assignment 7

June 17, 2023

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: ___ / ___ = ___

[29]:

```
a = 13
b = 20

addition: a + b
subtraction: a - b
multiplication: a * b
division: a / b

print(f"first variable is {a}and second variable is {b}.")
print(f"Addition:{a} + {b} = {addition}")
print(f"Subtraction:{a} - {b} = {subtraction}")
print(f"Multiplication:{a} * {b} = {multiplication}")
print(f"Division:{a} / {b} = {division}")
```

```
first variable is 13and second variable is 20.
Addition:13 + 20 = 15
Subtraction:13 - 20 = 5
Multiplication:13 * 20 = 50
Division:13 / 20 = 2.0
```

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

(i) ‘/’ & ‘//’

(ii) ’**’ & ‘^’

ANS:The difference between the / and // operators in Python is that / performs floating-point division and returns a float value, while // performs integer division and returns an integer value .

For example, if we divide 11 by 3 using /, we get 3.6666666666666665 as the result. However, if we use //, we get 3 as the result.

On the other hand, the ** operator is used for exponentiation in Python. It raises the left operand to the power of the right operand 3. For example, 2 ** 3 returns 8.

The ^ operator is used for bitwise XOR operation in Python. It returns a binary number which is the result of XOR operation on two given numbers . For example, 10 ^ 12 returns 6.

Q.3. List the logical operators.

ANS: The logical operators in pyhton are: And, or, not.

logical and operator: Logical and operator returns True if both the operands are True else it returns False.

logical or operator: Logical or operator returns True if either of the operands is True.

logical not operator: Logical not operator work with the single boolean value. If the boolean value is True it returns False and vice-versa.

True and True is True.

True and False is False.

False and True is False.

False and False is False.

True or True is True.

True or False is True.

False or True is True.

False or False is False.

not True is False.

not False is True.

[17]:

```
a = 10
b = 10
c = -10

if a > 0 and b > 0:
    print("The numbers are greater than 0")
if a > 0 and b > 0 and c > 0:
    print("The numbers are not greater than 0")
else:
    print("Atleast one number is not greater than 0")
```

```
The numbers are greater than 0
Atleast one number is not greater than 0
```

[22]:

```
a = 10
b = 12
c = 0

if a or b or c:
    print("Atleast one number has boolean value as True")
else:
    print("All the numbers have boolean value as False")
```

```
Atleast one number has boolean value as True
```

[32]:

```
x = 50

if not (20 <= x < 40):
    print(f"{x} is outside")
```

```
50 is outside
```

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

ANS:The right shift operator works by shifting the binary representation of the first operand to the right by the number of positions specified by the second operand. Each position that the bits are shifted right is equivalent to dividing the number by 2. The right shift operator is primarily used for performing bitwise operations in Python, such as extracting specific bits from a binary number or efficiently packing multiple values into a single binary number. It is a useful tool for manipulating binary numbers in a variety of programming applications.

The bitwise left shift operator shifts the bits of the binary representation of the input number to the left side by a specified number of places. The empty bits created by shifting the bits are filled by 0s. The syntax for the bitwise left shift is a « n, where 'a' is the number whose bits will be shifted by 'n' places to the left.

[8]:

```
# Example of right sift opeartor

a = 10 # 1010
result = a >> 1
print(result)

# 10 =  1 0 1 0
# 5  =  0 1 0 1
```

```
5
```

[9]:

```
# left shift
a = 10
result = a << 3
#result = a << 2
print(result)
```

```
# 5  =  0 1 0 1
# 20 =  1 0 1 0 0

bin(20)
```

```
80
```

[9]:

```
'0b10100'
```

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.

[34]:

```
# Create a list containing 15 integers
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

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

```
10 is present in the list
```

[ ]: