**Q.1. Create two int type variables, apply addition, subtraction, division and multiplications**

**and store the results in variables. Then print the data in the following format by calling the**

**variables:**

**First variable is \_\_ & second variable is \_\_.**

**Addition: \_\_ + \_\_ = \_\_**

**Subtraction: \_\_ - \_\_ = \_\_**

**Multiplication: \_\_ \* \_\_ = \_\_**

**Division: \_\_ / \_\_ = \_\_**

ANS Certainly! Here's an example of how you can create two int type variables, perform the specified arithmetic operations, store the results, and then print them in the desired format:

first\_variable = 10

second\_variable = 5

addition\_result = first\_variable + second\_variable

subtraction\_result = first\_variable - second\_variable

multiplication\_result = first\_variable \* second\_variable

division\_result = first\_variable / second\_variable

print("First variable is", first\_variable, "& second variable is", second\_variable)

print("Addition:", first\_variable, "+", second\_variable, "=", addition\_result)

print("Subtraction:", first\_variable, "-", second\_variable, "=", subtraction\_result)

print("Multiplication:", first\_variable, "\*", second\_variable, "=", multiplication\_result)

print("Division:", first\_variable, "/", second\_variable, "=", division\_result)

Make sure to replace the values of first\_variable and second\_variable with the integers of your choice. When you run this code, it will calculate the results of the arithmetic operations and print them in the format you specified.

**Q.2. What is the difference between the following operators:**

**(i) ‘/’ & ‘//’**

**(ii) ‘\*\*’ & ‘^’**

ANS; (i) '/' and '//' Operators:

'/' Operator (Division): The '/' operator is used for division between two numbers. It returns the result of the division as a floating-point number, even if the operands are integers.

'//' Operator (Floor Division): The '//' operator is used for floor division between two numbers. It returns the largest integer that is less than or equal to the result of the division. This result is also called the quotient. If both operands are integers, the result will be an integer.

Here's an example to illustrate the difference:

a = 10

b = 3

result\_division = a / b

result\_floor\_division = a // b

print("Division:", result\_division)

print("Floor Division:", result\_floor\_division)

(ii) '' Operator (Exponentiation) and '^' Operator:\*\*

'' Operator (Exponentiation):\*\* The '\*\*' operator is used for exponentiation, which means raising a number to a certain power.

'^' Operator: The '^' operator is not a built-in exponentiation operator in Python. In some programming languages, it might be used for this purpose, but in Python, it does not represent exponentiation.

Here's an example using the '\*\*' operator:

x = 2

y = 3

result\_exponentiation = x \*\* y

print("Exponentiation:", result\_exponentiation)

To summarize:

'/' performs regular division, returning a float.

'//' performs floor division, returning an integer quotient.

'\*\*' performs exponentiation.

**Q.3. List the logical operators.**

ANS; C there are three logical operators that are used to perform logical operations on boolean values (True or False). Here's a list of the logical operators :

and: This operator returns True if both operands are True, otherwise, it returns False.

or: This operator returns True if at least one of the operands is True, otherwise, it returns False.

not: This operator is a unary operator that returns the opposite of the operand's boolean value. If the operand is True, not returns False, and if the operand is False, not returns True.

Here's a quick example of how these logical operators work:

a = True

b = False

result\_and = a and b

result\_or = a or b

result\_not\_a = not a

result\_not\_b = not b

print("Logical AND:", result\_and)

print("Logical OR:", result\_or)

print("Logical NOT (a):", result\_not\_a)

print("Logical NOT (b):", result\_not\_b)

These logical operators are often used to combine conditions in conditional statements, loops, and more to control the flow of the program based on boolean values.

**Q.4. Explain right shift operator and left shift operator with examples.**

**ANS;** **the right shift (>>) and left shift (<<) operators are bitwise operators used to shift the bits of an integer to the right or left, respectively. These operators are often used in low-level programming and bit manipulation tasks.**

**Right Shift Operator (>>):**

**The right shift operator shifts the bits of a number to the right by a specified number of positions. The vacant positions on the left are filled with zeros. Each right shift by one position effectively divides the number by 2 (integer division).**

**Here's an example of the right shift operator:**

**x = 16**

**result = x >> 2 # Binary result: 00100 (Decimal: 4)**

**print("Original value:", x)**

**print("After right shift:", result)**

**Left Shift Operator (<<):**

**The left shift operator 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 left shift by one position effectively multiplies the number by 2.**

**Here's an example of the left shift operator:**

**x = 5**

**result = x << 2 # Binary result: 10100 (Decimal: 20)**

**print("Original value:", x)**

**print("After left shift:", result)**

**Q.5. Create a list containing int type data of length 15. Then write a code to check if 10 is**

**present in the list or not.**

ANS; my\_list = [5, 8, 2, 15, 10, 7, 12, 6, 3, 9, 18, 1, 4, 11, 14]

if 10 in my\_list:

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

else:

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

In this code, the in operator is used to check if the value 10 is present in the my\_list. If it is, the program prints that 10 is present; otherwise, it prints that 10 is not present.