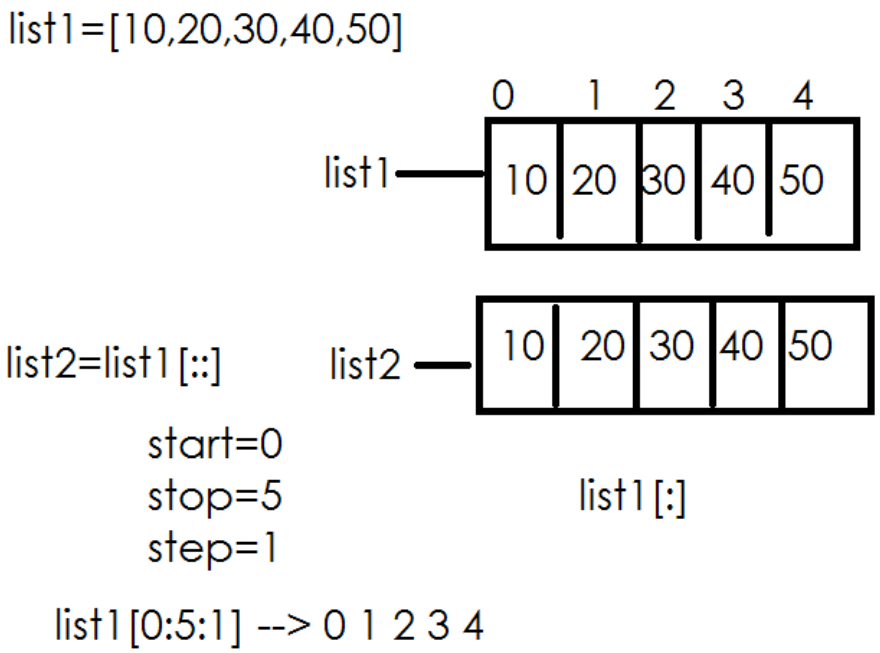
Slicing
Slicing is a process of reading more than one element/value from sequence.
Slicing returns a sequence.

What is difference between indexing and slicing?
Slicing in Python refers to extracting a subset or specific part of the sequence list, tuple, or string in a specific range. While indexing refers to accessing a single element from a sequence, it is used to get slices of sequence.

Syntax of Index : sequence-name[index] → One Value
Syntax of slicing : sequence-name[start:stop:step]

Stop index is excluded, start index is included.
Start,stop and step values are optional.

Syntax1: sequence-name[::] (OR) sequence-name[:]
Start:0,stop:length of sequence,step:+1
This syntax creates copy of sequence.

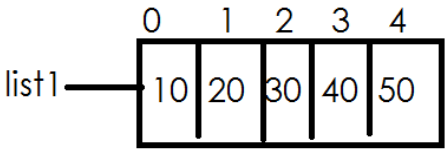


Example:

```
>>> list1=[10,20,30,40,50]
>>> list2=list1[::]
>>> print(list1)
[10, 20, 30, 40, 50]
>>> print(list2)
[10, 20, 30, 40, 50]
>>> list3=list1[:]
>>> print(list3)
```

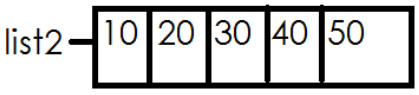
Syntax2: sequence-name[::step]
if step value is +ve start=0,stop=length of sequence
if step value is –ve start=-1,stop=-(length of sequence+1)

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



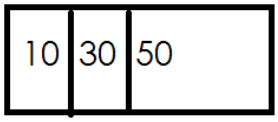
list2=list1[::1]

start=0,stop=5,step=1 --> 0 1 2 3 4



list3=list1[::2]

start=0,stop=5,step=2 0 2 4

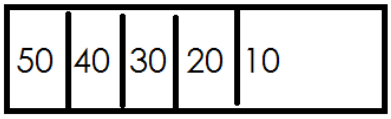


list4=list1[::-1]

start=-1,stop=-6,step=-1

-1 -2 -3 -4 -5

list5=list1[::-2]



-1 -3 -5

Example:

```
>>> list1=list(range(10,60,10))
```

```
>>> print(list1)
[10, 20, 30, 40, 50]
>>> list2=list1[::1]
>>> print(list2)
[10, 20, 30, 40, 50]
>>> list3=list1[::2]
>>> print(list3)
[10, 30, 50]
>>> list4=list1[::-1]
>>> print(list4)
[50, 40, 30, 20, 10]
>>> list5=list1[::-2]
>>> print(list5)
[50, 30, 10]
```

Syntax3: sequence-name[:stop:]
Default start=0,step=1, the value of stop can be +ve or –ve. This syntax read values from left to right.

```
>>> list1=list(range(100,110))
>>> print(list1)
[100, 101, 102, 103, 104, 105, 106, 107, 108, 109]
>>> list2=list1[:5:]
>>> print(list2)
[100, 101, 102, 103, 104]
>>> list3=list1[:-5:]
>>> print(list3)
[100, 101, 102, 103, 104]
>>> list4=list1[:-2:]
>>> print(list4)
```

Syntax4: sequence-name[start::] or seqnece-name[start:]
Default stop=end of list,step=1, this syntax read elements from left to right.

```
>>> list1=list(range(1,21,2))
>>> print(list1)
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
>>> list2=list1[5:]
>>> print(list2)
[11, 13, 15, 17, 19]
```

```
>>> list3=list1[-3:]
>>> print(list3)
[15, 17, 19]
```

Syntax5: sequence-name[start:stop]
Default step value +1, This syntax read values from left to right.

```
>>> list1=list(range(2,21,2))
>>> print(list1)
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
>>> list2=list1[2:5]
>>> print(list2)
[6, 8, 10]
>>> list3=list1[-4:-2]
>>> print(list3)
[14, 16]
>>> list4=list1[2:-2]
>>> print(list4)
[6, 8, 10, 12, 14, 16]
```

Syntax-6: sequence-name[start:stop:step]

```
>>> list1=list(range(10,100,10))
>>> print(list1)
[10, 20, 30, 40, 50, 60, 70, 80, 90]
>>> list2=list1[0:8:2]
>>> print(list2)
[10, 30, 50, 70]
>>> list3=list1[-2:-8:-1]
>>> print(list3)
[80, 70, 60, 50, 40, 30]
>>> list3=list1[-2:-1:-1]
>>> print(list3)
[]
```

for loop
for loop is an iterator, which is used to iterate or read values from collections or iterables.

Syntax:

```
for variable-name in iterable:
```

```
    statement-1
    statement-2
```

for loop each time read value from iterable and execute statement-1 and statement-2.

Example:

```
list1=[1,2,-1,-2,-3,-4,0,3,4,5,0,0,-5,-6,-7]
pc=0
nc=0
zc=0
for num in list1:
    if num>0:
        pc+=1
    elif num<0:
        nc+=1
    else:
        zc+=1

print(f'+ve number count {pc}')
print(f'-ve number count {nc}')
print(f'zero count {zc}')
```

Output:

```
+ve number count 5
-ve number count 7
zero count 3
```

Example:

```
salesList=[1000,2000,3000,2000,4000,5000,6000,7000,8000]
tot=0
for amt in salesList:
    tot=tot+amt

print(f'SalesList {salesList}')
print(f'Total Sales {tot}')
```

Output:

```
SalesList [1000, 2000, 3000, 2000, 4000, 5000, 6000, 7000, 8000]
```

Total Sales 38000

Iterator