**Practical 6**

**Operations on array**

**Q1)Create an array**

a = ["pratham",64,1252,16,4,3 ]

print("Create a array:")

print(a)

**Q2)Access**

print("Accessing array:",a[2])

**Q3)Traverse**

for i in a:

print("traversal:",i)

**Q4)Delete at end and start**

a.pop(0)

print("deletion at start:")

print(a)

print("deletion at end")

a.pop()

print(a)

**Q5)Insert at end and at start**

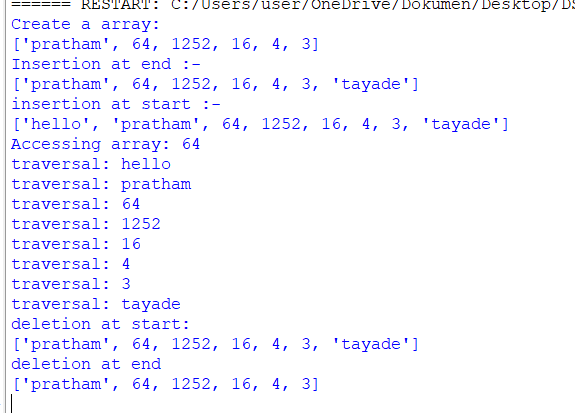
print("Insertion at end :-")

a.append("tayade")

print(a)

print("insertion at start :-")

a.insert(0,"hello")

print(a)

**Practical 7**

**Insertion Sort**

def insertionSort(arr):

n = len(arr)

if n <= 1:

return

for i in range(1, n):

index = arr[i]

j = i-1

while j >= 0 and index < arr[j]:

arr[j+1] = arr[j]

j -= 1

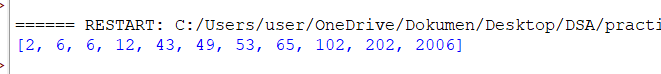
arr[j+1] = index

arr = [6,2006, 2,6,102,53,49,43,65,12,202]

insertionSort(arr)

print(arr)

output:-

****

**Practical 8**

**Selection of Sort**

def selection(arr1):

n=len(arr1)

for i in range(n-1):

mini =i

for j in range(i+1,n):

if arr1[1]<arr1[mini]:

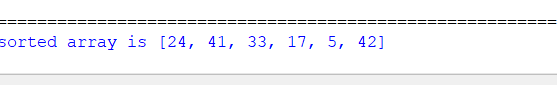
mini=j

arr1[i],arr1[mini]=arr1[mini],arr1[i]

arr1=[24,41,33,42,17]

selection(arr1)

print("sorted array is",arr1)



**Practical 9**

**Demonstration of bubble sort**

def bubble(arr1):

n=len(arr1)

for i in range(n):

swapped=False

for j in range(0,n-i-1):

if arr1[j]>arr1[j+1]:

arr1[j],arr1[j+1]=arr1[j+1],arr1[j]

swapped=True

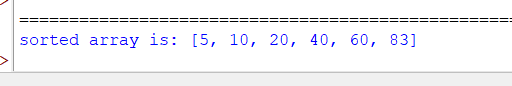
if not swapped:

break

arr1=[40,60,20,10,83,5]

bubble(arr1)

print("sorted array is:",arr1)



**Practical 10**

**Linked list**

**Creating a linked list**

class node:

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

self.data=data

self.next=None

node1=node(10)

node2=node(20)

node3=node(30)

node4=node(40)

node1.next=node2

node2.next=node3

node3.next=node4

head=node1

new\_node=node(50) #

new\_node.next=head

head=new\_node

current = head

while current is not None:

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

current = current.next

print**("None")**

**output:-**

**Deletion of linked list**

**Deletion at the start**

class node:

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

self.data=data

self.next=None

head=node(10)

node2=node(20)

node3=node(30)

node4=node(40)

head.next=node2

node2.next=node3

node3.next=node4

if head is not None:

head=head.next #update the head to the next node

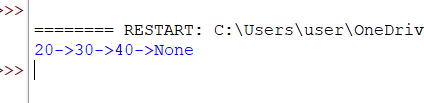
current=head

while current is not None:

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

current=current.next

print("None")

**output:-**

**Deletion at the specific position**

class node:

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

self.data = data

self.next = None

head = node(10)

node2 = node(20)

node3 = node(30)

node4 = node(40)

head.next = node2

node2.next = node3

node3.next = node4

current = head

if current.data == 30:

head = current.next

else:

while current.next is not None: # Traverse till we find the node

if current.next.data == 30:

current.next = current.next.next # Delete the node

break

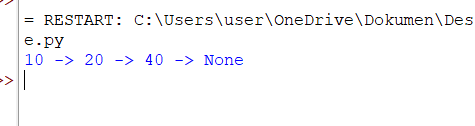
current = current.next

current = head

while current is not None:

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

current = current.next

****print("None")

**Deletion at the end**

class node:

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

self.data = data

self.next = None

head = node(10)

node2 = node(20)

node3 = node(30)

node4 = node(40)

head.next = node2

node2.next = node3

node3.next = node4

current = head

if current is not None and current.next is not None:

while current.next.next is not None:

current = current.next

current.next = None

else:

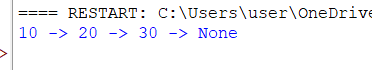
head = None

current = head

while current is not None:

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

current = current.next

print("None")

**Insertion at the specific position**

class node:

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

self.data=data

self.next=None

node1=node(10)

node2=node(20)

node3=node(30)

node4=node(40)

node1.next=node2

node2.next=node3

node3.next=node4

new\_node=node(25)

current = node1

while current is not None and current.data != 20:

current = current.next

if current is not None: # Ensuring 20 exists in the list

new\_node.next = current.next

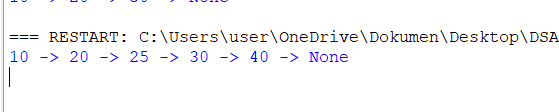
current.next = new\_node

current = node1

while current is not None:

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

current = cur**rent.next**

print("None")

**Insertion at the start**

class node:

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

self.data=data

self.next=None

node1=node(10)

node2=node(20)

node3=node(30)

node4=node(40)

node5=node(50)

node1.next=node2

node2.next=node3

node3.next=node4

node4.next=node5

head=node1

new\_node=node(50)

new\_node.next=head

head=new\_node

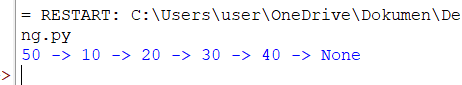
current = head

while current is not None:

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

current = current.next

print("None")

****

**Insertion at the end**

class node:

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

self.data = data

self.next = None

node1 = node(10)

node2 = node(20)

node3 = node(30)

node4 = node(40)

node1.next = node2

node2.next = node3

node3.next = node4

new\_node = node(50)

head = node1

current = head

while current.next is not None:

current = current.next

current.next = new\_node

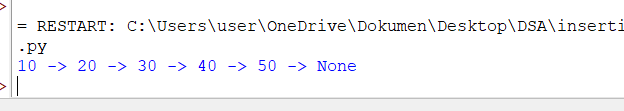
current = head

while current is not None:

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

current = current.next

print("None")

****