🔹 1. `print()` Function

Definition:

The `print()` function is used to **display messages, numbers, or results** on the screen.

Explanation:

When we write a Python program, we often want to **see something as output** — for example, a welcome message, a calculation, or even a user's name.
We use `print()` to ask the computer to **show that output**.

We place the content (text or numbers) inside **round brackets** `()` and use **quotes** for text messages.

Examples:

Example 1: Print text (message)

```
print("Hello, World!")
```

📤 Output:

```
Hello, World!
```

Example 2: Print a number

```
print(100)
```

📤 Output:

```
100
```

Example 3: Print text and number together

```
print("My score is", 90)
```

📤 Output:

```
My score is 90
```

📝 Practice Questions:

1. Use `print()` to show your name.
2. Show a welcome message using `print()`.
3. Print your school name and your favorite subject.
4. Use `print()` to display the result of `5 + 3`.
5. Try printing text and a number together like:
   `"My age is 20"`

🔹 2. Printing Multiple Lines

Definition:

To print more than one line, we can use:

- Multiple `print()` functions
- Special symbol `\n` (new line)

Explanation:

Sometimes we want to print many lines of output.
We can do it either by writing `print()` multiple times or by using **`\n`**, which tells Python to move to a **new line**.

Examples:

Example 1: Using multiple `print()` statements

```
print("Line 1")
print("Line 2")
print("Line 3")
```

📤 Output:

```
Line 1
Line 2
Line 3
```

Example 2: Using `\n` inside one `print()`

```
print("Line 1\nLine 2\nLine 3")
```

📤 Output:

```
Line 1
Line 2
Line 3
```

Example 3: Triple quotes for long messages

```
print("""Hello
This is Python
Enjoy learning!""")
```

📤 Output:

```
Hello
This is Python
Enjoy learning!
```

📝 Practice Questions:

1. Print the following using 3 `print()` statements:
   - "My name is __"
   - "I am __ years old"
   - "I love Python"
2. Now print the same using just **one `print()` statement** and `\n`.

   Try this:

   ```
   print("Python is fun\nLet's learn more!")
   ```

3. What do you think the output will be?
4. Use triple quotes to print a short poem or paragraph of 3 lines.

🔹 3. Comments in Python

Definition:

Comments are lines in your code that **Python ignores**. They are meant to help **you or others understand the code** better.

Explanation:

- Comments are for **humans**, not computers.
- Used to **explain what each part of code does**.
- Makes your program **easier to read and debug**.

There are two main types of comments:

- **Single-line** (using `#`)
- **Multi-line** (using `''' ... '''` or `""" ... """`)

🧪 Examples:

Example 1: Single-line comment

```
# This line prints a welcome message
print("Welcome to Python!")
```

Example 2: Comments beside code

```
print("Hello")  # This prints Hello
```

Example 3: Multi-line comment

```
'''
This program shows:
1. My name
2. My age
3. My city
'''
print("My name is Aisha")
```

📝 Practice Questions:

1. Write a program that prints your:
    - Name
    - City

- Age
  Add a comment before each line explaining what it does.

2. Use a **multi-line comment** to describe what your program does.

Try disabling a line using a comment like:

```
# print("This will not run")
```

3.
4. What happens if you remove the # from a comment line?

---

1. What is a Variable?

Definition:

A **variable** is like a **container** or **box** that stores some information (data) which can be used later in the program.

Explanation:

Imagine a container where you can store your name, age, marks, or any value.
In Python, we use **variables** to store this data.

You can **name** the container (variable) anything you like (with some rules).
The value **inside** the container can be **text, number, or any data**.

We use = (equal sign) to store (assign) values into variables.

🧪 Examples:

Example 1: Storing a name

```
name = "Aisha"
print(name)
```

📤 Output:

```
Aisha
```

Example 2: Storing a number

```
age = 18
print(age)
```

📤 Output:

```
18
```

Example 3: Changing the value

```
city = "Delhi"
print(city)

city = "Mumbai"
print(city)
```

📤 Output:

```
Delhi
```

```
Mumbai
```

📝 Practice Questions:

1. Store your name in a variable and print it.
2. Create a variable for your age and print it.
3. Change the value of your favorite color and print it before and after change.

Try this:

```
number = 5
number = number + 3
print(number)
```

4. What do you think it will print?

🔹 2. Rules for Naming Variables

Definition:

Python has **certain rules** for naming variables. These rules must be followed, or your code will show an error.

Explanation:

✅ **Valid variable names**:

- Must **start with a letter** (a-z, A-Z) or an **underscore** _
- Can contain **letters**, **numbers**, and **underscores** (no spaces)
- **Cannot** start with a number

- **Are case-sensitive** (`Name` and `name` are different)
- Can't use Python Keywords (if, else, class)

🚫 **Invalid examples**:

- `2name` ❌ (starts with a number)
- `my name` ❌ (has a space)
- `class` ❌ (Python keyword)

🧪 Examples:

Valid variable names:

```
student_name = "Anil"
age = 16
_score = 98
```

Invalid variable names:

```
2name = "Amit"       # ❌ Starts with number
my name = "Rahul"    # ❌ Has a space
class = "Math"       # ❌ 'class' is a keyword
```

🔍 Tip:

Use **meaningful variable names** so your code is easy to understand.
For example:
✅ `marks`, `student_name`, `total_amount`
❌ `x`, `a1`, `temp` (okay, but unclear)

📝 Practice Questions:

1. Create variables with the following names and assign values:
    - `student_name`
    - `student_age`
    - `student_city`
2. Which of these are **valid** and which are **invalid**? Try them in Python:
    - `marks1`
    - `1marks`
    - `my name`
    - `_marks`
    - `TotalMarks`
3. Create a variable with your favorite subject and print it.
4. Try changing a variable's value 2 times and print it each time.

🔹 1. What are Data Types?

Definition:

A **data type** tells Python what kind of value a variable holds — like a number, decimal, or text.

Explanation:

Every value in Python has a **type**. For example:

- `"Aisha"` is a **string** (text)
- `20` is an **integer** (whole number)
- `99.5` is a **float** (decimal number)

Python automatically **detects the type** based on the value you assign.

We mostly use these 3 basic types:

Data Type	What it stores	Example
`int`	Whole numbers	5, 100, -12
`float`	Decimal numbers	3.14, 9.5
`str`	Text (characters/words)	"hello", "A1"

🧪 Examples:

Example 1: Integer (`int`)

```
age = 18
print(age)
print(type(age))
```

📤 Output:

```
18
<class 'int'>
```

Example 2: Float (`float`)

```
price = 49.99
print(price)
print(type(price))
```

📤 Output:

```
49.99
<class 'float'>
```

Example 3: String (`str`)

```
name = "Ravi"
print(name)
print(type(name))
```

📤 Output:

```
Ravi
<class 'str'>
```

📝 Practice Questions:

1. Create a variable for your marks (like 85) and print its type.
2. Create a variable for height (like 5.6) and check its data type.
3. Store your best friend's name and print it along with its type.

4. Try printing types of the following directly:

```
print(type(42))
```

```
print(type(3.5))
print(type("Python"))
```

🔹 2. Changing Between Data Types (Type Casting)

Definition:

Type casting means converting one data type into another — like changing a number into a string or vice versa.

Explanation:

Python gives us built-in functions for casting:

- `int()` → to convert to integer
- `float()` → to convert to float
- `str()` → to convert to string

🧪 Examples:

Example 1: int → float

```
x = 5
y = float(x)
print(y)
```

📤 Output:

```
5.0
```

Example 2: float → int

```
a = 3.99
b = int(a)
print(b)
```

📤 Output:

```
3  # decimals are removed
```

Example 3: int → str

```
age = 20
age_text = str(age)
print("I am " + age_text + " years old.")
```

📤 Output:

```
I am 20 years old.
```

📝 Practice Questions:

1. Convert `45.6` to integer and print it.

2. Convert your age (as an `int`) to `str` and join with a sentence using `+`.

3. Try converting `"100"` to `int` using `int("100")` — what happens?

4. Try this:

```
x = "50"
y = int(x)
z = y + 10
print(z)
```

🔹 1. Getting Input from the User

Definition:

`input()` is a built-in Python function used to take input (data) from the user while the program is running.

Explanation:

When we want the user to **enter something** — like their name, age, or marks — we use the `input()` function.

- By default, whatever the user types is treated as **text (string)**.
- If we want numbers, we have to **convert** the input using `int()` or `float()`.

📢 Syntax:

```
variable = input("Your message here: ")
```

🧪 Examples:

Example 1: Basic input

```
name = input("Enter your name: ")
print("Hello", name)
```

📤 Output:

```
Enter your name: Aisha
```

```
Hello Aisha
```

Example 2: Input a number (without conversion)

```
age = input("Enter your age: ")
print("You are", age, "years old")
```

🔎 Note: `age` is stored as a string.

🔹 2. Type Conversion with Input

Definition:

We convert (or cast) the input to the required type using functions like `int()` or `float()`.

Explanation:

When you want to perform **mathematical operations**, you must convert input into numbers.

🧪 Examples:

Example 1: Add two numbers from user

```
a = input("Enter first number: ")
b = input("Enter second number: ")

# Convert input to integers
sum = int(a) + int(b)
print("The sum is:", sum)
```

📤 Output:

```
Enter first number: 10
Enter second number: 20
The sum is: 30
```

Example 2: Input decimal numbers

```
price = float(input("Enter price: "))
qty = int(input("Enter quantity: "))

total = price * qty
print("Total cost is:", total)
```

Common Mistake:

```
a = input("Enter number: ")
b = input("Enter number: ")
print(a + b)
```

📤 Output (if input is 5 and 6): `56` ❌

💡 Because both are **strings**, they are joined like text.

✅ Fix it with: `int(a) + int(b)`

📝 Practice Questions:

1. Ask user for their name and age. Print: `Hello John, you are 25 years old`.
2. Take two numbers from user and print their sum.
3. Input marks of 3 subjects and print total and average.
4. Ask user for their birth year and calculate current age.

5. Take user's height (in float) and weight, then print them in one line.

🔹 1. What are Operators?

Definition:

Operators are special symbols in Python used to **perform operations** on values or variables — like addition, subtraction, comparison, etc.

Explanation:

Operators help us do **math**, **assign values**, **compare values**, and more.

The most common are:

- **Arithmetic Operators** (like +, -, *, /)
- **Assignment Operators** (like =, +=, -=)

🔹 2. Arithmetic Operators

Operator	Meaning	Example (`a = 10`, `b = 5`)	Result
`+`	Addition	`a + b`	`15`
`-`	Subtraction	`a - b`	`5`
`*`	Multiplication	`a * b`	`50`
`/`	Division (float)	`a / b`	`2.0`

`//`	Division (floor)	`a // b`	`2`
`%`	Modulus (remainder)	`a % b`	`0`
`**`	Power (exponent)	`a ** b`	`10000` `0`

🧪 Examples:

Example 1: Addition

```
x = 7
y = 3
print(x + y)  # 10
```

Example 2: Division

```
print(10 / 3)   # 3.333...
print(10 // 3)  # 3 (only whole part)
```

Example 3: Remainder

```
print(10 % 3)  # 1
```

📝 Practice Questions:

1. Take two numbers from the user and print their sum, difference, product.
2. Print the remainder when 29 is divided by 5.
3. Try `2 ** 3` and explain what it does.
4. Try floor division `23 // 4` — what do you get?

🔹 3. Assignment Operators

Definition:

Assignment operators are used to **assign or update** the value of a variable.

Explanation:

You already know the basic assignment:

```
x = 5  # Assigns 5 to x
```

But we also have **shorthand** versions to update values:

Operator	Meaning	Example	Same As
`=`	Assign value	`x = 5`	`x = 5`
`+=`	Add and assign	`x += 2`	`x = x + 2`
`-=`	Subtract and assign	`x -= 1`	`x = x - 1`
`*=`	Multiply and assign	`x *= 3`	`x = x * 3`
`/=`	Divide and assign	`x /= 2`	`x = x / 2`

Examples:

```
x = 10
x += 5
print(x)  # 15

x *= 2
print(x)  # 30

x -= 10
```

```
print(x)  # 20
```

📝 Practice Questions:

1. Set `x = 20`. Then use `x += 10`, `x -= 5`, and print each time.
2. Create a variable `marks = 80`. Increase it by 10 using `+=`.
3. Try `x = 9`, then `x *= 3`. What is the value of `x`?
4. Use `input()` to take a number, increase it by 10, and print the result.