All Together

Let's combine it all:

```
def calculator(a, b, operation):
    if operation == "add":
        return a + b
    elif operation == "sub":
        return a - b
    elif operation == "mul":
        return a * b
    elif operation == "div":
        return a / b
    else:
        return "Invalid operation"
```

Now:

```
print(calculator(10, 5, "add"))  # 15
print(calculator(10, 5, "mul"))  # 50
```

You just built your first **multi-purpose function.**

🔷 Default Parameters

You can also give a **default value** to parameters — used if no argument is passed.

```
def greet(name="AMK"):
    print("Hello,", name)
```

Now:

```
greet()          # Hello, AMK
greet("HRS")   # Hello, HRS
```

🔷 Return vs Print (Very Important Difference)

- `print()` → displays the result on screen
- `return` → *sends the result back to your program* (so you can use it again)

Example:

```
def square(n):
    return n * n

result = square(4)
print(result)  # Output: 16
```

If you had used `print()` inside the function instead of `return`, you couldn't reuse that result in other calculations.

Mental Model

Concept	What it means
Function	A mini program that performs a task
Parameter	A variable inside the function (placeholder for input)
Argument	The real value you pass to it
Return	Sends the result back
Call	Using the function to do something

Mini Practice

1. Write a function `square(number)` → returns the square of a number.
2. Write a function `is_even(n)` → returns True if number is even, else False.
3. Write a function `greet_user(name, age)` → prints "Hello Ameya, you are 17 years old!"