21 May

Python Basic - 2

ASSIGNMENT 7

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:_____/____= ____

```
# Create two integer variables
first_variable = 10
second_variable = 5

# Perform arithmetic operations
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 the results in the specified format
print(f"First variable is {first_variable}& second variable is {second_variable}")
print(f"Addition: {first_variable}+{second_variable} = {addition_result}")
print(f"Subtraction: {first_variable}-{second_variable}  = {subtraction_result}")
print(f"Multiplication: {first_variable}*{second_variable} = {multiplication_result}")
print(f"Division: {first_variable}/{second_variable}  = {division_result}")
```

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

(i) '/' & '//'

(ii) '' & '^'**

(i) Difference between '/' and '//':

/ (Forward Slash): This is the division operator. It performs regular division and returns a floating-point result if at least one of the operands is a float. For example, 5 / 2 would result in 2.5

//(Double Forward Slash): This is the floor division operator in Python. It performs division and returns the largest integer less than or equal to the result. It effectively rounds down the result to the nearest integer. For example, `5 // 2` would result in `2` because it discards the fractional part.

Difference between ** and ^:

**** (Double Asterisk):** This is the exponentiation operator. It is used to raise a number to a certain power. For example, 2 ** 3 results in 8 because it calculates 2 raised to the power of 3.

^ (Caret): The caret symbol (^) is used as a bitwise XOR operator for performing bitwise exclusive OR operations between binary numbers. For example, in Python, 5 ^ 3 would result in 6 because it performs a bitwise XOR operation on the binary representations of 5 and 3 (0101 ^ 0011 = 0110).

Q.3. List the logical operators.

1 and: The "and" operator returns True if both operands are True. Otherwise, it returns False.

2. or: The "or" operator returns True if at least one of the operands is True. If both operands are False, it returns False.

3. not: The "not" operator returns the opposite of the Boolean value of its operand. If the operand is True, "not" returns False, and if the operand is False, "not" returns True.

These operators are often used to combine and manipulate Boolean values to make decisions and control the flow of a program based on conditions.

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

In Python, the right shift (>>) and left shift (<<) operators are used to perform bitwise shift operations on integers. These operators allow you to shift the bits of an integer to the right or left, effectively multiplying or dividing the integer by powers of 2.

1. Left Shift (<<): The left shift operator (<<) shifts the bits of an integer to the left by a specified number of positions. Each left shift by one position effectively multiplies the integer by 2. Shifting by n positions multiplies the integer by 2^n.

```
x = 5                    # Binary representation: 0b101
result = x << 2          # Shift left by 2 positions
print(result)            # Output: 20 (Binary representation: 0b10100)
```

In this example, shifting 5 left by 2 positions results in 20, which is 5 multiplied by 2^2.

2. Right Shift (>>): The right shift operator (>>) shifts the bits of an integer to the right by a specified number of positions. Each right shift by one position effectively divides the integer by 2. Shifting by n positions divides the integer by 2^n.

```
x = 20                   # Binary representation: 0b10100
result = x >> 2          # Shift right by 2 positions
print(result)            # Output: 5 (Binary representation: 0b101)
```

In this example, shifting 20 right by 2 positions results in 5, which is 20 divided by 2^2.

Bitwise shift operations are commonly used in low-level programming, hardware manipulation, and certain algorithms that involve binary representations of data. They can be useful for optimizing code when working with specific bit-level patterns and calculations.

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.

```
                                                    # Create a list of integers
my_list = [5, 8, 2, 15, 10, 7, 3, 12, 18, 20, 1, 6, 9, 4, 11]

                                                    # 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.")
```