Python Basic – 2(Assignment-7)

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 \_\_.

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

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

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

Solution:

First\_num=int(input('Enter 1st number:'))

Second\_num=int(input('Enter 2nd number:'))

Add=First\_num+Second\_num

sub=First\_num-Second\_num

multi=First\_num\*Second\_num

div=First\_num/Second\_num

print('Addition:',First\_num,'+',Second\_num,'=',Add)

print('subtraction:',First\_num,'-',Second\_num,'=',sub)

print('Multiplication:',First\_num,'\*',Second\_num,'=',multi)

print('Division:',First\_num,'/',Second\_num,'=',div)

output:

Enter 1st number:20

Enter 2nd number:5

Addition: 20 + 5 = 25

subtraction: 20 - 5 = 15

Multiplication: 20 \* 5 = 100

Division: 20 / 5 = 4.0

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

Solution:

1. ‘/’ & ‘//’

The first one is **Float Division("/")** and the second is **Integer Division("//")** or Floor Division.

**Float Division("/"):** Divides left hand operand by right hand operand.  
5/2 = 2.5

Division works in Python the way it's mathematically defined.

x/y==float(x/y)

**Floor Division("//"):** The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative**,** the result is floored , i.e., rounded away from zero (towards negative infinity).

5//2=2

-11//3 = -4

Floor division works in Python the way it's mathematically defined.

x // y == math.floor(x/y)

(ii)‘\*\*’ & ‘^’

In python, \*\* is the exponentiation operator which is used for to get power value , and ^ is the XOR operator .

For exponent operator(example); 2\*\*3=8

XOR is the bitwise XOR of the two numbers. In other words, write each number in binary, then take XOR of the two numbers one place at a time (A XOR B is true if A and B are different, and false if they're the same. 0 is false, 1 is true).

Here's an example computing 5 XOR 3:

5 ^ 3

101 = 5

011 = 3

from left to right:

1 ^ 0 -> 1

0 ^ 1 -> 1

1 ^ 1 -> 0

0 ^ 0 -> 0

101

^ 011

-------

110

so 5^3=6

Q.3. List the logical operators.

Solution:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Example result** |
| && | AND. True only if both operands are true. | 0 (only one is true) |
| || | OR. True if either operand is true. | 1 (the first test is true) |
| ∼ | NOT. Changes true to false and false to true. | 1 (the strings are not equal) |

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

Solution:

[(i)Bitwise Left Shift Operator](https://www.digitalocean.com/community/tutorials/python-bitwise-operators#5-bitwise-left-shift-operator)

Python bitwise left shift operator shifts the left operand bits towards the left side for the given number of times in the right operand. In simple terms, the binary number is appended with 0s at the end.

>>> 10 << 2

40

>>>

A=10=>1010(Binary)

A<<2=1010<<2

=101000

=40(Decimal)

(ii)[Bitwise Right Shift Operator](https://www.digitalocean.com/community/tutorials/python-bitwise-operators#6-bitwise-right-shift-operator):

Python right shift operator is exactly the opposite of the left shift operator. Then left side operand bits are moved towards the right side for the given number of times. In simple terms, the right side bits are removed.

>>> 10 >> 2

2

>>>

A=10=>1010(Binary)

A>>2=1010>>2

=10

=2(Decimal)

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.

Solution:

List=[2,4,6,8,10,12,14,16,18,20,22,24,26,28,30]

print('The lenght of the list:',len(List))

if 10 in List:

print('The value 10 is present in the list')

else:

print('The value 10 is not present in the list')

output:

The lenght of the list: 15

The value 10 is present in the list