21 May

Python Basic - 2

* 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: / =

**Solution 1.**

# Create two int type 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("First variable is\t\t", first\_variable, " & second variable is\t", second\_variable)

print("Addition:\t\t+", "=\t", addition\_result)

print("Subtraction:\t\t-", "=\t", subtraction\_result)

print("Multiplication:\t\t\*", "=\t", multiplication\_result)

print("Division:\t\t/", "=\t", division\_result)

* 1. What is the difference between the following operators:

(i) ‘/’ & ‘//’

(ii) ‘\*\*’ & ‘^’

**Solution 2.**

(i) The '/' operator is used for division in Python, and it returns the result as a float. For example, 10 / 3 would result in 3.3333333333333335.

On the other hand, the '//' operator is used for floor division, also known as integer division. It returns the quotient as an integer, discarding any decimal part. For example, 10 // 3 would result in 3.

(ii) The '\*\*' operator is used for exponentiation in Python. It raises the base number to the power of the exponent. For example, 2 \*\* 3 would result in 8, as 2 raised to the power of 3 is 8.

The '^' operator, on the other hand, is not used for exponentiation in Python. Instead, it is the bitwise XOR operator. It performs a bitwise exclusive OR operation on the binary representations of the operands. It is used in bitwise operations, not for exponentiation.

* 1. List the logical operators.

Solution 3.

The logical operators in Python are:

* and: Returns True if both operands are True, otherwise returns False.
* or: Returns True if at least one of the operands is True, otherwise returns False.
* not: Returns the opposite boolean value of the operand. If the operand is True, not returns False, and if the operand is False, not returns True.

These logical operators are used to combine and manipulate boolean values or expressions in conditional statements and boolean operations.

* 1. Explain right shift operator and left shift operator with examples.

**Solution 4.**

In Python, the right shift (>>) and left shift (<<) operators are used for bitwise shifting of the binary representations of integers. They manipulate the bits of an integer value by shifting them to the right or left.

1. The right shift operator (>>) shifts the bits of the binary representation of an integer to the right by a specified number of positions. It effectively divides the integer by 2 raised to the power of the specified number of positions. The rightmost bit is discarded, and the leftmost bit is filled with the sign bit (0 for positive numbers, 1 for negative numbers).

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

x = 10 # Binary representation: 1010

# Right shifting the bits by 2 positions

result = x >> 2 # Binary representation: 0010 (Decimal: 2)

print(result) # Output: 2

2. The left shift operator (<<) shifts the bits of the binary representation of an integer to the left by a specified number of positions. It effectively multiplies the integer by 2 raised to the power of the specified number of positions. The rightmost positions are filled with zeroes.

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

x = 5 # Binary representation: 0101

# Left shifting the bits by 2 positions

result = x << 2 # Binary representation: 010100 (Decimal: 20)

print(result) # Output: 20

* 1. 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.

**Solution 5.**

# Create a list of integers

my\_list = [2, 5, 8, 10, 13, 17, 20, 23, 26, 29, 32, 35, 38, 41, 44]

# 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.")

Output:

10 is present in the list.