

Name: Vikas Mourya
Class: SYCS
Old Roll No: 4057
New Roll No: 333
Subject: DS
Git Hub Link: <https://github.com/thisisvikasmourya/DS>

Practical No:1-A

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:

Array is a container which can hold a fix number of items and these items should be of the same type. Most of the data structures make use of arrays to implement their algorithms. Following are the important terms to understand the concept of Array.

Element– Each item stored in an array is called an element.

Index – Each location of an element in an array has a numerical index, which is used to identify the element.

Basic Operations

Following are the basic operations supported by an array.

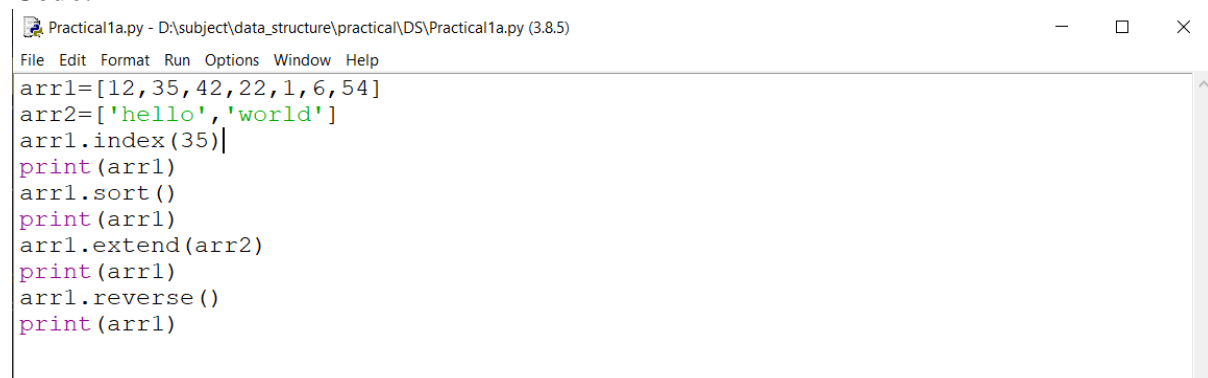
Traverse – print all the array elements one by one.

Insertion – Adds an element at the given index.

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

Sorting-Means Arranging the Element in particular order. i.e. Ascending or Descending order

Code:



```
Practical1a.py - D:\subject\data_structure\practical\DS\Practical1a.py (3.8.5)
File Edit Format Run Options Window Help
arr1=[12,35,42,22,1,6,54]
arr2=['hello','world']
arr1.index(35)
print(arr1)
arr1.sort()
print(arr1)
arr1.extend(arr2)
print(arr1)
arr1.reverse()
print(arr1)
```

Name: Vikas Mourya

Class: SYCS

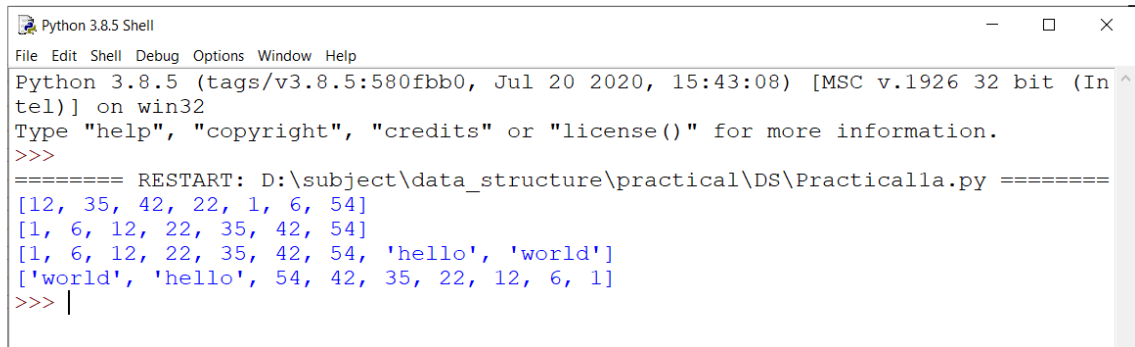
Old Roll No: 4057

New Roll No: 333

Subject: DS

Git Hub Link: <https://github.com/thisisvikasmourya/DS>

Output:

A screenshot of a Python 3.8.5 Shell window. The window title is "Python 3.8.5 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area shows the following content: "Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (Intel)] on win32", "Type \"help\", \"copyright\", \"credits\" or \"license()\" for more information.", and a prompt ">>>". Below the prompt, there is a line "===== RESTART: D:\subject\data_structure\practical\DS\Practicalla.py =====" followed by four lines of Python code: "[12, 35, 42, 22, 1, 6, 54]", "[1, 6, 12, 22, 35, 42, 54]", "[1, 6, 12, 22, 35, 42, 54, 'hello', 'world']", and ["'world', 'hello', 54, 42, 35, 22, 12, 6, 1]". The prompt ">>>" is followed by a vertical cursor bar.

```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\subject\data_structure\practical\DS\Practicalla.py =====
[12, 35, 42, 22, 1, 6, 54]
[1, 6, 12, 22, 35, 42, 54]
[1, 6, 12, 22, 35, 42, 54, 'hello', 'world']
['world', 'hello', 54, 42, 35, 22, 12, 6, 1]
>>> |
```

Name: Vikas Mourya

Class: SYCS

Old Roll No: 4057

New Roll No: 333

Subject: DS

Git Hub Link: <https://github.com/thisisvikasmourya/DS>

Practical No: 1-B

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

Theory:

Matrix is a special case of two dimensional array where each data element is of strictly same size. So every matrix is also a two dimensional array but not vice versa. Matrices are very important data structures for many mathematical and scientific calculations. As we have already discussed two dimensional array data structure in matrices.

In Python, we can implement a matrix as nested list (list inside a list).

We can treat each element as a row of the matrix.

For example `X = [[1, 2], [4, 5], [3, 6]]` would represent a `3x2` matrix.

The first row can be selected as `X[0]`. And, the element in first row, first column can be selected as `X[0][0]`.

Multiplication of two matrices `X` and `Y` is defined only if the number of columns in `X` is equal to the number of rows `Y`.

If `X` is a `n x m` matrix and `Y` is a `m x l` matrix then, `XY` is defined and has the dimension `n x l` (but `YX` is not defined). Here are a couple of ways to implement matrix multiplication in Python.

Name: Vikas Mourya

Class: SYCS

Old Roll No: 4057

New Roll No: 333

Subject: DS

Git Hub Link: <https://github.com/thisisvikasmourya/DS>

Code:

```
Practical1b.py - D:\subject\data_structure\practical\DS\Practical1b.py (3.8.5)
File Edit Format Run Options Window Help
# this is practical 1b
X = [[11,7,3],
      [4,5,6],
      [7,8,9]]

Y = [[5,8,1],
      [6,7,3],
      [4,5,9]]

result = [[0,0,0],
           [0,0,0],
           [0,0,0]]

for i in range(len(X)):
    for j in range(len(X[0])):
        result[i][j] = X[i][j] + Y[i][j]
        for r in result:
            print(r)

X = [[12,7,3],
      [4,5,6],
      [7,8,9]]
# 3x4 matrix
Y = [[5,8,1,2],
      [6,7,3,0],
      [4,5,9,1]]
# result is 3x4
result = [[0,0,0,0],
           [0,0,0,0],
           [0,0,0,0]]

# iterate through rows of X
for i in range(len(X)):
# iterate through columns of Y
    for j in range(len(Y[0])):
# iterate through rows of Y
        for k in range(len(Y)):
            result[i][j] += X[i][k] * Y[k][j]
```

Name: Vikas Mourya

Class: SYCS

Old Roll No: 4057

New Roll No: 333

Subject: DS

Git Hub Link: <https://github.com/thisisvikasmourya/DS>

```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\subject\data_structure\practical\DS\Practical1b.py =====
[16, 0, 0]
[0, 0, 0]
[0, 0, 0]
[16, 15, 0]
[0, 0, 0]
[0, 0, 0]
[16, 15, 4]
[0, 0, 0]
[0, 0, 0]
[16, 15, 4]
[10, 0, 0]
[0, 0, 0]
[16, 15, 4]
[10, 12, 0]
[0, 0, 0]
[16, 15, 4]
[10, 12, 9]
[0, 0, 0]
[16, 15, 4]
[10, 12, 9]
[11, 0, 0]
[16, 15, 4]
[10, 12, 9]
[11, 13, 0]
[16, 15, 4]
[10, 12, 9]
[11, 13, 18]
[60, 0, 0, 0]
[0, 0, 0, 0]
[0, 0, 0, 0]
[102, 0, 0, 0]
[0, 0, 0, 0]
[0, 0, 0, 0]
r114 0 0 0
```

Output:

Name: Vikas Mourya

Class: SYCS

Old Roll No: 4057

New Roll No: 333

Subject: DS

Git Hub Link: <https://github.com/thisisvikasmourya/DS>

Practical No: 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.

Name: Vikas Mourya

Class: SYCS

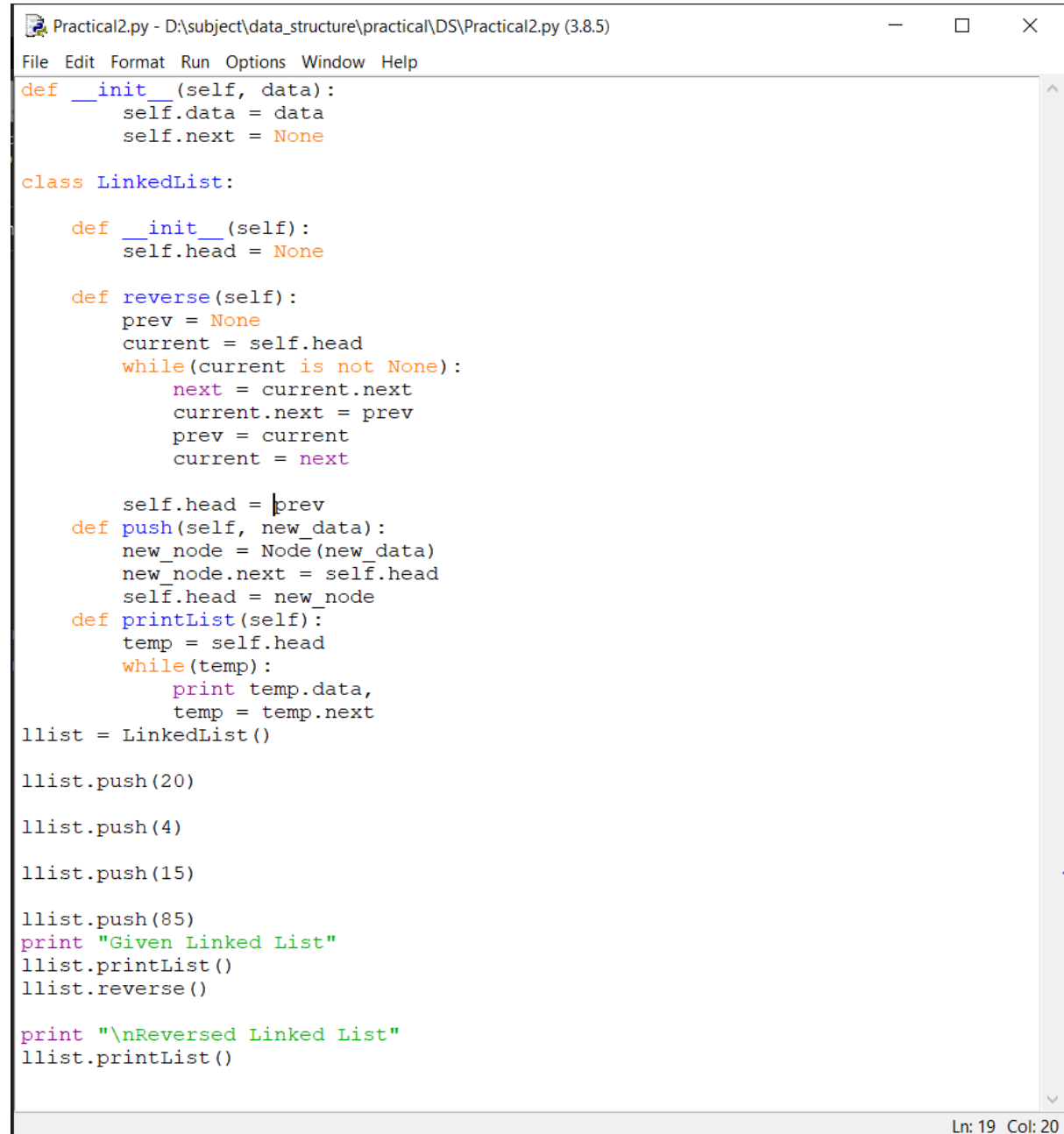
Old Roll No: 4057

New Roll No: 333

Subject: DS

Git Hub Link: <https://github.com/thisisvikasmourya/DS>

Code:

A screenshot of a Python IDE window titled 'Practical2.py - D:\subject\data_structure\practical\DS\Practical2.py (3.8.5)'. The window contains a Python script for a linked list. The script defines a Node class and a LinkedList class. The LinkedList class has methods for __init__, reverse, push, and printList. The main code creates a LinkedList object, pushes values 20, 4, 15, and 85, prints the list, reverses it, and prints it again. The status bar at the bottom right shows 'Ln: 19 Col: 20'.

```
Practical2.py - D:\subject\data_structure\practical\DS\Practical2.py (3.8.5)
File Edit Format Run Options Window Help

def __init__(self, data):
    self.data = data
    self.next = None

class LinkedList:

    def __init__(self):
        self.head = None

    def reverse(self):
        prev = None
        current = self.head
        while(current is not None):
            next = current.next
            current.next = prev
            prev = current
            current = next

        self.head = prev

    def push(self, new_data):
        new_node = Node(new_data)
        new_node.next = self.head
        self.head = new_node

    def printList(self):
        temp = self.head
        while(temp):
            print temp.data,
            temp = temp.next

l1list = LinkedList()

l1list.push(20)

l1list.push(4)

l1list.push(15)

l1list.push(85)
print "Given Linked List"
l1list.printList()
l1list.reverse()

print "\nReversed Linked List"
l1list.printList()

Ln: 19 Col: 20
```

Name: Vikas Mourya

Class: SYCS

Old Roll No: 4057

New Roll No: 333

Subject: DS

Git Hub Link: <https://github.com/thisisvikasmourya/DS>

Practical No: 3-A

Aim: Implement the following for Stack

Theory:

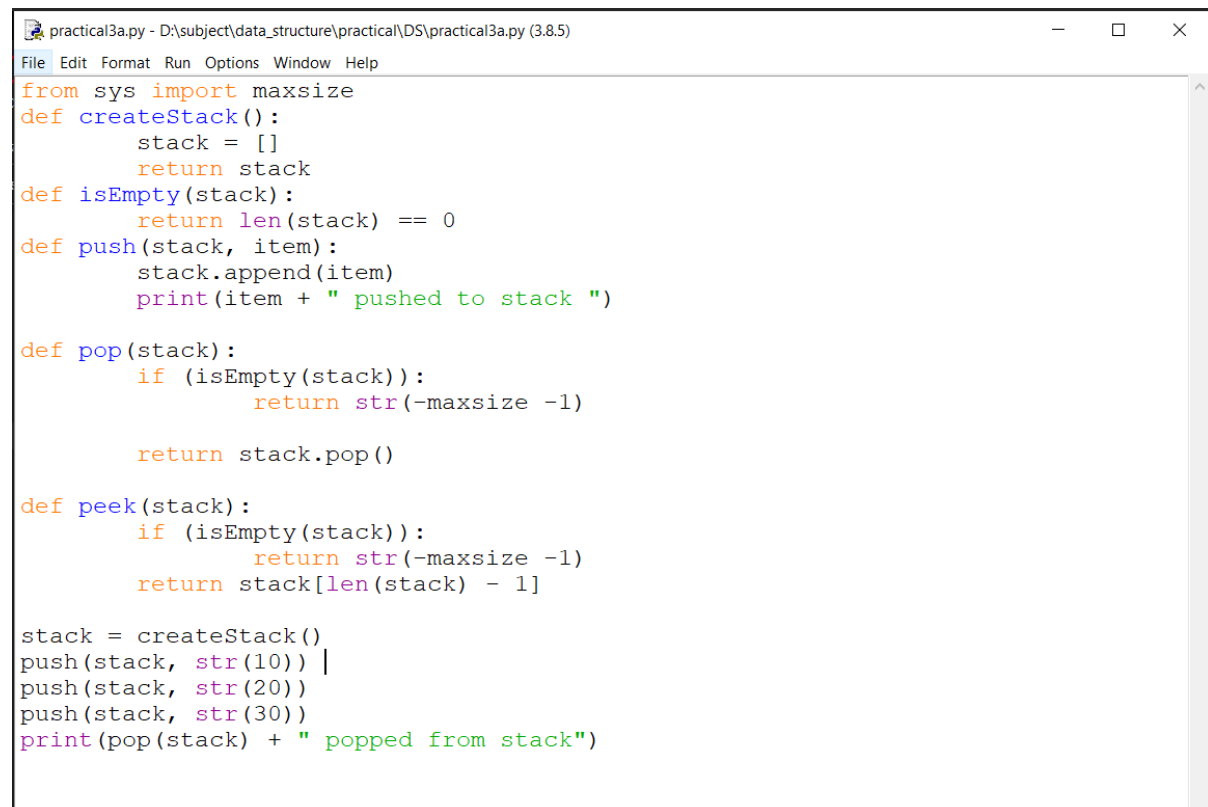
Array is a container which can hold a fix number of items and these items should be of the same type. Most of the data structures make use of arrays to implement their algorithms.

Following are the important terms to understand the concept of Array.

Element– Each item stored in an array is called an element.

Index – Each location of an element in an array has a numerical index, which is used to identify the element.

Code:

A screenshot of a Python IDE window titled 'practical3a.py - D:\subject\data_structure\practical\DS\practical3a.py (3.8.5)'. The window contains Python code for implementing a stack using a list. The code defines functions for creating a stack, checking if it's empty, pushing elements, popping elements, and peeking at the top element. It then demonstrates the stack operations by creating a stack, pushing 10, 20, and 30, and finally popping an element and printing the result.

```
from sys import maxsize
def createStack():
    stack = []
    return stack
def isEmpty(stack):
    return len(stack) == 0
def push(stack, item):
    stack.append(item)
    print(item + " pushed to stack ")

def pop(stack):
    if (isEmpty(stack)):
        return str(-maxsize -1)

    return stack.pop()

def peek(stack):
    if (isEmpty(stack)):
        return str(-maxsize -1)
    return stack[len(stack) - 1]

stack = createStack()
push(stack, str(10)) |
push(stack, str(20))
push(stack, str(30))
print(pop(stack) + " popped from stack")
```


Name: Vikas Mourya

Class: SYCS

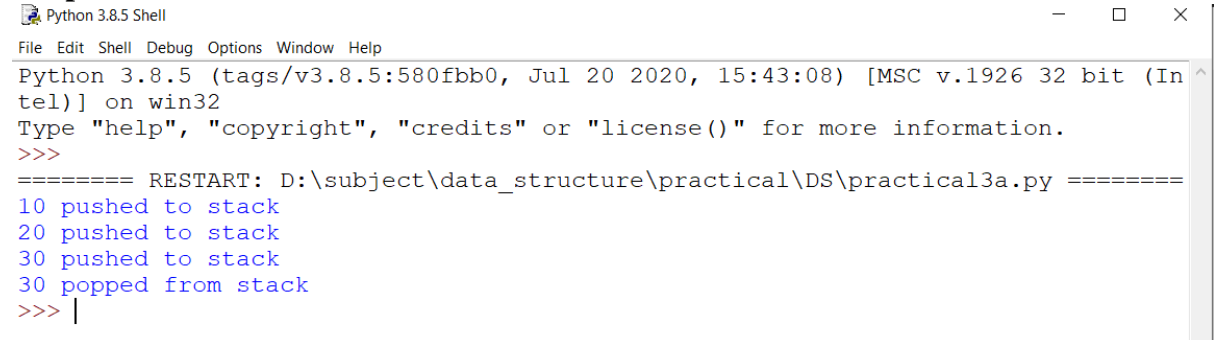
Old Roll No: 4057

New Roll No: 333

Subject: DS

Git Hub Link: <https://github.com/thisisvikasmourya/DS>

Output:

A screenshot of a Python 3.8.5 Shell window. The window title is "Python 3.8.5 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The status bar at the bottom shows "Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (Intel)] on win32". The main text area displays the following output:

```
>>> Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\subject\data_structure\practical\DS\practical3a.py =====
10 pushed to stack
20 pushed to stack
30 pushed to stack
30 popped from stack
>>> |
```

Name: Vikas Mourya

Class: SYCS

Old Roll No: 4057

New Roll No: 333

Subject: DS


Git Hub Link: <https://github.com/thisisvikasmourya/DS>

Practical No: 3-B

Aim:

Theory:

Code:

 P3-B.py - D:/ARYA VIRA/Gangashankar Sir/Practicals/Journal/P3-B/P3-B.py (3.8.5)


File Edit Format Run Options Window Help

```
#Name: Aarya Vira
#Roll No: 374

def TowerOfHanoi(n , source, destination, auxiliary):
    if n==1:
        print ("Move disk 1 from source",source,"to destination",destination )
        return
    TowerOfHanoi(n-1, source, auxiliary, destination)
    print ("Move disk",n,"from source",source,"to destination",destination )
    TowerOfHanoi(n-1, auxiliary, destination, source)

n = 4
TowerOfHanoi(n, 'A', 'B', 'C')
|
```

Output:

 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: D:/ARYA VIRA/Gangashankar Sir/Practicals/Journal/P3-B/P3-B.py ====
Move disk 1 from source A to destination C
Move disk 2 from source A to destination B
Move disk 1 from source C to destination B
Move disk 3 from source A to destination C
Move disk 1 from source B to destination A
Move disk 2 from source B to destination C
Move disk 1 from source A to destination C
Move disk 4 from source A to destination B
Move disk 1 from source C to destination B
Move disk 2 from source C to destination A
Move disk 1 from source B to destination A
Move disk 3 from source C to destination B
Move disk 1 from source A to destination C
Move disk 2 from source A to destination B
Move disk 1 from source C to destination B
>>> |
```

Full-screen S

Name: Vikas Mourya

Class: SYCS

Old Roll No: 4057

New Roll No: 333

Subject: DS

Git Hub Link: <https://github.com/thisisvikasmourya/DS>

Practical No: 3-C

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

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.

Code:

```
*Practical 3c.py - D:\subject\data_structure\practical\DS\Practical 3c.py (3.8.5)*
File Edit Format Run Options Window Help
def add(A, B, m, n):
    size = max(m, n)
    sum = [0 for i in range(size)]
    for i in range(0, m, 1):
        sum[i] = A[i]
    for i in range(n):
        sum[i] += B[i]
    return sum
def printPoly(poly, n):
    for i in range(n):
        print(poly[i], end = "")
        if (i != 0):
            print("x^", i, end = "")
        if (i != n - 1):
            print(" + ", end = "")
if __name__ == '__main__':
    A = [5, 0, 10, 6]
    B = [1, 2, 4]
    m = len(A)
    n = len(B)
    print("First polynomial is")
    printPoly(A, m)
    print("\n", end = "")
    print("Second polynomial is")
    printPoly(B, n)
    print("\n", end = "")
    sum = add(A, B, m, n)
    size = max(m, n)
    print("sum polynomial is")
    printPoly(sum, size)
```

Name: Vikas Mourya

Class: SYCS

Old Roll No: 4057

New Roll No: 333

Subject: DS

Git Hub Link: <https://github.com/thisisvikasmourya/DS>

Output:

```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\subject\data_structure\practical\DS\Practical 3c.py
First polynomial is
5 + 0x^ 1 + 10x^ 2 + 6x^ 3
Second polynomial is
1 + 2x^ 1 + 4x^ 2
sum polynomial is
6 + 2x^ 1 + 14x^ 2 + 6x^ 3
>>> |
```

Name: Vikas Mourya

Class: SYCS

Old Roll No: 4057

New Roll No: 333

Subject: DS

Git Hub Link: <https://github.com/thisisvikasmourya/DS>

Practical No: 3-D(PART-I)

Aim:

Theory:

Code:

```
practical 3d.py - D:\subject\data_structure\practical\DS\practical 3d.py (3.8.5)
File Edit Format Run Options Window Help
def recur_factorial(n):
    if n == 1:
        return n
    else:
        return n*recur_factorial(n-1)

num = int(input("Enter a number: "))

if num < 0:
    print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
    print("The factorial of 0 is 1")
else:
    print("The factorial of",num,"is",recur_factorial(num))
def factorial(n):

    fact = 1
    for i in range(1, n + 1):
        fact = fact * i
    return fact
if __name__ == '__main__':
    print("The Factorial of", n, "is", factorial(n))
#using iteration
def fact(number):
    fact = 1

    for number in range(5, 1,-1):

        fact = fact * number
    return fact

number = int(input("Enter a number for iteration : "))

factorial = fact(number)
print("Factorial is "+str(factorial))
```

Output:

```
*Python 3.8.5 Shell*
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\subject\data_structure\practical\DS\practical 3d.py =====
Enter a number: 45
The factorial of 45 is 119622220865480194561963161495657715064383733760000000000
Enter a number for iteration : |
```

Name: Vikas Mourya

Class: SYCS

Old Roll No: 4057

New Roll No: 333

Subject: DS

Git Hub Link: <https://github.com/thisisvikasmourya/DS>

Practical No: 4

Aim: Perform Queues operations using Circular Array implementation.

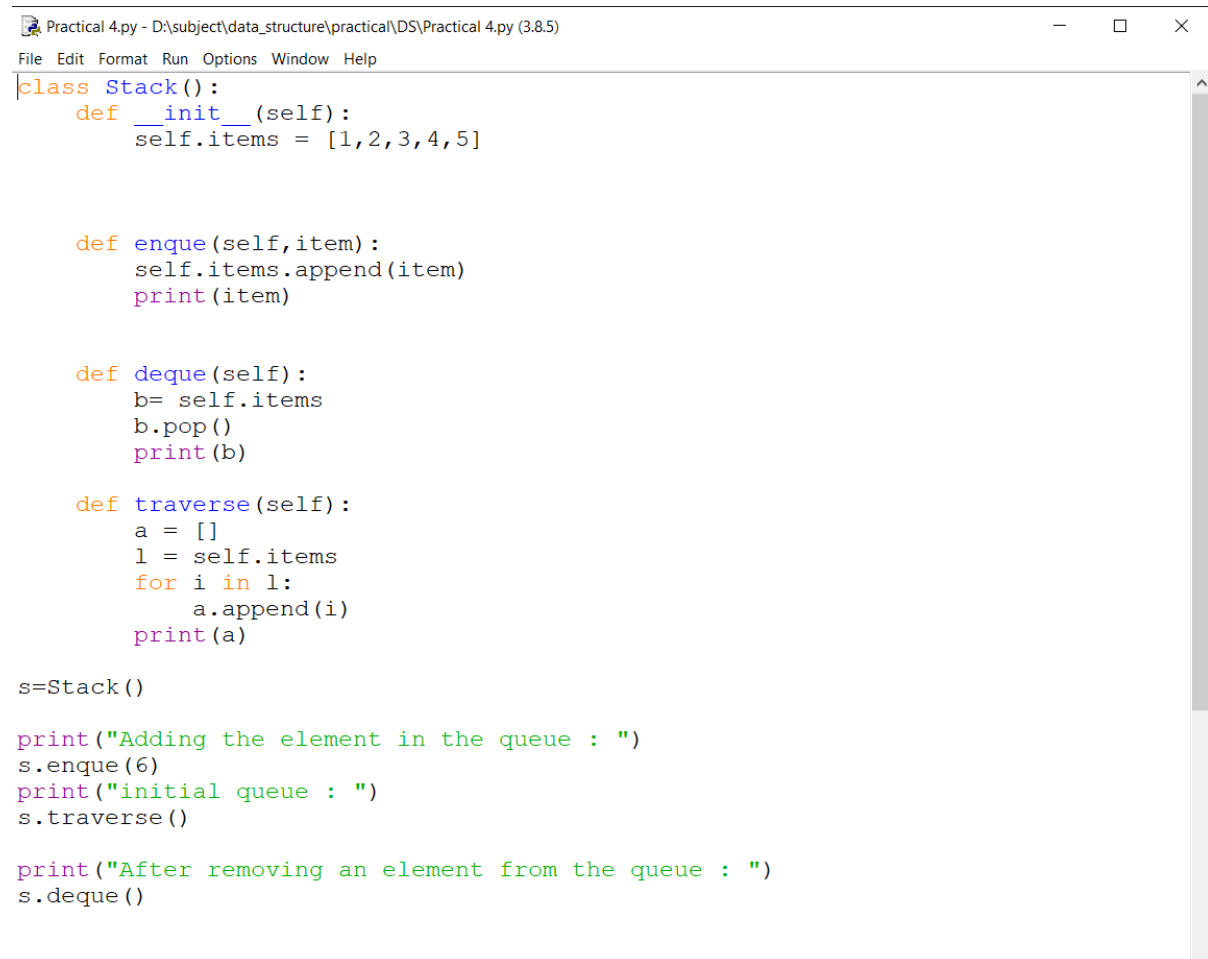
Theory:

A Circular Queue is a queue data structure but circular in shape, therefore after the last position, the next place in the queue is the first position.

We recommend you to first go through the Linear Queue tutorial before Circular queue, as we will be extending the same implementation.

In case of Linear queue, we did not have the head and tail pointers because we used python **List** for implementing it. But in case of a circular queue, as the size of the queue is fixed, hence we will set a maxSize for our list used for queue implementation.

Code:



```
Practical 4.py - D:\subject\data_structure\practical\DS\Practical 4.py (3.8.5)
File Edit Format Run Options Window Help

class Stack():
    def __init__(self):
        self.items = [1,2,3,4,5]

    def enqueue(self,item):
        self.items.append(item)
        print(item)

    def dequeue(self):
        b= self.items
        b.pop()
        print(b)

    def traverse(self):
        a = []
        l = self.items
        for i in l:
            a.append(i)
        print(a)

s=Stack()

print("Adding the element in the queue : ")
s.enqueue(6)
print("initial queue : ")
s.traverse()

print("After removing an element from the queue : ")
s.dequeue()
```

Name: Vikas Mourya

Class: SYCS

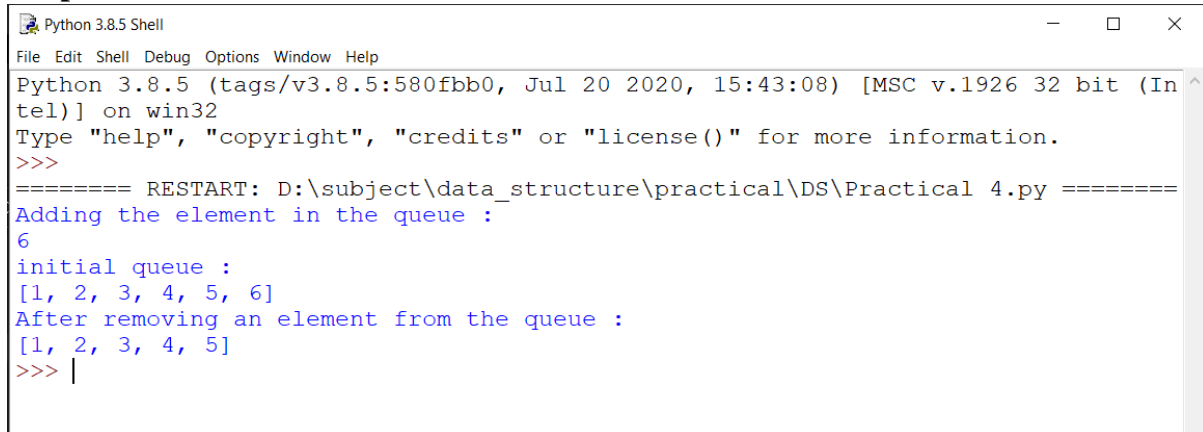
Old Roll No: 4057

New Roll No: 333

Subject: DS

Git Hub Link: <https://github.com/thisisvikasmourya/DS>

Output:



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\subject\data_structure\practical\DS\Practical 4.py =====
Adding the element in the queue :
6
initial queue :
[1, 2, 3, 4, 5, 6]
After removing an element from the queue :
[1, 2, 3, 4, 5]
>>> |
```

Name: Vikas Mourya

Class: SYCS

Old Roll No: 4057

New Roll No: 333

Subject: DS

Git Hub Link: <https://github.com/thisisvikasmourya/DS>

Practical No: 5

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

Theory:

Searching is a very basic necessity when you store data in different data structures. The simplest approach is to go across every element in the data structure and match it with the value you are searching for. This is known as Linear search. It is inefficient and rarely used, but creating a program for it gives an idea about how we can implement some advanced search algorithms.

Linear Search:

In this type of search, a sequential search is made over all items one by one. Every item is checked and if a match is found then that particular item is returned, otherwise the search continues till the end of the data structure.

Binary Search:

Search a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise narrow it to the upper half. Repeatedly check until the value is found or the interval is empty.

Name: Vikas Mourya

Class: SYCS

Old Roll No: 4057

New Roll No: 333

Subject: DS

Git Hub Link: <https://github.com/thisisvikasmourya/DS>

Code:

```
practical5.py - D:\subject\data_structure\practical\DS\practical5.py (3.8.5)
File Edit Format Run Options Window Help
list1 = [1,2,3,4,5,6,7,8,9,10]
print("List = ",list1)
size = len(list1)
def binary_search(x):
    print("BINARY SEARCHING")
    low = 0
    high = len(list1) - 1
    mid = 0
    while low <= high:
        mid = (high + low) // 2
        if list1[mid] < x:
            low = mid + 1
        elif list1[mid] > x:
            high = mid - 1
        else:
            return mid
    return "None it not in the list"

def linear_search(n):
    print("LINEAR SEARCHING")
    if n not in list1:
        print(n,"not in the list")
    else:
        for i in range(size):
            if list1[i]==n:
                print("index of ", n," is ",i)

n = input("Enter (L) for Linear search and (B) for Binary search :")
if n=="L" or n=="l":
    y = int(input("Enter a no. from the given list1 "))
    linear_search(y)
elif n=="B" or n=="b":
    y = int(input("Enter a no. from the given list1 "))
    print("index of ",y," is ",binary_search(y))
else:
    print("Invalid input")
```

Output:

```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\subject\data_structure\practical\DS\practical5.py =====
List = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Enter (L) for Linear search and (B) for Binary search :b
Enter a no. from the given list1 2
BINARY SEARCHING
index of 2 is 1
>>> |
```

Name: Vikas Mourya

Class: SYCS

Old Roll No: 4057

New Roll No: 333

Subject: DS

Git Hub Link: <https://github.com/thisisvikasmourya/DS>

Practical No: 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:

Sorting refers to arranging data in a particular format. Sorting algorithm specifies the way to arrange data in a particular order. Most common orders are in numerical or lexicographical order.

The importance of sorting lies in the fact that data searching can be optimized to a very high level, if data is stored in a sorted manner. Sorting is also used to represent data in more readable formats. Below we see five such implementations of sorting in python.

Bubble Sort

It is a comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order.

Insertion Sort

Insertion sort involves finding the right place for a given element in a sorted list. So in beginning we compare the first two elements and sort them by comparing them. Then we pick the third element and find its proper position among the previous two sorted elements. This way we gradually go on adding more elements to the already sorted list by putting them in their proper position.

Selection Sort

In selection sort we start by finding the minimum value in a given list and move it to a sorted list. Then we repeat the process for each of the remaining elements in the unsorted list. The next element entering the sorted list is compared with the existing elements and placed at its correct position. So at the end all the elements from the unsorted list are sorted.

Name: Vikas Mourya

Class: SYCS

Old Roll No: 4057

New Roll No: 333

Subject: DS

Git Hub Link: <https://github.com/thisisvikasmourya/DS>

Code:

```
*practical6.py - D:\subject\data_structure\practical\DS\practical6.py (3.8.5)*
File Edit Format Run Options Window Help
list1 = [5,1,78,3,45,12,4,15,44,34,62,54]
print("List = ",list1)
n = len(list1)
def bubbleSort():
    print("Bubble Sorting")
    for i in range(n-1):
        for j in range(0, n-i-1):
            if list1[j] > list1[j+1]:
                list1[j], list1[j+1] = list1[j+1], list1[j]
        print(list1)
def SelectionSort():
    print("Selection Sorting")
    for i in range(n):
        for j in range(i):
            if list1[i]<list1[j]:
                list1[i],list1[j] = list1[j],list1[i]
        print(list1)
def InsertionSort():
    print("Insertion Sorting")
    for i in range(1, n):
        c = list1[i]
        j = i-1
        while j >=0 and c < list1[j]:
            list1[j+1] = list1[j]
            j -= 1
        list1[j+1] = c
    print(list1)

inp = input("Enter (B) for Bubble Sort, (S) for elsection Sort and (I) for Inser
if inp=="B" or inp=="b":
    bubbleSort()
elif inp=="S" or inp=="s":
    SelectionSort()
elif inp=="I" or inp=="i":
    InsertionSort()
else:
    print("Invalid input")
```

Output:

```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\subject\data_structure\practical\DS\practical6.py =====
List = [5, 1, 78, 3, 45, 12, 4, 15, 44, 34, 62, 54]
Enter (B) for Bubble Sort, (S) for elsection Sort and (I) for Insertion Sort
Enter here:I
Insertion Sorting
[1, 3, 4, 5, 12, 15, 34, 44, 45, 54, 62, 78]
>>>
```

Name: Vikas Mourya

Class: SYCS

Old Roll No: 4057

New Roll No: 333

Subject: DS

Git Hub Link: <https://github.com/thisisvikasmourya/DS>

Practical No: 7-A

Aim:

Theory:

Code:

```
P7-A.py - D:\subject\data_structure\practical\P7-A.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:

        list_index = Hash(value, 0, len(list_of_keys)).get_key_value()
        if list_of_list_index[list_index]:
            print("Collission detected")
        else:
            list_of_list_index[list_index] = value

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

Output:

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

Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\subject\data_structure\practical\P7-A.py =====
Before : [None, None, None, None]
Collission detected
Collission detected
After: [None, 1, None, 23]
>>> |
```

Name: Vikas Mourya

Class: SYCS

Old Roll No: 4057

New Roll No: 333

Subject: DS

Git Hub Link: <https://github.com/thisisvikasmourya/DS>

Practical No: 7-B

Aim:

Theory:

Code:

Name: Vikas Mourya

Class: SYCS

Old Roll No: 4057

New Roll No: 333

Subject: DS

Git Hub Link: <https://github.com/thisisvikasmourya/DS>

```
*P7-B.py - D:\subject\data_structure\practical\P7-B.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, 32, 34, 67, 77, 45, 54]
    list_of_list_index = [None]*len(list_of_keys)
    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
                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
```

Name: Vikas Mourya

Class: SYCS

Old Roll No: 4057

New Roll No: 333

Subject: DS

Git Hub Link: <https://github.com/thisisvikasmourya/DS>

Output:

```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\subject\data_structure\practical\P7-B.py =====
Before : [None, None, None, None, None, None, None, None, None]
hash value for 23 is :3
hash value for 43 is :3
Collision detected for 43
hash value for 1 is :1
hash value for 87 is :7
hash value for 32 is :2
hash value for 34 is :4
Collision detected for 34
hash value for 67 is :7
Collision detected for 67
hash value for 77 is :7
Collision detected for 77
hash value for 45 is :5
Collision detected for 45
hash value for 54 is :4
Collision detected for 54
After: [54, 1, 32, 23, 43, 34, 45, 87, 67, 77]
>>> |
```

Name: Vikas Mourya

Class: SYCS

Old Roll No: 4057

New Roll No: 333

Subject: DS

Git Hub Link: <https://github.com/thisisvikasmourya/DS>

Practical No: 8

Aim:

Theory:

Code:

Name: Vikas Mourya

Class: SYCS

Old Roll No: 4057

New Roll No: 333

Subject: DS

Git Hub Link: <https://github.com/thisisvikasmourya/DS>

```
P8.py - D:\subject\data_structure\practical\P8.py (3.8.5)
File Edit Format Run Options Window Help
import random
random.seed(23)
class Node:
    def __init__(self, val):
        self.val = val
        self.leftChild = None
        self.rightChild = None
def insert(root, key):
    if root is None:
        return Node(key)
    else:
        if root.val == key:
            return root
        elif root.val < key:
            root.rightChild = insert(root.rightChild, key)
        else:
            root.leftChild = insert(root.leftChild, key)
    return root
def PrintInorder(root):
    if root:
        PrintInorder(root.leftChild)
        print(root.val, end=" ")
        PrintInorder(root.rightChild)
def printPreorder(root):
    if root:
        print(root.val, end=" ")
        printPreorder(root.leftChild)
        printPreorder(root.rightChild)
def printPostorder(root):
    if root:
        printPostorder(root.leftChild)
        printPostorder(root.rightChild)
        print(root.val, end=" ")
|
tree = Node(20)
for i in range(10):
    insert(tree, random.randint(2, 100))
```

Output:

```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\subject\data_structure\practical\P8.py =====
inorder
4 12 18 20 39 41 47 50 56 69 77

preorder
20 12 4 18 39 77 41 56 50 47 69

postorder
4 18 12 47 50 69 56 41 77 39 20
>>> |
```

Name: Vikas Mourya

Class: SYCS

Old Roll No: 4057

New Roll No: 333

Subject: DS

Git Hub Link: <https://github.com/thisisvikasmourya/DS>

Completed