PRACTICAL NO: 1

Q) Write a program for stack implementation in python.

Code:

```
# Stack implementation in python

# Creating a stack
def create_stack():
    stack = []
    return stack


# Creating an empty stack
def check_empty(stack):
    return len(stack) == 0


# Adding items into the stack
def push(stack, item):
    stack.append(item)
    print("pushed item: " + item)


# Removing an element from the stack
def pop(stack):
    if (check_empty(stack)):
        return "stack is empty"
```

```
        return stack.pop()


stack = create_stack()
push(stack, str(1))
push(stack, str(2))
push(stack, str(3))
push(stack, str(4))
print("popped item: " + pop(stack))
print("stack after popping an element: " + str(stack))
```

Output:

```
pushed item: 1
pushed item: 2
pushed item: 3
pushed item: 4
popped item: 4
stack after popping an element: ['1', '2', '3']
```

PRACTICAL NO:2

Q) Write a program for implementing queue in python.

Code:

```
# Queue implementation in Python


class Queue:


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


    # Add an element
    def enqueue(self, item):
        self.queue.append(item)


    # Remove an element
    def dequeue(self):
        if len(self.queue) < 1:
            return None
        return self.queue.pop(0)


    # Display  the queue
    def display(self):
```

```
        print(self.queue)

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



q = Queue()
q.enqueue(1)
q.enqueue(2)
q.enqueue(3)
q.enqueue(4)
q.enqueue(5)
print("Elements in queue:")
q.display()
q.dequeue()
print("After removing an element:")
q.display()
```

Output:

```
Elements in queue:
[1, 2, 3, 4, 5]
After removing an element:
[2, 3, 4, 5]
```

Practical No: 03

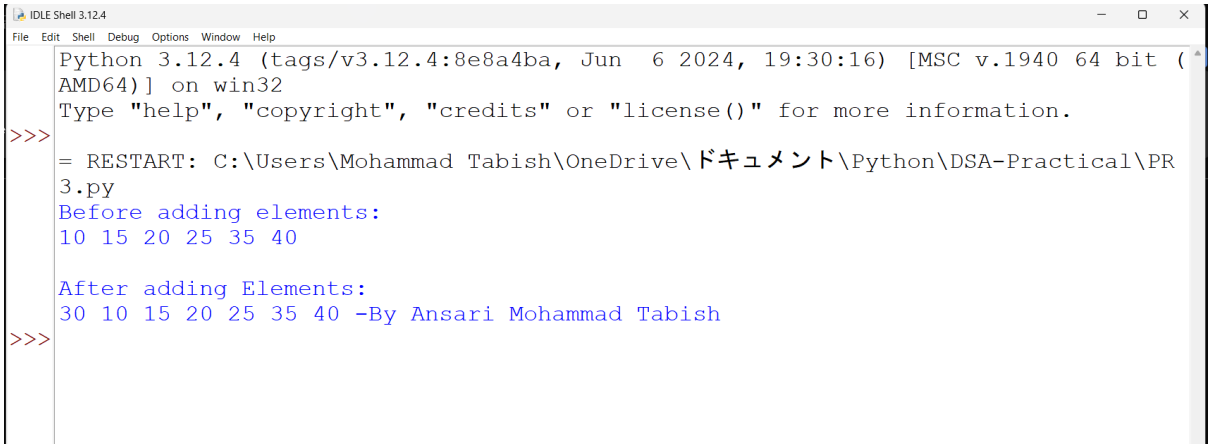
- Implementation of singly LinkedList as inserting elements in Python

- code:

```
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None
node1=Node(10)
node2=Node(15)
node3=Node(20)
node4=Node(25)
node5=Node(35)
node6=Node(40)

node1.next=node2
node2.next=node3
node3.next=node4
node4.next=node5
node5.next=node6

head=node1

current = head
print("Before adding elements: ")
while current is not None:
    print(current.data, end=' ')
    current=current.next

new_node = Node(30)
new_node.next=head
head=new_node
```

```
print("\n")
print("After adding Elements: ")
current1=head
while current1 is not None:
    print(current1.data, end=' ')
    current1=current1.next
print("-By Ansari Mohammad Tabish")
```

- OUTPUT:



Practical No:04

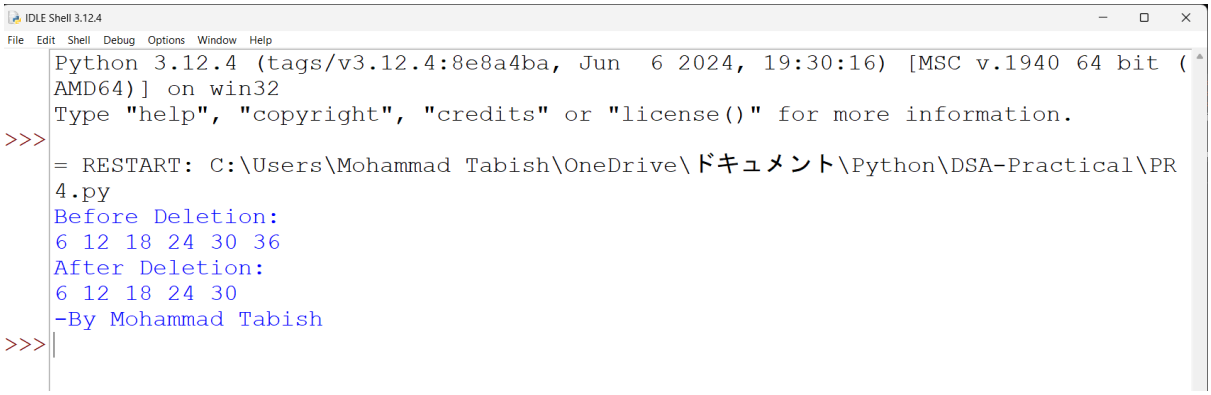
- Implementation of singly LinkedList for inserting elements in Python

- CODE:

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

head=Node(6)
node2=Node(12)
node3=Node(18)
node4=Node(24)
node5=Node(30)
node6=Node(36)

head.next=node2
node2.next=node3
node3.next=node4
node4.next=node5
node5.next=node6

current=head
current1=head
current2=head
print("Before Deletion: ")
while current1 is not None:
    print(current1.data, end=' ')
    current1=current1.next
```

```
while current.data != 30:
    current = current.next
current.next= current.next.next


print(" \nAfter Deletion: ")
while current2 is not None:
    print(current2.data, end=' ')
    current2=current2.next
print("\n-By Mohammad Tabish")
```

- OUTPUT :



Practical No:05

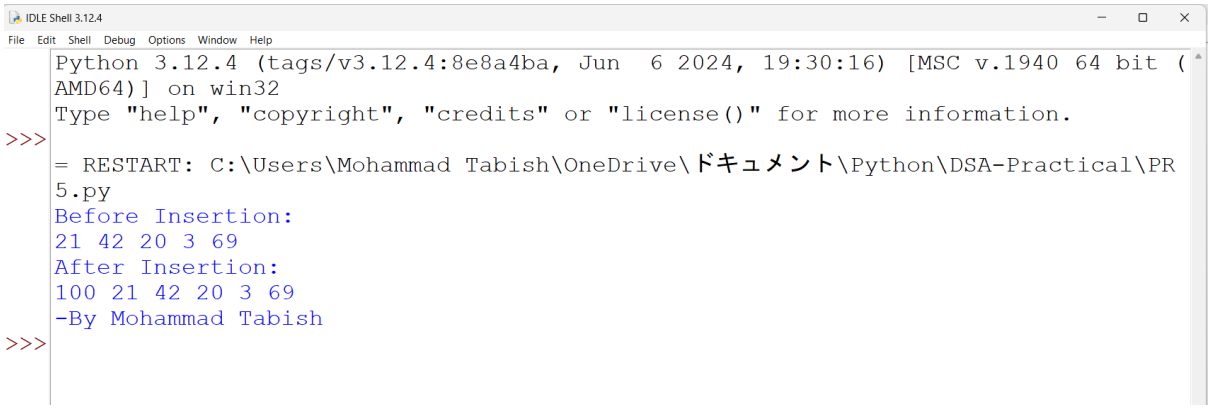
- Implementation of doubly Linkedlist for inserting elements in python

- CODE :

```
class Node:
    def __init__(self,data):
        self.data=data
        self.preveous=None
        self.next=None

node1=Node(21)
node2=Node(42)
node3=Node(20)
node4=Node(3)
node5=Node(69)

node1.next=node2
node2.next=node3
node3.next=node4
node4.next=node5

node5.preveous=node4
node4.preveous=node3
node3.preveous=node2
node2.preveous=node1

new_node = Node(100)
head = node1
current = head
print("Before Insertion: ")
while current is not None:
    print(current.data, end=' ')
    current=current.next
```

```
new_node.next = head

head.preveous = new_node

head = new_node


print("\nAfter Insertion: ")

current1 = head

while current1 is not None:

    print(current1.data, end=' ')

    current1=current1.next

print("\n-By Mohammad Tabish")
```

- OUTPUT :



Practical No:06

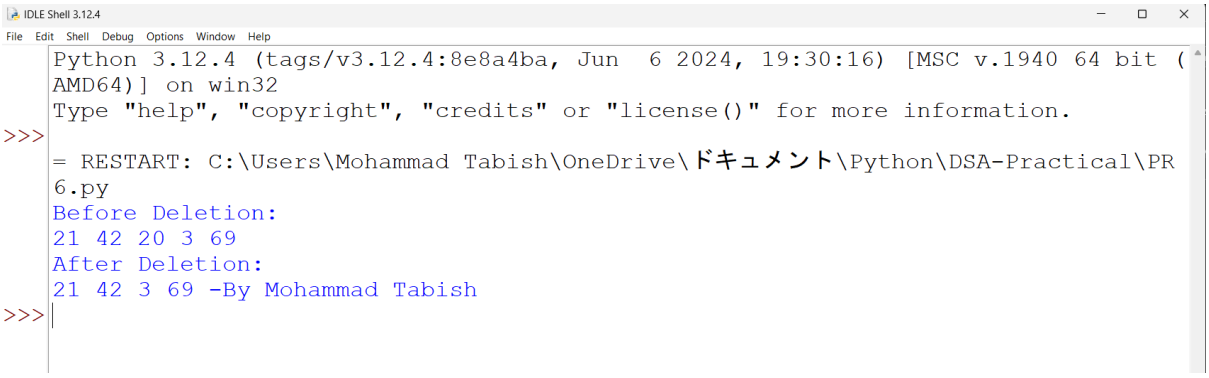
- Implementation of doubly Linkedlist for deleting elements in python

- CODE :

```
class Node:
    def __init__(self,data):
        self.data=data
        self.preveous=None
        self.next=None

node1=Node(21)
node2=Node(42)
node3=Node(20)
node4=Node(3)
node5=Node(69)

node1.next=node2
node2.next=node3
node3.next=node4
node4.next=node5

node5.preveous=node4
node4.preveous=node3
node3.preveous=node2
node2.preveous=node1

head=node1
current=head
current1=head
current2=head

print("Before Deletion: ")
while current1 is not None:
```

```
    print(current1.data, end=' ')
    current1=current1.next

while current.data != 42:
    current=current.next
current.next=current.next.next

print(" \nAfter Deletion: ")
while current2 is not None:
    print(current2.data, end=' ')
    current2=current2.next
print("-By Mohammad Tabish")
```

- OUTPUT :



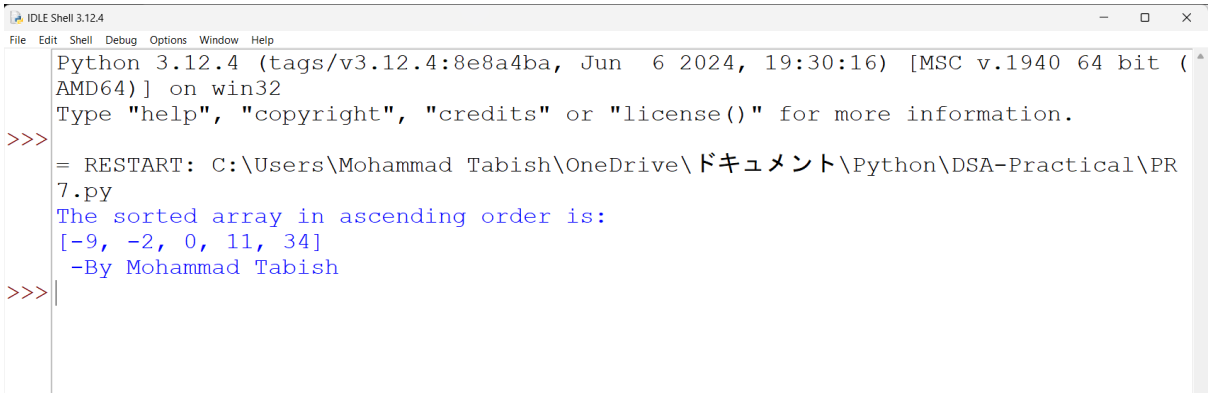
Practical No:07

- Implementation of selection sort algorithm in python

- CODE :

```
def selectionsort(array, size):
    for step in range(size):
        min_idx = step
        for i in range(step + 1, size):
            if array[i] < array[min_idx]:
                min_idx = i
        array[step], array[min_idx] = array[min_idx], array[step]


data = [-2, 34, 0, 11, -9]
size = len(data)
selectionsort(data, size)
print('The sorted array in ascending order is:')
print(data,"\n -By Mohammad Tabish")
```

- OUTPUT:



Practical No:08

- Implementation of Bubble sort algorithm in python

- CODE :

```
def bubblesort(array):
    for i in range(len(array)):
        for j in range(0, len(array) - i - 1):
            if array[j] > array[j + 1]:
                temp = array[j]
                array[j] = array[j + 1]
                array[j + 1] = temp


data = [-1, 4, -3, 0, 8]
bubblesort(data)
print('The sorted array is:')
print(data)
```

-OUTPUT: