Sequence type
In sequence type the data is organized in memory sequential order.
Sequence types are index based collections

List
List is a sequence data type, where data is organized in sequential order.
List is a mutable sequence. After creating list we can do changes (adding, updating and deleting).
List is ordered collection, where insertion order is preserved.
Lists are mutable sequences, typically used to store collections of homogeneous items (It also used for storing different types of objects)
List is collection of objects.
List allows duplicates
List is index based collection

How to create list?
Lists may be constructed in several ways:

- Using a pair of square brackets to denote the empty list: []
- Using square brackets, separating items with commas: [a], [a, b, c]
- Using a list comprehension: [x for x in iterable]
- Using the type constructor: list() or list(iterable)

Using a pair of square brackets to denote the empty list: []

```
>>> list1=[]
>>> print(type(list1))
<class 'list'>
>>> print(list1)
[]
>>>
```

Using square brackets, separating items with commas: [a], [a, b, c]

```
>>> list2=[10]
>>> list3=[10,20,30,40,50]
>>> list4=[101,"naresh",4000.0]
>>> print(list2,list3,list4,sep="\n")
[10]
[10, 20, 30, 40, 50]
[101, 'naresh', 4000.0]
```

```
>>>
```

Using the type constructor: list() or list(iterable)

```
>>> list5=list()
>>> print(list5)
[]
>>> list6=list(range(10,60,10))
>>> print(list6)
[10, 20, 30, 40, 50]
>>> list7=list(list6)
>>> print(list7)
[10, 20, 30, 40, 50]
>>> list8=list("PYTHON")
>>> print(list8)
['P', 'Y', 'T', 'H', 'O', 'N']
>>>
>>> list9=[10,10,10,20,30,40,20,30]
>>> print(list9)
[10, 10, 10, 20, 30, 40, 20, 30]
>>>
```

When application required store more than one object where duplicates are allowed we use list

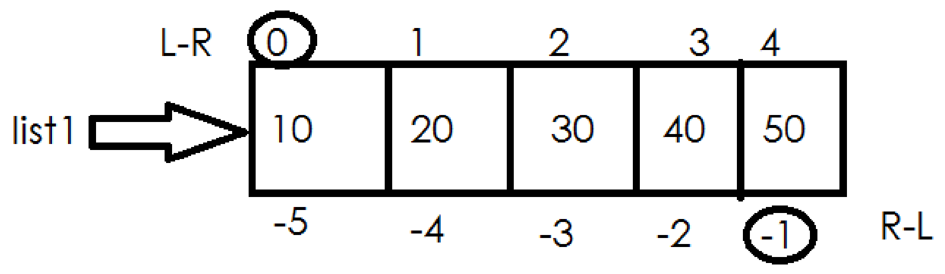
How to read elements/items from list?

1. Index
2. Slicing
3. Iterator
4. Enumerate
5. For loop

Index
What is index?
Index is an integer value which is used to identify each location or value in list/sequence. This index can be +ve or –ve
+VE index starts at 0, which is used to read elements in forward direction (Left to Right)
-VE index starts at -1, which is used to read elements in backward direction (Right to Left)

Index is used to read only one value/element from list
It allows reading in sequential or random order

list1=[10,20,30,40,50]



list1[0] ==> 10	list1[-1] ==> 50	list1[6] ==> IndexError
list1[1] ==> 20	list1[-2] ==> 40	list[-6] ==> IndexError
list1[2] ==> 30	list1[-3] ==> 30	
list1[3] ==> 40	list1[-4] ==> 20	
list1[4] ==> 50	list1[-5] ==> 10	

Example reading elements from list using index

```
list1=[10,20,30,40,50,60,70,80,90,100]
# reading elements from list using +ve index form L-R
for i in range(len(list1)): # 0 1 2 3 4 5 6 7 8 9
    print(list1[i],end=' ')

print()
# reading elements from list using -ve index from R-L
for i in range(-1,-(len(list1)+1),-1): # -1 -2 -3 -4 ... -10
    print(list1[i],end=' ')

print()
# reading elements from list using +ve index from R-L
for i in range(len(list1)-1,-1,-1):
    print(list1[i],end=' ')

print()
# reading elements from list using -ve index from L-R
```

```
for i in range(-len(list1),0):
    print(list1[i],end=' ')
```

Output:

```
10 20 30 40 50 60 70 80 90 100
100 90 80 70 60 50 40 30 20 10
100 90 80 70 60 50 40 30 20 10
10 20 30 40 50 60 70 80 90 100
>>>
```

Slicing

Slicing is process of reading more than one value/element from list
Slicing return another list by copying all the elements or items
Slicing is done using two approaches

1. Slice operator
2. Slice object

Slicing required the following values

1. Start index
2. Stop index
3. Step

Slicing uses internally range for generating indexes

Slice operator

Syntax1: list-name[start:stop:step]
Syntax2: list-name[start::]
Syntax-3: list-name[:stop:]
Syntax4: list-name[::step]
Syntax5:list-name[start:stop]
Syntax6: list-name[start::step]
Syntax7: list-name[:stop:step]
Syntax8: list-name[::]
Syntax9: list-name[:]