LISTS

Like a string, a list is a sequence of values. In a string, the values are characters; in a list, they can be any type. The values in a list are called elements or sometimes items.

There are several ways to create a new list; the simplest is to enclose the elements in square brackets ([and]):

In [15]:

```
list1=[1, 2,3,"python", 7.6,90.23,[10,20,30]]
print(type(list1[-1]))
```

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

In [17]:

```
list1=[1, 2, 3, 4,4.5,"string",True,False,[10,20,40]]
print(type(list1))
print(list1)
```

```
<class 'list'>
[1, 2, 3, 4, 4.5, 'string', True, False, [10, 20, 40]]
```

In [ ]:

```
list1=[3,4,5,6] #list of integers
list2=['first', "second", 'prog']  #list of words
```

The first example is a list of four integers. The second is a list of three strings. The elements of a list don't have to be the same type. The following list contains a string, a float, an integer,

In [1]:

```
list3=[45, 4.7,"pro"]
list4=[45,4.7,"pro",[3,2,'po']]   #list1,list2,list3..are list variables
```

A list within another list is nested. A list that contains no elements is called an empty list; you can create one with empty brackets, [].

In [ ]:

```
emp_list=[]   #empty list
```

In [18]:

```
str1="abcde"
#str1[0]='z'
list1=["a","b",4,8,9]
print(list1)
list1[0]="z"
print(list1)
```

```
['a', 'b', 4, 8, 9]
['z', 'b', 4, 8, 9]
```

***Important: Lists are mutable, strings are immutable

In [20]:

```
# s="pyhton"
# s[2]="k"
# print(s[2])
list1=[10,20,30,40]
list1[3]=50
print(list1)
```

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

In [2]:

```
list1=[4,5,6,7]

list1[2]="pro"
print(list1)
```

```
[4, 5, 'pro', 7]
```

In [19]:

```
st1="python"

print(len(st1))

#we get an error  because strings are immutable
```

```
6
```

Accessing the elements from list Index from 0 to last element List indices work the same way as string indices: • Any integer expression can be used as an index. • If you try to read or write an element that does not exist, you get an IndexError. • If an index has a negative value, it counts backward from the end of the list.

In [31]:

```
list1=[4,5,6,7,8]
l=len(list1)
print(list1[l-1])
print(list1[-l])
list1[-1]=[10,20,20]
print(list1)
# print(len(list1))
# print(list1[4])
# print(list1[-1])
```

```
8
4
[4, 5, 6, 7, [10, 20, 20]]
```

In [21]:

```

```

```
10
```

In operator on list: takes an element and list element (In operator) list

In [25]:

```
prog_lang=["java", "python", "C", "Cplus"]
c="java"
b="Java"
print("java" in prog_lang)
```

```
True
```

Traversing a List using for loop iterating over the elements of the list using for loop iterating over the length of the list using while loop for ele in [a,b,c,d] for ele in list1 for i in range(len(list1))

In [26]:

```
list1=[1,2,3,4,5,6]
for ele in list1:
    print(ele)
```

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

In [36]:

```
list1=[1,2,3,4,5,6]
l=len(list1)
for i in range(l):
    print(list1[i])
```

```
1
2
3
4
5
```

In [37]:

```
list1=[1,2,3,4,5,6]
l=len(list1)
i=0
while(i<l):
    print(list1[i])
    i=i+1
```

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

List Operations The + operator concatenates lists: The * operator repeats a list a given number of times:

In [30]:

```
str1="python"
str2="program"
print(str1+str2)
print(str1*3)
```

```
pythonprogram
pythonpythonpython
```

In [33]:

```
list1=[1,2,3,4]
list2=[4,5,6,7]
list3=list1+list2+list2
list4=list1*4
print(list4)
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
<ipython-input-33-0f480b85c53b> in <module>
      1 list1=[1,2,3,4]
      2 list2=[4,5,6,7]
----> 3 list3=list1-list2+list2
      4 list4=list1*4
      5 print(list4)

TypeError: unsupported operand type(s) for -: 'list' and 'list'
```

List Slices The slice operator also works on lists: If you omit the first index, the slice starts at the beginning. If you omit the second, the slice goes to the end. So if you omit both, the slice is a copy of the whole list:

In [41]:

```
list1=["A","B","C","D","E","D","U","O",9]
#l=len(list1)#0    to l-1  #-1 to -l   or -l to -1
print(list1[0:4])
list2=list1[-1:]
print(list2)

# print(list1[0:4])
# print(list1[:4])
# print(list1[3:l])
# print(list1[3:])
# print(list1[3:8])
```

```
['A', 'B', 'C', 'D']
[]
```

A slice operator on the left side of an assignment can update multiple elements:

t = ['a', 'b', 'c', 'd', 'e', 'f'] t[1:3] = ['x', 'y']

In [46]:

```
#extended slicing
list1=[1,2,3,4,5,6,7,8,9]
print(list1[-1:-7-1:-1])
```

```
[9, 8, 7, 6, 5, 4, 3]
```

In [ ]:

```
#print the list in reverse order
```

In [4]:

```
#How to read a list from keyboard
l=input("enter the list").split(" ")
#l=list(map(int,l))
print(l)
```

```
enter the list1 2 3 4
[1, 2, 3, 4]
```

In [65]:

```
list1=[0,1,2,3,4,5,6,7,8,9]
l=len(list1)
print(list1[3:])
print(list1[-6:-1:])
```

```
[3, 4, 5, 6, 7, 8, 9]
[4, 5, 6, 7, 8, 9]
```

In [71]:

```
list1=[0,1,2,3,4,5,6,7,8,9]
print(list1[9:1:-2],list1[1:9])
```

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