Operators

1.Arithmetic operators

In [2]:

```
a = 15
b = 5
```

In [3]:

```
print(a + b) #Addition
print(a - b) #subtraction
print(a * b) #multiplication
print(a / b) # float division
print( a // b) #int division
print( a % b) #modulus
```

```
20
10
75
3.0
3
0
```

Assignment operator

In [4]:

```
x =2
x
```

Out[4]:

```
2
```

In [5]:

```
x = x+2 #x is incremented by 2
x
```

Out[5]:

```
4
```

In [7]:

```
x += 2
x
```

Out[7]:

```
8
```

In [8]:

```
x -= 2
x
```

Out[8]:

```
6
```

In [9]:

```
x *= 2
x
```

Out[9]:

```
12
```

```
In [11]: x /= 2
         x
```

```
Out[11]: 3.0
```

```
In [16]: x //= 2
         x
```

```
Out[16]: 0.0
```

```
In [17]: a,b = 5,6
         print(a)
         print(b)
```

```
5
6
```

```
In [18]: a = 5
         b = 6
         print(a)
         print(b)
```

```
5
6
```

3.Unary operator

```
In [20]: n = 7
         n
```

```
Out[20]: 7
```

```
In [21]: m = -(n) #negation
         m
```

```
Out[21]: -7
```

```
In [22]: n
```

```
Out[22]: 7
```

```
In [23]: -n
```

```
Out[23]: -7
```

4.Relational operator

```
In [24]: # relational operators are used to compare values
         a = 5
         b= 6
```

```
In [25]: a < b
```

```
Out[25]: True
```

```
In [26]: a > b
```

```
Out[26]: False
```

```
In [27]: a == b
```

```
Out[27]: False
```

```
In [28]: a != b
```

```
Out[28]: True
```

```
In [29]: b = 5
         a = 5
         a == b
```

```
Out[29]: True
```

```
In [30]: a > b
```

```
Out[30]: False
```

```
In [32]: b >= a
```

```
Out[32]: True
```

```
In [33]: a < b
```

```
Out[33]: False
```

```
In [34]: a <= b
```

```
Out[34]: True
```

```
In [35]: b = 7
         a != b
```

```
Out[35]: True
```

5.Logical operator

```
In [36]: a = 5
         b = 4
```

```
In [37]: a < 8 and b < 5
```

```
Out[37]:  True

In [38]:  a < 8 and b < 2

Out[38]:  False

In [39]:  a < 8 or b < 2

Out[39]:  True

In [40]:  a > 8 or b < 2

Out[40]:  False

In [41]:  x = False
          x

Out[41]:  False

In [42]:  not x

Out[42]:  True

In [43]:  x = not x
          x

Out[43]:  True

In [44]:  x

Out[44]:  True

In [45]:  not x

Out[45]:  False
```

6.Bitwise operator

```
In [92]:  # complement (its a 2's complement of a number i.e 1's complement + 1
          ~12

Out[92]:  -13

In [93]:  ~45

Out[93]:  -46

In [94]:  ~56

Out[94]:  -57
```

```
In [95]:  ~-11

Out[95]:  10

In [96]:  #bitwise and,or operator
          12 & 13

Out[96]:  12

In [97]:  12 | 13

Out[97]:  13

In [100…  print(1 & 0)
          print(1 | 0)

         0
         1

In [102…  #XOR
          print(1 ^ 1)
          print(1 ^ 0)
          print(0 ^ 1)
          print( 0 ^ 0)

         0
         1
         1
         0

In [103…  12 ^ 13

Out[103…  1

In [104…  print(bin(25))
          print(bin(35))

         0b11001
         0b100011

In [105…  25 ^ 35

Out[105…  58

In [106…  bin(58)

Out[106…  '0b111010'

In [107…  #left shift : shift the bits to the left
          # (u get extra bits i.e gaining zerors at the right)
          bin(10)

Out[107…  '0b1010'
```

```
In [108…  10 << 1

Out[108…  20

In [110…  10 << 2

Out[110…  40

In [111…  10 << 3

Out[111…  80

In [112…  20 << 4

Out[112…  320

  In [ ]: #Right shift : shifting the bits to the right
          # we are going to loss the  bits

In [113…  bin(10)

Out[113…  '0b1010'

In [115…  10 >> 1

Out[115…  5

In [116…  10 >> 2

Out[116…  2

In [117…  10 >> 3

Out[117…  1

In [118…  bin(20)

Out[118…  '0b10100'

In [119…  20 >> 2

Out[119…  5
```

Number System

```
 In [46]: 25

Out[46]: 25
```

```
In [47]: bin(25) #binary number system

Out[47]: '0b11001'

In [50]: int(0b11001) #decimal system

Out[50]: 25

In [51]: bin(30)

Out[51]: '0b11110'

In [53]: int(0b110011)

Out[53]: 51

In [54]: oct(25) #octal system

Out[54]: '0o31'

In [55]: int(0o31)

Out[55]: 25

In [56]: oct(32)

Out[56]: '0o40'

In [57]: int(0o40)

Out[57]: 32

In [58]: bin(7)

Out[58]: '0b111'

In [59]: oct(7)

Out[59]: '0o7'

In [60]: hex(25) #hexadecimal system

Out[60]: '0x19'

In [61]: int(0x19)

Out[61]: 25

In [67]: hex(15)

Out[67]: '0xf'
```

In [68]:

```
0xa
```

Out[68]:

```
10
```

In [69]:

```
0xf
```

Out[69]:

```
15
```

In [70]:

```
hex(10)
```

Out[70]:

```
'0xa'
```

In [71]:

```
hex(256)
```

Out[71]:

```
'0x100'
```

In [74]:

```
print(bin(25))
print(int(25))
print(oct(25))
print(hex(25))
```

```
0b11001
25
0o31
0x19
```

swapping two numbers

In [75]:

```
a = 5
b = 6
```

In [76]:

```
a = b
b = a
print(a)
print(b)
```

```
6
6
```

In [77]:

```
# in above case we lost the value of a
a = 7
b =8
```

In [78]:

```
# number swapping with the help of 3rd variable
temp = a
a = b
b = temp
```

In [79]:

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

```
8
7
```

In [80]:

```
#number swapping without 3rd variable
a = 6
b = 3
a = a+b
b = a-b
a = a-b
```

In [81]:

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

```
3
6
```

In [87]:

```
#swapping using XOR
a = 5
b = 6
```

In [88]:

```
a = a ^ b #using XOR we can save memory
b = a ^ b
a = a ^ b
```

In [89]:

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

```
6
5
```

In [91]:

```
#swapping using rot_two()
# it swaps the two top most stack items using rotational concept
a = 8
b = 5
a , b = b , a
print(a)
print(b)
```

```
5
8
```

Range()

In [121…

```
r = range(0,10)
r
```

Out[121…

```
range(0, 10)
```

In [122…

```
type(r)
```

Out[122…

```
range
```

In [123…

```
list(range(10,20))
```

```
Out[123…  [10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

```
In [124…  r1 = list(r)
          r1
```

```
Out[124…  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
In [125…  even_num = list(range(2,10,2))
          even_num
```

```
Out[125…  [2, 4, 6, 8]
```

Math Module

```
In [126…  x = sqrt(16)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[126], line 1
----> 1 x = sqrt(16)

NameError: name 'sqrt' is not defined
```

```
In [127…  import math
          x = math.sqrt(16)
          x
```

```
Out[127…  4.0
```

```
In [128…  print(math.sqrt(25))
```

```
5.0
```

```
In [131…  print(math.floor(3.6)) #floor returns least integer value
```

```
3
```

```
In [132…  print(math.ceil(3.6)) #returns maximum integer value
```

```
4
```

```
In [133…  print(math.pow(3,2))
```

```
9.0
```

```
In [134…  print(math.pi)
```

```
3.141592653589793
```

```
In [135…  print(math.e)
```

```
2.718281828459045
```

```
In [136…  import math as m
          m.sqrt(10)
```

```
Out[136…   3.1622776601683795
```

```
In [137…   from math import sqrt,pow
           pow(2,3)
```

```
Out[137…   8.0
```

```
In [141…   #to import all functions in math moudule
           from math import *
           print(sqrt(16))
           print(pow(7,2))
```

```
4.0
49.0
```

input()

```
In [142…   x = input()
           y = input()
           z = x + y
           print(z)
```

```
4 4
```

```
In [144…   x = input('enter 1st number')
           y = input('enter 2nd number')
           z = x + y
           print(z)
```

```
44
```

```
In [145…   type(x)
```

```
Out[145…   str
```

```
In [146…   type(y)
```

```
Out[146…   str
```

```
In [147…   x = int(input('Enter the 1st number'))
           y = int(input('Enter the 2nd number'))
           z = x + y
           print(z)
```

```
8
```

```
In [150…   ch = input('enter a char')
           print(ch)
```

```
python
```

```
In [151…   print(ch[0])
```

```
p
```

In [152…

```
print(ch[1])
```

```
y
```

In [153…

```
ch = input('enter a char')[0]
print(ch)
```

```
p
```

In [154…

```
ch = input('enter a string')[1:4]
print(ch)
```

```
yth
```

In [155…

```
s = input('enter a string')
print(s)
```

```
2 + 3 - 5
```

In [158…

```
# Eval function
res = eval(input('enter an expression'))
print(res)
```

```
1.0
```