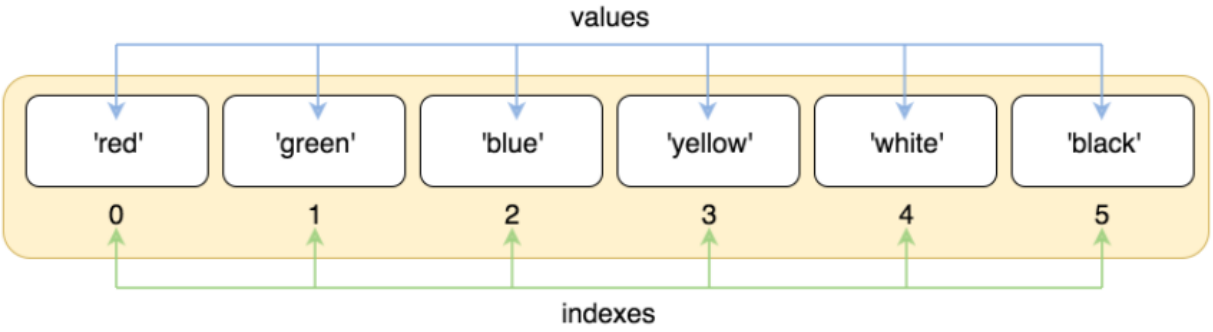
Indexing:

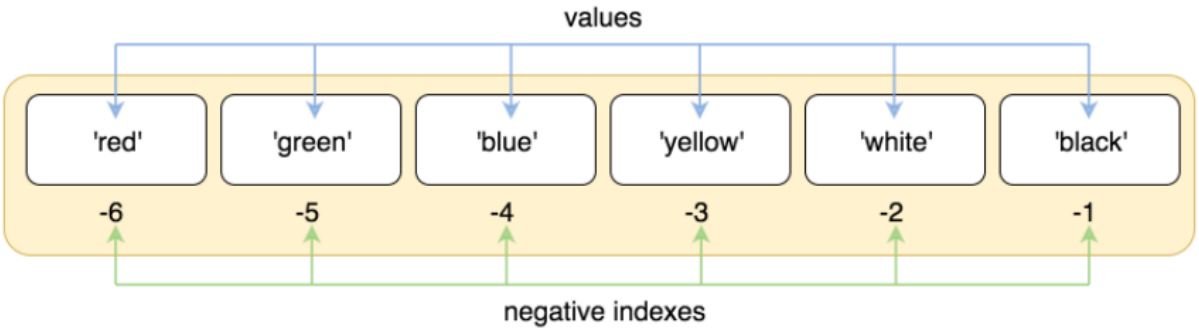
Let's take a simple example:

```
colors = ['red', 'green', 'blue', 'yellow', 'white', 'black']
```

Python uses zero-based indexing. That means, the first element(value 'red') has an index 0, the second(value 'green') has index 1, and so on.

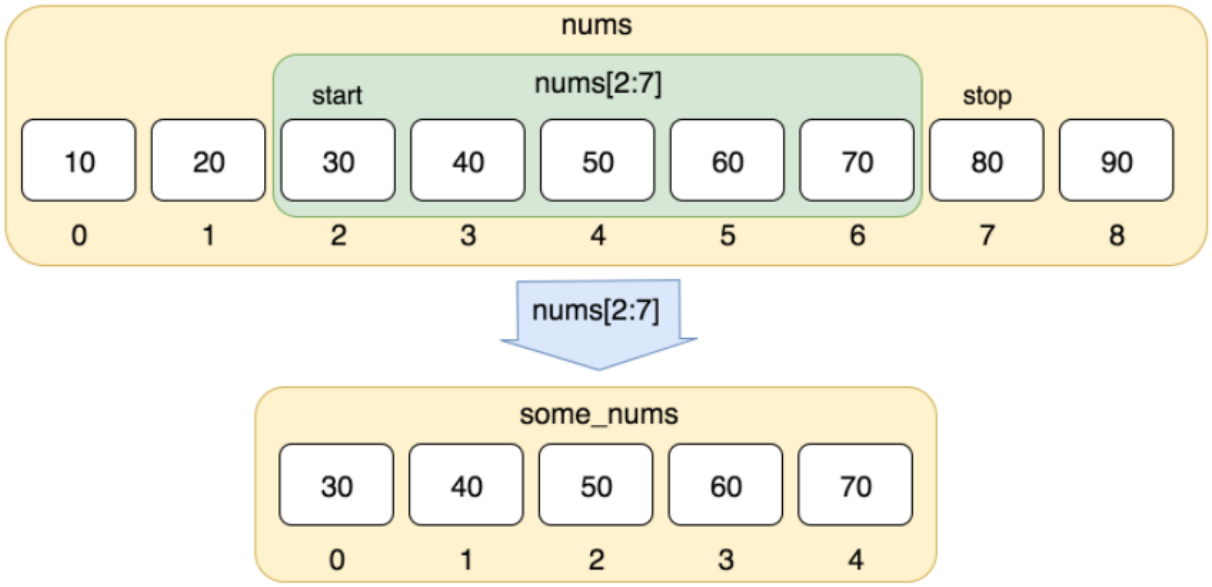


To address this requirement there is negative indexing. So, instead of using indexes from zero and above, we can use indexes from -1 and below.



In negative indexing system -1 corresponds to the last element of the list(value 'black'), -2 to the penultimate (value 'white'), and so on.

Slicing:




```
[start : stop : steps]

which means that slicing will start from index start
 will go up to stop in step of steps.
 Default value of start is 0, stop is last index of list
 and for step it is 1
```

append() and extend() in Python

Last Updated: 01-04-2020

Append: Adds its argument as a single element to the end of a list. The length of the list increases by one.

```
syntax:
# Adds an object (a number, a string or a
# another list) at the end of my_list
my_list.append(object)
```

```
my_list = ['geeks', 'for']
my_list.append('geeks')
print my_list
```

Output:

```
['geeks', 'for', 'geeks']
```

NOTE: A list is an object. If you append another list onto a list, the parameter list will be a single object at the end of the list.

```
my_list = ['geeks', 'for', 'geeks']
another_list = [6, 0, 4, 1]
my_list.append(another_list)
print my_list
```

Output:

```
['geeks', 'for', 'geeks', [6, 0, 4, 1]]
```

extend(): Iterates over its argument and adding each element to the list and extending the list. The length of the list increases by number of elements in it's argument.

```
syntax:
# Each element of an iterable gets appended
# to my_list
my_list.extend(iterable)
```

```
my_list = ['geeks', 'for']
another_list = [6, 0, 4, 1]
my_list.extend(another_list)
print my_list
```

Output:

```
['geeks', 'for', 6, 0, 4, 1]
```

NOTE: A string is an iterable, so if you extend a list with a string, you'll append each character as you iterate over the string.

```
my_list = ['geeks', 'for', 6, 0, 4, 1]
my_list.extend('geeks')
print my_list
```

Output:

```
['geeks', 'for', 6, 0, 4, 1, 'g', 'e', 'e', 'k', 's']
```

Tuple

Tuples are used to store multiple items in a single variable.

Tuple is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Set, and Dictionary, all with different qualities and usage.

A tuple is a collection which is ordered and **unchangeable**.

Tuples are written with round brackets.

Example

Create a Tuple:

```
thistuple = ("apple", "banana", "cherry")
print(thistuple)
```

1. Predict the output:

```
x = "Welcome Bennett"
print(x[3])
print(x[1:4])
print(x[3: ])
print(x[ :4])
print(x[1 : -2])
print(x[-3 : ])
print(x[ : -2])
print(x[1:6:2])
```

Sol.

```
c
elc
come Bennett
Welc
elcome Benne
ett
Welcome Benne
ecm
```

2. Accessing Values in Lists, predict the output?

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5, 6, 7];
print ("list1[0]: ", list1[0])
print ("list2[1:5]: ", list2[1:5])
```

Sol.

```
list1[0]:  physics
list2[1:5]:  [2, 3, 4, 5]
```

3. What is the difference between append and extend?

.append() adds an object to the end of a list
.extend() adds each value from a 2nd list as its own element. So extending a list with another list combines their values.

Sol.

```
a = [1,2,3]
a.append(4)
print(a)

a.append([5,6])
print(a)
```

```
b = [1,2,3]
b.extend([5, 6])
print(b)
```

4. What is the difference between "remove" and "pop"?

`.remove()` removes the first instance of a matching object.

Example:

```
a = ['a', 'a', 'b', 'b', 'c', 'c']
 a.remove('b')
print(a)
```

predict the output?

Sol.

```
['a', 'a', 'b', 'c', 'c']
```

`.pop()` removes an object by its index.

```
a = ['a', 'a', 'b', 'b', 'c', 'c']
a.pop(4)
```

predict the output?

Sol.

```
'c'
```

5. How to remove duplicates from a list?

Input: `[3, 2, 2, 1, 1, 1]`

Output: `[1, 2, 3]`

Sol.

```
li = [3, 2, 2, 1, 1, 1]
print(list(set(li)))
```

6. How to manipulate every element in a list with list comprehension?

E.g., *Return a new list with 1 added to every element.*

Input: `[0,25,50,100]`

Output: `[1, 26, 51, 101]`

Sol.

```
li = [0,25,50,100]
print(li)
li1 = [i+1 for i in li]
```

```
print(li1)
```

7. How to count the occurrence of a specific object in a list?

Input: `['dog','cat','fish','fish','cat']`

Output: 2

Sol.

```
pets = ['dog','cat','fish','fish','cat']
print(pets.count('fish'))
```

8. Sort the list

L= [1,3,5,2,7,8]

print(sorted(L))

9. `round1 = ['Welcome', 'to', 'Greater Noida', 'Sec-Mu1']`

Create a copy of round1, assign it to a new name, round2, and then remove the string `Sec-Mu1`.

Sol.

```
round1 = ['Welcome', 'to', 'Greater Noida', 'Sec-Mu1']
round2 = round1.copy()
round2.remove('Sec-Mu1')
print(round1)
print(round2)
```

10. What will be the of output of the following code?

```
A = [1, 60, 12]
n = len(A)
for i in range(1,n,1):
    j = i-1
    while j > 0:
       A[j+1] = A[j]
       j = j-1
 print(A)
```

Sol.

```
[1, 60, 60]
```

11. Write a Python program to create a tuple with different data types

```
tuplex = ("tuple", False, 3.2, 1)

print(tuplex)
```

Sol.

```
('tuple', False, 3.2, 1)
```

12. What will be the of output of the following code?

```
A = (1, 60, 12)
n = len(A)
for i in range(1,n,1):
      for j in range(1,n,1):
            A[i] = A[j]
print(A)
```

Sol.

```
TypeError: 'tuple' object does not support item assignment
```

13. What will be the output of the following?

```
x = (1,2,3,4)
y = (5,6,7,8)

z = x + y
print(z)
```

Sol.

```
(1, 2, 3, 4, 5, 6, 7, 8)
```

14. Predict the output: (difference between append, insert and extend)

```
List = [1,2,3,4]
List.append(12)
print (List)
List.insert(3, 12)
print (List)
List.extend([ 'Bennett', 'University'])
print (List)
```

Sol.

```
[1, 2, 3, 4, 12]
[1, 2, 3, 12, 4, 12]
[1, 2, 3, 12, 4, 12, 'Bennett', 'University']
```

15. Predict The Output:

```
data = [x for x in range(5)]

temp = [x for x in range(7) if x in data and x%2==0]

print(temp)
```

Sol.

```
[0, 2, 4]
```

16. Predict the output (difference between remove and pop)

```
List = [1, 2, 3, 4, 5, 6, 7, 8, 7, 10, 11, 12]
List.remove(7)
print("list-1",List)
for i in range(1, 5):
    List.remove(i)
print("list-2",List)
List = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
List.remove(7)
print("list-3",List)
List.pop(7)
print("list-4",List)
```

Sol.

```
list-1 [1, 2, 3, 4, 5, 6, 8, 7, 10, 11, 12]
list-2 [5, 6, 8, 7, 10, 11, 12]
list-3 [1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12]
list-4 [1, 2, 3, 4, 5, 6, 8, 10, 11, 12]
```

17. Predict the output (difference between remove and pop)

```
List = [1, 2, 3, 4, 5, 6, 7, 8, 7, 10, 11, 12]

List.remove(7)

print(List)

for i in range(1, 5):

List.remove(i)

print(List)

List = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

List.remove(7)

print(List)

List.pop(7)

print(List)
```

18. Write a Python program to remove duplicates from a list.

Input:

a = [10,20,30,20,10,50,60,40,80,50,40]

Output:

a = [10,20,30,20,10,50,60,40,80,50,40]

Sol:

a = [10,20,30,20,10,50,60,40,80,50,40]

```
dup_items = set()
uniq_items = []
for x in a:
    if x not in dup_items:
        uniq_items.append(x)
        dup_items.add(x)

print(dup_items)
```

19. Write a program to check if elements of a list are same or not it read from front or back.

E.g.- [2 3 15 15 3 2]

for the above list, print yes output.

Take 6 integer as input in the List from the users and check if elements of a list are same or not it read from front or back

Sol:

```
#L = [2, 3, 15, 15, 3, 2]

i = 6
L = []
while i>0:
  num = eval(input())
  L.append(num)
  i = i-1
  i = 0
length = len(L)
mid = length/2
same = True
while i<mid:
    if L[i] != L[length- 1 - i]:
       print("No")
       break
    i = i+1
if same == True:
    print("yes")
```

20. I have students grades in a List. As a B.Tech student, help me to check if the given grades is in ascending order or Not?

Sol.

Note: Do not use inbuilt function, and create your own function.

Sol:

```
list1 = [33, 47, 32, 50, 49, 80, 17, 90]
temp_list = list1[:]
list1.sort()
if temp_list == list1:
    print("Given List is in Ascending Order")
else:
    print("Given List is not in Ascending Order")


output: Given List is not in Ascending Order
```

21. Take 10 integer inputs from user and store them in a list. Now, copy all the elements in another list but in reverse order. Print original and reversed list both

Sol

```
i = 10
L = []

while i>0:
  num = eval(input())
  L.append(num)
  i = i-1


# reverse
  L.reverse()

#Copy to a new list
  L2 = L.copy()

#again back to initial order
  L.reverse()

print("original list", L)
print("reverse list", L2)
```