➢ Basic Operators in Python

- Python has several types of operators that allow you to perform a variety of operations.

1. **Arithmetic Operators**
2. **Comparison/Relational Operators**
3. **Assignment Operators**
4. **Logical Operators**
5. **Bitwise Operators**
6. **Membership Operators**
7. **Identity Operators**

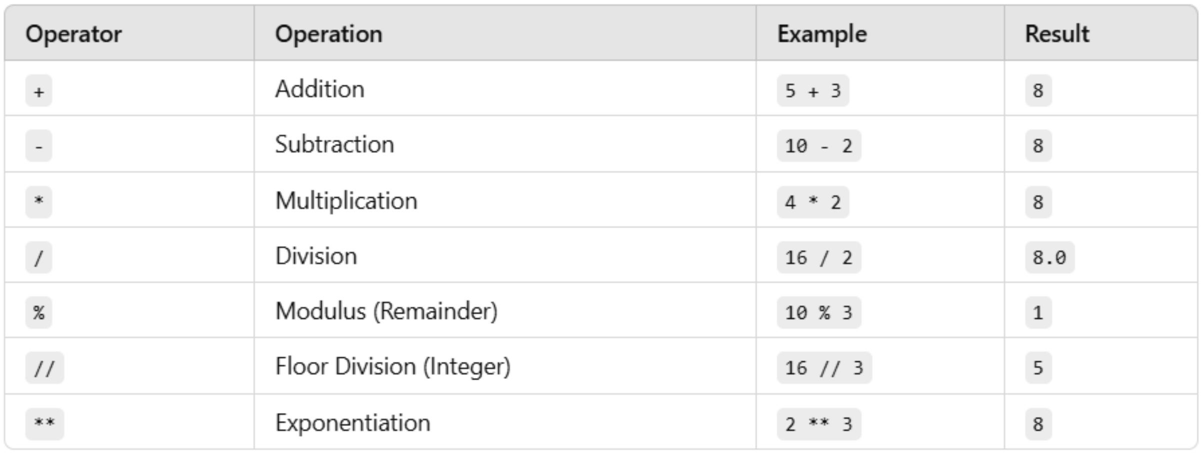
---

1. Arithmetic Operators

Definition

- Arithmetic operators perform basic mathematical operations like addition, subtraction, multiplication, etc.

Operators & Examples



Operator	Operation	Example	Result
`+`	Addition	`5 + 3`	`8`
`-`	Subtraction	`10 - 2`	`8`
`*`	Multiplication	`4 * 2`	`8`
`/`	Division	`16 / 2`	`8.0`
`%`	Modulus (Remainder)	`10 % 3`	`1`
`//`	Floor Division (Integer)	`16 // 3`	`5`
`**`	Exponentiation	`2 ** 3`	`8`

Example

```
# Arithmetic Example: Addition, Subtraction, Multiplication, Division
a = 10
b = 3
print("Addition:", a + b)        # Output: 13
print("Subtraction:", a - b)     # Output: 7
print("Multiplication:", a * b)  # Output: 30
print("Division:", a / b)        # Output: 3.333...
```

Marathi Fun Tip:

- “Arithmetic Operator म्हणजे बेरीज, वजाबाकी, गुणाकार आणणभागाकार सारख्या गणणर्तीणिया करय्णाचे साधन!”

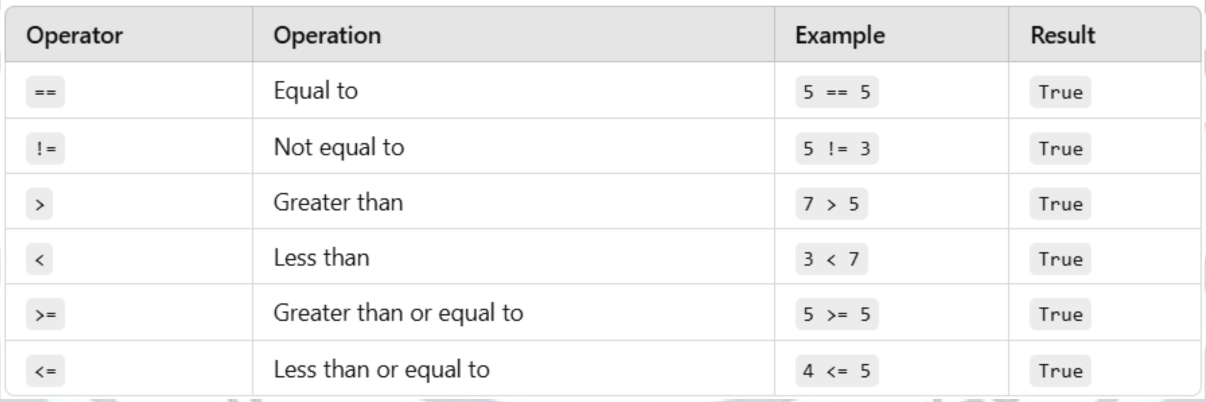
---

2. Comparison/Relational Operators

Definition

- Comparison operators are used to compare two values. They return True or False.

Operators & Examples



Operator	Operation	Example	Result
`==`	Equal to	`5 == 5`	`True`
`!=`	Not equal to	`5 != 3`	`True`
`>`	Greater than	`7 > 5`	`True`
`<`	Less than	`3 < 7`	`True`
`>=`	Greater than or equal to	`5 >= 5`	`True`
`<=`	Less than or equal to	`4 <= 5`	`True`

Example

```
# Comparison Example: Checking equality and greater than
x = 10
y = 5
print("x == y:", x == y)   # Output: False
print("x > y:", x > y)     # Output: True
```

Marathi Fun Tip:

- "Comparison Operator आपण दोन संख्या णकंवा मूल्◌ांची तुलना कूर शकतो. म्हणजे, 'बराबर आहे का?' णकंवा 'जात्स आहे का?' असे तपासणे!"

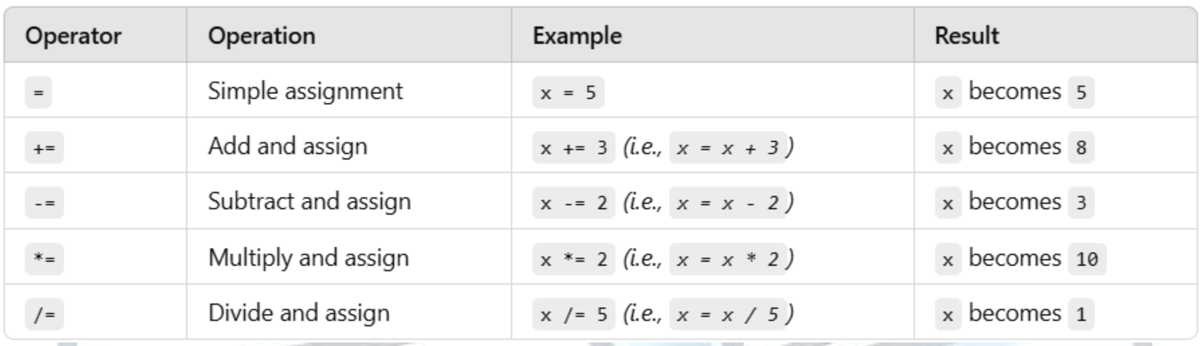
---

3. Assignment Operators

Definition

- Assignment operators are used to assign values to variables. They can also update the variable's value based on its current value.

Operators & Examples



Operator	Operation	Example	Result
`=`	Simple assignment	`x = 5`	`x` becomes `5`
`+=`	Add and assign	`x += 3` (i.e., `x = x + 3`)	`x` becomes `8`
`-=`	Subtract and assign	`x -= 2` (i.e., `x = x - 2`)	`x` becomes `3`
`*=`	Multiply and assign	`x *= 2` (i.e., `x = x * 2`)	`x` becomes `10`
`/=`	Divide and assign	`x /= 5` (i.e., `x = x / 5`)	`x` becomes `1`

Example

```
# Assignment Example: Using += operator
num = 10
num += 5  # Equivalent to num = num + 5
print("Updated num:", num)  # Output: 15
```

Marathi Fun Tip:

- " variable la नवीन मूल् देताना णकंवा अय्दयावतकरताना आपण Assignment Operators वापरतो. सोप्या भाषेत, 'हे मूल् दे' असे!"

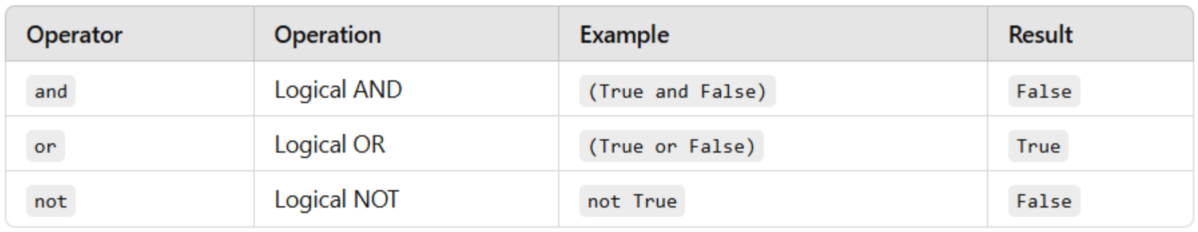
---

4. Logical Operators

Definition

- Logical operators combine conditional statements and return True or False.

Operators & Examples



Operator	Operation	Example	Result
`and`	Logical AND	`(True and False)`	`False`
`or`	Logical OR	`(True or False)`	`True`
`not`	Logical NOT	`not True`	`False`

Example

```
# Logical Example: Using 'and', 'or', and 'not'
a = True
b = False
print("a and b:", a and b)  # Output: False
print("a or b:", a or b)    # Output: True
print("not a:", not a)      # Output: False
```

Marathi Fun Tip:

“ Logical Operator आपण अनेक अटी एकत्र तपासू शकतो. जसं की ‘दोन्ही खरे असावेत’ किंवा ‘किमान एक खरं असावं’!”

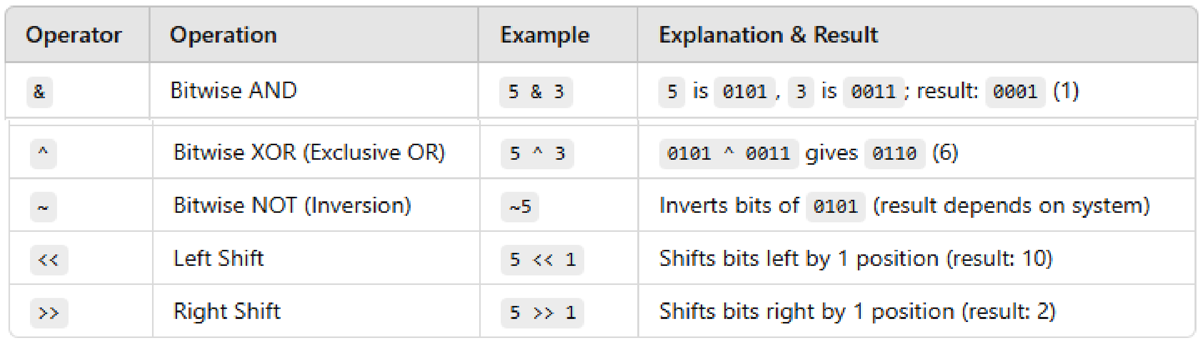
---

5. Bitwise Operators

Definition

- Bitwise operators work on bits (binary digits) of numbers.

Operators & Examples



Operator	Operation	Example	Explanation & Result
`&`	Bitwise AND	`5 & 3`	`5` is `0101`, `3` is `0011`; result: `0001` (1)
`^`	Bitwise XOR (Exclusive OR)	`5 ^ 3`	`0101 ^ 0011` gives `0110` (6)
`~`	Bitwise NOT (Inversion)	`~5`	Inverts bits of `0101` (result depends on system)
`<<`	Left Shift	`5 << 1`	Shifts bits left by 1 position (result: 10)
`>>`	Right Shift	`5 >> 1`	Shifts bits right by 1 position (result: 2)

Example

```
# Bitwise Example: Using AND (&) and OR (|)
x = 5   # In binary: 0101
y = 3   # In binary: 0011
print("Bitwise AND:", x & y)  # Output: 1 (binary: 0001)
print("Bitwise OR:", x | y)   # Output: 7 (binary: 0111)
```

Marathi Fun Tip:

- "Bitwise Operator म्हणजे संगणकाच्या आत्ला णबड्सवर काम करणारे ऑपरेटर. थोडं जादूचा खेळ समजून घ्या – णबड्स हलवणे!"

---

6. Membership Operators

Definition

- Membership operators test whether a value or variable is found in a sequence (like a string, list, or tuple).

Operators & Examples

Operator	Operation	Example	Result
`in`	Returns `True` if value is found	`"a" in "apple"`	`True`
`not in`	Returns `True` if value is not found	`"z" not in "apple"`	`True`

Example

```
# Membership Example: Checking if a letter is in a string
fruit = "apple"
print("'a' in fruit:", 'a' in fruit)      # Output: True
print("'z' not in fruit:", 'z' not in fruit)  # Output: True
```

Marathi Fun Tip:

- "Membership Operator आपण तपासू शकतो की एखादी गोट्ष्एखाद्याणलट्समध्ू/स्टिंगमधून आहे का. म्हणजे, 'असय्णाची तपासणी'!"

---

7. Identity Operators

Definition

- Identity operators compare the memory locations of two objects. They help you check if two variables point to the same object.

Operators & Examples

Operator	Operation	Example	Result
`is`	Returns `True` if both variables refer to the same object	`x is y` (if both `x` and `y` point to the same object)	`True` or `False` (depends on assignment)
`is not`	Returns `True` if both variables do not refer to the same object	`x is not y`	`True` or `False`

Example

```
# Identity Example: Checking if two variables refer to the same object
a = [1, 2, 3]
b = a  # b points to the same list as a
c = [1, 2, 3]
print("a is b:", a is b)      # Output: True
print("a is c:", a is c)      # Output: False
```

Marathi Fun Tip:

- "Identity Operator आपण तपासतो की दोन variable एकच वस्तू दाखवतात का, म्हणजे 'एकाच ओळखीचे' आहे का!"

➢ Python Operator Precedence (Paper la yet nhi just lakshat theva for knowledge)

- Understanding **operator precedence** helps us predict how expressions are evaluated in Python. It determines **which operator is executed first** in a complex expression.

1. What is Operator Precedence?

- When multiple operators are used in an expression, Python follows a specific order.
- **Example**:

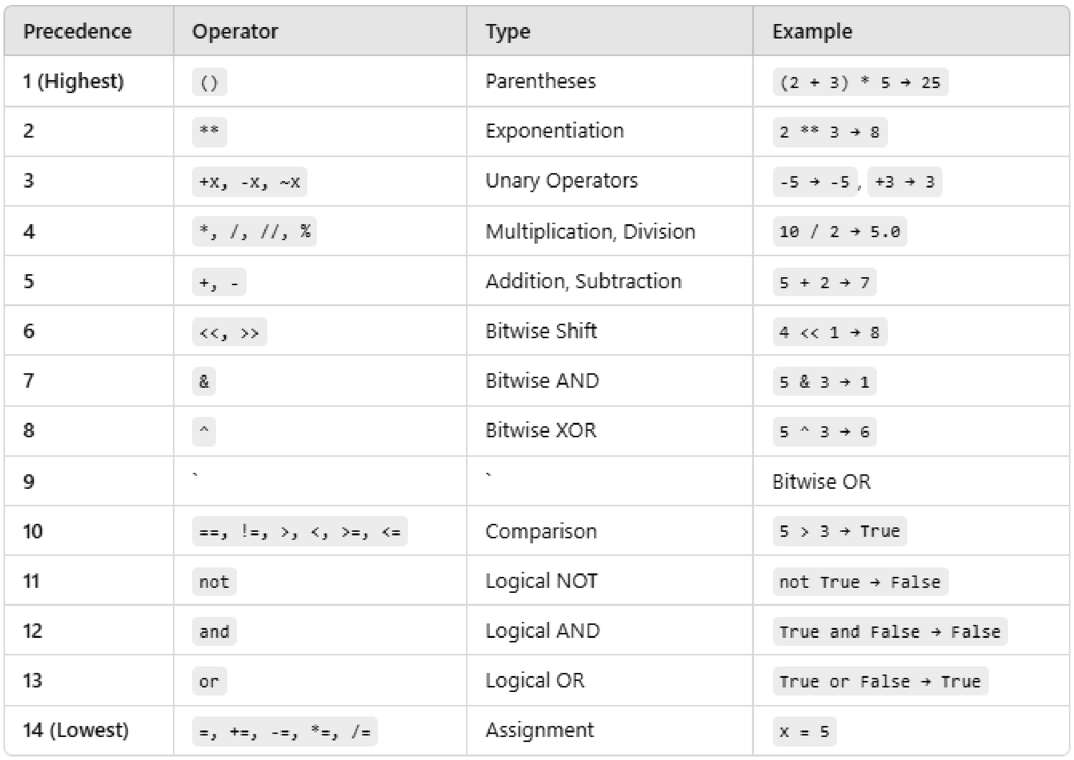
```
result = 10 + 5 * 2  # 5 * 2 executes first, then +10

print(result)  # Output: 20
```

- **Why?** Because multiplication (*) has higher precedence than addition (+).

---

2. Operator Precedence Table (Highest to Lowest)



Precedence	Operator	Type	Example
1 (Highest)	`()`	Parentheses	`(2 + 3) * 5 → 25`
2	`**`	Exponentiation	`2 ** 3 → 8`
3	`+x, -x, ~x`	Unary Operators	`-5 → -5`, `+3 → 3`
4	`*, /, //, %`	Multiplication, Division	`10 / 2 → 5.0`
5	`+, -`	Addition, Subtraction	`5 + 2 → 7`
6	`<<, >>`	Bitwise Shift	`4 << 1 → 8`
7	`&`	Bitwise AND	`5 & 3 → 1`
8	`^`	Bitwise XOR	`5 ^ 3 → 6`
9	`	`	Bitwise OR
10	`==, !=, >, <, >=, <=`	Comparison	`5 > 3 → True`
11	`not`	Logical NOT	`not True → False`
12	`and`	Logical AND	`True and False → False`
13	`or`	Logical OR	`True or False → True`
14 (Lowest)	`=, +=, -=, *=, /=`	Assignment	`x = 5`

<u>3. Summary & Quick Trick to Remember</u>

<u>Mnemonic:</u> *"P-E-U-M-D-A-B-C-L"!*

- **P**arentheses
- **E**xponentiation
- **U**nary Operators (+x, -x)
- **M**ultiplication/Division
- **D**ivision (//, %)
- **A**ddition/Subtraction
- **B**itwise Operators
- **C**omparison Operators
- **L**ogical Operators (not, and, or)

➢ Conditional Statements in Python

- Conditional statements are used to make decisions in a Python program based on conditions. Python provides the following conditional statements:

1. **if Statement**
2. **if-else Statement**
3. **Nested if Statement**
4. **if-elif-else Statement**

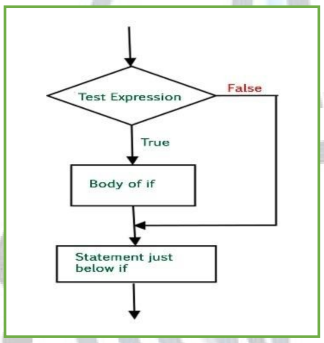
---

1. if Statement

Definition

- The if statement **executes a block of code only if the given condition is True**.

Diagram



Syntax

```
if condition:
    # Code to execute when condition is True
```

Example

```
age = 18
if age >= 18:
    print("You are eligible to vote!")
```

Output

```
You are eligible to vote!
```

Marathi Fun Tip

- "जर (if) म्हणजे 'जर काही अट खरी असेल, तर हे करा!' उदाहरणार्थ, जर वय १८ पेक्षा जास्त असेल तर तुम्ही मतदान करू शकता!"

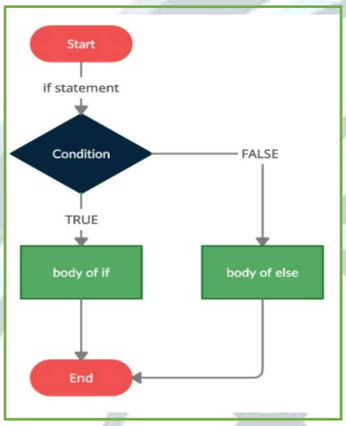
---

2. if-else Statement

Definition

- The if-else statement allows us to execute **one block of code if the condition is True, and another block if the condition is False**.

Diagram



Syntax

```
if condition:
   # Code to execute if condition is True
else:
   # Code to execute if condition is False
```

Example

```
marks = 35
if marks >= 40:
   print("You passed!")
else:
   print("You failed, better luck next time!")
```

Output

```
You failed, better luck next time!
```

Marathi Fun Tip

- "'जर..नाहीतर' (if-else) म्हणजे 'जर काही अट खरी असेल तर हे करा, नाहीतर दुसरं काही करा!' उदाहरणार्थ, जर गुण ४० पेक्षा जास्त असतील तर उत्तीर्ण, नाहीतर नापास."

3. nested if Statement

Definition

- A nested if statement **contains one or more if statements inside another if statement**. This is used when we need to check multiple conditions one after another.

Syntax

```
if condition1:
    if condition2:
        # Code to execute if both conditions are True
```

Example

```
num = 10
if num > 0:
    print("Positive number")
    if num % 2 == 0:
        print("Even number")
```

Output

```
Positive number
Even number
```

Marathi Fun Tip

- "'nested if' म्हणजे 'if च्या आत अजून if'! याचा उपयोग एकापेक्षा जास्त अटी तपासण्यासाठी होतो. उदाहरणार्थ, जर संख्या सकारात्मक असेल आणि जोडीने सम असेल तर दोन्ही संदेश छापले जातील!"

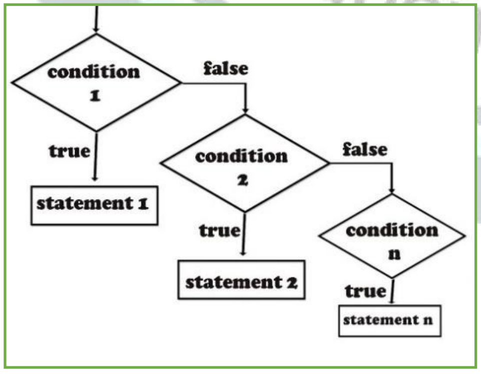
---

4. if...elif...else Statement

Definition

- This statement is used when there are **multiple conditions to check**.

Diagram



Syntax:

```
if condition1:
   # Executes if condition1 is True
elif condition2:
   # Executes if condition1 is False but condition2 is True
else:
   # Executes if all conditions are False
```

Example:

```
score = 75
if score >= 90:
   print("Grade: A")
elif score >= 75:
   print("Grade: B")
elif score >= 60:
   print("Grade: C")
else:
   print("Grade: D")
```

Marathi Tip:

- "जर पणहलीअट खरी नसेल, तर दुसरी तपासा, आणण्तेसुद्धा खोटी असेल, तर णतसरीउदा.: गुणांवूरन A, B, C णकंवा D ग्रेड ठरवा."

➢ Looping Statements in Python

- Loops in Python allow us to **repeat** a block of code multiple times. This helps in reducing repetitive tasks and making the code more efficient.

1. **while loop**
2. **for loop**
3. **Nested loop**

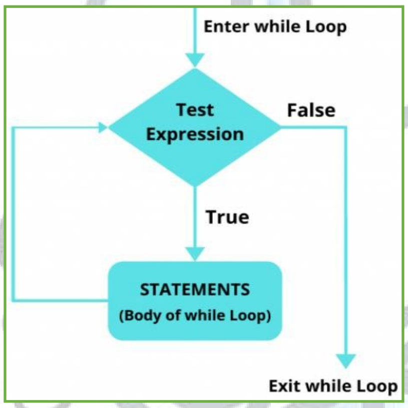
---

1. while Loop

Definition

- A while loop **executes a block of code as long as a condition is true**.
- The loop stops when the condition becomes **false**.

Diagram



Syntax

```
while condition:
    # Code block
```

Example

```
count = 1
while count <= 5:
    print("Number:", count)
    count += 1
```

Output:

```
Number: 1
Number: 2
Number: 3
Number: 4
Number: 5
```

Marathi Tip

- "while म्हणजे 'जोपर्यंत'. जोपर्यंत अट खरी आहे तोपर्यंत loop चालत राहतो!"

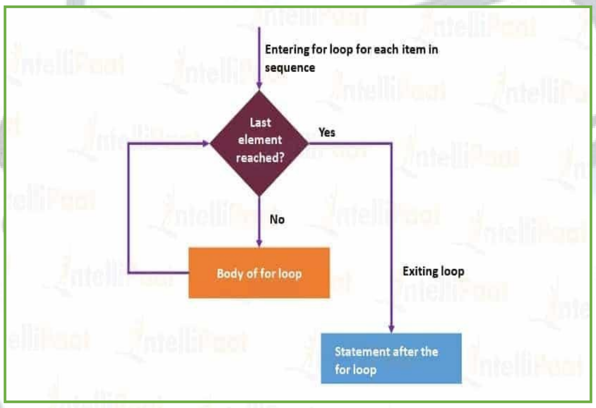
---

2. for Loop

Definition

- The for loop is used to **iterate over a sequence** (like a list, tuple, or string).
- It repeats the code **for each element** in the sequence.

Diagram


Entering for loop for each item in sequence
Last element reached?
Yes
No
Body of for loop
Exiting loop
Statement after the for loop

Syntax

```
for variable in sequence:
  # Code block
```

Example

```
fruits = ["Apple", "Banana", "Cherry"]
for fruit in fruits:
  print("I like", fruit)
```

Output:

```
I like Apple

I like Banana

I like Cherry
```

Using range() in for Loop

- The range() function generates numbers in a given range.

```
for num in range(1, 6):

    print(num)
```

Output:

```
1

2

3

4

5
```

Marathi Tip

- "for म्हणजे 'साठी'. List मधील प्रत्येक item साठी loop चालतो!"

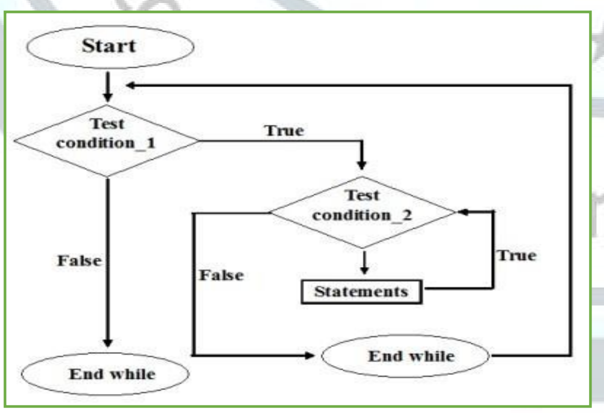
---

3. Nested Loops

Definition

- A **nested loop** is a loop inside another loop.
- The **inner loop executes completely for each iteration of the outer loop**.

Diagram



Syntax

```
for i in range(3):
    for j in range(2):
        # Code block
```

Example

```
for i in range(1, 4):
    for j in range(1, 3):
        print(f"Outer: {i}, Inner: {j}")
```

Output:

```
Outer: 1, Inner: 1
Outer: 1, Inner: 2
Outer: 2, Inner: 1
Outer: 2, Inner: 2
Outer: 3, Inner: 1
Outer: 3, Inner: 2
```

Marathi Tip

- "Nested म्हणजे 'आत्ला आत'. एका loop मध्ये अजून एक loop टाकला की Nested Loop तयार!"

➢ Loop Manipulation in Python

- Python provides special statements to **control the flow** inside loops. These are:

1. **continue** – Skips the current iteration and moves to the next.
2. **pass** – Does nothing, just a placeholder.
3. **break** – Exits the loop immediately.
4. **else** – Runs if the loop **completes normally** (without break).

---

1. The continue Statement

Definition

- The continue statement **skips** the current iteration of the loop and moves to the next one.

Example: Print numbers from 1 to 5 but skip 3

```
for i in range(1, 6):
   if i == 3:
      continue  # Skips when i is 3
   print(i)
```

Output:

```
1
2
4
5
```

Marathi Tip:

- "हे पुढे जा" म्हणतंय! जर काही पररट्सितीटाळायची असेल, तर continue वापरा. उदाहरणाथ्श३ वगळून १-५ िमांक छापतोय!

---

2. The pass Statement

Definition

- The pass statement **does nothing**. It is used when a statement is required syntactically but you don't want to execute anything.

Example: Placeholder for future code

```
for i in range(5):
  if i == 2:
    pass  # Does nothing
  else:
    print(i)
```

Output:

```
0
1
3
4
```

Marathi Tip:

- "मी आहे पण काही करत नाही!" – pass म्हणजे **फत्क जागा भरायला!**
  उदाहरणाथ्र ि◌मांक काही करत नाही, पण प्रोग्रॅम चालू राहतो!

3. The break Statement

Definition

- The break statement **immediately exits** the loop when encountered.

Example: Stop the loop when i is 3

```
for i in range(1, 6):
  if i == 3:
    break  # Exits loop when i is 3
  print(i)
```

Output:

```
1
2
```

Marathi Tip:

- "थांब! आता पुढे काहीच नाही!" – break लूप त्वररतथांबवतो.
  उदाहरणाथथ:३ वर आलो की लूप थांबतो!

---

4. The else Block with Loops

Definition

- In Python, a loop can have an **else block**.
- The else block runs **only if the loop completes normally** (without break).

Example: Check if a number is found in a list

```
numbers = [1, 2, 3, 4, 5]
for num in numbers:
  if num == 6:
    print("Number found!")
    break
else:
  print("Number not found!")
```

Output:

```
Number not found!
```

Marathi Tip:

- "लूप व्यवट्सितपूणथझाला की **else** चालतो!"
  उदाहरणाथथ:६ सापडत नाही म्हणून else चालतो.

➢ **Comparison Table:** continue vs pass vs break vs else

Statement	Action	When to Use?
`continue`	Skips the current iteration	To **skip** specific values
`pass`	Does nothing, just a placeholder	When the code is **not ready yet**
`break`	Stops the loop immediately	To **exit** when a condition is met
`else`	Runs if loop completes normally	To **check loop completion**

❖ UNIT 02. PYTHON OPERATORES AND CONTROL FLOW STATEMENT

❖ Summer 2022

1. List identity operators in python. **(2marks)**
2. Explain membership and assignment operators with example. **(4marks)**
3. Explain decision making statements If - else, if - elif - else with example. **(4marks)**

❖ Winter 2022

1. List comparision operators in Python. **(2marks)**
2. Describe bitwise operators in Python with example. **(4marks)**
3. Write python program to illustrate if else ladder. **(4marks)**

❖ Summer 2023

1. Describe membership operators in python. **(2marks)**
2. Describe Keyword "continue" with example. **(4marks)**
3. Explain Bitwise operator in Python with appropriate example. **(4marks)**

❖ Winter 2023

1. Write the use of elif keyword in python. **(2marks)**
2. Explain membership and Identity operators in Python. **(2marks)**
3. Explain use of Pass and Else keyword with for loops in python. **(4marks)**

❖ Summer 2024

1. Give membership operators in python. **(2marks)**

❖ Winter 2024

1. List membership operators in Python. **(2marks)**
2. Explain decision making statements if-else , if-elif-else with example. **(4marks)**
3. Explain identity and assignment operator with example. **(4marks)**
4. Explain loop control statement in Python. **(4marks)**