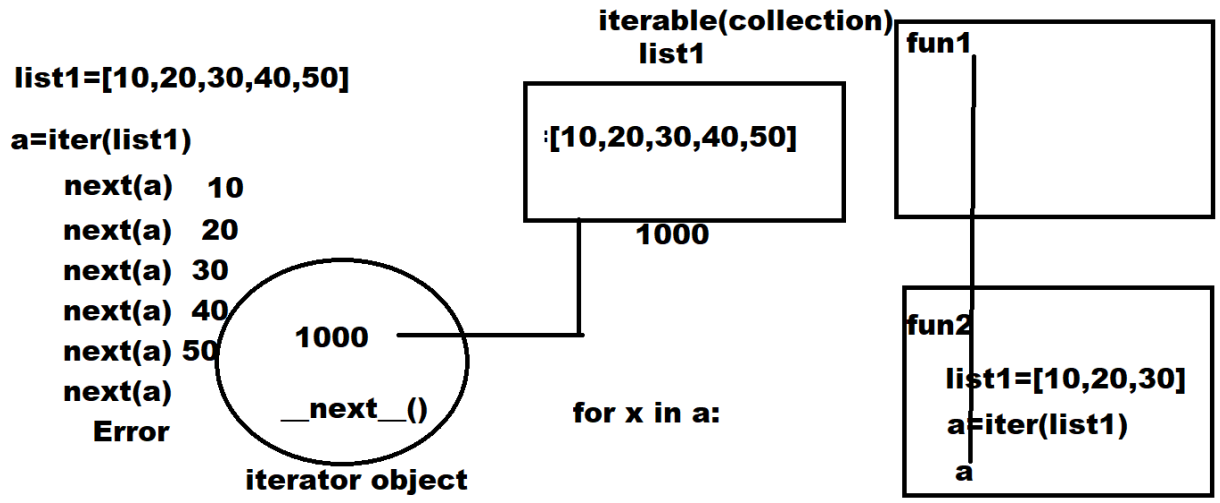
**Iterator**

Iterator is used to read items/values from collection/iterables.
Iterator it read value in forward direction.
Iterator is an immutable, it used to read but not to write.

Iter(iterable) : This function create iterator object on given iterable or collection.




Iterator is useful for non index based collection for reading data.

```
>>> A=[10,20,30,40,50]
>>> A[0]
10
>>> A[1]
20
>>> B={10,20,30,40,50}
>>> B[0]
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    B[0]
TypeError: 'set' object is not subscriptable
>>> a=iter(B)
>>> next(a)
```

```
50
>>> b=iter(A)
>>> next(b)
10
>>> next(b)
20
>>> next(b)
30
>>> next(b)
40
>>> next(b)
50
>>> next(b)
Traceback (most recent call last):
  File "<pyshell#13>", line 1, in <module>
    next(b)
StopIteration
>>> list1=[10,20,30,40,50,60,70,80,90,100]
>>> c=iter(list1)
>>> for x in c:
...     print(x)
...
...
10
20
30
40
50
60
70
80
90
100
>>> c[0]=100
Traceback (most recent call last):
  File "<pyshell#19>", line 1, in <module>
    c[0]=100
```

TypeError: 'list_iterator' object does not support item assignment
>>> list1[0]=100

| list1=[10,20,30,40,50]<br><br># Using index<br>print(list1[0],list1[1],list1[2],list1[3],list1[4])<br><br># using for loop<br>for value in list1:<br>  print(value)<br><br># using iterator<br>a=iter(list1)<br>for value in a:<br>  print(value) | Output<br>10 20 30 40 50<br>10<br>20<br>30<br>40<br>50<br>10<br>20<br>30<br>40<br>50 |
|---|---|

**Enumerate**

Enumerate also iterator object but this returns two values

1. Start
2. Value

Syntax: enumerate(iterable,start=0)

```
>>> names=["naresh","suresh","ramesh","kishore"]
>>> print(names)
['naresh', 'suresh', 'ramesh', 'kishore']
>>> e=enumerate(names,start=1)
>>> for x in e:
...     print(x)
...
...
(1, 'naresh')
(2, 'suresh')
(3, 'ramesh')
(4, 'kishore')
>>> sales=[10000,20000,30000,40000,50000]
>>> print(sales)
```

```
[10000, 20000, 30000, 40000, 50000]
>>> e=enumerate(sales,start=2000)
>>> for x in e:
...     print(x)
...
...
(2000, 10000)
(2001, 20000)
(2002, 30000)
(2003, 40000)
(2004, 50000)
>>>
```

**Mutable operations of list**

After creating list changes can be done using mutable operations.

**Replacing or Update value**

Replacing values in list can be done in 2 ways

1. Index
2. Slicing

Using index one value can be replaced
Using slicing more than one value can be replaced

Syntax: list-name[index]=value
Syntax: list-name[start:stop:step]=iterable

**Example:**

```
>>> list1=[10,20,30,40,50,60,70,80,90,100]
>>> print(list1)
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
>>> list1[0]=1
>>> print(list1)
[1, 20, 30, 40, 50, 60, 70, 80, 90, 100]
>>> list1[-1]=99
```

```
>>> print(list1)
[1, 20, 30, 40, 50, 60, 70, 80, 90, 99]
>>> list1[4]=55
>>> print(list1)
[1, 20, 30, 40, 55, 60, 70, 80, 90, 99]
>>> list1[10]=66
Traceback (most recent call last):
  File "<pyshell#41>", line 1, in <module>
    list1[10]=66
IndexError: list assignment index out of range
>>> list1[-11]=100
Traceback (most recent call last):
  File "<pyshell#42>", line 1, in <module>
    list1[-11]=100
IndexError: list assignment index out of range
```

**Example**

'''Given a list, write a Python program to swap first and last element of the list.

Examples:

Input : [12, 35, 9, 56, 24] Output : [24, 35, 9, 56, 12]
Input : [1, 2, 3] Output : [3, 2, 1]'''

```
list1=[12, 35, 9, 56, 24]

print(f'Before Swaping {list1}')
# Method-1
temp=list1[0]
list1[0]=list1[-1]
list1[-1]=temp
print(f'After Swaping {list1}')

#Method-2
list1[0],list1[-1]=list1[-1],list1[0]
```

```
print(f'After Swaping {list1}')
```

**Output**

```
Before Swaping [12, 35, 9, 56, 24]
After Swaping [24, 35, 9, 56, 12]
After Swaping [12, 35, 9, 56, 24]
```

**Example**

```
'''Given a list in Python and provided the positions of the elements,
write a program to swap the two elements in the list.

Examples:

Input : List = [23, 65, 19, 90], pos1 = 1, pos2 = 3
Output : [19, 65, 23, 90]

Input : List = [1, 2, 3, 4, 5], pos1 = 2, pos2 = 5
Output : [1, 5, 3, 4, 2]'''


list1=[23, 65, 19, 90]
pos1=1
pos2=3

print(f'Before swaping {list1}')
list1[pos1-1],list1[pos2-1]=list1[pos2-1],list1[pos1-1]
print(f'After Swaping {list1}')
```

**Output**

```
Before swaping [23, 65, 19, 90]
After Swaping [19, 65, 23, 90]
```

**Example of replacing more than one value using slicing**

```
>>> list1=[10,20,30,40,50,60,70,80,90,100]
>>> print(list1)
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

```
>>> list1[0:2]=[11,22]
>>> print(list1)
[11, 22, 30, 40, 50, 60, 70, 80, 90, 100]
>>> list1[-2:]=[99,111]
>>> print(list1)
[11, 22, 30, 40, 50, 60, 70, 80, 99, 111]
>>> list1[2:-2]=[33,44,55,66,77,88]
>>> print(list1)
[11, 22, 33, 44, 55, 66, 77, 88, 99, 111]
>>> list1[::2]=[1,2,3,4,5]
>>> print(list1)
[1, 22, 2, 44, 3, 66, 4, 88, 5, 111]
>>> list1[::-2]=[6,7,8,9,10]
>>> print(list1)
[1, 10, 2, 9, 3, 8, 4, 7, 5, 6]
```

**Example of Packing and Unpacking**

```
>>> list1=[10,20,30,40,50]
>>> a,b,c,d,e=list1
>>> print(a,b,c,d,e)
10 20 30 40 50
>>> list2=[10,20,30,40,50,60,70]
a,b=list2
Traceback (most recent call last):
  File "<pyshell#59>", line 1, in <module>
    a,b=list2
ValueError: too many values to unpack (expected 2)
>>> a,b,*c=list2
>>> print(a,b,c)
10 20 [30, 40, 50, 60, 70]
>>> x=list2
>>> print(x)
[10, 20, 30, 40, 50, 60, 70]
>>> l1=[1,2,3,4,5]
>>> l2=[6,7,8,9,10]
>>> l3=[*l1,*l2]
>>> print(l1)
```

```
[1, 2, 3, 4, 5]
>>> print(l2)
[6, 7, 8, 9, 10]
p
>>> print(l3)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```