# UNIT - 4

## List Data Type

### Topics Covered:

#### Introduction

1. **Creation of List Objects**

#### Accessing elements of List

1. **Traversing the elements of List**

#### Important functions of List :

* 1. To get information about list
  2. Manipulating elements of List
  3. Ordering elements of List

#### Aliasing and Cloning of List objects

1. **Using Mathematical operators for List Objects**

#### Comparing List objects

1. **Membership operators**

#### clear() function

1. **Nested Lists**
2. **List Comprehensions**

### Introduction:

If we want to represent a **group of individual objects as a single entity** where insertion order is preserved and duplicates are allowed, then we should go for List.

Insertion order preserved.

Duplicate objects are allowed

Heterogeneous objects are allowed.

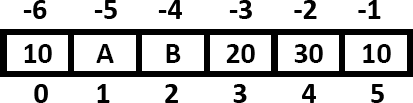
List is dynamic because based on our requirement we can increase the size and decrease the size.

In List the elements will be placed within square brackets and with comma seperator.

We can differentiate duplicate elements by using index and we can preserve insertion order by using index. Hence index will play very important role.

Python supports both positive and negative indexes. +ve index means from left to right where as negative index means right to left.

**Eg:** [10,"A","B",20, 30, 10]



List objects are mutable.i.e we can change the content.

### Creation of List Objects:

#### We can create empty list object as follows...

In [2]:

list**=**[]

print(list)

print(type(list))

[]

<class 'list'>

#### If we know elements already then we can create list as follows

In [4]:

list **=** [10,20,30,40]

print(list)

print(type(list))

[10, 20, 30, 40]

<class 'list'>

#### With dynamic input:

In [5]:

list**=**(input("Enter List:")) *# Entire input is considered as string*

print(list)

print(type(list))

Enter List:10,20,30,40 10,20,30,40

<class 'str'>

In [6]:

list**=**eval(input("Enter List:")) print(list)

print(type(list))

Enter List:[10,20,30,40] [10, 20, 30, 40]

<class 'list'>

In [8]:

list**=**eval(input("Enter List:")) print(list)

print(type(list))

Enter List:[ram,raj]

#### ---------------------------------------------------------------------------

**NameError** Traceback (most recent call last)

**<ipython-input-8-ac0b44db1317>** in <module>

**----> 1** list**=**eval**(**input**("Enter List:"))**

1. print**(**list**)**
2. print**(**type**(**list**))**

**<string>** in <module>

**NameError**: name 'ram' is not defined

In [9]:

list**=**eval(input("Enter List:")) print(list)

print(type(list))

Enter List:['ram','raj']

['ram', 'raj']

<class 'list'>

#### With list() function:

In [ ]:

l**=**list(range(0,10,2))

print(l)

*# Not working in jupyter notebook but executed in Editplus*

In [12]:

[0, 2, 4, 6, 8]

Out[12]:

[0, 2, 4, 6, 8]

In [ ]:

s**=**"durga"

l**=**list(s) print(l)

*# Not working in jupyter notebook but executed in editplus*

In [ ]:

['d', 'u', 'r', 'g', 'a']

#### With split() function:

In [16]:

s**=**"Learning Python is very very easy !!!" l**=**s.split()

print(l)

print(type(l))

['Learning', 'Python', 'is', 'very', 'very', 'easy', '!!!']

<class 'list'>

#### Note:

Sometimes we can take list inside another list,such type of lists are called **nested lists.** [10,20,[30,40]]

### 3. Accessing elements of List:

We can access elements of the list either by using **index** or by using **slice operator**(:)

#### By using index:

List follows zero based index. ie index of first element is zero.

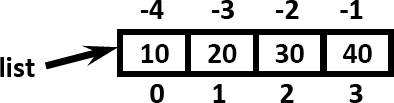
List supports both +ve and -ve indexes.

+ve index meant for Left to Right.

-ve index meant for Right to Left.

#### Eg :

list=[10,20,30,40]



list**=**[10,20,30,40]

print(list[0]) *#10*

print(list[**-**1]) *#40*

print(list[**-**4]) *#10*

print(list[10]) *#IndexError: list index out of range*

10

40

10

#### ---------------------------------------------------------------------------

**IndexError** Traceback (most recent call last)

**<ipython-input-20-c9b501127143>** in <module> 3 print**(**list**[-1]) #40**

1. print**(**list**[-4]) #10**

**----> 5** print**(**list**[10]) #IndexError: list index out of range IndexError**: list index out of range

In [23]:

list **=** [10,20,[30,40]]

print(list[2])

print(list[2][1])

[30, 40]

40

#### By using slice operator:

**Syntax:**

list2= list1[start:stop:step]

start ==>it indicates the index where slice has to start default value is 0

stop ===>It indicates the index where slice has to end default value is max allowed index of list ie length of the list

step ==>increment value

step default value is 1

In [24]:

l **=** [10,20,30,40,50,60]

print(l[::])

[10, 20, 30, 40, 50, 60]

In [25]:

l **=** [10,20,30,40,50,60]

print(l[::2])

[10, 30, 50]

In [26]:

l **=** [10,20,30,40,50,60]

print(l[::**-**1])

[60, 50, 40, 30, 20, 10]

In [27]:

l **=** [10,20,[30,40],50,60]

print(l[0:3:])

[10, 20, [30, 40]]

In [29]:

n**=**[1,2,3,4,5,6,7,8,9,10]

print(n[2:7:2]) *#3,5,7*

print(n[4::2]) *# 5,7,9*

print(n[3:7]) *#4,5,6,7*

print(n[8:2:**-**2]) *# 9,7,5*

print(n[4:100]) *# 5,6,7,8,9,10*

|  |  |  |
| --- | --- | --- |
| [3, | 5, | 7] |
| [5, | 7, | 9] |
| [4, | 5, | 6, 7] |
| [9, | 7, | 5] |
| [5, | 6, | 7, 8, 9, 10] |

### List vs mutability:

Once we creates a List object,we can modify its content. Hence List objects are mutable.

In [30]:

n**=**[10,20,30,40]

print(n) n[1]**=**777

print(n)

[10, 20, 30, 40]

[10, 777, 30, 40]

## Date: 02-05-2020 Day 2

### 4. Traversing the elements of List:

The sequential access of each element in the list is called traversal.

#### By using while loop:

In [2]:

n**=**[0,1,2,3,4,5,6,7,8,9,10]

i**=**0

**while** i**<**len(n): print(n[i]) i**=**i**+**1

0

1

2

3

4

5

6

7

8

9

10

#### By using for loop:

In [3]:

n**=**[0,1,2,3,4,5,6,7,8,9,10]

**for** n1 **in** n:

print(n1)

0

1

2

3

4

5

6

7

8

9

10

#### To display only even numbers:

In [4]:

n**=**[0,1,2,3,4,5,6,7,8,9,10]

**for** n1 **in** n:

**if** n1**%**2**==**0:

print(n1)

0

2

4

6

8

10

#### To display elements by index wise:

In [5]:

l**=**["A","B","C"]

x**=**len(l)

**for** i **in** range(x):

print(l[i],"is available at positive index: ",i,"and at negative index: ",i**-**x)

A is available at positive index: 0 and at negative index: -3 B is available at positive index: 1 and at negative index: -2 C is available at positive index: 2 and at negative index: -1

### Important functions of List:

#### What is the difference between function and method?

In Python you can use both these terms interchangeably.

#### -Function:

Function by default considered as method also.

If a function is declaring outside a class is called as function.

#### - Method :

If you are declaring a function inside a class is called as a method.

In other words, if you are calling any function with object reference is called as method.

### Note:

Python is both functional oriented as well as object oriented programming language.

### To get information about list:

#### len():

returns the number of elements present in the list

In [7]:

n**=**[10,20,30,40]

print(len(n))

4

#### count():

It returns the number of occurrences of specified item in the list.

n**=**[1,2,2,2,2,3,3]

print(n.count(1)) print(n.count(2)) print(n.count(3)) print(n.count(4))

1

4

2

0

#### index() function:

returns the index of first occurrence of the specified item.

|  |  |  |
| --- | --- | --- |
| In [9]: |  | |
| n**=**[1,2,2,2,2,3,3] |  |  |
| print(n.index(1)) | *#* | *0* |
| print(n.index(2)) | *#* | *1* |
| print(n.index(3)) | *#* | *5* |
| print(n.index(4)) |  |  |
| 0 |  |  |
| 1 |  |  |
| 5 |  |  |

#### ---------------------------------------------------------------------------

**ValueError** Traceback (most recent call last)

**<ipython-input-9-6eef43f6d8a4>** in <module>

1. print**(**n**.**index**(2)) # 1**
2. print**(**n**.**index**(3)) # 5**

**----> 5** print**(**n**.**index**(4))**

**ValueError**: 4 is not in list

#### Note:

If the specified element not present in the list then we will get **ValueError**.

Hence before index() method we have to check whether item present in the list or not by using **in** operator.

In [10]:

print( 4 **in** n) *#False*

False

#### Eg:

l **=** [10,20,30,40,10,20,10,10]

target **=** int(input('Enter value to search : '))

**if** target **in** l:

print(target,'available and its first occurrence is at ',l.index(target))

**else**:

print(target,' is not available')b

Enter value to search : 50

50 is not available

In [15]:

l **=** [10,20,30,40,10,20,10,10]

target **=** int(input('Enter value to search : '))

**if** target **in** l:

print(target,'available and its first occurrence is at ',l.index(target))

**else**:

print(target,' is not available')

Enter value to search : 20

20 available and its first occurrence is at 1

In [16]:

l **=** [10,20,30,40,10,20,10,10]

target **=** int(input('Enter value to search : '))

**if** target **in** l:

print(target,'available and its first occurrence is at ',l.index(target))

**else**:

print(target,' is not available')

Enter value to search : 10

10 available and its first occurrence is at 0

### Manipulating Elements of List:

#### append() function:

We can use append() function to add item at the end of the list.

By using this append function, we always add an element at last position.

In [11]:

list**=**[]

list.append("A")

list.append("B")

list.append("C") print(list)

['A', 'B', 'C']

#### Eg: To add all elements to list upto 100 which are divisible by 10

In [17]:

list**=**[]

**for** i **in** range(101):

**if** i**%**10**==**0:

list.append(i) print(list)

[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

#### Another Way:

In [19]:

list**=** []

**for** i **in** range(0,101,10): list.append(i)

print(list)

[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

#### insert() function:

To insert item at specified index position.

In [18]:

n**=**[1,2,3,4,5]

n.insert(1,888) print(n)

[1, 888, 2, 3, 4, 5]

In [23]:

n**=**[1,2,3,4,5]

n.insert(10,777) n.insert(**-**10,999) print(n)

print(n.index(777)) print(n.index(999))

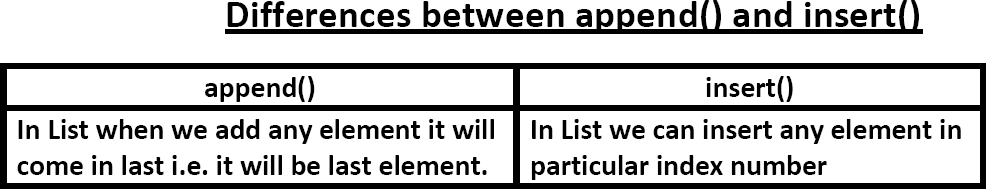
[999, 1, 2, 3, 4, 5, 777]

6

0

#### Note:

If the specified index is greater than max index then element will be inserted at last position. If the specified index is smaller than min index then element will be inserted at first position.



#### extend() function:

To add all items of one list to another list,we use **extend()** method.

#### Eg: l1.extend(l2)

all items present in l2 will be added to l1

#### Eg 1:

In [24]:

order1**=**["Chicken","Mutton","Fish"] order2**=**["RC","KF","FO"]

order1.extend(order2) print(order1)

print(order2)

['Chicken', 'Mutton', 'Fish', 'RC', 'KF', 'FO'] ['RC', 'KF', 'FO']

In [26]:

order1**=**["Chicken","Mutton","Fish"] order2**=**["RC","KF","FO"]

order3 **=** order1 **+** order2 print(order1)

print(order2) print(order3)

['Chicken', 'Mutton', 'Fish']

['RC', 'KF', 'FO']

['Chicken', 'Mutton', 'Fish', 'RC', 'KF', 'FO']

#### Eg 2:

In [1]:

l1 **=** [10,20,30]

l2 **=** [40,50,60]

l1.extend(l2) print(l1)

[10, 20, 30, 40, 50, 60]

#### Eg 3:

In [2]:

order**=**["Chicken","Mutton","Fish"]

order.extend("Mushroom")

print(order)

*# It adds every character as a single element to the list*

['Chicken', 'Mutton', 'Fish', 'M', 'u', 's', 'h', 'r', 'o', 'o', 'm']

#### Explanation :

Here, 'Mushroom' is a string type, in this string 8 elements are there. These elements are added seperately.

In [3]:

order**=**["Chicken","Mutton","Fish"]

order.append("Mushroom")

print(order)

*# It adds this string as a single element to the list*

['Chicken', 'Mutton', 'Fish', 'Mushroom']

## Date: 03-05-2020 Day 3

#### remove() function:

We can use this function to remove specified item from the list.

If the item present multiple times then only first occurrence will be removed.

In [4]:

n**=**[10,20,10,30]

n.remove(10) print(n)

[20, 10, 30]

If the specified item not present in list then we will get **ValueError**

n**=**[10,20,10,30]

n.remove(40) print(n)

#### ---------------------------------------------------------------------------

**ValueError** Traceback (most recent call last)

**<ipython-input-5-75e98f1b4fac>** in <module> 1 n**=[10,20,10,30]**

**----> 2** n**.**remove**(40)**

3 print**(**n**)**

**ValueError**: list.remove(x): x not in list

#### Note:

Hence before using remove() method first we have to check specified element present in the list or not by using in operator.

In [6]:

l1**=** [10,20,30,40,50,60,70]

x **=** int(input('Enter the element to be removed : '))

**if** x **in** l1:

l1.remove(x)

print('Element removed Successfully ') print(l1)

**else**:

print('Specified element is not available ')

Enter the element to be removed : 10 Element removed Successfully

[20, 30, 40, 50, 60, 70]

In [7]:

l1**=** [10,20,30,40,50,60,70]

x **=** int(input('Enter the element to be removed : '))

**if** x **in** l1:

l1.remove(x)

print('Element removed Successfully ') print(l1)

**else**:

print('Specified element is not available ')

Enter the element to be removed : 80 Specified element is not available

#### pop() function:

It removes and returns the last element of the list.

This is only function which manipulates list and returns some element.

#### Eg:

n**=**[10,20,30,40]

print(n.pop())

print(n.pop()) print(n)

40

30

[10, 20]

If the list is empty then pop() function raises IndexError

In [9]:

n**=**[]

print(n.pop())

#### ---------------------------------------------------------------------------

**IndexError** Traceback (most recent call last)

**<ipython-input-9-5146d826acc3>** in <module> 1 n**=[]**

**----> 2** print**(**n**.**pop**())**

**IndexError**: pop from empty list

#### Note:

* + 1. pop() is the only function which manipulates the list and returns some value
    2. In general we can use append() and pop() functions to implement stack datastructure by using list,which follows LIFO(Last In First Out) order.
    3. In general we can use pop() function to remove last element of the list. But we can use to remove elements based on index.

We can use pop() function in following ways:

**n.pop(index)**===>To remove and return element present at specified index.

**n.pop()**==>To remove and return last element of the list

In [11]:

n**=**[10,20,30,40,50,60]

print(n.pop()) *#60*

print(n.pop(1)) *#20*

print(n.pop(10)) *# IndexError: pop index out of range*

60

20

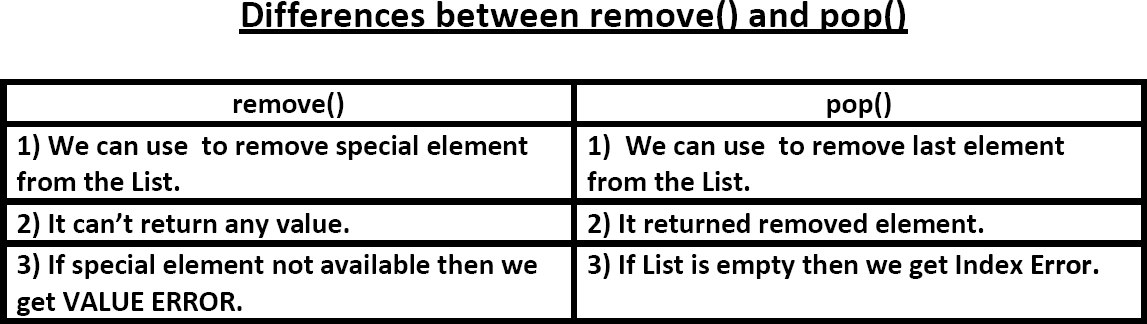
#### ---------------------------------------------------------------------------

**IndexError** Traceback (most recent call last)

**<ipython-input-11-c0a703a9cc2f>** in <module>

1. print**(**n**.**pop**()) #60**
2. print**(**n**.**pop**(1)) #20**

**----> 4** print**(**n**.**pop**(10)) # IndexError: pop index out of range IndexError**: pop index out of range



#### Note:

List objects are dynamic. i.e based on our requirement we can increase and decrease the size.

append(),insert() ,extend() ===>for increasing the size/growable nature

remove(), pop() ======>for decreasing the size /shrinking nature

1. **Ordering elements of List:**

#### reverse():

We can use to reverse() order of elements of list.

In [12]:

n**=**[10,20,30,40]

n.reverse() print(n)

[40, 30, 20, 10]

#### sort() function:

In list by default insertion order is preserved. If you want to sort the elements of list according to default natural sorting order then we should go for sort() method.

-For numbers ==> default natural sorting order is Ascending Order

-For Strings ==> default natural sorting order is Alphabetical Order

#### Eg 1:

In [13]:

n**=**[20,5,15,10,0]

n.sort() print(n)

[0, 5, 10, 15, 20]

#### Eg 2:

In [14]:

s**=**["Dog","Banana","Cat","Apple"] s.sort()

print(s)

['Apple', 'Banana', 'Cat', 'Dog']

In [17]:

s**=**["Dog","Banana","Cat","apple"]

s.sort() print(s)

*# Unicode values are used during comparison of alp*

['Banana', 'Cat', 'Dog', 'apple']

#### Note:

To use sort() function, **compulsory list should contain only homogeneous elements**, otherwise we will get **TypeError.**

#### Eg 3:

n**=**[20,10,"A","B"]

n.sort() print(n)

**---------------------------------------------------------------------------**

**TypeError** Traceback (most recent call last)

**<ipython-input-15-41c38805e086>** in <module> 1 n**=[20,10,"A","B"]**

**----> 2** n**.**sort**()**

3 print**(**n**)**

**TypeError**: '<' not supported between instances of 'str' and 'int'

#### Note:

In Python 2 if List contains both numbers and Strings then sort() function first sort numbers followed by strings.

In [16]:

n**=**[20,"B",10,"A"]

n.sort() print(n)

*# [10,20,'A','B'] It is valid in Python 2, but in Python 3 it is inv*

#### ---------------------------------------------------------------------------

**TypeError** Traceback (most recent call last)

**<ipython-input-16-bda452934197>** in <module> 1 n**=[20,"B",10,"A"]**

**----> 2** n**.**sort**()**

#### 3 print(n) # [10,20,'A','B'] But in Python 3 it is inval

**id.**

**TypeError**: '<' not supported between instances of 'str' and 'int'

#### Eg 4:

In [19]:

n**=**['20',"B",'10',"A"]

n.sort() print(n)

['10', '20', 'A', 'B']

#### How to sort the elements of list in reverse of default natural sorting order: One Simple Way

In [25]:

n**=**[40,10,30,20]

n.sort() n.reverse() print(n)

[40, 30, 20, 10]

#### Alternate Way :

We can sort according to reverse of default natural sorting order by using **reverse = True** argument.

In [27]:

n**=**[40,10,30,20]

n.sort() print(n)

*#[10,20,30,40]*

n.sort(reverse**=True**)

print(n) *#[40,30,20,10]*

n.sort(reverse**=False**)

print(n) *#[10,20,30,40]*

|  |  |  |  |
| --- | --- | --- | --- |
| [10, | 20, | 30, | 40] |
| [40, | 30, | 20, | 10] |
| [10, | 20, | 30, | 40] |

In [28]:

s**=**["Dog","Banana","Cat","Apple"]

s.sort(reverse**= True**) print(s)

*# reverse of Alphabetical order*

['Dog', 'Cat', 'Banana', 'Apple']

### 6 . Aliasing and Cloning of List objects:

The process of giving another reference variable to the existing list is called aliasing.

In [29]:

x**=**[10,20,30,40]

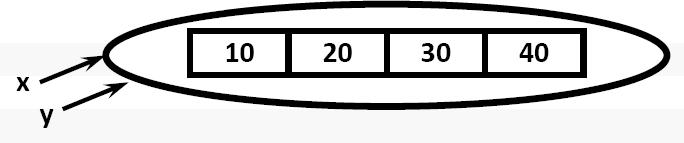
y**=**x

print(id(x))

print(id(y))

1709842944648

1709842944648



The problem in this approach is by using one reference variable if we are changing content,then those changes will be reflected to the other reference variable.

In [30]:

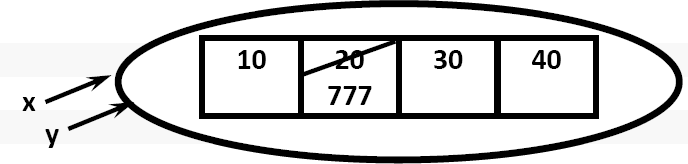
x**=**[10,20,30,40]

y**=**x y[1]**=**777

print(x)

*#[10,777,30,40]*

[10, 777, 30, 40]



To overcome this problem we should go for **cloning**.

**Cloning :**The process of creating exactly duplicate independent object is called cloning. We can implement cloning by using the following ways:

1. slice operator
2. copy() function

#### By using slice operator:

In [31]:

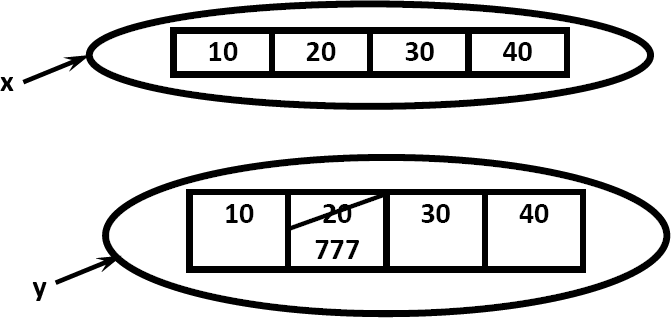
x**=**[10,20,30,40]

y**=**x[:] y[1]**=**777

print(x) *#[10,20,30,40]*

print(y) *#[10,777,30,40]*

|  |  |  |
| --- | --- | --- |
| [10, | 20, | 30, 40] |
| [10, | 777, | 30, 40] |



#### By using copy() function:

In [32]:

x**=**[10,20,30,40]

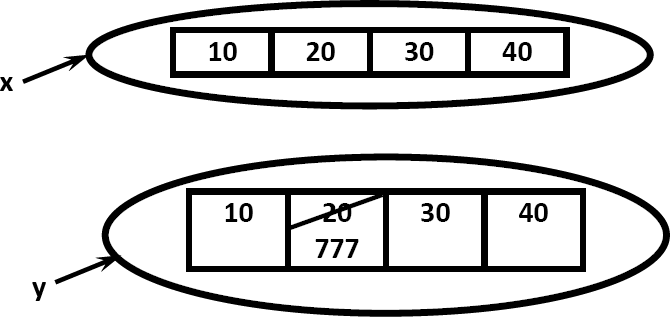
y**=**x.copy() y[1]**=**777

print(x) *# [10,20,30,40]*

print(y) *# [10,777,30,40]*

[10, 20, 30, 40]

[10, 777, 30, 40]



#### Q. What is the difference between = operator and copy() function?

= operator meant for aliasing copy() function meant for cloning

### 7. Using Mathematical operators for List Objects:

We can use + and \* operators for List objects.

#### Concatenation operator(+):

We can use + to concatenate 2 lists into a single list.

In [38]:

a**=**[10,20,30]

b**=**[40,50,60]

c**=**a**+**b

print(' a : ',a)

print(' b : ',b)

print(' c : ',c)

*# concatenation*

|  |  |  |  |
| --- | --- | --- | --- |
| a : | [10, | 20, | 30] |
| b : | [40, | 50, | 60] |
| c : | [10, | 20, | 30, 40, 50, 60] |

In [41]:

a**=**[10,20,30]

b**=**[40,50,60]

c**=**a.extend(b) *# extend() methodwon't return anything. it adds the content of 'b' to 'a'.*

print(' a : ',a)

print(' b : ',b)

print(' c : ',c)

a : [10, 20, 30, 40, 50, 60]

b : [40, 50, 60]

c : None

In [ ]:

#### Note:

To use + operator compulsory both arguments should be list objects,otherwise we will get TypeError.

#### Eg:

In [42]:

c**=**a**+**40

print(c)

*#TypeError: can only concatenate list (not "int") to list*

#### ---------------------------------------------------------------------------

**TypeError** Traceback (most recent call last)

**<ipython-input-42-aae8ea9a4fe9>** in <module>

#### ----> 1 c=a+40 #TypeError: can only concatenate list (not "int") to l ist

2 print**(**c**)**

**TypeError**: can only concatenate list (not "int") to list

In [43]:

c**=**a**+**[40]

print(c)

*#valid*

[10, 20, 30, 40, 50, 60, 40]

#### Repetition Operator(\*):

We can use repetition operator \* to repeat elements of list specified number of times

In [44]:

x**=**[10,20,30]

y**=**x**\***3

print(y)

*#[10,20,30,10,20,30,10,20,30]*

[10, 20, 30, 10, 20, 30, 10, 20, 30]

### Comparing List objects

We can use comparison operators for List objects.

#### Eg :

In [45]:

x**=**["Dog","Cat","Rat"]

y**=**["Dog","Cat","Rat"]

z**=**["DOG","CAT","RAT"]

print(x**==**y) print(x**==**z) print(x **!=** z)

*#True*

*#False #True*

True False True

#### Note:

Whenever we are using comparison operators(==,!=) for List objects then the following should be considered:

* 1. The number of elements
  2. The order of elements
  3. The content of elements (case sensitive)

#### Note:

When ever we are using relatational operators(<,<=,>,>=) between List objects,only first element comparison will be performed.

#### Eg :

In [46]:

x**=**[50,20,30]

y**=**[40,50,60,100,200]

print(x**>**y) *#True*

print(x**>=**y) *#True*

print(x**<**y) *#False*

print(x**<=**y) *#False*

True True False False

#### Eg :

x**=**["Dog","Cat","Rat"]

y**=**["Rat","Cat","Dog"] print(x**>**y) *#False*

print(x**>=**y) *#False*

print(x**<**y) *#True*

print(x**<=**y) *#True*

False False True True

In [48]:

x**=**["Dog","Cat","Rat"]

y**=**["Dat","Cat","Dog"] print(x**>**y)

True

### Membership operators:

We can check whether element is a member of the list or not by using memebership operators.

* 1. in operator
  2. not in operator

In [49]:

n**=**[10,20,30,40]

print (10 **in** n)

print (10 **not in** n) print (100 **not in** n) print (50 **in** n)

print (50 **not in** n)

True False True False True

### clear() function:

We can use clear() function to remove all elements of List.

n**=**[10,20,30,40]

print(n) n.clear() print(n)

[10, 20, 30, 40] []

## Date: 04-05-2020 Day 4

### Nested Lists:

Sometimes we can take one list inside another list. Such type of lists are called nested lists.

Consider the follwoing example:

In [1]:

n**=**[10,20,[30,40]]

print(n)

print(n[0])

print(n[2])

print(n[2][0])

print(n[2][1])

[10, 20, [30, 40]]

10

[30, 40]

30

40

#### Note:

We can access nested list elements by using index just like accessing multi dimensional array elements.

#### Nested List as Matrix

In Python we can represent matrix by using nested lists.

In [2]:

n**=**[[10,20,30],[40,50,60],[70,80,90]]

print(n)

print("Elements by Row wise:")

**for** r **in** n:

print(r)

print("Elements by Matrix style:")

**for** i **in** range(len(n)):

**for** j **in** range(len(n[i])):

print(n[i][j],end**=**' ') print()

[[10, 20, 30], [40, 50, 60], [70, 80, 90]]

Elements by Row wise:

|  |  |  |
| --- | --- | --- |
| [10, | 20, | 30] |
| [40, | 50, | 60] |
| [70, | 80, | 90] |

Elements by Matrix style:

|  |  |  |
| --- | --- | --- |
| 10 | 20 | 30 |
| 40 | 50 | 60 |
| 70 | 80 | 90 |

### List Comprehensions

It is very easy and compact way of creating list objects from any iterable objects(like list,tuple,dictionary,range etc) based on some condition.

#### Syntax:

**list=[expression for item in list if condition]**

Consider an example, If you want to store squares of numbers form 1 to 10 in a list,

In [12]:

l1**=**[]

**for** x **in** range(1,11): l1.append(x**\***x)

print(l1)

[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

In the above case, the program consisting 4 lines of code. Now for the same purpose we will write the following code in more concised way.

In [13]:

l1 **=** [x**\***x **for** x **in** range(1,21)]

l2 **=** [x **for** x **in** l1 **if** x **%** 2 **==** 0] print(l1)

print(l2)

[1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 32

4, 361, 400]

[4, 16, 36, 64, 100, 144, 196, 256, 324, 400]

#### Few more examples on List comprehensions :

In [10]:

l1 **=** [x**\***x **for** x **in** range(1,11)] print(l1)

[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

In [14]:

l **=**[2**\*\***x **for** x **in** range(1,11)] print(l)

[2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]

In [8]:

l **=** [x **for** x **in** range(1,11) **if** x**%**2**==**0] print(l)

[2, 4, 6, 8, 10]

In [15]:

l **=** [x **for** x **in** range(1,11) **if** x**%**2**==**1] print(l)

[1, 3, 5, 7, 9]

In [17]:

l **=** [x**\*\***2 **for** x **in** range(1,11) **if** (x**\*\***2)**%**2**==**1] print(l)

[1, 9, 25, 49, 81]

In [16]:

words**=**["Balaiah","Nag","Venkatesh","Chiranjeevi"] l**=**[w[0] **for** w **in** words]

print(l)

['B', 'N', 'V', 'C']

In [21]:

words**=**["Balaiah","Nag","Venkatesh","Chiranjeevi"] l**=**[w **for** w **in** words **if** len(w)**>**6]

print(l)

['Balaiah', 'Venkatesh', 'Chiranjeevi']

In [18]:

num1**=**[10,20,30,40]

num2**=**[30,40,50,60]

num3**=**[ i **for** i **in** num1 **if** i **not in** num2] print(num3)

*#[10,20]*

[10, 20]

In [22]:

words**=**"the quick brown fox jumps over the lazy dog".split()

print(words)

l**=**[[w.upper(),len(w)] **for** w **in** words] print(l)

*# All 26 alphabets used in t*

['the', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']

[['THE', 3], ['QUICK', 5], ['BROWN', 5], ['FOX', 3], ['JUMPS', 5], ['OVER',

4], ['THE', 3], ['LAZY', 4], ['DOG', 3]]

#### Example Program

**Q. Write a program to display unique vowels present in the given word.**

In [23]:

vowels**=**['a','e','i','o','u']

word**=**input("Enter the word to search for vowels: ") found**=**[]

**for** letter **in** word:

**if** letter **in** vowels:

**if** letter **not in** found: found.append(letter)

print(found)

print("The number of different vowels present in",word,"is",len(found))

Enter the word to search for vowels: Quality Education is useful ['u', 'a', 'i', 'o', 'e']

The number of different vowels present in Quality Education is useful is 5

Suppose if you want lower case and upper case vowels, what you can do is as follows:

In [2]:

vowels**=**['a','e','i','o','u','A','E','I','O','U']

word**=**input("Enter the word to search for vowels: ") found**=**[]

**for** letter **in** word:

**if** letter **in** vowels:

**if** letter **not in** found: found.append(letter)

print(found)

print("The number of different vowels present in",word,"is",len(found))

Enter the word to search for vowels: karthi Abc ['a', 'i', 'A']

The number of different vowels present in karthi Abc is 3

See the above code in another simplified way:

In [3]:

vowels**=**['a','e','i','o','u']

word**=**input("Enter the word to search for vowels: ") found**=**[]

**for** letter **in** word:

**if** letter.lower() **in** vowels:

**if** letter.lower() **not in** found: found.append(letter)

print(found)

print("The number of different vowels present in",word,"is",len(found))

Enter the word to search for vowels: KARTHIKEYA ['A', 'I', 'E', 'A']

The number of different vowels present in KARTHIKEYA is 4

In [4]:

vowels**=**['a','e','i','o','u']

word**=**input("Enter the word to search for vowels: ") found**=**[]

**for** letter **in** word:

**if** letter.lower() **in** vowels:

**if** letter.lower() **not in** found: found.append(letter.lower())

print(found)

print("The number of different vowels present in",word,"is",len(found))

Enter the word to search for vowels: KARTHIKEYA ['a', 'i', 'e']

The number of different vowels present in KARTHIKEYA is 3