1. Q1.

Here is the code to perform arithmetic operations on two integer variables and print the results:

# Define two integer variables

var1 = 5

var2 = 3

# Perform arithmetic operations

addition = var1 + var2

subtraction = var1 - var2

multiplication = var1 \* var2

division = var1 / var2

# Print the results

print("First variable is", var1, "& second variable is", var2)

print("Addition:", var1, "+", var2, "=", addition)

print("Subtraction:", var1, "-", var2, "=", subtraction)

print("Multiplication:", var1, "\*", var2, "=", multiplication)

print("Division:", var1, "/", var2, "=", division)

Output:

vbnet

Copy code

First variable is 5 & second variable is 3.

Addition: 5 + 3 = 8

Subtraction: 5 - 3 = 2

Multiplication: 5 \* 3 = 15

Division: 5 / 3 = 1.6666666666666667

1. Q2.

The differences between the mentioned operators are:

* + / performs regular division and returns a floating-point result, while // performs floor division and returns an integer result by discarding the decimal part.
  + \*\* is the exponentiation operator used to raise a number to a power, while ^ is not a valid exponentiation operator in Python. Instead, the ^ operator is used for bitwise XOR operation.

1. Q3.

The logical operators in Python are:

and: Returns True if both operands are True, otherwise False.

or: Returns True if at least one operand is True, otherwise False.

not: Returns the opposite Boolean value of the operand (True becomes False, and vice versa).

1. Q4.

The right shift (>>) and left shift (<<) operators are bitwise shift operators in Python.

Right Shift (>>): Shifts the bits of a number to the right by a specified number of positions. The vacant positions are filled with zeros. Each shift to the right is equivalent to dividing the number by 2.

Example: 10 >> 2 shifts the binary representation of 10 (1010) two positions to the right, resulting in 0010, which is equal to 2 in decimal.

Left Shift (<<): Shifts the bits of a number to the left by a specified number of positions. The vacant positions on the right are filled with zeros. Each shift to the left is equivalent to multiplying the number by 2.

Example: 10 << 2 shifts the binary representation of 10 (1010) two positions to the left, resulting in 101000, which is equal to 40 in decimal.

1. Q5.

Here is the code to check if 10 is present in a list:

my\_list = [1, 2, 3, 4, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55]

if 10 in my\_list:

print("10 is present in the list.")

else:

print("10 is not present in the list.")

Output:

10 is present in the list.