**Single Linked List Implementation**

class Node:

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

self.data = data

self.next = None

class SLL:

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

self.head = None

def insert(self,data):

new\_node = Node(data)

if(self.head is None):

self.head = new\_node

return

current = self.head

while(current.next is not None):

current = current.next

current.next = new\_node

def insertFirst(self,data):

new\_node = Node(data)

new\_node.next = self.head

self.head = new\_node

def insertAtPosition(self,pos,data): #1 2 3 4

new\_node = Node(data)

if(pos == 0):

new\_node.next = self.head

self.head = new\_node

current = self.head

pointer = 0

while(current and pointer < pos - 1):

current = current.next

pointer += 1

new\_node.next = current.next

current.next = new\_node

def delete\_at\_position(self,pos):

if(self.head is None):

return("List is Empty...")

if(pos == 0):

delete\_node = self.head

self.head = self.head.next

delete\_node.next = None

return("Deletion successful...")

current = self.head

pointer = 0

while(current and pointer < pos - 1):

current = current.next

pointer += 1

delete\_node = current.next

current.next = delete\_node.next

delete\_node.next = None

return "Deletion successful..."

def print\_list(self):

current = self.head

while(current):

print(current.data,end = "-->")

current = current.next

print()

def print\_reverse\_list(self):

lst = []

current = self.head

while(current is not None):

lst.append(current.data)

current = current.next

print(lst[::-1])

def size\_of\_list(self):

count = 0

current = self.head

while(current is not None):

count+=1

current = current.next

print(count)

def search\_element(self,key):

if(self.head is None):

return "List is empty..."

current = self.head

pos = -1

while(current is not None):

pos+=1

if(current.data == key):

return pos

current = current.next

return -1

sll = SLL()

sll.insert(12)

sll.insert(123)

sll.insert(24)

sll.insert(33)

sll.insert(29)

sll.print\_list()

sll.insertAtPosition(2,52)

sll.print\_list()

sll.print\_reverse\_list()

sll.size\_of\_list()

sll.delete\_at\_position(3)

sll.print\_list()

sll.size\_of\_list()

print(sll.search\_element(111))