Operators

July 1, 2022

1 Operators

1.1 Arithmetic Operators

[1]:

```
a = 9
b = 4
```

[2]:

```
#unary
print(+a)
print(-b)
```

```
9
-4
```

[3]:

```
# Addition of numbers
add = a + b
print(add)
```

```
13
```

[4]:

```
# Subtraction of numbers
sub = a - b
print(sub)
```

```
5
```

[5]:

```
# Multiplication of number
mul = a * b
print(mul)
```

```
36
```

[6]:

```
# Division(float) of number
div1 = a / b
print(div1)
```

```
2.25
```

[7]:

```
# Division(floor) of number
div2 = a // b
```

```
print(div2)
```

```
2
```

[8]:

```
# Modulo of both number
mod = a % b
print(mod)
```

```
1
```

[9]:

```
# Power
p = a ** b
print(p)
```

```
6561
```

1.2 Relational Operators

[10]:

```
a = 13
b = 33
```

[11]:

```
# a > b is False
print(a > b)
```

```
False
```

[12]:

```
# a < b is True
print(a < b)
```

```
True
```

[13]:

```
# a == b is False
print(a == b)
```

```
False
```

[14]:

```
# a != b is True
print(a != b)
```

```
True
```

[15]:

```
# a >= b is False
print(a >= b)
```

```
False
```

[16]:

```
# a <= b is True
print(a <= b)
```

```
True
```

[17]:

```
#Equality Comparison on Floating-Point Values
x = 1.1 + 2.2
print(x == 3.3)
```

```
False
```

[18]:

```
tolerance = 0.00001
x = 1.1 + 2.2
print(abs(x - 3.3) < tolerance)
```

```
True
```

1.3 Logical Operators

[19]:

```
a = True
b = False
```

[20]:

```
# Print a and b is False
print(a and b)
```

```
False
```

[21]:

```
# Print a or b is True
print(a or b)
```

```
True
```

[22]:

```
# Print not a is False
print(not a)
```

```
False
```

[23]:

```
# Print not b is True
print(not b)
```

```
True
```

1.4 Assignment Operators

[24]:

```
#Assign =
a = 2 ; b = 5
print(a,b)
```

```
2 5
```

[25]:

```
#Add and Assign
a = a+2
print(a)

a += 2
print(a)
```

```
4
6
```

[26]:

```
#Subtract and Assign
a -= 1    # a = a-1
print(a)
```

```
5
```

[27]:

```
#Multiply and Assign
a *= b    # a = a*b
print(a)
```

```
25
```

[28]:

```
#Divide And Assign
a /= 2    # a = a/2
print(a)
```

```
12.5
```

[29]:

```
#Floor Divide And Assign
a //= 2    # a = a//2
print(a)
```

```
6.0
```

[30]:

```
#Modulus And Assign
b %= 3    # b = b%3
print(b)
```

```
2
```

1.5 Bitwise Operators

[31]:

```
a = 10
b = 4
```

[32]:

```
print(bin(a))
print(bin(b))
```

```
0b1010
0b100
```

[33]:

```
# Print bitwise AND operation
print(a & b)
```

```
0
```

[34]:

```
# Print bitwise OR operation
print(a | b)
```

```
14
```

[35]:

```
# Print bitwise NOT operation
print(~a)
```

```
-11
```

[36]:

```
# print bitwise XOR operation
print(a ^ b)
```

```
14
```

[37]:

```
# print bitwise right shift operation
print(a >> 2)
```

```
2
```

[38]:

```
# print bitwise left shift operation
print(a << 2)
```

```
40
```

[39]:

```
binaryNumber = int(bin(100).replace("0b", ""))
binaryNumber
```

[39]:

```
1100100
```

1.6 Identity Operators

Compares Adresses

[40]:

```
x1 = 5
y1 = 5
```

[41]:

```
print(id(x1))
print(id(y1))
```

```
140715398866848
140715398866848
```

[42]:

```
# is not
print(x1 is not y1)
```

```
False
```

[43]:

```
x2 = 'Hello'
y2 = 'Hello'
```

[44]:

```
# is
print(x2 is y2)
```

```
True
```

[45]:

```python
x3 = [1,2,3]
y3 = [1,2,3]
```

[46]:

```python
print(id(x3))
print(id(y3))
```

```
2278850429312
2278850431744
```

[47]:

```python
print(x3 is y3)
```

```
False
```

Here, we see that x1 and y1 are integers of the same values, so they are equal as well as identical. Same is the case with x2 and y2 (strings).

But x3 and y3 are lists. They are equal but not identical. It is because the interpreter locates them separately in memory although they are equal.

1.7 Membership Operators

[48]:

```python
x = 'Hello world'
```

[49]:

```python
# in
print('H' in x)
```

```
True
```

[50]:

```python
# not in
print('hello' not in x)
```

```
True
```

Here, ‘H’ is in x but ‘hello’ is not present in x (remember, Python is case sensitive).

[51]:

```python
y = {1:'a',2:'b'}
```

[52]:

```python
print(1 in y)
```

```
True
```

[53]:

```python
print('a' in y)
```

```
False
```

1 is key and ‘a’ is the value in dictionary y. Hence, ‘a’ in y returns False.

1.8 Operators Precedence

[54]:

```python
v = 4 ; w = 5 ; x = 8 ; y = 2
```

[55]:
```
z = (v+w) * x / y
print("Value of (v+w) * x/ y is ",  z)
```

```
Value of (v+w) * x/ y is  36.0
```

1.9 Operator OverLoading

Operator Overloading means giving extended meaning beyond their predefined operational meaning.

For example operator + is used to add two integers as well as join two strings and merge two lists.

It is achievable because '+' operator is overloaded by int class and str class.

same built-in operator or function shows different behavior for objects of different classes.

This is called Operator Overloading.

[56]:
```
# Python program to show use of + , * operators for different purposes.

print(1 + 2)
print()

# concatenate two strings
print("Python"+"Training")
print()

# Merge two lists
print([1,2,3,4,"Hello"]+[2,3,4,5,"Student"])
print()

# Product two numbers
print(3 * 4)
print()
```

```python
# Repeat the String
print("Python"*4)
print()
```

```
3

PythonTraining

[1, 2, 3, 4, 'Hello', 2, 3, 4, 5, 'Student']

12

PythonPythonPythonPython
```

But , Python didn't know how to add two Point objects together.

However, we can achieve this task in Python through operator overloading.

1.10 To Do :

Calculate the multiplication and sum of two numbers by taking input from user

Write a Python program which accepts the radius of a circle from the user and compute the area

Write a Python program to calculate surface volume and area of a cylinder

Write a Python program to calculate surface volume and area of a sphere

Write a Python program to convert radian to degree and degree to radian

Write a Python program to convert celsius to fahrenheit

Python Program to Calculate the Area of a Triangle by taking 3 sides values from user

Write a Python program which accepts a sequence of comma-separated numbers from user and generate a list and a tuple with those numbers.[Hint : Spilt]

Write a Python program to concatenate all elements in a list into a string and return it

Write a Python program that accepts an integer (n) and computes the value of n+nn+nnn [Ex : 5+55+555]