# Ch-3 Functions in Python

## What is a Function?

A **function** is a block of reusable code that performs a specific task.

## Types of Functions

| Type | Description | Example |
|---|---|---|
| **Built-in Functions** | Pre-defined in Python | len(), type(), print() |
| **Module-defined** | Defined in standard or external modules | math.sqrt(), random.randint() |
| **User-defined** | Created by the programmer | def greet(): ... |

## Creating User-defined Functions

**Syntax:**

```
def function_name(parameters):
    # block of code
    return result
```

**Example:**

```
def greet(name):
    return "Hello " + name

print(greet("Ravi"))  # Output: Hello Ravi
```

## Arguments and Parameters

| Term | Description |
|---|---|
| Parameter | Variable in the function definition |
| Argument | Value passed when calling the function |

```
def square(x):     # x is parameter
    return x * x

square(5)          # 5 is argument
```

## Default Parameters

If no argument is provided, the default value is used.

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

greet()            # Hello Student
greet("Ravi")      # Hello Ravi
```

## Positional Parameters

Arguments passed in the same order as defined.

```
def display(name, age):
    print(name, age)

display("Amit", 20)  # Amit 20
```

## Function Returning Value(s)

You can use return to send data back from a function.

```
def add(x, y):
    return x + y

result = add(10, 5)
```

```
print(result)  # 15
```

## Flow of Execution

- Execution starts from the first line of the program.
- When a function is called, control jumps to the function.
- After return, control goes back to the caller.

## Scope of a Variable

| Scope | Meaning |
|---|---|
| Local | Defined inside a function; accessible only there |
| Global | Defined outside all functions; accessible everywhere |

```
x = 10  # Global

def show():
    x = 5  # Local
    print(x)

show()      # 5
print(x)    # 10
```

# Exception Handling in Python

## What is an Exception?

An **exception** is an error that occurs during program execution and disrupts the normal flow.
Examples: ZeroDivisionError, ValueError, IndexError, FileNotFoundError

## try-except Block

**Basic Syntax:**

```
try:
    # Code that may raise an error
except ExceptionType:
    # Code to handle the error
```

**Example:**

```
try:
    a = int(input("Enter number: "))
    b = 10 / a
    print("Result:", b)
except ZeroDivisionError:
    print("Cannot divide by zero!")
except ValueError:
    print("Invalid input! Please enter a number.")
```

## try-except-finally Block

- finally block always runs, even if an exception occurs.
- Useful for cleanup actions (like closing a file).

```
try:
    file = open("data.txt", "r")
    print(file.read())
except FileNotFoundError:
    print("File not found!")
finally:
    print("Execution completed.")
```