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- x=int(input("enter the first number"))

y=int(input("enter the second number"))

print(f"First variable is {x} & second variable is {y}.")

result=x+y

print(f"Addition: {x} + {y} = {result}")

result=x-y

print(f"Subtraction: {x} - {y} = {result}")

result=x\*y

print(f"Multiplication: {x} \* {y} = {result}")

result=x/y

print(f"Division: {x} / {y} ={result}")

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

(i) ‘/’ & ‘//’

(ii) ‘\*\*’ & ‘^’

Ans-

* The '/' operator performs division and returns the quotient as a floating-point number. It provides the result with decimal places, even if the division is exact.
* The '//' operator, known as the floor division operator, performs division and returns the quotient as an integer. It discards the decimal part of the result and provides the integer quotient.

a = 10

b = 3

result1 = a / b

result2 = a // b

print(result1) # Output: 3.3333333333333335

print(result2) # Output: 3

ii)

* The '\*\*' operator is the exponentiation operator and is used for raising a number to a power. For example, a \*\* b calculates a raised to the power of b.
* The '^' operator, on the other hand the '^' operator is used for bitwise XOR.

print(a\*\*b) # 1000

print(bin(a))

print(bin(b))

result=a^b

print(result)

print(bin(result))##9

Q.3. List the logical operators.

* and: The and operator returns True if both operands are True, and False otherwise. It evaluates the second operand only if the first operand is True.
* or: The or operator returns True if at least one of the operands is True, and False if both operands are False. It evaluates the second operand only if the first operand is False.
* not: The not operator is a unary operator that returns the opposite of the operand. It returns True if the operand is False, and False if the operand is True.

a = True

b = False

print(a and b) # Logical AND, Output: False

print(a or b) # Logical OR, Output: True

print(not a) # Logical NOT, Output: False

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

Ans- Right Shift (>>):

The right shift operator (>>) shifts the bits of a number to the right by a specified number of positions. It effectively divides the number by 2 raised to the power of the shift amount.

Left Shift(<<):

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

num = 16

shift\_amount = 2

result = num >> shift\_amount

print(result) # Output: 4

num = 3

shift\_amount = 2

result = num << shift\_amount

print(result) # Output: 12

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.

Ans- my\_list = [5, 3, 8, 1, 10, 6, 9, 2, 7, 4, 11, 15, 13, 12, 14]

if 10 in my\_list:

print("10 is present in the list.")

else:

print("10 is not present in the list.")