Python - Lists

Lists are ordered collection of elements. The elements can be of any type like strings, numbers, booleans, lists, tuples etc. Each element of list has an address associated with it i.e. index. This index can be used to get access to the list elements.

Creation

Empty list can be created with two ways. Usually [] brackets are used to define a list , whereas the list() function is used for the creation of list from other collection.

In [ ]:

```
empty_list = []
empty_list
```

In [ ]:

```
other_empty_list = list()
other_empty_list
```

List can have multiple element into it.

In [ ]:

```
my_number_list = [1, 2, 3, 4, 5, 6, 7]
my_number_list
```

But it can hold elements of different kind as well.

In [ ]:

```
my_mixed_list = [1, 1.1, 'a', True]
my_mixed_list
```

Type of list can be checked with type() funtion.

In [ ]:

```
type(my_mixed_list)
```

Type of individual elements can be checked by accessing them with index.

In [ ]:

```
type(my_mixed_list[0])  #first element is integer
```

In [ ]:

```
type(my_mixed_list[1])  #second element is float
```

Length of list can be determined using the len() fuction.

In [ ]:

```
len(my_number_list)
```

In [ ]:

```
len(my_mixed_list)
```

List can be printed using the print() function.

In [ ]:

```
print(my_number_list)
```

In [ ]:

```
print(my_mixed_list)
```

Concatenation and Repetition

'+' can be used to add one list to the another.

In [ ]:

```
list1 = [1, 2, 3]
list2 = [4, 5]
join_list = list1 + list2
join_list
```

'*' can be used to repeat the elements of the list.

In [ ]:

```
list3 = [1, 2, 3] * 3 # repeat the element 1,2,3 three times in a list
list3
```

In [ ]:

```
[0] * 5 # repeat the element zero five times in a list
```

Indices

Index is the location where the list element is present in the list. List elements are ordered i.e. they maintain the order in which they are inserted into the list. Indexing starts from zero.

In [ ]:

```
my_list = [1, 1.1, 'a', True]
my_list
```

In [ ]:

```
my_list[0]  # access first element of list
```

In [ ]:

```
my_list[len(my_list) - 1] # aceess the last element of list
```

Negative indexing is also allowed on the lists.

In [ ]:

```
print(my_list[0] , "    " , my_list[-4])
print(my_list[1] , "    " , my_list[-3])
print(my_list[2] , "    " , my_list[-2])
print(my_list[3] , "    " , my_list[-1])
```

Slicing

Slice is portion of list accessed using the indices. [] operator allows us to extract part of list using the indices.

In [ ]:

```
my_list = [1, 1.1, 'a', True]
my_list
```

In [ ]:

```
my_list[ : ]  # extracts complete list
```

In [ ]:

```
my_list[ 0 : 3 ]  # extracts first three elements of list
```

In [ ]:

```
my_list[ 2 : 3 ]  # extracts third element of list
```

In [ ]:

```
my_list[ 3 :  ]  # extract all elements of list from thrid element onwards
```

In [ ]:

```
my_list[  : 2 ]  # extracts first two elements of list
```

In [ ]:

```
my_list[ 0 : 4 : 2] # from first to fourth element, extract every second element
```

In [ ]:

```
my_list[-2 : ] # extract last two elements of list
```

The in operator

The in operator is helpful to find out if a list contains an element in it or not.

In [ ]:

```
my_list = [1, 1.1, 'a', True]
my_list
```

In [ ]:

```
1 in my_list  # check whether 1 is present in list
```

In [ ]:

```
1 not in my_list # check whether 1 is not present in list
```

In [ ]:

```
'this' in my_list #check whether 'this' is present in the list or not
```

Looping

One can iterate over the elements present in the list using the for loop.

In [ ]:

```
my_list = [1, 1.1, 'a', True]
my_list
```

In [ ]:

```
for element in my_list:
    print(element)
```

In [ ]:

```
for i in range(len(my_list)):
    print( "(", i, ") element -----> ", my_list[i])
```

Built-in fuctions

Some useful built in functions available on the list containing elements of same type :

In [ ]:

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

sum() can be used to get sum of all elements in list.

In [ ]:

```
sum(my_number_list)
```

min() can be used to get minimum value out of all elements in list.

In [ ]:

```
min(my_number_list)
```

max() can be used to get maximum value out of all elements in list.

In [ ]:

```
max(my_number_list)
```

List aliasing

The same list can be referred by two different list variables. Changing the one list affects the other as well.

In [ ]:

```
list1 = [1, 2, 3, 4, 5]
print("Original list1")
print("list1---->   ", list1)

print()

list2 = list1  # list2 is referring to list1, both having same elements
print("List2 is referring to list1, both having same elements")
print("list1---->   ", list1)
print("list2---->   ", list2)


print()
#Alter the element of list1 , both lists will get altered
list1[0] = 'a' #change first element of list1
print("After aletring the first element of list1, list2 is also altered")
print("list1---->   ", list1)
print("list2---->   ", list2)

print()
#Alter the element of list2 , both lists will get altered
list2[3] = '******' #change fourth element of list2
print("After aletring the fourth element of list2, list1 is also altered")
print("list2---->   ", list2)
print("list1---->   ", list1)
```

List copying

In order to avoid the side effect, list elements can be copied using the following way. By that way, alterations in one list does not affect the elements in the other list from which its copied.

There are three ways of list copying.
new_list = orig_list.copy()
new_list = orig_list[:]
new_list = list(orig_list)

In [ ]:

```
list1 = [1, 2, 3, 4, 5]
print("Original list1")
print("list1---->   ", list1)

print()

list2 = list1[:]  # copy all elements of list1 into list2
print("List1 and list2 are two different list , both having same elements")
print("list1---->   ", list1)
print("list2---->   ", list2)


print()
#Alter the element of list1
list1[0] = 'a' #change first element of list1
print("After aletring the first element of list1, list2 is not altered")
print("list1---->   ", list1)
print("list2---->   ", list2)

print()
#Alter the element of list2
list2[3] = '******' #change fourth element of list2
print("After aletring the fourth element of list2, list1 is not altered")
print("list2---->   ", list2)
print("list1---->   ", list1)
```

Exercise

Q1. Write a program that accepts 5 integers from the user. Store those integers as part of list. Output the following without using any of the built-in fuction available :
(a) square of all the integers in the list
(b) sum of all the integers in the list
(c) product of all the integers in the list

In [ ]:

```
#Try it here
```

Q2. Write a program that will reverse the elements of the list without using any built-in function.

In [ ]:

```
#Try it here
```

Q3. Start with the list [8, 9 , 10]. Do the following :
(a) Set the second entry to 17
(b) Add 4,5, and 6 to the end of list
(c) Remove the first entry from the list

(d) Double the list

(e) Insert 25 at the index 3

```
    The final list should look like [17, 10, 4, 25, 5, 6, 17, 10, 4, 5, 6 ]
```

In [ ]:

```
#Try it here
```