oprators and expressions in python

It is a symbol which is used to perform certain operations on given data is called operators. python has following operators: 1.Arthematic operators 2.Assignmet operators 3.Relational operators 4.Logical operators 5.Bitwise operators 6.Membership operator a.in b.not in 7.Identity operators a.is b.is not

1.Arthematic operation

Used to perform mathematical opertions,example +,-,*,/,//,%(*modulo/remainder),*(expo/power)

In [7]:
```
a=10
b=20
c=a+b
print(c)
```

```
30
```

In [9]:
```
10+20
```

Out[9]: 30

In [11]:
```
0+0
```

Out[11]: 0

In [13]:
```
print(-1+0)
```

```
-1
```

In [15]:
```
1-1
```

Out[15]: 0

In [17]:
```
2-0
```

Out[17]: 2

In [19]:
```
print(-1-0)
```

```
-1
```

In [21]:
```
10*20
```

Out[21]: 200

In [23]:
```
30*50
```

Out[23]: 1500

In [25]:
```
a=8.7
b=9.9
print(a*b)
```

```
86.13
```

In [27]:
```
9.9*4+7j
```

Out[27]: (39.6+7j)

In [29]:
```
2/7
```

Out[29]: 0.2857142857142857

In [31]:
```
2/7
```

Out[31]: 0.2857142857142857

In [33]:
```
2//7
```

Out[33]: 0

In [35]:
```
2.3/99
```

Out[35]: 0.02323232323232323

In [39]:
```
2.3//99
```

Out[39]: 0.0

In [41]:
```
25%5
```

Out[41]: 0

In [43]:
```
24%5
```

Out[43]: 4

In [45]:
```
2**6
```

Out[45]: 64

In [47]:
```
4**2  #4*4
```

Out[47]: 16

2.Assignment operators

To assign RHS value to the LHS variable -single line assignment -multiline assignment

In [56]:
```
a=10
print(a)
```

```
10
```

In [58]:
```
a,b=10,30
print(a,b)
```

```
10 30
```

In [60]:
```
a,b='hello','print'
print(a,b)
```

```
hello print
```

In [62]:
```
a,b=40,80
print(a+b)
```

```
120
```

3.Relational operators

To compare two value ,result can be eithe True or False <,>,==,!=,<=,>=

In [69]:
```
a=10
b=10
print(a==b)
```

```
True
```

In [71]:
```
a=10
b=20
print(a==b)
```

```
False
```

In [73]:
```
10!=20
```

Out[73]: True

In [75]:
```
10!=10
```

Out[75]: False

In [77]:
```
10>20
```

Out[77]: False

In [79]:
```
20<10
```

Out[79]: False

In [81]:
```
10>20
```

Out[81]: False

In [83]:
```
10<20
```

Out[83]: True

In [85]:
```
30<=20
```

Out[85]: False

In [87]:
```
30>=30
```

Out[87]: True

4.logical operators

Used to compare values 1.and #T and T=T, all F 2.or #F and F=F, all T 3.not #T becomes F and F becomes T

In [95]:
```
10==20 and 2==3
```

Out[95]: False

In [97]:
```
10==10 and 10==10
```

Out[97]: True

In [99]:
```
False and False
```

Out[99]: False

In [101…
```
False and True
```

Out[101… False

In [103…
```
True and False
```

Out[103… False

In [105…
```
True and True
```

Out[105… True

In [107…
```
10<20 and 20>30
```

Out[107… False

In [109…
```
False or False
```

Out[109… False

In [111…
```
False or True
```

Out[111… True

In [113…
```
True or False
```

Out[113… True

In [115…
```
True or  Ture
```

Out[115… True

In [117…
```
10==30 or 10==10
```

Out[117… True

In [119…
```
10==20 and 20==30
```

Out[119… False

In [121…
```
10<20 and 10<20
```

Out[121… True

In [123…
```
#special points:
"PYTHON"=="PYTHON"
```

Out[123… True

In [125…
```
"PYHTON">="python"
```

Out[125… False

In [127…
```
"INDIA">="INDIA"
```

Out[127… True

In [129…
```
'PYTHON'>'python'
```

Out[129… False

In [131…
```
"INDIA">='INDIa'
```

Out[131… False

In [133…
```
"INDIa">='INDIA'
```

Out[133… True

In [135…
```
100 and 200
```

Out[135… 200

In [137…
```
100 and 0
```

Out[137… 0

In [139…
```
-123 and -879
```

Out[139… -879

In [141…
```
100 and 300 and 0
```

Out[141… 0

In [143…
```
-123 and 432 and 0
```

Out[143… 0

In [145…
```
100 or 0
```

Out[145… 100

In [147…
```
399 or 0
```

Out[147… 399

In [149…
```
-123 or 0
```

Out[149… -123

In [151…
```
-123 or -234 or 0 or 87
```

Out[151… -123

In [163…
```
'pyhton' or 'java' or 'alina'
```

Out[163… 'pyhton'

In [155…
```
'python' and 'java' and 'alina'
```

Out[155… 'alina'

In [157…
```
not False
```

Out[157… True

In [159…
```
not True
```

Out[159… False

In [161…
```
not not True
```

Out[161… True

In [165…
```
not 133
```
Out[165… False

In [167…
```
not -0
```
Out[167… True

In [169…
```
not -122
```
Out[169… False

5.Bitwise operators

Internally,bitwise operator convert values from int to binary format. bitwise operator are those which are applicable on int only,some cases bool aslo.<<,>>,|,&,^,~

apply for object dection or | ,0 | 0 =0 ,all 1 and & ,1 $ 1=1 ,all 0 xor ^, 0 ^ 0=0 ,0 ^ 1 ^ 1=1,similar is 0, and dis-similar is 1left shift means we gain the bits right shift me we lose the bits

In [2]:
```
10<<2
```
Out[2]: 40

In [4]:
```
10<<3
```
Out[4]: 80

In [6]:
```
10>>3
```
Out[6]: 1

In [13]:
```
10>>5
```
Out[13]: 0

10>>2

In [2]:
```
a=10
b=20
a<<b
```
Out[2]: 10485760

In [ ]:
```

```

In [15]:
```
10 << 30
```
Out[15]: 10737418240

In [6]:
```
2 << 3
```
Out[6]: 16

In [8]:
```
True << False
```
Out[8]: 1

In [12]:
```
3.6j << 3.8j
```
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[12], line 1
----> 1 3.6j << 3.8j

TypeError: unsupported operand type(s) for <<: 'complex' and 'complex'
```

In [14]:
```
55.5 << 8
```
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[14], line 1
----> 1 55.5 << 8

TypeError: unsupported operand type(s) for <<: 'float' and 'int'
```

In [16]:
```
True+False << True
```

Out[16]: 2

In [18]:
```
100 >> 30
```

Out[18]: 0

In [20]:
```
449 >> 549
```

Out[20]: 0

In [22]:
```
2.8 >> 8
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[22], line 1
----> 1 2.8 >> 8

TypeError: unsupported operand type(s) for >>: 'float' and 'int'
```

In [24]:
```
True >> False
```

Out[24]: 1

In [26]:
```
True >> 7
```

Out[26]: 0

In [28]:
```
11 | 88
```

Out[28]: 91

In [30]:
```
True | 77
```

Out[30]: 77

In [32]:
```
89 | 89
```

Out[32]: 89

In [34]:
```
00 | 00
```

Out[34]: 0

In [36]:
```
bin(2)
```

Out[36]: '0b10'

In [38]:
```
2 | 2 #1 0 and 1 0=10 ,10=2
```

Out[38]: 2

In [42]:
```
6 | 8
```

Out[42]: 14

In [48]:
```
bin(4)  #100 and 100=100 =4
```

Out[48]: '0b100'

In [50]:
```
4 & 4
```

Out[50]: 4

In [52]:
```
bin(8) #1000 and 1000=1000 =8
```

Out[52]: '0b1000'

In [54]:
```
8 &8
```

Out[54]: 8

In [56]:
```
bin(3) #for ^ 11 and 11=00 =0
```

Out[56]: '0b11'

In [58]:
```
3 ^ 3
```

Out[58]: 0

In [60]:
```
8 ^ 8
```

Out[60]: 0

In [62]:
```
7^ 8
```

Out[62]: 15

In [64]:
```
76 ^ 76
```

Out[64]: 0

In [66]:
```
True ^ False #1 and 0=1 ,True
```

Out[66]: True

In [70]:
```
print(~10)
```

```
-11
```

In [72]:
```
print(~12)
```

```
-13
```

In [76]:
```
print(~-34)
```

```
33
```

In [78]:
```
print(~True)
```

```
-2
```

Membership operators

Used to check whether the value is present in iterable obj(contain more thn one value) or not. ex in(True when present) ,not in(True when not present)if str work on data with '' single.

In [82]:
```
a=10,30,550
10 in a
```

Out[82]: True

In [85]:
```
a='hello','pyhton','hell'
'h' in a
```

Out[85]: False

In [87]:
```
a='hello'
'h' in a
```

Out[87]: True

In [89]:
```
b='hell','pr'
'p' in b
```

Out[89]: False

In [91]:
```
b='hellhell'
'h' in b
```

Out[91]: True

In [93]:
```
'g' not in b
```

Out[93]: True

In [95]:
```
'h' not in b
```

Out[95]: False

7.Identity operators

Used to compare memory address of two obj,ex 'is' print True if addd is same ,and 'is not' print True if add is different.

In [98]:
```
a=10,80,80
b=89,98,76,55
```

In [100…
```
print(a,id(a))
print(b,id(b))
```

```
(10, 80, 80) 3027350676928
(89, 98, 76, 55) 3027351035296
```

In [102…
```
a is b
```

Out[102… False

In [104…
```
a is not b
```

Out[104… True

In [106…
```
b is a,b is not a
```

Out[106… (False, True)

In [111…
```
s={20,76,87}
b={39,989,98}
```

In [113…
```
s is b
```

Out[113… False

In [115…
```
s is not b
```

Out[115… True

In [117…
```
s1="python"
s2="python"
```

In [119…
```
s1 is s2
```

Out[119… True

In [121…
```
s1 is not s2
```

Out[121… False