1. Printing on Screen

❖Introduction to the print() function in Python.

What is print()?

- In Python 3, print() is a built-in function used to output text or other objects to the screen or another output stream.
- Internally, it converts its arguments to strings using str(), joins them with a separator, then writes to sys.stdout , and finally adds an end character .

Function signature :

Syntax :

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

- *objects: any number of items to print
- sep: string inserted between items (default: space)

- end: string appended after the last item (default: newline)
- file: output stream (default: sys.stdout)
- flush: if True, forces immediate output instead of buffering .

❖Formatting outputs using f-strings and format().

F-strings (formatted string literals) :

- Use by prefixing your string literal with f or F:

```
name = "Alice"
f"Hello, {name}!"
```

- You can embed expressions, apply format specifiers, and even call functions:

```
balance = 1234.567
f"Balance: ${balance:,.2f} and next year it'll be {balance*1.05:.2f}"
```

- They support the same advanced formatting mini-language as .format().

str.format() :

- Use .format() with {} placeholders:

```
"Hello, {}!".format(name)
"{name} is {age} years old".format(name="Bob", age=30)
```

- Great for templating strings defined separately from variables; useful for dictionary unpacking:

```
person = {'name': 'Bob', 'age': 30}
"Name: {name}, Age: {age}".format(**person)
```

- Works reliably across older Python versions since it's been around longer.