



**MALAD KANDIVALI EDUCATION SOCIETY'S**

**NAGINDAS KHANDWALA COLLEGE OF COMMERCE, ARTS &  
MANAGEMENT STUDIES & SHANTABEN NAGINDAS KHANDWALA  
COLLEGE OF SCIENCE**

**MALAD [W], MUMBAI – 64**

**AUTONOMOUS INSTITUTION**

(Affiliated To University Of Mumbai)

Reaccredited 'A' Grade by NAAC | ISO 9001:2015 Certified

**CERTIFICATE**

Name: Mr.PRATHAMESH J BASARE

Roll No: 307

Programme: BSc IT

Semester: III

This is certified to be a bonafide record of practical works done by the above student in the college laboratory for the course **Data Structures (Course Code: 2032UISPR)** for the partial fulfilment of Third Semester of BSc IT during the academic year 2020-21.

The journal work is the original study work that has been duly approved in the year 2020-21 by the undersigned.

---

External Examiner

---

Mr. Gangashankar Singh  
(Subject-In-Charge)

Date of Examination:

(College Stamp)

**Subject: Data Structures**

## INDEX

Sr No	Date	Topic	Sign
1	04/09/2020	Implement the following for Array: a) Write a program to store the elements in 1-D array and provide an option to perform the operations like searching, sorting, merging, reversing the elements. b) Write a program to perform the Matrix addition, Multiplication and Transpose Operation.	
2	11/09/2020	Implement Linked List. Include options for insertion, deletion and search of a number, reverse the list and concatenate two linked lists.	
3	18/09/2020	Implement the following for Stack: a) Perform Stack operations using Array implementation. b. b) Implement Tower of Hanoi. c) WAP to scan a polynomial using linked list and add two polynomials. d) WAP to calculate factorial and to compute the factors of a given no. (i) using recursion, (ii) using iteration	
4	25/09/2020	Perform Queues operations using Circular Array implementation.	
5	01/10/2020	Write a program to search an element from a list. Give user the option to perform Linear or Binary search.	
6	09/10/2020	WAP to sort a list of elements. Give user the option to perform sorting using Insertion sort, Bubble sort or Selection sort.	
7	16/10/2020	Implement the following for Hashing: a) Write a program to implement the collision technique. b) Write a program to implement the concept of linear probing.	
8	23/10/2020	Write a program for inorder, postorder and preorder traversal of tree.	

## Practical 1a

Aim: Write a program to store the elements in 1-D array and provide an option to perform the operations like searching, sorting, merging, reversing the elements.

Theory : Storing Data in Arrays. Assigning values to an element in an array is similar to assigning values to scalar variables. Simply reference an individual element of an array using the array name and the index inside parentheses, then use the assignment operator (=) followed by a value.

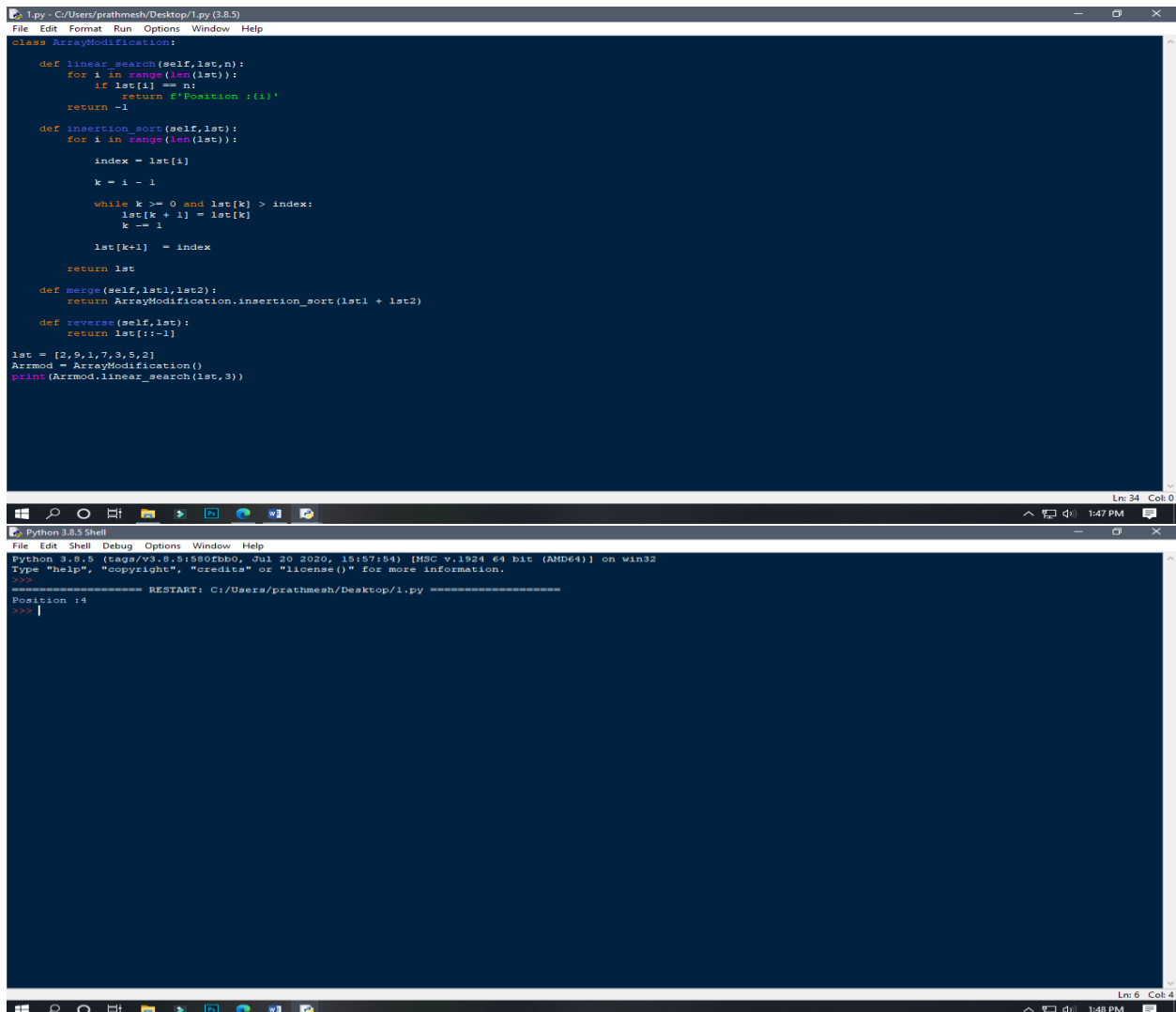
Following are the basic operations supported by an array.

Traverse – print all the array elements one by one.

Insertion – Adds an element at the given index.

Deletion – Deletes an element at the given index.

Search – Searches an element using the given index or by the value



```
1.py - C:/Users/prathmesh/Desktop/1.py (3.8.5)
File Edit Format Run Options Window Help

class ArrayModification:

    def linear_search(self, lst, n):
        for i in range(len(lst)):
            if lst[i] == n:
                return f'Position : {i}'
            return -1

    def insertion_sort(self, lst):
        for i in range(len(lst)):
            index = lst[i]
            k = i - 1
            while k >= 0 and lst[k] > index:
                lst[k + 1] = lst[k]
                k -= 1
            lst[k + 1] = index
        return lst

    def merge(self, lst1, lst2):
        return ArrayModification.insertion_sort(lst1 + lst2)

    def reverse(self, lst):
        return lst[::-1]

lst = [2, 9, 1, 7, 3, 5, 2]
Arrmod = ArrayModification()
print(Arrmod.linear_search(lst, 3))

Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:3506b39, Jul 20 2020, 13:07:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Users/prathmesh/Desktop/1.py
Position : 4
>>>
```

## Practical 1b

Aim: Write a program to perform the Matrix addition, Multiplication and Transpose Operation.

Theory : add() – add elements of two matrices.

subtract() – subtract elements of two matrices.

divide() – divide elements of two matrices.

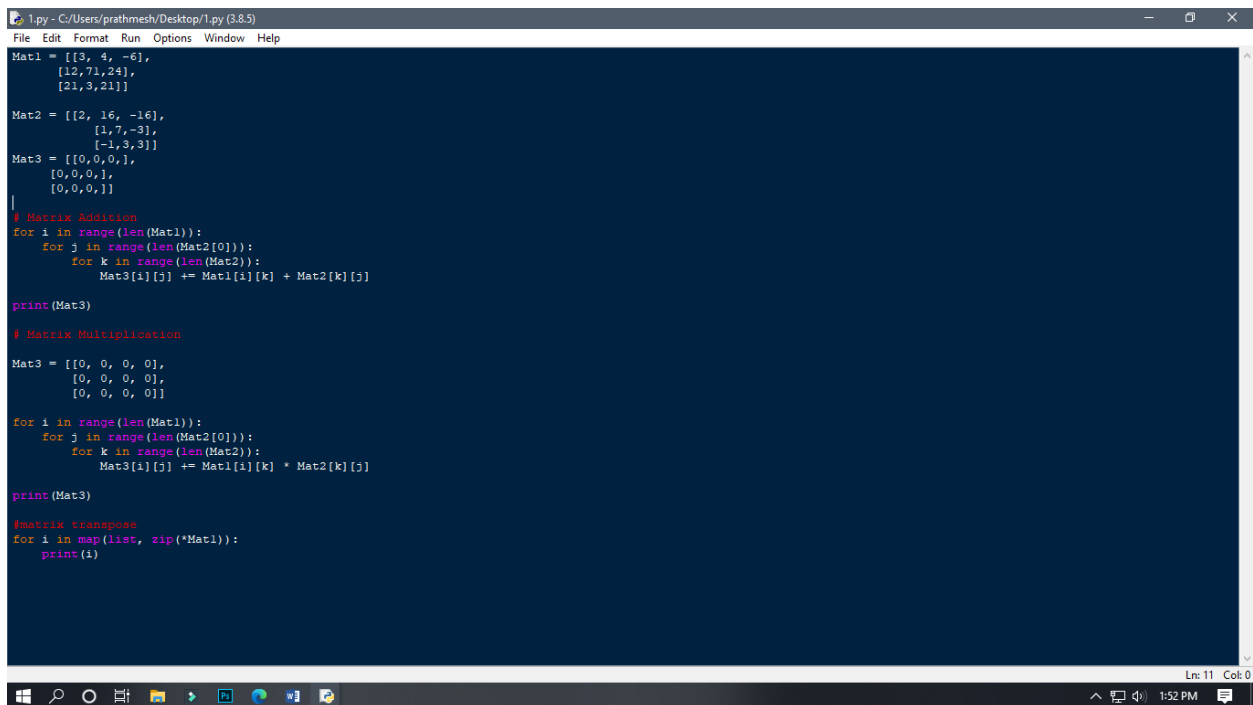
multiply() – multiply elements of two matrices.

dot() – It performs matrix multiplication, does not element wise multiplication.

sqrt() – square root of each element of matrix.

sum(x,axis) – add to all the elements in matrix. Second argument is optional, it is used when we want to compute the column sum if axis is 0 and row sum if axis is 1.

“T” – It performs transpose of the specified matrix.



```
1.py - C:/Users/prathmesh/Desktop/1.py (3.8.5)
File Edit Format Run Options Window Help

Mat1 = [[3, 4, -6],
        [12, 71, 24],
        [21, 3, 21]]

Mat2 = [[2, 16, -16],
        [1, 7, -3],
        [-1, 3, 3]]

Mat3 = [[0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]]

# Matrix Addition
for i in range(len(Mat1)):
    for j in range(len(Mat2[0])):
        for k in range(len(Mat2)):
            Mat3[i][j] += Mat1[i][k] + Mat2[k][j]

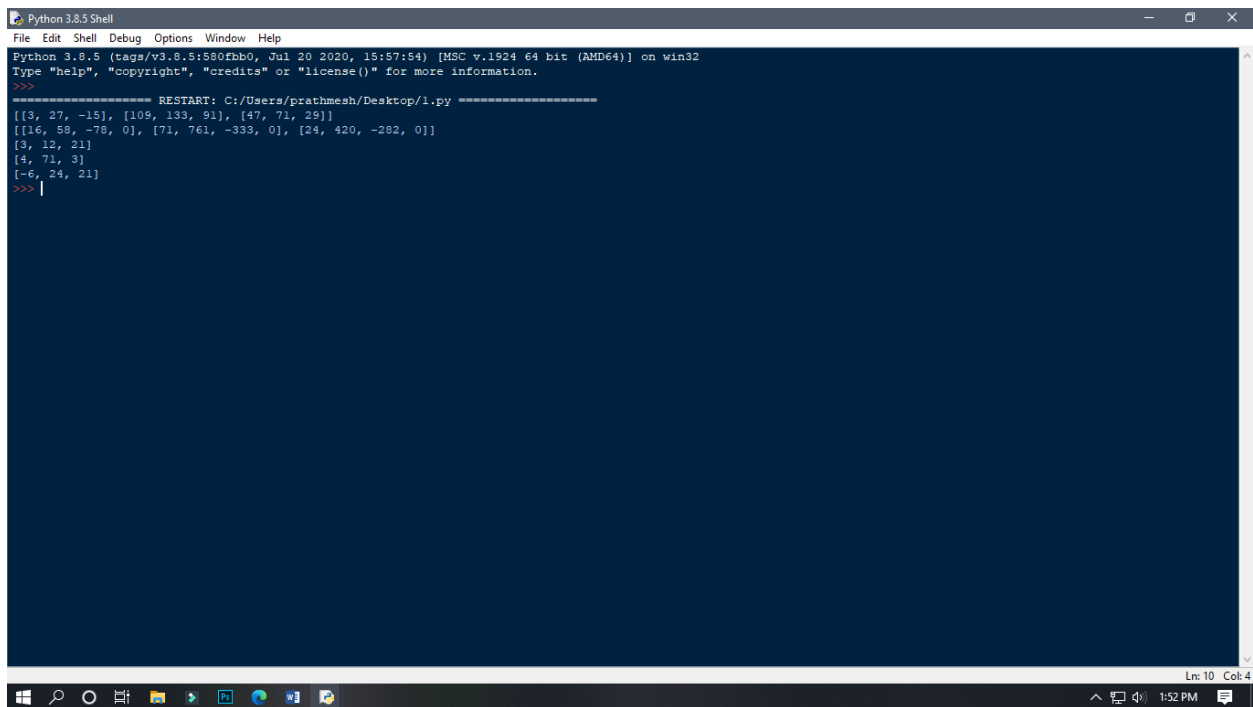
print(Mat3)

# Matrix Multiplication
Mat3 = [[0, 0, 0, 0],
        [0, 0, 0, 0],
        [0, 0, 0, 0]]

for i in range(len(Mat1)):
    for j in range(len(Mat2[0])):
        for k in range(len(Mat2)):
            Mat3[i][j] += Mat1[i][k] * Mat2[k][j]

print(Mat3)

# Matrix transpose
for i in map(list, zip(*Mat1)):
    print(i)
```

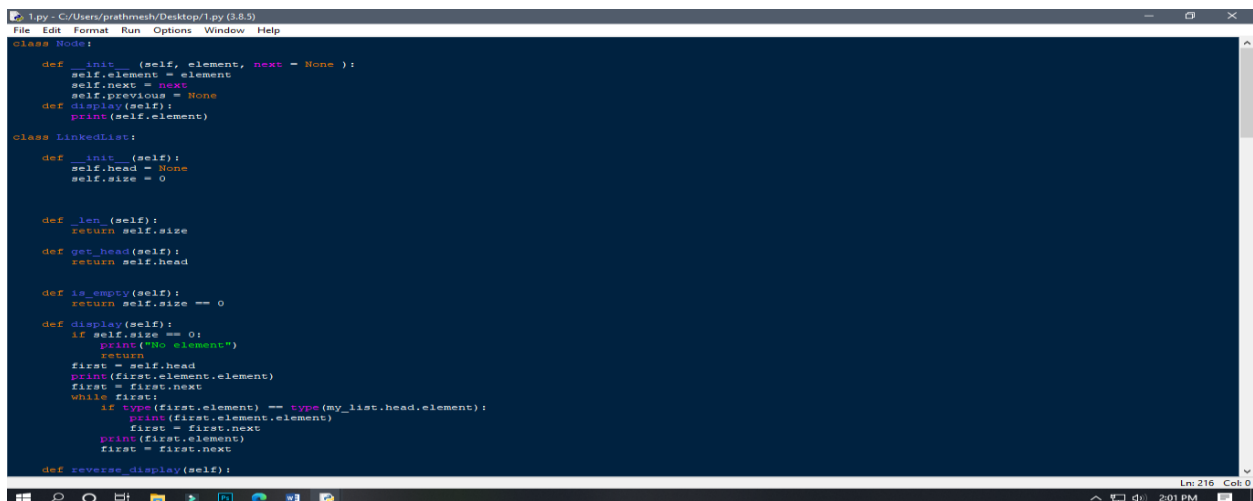


```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:6580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/prathmesh/Desktop/1.py =====
[[3, 27, -15], [109, 133, 91], [47, 71, 29]]
[[16, 58, -78, 0], [71, 761, -333, 0], [24, 420, -282, 0]]
[3, 12, 21]
[4, 71, 3]
[-6, 24, 21]
>>>
```

## Practical 2

Aim: Implement Linked List. Include options for insertion, deletion and search of a number, reverse the list and concatenate two linked lists.

Theory: A linked list is a sequence of data elements, which are connected together via links. Each data element contains a connection to another data element in form of a pointer. Python does not have linked lists in its standard library. We implement the concept of linked lists using the concept of nodes as discussed in the previous chapter. We have already seen how we create a node class and how to traverse the elements of a node. In this chapter we are going to study the types of linked lists known as singly linked lists. In this type of data structure there is only one link between any two data elements. We create such a list and create additional methods to insert, update and remove elements from the list.



```
1.py - C:/Users/prathmesh/Desktop/1.py (3.8.5)
File Edit Format Run Options Window Help

class Node:
    def __init__(self, element, next = None):
        self.element = element
        self.next = next
        self.previous = None
    def display(self):
        print(self.element)

class LinkedList:
    def __init__(self):
        self.head = None
        self.size = 0

    def len(self):
        return self.size

    def get_head(self):
        return self.head

    def is_empty(self):
        return self.size == 0

    def display(self):
        if self.size == 0:
            print("No element")
            return
        first = self.head
        print(first.element.element)
        first = first.next
        while first:
            if type(first.element) == type(my_list.head.element):
                print(first.element.element)
                first = first.next
            print(first.element)
            first = first.next
    def reverse_display(self):
```

```
1.py - C:/Users/prathmesh/Desktop/1.py (3.8.5)
File Edit Format Run Options Window Help

def reverse_display(self):
    if self.size == 0:
        print("No element")
        return None
    last = my_list.get_tail()
    print(last.element)
    while last.previous:
        if type(last.previous.element) == type(my_list.head):
            print(last.previous.element.element)
            if last.previous == self.head:
                return None
            else:
                last = last.previous
        print(last.previous.element)
        last = last.previous

def add_head(self,e):
    temp = self.head
    self.head = Node(e)
    self.head.next = temp
    self.size += 1

def get_tail(self):
    last_object = self.head
    while (last_object.next != None):
        last_object = last_object.next
    return last_object

def remove_head(self):
    if self.is_empty():
        print("Empty Singly linked list")
    else:
        print("Removing")
        self.head = self.head.next
        self.head.previous = None
        self.size -= 1

def add_tail(self,e):
    new_value = Node(e)
    new_value.previous = self.get_tail()
    self.get_tail().next = new_value
    self.size += 1

def find_second_last_element(self):
    #second_last_element = None

    if self.size >= 2:
        first = self.head
        temp_counter = self.size - 2
        while temp_counter > 0:
            first = first.next
            temp_counter -= 1
        return first

    else:
        print("Size not sufficient")

    return None

def remove_tail(self):
    if self.is_empty():
        print("Empty Singly linked list")
    elif self.size == 1:
        self.head == None
        self.size -= 1
    else:
        Node = self.find_second_last_element()
        if Node:
            Node.next = None
            self.size -= 1

def get_node_at(self,index):
    element_node = self.head
    counter = 0
```

```
1.py - C:/Users/prathmesh/Desktop/1.py (3.8.5)
File Edit Format Run Options Window Help

element_node = self.head
counter = 0
if index == 0:
    return element_node.element
if index > self.size-1:
    print("Index out of bound")
    return None
while(counter < index):
    element_node = element_node.next
    counter += 1
return element_node

def get_previous_node_at(self,index):
    if index == 0:
        print('No previous value')
        return None
    return my_list.get_node_at(index).previous

def remove_between_list(self,position):
    if position > self.size-1:
        print("Index out of bound")
    elif position == self.size-1:
        self.remove_tail()
    elif position == 0:
        self.remove_head()
    else:
        prev_node = self.get_node_at(position-1)
        next_node = self.get_node_at(position+1)
        prev_node.next = next_node
        next_node.previous = prev_node
        self.size -= 1

def add_between_list(self,position,element):
    element_node = Node(element)
    if position > self.size:
        print("Index out of bound")
    elif position == self.size:
        self.add_tail(element)
    elif position == 0:
        self.add_head(element)
    else:
        prev_node = self.get_node_at(position-1)
        element_node.next = current_node
        current_node.previous = element_node
        self.size += 1

def search(self,search_value):
    index = 0
    while (index < self.size):
        value = self.get_node_at(index)
        if type(value.element) == type(my_list.head):
            print("Searching at " + str(index) + " and value is " + str(value.element.element))
        else:
            print("Searching at " + str(index) + " and value is " + str(value.element))
        if value.element == search_value:
            print("Found value at " + str(index) + " location")
            return True
        index += 1
    print("Not Found")
    return False

def merge(self,linkedlist_value):
    if self.size > 0:
        last_node = self.get_node_at(self.size-1)
        last_node.next = linkedlist_value.head
        linkedlist_value.head.previous = last_node
        self.size = self.size + linkedlist_value.size
    else:
        self.head = linkedlist_value.head
        self.size = linkedlist_value.size

l1 = Node('Element 1')
my_list = LinkedList()
my_list.add_head(l1)
my_list.add_tail('Element 2')
my_list.add_tail('Element 3')
my_list.add_tail('Element 4')
my_list.get_head().element.element
my_list.add_between_list(2,'Element between')
my_list.remove_between_list(2)

my_list2 = LinkedList()
l2 = Node('Element 5')
```

```
1.py - C:/Users/prathmesh/Desktop/1.py (3.8.5)
File Edit Format Run Options Window Help

    print("Searching at " + str(index) + " and value is " + str(value.element.element))
else:
    print("Searching at " + str(index) + " and value is " + str(value.element))
if value.element == search_value:
    print("Found value at " + str(index) + " location")
    return True
    index += 1
print("Not Found")
return False

def merge(self, linkedlist_value):
    if self.size > 0:
        last_node = self.get_node_at(self.size-1)
        last_node.next = linkedlist_value.head
        linkedlist_value.head.previous = last_node
        self.size = self.size + linkedlist_value.size
    else:
        self.head = linkedlist_value.head
        self.size = linkedlist_value.size

l1 = Node('Element 1')
my_list = LinkedList()
my_list.add_head(l1)
my_list.add_tail('Element 2')
my_list.add_tail('Element 3')
my_list.add_tail('Element 4')
my_list.get_head().element.element
my_list.add_between_list(2, 'Element between')
my_list.remove_between_list(2)

my_list2 = LinkedList()
l2 = Node('Element 5')
my_list2.add_head(l2)
my_list2.add_tail('Element 6')
my_list2.add_tail('Element 7')
my_list2.add_tail('Element 8')
my_list.merge(my_list2)
my_list.get_previous_node_at(3).element
my_list.reverse_display()
my_list.search('Element 6')

Ln: 214 Col: 0

Python 3.8.5 Shell
File Edit Shell Debug Options Window Help

Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/prathmesh/Desktop/1.py =====
Element 8
Element 7
Element 6
Element 5
Element 4
Element 3
Element 2
Element 1
Searching at 0 and value is Element 1
Searching at 1 and value is Element 2
Searching at 2 and value is Element 3
Searching at 3 and value is Element 4
Searching at 4 and value is Element 5
Searching at 5 and value is Element 6
Found value at 5 location
>>>

Ln: 20 Col: 4
```



## Practical 3a

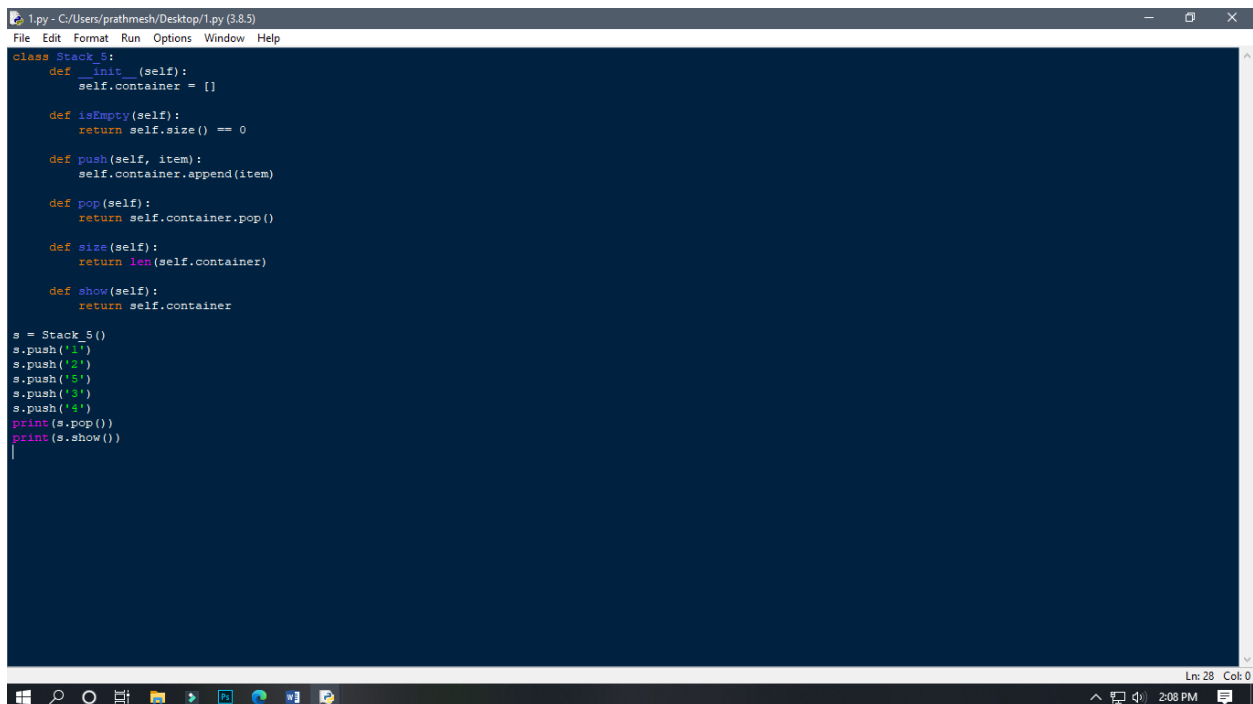
**Aim:** Perform Stack operations using Array implementation.

**Theory:** **Stacks** is one of the earliest data structures defined in computer science. In simple words, **Stack** is a linear collection of items. It is a collection of objects that supports fast last-in, first-out (LIFO) semantics for insertion and deletion. It is an array or list structure of function calls and parameters used in modern computer programming and CPU architecture. Similar to a stack of plates at a restaurant, elements in a stack are added or removed from the top of the stack, in a “last in, first out” order. Unlike lists or arrays, random access is not allowed for the objects contained in the stack.

There are two types of operations in Stack-

**Push**– To add data into the stack.

**Pop**– To remove data from the stack

A screenshot of a Python IDE window titled '1.py - C:/Users/prathmesh/Desktop/1.py (3.8.5)'. The window contains a Python script that implements a Stack class using a list. The class has methods for push, pop, isEmpty, size, and show. The script creates an instance of the Stack class, pushes five items ('1' through '5'), pops one item, and prints the results. The IDE interface includes a menu bar (File, Edit, Format, Run, Options, Window, Help) and a status bar at the bottom showing 'Ln: 28 Col: 0' and the system clock '2:08 PM'.

```
class Stack:
    def __init__(self):
        self.container = []

    def isEmpty(self):
        return self.size() == 0

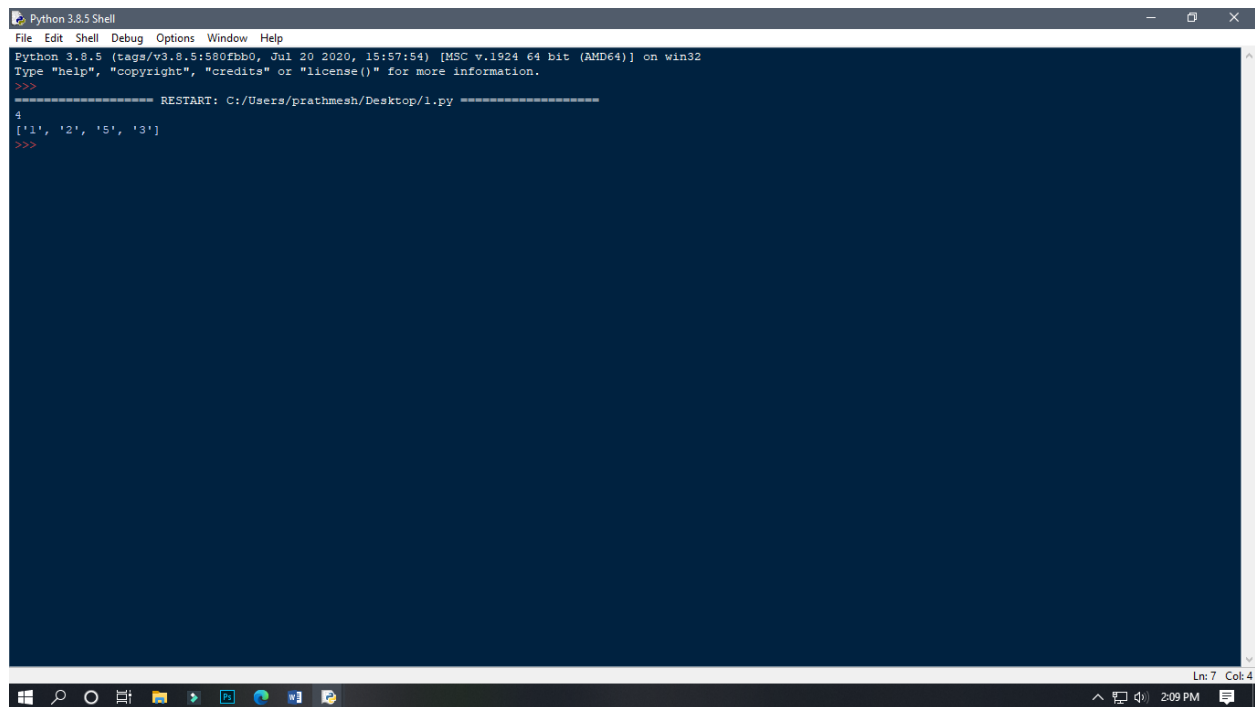
    def push(self, item):
        self.container.append(item)

    def pop(self):
        return self.container.pop()

    def size(self):
        return len(self.container)

    def show(self):
        return self.container

s = Stack()
s.push('1')
s.push('2')
s.push('3')
s.push('4')
s.push('5')
print(s.pop())
print(s.show())
```



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/prathmesh/Desktop/l.py =====
4
['1', '2', '5', '3']
>>>
```

## Practical 3b

**Aim:** Implement Tower of Hanoi.

Theory: The factorial of a number is the product of all the integers from 1 to that number.

For example, the factorial of 6 is  $1*2*3*4*5*6 = 720$ . Factorial is not defined for negative numbers and the factorial of zero is one,  $0! = 1$ .

Recursion: In Python, we know that a function can call other functions. It is even possible for the function to call itself. These types of construct are termed as recursive functions.

Iteration: Repeating identical or similar tasks without making errors is something that computers do well and people do poorly. Repeated execution of a set of statements is called iteration. Because iteration is so common, Python provides several language features to make it easier.

```
1.py - C:/Users/prathmesh/Desktop/1.py (3.8.5)
File Edit Format Run Options Window Help

class Stack:

    def __init__(self):
        self.stack_arr = []

    def push(self,value):
        self.stack_arr.append(value)

    def pop(self):
        if len(self.stack_arr) == 0:
            print('Stack is empty!')
            return None
        else:
            self.stack_arr.pop()

    def get_head(self):
        if len(self.stack_arr) == 0:
            print('Stack is empty!')
            return None
        else:
            return self.stack_arr[-1]

    def display(self):
        if len(self.stack_arr) == 0:
            print('Stack is empty!')
            return None
        else:
            print(self.stack_arr)

A = Stack()
B = Stack()
C = Stack()
def towerOfHanoi(n, fromrod,to,temp):
    if n == 1:
        fromrod.pop()
        to.push('disk 1')
        if to.display() != None:
            print(to.display())
        else:
            towerOfHanoi(n-1, fromrod, temp, to)

1.py - C:/Users/prathmesh/Desktop/1.py (3.8.5)
File Edit Format Run Options Window Help

        self.stack_arr.pop()

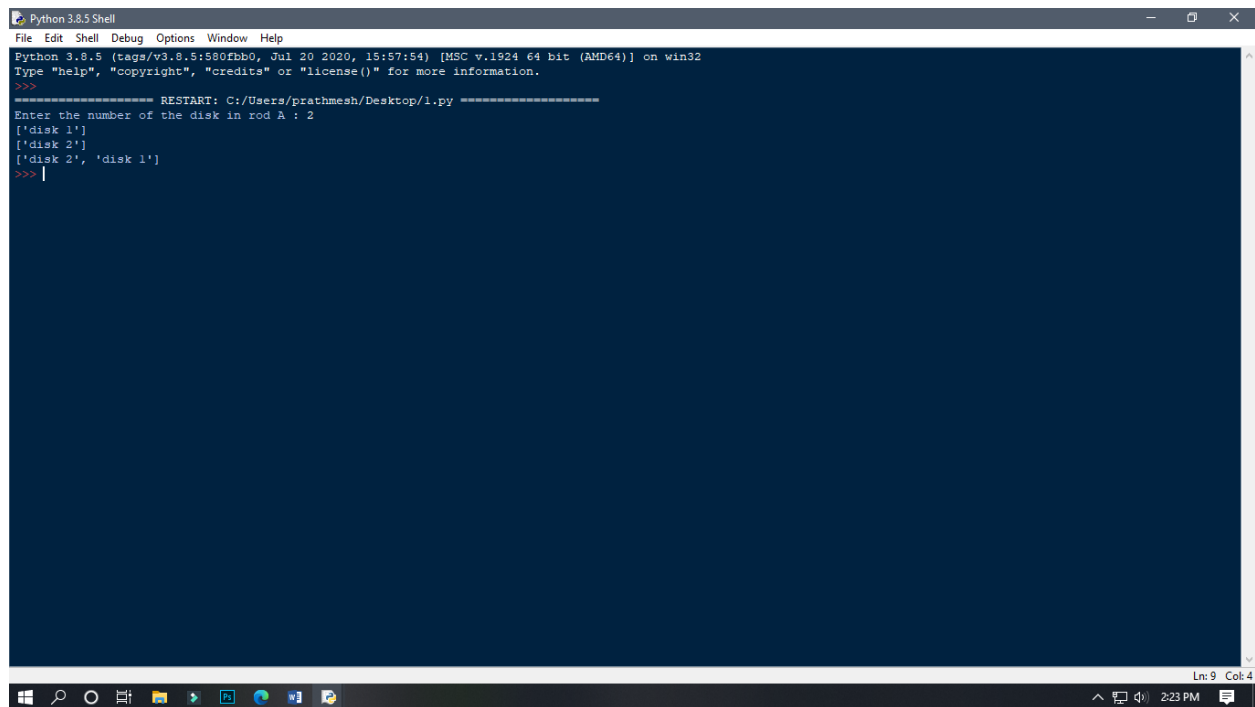
    def get_head(self):
        if len(self.stack_arr) == 0:
            print('Stack is empty!')
            return None
        else:
            return self.stack_arr[-1]

    def display(self):
        if len(self.stack_arr) == 0:
            print('Stack is empty!')
            return None
        else:
            print(self.stack_arr)

A = Stack()
B = Stack()
C = Stack()
def towerOfHanoi(n, fromrod,to,temp):
    if n == 1:
        fromrod.pop()
        to.push('disk 1')
        if to.display() != None:
            print(to.display())
        else:
            towerOfHanoi(n-1, fromrod, temp, to)
            fromrod.pop()
            to.push(f'disk {n}')
            if to.display() != None:
                print(to.display())
            towerOfHanoi(n-1, temp, to, fromrod)

n = int(input('Enter the number of the disk in rod A : '))
for i in range(n):
    A.push(f'disk {i+1} ')

towerOfHanoi(n, A, C, B)
```

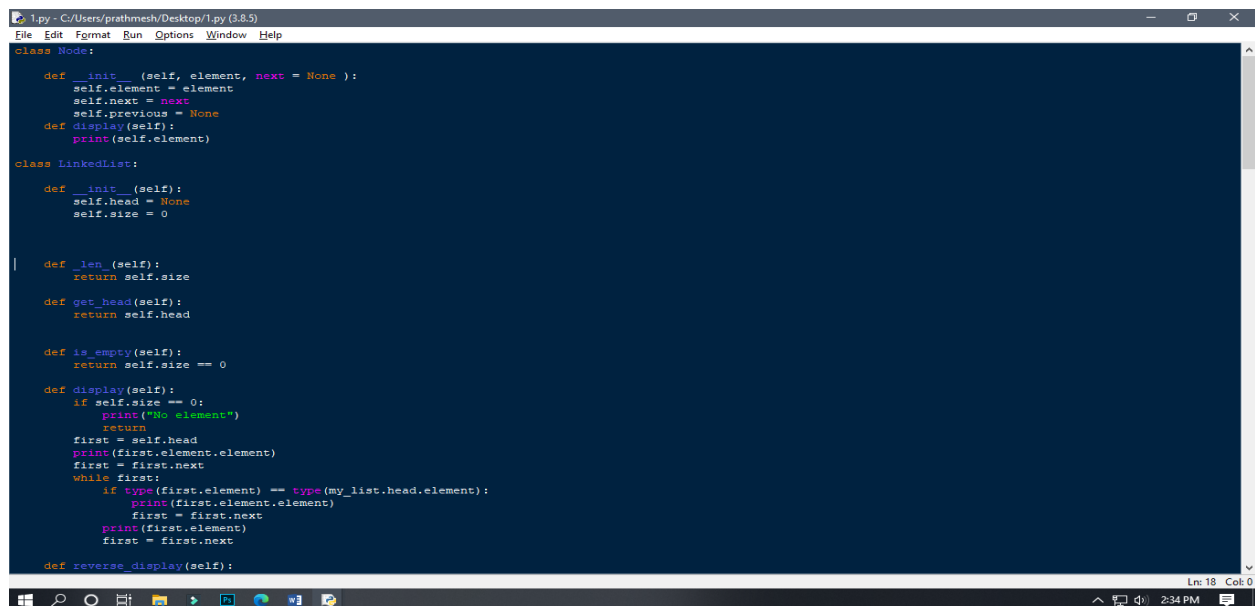


```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:580fb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/prathmesh/Desktop/l.py =====
Enter the number of the disk in rod A : 2
['disk 1']
['disk 2']
['disk 2', 'disk 1']
>>> |
```

### Practical 3c

Aim: WAP to scan a polynomial using linked list and add two polynomials.

Theory: Polynomial is a mathematical expression that consists of variables and coefficients. for example  $x^2 - 4x + 7$ . In the Polynomial linked list, the coefficients and exponents of the polynomial are defined as the data node of the list. For adding two polynomials that are stored as a linked list. We need to add the coefficients of variables with the same power. In a linked list node contains 3 members, coefficient value link to the next node a linked list that is used to store Polynomial looks like –Polynomial :  $4x^7 + 12x^2 + 45$ .



```
l.py - C:/Users/prathmesh/Desktop/l.py (3.8.5)
File Edit Format Run Options Window Help

class Node:
    def __init__(self, element, next = None):
        self.element = element
        self.next = next
        self.previous = None
    def display(self):
        print(self.element)

class LinkedList:
    def __init__(self):
        self.head = None
        self.size = 0

    def __len__(self):
        return self.size

    def get_head(self):
        return self.head

    def is_empty(self):
        return self.size == 0

    def display(self):
        if self.size == 0:
            print("No element")
            return
        first = self.head
        print(first.element.element)
        first = first.next
        while first:
            if type(first.element) == type(my_list.head.element):
                print(first.element.element)
                first = first.next
            print(first.element)
            first = first.next

    def reverse_display(self):
```

```
1.py - C:/Users/prathmesh/Desktop/1.py (3.8.5)
File Edit Format Run Options Window Help

def reverse_display(self):
    if self.size == 0:
        print("No element")
        return None
    last = my_list.get_tail()
    print(last.element)
    while last.previous:
        if type(last.previous.element) == type(my_list.head):
            print(last.previous.element)
            if last.previous == self.head:
                return None
            else:
                last = last.previous
        print(last.previous.element)
        last = last.previous

def add_head(self,e):
    #temp = self.head
    self.head = Node(e)
    #self.head.next = temp
    self.size += 1

def get_tail(self):
    last_object = self.head
    while (last_object.next != None):
        last_object = last_object.next
    return last_object

def remove_head(self):
    if self.is_empty():
        print("Empty Singly linked list")
    else:
        print("Removing")
        self.head = self.head.next
        self.head.previous = None
        self.size -= 1

def add_tail(self,e):
    new_value = Node(e)

new_value = Node(e)
new_value.previous = self.get_tail()
self.get_tail().next = new_value
self.size += 1

def find_second_last_element(self):
    #second_last_element = None

    if self.size >= 2:
        first = self.head
        temp_counter = self.size - 2
        while temp_counter > 0:
            first = first.next
            temp_counter -= 1
        return first

    else:
        print("Size not sufficient")

    return None

def remove_tail(self):
    if self.is_empty():
        print("Empty Singly linked list")
    elif self.size == 1:
        self.head = None
        self.size -= 1
    else:
        Node = self.find_second_last_element()
        if Node:
            Node.next = None
            self.size -= 1

def get_node_at(self,index):
    element_node = self.head
    counter = 0
    if index == 0:
        return element_node.element

Ln: 83 Col: 27
Ln: 124 Col: 39
```

```

1.py - C:/Users/prathmesh/Desktop/1.py (3.8.5)
File Edit Format Run Options Window Help

    return element_node.element
    if index > self.size-1:
        print("Index out of bound")
        return None
    while(counter < index):
        element_node = element_node.next
        counter += 1
    return element_node

def get_previous_node_at(self,index):
    if index == 0:
        print('No previous value')
        return None
    return my_list.get_node_at(index).previous

def remove_between_list(self,position):
    if position > self.size-1:
        print("Index out of bound")
    elif position == self.size-1:
        self.remove_tail()
    elif position == 0:
        self.remove_head()
    else:
        prev_node = self.get_node_at(position-1)
        next_node = self.get_node_at(position+1)
        prev_node.next = next_node
        next_node.previous = prev_node
        self.size -= 1

def add_between_list(self,position,element):
    element_node = Node(element)
    if position > self.size:
        print("Index out of bound")
    elif position == self.size:
        self.add_tail(element)
    elif position == 0:
        self.add_head(element)
    else:
        prev_node = self.get_node_at(position-1)
        current_node = self.get_node_at(position)
        prev_node.next = element_node
        element_node.previous = prev_node

Ln: 165 Col: 45

1.py - C:/Users/prathmesh/Desktop/1.py (3.8.5)
File Edit Format Run Options Window Help

    element_node.previous = prev_node
    element_node.next = current_node
    current_node.previous = element_node
    self.size += 1

def search (self,search_value):
    index = 0
    while (index < self.size):
        value = self.get_node_at(index)
        if value.element == search_value:
            return value.element
        index += 1
    print("Not Found")
    return False

def merge(self,linkedlist_value):
    if self.size > 0:
        last_node = self.get_node_at(self.size-1)
        last_node.next = linkedlist_value.head
        linkedlist_value.head.previous = last_node
        self.size = self.size + linkedlist_value.size
    else:
        self.head = linkedlist_value.head
        self.size = linkedlist_value.size

my_list = LinkedList()
order = int(input('Enter the order for polynomial : '))
my_list.add_head(Node(int(input(f"Enter coefficient for power {order} : "))))
for i in reversed(range(order)):
    my_list.add_tail(int(input(f"Enter coefficient for power {i} : ")))

my_list2 = LinkedList()
my_list2.add_head(Node(int(input(f"Enter coefficient for power {order} : "))))
for i in reversed(range(order)):
    my_list2.add_tail(int(input(f"Enter coefficient for power {i} : ")))

for i in range(order + 1):
    print(my_list.get_node_at(i).element + my_list2.get_node_at(i).element)

Ln: 206 Col: 0

```

### Practical 3d

Aim: WAP to calculate factorial and to compute the factors of a given no.

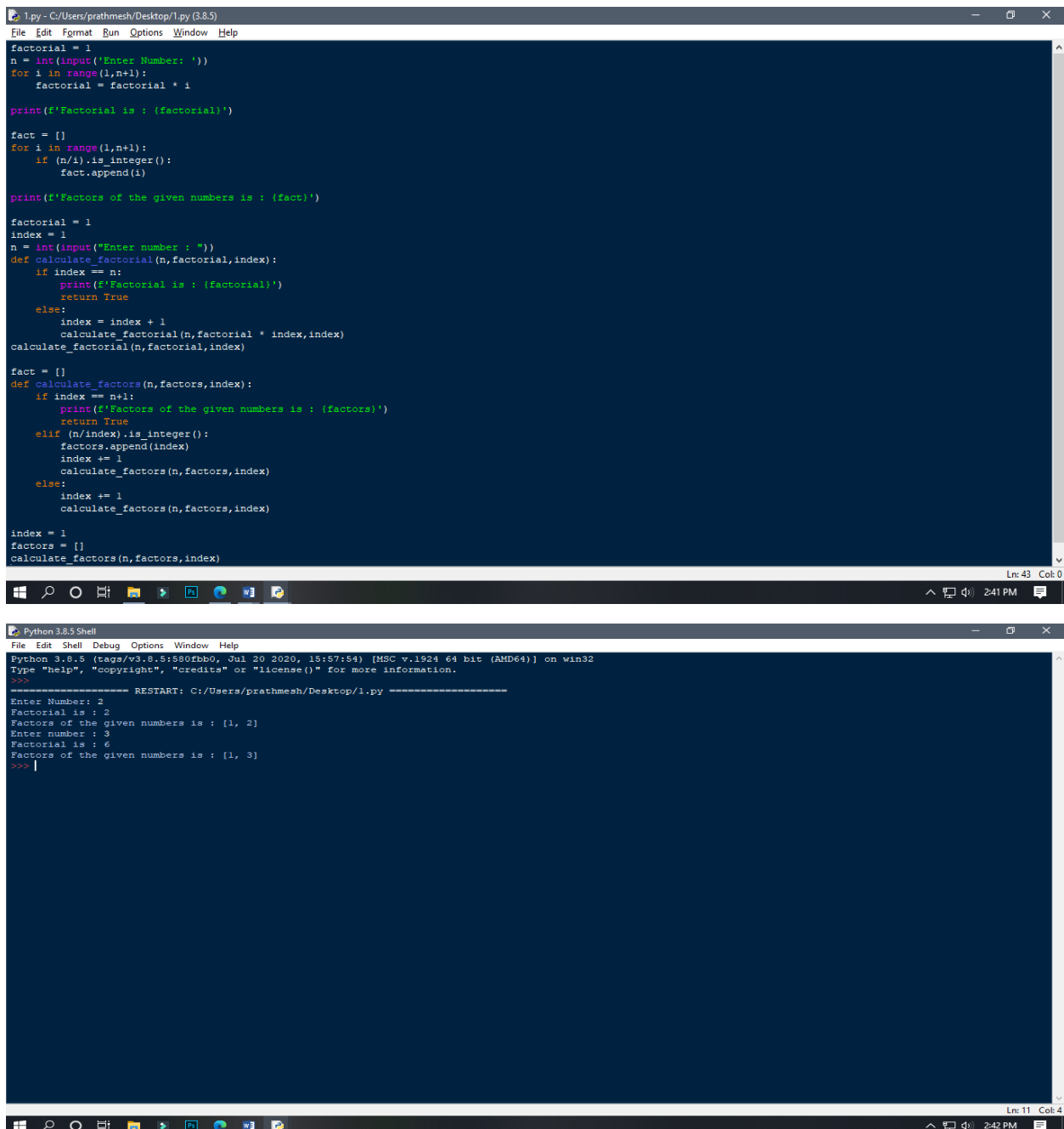
(i)using recursion, (ii) using iteration

Theory: The factorial of a number is the product of all the integers from 1 to that number.

For example, the factorial of 6 is  $1*2*3*4*5*6 = 720$ . Factorial is not defined for negative numbers and the factorial of zero is one,  $0! = 1$ .

Recursion: In Python, we know that a function can call other functions. It is even possible for the function to call itself. These types of construct are termed as recursive functions.

Iteration: Repeating identical or similar tasks without making errors is something that computers do well and people do poorly. Repeated execution of a set of statements is called iteration. Because iteration is so common, Python provides several language features to make it easier.



```
1.py - C:/Users/prathmesh/Desktop/1.py (3.8.5)
File Edit Format Run Options Window Help

factorial = 1
n = int(input('Enter Number: '))
for i in range(1,n+1):
    factorial = factorial * i

print(f'Factorial is : {factorial}')

fact = []
for i in range(1,n+1):
    if (n/i).is_integer():
        fact.append(i)

print(f'Factors of the given numbers is : {fact}')

factorial = 1
index = 1
n = int(input("Enter number : "))
def calculate_factorial(n,factorial,index):
    if index == n:
        print(f'Factorial is : {factorial}')
        return True
    else:
        index = index + 1
        calculate_factorial(n,factorial * index,index)
calculate_factorial(n,factorial,index)

fact = []
def calculate_factors(n,factors,index):
    if index == n+1:
        print(f'Factors of the given numbers is : {factors}')
        return True
    elif (n/index).is_integer():
        factors.append(index)
        index += 1
        calculate_factors(n,factors,index)
    else:
        index += 1
        calculate_factors(n,factors,index)

index = 1
factors = []
calculate_factors(n,factors,index)
```

```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help

Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/prathmesh/Desktop/1.py =====
Enter Number: 2
Factorial is : 2
Factors of the given numbers is : [1, 2]
Enter number : 3
Factorial is : 6
Factors of the given numbers is : [1, 3]
>>> |
```

## Practical 4

Aim: Perform Queues operations using Circular Array implementation.

Theory: Circular queue avoids the wastage of space in a regular queue implementation using arrays.

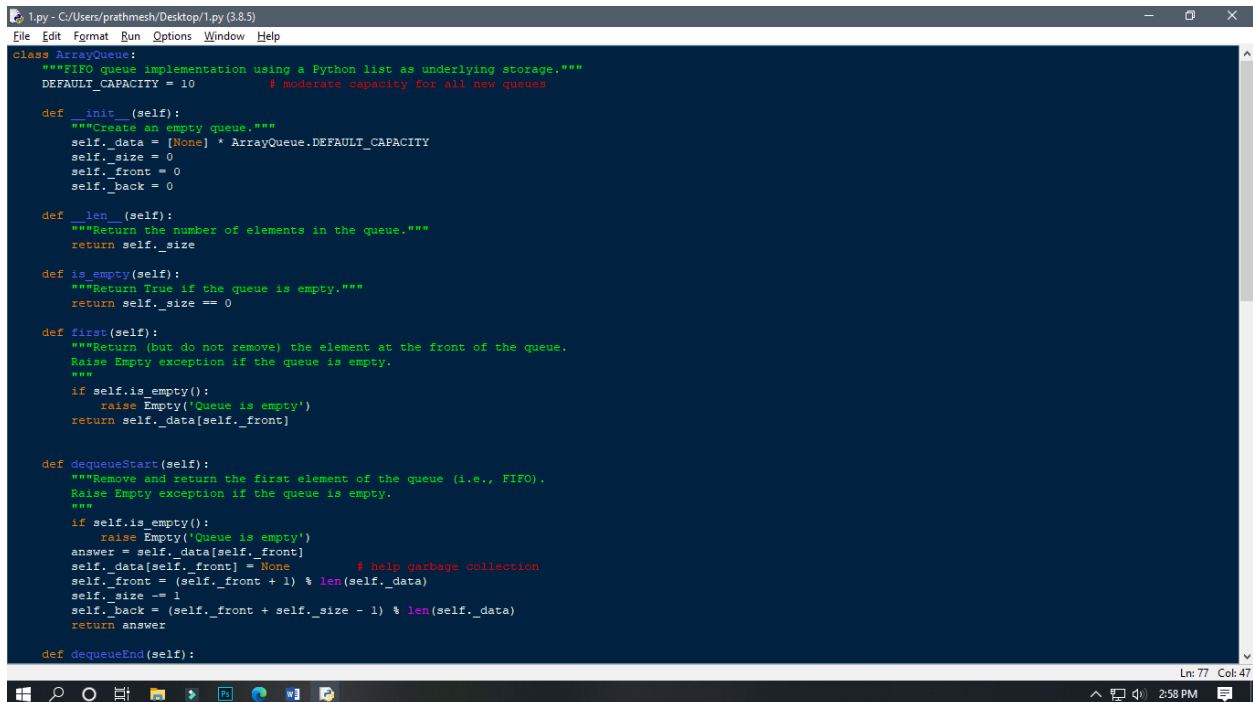
Circular Queue works by the process of circular increment i.e. when we try to increment the pointer and we reach the end of the queue, we start from the beginning of the queue.

Here, the circular increment is performed by modulo division with the queue size. That is, if  $REAR + 1 == 5$  (overflow!),  $REAR = (REAR + 1) \% 5 = 0$  (start of queue)

The circular queue work as follows: two pointers FRONT and REAR FRONT track the first element of the queue REAR track the last elements of the queue initially, set value of FRONT and REAR to -1

1. Enqueue Operation: check if the queue is full for the first element, set value of FRONT to 0 circularly increase the REAR index by 1 (i.e. if the rear reaches the end, next it would be at the start of the queue) add the new element in the position pointed to by REAR

2. Dequeue Operation :check if the queue is empty return the value pointed by FRONT circularly increase the FRONT index by 1 for the last element, reset the values of FRONT and REAR to -1



```
1.py - C:/Users/prathmesh/Desktop/1.py (3.8.5)
File Edit Format Run Options Window Help

class ArrayQueue:
    """FIFO queue implementation using a Python list as underlying storage."""
    DEFAULT_CAPACITY = 10          # moderate capacity for all new queues

    def __init__(self):
        """Create an empty queue."""
        self._data = [None] * ArrayQueue.DEFAULT_CAPACITY
        self._size = 0
        self._front = 0
        self._back = 0

    def __len__(self):
        """Return the number of elements in the queue."""
        return self._size

    def is_empty(self):
        """Return True if the queue is empty."""
        return self._size == 0

    def first(self):
        """Return (but do not remove) the element at the front of the queue.
        Raise Empty exception if the queue is empty.
        """
        if self.is_empty():
            raise Empty('Queue is empty')
        return self._data[self._front]

    def dequeueStart(self):
        """Remove and return the first element of the queue (i.e., FIFO).
        Raise Empty exception if the queue is empty.
        """
        if self.is_empty():
            raise Empty('Queue is empty')
        answer = self._data[self._front]
        self._data[self._front] = None          # help garbage collection
        self._front = (self._front + 1) % len(self._data)
        self._size -= 1
        self._back = (self._front + self._size - 1) % len(self._data)
        return answer

    def dequeueEnd(self):
```



```

1.py - C:/Users/prathmesh/Desktop/1.py (3.8.5)
File Edit Format Run Options Window Help

def dequeueEnd(self):
    """Remove and return the Last element of the queue.
    Raise Empty exception if the queue is empty.
    """
    if self.is_empty():
        raise Empty('Queue is empty')
    back = (self._front + self._size - 1) % len(self._data)
    answer = self._data[back]
    self._data[back] = None # help garbage collection
    self._front = self._front
    self._size -= 1
    self._back = (self._front + self._size - 1) % len(self._data)
    return answer

def enqueueEnd(self, e):
    """Add an element to the back of queue."""
    if self._size == len(self._data):
        self._resize(2 * len(self._data)) # double the array size
    avail = (self._front + self._size) % len(self._data)
    self._data[avail] = e
    self._size += 1
    self._back = (self._front + self._size - 1) % len(self._data)

def enqueueStart(self, e):
    """Add an element to the start of queue."""
    if self._size == len(self._data):
        self._resize(2 * len(self._data))
    self._front = (self._front - 1) % len(self._data)
    avail = (self._front + self._size) % len(self._data)
    self._data[avail] = e
    self._size += 1
    self._back = (self._front + self._size - 1) % len(self._data)

def _resize(self, cap):
    """Resize to a new list of capacity >= len(self)."""
    old = self._data
    self._data = [None] * cap
    walk = self._front
    for k in range(self._size):
        self._data[k] = old[walk]
        walk = (1 + walk) % len(old)
    self._front = 0

queue = ArrayQueue()
queue.enqueueEnd(1)
print(f"First Element: {queue._data[queue._front]}, Last Element: {queue._data[queue._back]}")
queue._data
queue.enqueueEnd(2)
print(f"First Element: {queue._data[queue._front]}, Last Element: {queue._data[queue._back]}")
queue._data
queue.dequeueStart()
print(f"First Element: {queue._data[queue._front]}, Last Element: {queue._data[queue._back]}")
queue.enqueueEnd(3)
print(f"First Element: {queue._data[queue._front]}, Last Element: {queue._data[queue._back]}")
queue.enqueueEnd(4)
print(f"First Element: {queue._data[queue._front]}, Last Element: {queue._data[queue._back]}")
queue.dequeueStart()
print(f"First Element: {queue._data[queue._front]}, Last Element: {queue._data[queue._back]}")
queue.enqueueStart(5)
print(f"First Element: {queue._data[queue._front]}, Last Element: {queue._data[queue._back]}")
queue.dequeueEnd()
print(f"First Element: {queue._data[queue._front]}, Last Element: {queue._data[queue._back]}")
queue.enqueueEnd(6)
print(f"First Element: {queue._data[queue._front]}, Last Element: {queue._data[queue._back]}")

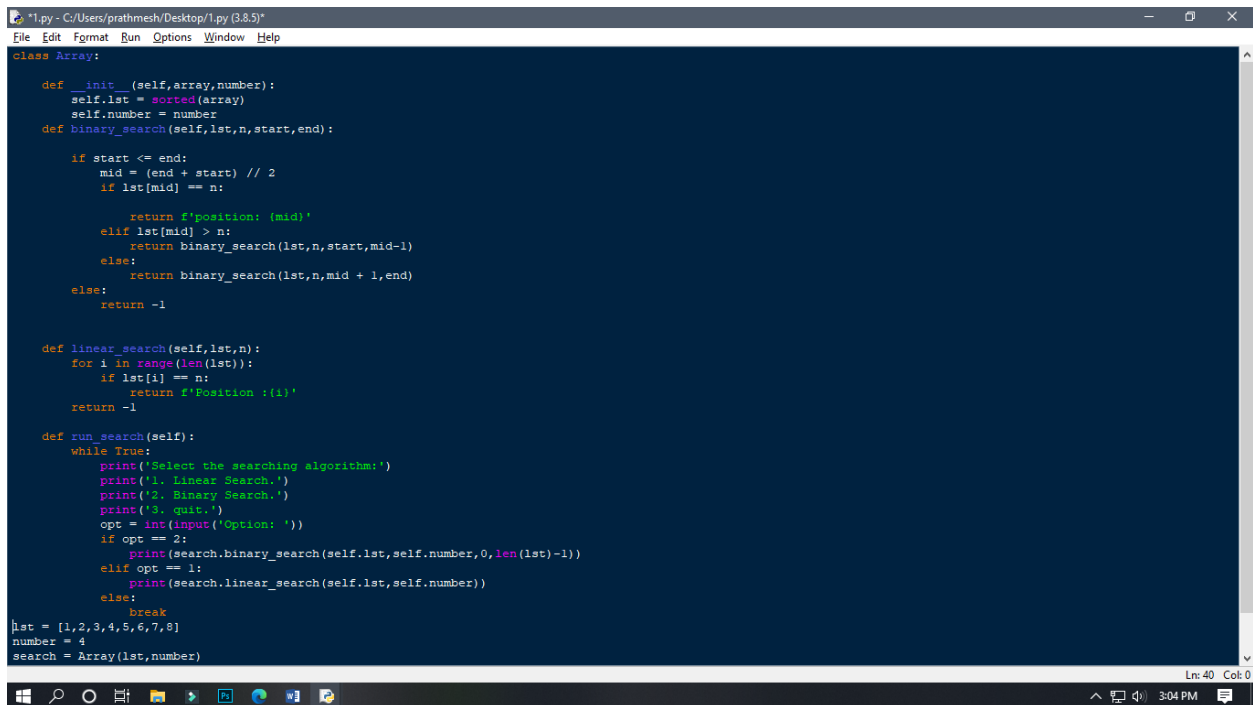
```

## Practical 5

**Aim:** Write a program to search an element from a list. Give user the option to perform Linear or Binary search.

**Theory:** Linear Search: This linear search is a basic search algorithm which searches all the elements in the list and finds the required value. ... This is also known as sequential search.

Binary Search: In computer science, a binary search or half-interval search algorithm finds the position of a target value within a sorted array. The binary search algorithm can be classified as a dichotomies divide-and-conquer search algorithm and executes in logarithmic time.

A screenshot of a Python IDE window titled "1.py - C:/Users/prathmesh/Desktop/1.py (3.8.5)". The window contains a Python script for searching an element in a list. The script defines a class 'Array' with methods for binary search, linear search, and a main search loop. The list is [1, 2, 3, 4, 5, 6, 7, 8] and the number to search is 4. The search method prompts the user to choose between linear and binary search. The status bar at the bottom shows 'Ln: 40 Col: 0' and the system clock shows '3:04 PM'.`*1.py - C:/Users/prathmesh/Desktop/1.py (3.8.5)
File Edit Format Run Options Window Help

class Array:

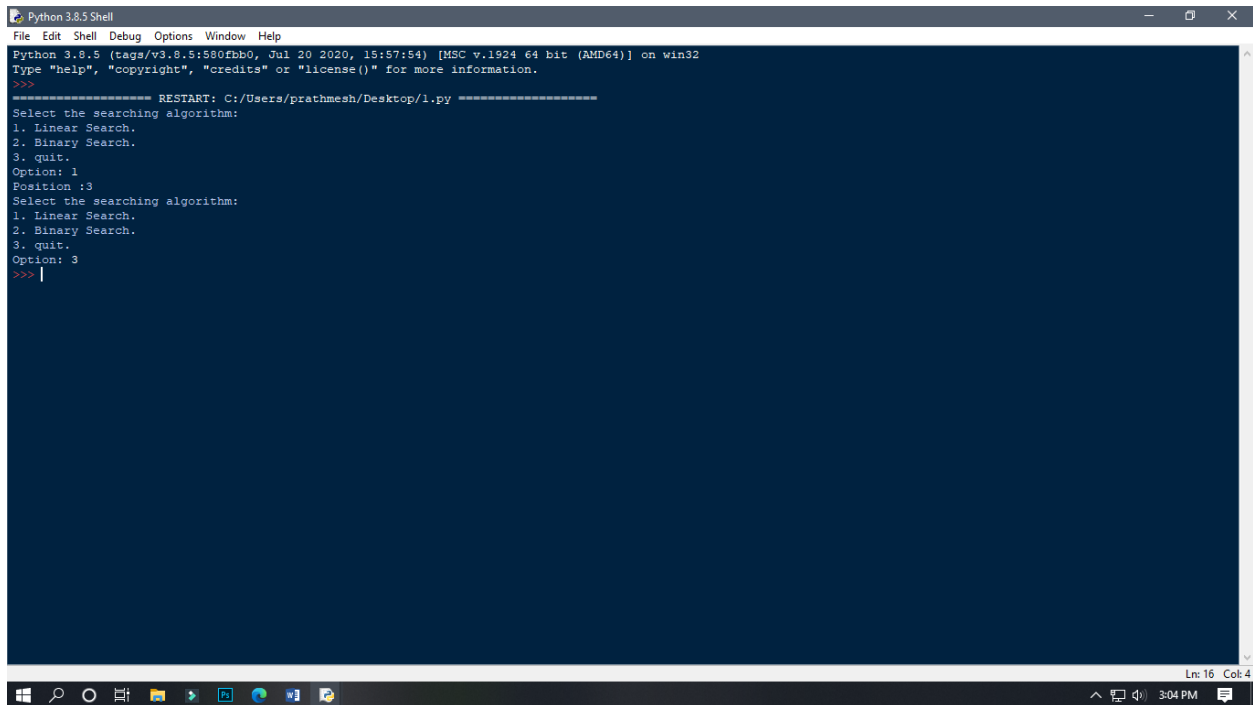
 def __init__(self,array,number):
 self.lst = sorted(array)
 self.number = number
 def binary_search(self,lst,n,start,end):

 if start <= end:
 mid = (end + start) // 2
 if lst[mid] == n:
 return f'position: {mid}'
 elif lst[mid] > n:
 return binary_search(lst,n,start,mid-1)
 else:
 return binary_search(lst,n,mid + 1,end)
 else:
 return -1

 def linear_search(self,lst,n):
 for i in range(len(lst)):
 if lst[i] == n:
 return f'Position :{i}'
 return -1

 def run_search(self):
 while True:
 print('Select the searching algorithm:')
 print('1. Linear Search.')
 print('2. Binary Search.')
 print('3. quit.')
 opt = int(input('Option: '))
 if opt == 2:
 print(search.binary_search(self.lst,self.number,0,len(lst)-1))
 elif opt == 1:
 print(search.linear_search(self.lst,self.number))
 else:
 break

lst = [1,2,3,4,5,6,7,8]
number = 4
search = Array(lst,number)`



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/prathmesh/Desktop/1.py =====
Select the searching algorithm:
1. Linear Search.
2. Binary Search.
3. quit.
Option: 1
Position :3
Select the searching algorithm:
1. Linear Search.
2. Binary Search.
3. quit.
Option: 3
>>> |
```

## Practical 6

**Aim:** WAP to sort a list of elements. Give user the option to perform sorting using Insertion sort, Bubble sort or Selection sort.

**Theory:** Bubble Sort: Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.

**Selection Sort:** The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array

**Insertion Sort:** Insertion sort iterates, consuming one input element each repetition, and growing a sorted output list. At each iteration, insertion sort removes one element from the input data, finds the location it belongs within the sorted list, and inserts it there. It repeats until no input elements remain.

```
1.py - C:/Users/prathmesh/Desktop/1.py (3.8.5)
File Edit Format Run Options Window Help

class Sorting:

    def __init__(self, lst):
        self.lst = lst

    def bubble_sort(self, lst):
        for i in range(len(lst)):
            for j in range(len(lst)):
                if lst[i] < lst[j]:
                    lst[i], lst[j] = lst[j], lst[i]
                else:
                    pass
            return lst

    def selection_sort(self, lst):
        for i in range(len(lst)):
            smallest_element = i
            for j in range(i+1, len(lst)):
                if lst[smallest_element] > lst[j]:
                    smallest_element = j
            lst[i], lst[smallest_element] = lst[smallest_element], lst[i]
            return lst

    def insertion_sort(self, lst):
        for i in range(1, len(lst)):
            index = lst[i]
            j = i-1
            while j >= 0 and index < lst[j]:
                lst[j+1] = lst[j]
                j -= 1
            lst[j+1] = index
            return lst

    def run_sort(self):
        while True:
            print('Select the sorting algorithm:')
            print('1. Bubble Sort.')
            print('2. Selection Sort.')
            print('3. Insertion Sort.')
            print('4. Quit')
            opt = int(input('Option: '))
            if opt == 1:
                pass
                return lst

            def selection_sort(self, lst):
                for i in range(len(lst)):
                    smallest_element = i
                    for j in range(i+1, len(lst)):
                        if lst[smallest_element] > lst[j]:
                            smallest_element = j
                    lst[i], lst[smallest_element] = lst[smallest_element], lst[i]
                    return lst

            def insertion_sort(self, lst):
                for i in range(1, len(lst)):
                    index = lst[i]
                    j = i-1
                    while j >= 0 and index < lst[j]:
                        lst[j+1] = lst[j]
                        j -= 1
                    lst[j+1] = index
                    return lst

            def run_sort(self):
                while True:
                    print('Select the sorting algorithm:')
                    print('1. Bubble Sort.')
                    print('2. Selection Sort.')
                    print('3. Insertion Sort.')
                    print('4. Quit')
                    opt = int(input('Option: '))
                    if opt == 1:
                        print(sort.bubble_sort(self.lst))
                    elif opt == 2:
                        print(sort.selection_sort(self.lst))
                    elif opt == 3:
                        print(sort.insertion_sort(self.lst))
                    else:
                        break
lst = [4, 2, 3, 9, 12, 1]
sort = Sorting(lst)
sort.run_sort()

Ln: 37 Col: 0
```

```
1.py - C:/Users/prathmesh/Desktop/1.py (3.8.5)
File Edit Format Run Options Window Help

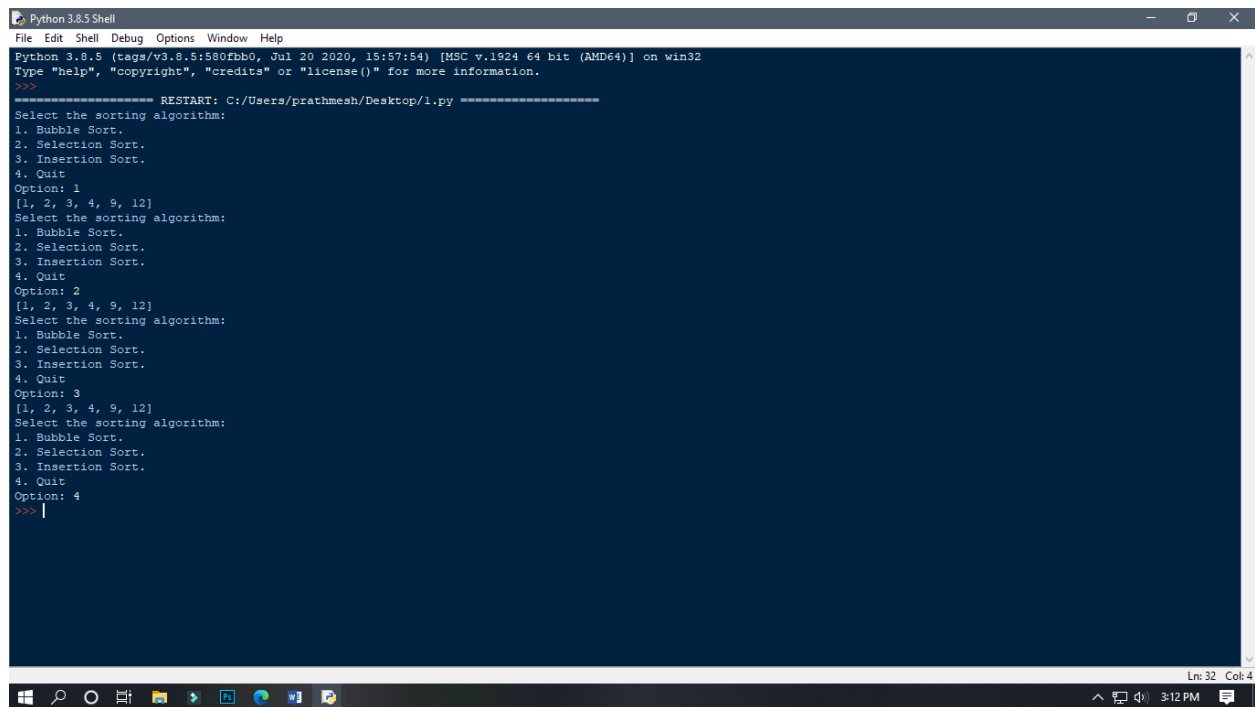
                pass
                return lst

            def selection_sort(self, lst):
                for i in range(len(lst)):
                    smallest_element = i
                    for j in range(i+1, len(lst)):
                        if lst[smallest_element] > lst[j]:
                            smallest_element = j
                    lst[i], lst[smallest_element] = lst[smallest_element], lst[i]
                    return lst

            def insertion_sort(self, lst):
                for i in range(1, len(lst)):
                    index = lst[i]
                    j = i-1
                    while j >= 0 and index < lst[j]:
                        lst[j+1] = lst[j]
                        j -= 1
                    lst[j+1] = index
                    return lst

            def run_sort(self):
                while True:
                    print('Select the sorting algorithm:')
                    print('1. Bubble Sort.')
                    print('2. Selection Sort.')
                    print('3. Insertion Sort.')
                    print('4. Quit')
                    opt = int(input('Option: '))
                    if opt == 1:
                        print(sort.bubble_sort(self.lst))
                    elif opt == 2:
                        print(sort.selection_sort(self.lst))
                    elif opt == 3:
                        print(sort.insertion_sort(self.lst))
                    else:
                        break
lst = [4, 2, 3, 9, 12, 1]
sort = Sorting(lst)
sort.run_sort()

Ln: 53 Col: 0
```



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/prathmesh/Desktop/l.py =====
Select the sorting algorithm:
1. Bubble Sort.
2. Selection Sort.
3. Insertion Sort.
4. Quit
Option: 1
[1, 2, 3, 4, 9, 12]
Select the sorting algorithm:
1. Bubble Sort.
2. Selection Sort.
3. Insertion Sort.
4. Quit
Option: 2
[1, 2, 3, 4, 9, 12]
Select the sorting algorithm:
1. Bubble Sort.
2. Selection Sort.
3. Insertion Sort.
4. Quit
Option: 3
[1, 2, 3, 4, 9, 12]
Select the sorting algorithm:
1. Bubble Sort.
2. Selection Sort.
3. Insertion Sort.
4. Quit
Option: 4
>>> |
```

## Practical 7 a

**Aim :** Write a program to implement the collision technique.

**Theory: Collisions:** A Hash Collision Attack is an attempt to find two input strings of a hash function that produce the same hashresult. If two separate inputs produce the same hashoutput, it is called a collision.

**Separate Chaining:** The idea is to make each cell of hash table point to a linked list of records that have same hash function value.

**Open Addressing:** Like separate chaining, open addressing is a method for handling collisions. In Open Addressing, all elements are stored in the hash table itself. So at any point, the size of the table must be greater than or equal to the total number of keys (Note that we can increase table size by copying old data if needed).

```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/prathmesh/Desktop/l.py =====
Before : [None, None, None, None]
Collision detected
Collision detected
After: [None, 1, None, 23]
>>>
```

```
l.py - C:/Users/prathmesh/Desktop/l.py (3.8.5)
File Edit Format Run Options Window Help
class Hash:
    def __init__(self, keys, lowerrange, higherrange):
        self.value = self.hashfunction(keys, lowerrange, higherrange)

    def get_key_value(self):
        return self.value

    def hashfunction(self, keys, lowerrange, higherrange):
        if lowerrange == 0 and higherrange > 0:
            return keys[higherrange]

if __name__ == '__main__':
    list_of_keys = [23, 43, 1, 87]
    list_of_list_index = [None, None, None, None]
    print("Before: " + str(list_of_list_index))
    for value in list_of_keys:
        present = Hash(value, 0, len(list_of_keys)).get_key_value()
        list_index = Hash(value, 0, len(list_of_keys)).get_key_value()
        if list_of_list_index[list_index]:
            print("Collision detected")
        else:
            list_of_list_index[list_index] = value
    print("After: " + str(list_of_list_index))
```

## Practical 7b

**Aim :** Write a program to implement the concept of linear probing.

**Theory :** Linear probing is a scheme in computer programming for resolving collisions in hash tables, data structures for maintaining a collection of key–value pairs and looking up the value associated with a given key. ... Along with quadratic probing and double hashing, linear probing is a form of open addressing.

```

1.py - C:/Users/prathmesh/Desktop/1.py (3.8.5)
File Edit Format Run Options Window Help
class Hash:
    def __init__(self, keys, lowerrange, higherrange):
        self.value = self.hashfunction(keys, lowerrange, higherrange)

    def get_key_value(self):
        return self.value

    def hashfunction(self, keys, lowerrange, higherrange):
        if lowerrange == 0 and higherrange > 0:
            return keys % (higherrange)

if __name__ == '__main__':
    linear_probing = True
    list_of_keys = [23, 43, 1, 87]
    list_of_list_index = [None, None, None, None]
    print("Before : " + str(list_of_list_index))
    for value in list_of_keys:
        #print(Hash(value, 0, len(list_of_keys)).get_key_value())
        list_index = Hash(value, 0, len(list_of_keys)).get_key_value()
        print("hash value for " + str(value) + " is : " + str(list_index))
        if list_of_list_index[list_index]:
            print("Collision detected for " + str(value))
            if linear_probing:
                old_list_index = list_index
                if list_index == len(list_of_list_index) - 1:
                    list_index = 0
                else:
                    list_index += 1
                list_full = False
                while list_of_list_index[list_index]:
                    if list_index == old_list_index:
                        list_full = True
                        break
                if list_index + 1 == len(list_of_list_index):
                    list_index = 0
                else:
                    list_index += 1
            if list_full:
                print("List was full . Could not save")
            else:
                list_of_list_index[list_index] = value

Ln: 47 Col: 0

1.py - C:/Users/prathmesh/Desktop/1.py (3.8.5)
File Edit Format Run Options Window Help
        return self.value

    def hashfunction(self, keys, lowerrange, higherrange):
        if lowerrange == 0 and higherrange > 0:
            return keys % (higherrange)

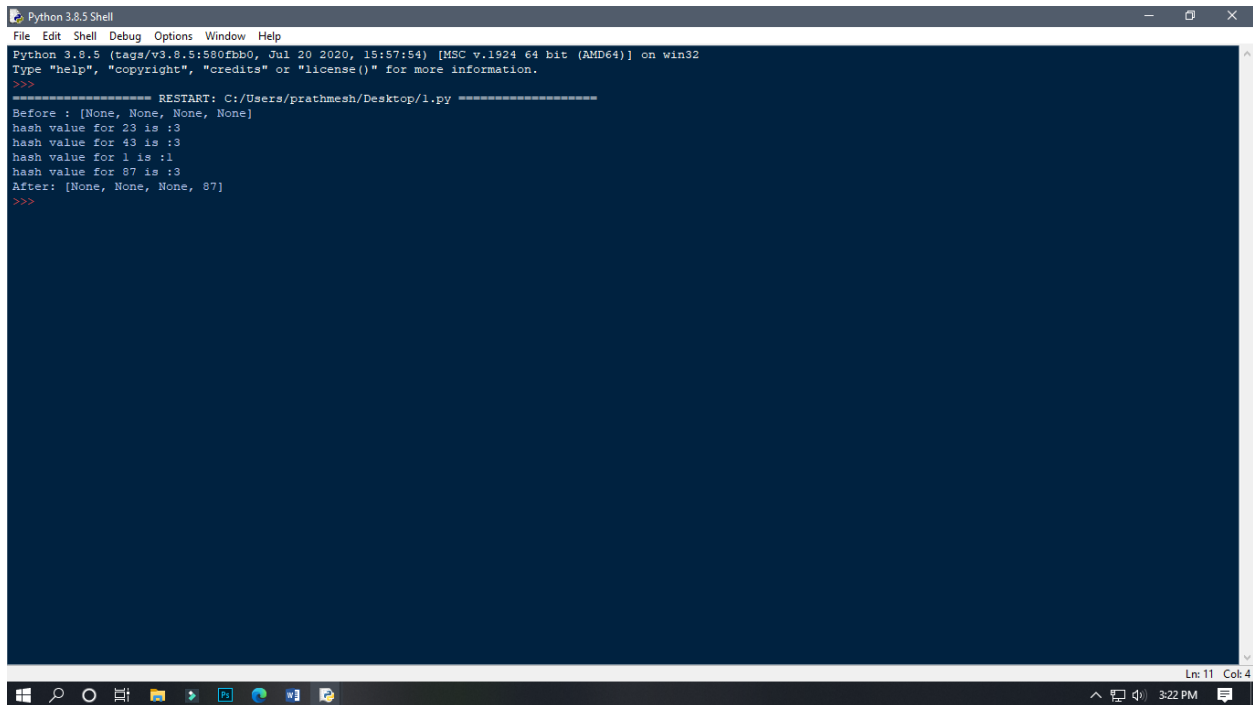
if __name__ == '__main__':
    linear_probing = True
    list_of_keys = [23, 43, 1, 87]
    list_of_list_index = [None, None, None, None]
    print("Before : " + str(list_of_list_index))
    for value in list_of_keys:
        #print(Hash(value, 0, len(list_of_keys)).get_key_value())
        list_index = Hash(value, 0, len(list_of_keys)).get_key_value()
        print("hash value for " + str(value) + " is : " + str(list_index))
        if list_of_list_index[list_index]:
            print("Collision detected for " + str(value))
            if linear_probing:
                old_list_index = list_index
                if list_index == len(list_of_list_index) - 1:
                    list_index = 0
                else:
                    list_index += 1
                list_full = False
                while list_of_list_index[list_index]:
                    if list_index == old_list_index:
                        list_full = True
                        break
                if list_index + 1 == len(list_of_list_index):
                    list_index = 0
                else:
                    list_index += 1
            if list_full:
                print("List was full . Could not save")
            else:
                list_of_list_index[list_index] = value

    else:
        list_of_list_index[list_index] = value

    print("After: " + str(list_of_list_index))

Ln: 47 Col: 0

```



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/prathmesh/Desktop/l.py =====
Before : [None, None, None, None]
hash value for 23 is :3
hash value for 43 is :3
hash value for 1 is :1
hash value for 87 is :3
After: [None, None, None, 87]
>>>
```

## Practical 8

Aim : Write a program for inorder, postorder and preorder traversal of tree.

Theory : Inorder: In case of binary search trees (BST), Inorder traversal gives nodes in non-decreasing order. To get nodes of BST in non-increasing order, a variation of Inorder traversal where Inorder traversal s reversed can be used.

Preorder: Preorder traversal is used to create a copy of the tree. Preorder traversal is also used to get prefix expression on of an expression tree. Postorder traversal is also useful to get the postfix expression of an expression tree.



```

1.py - C:/Users/prathmesh/Desktop/1.py (3.8.5)
File Edit Format Run Options Window Help

class Node:
    def __init__(self, key):
        self.left = None
        self.right = None
        self.value = key

    def PrintTree(self):
        if self.left:
            self.left.PrintTree()
        print(self.value)
        if self.right:
            self.right.PrintTree()

    def Printpreorder(self):
        if self.value:
            print(self.value)
            if self.left:
                self.left.Printpreorder()
            if self.right:
                self.right.Printpreorder()

    def Printinorder(self):
        if self.value:
            if self.left:
                self.left.Printinorder()
            print(self.value)
            if self.right:
                self.right.Printinorder()

    def Printpostorder(self):
        if self.value:
            if self.left:
                self.left.Printpostorder()
            if self.right:
                self.right.Printpostorder()
            print(self.value)

    def insert(self, data):
        if self.value:
            if data < self.value:
                if self.left is None:
                    self.left = Node(data)

```

```

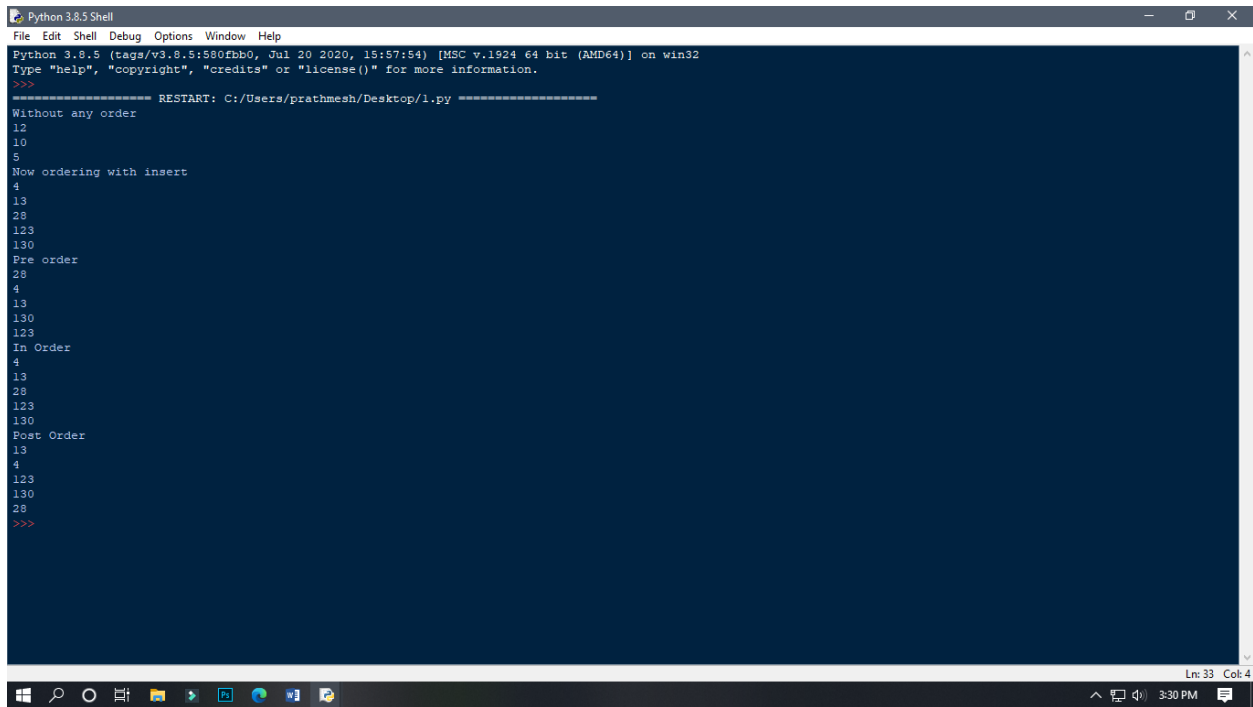
1.py - C:/Users/prathmesh/Desktop/1.py (3.8.5)
File Edit Format Run Options Window Help

        self.left.Printpostorder()
    if self.right:
        self.right.Printpostorder()
    print(self.value)

    def insert(self, data):
        if self.value:
            if data < self.value:
                if self.left is None:
                    self.left = Node(data)
                else:
                    self.left.insert(data)
            elif data > self.value:
                if self.right is None:
                    self.right = Node(data)
                else:
                    self.right.insert(data)
            else:
                self.value = data

if __name__ == '__main__':
    root = Node(10)
    root.left = Node(12)
    root.right = Node(5)
    print("Without any order")
    root.PrintTree()
    root_l = Node(None)
    root_l.insert(28)
    root_l.insert(4)
    root_l.insert(13)
    root_l.insert(130)
    root_l.insert(123)
    print("Now ordering with insert")
    root_l.PrintTree()
    print("Pre order")
    root_l.Printpreorder()
    print("In Order")
    root_l.Printinorder()
    print("Post Order")
    root_l.Printpostorder()

```



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/prathmesh/Desktop/l.py =====
Without any order
12
10
5
Now ordering with insert
4
13
28
123
130
Pre order
28
4
13
130
123
In Order
4
13
28
123
130
Post Order
13
4
123
130
28
>>>
```