📘 Module 34: Control Flow and Loops

🔰 1. Python Operators

Python provides various operators for performing operations on variables and values. Operators are categorized as follows:

🔷 Arithmetic Operators

Operator	Description	Example	Result
`+`	Addition	`5 + 2`	`7`
`-`	Subtraction	`5 - 2`	`3`
`*`	Multiplication	`5 * 2`	`10`
`/`	Division	`5 / 2`	`2.5`
`//`	Floor Division	`5 // 2`	`2`
`%`	Modulus	`5 % 2`	`1`
`**`	Exponentiation	`2 ** 3`	`8`

🔷 Assignment Operators

Operator	Description	Example	Meaning
`=`	Assign	`x = 5`	`x = 5`
`+=`	Add and assign	`x += 2`	`x = x + 2`
`-=`	Subtract and assign	`x -= 2`	`x = x - 2`
`*=`	Multiply and assign	`x *= 2`	`x = x * 2`
`/=`	Divide and assign	`x /= 2`	`x = x / 2`
`//=`	Floor divide and assign	`x //= 2`	`x = x // 2`

🔷 Comparison Operators

Operator	Description	Example	Result
`==`	Equal to	`x == 5`	`True/False`
`!=`	Not equal to	`x != 3`	`True/False`
`>`	Greater than	`x > 3`	`True/False`
`<`	Less than	`x < 3`	`True/False`
`>=`	Greater or equal	`x >= 3`	`True/False`
`<=`	Less or equal	`x <= 3`	`True/False`

🔷 Logical Operators

Operator	Description	Example	Result
`and`	True if both are True	`x > 2 and x < 10`	`True/False`
`or`	True if one is True	`x < 2 or x > 10`	`True/False`
`not`	Reverse the result	`not(x > 3)`	`True/False`

🔷 Identity Operators

Operator	Description	Example	Result
`is`	True if same object	`x is y`	`True/False`
`is not`	True if not same obj	`x is not y`	`True/False`

🔷 Membership Operators

Operator	Description	Example	Result
`in`	Checks if value in sequence	`"a" in "apple"`	`True`
`not in`	Checks if not in sequence	`"x" not in "box"`	`True`

📝 Commenting in Python

- **Single-line comment**: Use `#`

```
# This is a comment
```

- **Multi-line comment**: Use triple quotes (as docstring or block comment)

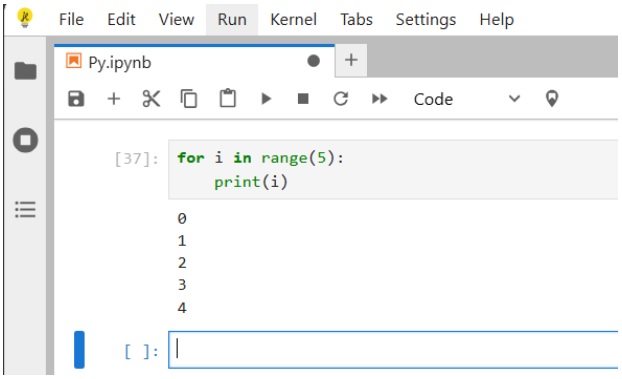
```
"""
This is a
multi-line comment
"""
```

🔁 2. Loops in Python

Python supports two types of loops: `for` and `while`.

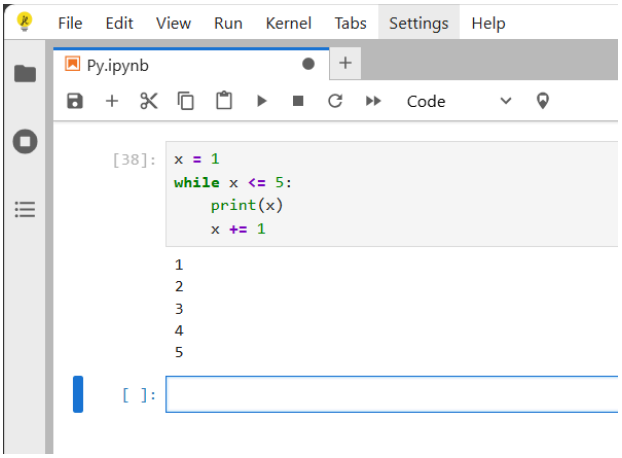
🔷 For Loop

Used for iterating over a sequence (list, string, range, etc.)



🔷 While Loop

Repeats as long as the condition is `True`.

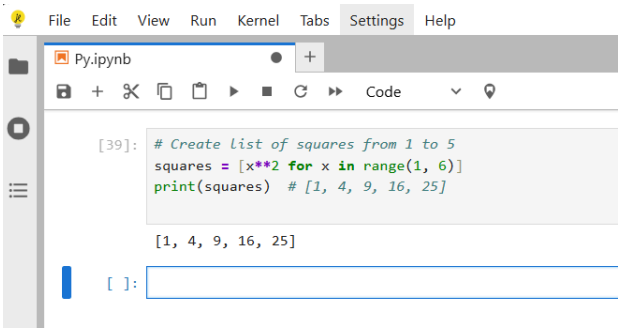


🔶 Loop Control Statements

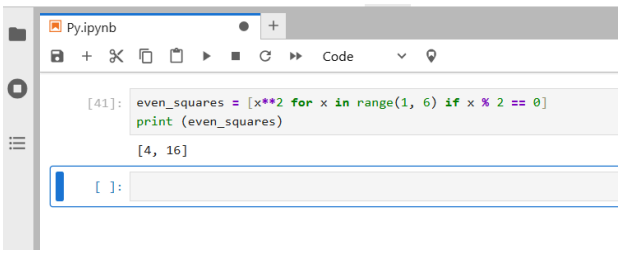
Statement	Description
`break`	Exits the loop immediately
`continue`	Skips current iteration
`pass`	Placeholder (does nothing)

🔄 3. List Comprehension

A concise way to create lists in a single line.



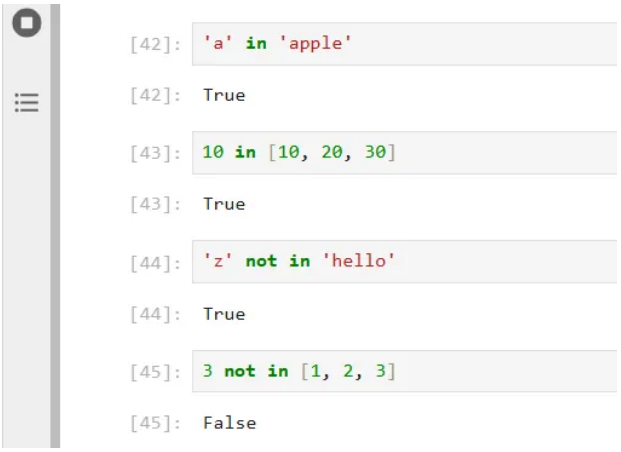
✅ You can also use conditions:



🔍 4. `in` and `not in`

Used to **check membership** in sequences (lists, strings, tuples, etc.)

🔷 Examples:



✅ Commonly used in **conditions and loops**.

📌 Key Points to Remember

- Operators are essential building blocks for any logic.
- Loops reduce repetition and automate tasks.
- List comprehension makes code **shorter** and **faster**.
- `in` and `not in` are **membership checks**, useful in conditionals.
- Comments are critical for making your code readable.
- Python's simplicity in control flow makes it ideal for beginners and professionals alike.
