**Operators :**

**1. Arithmetic Operators**

Arithmetic operators are used for performing basic mathematical operations. **Addition ( + )**: Adds two operands.

x = 5 + 3

print(x) # Output: 8

**Subtraction ( )**: Subtracts the right operand from the left.

y = 10 - 4

print(y) # Output: 6

**Multiplication ( )**: Multiplies two operands.

z = 3 \* 7

print(z) # Output: 21

**Division ( / )**: Divides the left operand by the right. Always returns a float.

a = 20 / 4

print(a) # Output: 5.0

Operators : 1

**Floor Division ( // )**: Divides the left operand by the right and returns the largest integer less than or equal to the result.

b = 17 // 5

print(b) # Output: 3

**Modulus ( % )**: Returns the remainder when the left operand is divided by the right.

c = 17 % 5

print(c) # Output: 2

**Exponentiation ( \* )**: Raises the left operand to the power of the right operand.

d = 2 \*\* 3

print(d) # Output: 8

**2. Comparison Operators**

Comparison operators are used to compare two values. The result is always a boolean ( True or False ).

**Equal to ( == )**: Checks if the values of two operands are equal. print(5 == 5) # Output: True

**Not equal to ( != )**: Checks if the values of two operands are not equal.

Operators : 2

print(5 != 6) # Output: True

**Greater than ( > )**: Checks if the left operand is greater than the right. print(7 > 3) # Output: True

**Less than ( < )**: Checks if the left operand is less than the right. print(2 < 8) # Output: True

**Greater than or equal to ( >= )**: Checks if the left operand is greater than or equal to the right.

print(5 >= 5) # Output: True

**Less than or equal to ( <= )**: Checks if the left operand is less than or equal to the right.

print(4 <= 3) # Output: False

**3. Logical Operators**

Logical operators are used to combine conditional statements and return boolean values ( True or False ).

Operators : 3

**and** : Returns True if both statements are True .

x = 5

print(x > 3 and x < 10) # Output: True

**or** : Returns True if at least one statement is True .

y = 12

print(y < 5 or y > 10) # Output: True

**not** : Reverses the result; returns True if the result is False .

z = False

print(not z) # Output: True

**4. Assignment Operators**

Assignment operators are used to assign values to variables, often combining arithmetic and assignment in one operation.

**=** : Assigns the value on the right to the variable on the left.

a = 5

print(a) # Output: 5

**+=** : Adds the right operand to the left operand and assigns the result to the left operand.

Operators : 4

b = 10

b += 3

print(b) # Output: 13

**=** : Subtracts the right operand from the left operand and assigns the result to the left operand.

c = 8

c -= 2

print(c) # Output: 6

**=** : Multiplies the left operand by the right operand and assigns the result to the left operand.

d = 4

d \*= 3

print(d) # Output: 12

**/=** : Divides the left operand by the right operand and assigns the result to the left operand.

e = 15

e /= 3

print(e) # Output: 5.0

**//=** : Performs floor division on the left operand by the right operand and assigns the result to the left operand.

Operators : 5

f = 17

f //= 5

print(f) # Output: 3

**%=** : Takes the modulus of the left operand by the right operand and assigns the result to the left operand.

g = 18

g %= 5

print(g) # Output: 3

**\*=** : Raises the left operand to the power of the right operand and assigns the result to the left operand.

h = 2

h \*\*= 3

print(h) # Output: 8

**5. Bitwise Operators**

Bitwise operators are used to perform bitwise calculations on binary representations of integers.

**& (Bitwise AND)**: Compares each bit of the left operand with the corresponding bit of the right operand. The result is 1 if both bits are 1 .

x = 5 # 101 in binary

y = 3 # 011 in binary

Operators : 6

print(x & y) # Output: 1 (001 in binary)

**| (Bitwise OR)**: Compares each bit of the left operand with the corresponding bit of the right operand. The result is 1 if at least one bit is 1 .

print(x | y) # Output: 7 (111 in binary)

**^ (Bitwise XOR)**: Compares each bit of the left operand with the corresponding bit of the right operand. The result is 1 if the bits are different.

print(x ^ y) # Output: 6 (110 in binary)

**~ (Bitwise NOT)**: Inverts all the bits of the operand.

print(~x) # Output: -6

**<< (Left Shift)**: Shifts the bits of the left operand to the left by the number of positions specified by the right operand, effectively multiplying by powers of 2.

print(x << 1) # Output: 10 (1010 in binary)

**>> (Right Shift)**: Shifts the bits of the left operand to the right by the number of positions specified by the right operand, effectively dividing by powers of 2.

Operators : 7

print(x >> 1) # Output: 2 (10 in binary)

**6. Identity Operators**

Identity operators are used to compare the memory locations of two objects. **is** : Returns True if both variables point to the same object in memory.

a = [1, 2, 3]

b = [1, 2, 3]

c = a

print(a is c) # Output: True

print(a is b) # Output: False

**is not** : Returns True if both variables do not point to the same object in memory.

print(a is not b) # Output: True

**7. Membership Operators**

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

**in** : Returns True if the specified value is found in the sequence.

fruits = ["apple", "banana", "cherry"]

print('banana' in fruits) # Output: True

Operators : 8

**not in** : Returns True if the specified value is not found in the sequence. print('orange' not in fruits) # Output: True

Operators : 9