Built-in Functions with List

all()	any()	enumerate()
len()	list()	max()
min()	sum()	sorted()

all() Function:
It returns True when all elements in the given iterable are true. If not, it returns False.

Syntax:
```
all(iterable)
```

all() Parameters
iterable - any iterable (list, tuple, dictionary, etc.) which contains the elements

The all() method returns:
True - If all elements in an iterable are true
False - If any element in an iterable is false

Truth table for all()

When	Return Value
All values are true	True
All values are false	False
One value is true (others are false)	False
One value is false (others are true)	False
Empty Iterable	True

NOTE: 0 and 1 are the binary values like False, True

Example:How all() works for tuple and lists?
```
s = [1, 3, 4, 5]
print(all(s))

s = [0, False]
print(all(s))

s = [0, False, 5]
print(all(s))

s = []
print(all(s))
```

Python any()
It Returns True if any element of an iterable is true. If not, this method returns False.

Syntax:
```
any(iterable)
```

The any method returns:
True if at least one element of an iterable is true
False if all elements are false or if an iterable is empty

When	Return Value

All values are true	True
All values are false	False
One value is true (others are false)	True
One value is false (others are true)	True
Empty Iterable	False

NOTE: 0 and 1 are the binary values like False, True

Example:
```
s =[1, 3, 4, 0]
print(any(s))

s = [0, False]
print(any(s))

s = [0, False, 5]
print(any(s))

s = []
print(any(s))
```

Python enumerate()
It adds counter to an iterable and returns it (the enumerate object).

Syntax
```
enumerate(iterable, start=0)
```

Parameters:
iterable: A sequence, an iterator, or objects that supports iteration
start(optional): It starts counting from this number. If start is omitted, 0 is taken as start.

How enumerate() works in Python?
```
BigData=['Big Data', 'Hadoop', 'Spark','Data Science']
eData = enumerate(BigData)
print(type(eData))
print(list(eData))
```

Looping Over an Enumerate object
```
bd = ['Big Data', 'Hadoop', 'Spark','Data Science']
for item in enumerate(bd):
  print(item)
```

Example:
```
bd = ['Big Data', 'Hadoop', 'Spark','Data Science']
for count, item in enumerate(bd):
  print(count, item)
```

Example:
```
bd = ['Big Data', 'Hadoop', 'Spark','Data Science']
for count, item in enumerate(bd, 100):
  print(count, item)
```

Example:
```
names = ['Bob', 'Alice', 'Guido']
print(list(enumerate(names)))
```

Example:
```
names = ['Bob', 'Alice', 'Guido']
for index, value in enumerate(names):
    print(f'{index}: {value}')
```

len() Function:
It displays length of characters in numeric format.

Syntax:
```
len(iterable)
```

Example:
```
PyList=[1,2,3,4,5]
print(len(PyList))
```

list():
It is converting into list data type.

Syntax:
```
list(iterable)
```

Example:
```
PyStr="Hello"
print(list(PyStr)) #['H', 'e', 'l', 'l', 'o']
```

max():
It is used to display max character based on ASCII or Unicode Value

Syntax:
```
max(iterable)
```

Example:
```
print(max(1,2,3,4))
print(max('a','b','c','d','E'))
```

min()
It is used to display min character based on ASCII or Unicode value.

Syntax:
```
min(iterable)
```

Example:
```
print(min(1,2,3,4))
print(min('a','b','c','d','E'))
```

sum()
It is used to display sum of values in the list, only for numeric values..

Syntax:
```
sum(iterable)
```

Example:
```
print(sum([1,2,3,4]))
```

sorted() for list, tupe & dictinary
It returns a sorted list of the specified iterable object.You can

specify ascending or descending order. Strings are sorted alphabetically, and numbers are sorted numerically.

Note: You cannot sort a list that contains BOTH string values AND numeric values.

Syntax
sorted(iterable, key=key, reverse=reverse)

Parameters:

iterable	Required. The sequence to sort, list, dictionary, tuple etc.
key	Optional. A Function to execute to decide the order. Default is None
reverse	Optional. A Boolean. False will sort ascending, True will sort descending. Default is False

Example:
```
a = ["b", "g", "a", "d", "f", "c", "h", "e"]#List
x = sorted(a)
print(x)
a = ("b", "g", "a", "d", "f", "c", "h", "e")#Tuple
x = sorted(a)
print(x)
a = {"b":"g", "a":"d", "f":"c", "h":"e"}#Dictionary
x = sorted(a)
print(x)
```

Example:
```
a = ["b", "g", "a", "d", "f", "c", "h", "e"]
x = sorted(a,reverse=True)
print(x)
a = ("b", "g", "a", "d", "f", "c", "h", "e")
x = sorted(a,reverse=True)
print(x)
a = {"b":"g", "a":"d", "f":"c", "h":"e"}
x = sorted(a,reverse=True)
print(x)
```

Example:
```
a = ["bb", "ggg", "aaaa", "dddd", "ff"]
x = sorted(a,key=len)
print(x)
a = ("bb", "ggg", "aaaa", "dddd", "ff")
x = sorted(a,key=len)
print(x)
a = {"bb":"ggg", "aaaa":"d", "ff":"c"}
x = sorted(a,key=len)
print(x)
```

Example:
```
a = {"bb":"ggg", "aaaa":"d", "ff":"c"}
x = sorted(a,key=len)
print(x)
print(sorted(a.values()))
print(sorted(a.keys()))
print(sorted(a.items()))
```

```
List_of_Lists
Example:
A=[1]*2
print(A)
B=[[1]*2]*5
print(B)
C=[[2,0]*1]*4
print(C)

Example:
A=[[1]*2]*5
print(A)
B=[[2,0]*1]*4
print(B)

Example:
#List_of_Lists
A=[2]*3
print(A)
B=[[2]*3]*2
print(B)
C=[[1,0]*1]*5
print(C)

Example:
List shortcuts
Fives = [5]*4
print(Fives)

Example:
A=[1,2,3,4,5]
print(A[0]*2)
print(A*2)
print(A[3]*3)
print(A[4]*1)
print(A*3)

Example:
PyList=[1,[2,[3,[4,[5]]]]]
print(PyList[0])
print(PyList[1][0])
print(PyList[1][1][0])

Example:
PyList=[1,[2,[3,[4,[5]]]]]
print(PyList)
print(PyList[0])
print(PyList[1])
print(PyList[1][0])
print(PyList[1][1])
print(PyList[1][1][0])
print(PyList[1][1][1])
print(PyList[1][1][1][0])
print(PyList[1][1][1][1])
print(PyList[1][1][1][1][0])
```

```
Example:
PyList=[1,[2,[3,[4,[5]]]]]
print(PyList)
print(PyList[0])
print(PyList[1])
print(PyList[1][1])
print(PyList[1][1][1])
print(PyList[1][1][1][1])
print(PyList[1][1][1][1][-1])
print(PyList[1][1][1][1][0])
```