**Program 1:**

**Create Array and perform the following operations: a)Update array b)slice array element c)extend array d)append array e)remove element.**

**Solution:**

import array as arr

data = arr.array('i',[10,20,30,40,50])

print(data)

print("Array after updating: ")

data[2] = 90

print(data)

print("After slicing array from 2nd position to 3rd position: ")

print(data[1:4])

print("After extending array: ")

data.extend([60,70,80])

print(data)

print("After appending array: ")

data.append(100)

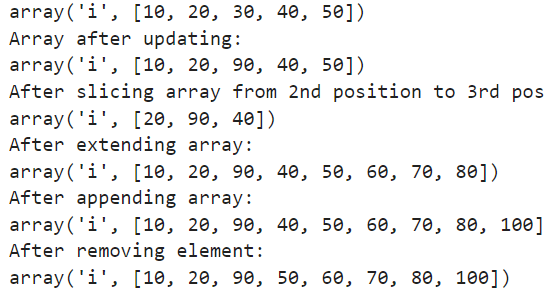
print(data)

print("After removing element: ")

data.remove(40)

print(data)

**Output:**

****

**Program 2:**

**Create an empty list, take size of list from user and append oneone element in empty list.**

**Solution:**

data=[]

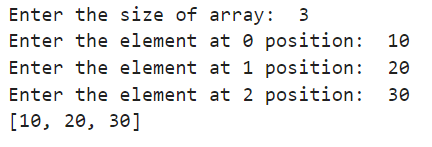
n = int(input("Enter the size of array: "))

for i in range(0,n):

data.append(int(input(f"Enter the element at {i} position: ")))

print(data)

**Output:**

****

**Program 3:**

**Create a list and perform following operations: a)update b)count c)sort d)insert e)append f)extend g)pop**

**Solution:**

data = [34,5,67,43,5,87,89]

print(data)

print("list after updation: ")

data[2]=90

print(data)

print("The element 5 is occured",data.count(5),"Times")

print("list after sorting: ")

data.sort()

print(data)

print("Inserting element 45 at position 4")

data.insert(3,45)

print(data)

print("After appending the list")

data.append(100)

print(data)

print("After extending the list with 3 elements")

data.extend([4,5,6])

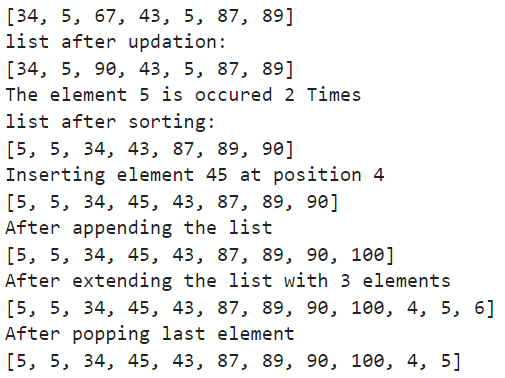
print(data)

print("After popping last element")

data.pop()

print(data)

**Output:**

****

**Program 4:**

**Create function Employee details with user define data like empid, empname, empqualification, empdesignation, empsalary, emp\_currentmonthrank. If current month rank is >4 than add**

**incentive Rs.2500 in salary.**

**Solution:**

def emp\_details():

emp\_id=int(input("Enter the employee id: "))

emp\_name=(input("Enter the employee name: "))

emp\_qualification=(input("Enter the employee qualification: "))

emp\_designation=(input("Enter the employee designation: "))

emp\_salary=int(input("Enter the employee salary: "))

emp\_currentmonthrank=int(input("Enter the employee's current month rank: "))

print("-------------------------------------------------------------------")

print("Id=",emp\_id)

print("Name=",emp\_name)

print("Qualification=",emp\_qualification)

print("Designation=",emp\_designation)

print("Salary=",emp\_salary)

print("Current month rank=",emp\_currentmonthrank)

if emp\_currentmonthrank>4:

emp\_salary=emp\_salary+2500

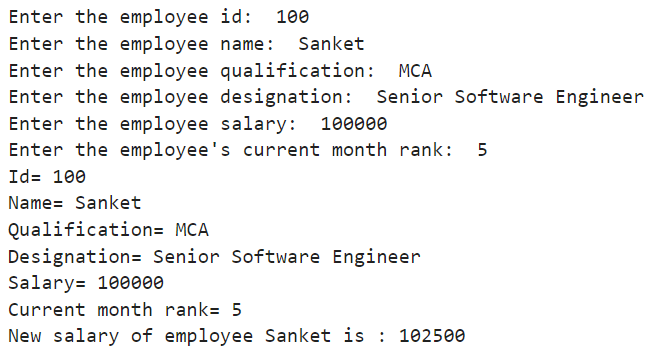
print(f"New salary of employee {emp\_name} is :",emp\_salary)

else:

print("Salary=",emp\_salary)

emp\_details()

**Output:**

****

**Program 5:**

**Create class Pizza\_order\_management\_system with the class attributes like ordered, Pizza\_order, Pizza\_type, Quantity, Address, Distance\_Kilometer. If the customer distance kilometer is greater than 3.5 than Rs 45 will the extra charge will be added in bill.Print bill details in printBill()**

**Solution:**

class Pizza\_order\_management\_system():

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

self.cust\_Name=input("Enter customer name: ")

self.cust\_MoNo=input("Enter customer Mo No:")

self.pizza\_Name=input("Enter Name of pizza: ")

self.pizza\_Type=input("Enter pizza type -> 1)Small 2)Extra 3)Large 4)Extra Large\n")

self.pizza\_Price=int(input("Enter Price of pizza: "))

self.Quantity=int(input("Enter quantity: "))

self.Address=input("Enter address for delivery: ")

self.distance\_Km=int(input("Enter the distance in Kilometer: "))

def printBill(self):

self.Total=self.pizza\_Price\*self.Quantity

if self.distance\_Km>3.5:

self.dist=self.distance\_Km-3.5

self.Total=self.Total+45\*round(self.dist)

print("\n\n\n")

print("--------------------------------------------")

print(self.cust\_Name)

print(self.cust\_MoNo)

print("Order\_> ",self.pizza\_Name," ",self.pizza\_Type)

print("Total-> ",self.pizza\_Price," \* ",self.Quantity," = ",self.Total)

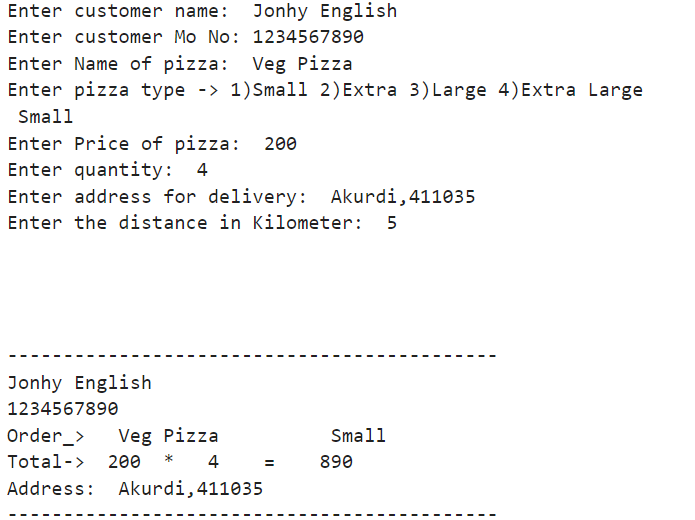
print("Address: ",self.Address)

print("--------------------------------------------")

pizza=Pizza\_order\_management\_system()

pizza.printBill()

**Output:**

****

**Program 6:**

**Create Singly linked list and perform the following operations: a)Insert from start b)Insert from end c)traverse**

**Solution:**

class Node():

def \_\_init\_\_(self,data): #Create a new Node

self.data=data

self.next=None

class LinkedList():

def \_\_init\_\_(self): #Define head and tail

self.head=None

self.tail=None

def insert\_beg(self,data): #Insert at beginning

newNode=Node(data)

if self.head==None:

self.head=self.tail=newNode

else:

newNode.next=self.head

self.head=newNode

def insert\_end(self,data): #Insert at End

newNode=Node(data)

if self.head==None:

self.head=self.tail=newNode

else:

self.tail.next=newNode

self.tail=newNode

def display(self): #Traverse the linked list

traverse=self.head

while traverse!=None:

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

traverse=traverse.next

print("None")

ll=LinkedList()

ll.insert\_beg(10)

ll.insert\_beg(20)

ll.insert\_end(30)

ll.insert\_end(40)

ll.display()

**Output:**

****

**Program 7:**

**Create Singly linked list and perform the following operations: a)Insert from specific position b)traverse**

**Solution:**

class Node():

def \_\_init\_\_(self,data): #Create a new Node

self.data=data

self.next=None

class LinkedList():

def \_\_init\_\_(self): #Define head and tail

self.head=None

self.tail=None

def insert\_beg(self,data): #Insert at beginning

newNode=Node(data)

if self.head==None:

self.head=self.tail=newNode

else:

newNode.next=self.head

self.head=newNode

def insert\_spec\_loc(self,data,loc): #Insert at specific position

cNode=self.head

cLoc=1

newNode=Node(data)

if self.head==None:

self.head=self.tail=newNode

elif loc==1:

newNode.next=self.head

self.head=newNode

else:

while cNode.next!=None and cLoc<loc-1:

cNode=cNode.next

cLoc=cLoc+1

newNode.next=cNode.next

cNode.next=newNode

def display(self): #Traverse the linked list

traverse=self.head

while traverse!=None:

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

traverse=traverse.next

print("None")

ll=LinkedList()

ll.insert\_beg(30)

ll.insert\_beg(20)

ll.insert\_beg(10)

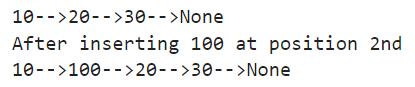
ll.display()

print("After inserting 100 at position 2nd")

ll.insert\_spec\_loc(100,2)

ll.display()

**Output:**

****