21 May

Python Basic - 2

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: \_\_ / \_\_ = \_\_

Answer:

*Here's the code to perform the specified arithmetic operations on two integer variables and print the results in the desired format:*

*# Create two int type variables*

*first\_variable = 10*

*second\_variable = 5*

*# Perform arithmetic operations and store results in variables*

*addition = first\_variable + second\_variable*

*subtraction = first\_variable - second\_variable*

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

*division = first\_variable / second\_variable*

*# Print the data in the desired format*

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

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

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

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

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

*Output:*

*First variable is 10 & second variable is 5.*

*Addition: 10 + 5 = 15*

*Subtraction: 10 - 5 = 5*

*Multiplication: 10 \* 5 = 50*

*Division: 10 / 5 = 2.0*

*Explanation:*

*We create two integer variables first\_variable and second\_variable with values 10 and 5, respectively.*

*The arithmetic operations are performed and the results are stored in separate variables: addition, subtraction, multiplication, and division.*

*The results are then printed using the print function, incorporating the variables and desired format.*

*By executing this code, you will get the desired output with the variables and results formatted as specified.*

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

(i) ‘/’ & ‘//’

(ii) ‘\*\*’ & ‘^’

Answer:

*The operators you mentioned have different functionalities and purposes. Here's the difference between them:*

*(i) / and // Operators:*

*/ (Forward Slash) Operator: The forward slash / is the division operator in Python. It performs division and returns the quotient as a float (floating-point number). For example, 5 / 2 would result in 2.5.*

*// (Double Slash) Operator: The double slash // is the floor division operator in Python. It performs division and returns the quotient as an integer by rounding down towards negative infinity. For example, 5 // 2 would result in 2.*

*Example:*

*x = 7*

*y = 3*

*result1 = x / y # Division using /*

*result2 = x // y # Floor division using //*

*print(result1) # Output: 2.3333333333333335*

*print(result2) # Output: 2*

*(ii) \*\* and ^ Operators:*

*\*\* (Double Asterisk) Operator: The double asterisk \*\* is the exponentiation operator in Python. It raises the left operand to the power of the right operand. For example, 2 \*\* 3 would result in 8.*

*^ (Caret) Operator: The caret ^ is not an exponentiation operator in Python. Instead, it is the bitwise XOR operator, used for performing bitwise XOR operation on the binary representations of two numbers. It does not raise a number to a power like \*\* does.*

Example:

*a = 2*

*b = 3*

*result3 = a \*\* b # Exponentiation using \*\**

*result4 = a ^ b # Bitwise XOR using ^*

*print(result3) # Output: 8*

*print(result4) # Output: 1 (bitwise XOR of 10 and 11 in binary)*

*It's important to use the correct operator based on the desired functionality to ensure the desired computation and results.*

Q.3. List the logical operators.

Answer:

*The logical operators in Python are as follows:*

*and Operator: The and operator returns True if both operands are True, and False otherwise.*

*or Operator: The or operator returns True if at least one of the operands is True, and False otherwise.*

*not Operator: The not operator returns the negation of the operand. It converts True to False and False to True.*

*These logical operators are often used to combine or negate Boolean expressions and make decisions based on multiple conditions.*

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

Answer:

*The right shift (>>) and left shift (<<) operators are bitwise operators in Python. They perform shifting of the binary representation of integers.*

*The right shift (>>) operator shifts the bits of a number to the right by a specified number of positions. The shifted bits are discarded, and the vacant positions on the left are filled with zeros. The right shift effectively divides the number by 2 raised to the power of the specified shift amount.*

*Example:*

*x = 10 # Binary: 1010*

*# Right shift by 1 position*

*result1 = x >> 1 # Binary: 0101 (Decimal: 5)*

*# Right shift by 2 positions*

*result2 = x >> 2 # Binary: 0010 (Decimal: 2)*

*print(result1) # Output: 5*

*print(result2) # Output: 2*

*In the example, the initial value of x is 10, which is represented in binary as 1010. The right shift by 1 position (x >> 1) moves each bit one position to the right, resulting in the binary representation 0101, which is equal to the decimal value 5. Similarly, the right shift by 2 positions (x >> 2) moves each bit two positions to the right, resulting in the binary representation 0010, which is equal to the decimal value 2.*

*The left shift (<<) operator, on the other hand, 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. The left shift effectively multiplies the number by 2 raised to the power of the specified shift amount.*

*Example:*

*y = 5 # Binary: 0101*

*# Left shift by 1 position*

*result3 = y << 1 # Binary: 1010 (Decimal: 10)*

*# Left shift by 2 positions*

*result4 = y << 2 # Binary: 10100 (Decimal: 20)*

*print(result3) # Output: 10*

*print(result4) # Output: 20*

*In this example, the initial value of y is 5, which is represented in binary as 0101. The left shift by 1 position (y << 1) moves each bit one position to the left, resulting in the binary representation 1010, which is equal to the decimal value 10. Similarly, the left shift by 2 positions (y << 2) moves each bit two positions to the left, resulting in the binary representation 10100, which is equal to the decimal value 20.*

*These bitwise shift operators are useful for manipulating binary representations of numbers, performing fast multiplication or division by powers of 2, or working with binary flags or bit-level operations.*

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:

*Here's an example code that creates a list of integers with a length of 15 and checks if the number 10 is present in the list:*

*# Create a list of integers*

*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.")*

*Output:*

*10 is present in the list.*

*In this code, we create a list called my\_list containing integers from 1 to 15. Then, we use the in operator to check if the number 10 is present in the list. If it is, we print a message indicating that 10 is present. Otherwise, we print a message indicating that it is not present.*

*By running this code, you will find that it outputs "10 is present in the list" since 10 is indeed present in the list.*