21 May

Python Basic - 2

Q.1. Create two int type variables, apply addition, subtraction, division, and multiplication and store the results in variables. Then print the data in the following format by calling the variables:

Answer:

```
# Variables
x = 10
y = 5
# Operations
addition = x + y
subtraction = x - y
multiplication = x * y
division = x / y
# Print formatted results
print(f"First variable is {x} & second variable is {y}.")
print(f"Addition: {x} + {y} = {addition}")
print(f"Subtraction: {x} - {y} = {subtraction}")
print(f"Multiplication: {x} * {y} = {multiplication}")
print(f"Division: {x} / {y} = {division}")
```

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

(i) / & //

/ performs **floating-point division** and always returns a float. Example:

```
7 / 2  # Output: 3.5
```

// performs **floor division** and returns the largest integer less than or equal to the result. Example:

```
7 // 2  # Output: 3
```

(ii) ** & ^

****** is the **exponentiation operator**. It raises one number to the power of another. Example:

```
2 ** 3  # Output: 8 (2 raised to the power of 3)
```

^ is the **bitwise XOR operator**. It compares the bits of two numbers and returns 1 for each bit that differs. Example:

```
5 ^ 3  # Output: 6 (binary: 101 ^ 011 = 110)
```

Q.3. List the logical operators.

Answer:

and: Returns True if both conditions are true.

```
True and True  # Output: True
```

or: Returns True if at least one condition is true.

```
True or False  # Output: True
```

not: Inverts the Boolean value.

```
not True  # Output: False
```

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

Answer:

Right Shift Operator (>>): Shifts the bits of the number to the right by a specified number of positions. Each shift to the right effectively divides the number by 2.

Example:

```
8 >> 2  # Output: 2 (binary: 1000 >> 2 = 0010)
```

Left Shift Operator (<<): Shifts the bits of the number to the left by a specified number of positions. Each shift to the left effectively multiplies the number by 2.

Example:

```
3 << 2  # Output: 12 (binary: 0011 << 2 = 1100)
```

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.

Answer:

```
# Creating a list of length 15
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
# Check if 10 is present in the list
if 10 in my_list:
   print("10 is present in the list.")
else:
   print("10 is not present in the list.")
```