**1. Arithmetic Operators**

**Definition:** These are used to perform standard mathematical operations.

* + (Addition): 20 + 2 = 22
* - (Subtraction): 20 - 2 = 18
* \* (Multiplication): 20 \* 2 = 40
* / (Float Division): 20 / 2 = 10.0 (always returns a float)
* // (Floor Division): 20 // 2 = 10 (returns an integer, discarding the remainder)
* % (Modulo): 20 % 2 = 0 (returns the remainder of the division)

**2. Comparison Operators**

**Definition:** These operators compare two values and return a boolean (True or False).

* == (Equal to): x == y
* != (Not equal to): x != y
* > (Greater than): y > x
* < (Less than): x < y
* >= (Greater than or equal to): y >= x
* <= (Less than or equal to)

**Example:**

x = 20

y = 22

print(x > y) # Output: False

print(y > x) # Output: True

print((x + 2) == y) # Output: True

**Note on Sets:** You cannot use + to combine sets. Use the | (union) operator.

l = {1, 2}

m = {3, 4}

# print(l + m) # This causes a TypeError

print((l | m) == {1, 2, 3, 4}) # Output: True

Here's a simpler, more organized way to think about logical operators.

**3. Logical Operators**

**Definition:** These operators are used to combine or reverse True and False values.

**and**

* **What it does:** Returns True only if **both** sides are true.
* **Simple Example:**

Python

print(True and True) # Output: True

print(True and False) # Output: False

* **Code Example:**

Python

a = 20

b = 30

# (a != b) is True

# not(a == b) is True

# Since both sides are True, the result is True.

print((a != b) and not(a == b)) # Output: True

**or**

* **What it does:** Returns True if **at least one** side is true.
* **Simple Example:**

Python

print(True or False) # Output: True

print(False or False) # Output: False

**not**

* **What it does:** Flips the value. True becomes False, and False becomes True.
* **Simple Example:**

Python

print(not True) # Output: False

print(not False) # Output: True

**4. Assignment Operators**

**Definition:** These operators are used to assign or update the value of a variable.

**= (Basic Assignment)**

This is the simplest assignment operator. It gives a variable its initial value or overwrites its old value.

**Example:**

# Assigns the value 20 to the variable 'a'

a = 20

print(a)

**Output:**

20

**+= (Add and Assign)**

This adds the value on the right to the variable's current value and then assigns the result back to the variable.

**Example:**

# Start with a = 20

a = 20

# Add 2 to 'a' (a becomes 20 + 2)

a += 2

print(a)

**Output:**

22

**-= (Subtract and Assign)**

This subtracts the value on the right from the variable's current value and assigns the result back to the variable.

**Example:**

# Start with a = 20

a = 20

# Subtract 2 from 'a' (a becomes 20 - 2)

a -= 2

print(a)

**Output:**

18

**\*= (Multiply and Assign)**

This multiplies the variable's current value by the value on the right and assigns the result back to the variable.

**Example:**

# Start with a = 20

a = 20

# Multiply 'a' by 2 (a becomes 20 \* 2)

a \*= 2

print(a)

**Output:**

40

**/= (Divide and Assign)**

This divides the variable's current value by the value on the right and assigns the result (a float) back to the variable.

**Example:**

# Start with a = 20

a = 20

# Divide 'a' by 2 (a becomes 20 / 2)

a /= 2

print(a)

**Output:**

10.0

**//= (Floor Divide and Assign)**

This divides the variable's current value by the value on the right, drops the decimal part (floors), and assigns the integer result back to the variable.

**Example:**

# Start with a = 20

a = 20

# Floor divide 'a' by 2 (a becomes 20 // 2)

a //= 2

print(a)

**Output:**

10

**%= (Modulo and Assign)**

This calculates the remainder of dividing the variable's current value by the value on the right and assigns that remainder back to the variable.

**Example:**

# Start with a = 20

a = 20

# Get the remainder of 'a' divided by 2 (a becomes 20 % 2)

a %= 2

print(a)

**Output:**

0

**5. Identity Operators**

**Definition:** These operators check if two variables point to the **exact same object** in memory.

**is vs. == (A Very Important Concept) 💡**

This is a common point of confusion. Here's the simple difference:

* **== (Equality):** Checks if the **values** are the same.
  + *Analogy:* "Do these two $10 bills have the same value?" **Yes.**
* **is (Identity):** Checks if they are the **exact same object** in memory.
  + *Analogy:* "Are these two $10 bills the *exact same physical bill*?" **No.**

**The Operators**

* **is**: Returns True only if both variables are the exact same object.
* **is not**: Returns True if they are two different objects (even if their values are the same).

**Example**

In this example, a and b are like two identical-looking books, but they are separate copies. They have the same content (value), but they are not the same single physical object (identity).

# 'a' is one tuple object in memory

a = (1, 2)

# 'b' is a \*new\*, separate tuple object in memory

b = (1, 2)

# 1. Check for VALUE (==)

# Do 'a' and 'b' have the same value? Yes.

print(f"a == b: {a == b}")

# 2. Check for IDENTITY (is)

# Are 'a' and 'b' the exact same object in memory? No.

print(f"a is b: {a is b}")

# 3. Check for LACK OF IDENTITY (is not)

# Is it true that 'a' and 'b' are different objects? Yes.

print(f"a is not b: {a is not b}")

**Output:**

a == b: True

a is b: False

a is not b: True

**6. Membership Operators**

**Definition:** These operators check if a specific value exists within a sequence (like a list, tuple, or string). 🧐

**The Operators**

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

**Example 1: Simple Membership**

This is the most common use. You check if an item is inside a collection.

a = (1, 2)

# Is the number 2 found in the tuple 'a'? Yes.

print(f"2 in a: {2 in a}")

# Is it true that 6 is not in 'a'? Yes.

print(f"6 not in a: {6 not in a}")

**Output:**

2 in a: True

3 in a: False

6 not in a: True

**Example 2: Nested Membership**

This checks if an *entire object* (like another tuple) exists as a *single element* inside a larger collection.

a = (1, 2)

x = [1, 2, (1, 2)] # The list 'x' has 3 elements: 1, 2, and the tuple (1, 2)

# We are checking if the \*entire tuple 'a'\*

# exists as one of the elements in 'x'. It does (it's the third element).

print(f"a in x: {a in x}")

**Output:**

a in x: True

**Example 3: Common Pitfall**

This shows the important difference between an *element* and the *collection itself*.

a = (1, 2)

# This asks: "Is the tuple (1, 2) an element \*inside\* the tuple (1, 2)?"

# The answer is no. The elements of 'a' are the \*integers\* 1 and 2,

# not the tuple (1, 2).

print(f"(1, 2) in a: {(1, 2) in a}")

**Output:**

(1, 2) in a: False

**⚠️ Important: TypeError**

The in operator needs a sequence (an "iterable") on the **right** side. You cannot check for membership inside a single, non-iterable item like an integer.

# This code will fail

# print(6 not in 6)

**Output:**

TypeError: argument of type 'int' is not iterable

**Basic Algorithms**

**1. Sequential Algorithm**

**Example 1: Simple Calculation** This code calculates 20 + 30 and prints the result.

a = 20

b = 30

c = a + b

print(c)

**Output:**

50

**Example 2: Using User Input** This code asks the user for two numbers, adds them, and prints the sum. The output will change depending on what the user types.

a = int(input("Enter a number"))

b = int(input("Enter a number"))

c = a + b

print(c)

**Example Output (from your notes):**

Enter a number45

Enter a number24

69

**2. Conditional Algorithm**

**Example 1: Age Check** This code asks for an age. If the age is 18 or more, it prints "Yes"; otherwise, it prints "Not".

a = int(input("Enter age "))

if a >= 18:

print("Yes")

else:

print("Not")

**Example Output (from your notes, for an input of 14):**

Enter age 14

Not

**Example 2: Password Check** This code asks for a password. If it exactly matches "@qytpoIhd458", it prints a welcome message; otherwise, it prints a "thanks" message.

a = str(input("Enter Password "))

if a == "@qytpoIhd458":

print("Welcome to the application")

else:

print("Thanks for visiting")

**Example Output (from your notes, for a correct password):**

Enter Password @qytpoIhd458

Welcome to the application