Built-In functions

- enumerate()
- zip()
- map()
- filter()
- sorted()
- reduce()

enumerate()

- An enumerator built-in-function ***adds a counter of iterable numbers*** to the provided data structure of integers, characters or strings and many more.
- The data structure might be any list, tuple, dictionary or sets.
- If the counter is not provided by the user, then it starts from 0 by default.
- Based on the number provided the enumerator function iterates.
- **Syntax: enumerate(iterable, start)**
- The return type of an enumerate function is an ***object*** type.
- So the enumerate function returns an object by adding the iterating counter value to it. You can also convert the enumerator object into a list(), tuple(), set() and many more.

zip() built-in function

- zip() : function take iterables (can be zero or more), makes iterator that aggregates elements based on the iterables passed, and returns an iterator of tuples.

zip(*iterables)

- The zip() function returns an iterator of tuples based on the iterable object.

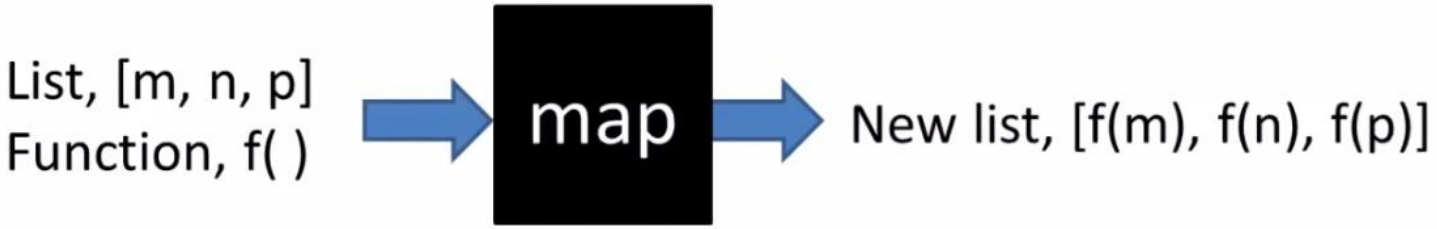
```
name = ["Akshay", "Dravid", "Sachin"]
roll_no = [10, 20, 30]
marks = [90, 88, 75]

mapped = zip(name, roll_no, marks)

print(list(mapped))
```

```
1  name = ["Akshay", "Dravid", "Sachin"]
2  roll_no = [10, 20, 30]
3  marks = [90, 88, 75]
4
5  mapped = zip(name, roll_no, marks)
6
7  print(list(mapped))
```

```
[('Akshay', 10, 90), ('Dravid', 20, 88), ('Sachin', 30, 75)]
```

map() built in function

- **map(fun, iter, ...)** function applies a given function to each element of an iterable.
- ***fun :*** *It is a function to which map passes each element of given iterable.* ***iter :*** *It is a iterable which is to be mapped.*
- The returned value from map() (map object) then can be passed to functions like list(), set().




```
nums = [1, 2, 3, 4, 5]
def sq(n):
    return n*n

square = list(map(sq, nums))
print(square)
```

```
1  nums = [1, 2, 3, 4, 5]
2
3  def sq(n):
4      return n*n
5
6  square = list(map(sq, nums))
7
8  print(square)
```

```
[1, 4, 9, 16, 25]
```

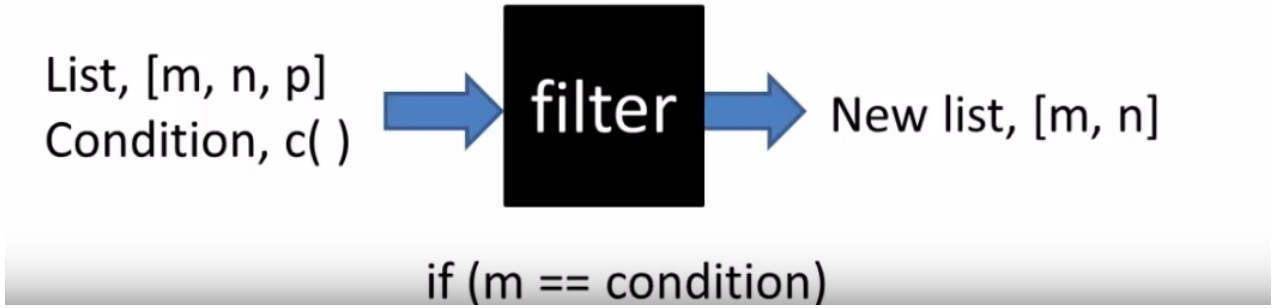
square of all elements in list

filter() built-in function

- filter() function filters the given iterable with the help of a function that tests each element in the iterable to be true or not.

filter(fun, Iter)

- **fun:** function that tests if each element of a sequence true or not.
- **Iter:** Iterable which needs to be filtered.



filter() built-in function

- **Function to filter out vowels from list**

```
alphabets = ['a', 'b', 'd', 'e', 'i', 'j', 'o']

def filterVowels(alphabet):
    vowels = ['a', 'e', 'i', 'o', 'u']

    if(alphabet in vowels):
        return True
    else:
        return False

filteredVowels = filter(filterVowels, alphabets)

print('The filtered vowels are:')
for vowel in filteredVowels:
    print(vowel,end=" ")
```

```
1  alphabets = ['a', 'b', 'd', 'e', 'i', 'j', 'o']
2
3  def filterVowels(alphabet):
4      vowels = ['a', 'e', 'i', 'o', 'u']
5
6      if(alphabet in vowels):
7          return True
8      else:
9          return False
10
11 filteredVowels = filter(filterVowels, alphabets)
12
13 print('The filtered vowels are:')
14 for vowel in filteredVowels:
15     print(vowel,end=" ")
```

```
The filtered vowels are:
a e i o
```

filter()

- It takes a function and applies it to each item in the list to create a new list with only those items that cause the function to return True.

```
def checkAge(age):
    if age > 18:
        return True
    else:
        return False

lst = [10,14,18,22,24]
adults = filter(checkAge, lst)
print(list(adults))
```

```
age = [10,14,18,22,24]

adults = filter(lambda x: x > 18, age)

print(list(adults))
```

```
1  age = [10,14,18,22,24]
2  adults = filter(lambda x: x > 18, age)
3  print(list(adults))
```

```
[22, 24]
```

using built-in function sorted()

names = ['Guido van Rossum', 'Bjarne Stroustrup' , 'James Gosling']

print(sorted(names, key= lambda name: name.split()[-1]))

In [ ]:
```
a = [1,4,3,2,8]
print(sorted(a))

print(sorted(a, reverse=True))
```

In [ ]:
```
c = ['a','z','y','b']
print(sorted(c, reverse=True))
```

using built-in function reduce()

- The reduce(fun,seq) function is used to apply a particular function passed in its argument to all of the list elements mentioned in the sequence.
- This function is defined in “functools” module.

from functools import reduce

reduce(lambda x,y: x+y, [1,2,3,4])

```
1  from functools import reduce
2  reduce(lambda x,y: x+y, [1,2,3,4])
```

```
10
```

using built-in functions max(), min()

studmarks = [('ABC', 35), ('CDE', 25), ('XYZ', 30),('PQR', 20),]

maxlst = max(studmarks, key=lambda student: student[1])

minlst = min(studmarks, key=lambda student: student[1])

print(maxlst)

print(minlst)

```
1  studmarks = [('ABC', 35), ('CDE', 25), ('XYZ', 30),('PQR', 20), ]
2  maxlst = max(studmarks, key=lambda student: student[1])
3  minlst = min(studmarks, key=lambda student: student[1])
4  print(maxlst)
5  print(minlst)
```

```
('ABC', 35)
('PQR', 20)
```