📘 **Chapter 4 : print() Function & Related Basics in Python**

**1. What print() Does**

* **Built‑in function** that sends text (or other printable objects) from your code to the console/output window.
* Commonly used for:
  + Communicating messages to end‑users.
  + Showing intermediate values while *debugging* or *testing*.
  + Logging progress in longer scripts.

**2. General Syntax**

print(<item1>, <item2>, ..., sep=' ', end='\n')

* Parentheses are required in modern Python (3.x).
* Each item is converted to a string and joined with sep (default single space).
* Output ends with end (default newline).

**3. Quotes & String Delimiters**

| **Goal** | **Valid Example** | **Notes** |
| --- | --- | --- |
| Basic string | print("Hi Python") | Double quotes |
| Same with single quotes | print('Hi Python') | Either style works |
| Mixing quotes inside a string | print("He said 'hello'") | Outside quotes differ from inside |
| Escaping same quote type | print("He said \"hello\"") | Use backslash (\) before the quote |

**4. Escape Sequences (start with \)**

| **Sequence** | **Effect** | **Example → Output** |
| --- | --- | --- |
| \n | New line | print("Line1\nLine2") → two lines |
| \t | Horizontal tab | print("A\tB") → A B |
| \\ | Literal backslash | print("C:\\Users") → C:\Users |
| \', \" | Literal single/double quote | As above examples |

**Remember:** a lone backslash without a known escape code causes a *syntax error*.

**5. Multi‑Line & Nicely Formatted Strings**

1. **Explicit escape codes**
2. print("Your learning path:\n\t- Python basics\n\t- Data Engineering\n\t- AI")
3. **Triple‑quoted strings** (preserve line breaks)
4. print("""
5. Your learning path:
6. - Python basics
7. - Data Engineering
8. - AI
9. """)
   * Triple single (''') or triple double (""") quotes both work.
   * Line breaks and indentation inside the quotes appear in the output (be mindful of leading spaces).

**6. Why & Where You Use print() in Real Projects**

| **Use‑Case** | **Example** |
| --- | --- |
| **Quick result display** | print("Final total:", final\_total) |
| **Debugging checkpoints** | print("Subtotal =", subtotal) inside loops or complex logic |
| **Progress logging** | print(f"Processed {i}/{total} records") |

Replace ad‑hoc print() with proper logging (logging module) in production code, but print() is perfect for quick scripts, teaching, and exploratory work.

**7. Functions Refresher (context)**

* **Built‑in functions**: shipped with Python (e.g., print, len, input, max).
* **Third‑party functions**: come from external libraries like *pandas* or *NumPy*.
* **User‑defined functions**: the ones *you* create via def.

Think of a function as a *coffee machine*: you pass inputs, internal mechanisms work, and you receive an output—without worrying about internal wiring.

**8. Common Pitfalls & Tips**

| **Pitfall** | **How to Avoid** |
| --- | --- |
| Mixing start/end quote types | Always close with the same quote you opened. |
| Using backslash incorrectly in file paths | Use raw strings r"C:\Users\Bob" **or** double backslashes. |
| Forgetting newline between consecutive prints | Chain strings with \n or call print() separately. |
| Overusing print() in production | Transition to the logging module for serious apps. |

**9. Mini‑Cheat Sheet**

# Basic message

print("Hello, world!")

# Concatenation with commas (automatic spaces)

name, score = "Alice", 92

print("User:", name, "| Score:", score) # User: Alice | Score: 92

# No newline at the end

print("Loading...", end="") # keeps cursor on same line

# Custom separator

print("a","b","c", sep="|") # a|b|c

# Debug example

subtotal = 1200

discount = 0.15 \* subtotal

final\_total = subtotal - discount

print("Subtotal:", subtotal)

print("Discount:", discount)

print("Final total:", final\_total)