List

August 15, 2024

List

[7]:
```
list1 = [] #Empty
```

[9]:
```
print(type(list1))
```

```
<class 'list'>
```

[11]:
```
list2 = [10,30,60] #list of interger number
```

[16]:
```
list3 = [10.77,30.66,60.89] #list of float number
list3
```

[16]:
```
[10.77, 30.66, 60.89]
```

[18]:
```
list4 = ['one','two','three'] #List of string
```

[22]:
```
list5 = ['Mandeep', 25, [50,100],[150, 90]] #Nested Lists
list5
```

[22]:
```
['Mandeep', 25, [50, 100], [150, 90]]
```

[26]:
```
list6 = [100, 'Mandeep', 17.568] #List of mixed data types
list6
```

[26]:
```
[100, 'Mandeep', 17.568]
```

[28]:
```
list7 = ['Mandeep', 25, [50, 100],[150,90],{'John', 'David'}]
list7
```

[28]:
```
['Mandeep', 25, [50, 100], [150, 90], {'David', 'John'}]
```

[30]:
```
len(list6) #Length of list
```

[30]:
```
3
```

List Indexing

[33]:
```
list2[0] #Retreive First element of the list
```

[33]:
```
10
```

[35]:
```
list4[0] # Retreive first element of the list
```

[35]:
```
'one'
```

[37]:
```
list4[0][0] # Nested indexing - Access the first character of the first list ele
```

[37]:
```
'o'
```

[39]:
```
list4[-1] #Last item of the list
```

[39]:
```
'three'
```

[41]:
```
list5[-1] #Last item of the list
```

[41]:
```
[150, 90]
```

List Slicing

[4]:
```
mylist = ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
```

[6]:
```
mylist[0:3] #Return all items from 0th to 3rd index location excluding the item
```

[6]:
```
['one', 'two', 'three']
```

[8]:
```
mylist[2:5] # List all items from 2nd to 5th index location excluding the item a
```

[8]:
```
['three', 'four', 'five']
```

[10]:
```
mylist[:3] # Return first three items
```

[10]:
```
['one', 'two', 'three']
```

[12]:
```
mylist[:2] # Return first two items
```

[12]:
```
['one', 'two']
```

[14]:
```
mylist[-3:] #Return last two items
```

[14]:
```
['six', 'seven', 'eight']
```

[16]:
```
mylist[-2:] #Return last two items
```

[16]:
```
['seven', 'eight']
```

[18]:
```
mylist[-1] #Return last item of the list
```

[18]:
```
'eight'
```

[20]:
```
mylist[:] #Return whole list
```

[20]:
```
['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
```

Add , Remove & Change Items

[58]:
```
mylist = ['one','two','three','four','five','six','seven','eight']
mylist
```

[58]:
```
['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
```

1. Append Add an item at the end of the list

[60]:
```
mylist.append('nine')
mylist
```

[60]:
```
['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine']
```

2. Insert Add item at the particular give location in the list

[62]:
```
mylist.insert(9,'ten')
mylist
```

[62]:
```
['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine', 'ten']
```

[64]:
```
mylist.insert(0,'NUMBER')
mylist
```

[64]:
```
['NUMBER',
 'one',
 'two',
 'three',
 'four',
 'five',
 'six',
 'seven',
 'eight',
 'nine',
 'ten']
```

3. Remove

[67]:
```
mylist.remove('NUMBER')
mylist
```

[67]:
```
['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine', 'ten']
```

4.POP Remove lat item of the list

[71]:
```
mylist.pop()
mylist
```

[71]:
```
['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine']
```

[73]:
```
mylist.pop(0) #Remove first element of the list
mylist
```

[73]:
```
['two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine']
```

5. Delete

[76]:
```
del mylist[7]
mylist
```

[76]:
```
['two', 'three', 'four', 'five', 'six', 'seven', 'eight']
```

6.Change Values

[79]:
```
mylist[0]= 1
mylist[1]= 2
mylist[2]= 3
mylist
```

[79]:
```
[1, 2, 3, 'five', 'six', 'seven', 'eight']
```

7. clear deletes all items in the list/Empty list

[83]:
```
mylist.clear()
mylist
```

[83]:
```
[]
```

8. Delete list

[86]:
```
del mylist
mylist
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[86], line 2
      1 del mylist
----> 2 mylist

NameError: name 'mylist' is not defined
```

9. Copy List

[89]:
```
mylist = ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight',␣
 ↪'nine']
mylist
```

[89]:
```
['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine']
```

[91]:
```
ourlist = mylist #New reference list "Yourlist"
```

New reference must be always before '='

[94]:
```
id(ourlist),id(mylist) #here both id/address will be same
```

[94]:
```
(2158470072960, 2158470072960)
```

But when you use copy function both will get different addresses

[97]:
```
ourlist = mylist.copy()
```

[99]:
```
id(ourlist), id(mylist)
```

[99]:
```
(2158470234368, 2158470072960)
```

[102]:
```
mylist[0]="Mandeep"
mylist
```

[102]:
```
['Mandeep', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine']
```

[104]:
```
ourlist
```

[104]:
```
['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine']
```

Here, as yourlist is a copy it will not affect any changes made in mylist

10. Join List

[108]:
```
list1 = ['one', 'two', 'three', 'four']
list2 = ['five', 'six', 'seven', 'eight']
```

Below third list in made by merging two individual lists

[115]:
```
list3 = list1 + list2 #Join two lists by '+' operator
list3
```

[115]:
```
['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
```

10. 1 extend() Below, second list is merged into first make them one

[121]:
```
list1.extend(list2) #Append list2 with list1
list1
```

[121]:
```
['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
```

11. List Membership

If item is persent in list –> True ; else –> False

[125]:
```
mylist
```

[125]: ['Mandeep', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine']

[127]:
```
'Mandeep' in mylist
```

[127]: True

[129]:
```
'one' in mylist
```

[129]: False

[134]:
```
if 'five' in list1:  # Check if 'five' exist in the list
    print('five is present in the list')
else:
    print('five is not present in the list')
```

five is present in the list

12. Reverse & Sort List

[140]:
```
list99 = [55,2,9,595,599,52,29,5,6,95]
list99
```

[140]: [55, 2, 9, 595, 599, 52, 29, 5, 6, 95]

[142]:
```
list99.reverse() #List gets reverse
list99
```

[142]: [95, 6, 5, 29, 52, 599, 595, 9, 2, 55]

8.0.1 list = list[::-1] -> reverses the list

[145]:
```
list99 = list99[::-1]
list99
```

[145]: [55, 2, 9, 595, 599, 52, 29, 5, 6, 95]

[147]:
```
list99.sort()  #sort list in ascending order
list99
```

[147]: [2, 5, 6, 9, 29, 52, 55, 95, 595, 599]

[149]:
```
list99.sort(reverse=True) #Here it sorts in descesnding order
list99                    # First sorts in ascending and internally reverses it
```

[149]: [599, 595, 95, 55, 52, 29, 9, 6, 5, 2]

13. Loop

[153]:
```
m1 = ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
m1
```

[153]:
```
['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
```

[155]:
```
for i in m1:
    print (i)
```

```
one
two
three
four
five
six
seven
eight
```

[174]:
```
for i in enumerate(m1):
    print(i)
```

```
(0, 'one')
(1, 'two')
(2, 'three')
(3, 'four')
(4, 'five')
(5, 'six')
(6, 'seven')
(7, 'eight')
```

enumerate automatically provide an index for each item

14. COUNT Tells number of times an item occured in the list

[188]:
```
list10 = [1,2,3,4,5,5,5,6,6,7,8,1]
list10
```

[188]:
```
[1, 2, 3, 4, 5, 5, 5, 6, 6, 7, 8, 1]
```

[192]:
```
list10.count(5)
```

[192]:
```
3
```

[194]:
```
list10.count(10)
```

[194]:
```
0
```

15. All/Any The all( ) method returns: True - If all elements in a list are true;

False - If any element in a list is false Similar to AND Gate *

The any( ) function returns True if any element in the list is True. If not, any() returns False. Similar to OR Gate +

[209]:
```
L1 = [1,2,3,4,0]
L1
```

[209]: [1, 2, 3, 4, 0]

[211]:
```
all(L1) #Returns False because as one value is False i.e 0
```

[211]: False

[213]:
```
any(L1) #Return True because we have items with True value
```

[213]: True

[215]:
```
L2 = [10,20,50,50,True]
L2
```

[215]: [10, 20, 50, 50, True]

[217]:
```
all(L2)
```

[217]: True

[219]:
```
L3=[0,0,False]
L3
```

[219]: [0, 0, False]

[221]:
```
all(L3)
```

[221]: False

[223]:
```
any(L3)
```

[223]: False