07 21 May basic 2

August 16, 2023

1 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:** / ___ = ___

[7]:
```
v1 = int (9)
v2= int (4)
print ("First variable is {} & second variable is {}".format(v1,v2))
print ("Addition: {} + {} = {} ".format(v1,v2,v1+v2))
print ("Subtraction: {} - {} = {}".format(v1,v2,v1-v2))
print ("Multiplication: {} * {} = {}".format(v1,v2,v1*v2))
print ("Division: {} / {} = {}".format(v1,v2,v1/v2))
```

```
First variable is 9 & second variable is 4
Addition: 9 + 4 = 13
Subtraction: 9 - 4 = 5
Multiplication: 9 * 4 = 36
Division: 9 / 4 = 2.25
```

Q.2. What is the difference between the following operators: (i) '/' & '//' (ii) '**' & '^'

(i) '/' & '//'

[ ]:
```

```

/ (Division): This operator returns the quotient of the division in float form, even if the division is exact.

// (Floor Division): This operator returns the quotient of the division in integer form, discarding any fractional parts.

[6]:
```
print ("9/2 =", 9/2  ," and 9//2 =",9//2)
```

```
9/2 = 4.5  and 9//2 = 4
```

(ii) '' & '^'**

[ ]:
```

```

** (Exponentiation): This operator is used to raise a number to the power of another number.

^ (Bitwise XOR): This operator is used for bitwise exclusive or (XOR) operation. It is not used for exponentiation in Python. For each bit position in the result, the XOR operation returns 1 if exactly one of the corresponding bits of the operands is 1, and 0 otherwis

[7]:
```
print ("3**2=",3**2, " and 3^2 =", 3^2)
```

```
3**2= 9  and 3^2 = 1
```

Q.3. List the logical operators. and or not

Q.4. Explain right shift operator and left shift operator with examples. Right Shift Operator (»): The right shift operator shifts the bits of a binary representation of an integer to the right by a specified number of positions. The rightmost bits are discarded, and the leftmost positions are filled with the sign bit (the most significant bit) to preserve the sign of the number.

[8]:
```
x = 12  # Binary representation: 1100
y = x >> 1  # Shifting right by 1 position
print(y)  # Output: 6 (Binary representation: 0110)
```

```
6
```

Left Shift Operator («): The left shift operator shifts the bits of a binary representation of an integer to the left by a specified number of positions. Zeroes are filled in from the rightmost positions.

[9]:
```
x = 5  # Binary representation: 0101
y = x << 2  # Shifting left by 2 positions
print(y)  # Output: 20 (Binary representation: 10100)
```

```
20
```

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.

[9]:
```
# Create a list containing 15 integer elements
my_list = [2, 5, 8, 10, 15, 20, 10, 25, 30, 35, 40, 45, 50, 55, 60]

# Check if 10 is present in the list
if 10 in my_list:
    print("Number 10 is present in the list.")
else:
    print("Number 10 is not present in the list.")
```

```
Number 10 is present in the list.
```

[ ]:
```

```