**STACK IMPLEMENTATION**

**Stack using Linked List:**

**class Node(object):**

def \_\_init\_\_(self, value):

self.value=value

self.next=None

**class StackLL**:

def \_\_init\_\_(self):

self.top=None

self.bot=self.top

self.len=0

|  |  |
| --- | --- |
| **PUSH:**  def push(self,value):  new\_node=Node(value)  if(self.len==0):  self.top=new\_node  self.bot=self.top  else:  new\_node.next=self.top  self.top=new\_node  self.len+=1 | **POP:**  def pop(self):  if(self.len==0):  return  elif(self.len==1):  self.top=self.top.next  self.bot=self.top  else:  self.top=self.top.next  self.len-=1 |
| **PEEK:**  def peek(self):  return self.top.value | **PRINT:**  def printStack(self):  st = []  curr\_node=self.top  while(curr\_node!=None):  st.append(curr\_node.value)  curr\_node=curr\_node.next  print(st) |

**Stack using Array:**

class StackArr(object):

def \_\_init\_\_(self):

self.arr=[]

|  |  |
| --- | --- |
| **PUSH:**  def push(self,value):  self.arr.append(value) | **POP:**  def aPop(self):  return self.arr.pop() |
| **PEEK:**  def peek(self):  return self.arr[len(self.arr)-1] | **Is Stack Empty:**  def isEmpty(self):  if(len(self.arr)==0):  return true  else:  return false |