#Task\_01

class Node:

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

self.value=value

self.next=next

class MyList:

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

self.head = None

self.next=next

self.x=x

#Task\_02

class Node:

def \_\_init\_\_(self, data=None, next=None):

self.data = data

self.next = next

#---Constructor----(a)------

class MyList:

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

self.a=a

self.head = None

#-----02--------

def showList(self):

if self.head is None:

print("Linked list is empty")

return

n = self.head

string = ''

while n:

string += str(n.data)+' --> ' if n.next else str(n.data)

n = n.next

print(string)

#-------03-------

def isEmpty(self):

if self.head is None:

#print("Linked list is empty")

return True

print("True")

else:

False

print("False")

#------04------

def clear(self):

if self.isEmpty()==True:

print('The list is not empty.')

else:

node=Node(self.head,None)

temp=node.next

node.data=None

node.next=None

node=None

node=temp

#return node

def get\_length(self):

count = 0

itr = self.head

while itr:

count+=1

itr = itr.next

return count

def insert\_at\_begining(self, data):

node = Node(data, self.head)

self.head = node

def insert\_at\_end(self, data):

if self.head is None:

self.head = Node(data, None)

return

itr = self.head

while itr.next:

itr = itr.next

itr.next = Node(data, None)

#------05------

def insert(self, newElement):

self.head = None

for data in newElement:

self.insert\_at\_end(data)

#------06------

def insert\_at(self, newElement, index):

if index<0 or index>self.get\_length():

raise Exception("Invalid Index")

if index==0:

self.insert\_at\_begining(data)

return

count = 0

itr = self.head

while itr:

if count == index - 1:

node = Node(newElement, itr.next)

itr.next = node

break

itr = itr.next

count += 1

#------07------

def remove(self, deletekey):

if deletekey<0 or deletekey>=self.get\_length():

raise Exception("Invalid Index")

if deletekey==0:

self.head = self.head.next

return

count = 0

itr = self.head

while itr:

if count == deletekey - 1:

itr.next = itr.next.next

break

itr = itr.next

count+=1

if \_\_name\_\_ == '\_\_main\_\_':

ll = MyList(5)

ll.insert([1,7,2,8,5])

ll.showList()

ll.isEmpty()

ll.clear()

ll.insert([1,7,2,8,5])

ll.showList()

ll.insert\_at\_end(67)

ll.showList()

ll.insert\_at(2,2)

ll.showList()

ll.remove(3)

ll.showList()

Outputs:(Task\_02):

1 --> 7 --> 2 --> 8 --> 5

False

False

1 --> 7 --> 2 --> 8 --> 5

1 --> 7 --> 2 --> 8 --> 5 --> 67

1 --> 7 --> 2 --> 2 --> 8 --> 5 --> 67

1 --> 7 --> 2 --> 8 --> 5 --> 67

#Task\_03

class Node:

def \_\_init\_\_(self, data=None, next=None):

self.data = data

self.next = next

class MyList:

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

self.head = None

def showList(self):

if self.head is None:

print("Linked list is empty")

return

n = self.head

string = ''

while n:

string += str(n.data)+' --> ' if n.next else str(n.data)

n = n.next

print(string)

def insert\_at\_end(self, data):

if self.head is None:

self.head = Node(data, None)

return

n = self.head

while n.next:

n = n.next

n.next = Node(data, None)

def insert(self, newElement):

self.head = None

for data in newElement:

self.insert\_at\_end(data)

#------01-------

def even\_checker\_List(self):

node = self.head

head\_01 = None

tail\_01 = None

while node != None:

if node.data % 2 == 0:

new\_List = Node(node.data, None)

if head\_01 == None:

head\_01 = new\_List

tail\_01 = new\_List

else:

tail\_01.next = new\_List

tail\_01 = new\_List

node = node.next

head\_02 = head\_01

while head\_02!= None:

print(head\_02.data , end= "--> ")

head\_02 = head\_02.next

print("\n")

return head\_01

#------02-------

def element\_Checker(self,value):

node = self.head

there\_is\_element = False

while node != None:

if node.data == value:

there\_is\_element = True

break

node = node.next

print(there\_is\_element)

#------03-------

def reverse(self):

node = self.head

node\_02 = None

while node != None:

next\_value = node.next

node.next =node\_02

node\_2 = node

node = next\_value

self.head =node\_02

self.showList()

#------04-------

def sort\_List(x):

node=x.head

n=node

while n !=None:

i=n.next

while i !=Nonw:

if n.data>i.data:

n.value.i.value=i.value.n.value

i.i.next

n=n.next

self.showList()

#------05-------

def sum\_counter(self):

node = self.head

full = 0

while node!= None:

total += node.data

node = node.next

print(full)

#------06-------

def K\_time\_rotator(self,side,position):

node = self.head

count = 0

while node!=None:

node += 1

node= node.next

if position<count:

value = self.head

data = value

count\_02 = 0

head = None

tail = None

if side == "right":

value\_02 = self.head

data\_02 = value\_02

count\_03 = 0

head\_02 = None

tail\_02 = None

elif side == "left":

while value.next!=None:

if count\_02 == position-1:

head = value.next

tail = value

value = value.next

count\_02 += 1

value.next = data

tail.next = None

self.head = head

while value\_02.next != None:

if count\_03 == (count-1) - position :

head\_02 = value\_02.next

tail\_02 = value\_02

value\_02 = value\_02.next

count\_03 += 1

value\_02.next = data\_02

tail\_02.next = None

self.head = head\_02

elif count<position:

print("Invalid position")

else:

pass

if count>=position:

self.showList()

else:

pass

if \_\_name\_\_ == '\_\_main\_\_':

ll = MyList()

ll.insert([1,2,5,3,8])

ll.showList()

ll.even\_checker\_List()

ll.showList()

ll.element\_Checker([1,2,5,3,8,7])

ll.showList()

ll.reverse()

ll.showList()

ll.sort\_List()

ll.showList()

ll.sum\_counter()

ll.showList()

ll.K\_time\_rotator("left",3)

ll.showList()

Outputs:(Task\_03):

1 --> 2 --> 5 --> 3 --> 8

2--> 8-->

1 --> 2 --> 5 --> 3 --> 8

False

1 --> 2 --> 5 --> 3 --> 8

Linked list is empty

Linked list is empty

Linked list is empty

0

Linked list is empty

Linked list is empty

1

​