**Python Operators:**

Operators refer to special symbols that perform operations on values and variables. Furthermore, the operands in python, one of the programming languages, refer to the values on which the operator operates. Most noteworthy, operators can carry out arithmetic, relational, and logical operations.

**1. Arithmetic Operators:**

Arithmetic operators perform mathematical calculations on numeric operands.

* **Addition (+):** Adds two operands.

x = 5

y = 2

sum = x + y # sum = 7

* **Subtraction (-):** Subtracts the second operand from the first.

difference = x - y # difference = 3

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

product = x \* y # product = 10

* **Division (/):** Divides the first operand by the second, always resulting in a float.

quotient = x / y # quotient = 2.5

* **Floor Division (//):** Divides the first operand by the second, discarding the fractional part and returning an integer.

floor quotient = x // y # floor quotient = 2

* **Modulus (%):** Returns the remainder of the division of the first operand by the second.

remainder = x % y # remainder = 1

* **Exponentiation (**):\*\* Raises the first operand to the power of the second.

power = x \*\* y # power = 25

**2. Comparison Operators:**

Comparison operators compare two values and return a boolean result (True or False).

**Equal to (==):** Checks if two operands are equal.

is\_equal = (x == y) # is\_equal = False

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

is\_not\_equal = (x != y) # is\_not\_equal = True

* **Greater than (>):** Checks if the first operand is greater than the second.

is\_greater = (x > y) # is\_greater = True

* **Less than (<):** Checks if the first operand is less than the second.

is\_less = (x < y) # is\_less = False

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

is\_greater\_or\_equal = (x >= y) # is\_greater\_or\_equal = True

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

is\_less\_or\_equal = (x <= y) # is\_less\_or\_equal = False

**3. Assignment Operators:**

Assignment operators assign values to variables.

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

z = 10

* **Addition and assignment (+=):** Adds the right operand to the left operand and assigns the result to the left operand.

z += 5 # z = z + 5; z becomes 15

* **Subtraction and assignment (-=):** Subtracts the right operand from the left operand and assigns the result to the left operand.

z -= 3 # z = z - 3; z becomes 12

* \**Multiplication and assignment (=):* Multiplies the left operand by the right operand and assigns the result to the left operand.

z \*= 2 # z = z \* 2; z becomes 24

* **Division and assignment (/=):** Divides the left operand by the right operand and assigns the result to the left operand.

z /= 4 # z = z / 4; z becomes 6.0

* **Floor division and assignment (//=):** Performs floor division and assigns the result to the left operand.

z //= 2 # z = z // 2; z becomes 3.0

* **Modulus and assignment (%=):** Performs modulus and assigns the result to the left operand.

z %= 5 # z = z % 5; z becomes 3.0

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

z \*\*= 2 # z = z \*\* 2; z becomes 9.0

**4. Logical Operators:**

Logical operators combine or modify boolean values.

* **and:** Returns True if both operands are true.

a = True

b = False

result = a and b # result = False

* **or:** Returns True if at least one operand is true.

result = a or b # result = True

* **not:** Inverts the boolean value.

result = not a # result = False

**5. Bitwise Operators:**

Bitwise operators perform operations on individual bits of data.

* **Bitwise AND (&):** Performs a bitwise AND operation.

num1 = 5 # Binary: 0101

num2 = 3 # Binary: 0011

result = num1 & num2 # result = 1 (Binary: 0001)

* **Bitwise OR (|):** Performs a bitwise OR operation.

result = num1 | num2 # result = 7 (Binary: 0111)

* **Bitwise XOR (^):** Performs a bitwise XOR operation.

result = num1 ^ num2 # result = 6 (Binary: 0110)

* **Bitwise NOT (~):** Inverts the bits.

result = ~num1 # result = -6 (Two's complement)

* **Left shift (<<):** Shifts the bits to the left.

result = num1 << 2 # result = 20 (Binary: 10100)

* **Right shift (>>):** Shifts the bits to the right.

result = num1 >> 1 # result = 2 (Binary: 0010)

**6. Membership Operators:**

Membership operators check if a value is a member of a sequence (like a string, list, or tuple).

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

my\_list = [1, 2, 3, 4, 5]

is\_present = 3 in my\_list # is\_present = True

my\_string = "hello"

is\_char\_present = 'l' in my\_string # is\_char\_present = True

* **not in:** Returns True if the value is not found in the sequence.

is\_not\_present = 6 not in my\_list # is\_not\_present = True

**7. Identity Operators:**

Identity operators check if two variables refer to the same object in memory.

* **is:** Returns True if both variables refer to the same object.

x = [1, 2, 3]

y = x # y and x now refer to the same object

z = [1, 2, 3] # z is a different object, even with same content

result = x is y # result = True

result = x is z # result = False

* **is not:** Returns True if both variables do not refer to the same object.

result = x is not z # result = True