# Python:

## 1. Operators

Operators in Python are special symbols used to perform operations on variables and values.   
There are several types of operators in Python:  
  
**1. Arithmetic Operators:**

Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication, etc.

| **Operator** | **Description** | **Example** |
| --- | --- | --- |
| + | Addition | 5 + 3 = 8 |
| - | Subtraction | 10 - 4 = 6 |
| \* | Multiplication | 6 \* 2 = 12 |
| / | Division | 9 / 2 = 4.5 (float result) |
| // | Floor Division | 9 // 2 = 4 (integer result) |
| % | Modulus (Remainder) | 10 % 3 = 1 |
| \*\* | Exponentiation | 2 \*\* 3 = 8 |

**2. Comparison Operators:**

Comparison operators are used to compare two values. They return a Boolean result (True or False) based on the comparison.

| **Operator** | **Description** | **Example** | **Output** |
| --- | --- | --- | --- |
| == | Equal to | 5 == 5 | True |
| != | Not equal to | 5 != 3 | True |
| > | Greater than | 10 > 3 | True |
| < | Less than | 2 < 7 | True |
| >= | Greater than or equal to | 4 >= 4 | True |
| <= | Less than or equal to | 6 <= 10 | True |

**3. Logical Operators**

Logical operators are used to combine conditional statements and return True or False based on conditions.

| **Operator** | **Description** | **Example** | **Output** |
| --- | --- | --- | --- |
| and | Returns True if both conditions are True | (5 > 2) and (10 > 5) | True |
| or | Returns True if at least one condition is True | (5 > 2) or (10 < 5) | True |
| not | Reverses the logical state | not(5 > 2) | False |

**4. Bitwise Operators**

Bitwise operators perform operations on binary numbers.

| **Operator** | **Description** | **Example (a = 5, b = 3)** | **Output** |
| --- | --- | --- | --- |
| & | Bitwise AND | 5 & 3 (0101 & 0011) | 1 (0001) |
| ` | ` | Bitwise OR | `5 |
| ^ | Bitwise XOR | 5 ^ 3 (0101 ^ 0011) | 6 (0110) |
| ~ | Bitwise NOT | ~5 (~0101) | -6 |
| << | Left Shift | 5 << 1 | 10 |
| >> | Right Shift | 5 >> 1 | 2 |

**5. Assignment Operators**

Assignment operators are used to assign values to variables.

| **Operator** | **Example** | **Equivalent To** | **Output (x=10)** |
| --- | --- | --- | --- |
| = | x = 5 | x = 5 | 5 |
| += | x += 3 | x = x + 3 | 13 |
| -= | x -= 2 | x = x - 2 | 8 |
| \*= | x \*= 2 | x = x \* 2 | 20 |
| /= | x /= 2 | x = x / 2 | 5.0 |
| %= | x %= 3 | x = x % 3 | 1 |
| \*\*= | x \*\*= 2 | x = x \*\* 2 | 100 |
| //= | x //= 3 | x = x // 3 | 3 |

**6. Identity Operators**

Identity operators are used to compare the memory locations of two objects.

| **Operator** | **Description** | **Example** | **Output** |
| --- | --- | --- | --- |
| is | Returns True if two variables point to the same object | x is y | False |
| is not | Returns True if two variables point to different objects | x is not y | True |

**7. Membership Operators**

Membership operators check if a value is present in a sequence (like lists, tuples, or strings).

| **Operator** | **Description** | **Example** | **Output** |
| --- | --- | --- | --- |
| in | Returns True if a value exists in a sequence | "a" in "apple" | True |
| not in | Returns True if a value does not exist in a sequence | "z" not in "apple" | True |

**2. Data Types**

**Definition**: A data type is an attribute that specifies the type of data that a variable can hold.

**Importance**: Helps the program know what type of operations can be performed on data.  
**1. Numeric Types:**

**Integer (int)**

Whole numbers, positive or negative (e.g., 10, -3).

**Float (float)**

Numbers with decimal points (e.g., 3.14, -0.5).

**Complex (complex)**

Complex numbers which contains real and imaginary part (e.g., 1 + 2j).

**2. Sequence Types:**

**String (str)**

Ordered sequence of characters, which can be written in between single quotes and double quotes (e.g., "hello").

**List (list)**

Ordered and mutable collection, which can be written in Square Brackets

(e.g., [1, 2, 'apple']).

**Tuple (tuple)**

Ordered and immutable collection, which can be written in parathesis (e.g.,(1, 2, 3)).

**3. Set Types:**

**Set (set)**

Unordered collection of unique elements (e.g., {1, 2, 3}).

**Frozenset (frozenset)**

Immutable version of a set.

**4. Mapping Type:**

**Dictionary (dict)**

Unordered collection of key-value pairs (e.g., {'name': 'Alice', 'age': 25}).

**5. Boolean Type:**

**Boolean (bool)**

Represents True if condition is true or else it represents False.

**6. Binary Types:**

**Bytes (bytes)**

Immutable sequence of bytes (e.g., b'hello').

**Bytearray (bytearray)**

Mutable sequence of bytes.

**Memoryview (memoryview)**

Allows access to the internal data of an object.

**7.None Type:**

Represents the absence of value (None)

Example:

a = 10 # Integer  
b = 3.14 # Float  
c = "Python" # String  
d = [1, 2, 3] # List  
e = (4, 5, 6) # Tuple  
f = {7, 8, 9} # Set  
g = {"name": "Umesh", "age": 24} # Dictionary

## 3. Variables

Variables are used to store data in Python. They are dynamically typed and do not need explicit declaration.  
  
Example:  
x = 100  
y = "Hello"  
z = [1, 2, 3]  
print(x, y, z)

## 4. Loops

Python supports two types of loops:  
  
1. for loop: Iterates over a sequence (list, tuple, string, etc.).  
2. while loop: Repeats as long as a condition is true.

For Loop Example:

for i in range(5):  
 print(i) # Prints 0 to 4

While Loop Example:

count = 0  
while count < 5:  
 print(count)  
 count += 1 #It will execute until the count is less than 5

#prints 0 to 4

## 5. Conditional Statements

Conditional statements allow decision-making in Python using if, elif, and else.  
  
Example:  
  
age = 18  
if age >= 18:  
 print("You are an adult.")  
elif age > 13:  
 print("You are a teenager.")  
else:  
 print("You are a child.")

#if block will execute

#you are an adult