**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 \_\_.**

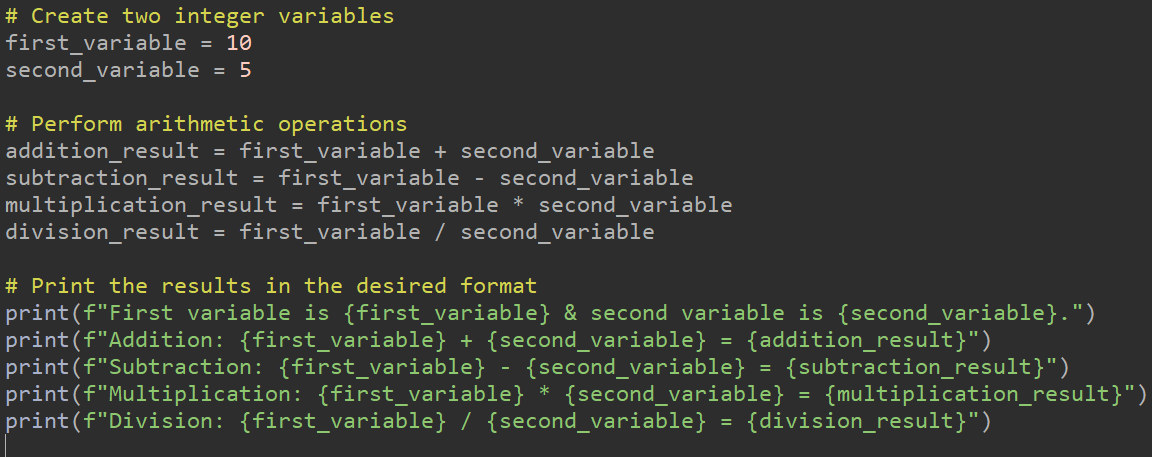
**Addition: \_\_ + \_\_ = \_\_**

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

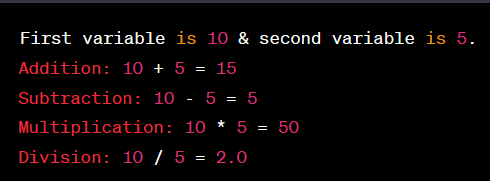
**Multiplication: \_\_ \* \_\_ = \_\_**

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

**Ans:**

Code: 

Output:



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

1. **‘/’ & ‘//’**
2. **‘\*\*’ & ‘^**’

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

* *'/' (Forward Slash): This operator is used for division in Python, and it returns a floating-point (decimal) result, even if the operands are integers. For example, `5 / 2` will result in `2.5`.*
* *'//' (Double Forward Slash): This operator is used for integer division or floor division in Python. It returns the quotient as an integer, effectively truncating the decimal part. For example, `5 // 2` will result in `2`.*

***(ii) Difference between '\*\*' and '^':***

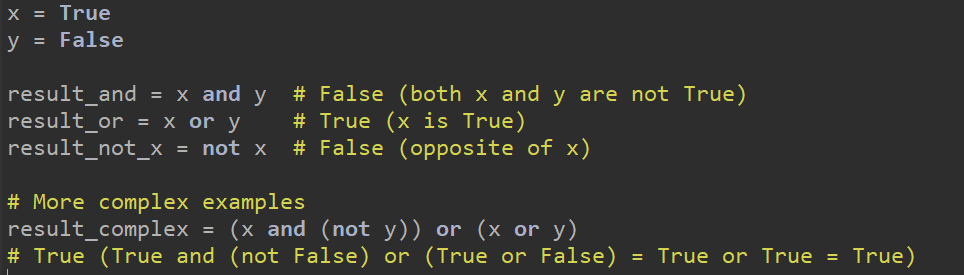
* *'\*\*' (Double Asterisk): This operator is used for exponentiation or raising a number to a power. For example, `2 \*\* 3` calculates 2 to the power of 3, resulting in `8`.*
* *'^' (Caret): In Python, the caret (^) is not an exponentiation operator but rather a bitwise XOR operator. It performs a bitwise exclusive OR operation on the binary representations of two numbers. For example, `5 ^ 3` performs a bitwise XOR operation between 5 and 3, which results in 6 (binary `0101 ^ 0011 = 0110`, which is 6 in decimal).*
  1. List the logical operators.

Ans: *In Python, logical operators are used to perform logical operations on Boolean values (True or False). The common logical operators in Python include:*

*1. `and`: The logical AND operator returns `True` if both operands are `True`, and `False` otherwise.*

*2. `or`: The logical OR operator returns `True` if at least one of the operands is `True`, and `False` if both are `False`.*

*3. `not`: The logical NOT operator is a unary operator that returns the opposite of the operand. If the operand is `True`, `not` returns `False`, and if the operand is `False`, `not` returns `True`.*



These logical operators are commonly used in conditional statements, boolean expressions, and logic-based decision-making in Python code.

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

Ans: *In Python, the right shift (`>>`) and left shift (`<<`) operators are used for bitwise shifting of binary representations of integers. These operators shift the bits of an integer value to the right or left, respectively. Here's an explanation of each operator with examples:*

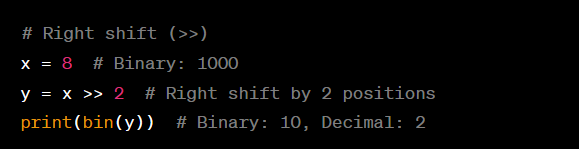
**Right Shift Operator (`>>`):**

*The right shift operator (`>>`) shifts the bits of a binary number to the right by a specified number of positions. It effectively divides the number by 2 raised to the power of the specified shift value and discards any fractional part.*

**Syntax:**



* *`x` is the integer value you want to shift.*
* *`y` is the number of positions you want to shift to the right.*
* **Example:**



*In this example, shifting the binary value of `x` by 2 positions to the right effectively divides it by 2^2 (4), resulting in a new value of 2.*

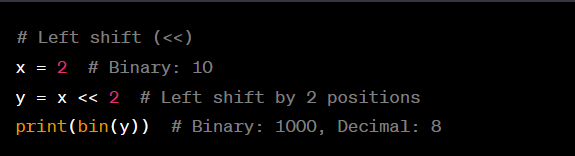
**Left Shift Operator (`<<`):**

*The left shift operator (`<<`) shifts the bits of a binary number to the left by a specified number of positions. It effectively multiplies the number by 2 raised to the power of the specified shift value.*

**Syntax:**



* + *`x` is the integer value you want to shift.*
  + *`y` is the number of positions you want to shift to the left.*
* Example:



*In this example, shifting the binary value of `x` by 2 positions to the left effectively multiplies it by 2^2 (4), resulting in a new value of 8.*

*The left shift and right shift operators are commonly used in low-level programming and bitwise operations, particularly for optimizing performance or working with binary representations of data.*

* 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**.

**Ans:** 