19CS301-Module12

Ex.No 12.a Stack Using Linked List in Python

This project demonstrates a basic stack implementation in Python using list methods to Stack Operations: Insertion, Deletion, and Displaying Remaining Elements

AIM

To perform basic stack operations by inserting three elements, deleting one element, and displaying the number of elements remaining in the stack.

ALGORITHM

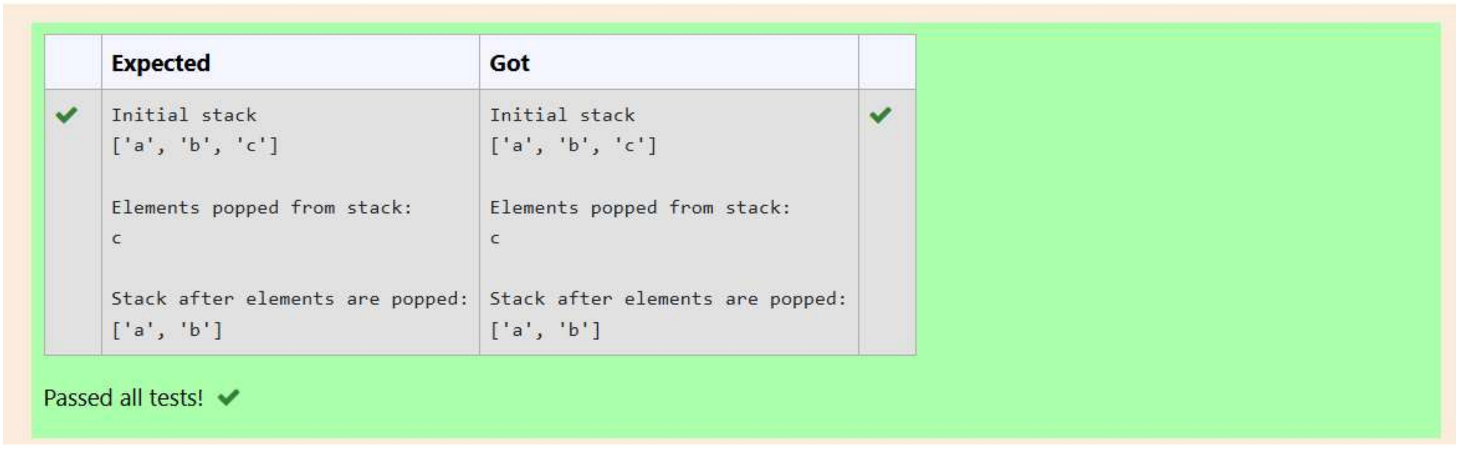
1.Start 2.Initialize an empty stack. 3.Push three elements onto the stack. Push Element 1 Push Element 2 Push Element 3 4.Pop (delete) the top element from the stack. 5.Count the number of elements remaining in the stack. 6.Display the number of remaining elements. 7.End

PROGRAM

Reg no: 212223020021 Name: Ranjith P

```
stack = []

stack.append('a')
stack.append('b')
stack.append('c')

print('Initial stack')
print(stack)
print("\nElements popped from stack:")
print(stack.pop())



print('\nStack after elements are popped:')
print(stack)
```

OUTPUT



	Expected	Got	

✓	`Initial stack`
	`['a', 'b', 'c']`
	
	`Elements popped from stack:`
	`c`
	
	`Stack after elements are popped:`
	`['a', 'b']`
		`Initial stack`
		`['a', 'b', 'c']`
		
		`Elements popped from stack:`
		`c`
		
		`Stack after elements are popped:`
		`['a', 'b']`
			✓

Passed all tests! ✓

RESULT

Thus, the given program is implemented and executed successfully.

Ex.No 12.b Stack Using Linked List – Push and Index Display

Aim

To write a Python program that takes 3 inputs from the user, inserts them into a stack, and displays each element along with its index.

Algorithm

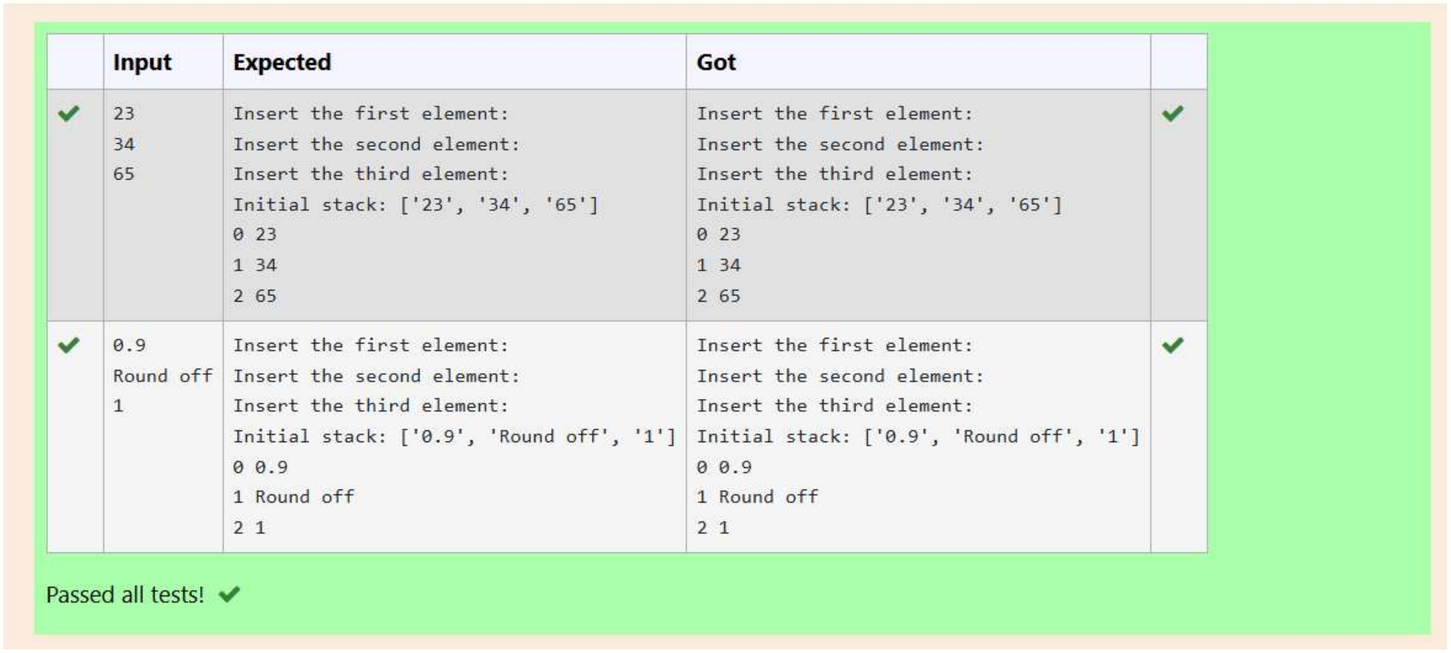
1.Start 2.Initialize an empty stack. 3.Repeat 3 times: Accept an input from the user. Insert the input into the stack using append(). 4.For each element in the stack: 5.Display its index and value. 6.End

Program

Reg no: 212223020021 Name: Ranjith P

```
stack = []

stack.append(input("Insert the first element:"))
stack.append(input("\nInsert the second element:"))
stack.append(input("\nInsert the third element:"))

print('\nInitial stack: ' + str(stack))

for i in range(len(stack)):
    print(i, end=" ")
    print(stack[i])
```

Output



	Input	Expected	Got	
✓	23 34 65	Insert the first element: Insert the second element: Insert the third element: Initial stack: ['23', '34', '65'] 0 23 1 34 2 65	Insert the first element: Insert the second element: Insert the third element: Initial stack: ['23', '34', '65'] 0 23 1 34 2 65	✓
✓	0.9 Round off 1	Insert the first element: Insert the second element: Insert the third element: Initial stack: ['0.9', 'Round off', '1'] 0 0.9 1 Round off 2 1	Insert the first element: Insert the second element: Insert the third element: Initial stack: ['0.9', 'Round off', '1'] 0 0.9 1 Round off 2 1	✓

Passed all tests! ✓

Result

Thus, the given program is implemented and executed successfully .

12 c Queue Using Linked List – Display, Peek, and Pop

Aim

To write a Python program to insert elements into a queue and check whether the queue is full or not.

Algorithm

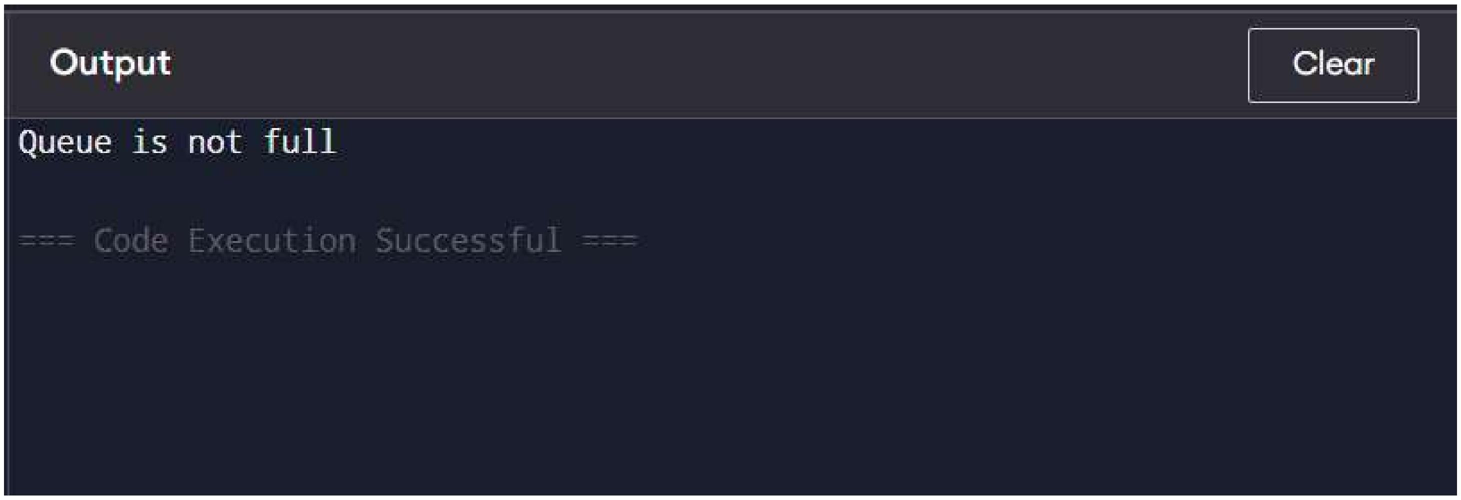
1. Start 2.Define a maximum size for the queue . 3.Initialize an empty queue. 4.Insert elements into the queue using a loop. 5.After each insertion, or at the end, check: If the length of the queue is equal to the maximum size → Queue is Full. Else → Queue is Not Full. 6.Display the queue status. 7.End

Program

Reg no: 212223020021 Name: Ranjith P

```
from queue import Queue

queue = Queue(maxsize = 4)

queue.put('a')
queue.put('b')
queue.put('c')

if queue.full():
    print("Queue is full")
else:
    print("Queue is not full")
```

OUTPUT



```
Queue is not full


=== Code Execution Successful ===
```

RESULT

Result: Thus, the given program is implemented and executed successfully .

19CS301:## 12 d Queue Using Linked List – Enqueue and All Operations

Aim

To write a Python program to add 4 elements to a queue and print the elements present at the front and rear of the queue.

Algorithm

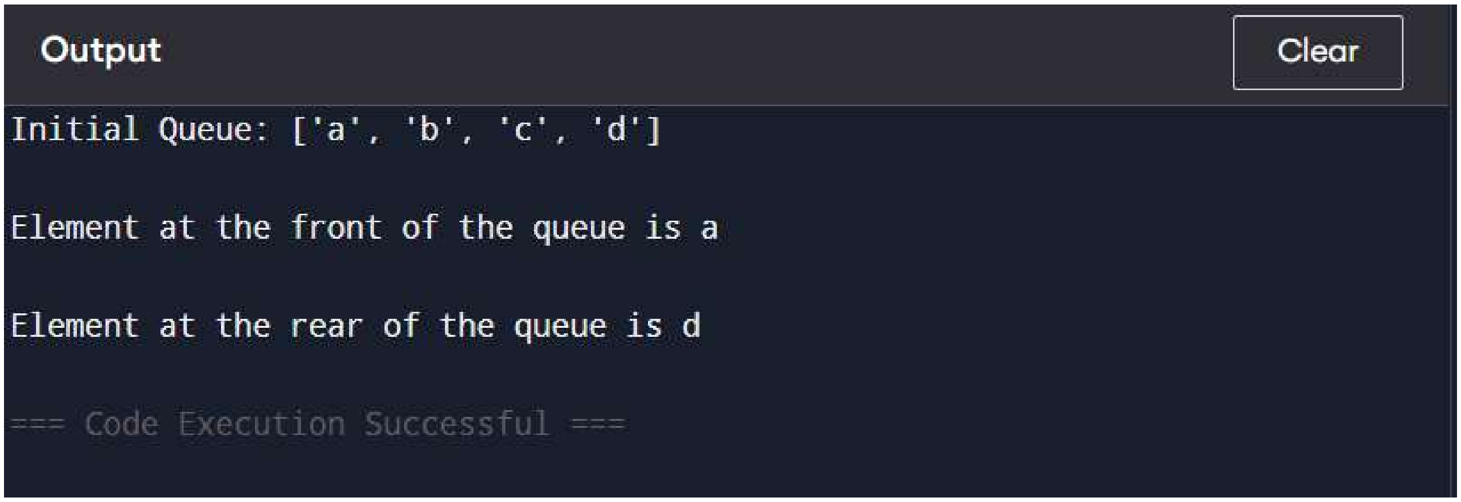
1. Start
2. Create a queue
3. Append elements to the queue
4. Print the front and rear elements in the queue
5. Stop

Program

Reg no:212223020021 Name: Ranjith P

```
queue = []

# Enqueue elements into the queue
queue.append('a')
queue.append('b')
queue.append('c')
queue.append('d')

# Display the initial queue
print('Initial Queue:', queue)

# Print front and rear elements
print("\nElement at the front of the queue is", queue[0])
print("\nElement at the rear of the queue is", queue[-1])
```

OUTPUT



```
Initial Queue: ['a', 'b', 'c', 'd']

Element at the front of the queue is a

Element at the rear of the queue is d

=== Code Execution Successful ===
```

RESULT

Result: Thus, the given program is implemented and executed successfully .

SEB - E) Stack Using Linked List in Python

This project demonstrates a basic stack implementation in Python using list methods to Stack Operations: Insertion, Deletion, and Displaying Remaining Elements

AIM

To perform basic stack operations by inserting three elements, deleting one element, and displaying the number of elements remaining in the stack.

ALGORITHM

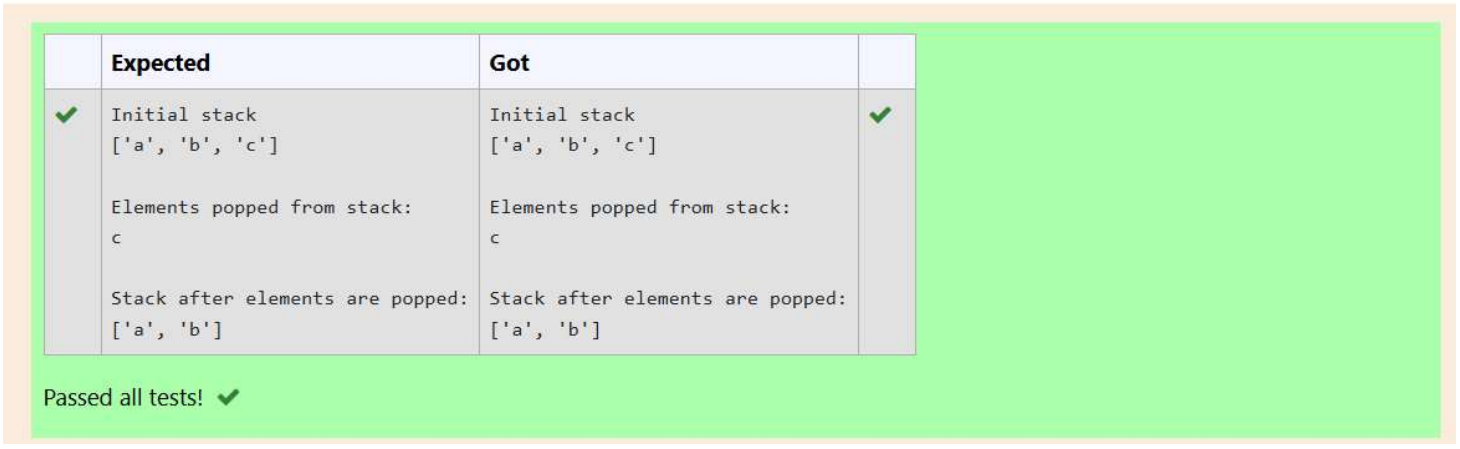
1.Start 2.Initialize an empty stack. 3.Push three elements onto the stack. Push Element 1 Push Element 2 Push Element 3 4.Pop (delete) the top element from the stack. 5.Count the number of elements remaining in the stack. 6.Display the number of remaining elements. 7.End

PROGRAM

Reg no: 212223020021 Name: Ranjith P

```
stack = []

stack.append('a')
stack.append('b')
stack.append('c')

print('Initial stack')
print(stack)
print("\nElements popped from stack:")
print(stack.pop())



print('\nStack after elements are popped:')
print(stack)
```

OUTPUT



	Expected	Got	
✓	Initial stack ['a', 'b', 'c'] Elements popped from stack: c Stack after elements are popped: ['a', 'b']	Initial stack ['a', 'b', 'c'] Elements popped from stack: c Stack after elements are popped: ['a', 'b']	✓

Passed all tests! ✓

RESULT

Thus, the given program is implemented and executed successfully.