In [ ]:

```
for loop and while loop
for -->if we know number of iterations example : arrays
while -->when we know the condition Example: linkedlist
```

In [2]:

```
"""classes and object
class --> blueprint of an object
object --> an instnace of a class"""
class className:
    def __init__(self,x):
        self.x=x

    def m1(self):
        print(self.x)
C=className(10)
C.m1()
```

```
10
```

In [ ]:

```
#Data Structure:is a way of storing the data so that we can use that data in an  effiicent
manner for our furthur use.
Types of Datastructure:
    Two types of Data Structure:
        1.Primitive -->integer , float, complex,.....
        2.Non primitive ---> Based on Implementation --> Physical (Arrays and Linkedlist) -->Logical(
        Stack,Queue,Trees and Graph)
                        --> Based on Storage -->Linear Ds(Arrays,lINKEDLIST,STACK,QUeue)
            =                                -->Non Linear Ds(Trees and Graph)

    Abstract Datatypes --> Entities that have definitions of data and operations but donot
    have implementation.
    Stack--> Push,pop,peek -->array,linkedlist
    Queue -->enqueue,Dequeue-->array,linkedlist
```

In [ ]:

```
Array --> Collection of similar datatypes with contigous memeory location.-->{10,20,30,40}
List--> list is also a collection of disimilar data --->[]
```

In [ ]:

```
#Basic operation of any Data struture:
1.Insertion --> inserting something in array(start,end,pos)
2.Deletioon --> deletiing something from array(start,end,pos)
3.Traversal --> Visitng eacha nd every elemnt of an array
```

In [2]:

```
#Implementation of Different operations in array

class arrays:
    def __init__(self):
        self.array=[]
    def insertion_at_begin(self,data):
        self.array.insert(0,data)
        return self.array
    def insertion_at_end(self,data):
        self.array.append(data)
        return self.array
    def insertion_at_pos(self,data,pos):
        if pos>len(self.array):
            self.array
        self.array.insert(pos,data)
        return self.array
    def deletion_at_begin(self):
        if len(self.array)<1:
            return self.array
        self.array.pop(0)
        return self.array
    def deletion_at_last(self):
        if len(self.array)<1:
            return self.array
        self.array.pop()
        return self.array
    def deletion_at_pos(self,pos):
        if len(self.array)<1 and pos>len(self.array):
            return self.array
        self.array.pop(pos)
        return self.array
    def traversal(self):
        for i in range(len(self.array)):
            print(self.array[i])
s=arrays()
s.insertion_at_begin(10)
s.traversal()
print("*************")
s.insertion_at_begin(20)
s.traversal()
print("*************")
s.insertion_at_begin(30)
s.traversal()
print("*************")
s.insertion_at_pos(50,0)
s.traversal()
print("*************")
s.insertion_at_pos(60,1)
s.traversal()
print("*************")
s.insertion_at_end(500)
s.traversal()
print("*************")
s.deletion_at_begin()
s.traversal()
print("*************")
s.deletion_at_last()
s.traversal()
print("*************")
s.deletion_at_pos(2000)
s.traversal()
```

```
10
*************
20
10
*************
30
20
10
*************
50
30
20
10
*************
50
60
30
20
10
*************
50
60
30
20
10
500
*************
60
30
20
10
500
*************
60
30
20
10
*************
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Input In [2], in <cell line: 60>()
     58 s.traversal()
     59 print("*************")
---> 60 s.deletion_at_pos(2000)
     61 s.traversal()

Input In [2], in arrays.deletion_at_pos(self, pos)
     28 if len(self.array)<1 and pos>len(self.array):
     29     return self.array
---> 30 self.array.pop(pos)
     31 return self.array

IndexError: pop index out of range
```

In [ ]:

```
list vs arrays
1.Arrays stores similar datatype whereas list stores disimilar datatype
2.Arrays are Fixed whwreas lists are dynamic in nature.
3.Arrays stores data at contigous memeory location whereas list is an object



array{5}--> 5 elemnt
list --> dynamic in nature you can chanage size
```

In [20]:

```
x=[10,20,30]
for i in x:
    print(i)
```

```
10
20
30
```

In [22]:

```
import numpy
sample_array=array.array("i",[10,20,30])
for i in sample_array:
    print(i)
```

```
10
20
30
```

In [ ]:

```
Applications of Array:
    In Computer Programming
    1:implementation fo tress and graphs
    2:Used for matrix and mathematical calculation
    3.CPU Scheduling
    In real World:
        1.Page of books
        2.Dictionary book
        3.Online Booking
```

In [1]:

```
#Searching->[10,20,30,40,50]  ele=50   return -1
#Linear Searching
#Given an array arr[] of N elements,
#the task is to write a function to search a given element x in arr[].
x=[10,20,30,40,50,60]
key=40
for i in range(len(x)):
    if x[i]==key:
        print("Element found at",i)
        break

else:
    print("Element not present")
```

```
0
1
2
3
4
5
Element not present
```

In [ ]: