Python - Lists Operations

Operations on lists

Lists are mutable, i.e. the elements can be added, altered or removed from the list.

In [ ]:

```
my_list = [1 , 2, 3, 4, 5]
my_list
```

append() can be used to add an element to the list.

In [ ]:

```
my_list.append(6) # add new element 6 at end of list
my_list
```

In [ ]:

```
my_list.append([7,8]) # add list as new element in list
my_list
```

extend() can also be used to add list of elements into list.

In [ ]:

```
my_new_list = [1 , 2, 3, 4, 5]
my_new_list
```

In [ ]:

```
my_new_list.extend([6])  # an element can be added into the list
my_new_list
```

In [ ]:

```
my_new_list.extend([7, 8, 9 ])  # list of elements can also be added into the list
my_new_list
```

insert() can be used to add an element at the desired position.

In [ ]:

```
my_list = [1 , 2, 3, 4, 5]
my_list
```

In [ ]:

```
my_list.insert(0, 'a')  #insert 'a' at first position
my_list
```

In [ ]:

```
my_list.insert(len(my_list) , '*****')  #insert '***** at last position
my_list
```

remove() can be used to find the first occurance of the element and to remove it.

In [ ]:

```
my_list = [1 , 2, 3, 4, 5]
my_list
```

In [ ]:

```
my_list.remove(3) # find first occurance of '3' and remove it from the list
my_list
```

pop() can be used to remove an element from specified index and store its value in a variable.

In [ ]:

```
my_list = [1 , 2, 3, 4, 5]
my_list
```

In [ ]:

```
popped_element = my_list.pop(4)
print("list--->", my_list)
print("popped element--->", popped_element)
```

del() can also used to remove element from the list at specified location.

In [ ]:

```
my_list = [1 , 2, 3, 4, 5]
my_list
```

In [ ]:

```
del(my_list[1])  # removes the element at 1 index
my_list
```

Assignment operator can be used to alter the element of list at the desired index.

In [ ]:

```
my_list = [1 , 2, 3, 4, 5]
my_list
```

In [ ]:

```
my_list[0] = 1111  # change the element at first position to 1111
my_list
```

In [ ]:

```
my_list[2] = 'abcd'  # change the element at third position to 'abcd'
my_list
```

List Methods

sort() can be used to sort the list elements. Default is ascending order.

In [ ]:

```
my_list = [11 , 2, 39, 4, 57]
print("list before sorting--->", my_list)

print()

my_list.sort()
print("list after sorting--->", my_list)
```

sort order can be specified as argument. Default is reverse = false. Arranges elements in descending order

In [ ]:

```
my_list = [11 , 2, 39, 4, 57]
print("list before sorting--->", my_list)

print()

my_list.sort(reverse=True)
print("list after sorting--->", my_list)
```

Alternative way to get the sorted list using sorted() method. Original list is not modified.

In [ ]:

```
my_list = [11 , 2, 39, 4, 57]
print("list before sorted--->", my_list)

print()

print("list after reversing the order--->", list(sorted(my_list)))
```

reverse() can be used to reverse the order of list elements.

In [ ]:

```
my_list = [11 , 2, 39, 4, 57]
print("list before reversing the order--->", my_list)

print()

my_list.reverse()
print("list after reversing the order--->", my_list)
```

Alternative way to get the reversed list using reversed() method. Original list is not modified.

In [ ]:

```
my_list = [11 , 2, 39, 4, 57]
print("list before reversing the order--->", my_list)

print()

print("list after reversing the order--->", list(reversed(my_list)))
```

index(x) can be used to determine the location of first occurance of element 'x' in the list

In [ ]:

```
my_list = [11 , 2, 39, 4, 57]
my_list.index(39)  # find the location at which element 39 first occures
```

In [ ]:

```
my_list = [11 , 2, 39, 4, 57]
my_list.index(45)  # find the location at which element 45 first occures, throws error as e
```

count(x) can be used to determine the count of occurances of element 'x' in the list

In [ ]:

```
my_list = [11 , 2, 39, 4, 57, 39, 34, 39 ]
my_list.count(39)  # count the number of times 39 appears in the list
```

In [ ]:

```
my_list = [11 , 2, 39, 4, 57, 39, 34, 39 ]
my_list.count(9)  # count the number of times 9 appears in the list
```

clear() can be used to remove all the elements of the list.

In [ ]:

```
my_list = [11 , 2, 39, 4, 57, 39, 34, 39 ]
print("Original list ---> ", my_list)

my_list.clear()
print("List after clearning the elements---> ", my_list)
```

Splitting

The split method returns a list of the words of string, assuming that words are separated by white spaces.

In [ ]:

```
my_string = "this is my string"
my_string
```

In [ ]:

```
my_string.split()  # returns the list of words based on white space as delimeter
```

In [ ]:

```
comma_separated_string = "this, is, comma, separated, string"
comma_separated_string
```

In [ ]:

```
list1 = comma_separated_string.split(',')  # returns the list of words based on comma as de
list1
```

List Comprehensions

List comprehensions are powerful way to create the lists.

In [ ]:

```
list1 = [i for i in range(3)]  # creates a list of 3 elements starting from 0
list1
```

In [ ]:

```
list2 = [i*i for i in range(5)]  # creates a list of 5 elements starting from 0 to 4 , each
list2
```

In [ ]:

```
list3 = [i*i*i for i in range(1, 4)]  # creates a list of 3 elements starting from 1 to 3 ,
list3
```

In [ ]:

```
L = [ 1, 2, 3]
list4 = [i*10 for i in L] # create a list whose elements are 10 times the elements of origi
list4
```

Exercise

Q1. Write a program that accepts 5 lists from the user, each one of which contains three values in it assignment marks, midsem marks and compre marks.
(a) Compute the total marks obtained by each student
(b) Print out the average of the class.

In [ ]:

```
#Try it here
```

Q2. Ask a use to enter a sentence
(a) Print out the third word of sentence
(b) Print out every second word of sentence

In [ ]:

```
#Try it here
```

Q3. Create the following lists using a for loop.
(a) A list consisting of the integers 0 through 49
(b) A list containing the squares of integers 1 through 50
(c) The list ['aa', 'bb', 'cc'.....] that ends with 26 th letter 'z'.

In [ ]:

```
#Try it here
```

Q4. Write a program that counts the number of occurances of word in a sentence entered by user without using built in function.

In [ ]:

```
#Try it here
```

Q5. Ask a user to enter a list of strings. Create a new list that consists of those strings with their first characters removed.

In [ ]:

```
#Try it here
```

In [ ]:

```

```