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

```
a=10
b=5
add=a+b
sub=a-b
mul=a*b
div=a/b
print("First variable is "+ str(int(a)) + " & second variable is " + str(int(b)))
print("Addition: "+ str(int(a)) +" + " + str(int(b))+ " = " + str(int(add)))
print("Subtraction: "+ str(int(a)) +" - " + str(int(b))+ " = " + str(int(sub)))
print("Multiplication: "+ str(int(a)) +" * " + str(int(b))+ " = " + str(int(mul)))
print("Division: "+ str(int(a)) +" / " + str(int(b))+ " = " + str(int(div)))
```

2. What is the difference between the following operators:

(i) ‘/’ & ‘//’
(ii) ‘**’ & ‘^’

Answer

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

The '/' operator in Python is used for division and returns a float value. It performs normal division and provides the exact quotient, including decimals.

The '//' operator is used for floor division, also known as integer division. It returns the largest possible integer quotient without considering the decimal part. The result is always rounded down to the nearest integer.

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

The '**' operator in Python is used for exponentiation or raising a number to a power. It calculates the value of the base raised to the power of the exponent.

The '^' operator, on the other hand, is not an exponentiation operator in Python. Instead, it is the bitwise XOR operator used for performing bitwise XOR operation on two integers.

3. List the logical operators.

Answers

and: The and operator returns True if both operands are True, otherwise it returns False. It performs a logical AND operation.

or: The or operator returns True if at least one of the operands is True, otherwise it returns False. It performs a logical OR operation.

not: The not operator returns the opposite boolean value of the operand. If the operand is True, it returns False, and if the operand is False, it returns True. It performs a logical NOT operation**.**

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

Answer

Right Shift (>>):

The right shift operator (>>) shifts the bits of a number to the right by a specified number of positions.

The rightmost bits are discarded, and the leftmost bits are filled with the sign bit (for signed integers) or with 0 (for unsigned integers).

Each shift to the right effectively divides the number by 2 (integer division).

Example:

```
x = 8  # Binary: 1000
# Right shifting by 2 positions
result = x >> 2  # Binary: 0010, Decimal: 2
print(result)
```

Left Shift (<<):

The left shift operator (<<) shifts the bits of a number to the left by a specified number of positions.

The rightmost bits are filled with 0, and the leftmost bits are discarded.

Each shift to the left effectively multiplies the number by 2.

Example:

```
x = 2  # Binary: 0010
# Left shifting by 2 positions
result = x << 2  # Binary: 1000, Decimal: 8
print(result)
```

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

```
lists=[2,4,6,8,10,12,14,16,18,20,22,24,26,28,30]
print(lists)
print("length of lists is "+str(len(lists)))
if 10 in lists:
    print("10 is present")
else:
    print("10 is not present")
```