**DSA - Mock Test 3 - Python**

**Que.** Implement a stack using a list in Python. Include the necessary methods such as push, pop, and isEmpty.

**------------**

**Solution :**

**------------**

class Node:

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

self.data = data

self.next = None

class Stack:

# head is default NULL

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

self.head = None

# Checks if stack is empty

def isempty(self):

if self.head == None:

return True

else:

return False

# Method to add data to the stack

def push(self, data):

if self.head == None:

self.head = Node(data)

else:

newnode = Node(data)

newnode.next = self.head

self.head = newnode

# Remove element that is the current head (start of the stack)

def pop(self):

if self.isempty():

return None

else:

poppednode = self.head

self.head = self.head.next

poppednode.next = None

return poppednode.data

# Prints out the stack

def display(self):

iternode = self.head

if self.isempty():

print("Stack Underflow")

else:

while(iternode != None):

print(iternode.data, end = "")

iternode = iternode.next

if(iternode != None):

print(" -> ", end = "")

return

if \_\_name\_\_ == "\_\_main\_\_":

MyStack = Stack()

MyStack.push(21)

MyStack.push(22)

MyStack.push(23)

MyStack.push(24)

MyStack.display()

MyStack.pop()

MyStack.pop()

MyStack.display()

Output : 21 -> 22 -> 23 -> 24

------

**-------------------------**

**Complexity Analysis :**

**-------------------------**

* **Time Complexity:** O(1), for all push(), as we are not performing any kind of traversal over the list.
* **Auxiliary Space:** O(N), where N is the size of the stack.

====================================x=================================