Comments and Indentation in Python

1. Comments in Python

Comments are **non-executable** lines that help explain the code. Python **ignores** comments when running the program.

Single-Line Comment

Start with `#`:

```
# This is a single-line comment
print("Hello, Python!")
```

Multi-Line Comment (Not Official Syntax, but Works)

Python doesn't have a direct multi-line comment like `/* */` in C, but you can use triple quotes (`'''` or `"""`) for **docstrings or block comments**:

```
'''
This is a multi-line comment
spanning more than one line
'''
print("Python is fun!")
```

Note: This creates a string object, but it's often ignored by the interpreter if not assigned to a variable.

Best Practices for Comments

- Explain **why** something is done, not just **what**
- Avoid obvious comments like:

```
x = 10  # assigning 10 to x  ❌
```

- Instead:

```
x = 10  # Initial count for iteration ✅
```

2. Indentation in Python

Python **uses indentation to define blocks of code**, unlike other languages that use `{}`.

All statement with the same level of indentation are considered as part of the same block.

Example:

```
if True:
    print("This is inside the if block")
print("This is outside")
```

Output:

```
This is inside the if block
This is outside
```

Incorrect Indentation (will cause error)

```
if True:
print("Hello")   # ❌ IndentationError
```

Common Indentation Rules

- Use **4 spaces** (not tabs) per indent level (Python convention)
- Be consistent in one file (tabs OR spaces, not both)
- Indent after these structures:
    - if, elif, else
    - for, while
    - def, class
    - try, except, finally

Example: Indentation in a Function

```
def greet(name):
    if name:
        print("Hello,", name)
    else:
        print("Hello, Stranger")
```