Operators

Arithmatic operators

In [46]:
```
x = 15
y = 4
```

In [ ]:
```
# Addition

ad = x+y
print("Addition of two numbers is: ",ad)
```

In [ ]:
```
# Subtraction

sub = x-y
print("Subtraction of two numbers is: ",sub)
```

In [ ]:
```
# Multiplication

mul = x*y

print("Multiplication of two numbers is: ",mul)
```

In [47]:
```
# Division ( / )

div = x/y
print("Division of two numbers is: ",div)
```

```
Division of two numbers is:  3.75
```

In [48]:
```
# Floor division

floor_div = x//y
print("Floor Division of two numbers is: ",floor_div)
```

```
Floor Division of two numbers is:  3
```

In [49]:
```
# Modulous division (Remainder)

mod_div = x%y
print("Mod Division of two numbers is: ",mod_div)
```

```
Mod Division of two numbers is:  3
```

In [ ]:
```
# power
a = 3
b = 2

sq = a**b
print(" value of 3^2 is :",sq)
```

In [ ]:
```

```

Identity Operators

In [1]:
```
x1 = 5
y1 = 5
x2 = "hello world"
y2 = "hello india"
```

In [2]:
```
# is : returs true if both the operands are similar

x1 is y1
```

Out[2]: True

In [4]:
```
# is not : returs true if both the operands are not similar
x2 is not y2
```

Out[4]: True

Membership Operator

In [5]:
```
x = "i am a coder"
```

In [6]:
```
# in

'H' in x
```

Out[6]: False

In [8]:
```
'i' in x
```

Out[8]: True

In [10]:
```
d = {1:'a',2:'b'}
```

In [11]:
```
1 in d
```

Out[11]: True

In [16]:
```
# not in

l = [1,2,3]
```

In [17]:
```
3 not in l
```

Out[17]: False

In [18]:
```
6 not in l
```

Out[18]: True

In [ ]:
```

```

Relational Operator

In [30]:
```
# equal to
1 == 2
```

Out[30]: False

In [28]:
```
2 ==2
```

Out[28]: True

In [31]:
```
# Not equal to

5!=6
```

Out[31]: True

In [32]:
```
5!=5
```

Out[32]: False

In [34]:
```
# less than
6<7
```

Out[34]: True

In [35]:
```
# greater than

18>400
```

Out[35]: False

In [36]:
```
# less than equal to : check for two conditions (less, equal)
# if any one is true, then my final answer is true

8<=10
```

Out[36]: True

In [40]:
```
# Greater than equal to : check for two conditions (greater, eq
# if any one is true, then my final answer is true


800>=500
```

Out[40]: True

In [38]:
```
# T T = T
# T F = T
# F T = T
# F F = F
```

In [43]:
```
7>=16
```

Out[43]: False

In [ ]:
```

```

Assignment Operators

In [52]:
```
# assign add   syntax : +=
a = 4
a+=2 # internally it performs a = a+2
```

In [53]:
```
a
```

Out[53]: 6

In [54]:
```
# assign sub   syntax : -=
b = 20
b-=5 # internally it performs b = b-5
print(b)
```

15

In [55]:
```
# assign Mul   syntax : *=
c = 8
c*=4 # internally it performs c = c*4
print(c)
```

32

In [56]:
```
# assign float division syntax : /=
d = 15
d/=4 # internally it performs d = d/4
print(d)
```

```
3.75
```

In [57]:
```
# assign floor division syntax : //=
e = 15
e//=6 # internally it performs e = e//6
print(e)
```

```
2
```

In [59]:
```
# assign Modulous division syntax : %=
f = 15
f%=6 # internally it performs f = f%6
print(f)
```

```
3
```

logical Operator

In [60]:
```
m = True
n = False
```

In [ ]:
```
# AND operator  syntax: and
```

In [ ]:
```
# T T = T
# T F = F
# F T = F
# F F = F
```

In [65]:
```
print(m and n)
```

```
False
```

In [66]:
```
# OR Operator  syntax: or
```

In [ ]:
```
# T T = T
# T F = T
# F T = T
# F F = F
```

In [71]:
```
print(m or n)
```

```
True
```

In [ ]:
```
# NOT operator  syntax: not
```

In [76]:
```
print(not m)
```

```
False
```

Bitwise operators

In [79]:
```
# Bitwise AND  syntax: &
```

In [77]:
```
a = 4  # binary  = 0100
b = 10 # binary = 1010
```

In [78]:
```
# 0     1    0    0
# |     |    |    |
# 2^3   2^2  2^1  2^0
```

In [81]:
```
b & a

# internal calculation : multiplication bit by bit

# 1 0 1 0
# 0 1 0 0
```

Out[81]:
```
0
```

In [82]:
```
p = 3  # binary  = 0011
q = 10 # binary = 1010
```

In [83]:
```
# internal calculation : multiplication bit by bit
p & q
# 0 0 1 1
# 1 0 1 0
# after multiply bit by bit output = 0 0 1 0 (binary)
# 0      0    1     0
# |      |    |     |
# 2^3   2^2  2^1  2^0


# decimal of 0010 = 2
```

Out[83]:
```
2
```

In [ ]:
```
# Bitwise OR  syntax: |
```

In [84]:
```
x = 4  # binary  = 0100
y = 10 # binary  = 1010
```

In [85]:
```
# internal calculation : addition bit by bit
x | y

# 0 1 0 0
# 1 0 1 0

# after adding bit by bit output = 1 1 1 0 (binary)
# 8+4+2+0
```

Out[85]: 14

In [89]:
```
# Bitwise not syntax : ~

s = 10 # 1010

# -1010
# -(1010+1)= -(1011) = -(11)

~s

```

Out[89]: -11

In [90]:
```
~4 # 0100
# one's complement = (0100+1) = -(0101)
```

Out[90]: -5

In [ ]:
```

```

In [92]:
```
# Bitwise left shift

l = 5 # binary (0000 0101)
```

In [93]:
```
l << 1 # (0000 1010)  = 10
```

Out[93]: 10

In [94]:
```
l<<2 #(0001 0100)

# 0   0    0    1   0   1    0   0
#2^7  2^6  2^5 2^4 2^3 2^2 2^1 2^0

#0+0+0+16+0+4+0+0
```

Out[94]: 20

In [ ]:
```

```

In [ ]:
```

```

In [ ]:

```
1
```

In [ ]:

```
1
```