باب-9

لسٹ (LISTS)

5196CH09

''کوڈ کی سطروں کے ذریعے پروگرامنگ کی پیش رفت کی پیمائش طیارہ سازی کی پیش رفت کی پیمائش اس کے وزن سے کرنے کے مترادف ہے۔''

بل گیٹس

(Bill Gates)

9.1 لسٹ کا تعارف

ڈیٹا ٹائپ لسٹ ایک مرتب تسلسل ہے جو قابل تغیر ہے اور یہ ایک یا ایک سے زیادہ عناصر پر مشتمل ہوتی ہے۔ اسٹرنگ (جو صرف کیریکٹر پر مشتمل ہوتی ہے) کے برعکس لسٹ صحیح عدد، فلوٹ جیسے مختلف ڈیٹا ٹائپ والے عناصر کے ساتھ ساتھ دیگر لسٹ پر مشتمل ہوتی ہے۔ لسٹ مخلوط ڈیٹا ٹائپ والے عناصر کی ایک ساتھ گروپ بندی کے لیے بہت مفید ہے۔ لسٹ کے عناصر کو بڑے بریکٹ میں لکھا جاتا ہے اور یہ عناصر کوما کے ذریعے ایک دوسرے سے علاحدہ رہتے ہیں۔ اسٹرنگ انڈیکس کی طرح لسٹ انڈیکس کی ابتدا بھی 0 سے ہوتی ہے۔

مثال 9.1

```
#list1 is the list of six even numbers
>>> list1 = [2,4,6,8,10,12]
>>> print(list1)
[2, 4, 6, 8, 10, 12]

#list2 is the list of vowels
>>> list2 = ['a','e','i','o','u']
>>> print(list2)
['a', 'e', 'i', 'o', 'u']

#list3 is the list of mixed data types
>>> list3 = [100,23.5,'Hello']
>>> print(list3)
[100, 23.5, 'Hello']

#list4 is the list of lists called nested
#list
>>> list4 =[['Physics',101],['Chemistry',202],
            ['Maths',303]]
>>> print(list4)
[['Physics', 101], ['Chemistry', 202],
 ['Maths', 303]]
```

9.1.1 لسٹ کے عناصر تک رسائی حاصل کرنا

لسٹ (List) کے عناصر تک رسائی حاصل کرنے کا طریقہ وہی ہے جو ایک اسٹرنگ (String) میں کیریکٹر (حرفی علامات) تک رسائی حاصل کرنے کا ہے۔

اس باب میں

- لسٹ کا تعارف
- لسٹ آپریشن
- لسٹ کا اعادہ
- لسٹ میتھڈ اور بلٹ ان فنکشن
- نیسٹیڈ لسٹ
- لسٹ کی نقل تیار کرنا
- فنکشن کے آرگیومینٹ کے طور پر لسٹ
- لسٹ میں ردوبدل

نوٹ

```
#initializes a list list1
>>> list1 = [2,4,6,8,10,12]
>>> list1[0] #return first element of list1
2
>>> list1[3] #return fourth element of list1
8
#return error as index is out of range
>>> list1[15]
IndexError: list index out of range
#an expression resulting in an integer index
>>> list1[1+4]
12
>>> list1[-1] #return first element from right
12
#length of the list list1 is assigned to n
>>> n = len(list1)
>>> print(n)
6
#return the last element of the list1
>>> list1[n-1]
12
#return the first element of list1
>>> list1[-n]
2
```

9.1.2 لسٹ قابل تغیر ہیں

پائتھن میں لسٹ قابل تغیر ہوتی ہیں۔ اس کا مطلب ہے کہ لسٹ کی تشکیل کے بعد اس کے مواد کو تبدیل کیا جاسکتا ہے۔

```
#List list1 of colors
>>> list1 = ['Red','Green','Blue','Orange']
#change/override the fourth element of list1
>>> list1[3] = 'Black'
>>> list1      #print the modified list list1
['Red', 'Green', 'Blue', 'Black']
```

9.2 لسٹ پر انجام دیے جانے والے عمل (LIST OPERATIONS)

ڈیٹا ٹائپ لسٹ کے مواد میں مختلف عملوں (Operations) کی مدد سے ردوبدل کی جاسکتی ہے جیسا کہ ذیل میں دکھایا گیا ہے۔

9.2.1 کنکٹینیشن (Concatenation)

پائتھن میں ہم + کی علامت سے ظاہر کیے جانے والے کنکٹینیشن آپریٹر کا استعمال کرکے دو یا دو سے زیادہ لسٹ کو یکجا کر سکتے ہیں۔

```
#list1 is list of first five odd integers
```

نوٹ

```
>>> list1 = [1,3,5,7,9]
#list2 is list of first five even integers
>>> list2 = [2,4,6,8,10]
#elements of list1 followed by list2
>>> list1 + list2
[1, 3, 5, 7, 9, 2, 4, 6, 8, 10]
>>> list3 = ['Red','Green','Blue']
>>> list4 = ['Cyan', 'Magenta', 'Yellow'
,'Black']
>>> list3 + list4
['Red','Green','Blue','Cyan','Magenta',
   'Yellow','Black']
```

غور کیجیے کہ رواں لسٹ یعنی list1، list2، list3، list4 کنکٹینیشن عمل کے بعد بھی وہی رہتی ہیں۔ اگر ہم دو لسٹ ایک دوسرے میں ضم کرنا چاہتے ہیں تو ہمیں متحدہ لسٹ کو دیگر لسٹ کے ساتھ منسوب کرنے کے لیے اسائنمینٹ بیان کو استعمال کرنا چاہیے۔ کنکٹینیشن آپریٹر '+' کے لیے زیر عمل مقدار کو صرف لسٹ ٹائپ ہونا چاہیے۔ اگر ہم ایک لسٹ کو کسی دوسرے ڈیٹا ٹائپ والے عناصر پر مشتمل لسٹ میں ضم (Concatenate) کرنا چاہتے ہیں تو TypeError ظاہر ہو جاتی ہے۔

```
>>> list1 = [1,2,3]
>>> str1 = "abc"
>>> list1 + str1
TypeError: can only concatenate list (not
"str") to list
```

9.2.2 تکرار (Repetition)

پائتھن ہمیں علامت * سے ظاہر کیے جانے والے تکراری آپریٹر کا استعمال کر کے لسٹ کو دہرانے میں مدد کرتا ہے۔

```
>>> list1 = ['Hello']
#elements of list1 repeated 4 times
>>> list1 * 4
['Hello', 'Hello', 'Hello', 'Hello']
```

9.2.3 رکنیت (Membership)

اسٹرنگ کی طرح ممبرشپ آپریٹر "in" اس بات کی جانچ کرتا ہے کہ آیا کوئی عنصر لسٹ میں موجود ہے یا نہیں۔ اگر موجود ہے تو نتیجے کے طور پر True ظاہر کر دیتا ہے بصورت دیگر False ظاہر کرتا ہے۔

```
>>> list1 = ['Red','Green','Blue']
>>> 'Green' in list1
True
>>> 'Cyan' in list1
False
```

نوٹ

"not in" آپریٹر کا نتیجہ اس وقت True کی شکل میں ظاہر ہوتا ہے جب لسٹ میں عنصر غیر موجود ہو بصورت دیگر اس کا نتیجہ False کی شکل میں ظاہر ہوتا ہے۔

```
>>> list1 = ['Red','Green','Blue']
>>> 'Cyan' not in list1
True
>>> 'Green' not in list1
False
```

9.2.4 سلائسنگ (Slicing)

اسٹرنگ کی طرح، سلائسنگ آپریشن کا اطلاق لسٹ پر بھی کیا جا سکتا ہے۔

```
>>> list1 =['Red','Green','Blue','Cyan',
'Magenta','Yellow','Black']
>>> list1[2:6]
['Blue', 'Cyan', 'Magenta', 'Yellow']

 #list1 is truncated to the end of the list
>>> list1[2:20] #second index is out of range
['Blue', 'Cyan', 'Magenta', 'Yellow',
'Black']

>>> list1[7:2]     #first index > second index
[]                 #results in an empty list

#return sublist from index 0 to 4
>>> list1[:5]      #first index missing
['Red','Green','Blue','Cyan','Magenta']


#slicing with a given step size
>>> list1[0:6:2]
['Red','Blue','Magenta']

#negative indexes
#elements at index -6,-5,-4,-3 are sliced
>>> list1[-6:-2]
['Green','Blue','Cyan','Magenta']


#both first and last index missing
>>> list1[::2]     #step size 2 on entire list
['Red','Blue','Magenta','Black']

#negative step size
#whole list in the reverse order
>>> list1[::-1]
['Black','Yellow','Magenta','Cyan','Blue',
'Green','Red']
```

9.3 لسٹ کا اعادہ (TRAVERSING A LIST)

ہم For لوپ یا while لوپ کا استعمال کر کے لسٹ کے ہر ایک عنصر کو ایکسس کر سکتے ہیں یا لسٹ کا اعادہ کر سکتے ہیں۔

(A) For لوپ کا استعمال کر کے لسٹ کا اعادہ

```
>>> list1 = ['Red','Green','Blue','Yellow',
             'Black']
>>> for item in list1:
        print(item)
```

نتیجہ:

```
Red
Green
Blue
Yellow
Black
```

لسٹ کے عناصر کو ایکسس کرنے کا دوسرا طریقہ range() اور len() فنکشن کا استعمال ہے۔

```
>>> for i in range(len(list1)):
        print(list1[i])
```

len(list1) لسٹ 1 کے عناصر کی کل تعداد کو بتاتا ہے۔

نتیجہ:

```
Red
Green
Blue
Yellow
Black
```

(B) While لوپ کا استعمال کر کے لسٹ کا اعادہ

```
>>> list1 = ['Red','Green','Blue','Yellow',
             'Black']
>>> i = 0
>>> while i < len(list1):
        print(list1[i])
        i += 1
```

نتیجہ:

```
Red
Green
Blue
Yellow
Black
```

9.4 لسٹ میتھڈ اور بِلٹ ان فنکشن
(LIST METHODS AND BUILT-IN FUNCTIONS)

ڈیٹا ٹائپ لسٹ میں متعدد بِلٹ ان فنکشن ہوتے ہیں جو پروگرامنگ کے لیے بہت کارآمد ہیں۔ان میں سے کچھ فنکشن کو جدول 9.1 میں دیے گئے ہیں۔

جدول 9.1 لسٹ میں ردوبدل کے لیے بِلٹ ان فنکشن

میتھڈ	وضاحت	مثال

len()	آرگیومینٹ کے طور پر پاس کی جانے والی لسٹ کی لمبائی یا تعداد کو ظاہر کرتا ہے۔	`>>> list1 = [10,20,30,40,50]`
		`>>> len(list1)`
		`5`

list()	اگر کوئی آرگیومینٹ پاس نہیں کیا جاتا ہے تو یہ خالی لسٹ کی تشکیل کرتا ہے۔
	اگر کوئی تسلسل آرگیومینٹ کے طور پر پاس کر دیا جاتا ہے تو یہ لسٹ کی تشکیل کرتا ہے۔
		`>>> list1 = list()`
		`>>> list1`
		`[ ]`
		`>>> str1 = 'aeiou'`
		`>>> list1 = list(str1)`
		`>>> list1`
		`['a', 'e', 'i', 'o', 'u']`

append()	آرگیومینٹ کے طور پر پاس کیے جانے والے واحد عنصر کو لسٹ کے آخر میں شامل کر دیتا ہے۔
	واحد عنصر بھی لسٹ ہو سکتا ہے۔
		`>>> list1 = [10,20,30,40]`
		`>>> list1.append(50)`
		`>>> list1`
		`[10, 20, 30, 40, 50]`
		`>>> list1 = [10,20,30,40]`
		`>>> list1.append([50,60])`
		`>>> list1`
		`[10, 20, 30, 40, [50, 60]]`

extend()	آرگیومینٹ کے طور پر پاس کیے جانے والے ہر ایک عنصر کو لسٹ کے آخر میں شامل کر دیتا ہے۔	`>>> list1 = [10,20,30]`
		`>>> list2 = [40,50]`
		`>>> list1.extend(list2)`
		`>>> list1`
		`[10, 20, 30, 40, 50]`

insert	ایک مخصوص انڈیکس پر عنصر کو لسٹ میں داخل کرتا ہے۔	`>>> list1 = [10,20,30,40,50]`
		`>>> list1.insert(2,25)`
		`>>> list1`
		`[10, 20, 25, 30, 40, 50]`
		`>>> list1.insert(0,5)`
		`>>> list1`
		`[5, 10, 20, 25, 30, 40, 50]`

count()	دیا ہوا کوئی عنصر کسی لسٹ میں جتنی مرتبہ ظاہر ہوتا ہے اس تعداد کو بتاتا ہے۔	`>>> list1 = [10,20,30,10,40,10]`
		`>>> list1.count(10)`
		`3`
		`>>> list1.count(90)`
		`0`

index()	لسٹ میں عنصر کے پہلے وقوع کے انڈیکس کو ظاہر کرتا ہے۔ اگر عنصر موجود نہیں ہے تو ValueError ظاہر ہوجاتی ہے۔	`>>> list1 = [10,20,30,20,40,10]`
		`>>> list1.index(20)`
		`1`
		`>>> list1.index(90)`
		`ValueError: 90 is not in list`

remove()	دیے ہوئے عنصر کو لسٹ سے خارج کر دیتا ہے۔ اگر وہ عنصر متعدد مرتبہ موجود ہے تو صرف پہلی مرتبہ وقوع پذیر ہونے والے عنصر کو خارج کرتا ہے۔ اگر عنصر موجود نہیں ہے تو ValueError ظاہر ہوجاتی ہے۔	`>>> list1 = [10,20,30,40,50,30]`
		`>>> list1.remove(30)`
		`>>> list1`
		`[10, 20, 40, 50, 30]`
		
		`>>> list1.remove(90)`
		`ValueError:list.remove(x):x not in list`

pop()	اس عنصر کو ظاہر کرتا ہے جس کا انڈیکس اس فنکشن کے پیرامیٹر کے طور پر پاس کیا جاتا ہے اور اسے لسٹ سے بھی خارج کر دیتا ہے۔ اگر کوئی بھی پیرامیٹر نہیں دیا ہوا ہے تو یہ لسٹ کے آخری عنصر کو ظاہر کرتا ہے اور اسے لسٹ سے خارج کر دیتا ہے۔	`>>> list1 = [10,20,30,40,50,60]`
		`>>> list1.pop(3)`
		`40`
		`>>> list1`
		`[10, 20, 30, 50, 60]`
		`>>> list1 = [10,20,30,40,50,60]`
		`>>> list1.pop()`
		`60`
		`>>> list1`
		`[10, 20, 30, 40, 50]`

reverse()	دی ہوئی لسٹ میں عناصر کی ترتیب کو پلٹ دیتا ہے۔	`>>> list1 = [34,66,12,89,28,99]`
		`>>> list1.reverse()`
		`>>> list1`
		`[ 99, 28, 89, 12, 66, 34]`
		
		`>>> list1 = [ 'Tiger' ,'Zebra' , 'Lion' , 'Cat' ,'Elephant' ,'Dog']`
		`>>> list1.reverse()`
		`>>> list1`
		`['Dog', 'Elephant', 'Cat', 'Lion', 'Zebra', 'Tiger']`

sort()	یہ لسٹ کو پیرامیٹر کے طور پر لیتا ہے اور ایک نئی لسٹ کی تشکیل کرتا ہے جس میں وہی عناصر صعودی یا نزولی ترتیب میں موجود ہوتے ہیں۔	`>>>list1=['Tiger','Zebra','Lion', 'Cat', 'Elephant' ,'Dog']`
		`>>> list1.sort()`
		`>>> list1`
		`['Cat', 'Dog', 'Elephant', 'Lion', 'Tiger', 'Zebra']`
		
		`>>> list1 = [34,66,12,89,28,99]`
		`>>> list1.sort(reverse = True)`
		`>>> list1`
		`[99,89,66,34,28,12]`

Sorted()		>>> list1 = [23,45,11,67,85,56]
		>>> list2 = sorted(list1)
		>>> list1
		[23, 45, 11, 67, 85, 56]
		>>> list2
		[11, 23, 45, 56, 67, 85]

min()
max()
sum()
	لسٹ کے کمترین یا سب سے چھوٹے عنصر کو ظاہر کرتا ہے۔
	لسٹ کے بیش ترین یا سب سے بڑے عنصر کو ظاہر کرتا ہے۔
	لسٹ کے عناصر کے حاصل جمع کو ظاہر کرتا ہے۔
		>>> list1 = [34,12,63,39,92,44]
		>>> min(list1)
		12
		>>> max(list1)
		92
		>>> sum(list1)
		284

9.5 نیسٹیڈ لسٹ (NESTED LIST)

جب کوئی لسٹ کسی دوسری لسٹ کے عنصر کے طور پر ظاہر ہوتی ہے تو اسے نیسٹیڈ لسٹ کہا جاتا ہے۔

مثال 9.2

```
>>> list1 = [1,2,'a','c',[6,7,8],4,9]
#fifth element of list is also a list
>>> list1[4]
[6, 7, 8]
```

نیسٹیڈ لسٹ List1 کے عنصر کو ایکسس کرنے کے لیے، ہمیں دو انڈیکس list1[i][j] کی تخصیص کرنی ہوگی۔
پہلا انڈیکس i ہمیں مطلوبہ نیسٹیڈ لسٹ تک لے جائے گا اور دوسرا انڈیکس j ہمیں اس نیسٹیڈ لسٹ کے مطلوبہ عنصر تک لے جائے گا۔

پروگرام 2-10 عارضی متغیر کا استعمال کیے بغیر دو اعداد کو ایک دوسرے سے بدلنے کے لیے پروگرام لکھیے۔

```
>>> list1[4][1]
7
#index i gives the fifth element of list1
#which is a list
#index j gives the second element in the
#nested list
```

9.6 لسٹ کی نقل تیار کرنا (COPYING LISTS)

ایک لسٹ دی ہوئی ہے، اس لسٹ کی نقل تیار کرنے کا آسان طریقہ یہ ہے کہ اسے دوسری لسٹ کو تفویض کر دیا جائے۔

```
>>> list1 = [1,2,3]
>>> list2 = list1
>>> list1
[1, 2, 3]
>>> list2
[1, 2, 3]
```

بیان list2 = list1 نئی لسٹ کی تشکیل نہیں کرتا ہے یہ صرف list1 بناتا ہے اور list2 اسی لسٹ آبجیکٹ کو ظاہر کرتی ہے۔ یہاں list2، list1 کی نقل بن جاتی ہے۔ لہٰذا، ان میں سے کسی ایک لسٹ میں کی گئی تبدیلی دوسری لسٹ میں بھی ظاہر ہوگی۔

```
>>> list1.append(10)
>>> list1
[1, 2, 3, 10]
>>> list2
[1, 2, 3, 10]
```

ہم کسی لسٹ کی نقل یا اس کا مثنیٰ ایک ممتاز آبجیکٹ کے طور پر تین طریقوں کی مدد سے تیار کر سکتے ہیں۔ پہلے طریقے میں سلائسنگ، دوسرے طریقے میں بلٹ ان فنکشن list() کا استعمال کرتے ہیں اور تیسرے طریقے کے تحت پائتھن لائبریری کے copy() فنکشن کا استعمال کیا جاتا ہے۔

طریقہ 1

ہم اپنی اصل لسٹ کو سلائس کر کے نئے متغیرہ میں اسٹور کر سکتے ہیں جیسا کہ نیچے دکھایا گیا ہے۔

```
newList = oldList[:]
```

مثال 9.3

```
>>> list1 = [1,2,3,4,5]
>>> list2 = list1[:]
>>> list2
[1, 2, 3, 4, 5]
```

طریقہ 2

ہم مندرجہ ذیل طریقے سے بلٹ ان فنکشن list() کا استعمال کر سکتے ہیں:

```
newList = list(oldList)
```

مثال 9.4

```
>>> list1 = [10,20,30,40]
>>> list2 = list(list1)
>>> list2
[10, 20, 30, 40]
```

طریقہ 3

ہم copy() فنکشن کا استعمال مندرجہ ذیل طریقے سے کر سکتے ہیں:

```
import copy        #import the library copy
#use copy()function of library copy
newList = copy.copy(oldList)
```

مثال 9.5

```
>>> import copy
>>> list1 = [1,2,3,4,5]
>>> list2 = copy.copy(list1)
>>> list2
[1, 2, 3, 4, 5]
```

9.7 فنکشن کے آرگیومینٹ کے طور پر لسٹ
(LIST AS ARGUMENT TO A FUNCTION)

جب کسی لسٹ کو فنکشن کے آرگیومینٹ کے طور پر پاس کیا جاتا ہے تو ہمیں دو صورت حال کا سامنا ہوتا ہے۔

(A) اصل لسٹ کے عناصر تبدیل ہو سکتے ہیں۔

یعنی فنکشن کے تحت لسٹ میں کی گئی تبدیلیاں کالنگ فنکشن میں ظاہر ہوتی ہیں۔

مثال کے طور پر مندرجہ ذیل پروگرام میں اعداد کی لسٹ list1 کو آرگیومینٹ کے طور پر فنکشن increment() کو پاس کیا جاتا ہے۔ یہ فنکشن لسٹ کے ہر ایک عنصر میں 5 کا اضافہ کر دیتا ہے۔

پروگرام 9-1 لسٹ کے عناصر میں اضافہ کرنے کے لیے پروگرام۔ لسٹ کو آرگیومینٹ کے طور پر فنکشن کو پاس کر دیا جاتا ہے۔

```
#Program 9-1
#Function to increment the elements of the list passed as argument
def increment(list2):
     for i in range(0,len(list2)):
          #5 is added to individual elements in the list
           list2[i] += 5
     print('Reference of list Inside Function',id(list2))
#end of function
list1 = [10,20,30,40,50]   #Create a list
print("Reference of list in Main",id(list1))
print("The list before the function call")
print(list1)
increment(list1)           #list1 is passed as parameter to function
print("The list after the function call")
print(list1)
```

نتیجہ:

```
Reference of list in Main 70615968
The list before the function call
[10, 20, 30, 40, 50]
Reference of list Inside Function 70615968 #The id remains same
The list after the function call
[15, 25, 35, 45, 55]
```

مشاہدہ کیجیے کہ جب ہم لسٹ کو آرگیومینٹ کے طور پر پاس کرتے ہیں تو درحقیقت ہم لسٹ کا حوالہ پاس کرتے ہیں۔ لہٰذا فنکشن کے اندر list2 میں کی گئی کوئی بھی تبدیلی اصل لسٹ list1 میں بھی نظر آئے گی۔

(B) اگر فنکشن کے اندر لسٹ کو ایک نئی قدر تفویض کی جاتی ہے تو ایک نئے لسٹ آبجیکٹ کی تشکیل کر دی جاتی ہے اور یہ فنکشن کی نقل بن جاتی ہے۔ فنکشن کی نقل میں کی جانے والی کوئی بھی تبدیلی طلب کیے جانے والے فنکشن میں ظاہر نہیں ہوتی ہے۔

پروگرام 9-2 پیرامیٹر کے طور پر پاس کی گئی لسٹ کے عناصر میں اضافہ کرنے والا پروگرام

```
#Program 9-2
#Function to increment the elements of the list passed as argument
def increment(list2):
     print("\nID of list inside function before assignment:",
id(list2))
     list2 = [15,25,35,45,55] #List2 assigned a new list
     print("ID of list changes inside function after assignment:",
id(list2))
     print("The list inside the function after assignment is:")
     print(list2)
#end of function

list1 = [10,20,30,40,50]          #Create a list
print("ID of list before function call:",id(list1))
print("The list before function call:")
print(list1)
increment(list1)  #list1 passed as parameter to function
print('\nID of list after function call:',id(list1))
print("The list after the function call:")
print(list1)
```

نتیجہ:

```
ID of list before function call: 65565640
The list before function call:
[10, 20, 30, 40, 50]

ID of list inside function before assignment:65565640
ID of list changes inside function  after assignment:65565600
```

```
The list inside the function after assignment is:
[15, 25, 35, 45, 55]

ID of list after function call: 65565640
The list after the function call:
[10, 20, 30, 40, 50]
```

9.8 لسٹ میں ردوبدل (LIST MANIPULATION)

اس باب میں ہم لسٹ کی تشکیل کرنے اور لسٹ میں ردو بدل کے مختلف طریقوں کا مطالعہ کر چکے ہیں۔ مندرجہ ذیل پروگراموں میں ہم لسٹ میں ردوبدل کے مختلف طریقوں کا اطلاق کریں گے۔

پروگرام 9-3 لسٹ پر مندرجہ ذیل مختلف عملوں کو انجام دینے کے لیے مینو کی مدد سے چلایا جانے والا ایک پروگرام لکھیے۔

- لسٹ میں عنصر کو جوڑنا
- عنصر کو داخل کرنا
- ایک لسٹ کو دی ہوئی لسٹ کے ساتھ جوڑنا
- موجودہ عنصر میں ترمیم کرنا
- موجودہ عنصر کو اس کے مقام سے حذف کرنا
- دی ہوئی قدر (ویلیو) والے موجودہ عنصر کو حذف کرنا
- لسٹ کو صعودی ترتیب میں لگانا
- لسٹ کو نزولی ترتیب میں لگانا
- لسٹ کو ڈسپلے کرنا

```
#Program 9-3
#Menu driven program to do various list operations
myList = [22,4,16,38,13] #myList already has 5 elements
choice = 0
while True:
    print("The list 'myList' has the following elements", myList)
    print("\nL I S T   O P E R A T I O N S")
    print(" 1. Append an element")
    print(" 2. Insert an element at the desired position")
    print(" 3. Append a list to the given list")
    print(" 4. Modify an existing element")
    print(" 5. Delete an existing element by its position")
    print(" 6. Delete an existing element by its value")
    print(" 7. Sort the list in ascending order")
```

```
print(" 8. Sort the list in descending order")
print(" 9. Display the list")
print(" 10. Exit")
choice = int(input("ENTER YOUR CHOICE (1-10): "))

#append element
if choice == 1:
    element = int(input("Enter the element to be appended: "))
    myList.append(element)

        print("The element has been appended\n")

    #insert an element at desired position
    elif choice == 2:
        element = int(input("Enter the element to be inserted: "))
        pos = int(input("Enter the position:"))
        myList.insert(pos,element)
        print("The element has been inserted\n")

    #append a list to the given list
    elif choice == 3:
       newList = eval(input( "Enter the elements separated by commas"))
       myList.extend(list(newList))
       print("The list has been appended\n")

    #modify an existing element
    elif choice == 4:
        i = int(input("Enter the position of the element to be
modified: "))
        if i < len(myList):
            newElement = int(input("Enter the new element: "))
            oldElement = myList[i]
            myList[i] = newElement
            print("The element",oldElement,"has been modified\n")
        else:
            print("Position of the element is more than the length
of list")

    #delete an existing element by position
    elif choice == 5:
        i = int(input("Enter the position of the element to be
deleted: "))
        if i < len(myList):
            element = myList.pop(i)
            print("The element",element,"has been deleted\n")
        else:
            print("\nPosition of the element is more than the length
of list")
    #delete an existing element by value
    elif choice == 6:
        element = int(input("\nEnter the element to be deleted: "))
```

```
    if element in myList:
        myList.remove(element)
        print("\nThe element",element,"has been deleted\n")
    else:
        print("\nElement",element,"is not present in the list")

#list in sorted order
    elif choice == 7:
        myList.sort()
        print("\nThe list has been sorted")

    #list in reverse sorted order
    elif choice == 8:
        myList.sort(reverse = True)
        print("\nThe list has been sorted in reverse order")

    #display the list
    elif choice == 9:
        print("\nThe list is:", myList)

    #exit from the menu
    elif choice == 10:
        break
    else:
        print("Choice is not valid")
        print("\n\nPress any key to continue.............")
        ch = input()
```

نتیجہ:

```
The list 'myList' has the following elements [22, 4, 16, 38, 13]

L I S T   O P E R A T I O N S
 1. Append an element
 2. Insert an element at the desired position
 3. Append a list to the given list
 4. Modify an existing element
 5. Delete an existing element by its position
 6. Delete an existing element by its value
 7. Sort the list in ascending order
 8. Sort the list in descending order
 9. Display the list
10. Exit
ENTER YOUR CHOICE (1-10): 8

The list has been sorted in reverse order
```

```
The list 'myList' has the following elements [38, 22, 16, 13, 4]

L I S T   O P E R A T I O N S
 1. Append an element
 2. Insert an element at the desired position
 3. Append a list to the given list
 4. Modify an existing element
 5. Delete an existing element by its position
 6. Delete an existing element by its value
 7. Sort the list in ascending order
 8. Sort the list in descending order
 9. Display the list
10. Exit

ENTER YOUR CHOICE (1-10): 5
Enter the position of the element to be deleted: 2
The element 16 has been deleted

The list 'myList' has the following elements [38, 22, 13, 4]

L I S T   O P E R A T I O N S
 1. Append an element
 2. Insert an element at the desired position
 3. Append a list to the given list
 4. Modify an existing element
 5. Delete an existing element by its position
 6. Delete an existing element by its value
 7. Sort the list in ascending order
 8. Sort the list in descending order
 9. Display the list
10. Exit
```

پروگرام 9-4 فنکشن کا استعمال کرکے n طلبا کے اوسط نمبروں کی تحسیب کرنے کے لیے پروگرام جس میں n کو استعمال کنندہ کے ذریعے داخل کیا جاتا ہے۔

```
#Program 9-4
#Function to calculate average marks of n students
def computeAverage(list1,n):
     #initialize total
     total = 0
     for marks in list1:
          #add marks to total
           total = total + marks
     average = total / n
     return average
```

```
#create an empty list
list1 = []
print("How many students marks you want to enter: ")
n = int(input())
for i in range(0,n):
     print("Enter marks of student",(i+1),":")
     marks = int(input())
     #append marks in the list
     list1.append(marks)
average = computeAverage(list1,n)
print("Average marks of",n,"students is:",average)
```

نتیجہ:

```
How many students marks you want to enter:
5
Enter marks of student 1:
45
   Enter marks of student 2:
   89
   Enter marks of student 3:
   79
   Enter marks of student 4:
   76
   Enter marks of student 5:
   55
   Average marks of 5 students is: 68.8
```

پروگرام 9-5 استعمال کنندہ کے ذریعے تعریف شدہ فنکشن لکھیے جس کی مدد سے اس بات کی جانچ کی جاسکے کہ آیا کوئی عدد لسٹ میں موجود ہے یا نہیں۔ اگر عدد موجود ہے تو عدد کا مقام ظاہر کیجیے۔ اگر عدد موجود نہیں ہے تو ایک مناسب میسیج پرنٹ کیجیے۔

```
#Program 9-5
#Function to check if a number is present in the list or not
def linearSearch(num,list1):
     for i in range(0,len(list1)):
          if list1[i] == num:        #num is present
               return i              #return the position
     return None                     #num is not present in the list
#end of function

list1 = []                           #Create an empty list
```

```
print("How many numbers do you want to enter in the list: ")
maximum = int(input())
print("Enter a list of numbers: ")
for i in range(0,maximum):
     n = int(input())
     list1.append(n)                    #append numbers to the list
num = int(input("Enter the number to be searched: "))
result = linearSearch(num,list1)
if result is None:
     print("Number",num,"is not present in the list")
else:
     print("Number",num,"is present at",result + 1, "position")
```

نتیجہ:

```
How many numbers do you want to enter in the list:
5
Enter a list of numbers:
23
567
12
89
324
Enter the number to be searched:12
Number 12 is present at 3 position
```

خلاصہ

- پائتھن میں لسٹ قابل تغیر تسلسل ہیں یعنی ہم لسٹ کے عناصر کو تبدیل کر سکتے ہیں۔
- لسٹ کے عناصر کو چھوٹے بریکٹ (Round Bracket) میں لکھا جاتا ہے اور یہ عناصر ایک دوسرے سے کوما کے ذریعے علاحدہ رہتے ہیں۔
- اگر ایک لسٹ کے اندر دوسری لسٹ موجود ہے تو اسے نیسٹیڈ لسٹ کہتے ہیں۔ لسٹ کی انڈیکسنگ کا طریقہ اسٹرنگ کے مانند ہی ہے اور یہ 0 سے شروع ہوتی ہے۔ دوطرفہ انڈیکسنگ کی مدد سے آگے اور پیچھے دونوں سمتوں میں لسٹ کا اعادہ کیا جا سکتا ہے۔
- آپریٹر '+' ایک لسٹ کو دوسری لسٹ کے اختتامی سرے کے ساتھ جوڑتا ہے۔
- آپریٹر '*' لسٹ کی متعینہ تعداد میں تکرار کرتا ہے۔
- ممبرشپ آپریٹر 'in' یہ بتاتا ہے کہ آیا کوئی عنصر لسٹ میں موجود ہے یا نہیں اور 'not in' آپریٹر اس کے برعکس کام کرتا ہے۔

نوٹ

- سلائسنگ کا استعمال لسٹ کے کسی ایک حصے کے استخراج کے لیے کیا جاتا ہے۔
- لسٹ میں ردو بدل کے لیے متعدد فنکشن مندرجہ ذیل ہیں: len(), list(), append(), extend(), insert(),count(), find(), remove(), pop(), reverse(), sort(), sorted(), min(), max(), sum()

مشق

1۔ مندرجہ ذیل بیانات کا نتیجہ (آؤٹ پُٹ) کیا ہوگا؟

i.
```
list1 = [12,32,65,26,80,10]
list1.sort()
print(list1)
```

ii.
```
list1 = [12,32,65,26,80,10]
sorted(list1)
print(list1)
```

iii.
```
list1 = [1,2,3,4,5,6,7,8,9,10]
list1[::-2]
list1[:3] + list1[3:]
```

iv.
```
list1 = [1,2,3,4,5]
list1[len(list1)-1]
```

2۔ مندرجہ ذیل لسٹ myList پر پر غور کیجیے۔ مندرجہ ذیل دونوں عملوں کو انجام دینے کے بعد myList کے عناصر کیا ہوں گے؟

```
myList = [10,20,30,40]
 i. myList.append([50,60])
ii. myList.extend([80,90])
```

3۔ مندرجہ ذیل کوڈنگ کا کیا نتیجہ برآمد ہوگا؟

```
myList = [1,2,3,4,5,6,7,8,9,10]
for i in range(0,len(myList)):
    if i%2 == 0:
        print(myList[i])
```

4۔ مندرجہ ذیل کوڈنگ کا کیا نتیجہ برآمد ہوگا؟

a.
```
myList = [1,2,3,4,5,6,7,8,9,10]
del myList[3:]
print(myList)
```

b.
```
myList = [1,2,3,4,5,6,7,8,9,10]
```

نوٹ

```
    del myList[:5]
    print(myList)
c.  myList = [1,2,3,4,5,6,7,8,9,10]
c.  del myList[::2]
    print(myList)
```

5۔ لسٹ کے append() اور extend() فنکشن کے درمیان فرق بتائیے۔

6۔ مندرجہ ذیل لسٹ پر غور کیجیے:

```
list1 = [6,7,8,9]
```

list1 پر مندرجہ ذیل عملوں (آپریشن) کے درمیان کیا فرق ہے؟

```
a. list1 * 2
b. list1 *= 2
c. list1 =  list1 * 2
```

7۔ ایک طالب علم کے ریکارڈ (نام، رول نمبر، پانچ مضامین کے نمبر اور نمبروں کا فیصد) کو مندرجہ ذیل لسٹ میں اسٹور کیا گیا ہے۔

```
stRecord = ['Raman','A-36',[56,98,99,72,69],
           78.8]
```

لسٹ stRecord سے مندرجہ ذیل معلومات کی بازیافت کے لیے پائتھن اسٹیٹمنٹ لکھیے۔

(a) طالب علم کا فیصد

(b) پانچویں مضمون کے نمبر

(c) طالب علم کے زیادہ سے زیادہ نمبر

(d) طالب علم کا رول نمبر

(e) طالب کے نام کو تبدیل کر کے Raman سے Raghav کیجیے۔

پروگرامنگ سوالات (PROGRAMMING PROBLEMS)

1۔ ایک پروگرام لکھیے جس کی مدد سے یہ معلوم کیا جا سکے کہ کوئی عنصر لسٹ میں کتنی مرتبہ موجود ہے۔

2۔ n صحیح اعداد (مثبت اور منفی) پر مشتمل ایک لسٹ کو پڑھنے کے لیے پروگرام لکھیے۔ دو نئی لسٹ اس طرح تشکیل دیجیے کہ ان میں سے ایک لسٹ میں سبھی مثبت اعداد اور دوسری میں سبھی منفی اعداد موجود ہوں۔

3۔ ایک ایسا فنکشن لکھیے جو پیرامیٹر کے طور پر پاس کی گئی لسٹ کے سب سے بڑے عنصر کو ظاہر کرتا ہے۔

4۔ ایک ایسا فنکشن لکھیے جو لسٹ میں موجود دوسرے سب سے بڑے عنصر کو ظاہر کرتا ہے۔

5۔ n صحیح اعداد پر مشتمل ایک لسٹ کو پڑھنے اور ان کا وسطانیہ معلوم کرنے کے لیے پروگرام لکھیے۔

نوٹ: مختلف قدروں پر مشتمل لسٹ کی وسطانی قدر درمیان میں موجود قدر ہوتی ہے بشرطیکہ قدروں کو

نوٹ

صعودی یا نزولی ترتیب میں مرتب کیا گیا ہو۔ اگر درمیان میں دو قدریں موجود ہیں تو ان دونوں قدروں کا اوسط لیجیے۔

اشارہ: آپ لسٹ کو صعودی یا نزولی ترتیب میں لانے کے لیے بلٹ ان فنکشن کا استعمال کر سکتے ہیں۔

6۔ عناصر پر مشتمل ایک لسٹ کو پڑھنے کے لیے پروگرام لکھیے۔ اس لسٹ میں اس طرح ترمیم کیجیے کہ اس کے اندر کسی بھی عنصر کا مثنیٰ موجود نہ ہو یعنی لسٹ میں متعدد مرتبہ موجود سبھی عناصر صرف ایک مرتبہ ظاہر ہونے چاہئیں۔

7۔ عناصر پر مشتمل ایک لسٹ کو پڑھنے کے لیے پروگرام لکھیے۔ لسٹ میں داخل کرنے کے لیے استعمال کنندہ سے ایک عنصر کو ان پُٹ کرنے کے لیا کہا جائے۔ وہ مقام بھی ان پُٹ کیجیے جہاں اس عنصر کو داخل کیا جانا ہے۔ اس عنصر کو لسٹ میں مطلوبہ مقام پر داخل کرنے کے لیے استعمال کنندہ کے ذریعے تعریف شدہ فنکشن لکھیے۔

8۔ عناصر پر مشتمل ایک لسٹ کو پڑھنے کے لیے پروگرام لکھیے۔

(a) اس پروگرام کے ذریعے لسٹ سے حذف کیے جانے والے عنصر کے مقام کو ان پُٹ کرنے کے لیے کہا جائے۔ لسٹ میں مطلوبہ مقام پر موجود عنصر کو حذف کرنے کے لیے فنکشن لکھیے۔

(b) اس پروگرام کے ذریعے لسٹ سے حذف کیے جانے والے عنصر کی قدر (ویلیوں) کو ان پُٹ کرنے کے لیے کہا جائے۔ مطلوبہ قدر والے عنصر کو لسٹ سے حذف کرنے کے لیے فنکشن لکھیے۔

9۔ عناصر پر مشتمل ایک لسٹ کو پڑھیے۔ یہ لسٹ ایک ایسے فنکشن کو پاس کیجیے جو نئی لسٹ کی تشکیل کیے بغیر اس لسٹ کی ترتیب کو الٹ دیتا ہے۔