PRINT() INBUILD IN PYTHON

- The print() function is one of the most commonly used functions in Python.
- It is used to send output to the console

Basic Syntax of `print()`

```python
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
```

- `*objects` : One or more objects (strings, numbers, etc.) that you want to print.
- `sep` : String inserted between the objects. By default, it is a space (`' '`).
- `end` : String appended after the last object. By default, it is a newline (`'\n'`), which moves the cursor to the next line.
- `file` : The stream where the output is sent. By default, it is `sys.stdout` (standard output, i.e., the console).
- `flush` : Whether to flush the output buffer. Default is `False` .

Printing a String

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

Printing Numbers

```python
print(42)
print(3.14)
```

Printing multiple objects →

```
x = 10
y = 20
print("The value of x is", x, "and the value of y is", y)
```

Customizing the separator →

```
print("apple", "banana", "cherry", sep=", ")
```

No Separator

```python
print("apple", "banana", "cherry", sep="")
```

- Output: `applebananacherry`

Custom End Character

```python
print("Hello", end=" ")
print("World!")
```

No Line Break

```python
print("Hello", end="")
print("World!")
```

- Output: `HelloWorld!`

Redirecting Output to a File→

By default, the output of print() is sent to the console (standard output).
Output will generate in another file

Writing to a File

```python
with open('output.txt', 'w') as file:
    print("Hello, World!", file=file)
```

Printing Formatted Strings→

You can combine print() with string formatting to create more dynamic and readable output. There are several ways to format strings in Python.

Old-Style String Formatting (`%`)

```python
name = "Alice"
age = 30
print("My name is %s and I am %d years old" % (name, age))
```

- **Output:** `My name is Alice and I am 30 years old`

`str.format()` Method

```python
name = "Alice"
age = 30
print("My name is {} and I am {} years old".format(name, age))
```

- **Output:** `My name is Alice and I am 30 years old`

Printing Special Characters →

The print() function also handles special characters, such as newlines (\n), tabs (\t),

```
print("Hello\nWorld!")
```

Tab Character (`\t`)

```python
print("apple\tbanana\tcherry")
```

- **Output:** `apple banana cherry`

Handling Unicode Characters→

Python's print() function supports Unicode, which means you can print characters from different languages or special symbols.

```
print("Hello, 🌍!")
```

- **Output:** `Hello, 🌍!`
- The 🌍 is a Unicode emoji that can be printed directly.

Using Multiple print () Calls→

Using Multiple `print()` Calls

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

Using `\n` to Create New Lines

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

- **Output:**

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

- The print() function is a powerful and flexible tool in Python for outputting information.
- By using its various parameters and features, you can control the output format, direct it to files, and print different types of data efficiently.
- It's an essential function for both debugging and user interaction in Python programs.