02.Lists

September 17, 2025

0.1 Creating lists

Note Book Owner: Emdadul Hoque

0.1.1 creating list using constructor of the list class

[1]:
```
#create an empty list
l1 = list()
l1
```

[1]:
```
[]
```

[3]:
```
#create a list with integer elements
l2 = list([10, 20, 30, 40])
l2
```

[3]:
```
[10, 20, 30, 40]
```

[4]:
```
#create a list with string elements
l3 = list(["Banana", "Apple", "Grapes"])
l3
```

[4]:
```
['Banana', 'Apple', 'Grapes']
```

[9]:
```
#create a list with inbuilt range() function
l4 = list(range(1, 11))
l4
```

[9]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

[10]:
```
l5 = list("xyz")
l5
```

[10]:
```
['x', 'y', 'z']
```

0.1.2 Creat a list with out using constructor

[11]:
```
#create a list with any integer elements

l1 = [10, 20, 30, 40, 50]
l1
```

[11]: [10, 20, 30, 40, 50]

[12]:
```
#create a list with three string elements
l2 = ["Apple", "Banana", "Mangoes"]
l2
```

[12]: ['Apple', 'Banana', 'Mangoes']

[13]:
```
#create a list with different kinds of data types

l3 = ["Apple", "Banana", "Mangoes", 10, 20, 30, 40, 3.58, ["Emdadul",␣
 ↪"Tareque"]]
l3
```

[13]: ['Apple', 'Banana', 'Mangoes', 10, 20, 30, 40, 3.58, ['Emdadul', 'Tareque']]

0.2 Accessing the elements of a list

[14]:
```
l1 = [10, 20 , 30, 40, 50]
```

[16]:
```
l1[0]
```

[16]: 10

[17]:
```
l1[3]
```

[17]: 40

[23]:
```
# Access element of a list using negative indexing

print(l1)
print(l1[-1], l1[-2], l1[-3])
```

```
[10, 20, 30, 40, 50]
50 40 30
```

[24]:
```
# Access elemnt of list using loop

lst = [1, 2, 3, 4, 5, 6, 7, 8]

for item in lst:
    print(item, end=" ")
```

```
1 2 3 4 5 6 7 8
```

Problem 01: Find even number from a list

[27]:
```
lst = [1, 2, 3, 4, 5, 6, 7, 8]
print("Given list: ", lst)
print("Even numbers:", end=" ")
for item in lst:
    if item % 2 == 0:
        print(item, end=" ")
```

```
Given list:  [1, 2, 3, 4, 5, 6, 7, 8]
Even numbers: 2 4 6 8
```

0.3 List Slicing

[28]:
```
l1 = [10, 20, 30, 40, 50]
l1[1: 4]
```

[28]:
```
[20, 30, 40]
```

[29]:
```
l1[1::2]
```

[29]:
```
[20, 40]
```

[30]:
```
# reverse printing
l1[::-1]
```

[30]:
```
[50, 40, 30, 20, 10]
```

[31]:
```
l1[-2::-1]
```

[31]:
```
[40, 30, 20, 10]
```

0.4 Python inbuild functions for lists

len() –> Returns the number of elements in a list max() –> Returns the elements with the greatest value min() –> Returns the elements with the lowest value sum() –> Returns the sum of all the elements random.shuffle() –> Shuffles the elements randomly

[32]:
```
l1 = [10, 20, 30, 40, 50]
len(l1)
```

[32]:
```
5
```

[33]:
```
max(l1)
```

[33]:
```
50
```

[34]:
```
min(l1)
```

[34]: 10

[35]:
```
sum(l1)
```

[35]: 150

[38]:
```
import random
random.shuffle(l1)
```

[39]:
```
l1
```

[39]: [40, 30, 10, 20, 50]

0.5 The List Operator

[40]:
```
# the + operator: The concatenation operator is used to join two lists

a = [1, 2, 3]
b = [4, 5, 6]

print(a+b)
```

```
[1, 2, 3, 4, 5, 6]
```

[43]:
```
# the * operator: the multiplication operator is used to replicate the elements␣
 ↪of a list

list1 = [10, 20, 30]
list2 = [20, 30, 40]
```

[45]:
```
list2*2
```

[45]: [20, 30, 40, 20, 30, 40]

[46]:
```
# The in operator: The in operator used to determine whether an element is in a␣
 ↪list. it returns True if the elements is present and False return if the␣
 ↪element is absent in the list.

list1 = [10, 20 , 30]

30 in list1
```

[46]: True

[48]:
```
40 in list1
```

[48]: False

[49]:
```
# The is operator: is operator used for checking is two variable indocate samme␣
 ↪object or not

A = "Emdadul"
B = "Emdadul"

A is B
```

[49]: True

[50]:
```
A = [10, 20 , 30]
B = [10, 20, 30]

A is B
```

[50]: False

The del Operator: The del operator stands for Delete. The del operator is used to remove the elements from a list. To delete the element of a list, the elements of the list are accessed using their index position and the del operator is placed before them.

[51]:
```
lst = [10, 20, 30, 40, 50]
```

[52]:
```
del lst[3] #Remove 3rd element from the list
```

[53]:
```
lst
```

[53]: [10, 20, 30, 50]

[54]:
```
del lst[-1] # Remove lst element from the list
```

[55]:
```
lst
```

[55]: [10, 20, 30]

[56]:
```
del lst[1:3]
```

[57]:
```
lst
```

[57]: [10]

0.6 List comprehensions

[61]:
```
#with out list comprehension
lst = [10, 20, 30, 40, 50]

for i in range(0, len(lst)):
    lst[i] += 10
```

[62]:
```
lst
```

[62]:
```
[20, 30, 40, 50, 60]
```

[63]:
```
#Using list comprehension

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

lst = [x+10 for x in lst]
```

[67]:
```
lst
```

[67]:
```
[1, 2, 3, 4, 5, 6]
```

Problem 02: Write a program to create a list with lements 1,2,3,4,5. Display even elements of the list using list comprehension.

[72]:
```
lst = [1, 2, 3, 4, 5, 6]
print("Element of the list: ", lst)
even = [x for x in lst if x%2 == 0]
print("Even numbers from the list: ",even)
```

```
Element of the list:  [1, 2, 3, 4, 5, 6]
Even numbers from the list:  [2, 4, 6]
```

Problem 03: Write a program to create a list 'A' to generate squares of a number (from 1 to 10), list 'B' to generate cubes of a number (from 1 to 10) and list 'C' with those elements that are even and present in list 'A'.

[1]:
```
lst = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
A = [x*x for x in lst]
print("list A: ", A)
B = [x*x*x for x in lst]
print("list B: ", B)
c = [x for x in A if x%2==0]
print("Even number for A: ", c)
```

```
list A:  [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
list B:  [1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
Even number for A:  [4, 16, 36, 64, 100]
```

0.7 Traversing List

[5]:
```
lst = [1, 2, 3, 4, 5, 6]
#travers list in pythonic way

for item in lst:
    print(item, end=", ")
```

```
1, 2, 3, 4, 5, 6,
```

[6]:
```
#if you want to access index in pythonic way

for index, item in enumerate(lst):
    print(index, ": ", item)
```

```
0 :  1
1 :  2
2 :  3
3 :  4
4 :  5
5 :  6
```

0.8 List Methods

append() –> Adds an element x to the end of the list. None is the return type of method appended

[90]:
```
lst = [1, 2, 3, 4]
lst.append(5)
```

[91]:
```
lst
```

[91]:
```
[1, 2, 3, 4, 5]
```

extend() –> Appends all the elements of list

[92]:
```
lst.extend([6, 8, 9]) #                                    append()                ␣
 ↪                    extend()
```

[93]:
```
lst
```

[93]:
```
[1, 2, 3, 4, 5, 6, 8, 9]
```

clear() –> Removes all the items from the list.

[94]:
```
lst = [1, 2, 3, 4]
lst.clear()
```

[96]:
```
lst
```

[96]:
```
[]
```

count() –> Returns the number of times the element x appears in the list.

[97]:
```
lst = [1, 2, 3, 4, 1, 2, 3, 3, 3, 4, 5, 5]
lst.count(3)
```

[97]:
```
4
```

copy () –> This method returns a shallow copy of the list.

[98]:
```
lst = [1, 2 ,4, 3, 4, 5]
lst2 = lst.copy()
```

[99]:
```
lst2
```

[99]:
```
[1, 2, 4, 3, 4, 5]
```

[101]:
```
#different way
lst = [1, 2 ,4, 3, 4, 5]
lst2 = lst[:]
```

[102]:
```
lst2
```

[102]:
```
[1, 2, 4, 3, 4, 5]
```

index() –> Returns the index of the first occurrence of the element x from the list

[104]:
```
lst = [10, 20, 30, 40, 50]
lst.index(50)
```

[104]:
```
4
```

insert() –> Insert the element at a given index.

[133]:
```
lst = [10, 20, 30, 40]
lst.insert(3, 100)
```

[134]:
```
lst
```

[134]:
```
[10, 20, 30, 100, 40]
```

[135]:
```
import random
for i in range(0, 5):
    lst.insert(i, random.randint(0,10))
```

[136]:
```
print(lst)
```

```
[1, 7, 7, 4, 2, 10, 20, 30, 100, 40]
```

pop() –> Removes the element from the given position. Also, it returns the removed element.

[137]:
```
lst = [1, 2, 3, 4, 5, 6]
lst.pop(2)
```

[137]:
```
3
```

[138]:
```
lst
```

[138]:
```
[1, 2, 4, 5, 6]
```

remove() –> Removes the first occurrence of element x from the list

[139]:
```
lst = [1, 2, 3, 4, 5 ,6, 1, 2, 3, 4, 5]
lst.remove(3)
```

[140]:
```
lst
```

[140]:
```
[1, 2, 4, 5, 6, 1, 2, 3, 4, 5]
```

[141]:
```
lst.reverse() #reverse the elements of the list
```

[142]:
```
lst
```

[142]:
```
[5, 4, 3, 2, 1, 6, 5, 4, 2, 1]
```

[144]:
```
lst.sort() #sort the elements of list
```

[145]:
```
lst
```

[145]:
```
[1, 1, 2, 2, 3, 4, 4, 5, 5, 6]
```

0.9 Lists and Strings

[146]:
```
string = "python"
lst = list(string)
```

[147]:
```
lst
```

[147]:
```
['p', 'y', 't', 'h', 'o', 'n']
```

[150]:
```
#string split in a list
P="My-Data-of-Birth-03-June-1991"
P.split("-")
```

[150]:
```
['My', 'Data', 'of', 'Birth', '03', 'June', '1991']
```