In [ ]:
```
#Agenda of the Day:
                    1. Data Structures in Python
                        Problem Solving
```

In [ ]:
```
#Introduction about DS:
       Data Structure is way of organize our data in such a way that
      enables you to store collections of data,relate them and perform
      operations on them accordingly.

        (i) Organizing the data.
        (ii) managing the data.
        (iii) storing the data.
        (iv) accessing the data.

#Types of Data Structures:
             1. Lists
             2. Tuples
             3. Sets
             4. Dictionaries.
```

In [ ]:
```
#Lists:
    Lists are used to store the data of different types in a
    Sequential manner.
    #Key points:
        1. Ordered collection of different type of elements.
        2. Its mutable (Changable)
        3. elements in list are addressed with index value.
              list = [1, 2, 3, 4, 5, 6 ,7, 8]
              index = 0  1  2  3  4  5  6  7

            Index starts with 0
        4. we use square brackets representing the lists.[]
        5. Lists have more built-in functions.
```

In [3]:
```
#How to create lists?
li = []  #create empty list
print(type(li))
li1 = list()    #Way-2
print(type(li1))
```

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

In [7]:
```
#How to assign vales into lists:
li = [1,2,3,4,5,6,7,8,9,10]   #list with interger values
li1 = ["a","b","c","d","Python","Monday"]  #List of strings
li2 = [True,10,30,50,"Data Science","Machine Learning",4.5,8.9]
                              #list with Different types
print(li)
print(li1)
print(li2)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
['a', 'b', 'c', 'd', 'Python', 'Monday']
[True, 10, 30, 50, 'Data Science', 'Machine Learning', 4.5, 8.9]
```

In [15]:
```
#Accessing the elements from the lists:  #n = 7 elements
li = [3,6,9,"Today","Coding",50.3,90.5]  #index : 0-n-1 0-6
li[0]
li[3]                  #for getting only one value
li[6]
li[-1]
li[-3]
li[-6]
li[-7]=li[0]
```

Out[15]: 6

In [25]:
```
#How to access the group of values: (slicing : operator)
li = [3,6,9,"Today","Coding",50.3,90.5]
li1 = ["a","b","c","d","Python","Monday"]
print(li[0:5])                     #Positive index
print(li[::-1])       #reverse the given list of elements
print(li[0:])
print(li[-5:-1])          #range of negative index
print(li1[1:5])
```

```
[3, 6, 9, 'Today', 'Coding']
[90.5, 50.3, 'Coding', 'Today', 9, 6, 3]
[3, 6, 9, 'Today', 'Coding', 50.3, 90.5]
[9, 'Today', 'Coding', 50.3]
['b', 'c', 'd', 'Python']
```

In [35]:
```
#Adding elements to lists? (append(),insert(),extend())
#append(): (its adds the elemets into the its as group of elements)
li = [10,20,30,40,50,60]
print("initial list:",li)
li.append(70)                       #individual elements adding
li.append(80)
li.append("coding")
li.append([1000,"Data Science",90.1])
print("appeneded list:",li)
```

```
initial list: [10, 20, 30, 40, 50, 60]
appeneded list: [10, 20, 30, 40, 50, 60, 70, 80, 'coding', [1000, 'Data Scienc
e', 90.1]]
```

In [40]:
```
#Extend() operation:
#(this Functions adds the elements one by one into list)
li = [10,20,30,40,50,60]
print("initial list:",li)
print(len(li))
li.extend([1000,"Data Science",90.1])
li.extend([30.5,80.6,True])
print("extended list:",li)
print(len(li))
```

```
initial list: [10, 20, 30, 40, 50, 60]
6
extended list: [10, 20, 30, 40, 50, 60, 1000, 'Data Science', 90.1, 30.5, 80.6,
True]
12
```

In [55]:
```
#insert(): #syntax:  li.insert(index,value)

li = [10,20,30,40,50,60]
print("initial list:",li)
li.insert(0,"kits")    #insert new value at index 0
li.insert(3,[90,100,500])
li.insert(6,["hello","Kits",1000.6])
li.insert(1,6000)
print("updated list:",li)
```

```
initial list: [10, 20, 30, 40, 50, 60]
updated list: ['kits', 6000, 10, 20, [90, 100, 500], 30, 40, ['hello', 'Kits',
1000.6], 50, 60]
```

In [ ]:
```
#How to explore and check all built-in
#functions of any function,module,datastructure,
#class,package.
```

In [58]:
```
print(dir(str),end=" ")
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewa
rgs__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__l
e__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce
__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__siz
eof__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'cou
nt', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'inde
x', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'i
slower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join',
'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'replace', 'rfind', 'rind
ex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startsw
ith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
```

In [56]:
```
print(dir(list),end=" ")
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir_
_', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem
__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__init_subclass
__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__',
'__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setat
tr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'c
lear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'revers
e', 'sort']
```

In [66]:
```
#copy and clear():
li = [1,2,3,"A","B","C"]
li1 = []              #empty list
li1=li.copy()
print(li)
print(li1)
li.clear()
print(li1)
print(li)
```

```
[1, 2, 3, 'A', 'B', 'C']
[1, 2, 3, 'A', 'B', 'C']
[1, 2, 3, 'A', 'B', 'C']
[]
```

In [81]:
```
#index() and count() :            #stackoverflow
li = [50,100,150,"Cse","AI","IT",7500,90.8,100,50,100,"AI","IT"]
len(li)
#print(li.index("AI"))
#print(li.index(7500))
#print(li.index(150))
#print(li[3])
#print(li.index("Cse"))
print(li.count(100))
print(li.count(50))
print(li.count(150))
print(li.count("AI"))
```

```
3
2
1
2
```

In [99]:
```
#sorted() and deleting the elements from the lists:
li = [100,90,80,70,60,50,40,30,20,10]
#print(sorted(li))
print("initial list",li)
#li.remove(50)   #this fuction deletes individual element
#li.remove(30)
#li.remove(100)
li.remove
li.pop()      #this function deletes the elements from right to left.
li.pop()
li.pop()
li.pop()
li.pop()
li.pop()
li.pop()
li.pop()
li.pop()
li.pop()
print("after deletion:",li)
```

```
initial list [100, 90, 80, 70, 60, 50, 40, 30, 20, 10]
after deletion: []
```

In [102]:
```
#li.remove(x)  #its deletes one element at time.
li = [[100,90],80,70,60,50,40,30,20,10]
li.remove([100,90])
li
```

Out[102]:
```
[80, 70, 60, 50, 40, 30, 20, 10]
```

In [110]:
```
li = [[100,90],80,70,60,50,40,30,20,10]
li.pop()
li.pop()
li.pop()
li.pop(3)
li.pop(0)
print(li)
```

```
[80, 70, 50, 40]
```

In [112]:
```
#reverse() its reverse the all elements from given input list
li = [[100,90],80,70,60,50,40,30,20,10]
li.reverse()
li
```

Out[112]:
```
[10, 20, 30, 40, 50, 60, 70, 80, [100, 90]]
```

In [113]:
```
#we can access the elements from the list by iterating

li = [[100,90],80,70,60,50,40,30,20,10]
for val in li:
    print(val,end=" ")
```

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

In [135]:
```
#example:
li = [12,16,19,24,50,25,91,100]
for i in li:
    if i%2==0:
        print(i,end=" ")
```

```
12 16 24 50 100
```

In [137]:
```
#given range (1,100):
evennum = []
oddnum = []
for i in range(1,101):
    if i%2==0:
        evennum.append(i)
    else:
        oddnum.append(i)
print("even list:",evennum)
print("odd num list:",oddnum)
```

```
even list: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36,
38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76,
78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100]
odd num list: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 3
5, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73,
75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99]
```

In [ ]: