```
In [2]: 25

Out[2]: 25

In [3]: bin(25)

Out[3]: '0b11001'

In [4]: int(0b11001)

Out[4]: 25

In [5]: bin(35)

Out[5]: '0b100011'

In [6]: int(0b100011)

Out[6]: 35

In [7]: bin(20)

Out[7]: '0b10100'

In [8]: int(0b10100)

Out[8]: 20

In [9]: oct(15)

Out[9]: '0o17'

In [10]: int(0o17)

Out[10]: 15

In [11]: hex(9)

Out[11]: '0x9'

In [12]: oxf
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[12], line 1
----> 1 oxf

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

```
In [13]: hex(10)

Out[13]: '0xa'
```

```
In [14]: hex(25)

Out[14]: '0x19'
```

Swap Variable

```
In [15]: a = 8
         b= 5
         x= a+b

In [16]: x

Out[16]: 13

In [17]: a = x-a
         b= x-b

In [18]: a

Out[18]: 5

In [19]: b

Out[19]: 8

In [20]: a1 = 7
         b1 = 8

In [22]: temp = a1
         temp

Out[22]: 7

In [23]: a1 = b1
         a1

Out[23]: 8

In [25]: b1 = temp
         b1

Out[25]: 7

In [26]: a2 = 9
         b2 = 4
         a2 = a2+b2
         b2 = a2-b2
         a2 = a2-b2

In [27]: a2
```

```
Out[27]: 4
```

```
In [28]: b2
```

```
Out[28]: 9
```

```
In [30]: print(0b101)
         print(0b110)
```

```
5
6
```

```
In [33]: print(bin(11))
         print(0b1011)
```

```
0b1011
11
```

Bitwise Operators

we have 6 operators (~)||AND(&)||or(|)||XOR(^)||LEFTSHIFT(<<)||Rightshift(>>)

```
In [34]: ~12
```

```
Out[34]: -13
```

```
In [35]: ~45
```

```
Out[35]: -46
```

```
In [36]: ~6
```

```
Out[36]: -7
```

```
In [37]: ~-6 # how negative compliment works
```

```
Out[37]: 5
```

```
In [38]: ~-1
```

```
Out[38]: 0
```

And operator

1&1 is 1

```
In [39]: 12&13
```

```
Out[39]: 12
```

```
In [40]: 12|13
```

Out[40]:
```
13
```

In [41]:
```
1&1
```

Out[41]:
```
1
```

In [42]:
```
1&0
```

Out[42]:
```
0
```

In [43]:
```
35 & 40
```

Out[43]:
```
32
```

In [44]:
```
35|40
```

Out[44]:
```
43
```

In [45]:
```
32|40
```

Out[45]:
```
40
```

In [46]:
```
12^13 # in XOR if both numbers are different then we will get 1 otherwise will
```

Out[46]:
```
1
```

In [47]:
```
25^30
```

Out[47]:
```
7
```

In [48]:
```
bin(25)
```

Out[48]:
```
'0b11001'
```

In [49]:
```
bin(30)
```

Out[49]:
```
'0b11110'
```

In [50]:
```
int(0b000111)
```

Out[50]:
```
7
```

Bitwise left Operator

bit wise left operator by default it will take 2 zeros() 10 binary operator is 1010|

In [51]:
```
10<<2
```

Out[51]:
```
40
```

In [52]:
```
20<<4
```

```
Out[52]: 320
```

Bitwise right Operator

```
In [54]: 10>>2

Out[54]: 2

In [55]: bin(20)

Out[55]: '0b10100'

In [56]: 20>>4

Out[56]: 1
```

### import math module

https://docs.python.org/3/library/math.html

```
In [57]: x = sqrt(25)

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[57], line 1
----> 1 x = sqrt(25)

NameError: name 'sqrt' is not defined

In [60]: import math # importing math module

In [61]: x = sqrt(25)

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[61], line 1
----> 1 x = sqrt(25)

NameError: name 'sqrt' is not defined

In [62]: x = math.sqrt(25)

In [63]: x

Out[63]: 5.0

In [64]: x1 = math.sqrt(15)
         x1

Out[64]: 3.872983346207417
```

In [66]:
```
print(math.floor(2.9)) # floor- minimum or least value
```
```
2
```

In [68]:
```
print(math.ceil(2.9))
```
```
3
```

In [69]:
```
print(math.pow(3,2))
```
```
9.0
```

In [70]:
```
print(math.pi)
```
```
3.141592653589793
```

In [71]:
```
print(math.e)
```
```
2.718281828459045
```

In [72]:
```
import math as m # alias word to make user comfortable while scripting large co
m.sqrt(10)
```
Out[72]:
```
3.1622776601683795
```

In [74]:
```
from math import sqrt,pow #math has many functionas if you want to call specif
```

In [75]:
```
print(pow(2,3))
print(floor(2,3))
```
```
8.0
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[75], line 2
      1 print(pow(2,3))
----> 2 print(floor(2,3))

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

In [76]:
```
from math import pow,floor
```

In [78]:
```
print(pow(2,3))
print(floor(2.73))
```
```
8.0
2
```

In [79]:
```
from math import * # * function imports all functions in math
```

In [80]:
```
print(pow(2,3))
print(floor(2.3))
```
```
8.0
2
```

In [81]:
```
round(pow(2,3))
```
Out[81]:
```
8
```

User input function in Python || Command line input

In [82]:
```
x = input('Enter the 1st number') # input function always returns the string ou
y = input('Enter the 2nd number') # it goes with concatination, arithamtic ope
z = x+y
print(z) # console wait for user to entert the input
```

```
Enter the 1st number25
Enter the 2nd number50
2550
```

In [83]:
```
type(x)
type(y)
```

Out[83]:
```
str
```

In [84]:
```
print(type(x))
print(type(y))
```

```
<class 'str'>
<class 'str'>
```

In [85]:
```
x1 = int(input('enter 1st number'))
y1 = int(input('enter 2nd number'))
z1 = x1+y1
print(z1)
```

```
enter 1st number12
enter 2nd number12
24
```

lets take input from the user in char format, but we dont have char foramt in python

In [90]:
```
ch = input('enter a char')
print(ch)
```

```
enter a chari love python
i love python
```

In [91]:
```
print(ch[0])
```

```
i
```

In [92]:
```
print(ch[1])
```

```

```

In [93]:
```
print(ch[-1])
```

```
n
```

In [94]:
```
ch = input('enter a char')[1:3]
print(ch)
```

```
enter a charpython
yt
```

In [95]:
```
ch = input('enter a char')
print(ch)
```

```
enter a char2+6-1
2+6-1
```

in the above example eval function helps to evaluate the expression

In [96]:
```
result = eval(input('enter an expression'))
print(result)
```

```
enter an expression3+9-2
10
```

In [ ]: