MAP

map() function returns a map object(which is an iterator) of the results after applying the given function to each item of a given iterable (list, tuple etc.) map(fun, iter) fun : It is a function to which map passes each element of given iterable. iter : It is a iterable which is to be mapped.

Returns a list of the results after applying the given function
to each item of a given iterable (list, tuple etc.)

In [12]:

```
l=["1",'2','3','4']
l1=list(map(int,l))
print(l1)
print(sum(l1))
```

```
[1, 2, 3, 4]
10
```

LIST Methods

In [4]:

```
dir(str)
```

Out[4]:

```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getnewargs__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mod__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rmod__',
 '__rmul__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'capitalize',
 'casefold',
 'center',
 'count',
 'encode',
 'endswith',
 'expandtabs',
 'find',
 'format',
 'format_map',
 'index',
 'isalnum',
 'isalpha',
 'isascii',
 'isdecimal',
 'isdigit',
 'isidentifier',
 'islower',
 'isnumeric',
 'isprintable',
 'isspace',
 'istitle',
 'isupper',
 'join',
 'ljust',
 'lower',
```

```
 'lstrip',
 'maketrans',
 'partition',
 'replace',
 'rfind',
 'rindex',
 'rjust',
 'rpartition',
 'rsplit',
 'rstrip',
 'split',
 'splitlines',
 'startswith',
 'strip',
 'swapcase',
 'title',
 'translate',
 'upper',
 'zfill']
```

In [8]:

```
s="pyhton"
list1=[1,2,3,4]
s1=list1.capitalize()
print(s1)
```

```
---------------------------------------------------------------------------
-
AttributeError                            Traceback (most recent call las
t)
<ipython-input-8-f93c65d6e265> in <module>
      1 s="pyhton"
      2 list1=[1,2,3,4]
----> 3 s1=list1.capitalize()
      4 print(s1)

AttributeError: 'list' object has no attribute 'capitalize'
```

In [5]:

```
dir(list)
```

Out[5]:

```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__gt__',
 '__hash__',
 '__iadd__',
 '__imul__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__reversed__',
 '__rmul__',
 '__setattr__',
 '__setitem__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'append',
 'clear',
 'copy',
 'count',
 'extend',
 'index',
 'insert',
 'pop',
 'remove',
 'reverse',
 'sort']
```

In [11]:

```
help(list.append)
```

```
Help on method_descriptor:

append(self, object, /)
    Append object to the end of the list.
```

In [17]:

```
list1=[]
list1.append(9)
print(list1)
list1.append("python")
print(list1)
list1.append([1,2,3,4])
print(list1)
list1.append((2,3))
print(list1)
```

```
[9]
[9, 'python']
[9, 'python', [1, 2, 3, 4]]
[9, 'python', [1, 2, 3, 4], (2, 3)]
```

In [21]:

```
l=5
list1=[]
print("enter the elements")
for ele in range(l):
    list1.append(int(input()))
print(list1)
```

```
enter the elements
1
2
3
4
5
[1, 2, 3, 4, 5]
```

help(list.append) append(self, object, /) Append object to the end of the list.

In [17]:

```
list1=[20,30,40]
list1.append("end")
print(list1)
```

```
[20, 30, 40, 'end']
```

clear' clear(self, /) Remove all items from list.

In [22]:

```
list1=[20, 30, 40, 'end']
print(list1)
list1.clear()
print(list1)
```

```
[20, 30, 40, 'end']
[]
```

copy copy(self, /) Return a shallow copy of the list.

In [30]:

```
help(list.copy)
list2=[30,40]
list1=[20, 30, 40, 'end',90]
print(list2)
list2=list1.copy()
print(list2)
list1[2]=100
print(list1)
print(list2)
```

```
Help on method_descriptor:

copy(self, /)
    Return a shallow copy of the list.

[30, 40]
[20, 30, 40, 'end', 90]
[20, 30, 100, 'end', 90]
[20, 30, 40, 'end', 90]
```

In [26]:

```
l1=[1,2,3]
l2=[9,4,6]
print(l2)
l2=l1
print(l2)
l1[1]=100
print(l1)
print(l2)
```

```
[9, 4, 6]
[1, 2, 3]
[1, 100, 3]
[1, 100, 3]
```

Help on method_descriptor:

count(self, value, /) Return number of occurrences of value.

In [33]:

```
list1=[20,30, 20,40,20,90]
cnt=list1.count(90)
print(cnt)
```

1

extend

In [35]:

```
help(list.extend)
list1=[30,40,90,100]
list2=["pyhton", "c","java"]
list2.extend(list1)
print(list1)
print(list2)
```

```
Help on method_descriptor:

extend(self, iterable, /)
    Extend list by appending elements from the iterable.

[30, 40, 90, 100]
['pyhton', 'c', 'java', 30, 40, 90, 100]
```

index index(self, value, start=0, stop=9223372036854775807, /) Return first index of value.

```
    Raises ValueError if the value is not present.
```

In [37]:

```
list1=[30,40,30,90,20,20,20]
ind=list1.index(20)
print(ind)
```

4

insert insert(self, index, object, /) Insert object before index.

In [41]:

```
list1=[1,2,3,4,5,6]
list1.insert(4)
print(list1)
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
<ipython-input-41-b94831328a9f> in <module>
      1 list1=[1,2,3,4,5,6]
----> 2 list1.insert(4)
      3 print(list1)

TypeError: insert() takes exactly 2 arguments (1 given)
```

pop pop(self, index=-1, /) Remove and return item at index (default last).

    Raises IndexError if list is empty or index is out of range.

In [44]:

```
list1=[30,40,50,60,70,1000]
list1.pop(3)
print(list1)
```

```
[30, 40, 50, 70, 1000]
```

remove remove(self, value, /) Remove first occurrence of value.

    Raises ValueError if the value is not present.

In [46]:

```
list1=[2,3,4,5,6,7,2,3,4,2,2]
list1.remove(4)
print(list1)
```

```
[2, 3, 5, 6, 7, 2, 3, 4, 2, 2]
```

reverse reverse(self, /) Reverse *IN PLACE*.

In [47]:

```
list1=[10,20,30,40,50]
list1.reverse()
print(list1)
# list2=list1.reverse()
# print(list2)
```

```
[50, 40, 30, 20, 10]
```

sort sort(self, /, , *key=None, reverse=False) Stable sort* IN PLACE*.

In [51]:

```
list1=["a","x","b"]
list1.sort()  #default reverse=False
print(list1)
```

```
['a', 'b', 'x']
```

In [41]:

```
#How to read list from keyboard

l=int(input("enter the length of list"))
print("enter the %d elements"%l)
list1=[]
for i in range(l):
    list1.append(int(input()))
print(list1)
```

```
enter the length of list5
enter the 5 elements
1
2
3
4
5
[1, 2, 3, 4, 5]
```

In [ ]:

```
l=input("enter the list").split(" ")
l=list(map(int,l))
print(l)
```

In [45]:

```
#List compreshension
l=[int(ele) for ele in input().split(" ")]
print(l)
```

```
1 2 3 4 5
[1, 2, 3, 4, 5]
```

In [ ]:

```

```