PYTHON OPERATOR

In [4]:
```
help(list)
```

```
Help on class list in module builtins:

class list(object)
 |  list(iterable=(), /)
 |
 |  Built-in mutable sequence.
 |
 |  If no argument is given, the constructor creates a new empty list.
 |  The argument must be an iterable if specified.
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return bool(key in self).
 |
 |  __delitem__(self, key, /)
 |      Delete self[key].
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __getitem__(self, index, /)
 |      Return self[index].
 |
 |  __gt__(self, value, /)
 |      Return self>value.
 |
 |  __iadd__(self, value, /)
 |      Implement self+=value.
 |
 |  __imul__(self, value, /)
 |      Implement self*=value.
 |
 |  __init__(self, /, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __iter__(self, /)
 |      Implement iter(self).
 |
 |  __le__(self, value, /)
 |      Return self<=value.
 |
 |  __len__(self, /)
 |      Return len(self).
 |
 |  __lt__(self, value, /)
 |      Return self<value.
 |
 |  __mul__(self, value, /)
 |      Return self*value.
 |
```

```
 |  __ne__(self, value, /)
 |      Return self!=value.
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |  __reversed__(self, /)
 |      Return a reverse iterator over the list.
 |
 |  __rmul__(self, value, /)
 |      Return value*self.
 |
 |  __setitem__(self, key, value, /)
 |      Set self[key] to value.
 |
 |  __sizeof__(self, /)
 |      Return the size of the list in memory, in bytes.
 |
 |  append(self, object, /)
 |      Append object to the end of the list.
 |
 |  clear(self, /)
 |      Remove all items from list.
 |
 |  copy(self, /)
 |      Return a shallow copy of the list.
 |
 |  count(self, value, /)
 |      Return number of occurrences of value.
 |
 |  extend(self, iterable, /)
 |      Extend list by appending elements from the iterable.
 |
 |  index(self, value, start=0, stop=9223372036854775807, /)
 |      Return first index of value.
 |
 |      Raises ValueError if the value is not present.
 |
 |  insert(self, index, object, /)
 |      Insert object before index.
 |
 |  pop(self, index=-1, /)
 |      Remove and return item at index (default last).
 |
 |      Raises IndexError if list is empty or index is out of range.
 |
 |  remove(self, value, /)
 |      Remove first occurrence of value.
 |
 |      Raises ValueError if the value is not present.
 |
 |  reverse(self, /)
 |      Reverse *IN PLACE*.
 |
 |  sort(self, /, *, key=None, reverse=False)
 |      Sort the list in ascending order and return None.
 |
 |      The sort is in-place (i.e. the list itself is modified) and stable (i.e.
the
 |      order of two equal elements is maintained).
```

```
 |
 |      If a key function is given, apply it once to each list item and sort the
m,
 |      ascending or descending, according to their function values.
 |
 |      The reverse flag can be set to sort in descending order.
 |
 |  ----------------------------------------------------------------------
 |  Class methods defined here:
 |
 |  __class_getitem__(...)
 |      See PEP 585
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs)
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  __hash__ = None
```

In [6]:
```
help(tuple)
```

```
Help on class tuple in module builtins:

class tuple(object)
 |  tuple(iterable=(), /)
 |
 |  Built-in immutable sequence.
 |
 |  If no argument is given, the constructor returns an empty tuple.
 |  If iterable is specified the tuple is initialized from iterable's items.
 |
 |  If the argument is a tuple, the return value is the same object.
 |
 |  Built-in subclasses:
 |      asyncgen_hooks
 |      MonthDayNano
 |      UnraisableHookArgs
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return bool(key in self).
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __getitem__(self, key, /)
 |      Return self[key].
 |
 |  __getnewargs__(self, /)
 |
 |  __gt__(self, value, /)
 |      Return self>value.
 |
 |  __hash__(self, /)
 |      Return hash(self).
 |
 |  __iter__(self, /)
 |      Implement iter(self).
 |
 |  __le__(self, value, /)
 |      Return self<=value.
 |
 |  __len__(self, /)
 |      Return len(self).
 |
 |  __lt__(self, value, /)
 |      Return self<value.
 |
 |  __mul__(self, value, /)
 |      Return self*value.
 |
```

```
|  __ne__(self, value, /)
|      Return self!=value.
|
|  __repr__(self, /)
|      Return repr(self).
|
|  __rmul__(self, value, /)
|      Return value*self.
|
|  count(self, value, /)
|      Return number of occurrences of value.
|
|  index(self, value, start=0, stop=9223372036854775807, /)
|      Return first index of value.
|
|      Raises ValueError if the value is not present.
|
|  ----------------------------------------------------------------------
|  Class methods defined here:
|
|  __class_getitem__(...)
|      See PEP 585
|
|  ----------------------------------------------------------------------
|  Static methods defined here:
|
|  __new__(*args, **kwargs)
|      Create and return a new object.  See help(type) for accurate signature.
```

In [8]:
```
num=5
id(num)
```

Out[8]:
```
140728634718776
```

Arithmetic operator

In [12]:
```
x,y=10,5
```

In [15]:
```
x+y
```

Out[15]:
```
15
```

In [17]:
```
x-y
```

Out[17]:
```
5
```

In [19]:
```
x*y
```

Out[19]:
```
50
```

In [21]:
```
x/y
```

Out[21]:
```
2.0
```

In [23]:
```
x//y
```

Out[23]: 2

In [25]:
```
x%y
```

Out[25]: 0

In [27]:
```
x**y
```

Out[27]: 100000

In [29]:
```
2**4
```

Out[29]: 16

+,-,*,/,% -- Arithmetic operator

Assignment operator

In [57]:
```
x=2
```

In [59]:
```
x=x+2
```

In [61]:
```
x
```

Out[61]: 4

In [63]:
```
x+=2
x
```

Out[63]: 6

In [65]:
```
x*=2
x
```

Out[65]: 12

In [67]:
```
x/=2
x
```

Out[67]: 6.0

In [69]:
```
a,b=5,6
```

In [71]:
```
a
```

Out[71]: 5

In [75]:
```
b
```

Out[75]: 6

+=,-=,*=,/= Assignment operator

Unary operator

In [92]:
```
n=7 # nagation
```

In [94]:
```
m=-(n)
```

In [96]:
```
m
```

Out[96]: -7

In [98]:
```
n
```

Out[98]: 7

In [100…
```
-n
```

Out[100… -7

Relational operatior

In [104…
```
a=5
b=7
```

In [106…
```
a==b
```

Out[106… False

In [108…
```
a<b
```

Out[108… True

In [110…
```
b<a
```

Out[110… False

In [116…
```
b=5
```

In [118…
```
a==b
```

Out[118… True

In [120…
```
a=10
```

In [122…
```
a!=b
```

Out[122… True

In [124…
```
b=10
```

In [126…
```
a==b
```

Out[126… True

In [128…
```
a>=b
```

Out[128… True

In [130…
```
a<=b
```

Out[130… True

In [132…
```
a<b
```

Out[132… False

In [134…
```
a>b
```

Out[134… False

In [138…
```
b=7
```

In [144…
```
a!=b
```

Out[144… True

==,<,>,<=,>= Relational operatior

In [148…
```
a=5
b=4
```

In [152…
```
a<8 & b<2
```

Out[152… False

In [154…
```
a>8 & b<2
```

Out[154… True

In [158…
```
a<8 | b<2
```

Out[158… False

In [160…
```
x=False
x
```

Out[160… False

In [162…
```
not x
```

Out[162… True

In [166…
```
x=not x
x
```

Out[166…
```
False
```

In [168…
```
not x
```

Out[168…
```
True
```

Number system conversion

In [171…
```
25
```

Out[171…
```
25
```

In [173…
```
bin(25)
```

Out[173…
```
'0b11001'
```

In [175…
```
0b11001
```

Out[175…
```
25
```

In [177…
```
int(0b11001)
```

Out[177…
```
25
```

In [179…
```
bin(35)
```

Out[179…
```
'0b100011'
```

In [181…
```
int(0b100011)
```

Out[181…
```
35
```

In [183…
```
bin(20)
```

Out[183…
```
'0b10100'
```

In [185…
```
int(0b10100)
```

Out[185…
```
20
```

In [187…
```
0b1111
```

Out[187…
```
15
```

In [189…
```
oct(15)
```

Out[189…
```
'0o17'
```

In [191…
```
0o17
```

Out[191… 15

In [193…
```
hex(9)
```

Out[193… '0x9'

In [195…
```
0xf
```

Out[195… 15

In [197…
```
hex(10)
```

Out[197… '0xa'

In [199…
```
0xa
```

Out[199… 10

In [201…
```
hex(25)
```

Out[201… '0x19'

In [203…
```
0x19
```

Out[203… 25

In [205…
```
0x15
```

Out[205… 21

Swap variable in python

In [253…
```
a=5
b=6
```

In [255…
```
a=b
b=a
```

In [257…
```
print(a)
print(b)
```

```
6
6
```

In [265…
```
a=5 #swap 2 number using 3 rd variable
b=7
```

In [267…
```
temp=a
a=b
b=temp
```

In [269…
```
print(a)
print(b)
```

```
7
5
```

In [280…
```
a1=8 # without using 3rd variable
b1=9
```

In [282…
```
a1,b1=b1,a1
```

In [284…
```
print(a1)
print(b1)
```

```
9
8
```

In [300…
```
a2=15
b2=21 #using xor swap 2 number
```

In [306…
```
a2=a2^b2
b2=a2^b2
a2=a2^b2
```

In [308…
```
print(a2)
print(b2)
```

```
15
21
```

Bitwise operator

complement

In [312…
```
~45
```

Out[312… -46

In [314…
```
~(85)
```

Out[314… -86

In [316…
```
~(-85)
```

Out[316… 84

bitwise AND ,OR,XOR

In [319…
```
12&13
```

Out[319… 12

In [321…
```
1&1
```

Out[321… 1

In [325…
```
1&0
```
Out[325…
```
0
```
In [327…
```
1|0
```
Out[327…
```
1
```
In [329…
```
1&0
```
Out[329…
```
0
```
In [331…
```
12|13
```
Out[331…
```
13
```
In [333…
```
35&40
```
Out[333…
```
32
```
In [335…
```
35|40
```
Out[335…
```
43
```
In [337…
```
40|35
```
Out[337…
```
43
```
In [339…
```
bin(12)
```
Out[339…
```
'0b1100'
```
In [341…
```
bin(13)
```
Out[341…
```
'0b1101'
```
In [345…
```
12^13 #XOR^
```
Out[345…
```
1
```
In [347…
```
23^45
```
Out[347…
```
58
```
In [349…
```
20<<1
```
Out[349…
```
40
```
In [357…
```
20<<2
```
Out[357…
```
80
```
In [359…
```
20>>1
```
Out[359…
```
10
```

In [361…
```
20>>2
```

Out[361…
```
5
```

import math module

In [364…
```
x=sqrt(25)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[364], line 1
----> 1 x=sqrt(25)

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

In [366…
```
import math
```

In [368…
```
x1= math.sqrt(25)
```

In [370…
```
x1
```

Out[370…
```
5.0
```

In [374…
```
print(math.floor(2.9)) # floor gives minimum value .
```

```
2
```

In [380…
```
print(math.ceil(2.9)) # ceil gives maximum value
```

```
3
```

In [382…
```
print(math.pow(3,2))
```

```
9.0
```

In [386…
```
print(math.e)
```

```
2.718281828459045
```

In [392…
```
import math as m
m.sqrt(10)
```

Out[392…
```
3.1622776601683795
```

In [394…
```
import math as m
m.sqrt(88)
```

Out[394…
```
9.38083151964686
```

In [396…
```
from math import sqrt,pow
pow(2,3)
```

Out[396…
```
8.0
```

In [400…
```
round(pow(2,3))
```

Out[400… 8

In [408…
```
round(89/3,3)
```

Out[408… 29.667

user input function in python || command line input

In [4]:
```
x=input()
y=input()
z=x+y
print(z)
```

105

In [6]:
```
x1=input('enter the 1st number')
y1=input('enter the 2nd number')
z1=x1+y1
print(z1)
```

58

In [8]:
```
type(x1)
type(y1)
```

Out[8]: str

In [12]:
```
x2= input('Enter the 1st number')
a=int(x2)
y2=input('enter the 2nd number')
b=int(y2)
z1=a+b
print(z1)
```

17

In [14]:
```
x3=int(input('enter the 1st number'))
y3=int(input('enter the 2nd number'))
z3=x3+y3
print(z3)
```

17

let take input from user in character format

In [17]:
```
ch=input('enter a char')
print(ch)
```

hello

In [19]:
```
print(ch[0])
```

h

In [21]:
```
print(ch[1])
```

```
e
```

In [23]:
```
print(ch[0:3])
```

```
hel
```

In [1]:
```
ch=input('enter a char')[0:4]
print(ch)
```

```
hell
```

In [1]:
```
ch1=input('enter a char')
print(ch1)
```

```
2+3-4+10
```

Eval function using input

In [6]:
```
result=eval(input('enter an expr'))
print(result)
```

```
-35
```

In [ ]:
```

```

completed