1. Write a Python program to create a class representing a stack data structure. Include methods for pushing and popping elements.
2. Write a Python program to create a class representing a stack data structure. Include methods for pushing, popping and displaying elements.

**Note:** Both questions answer is added in a single logic

**Answer)**

**class StackStruc:**

**def \_\_init\_\_(self):** *#Function to initialize the stack for with empty list.*

**self.items = []**

**def pushval(self, item):** *#Function to push an item*

**self.items.append(item)**

**def popval(self,val):**

**if not self.is\_empty():** *#Function to pop an item from list based on if else condition*

**return self.items.pop()**

**else:**

**raise IndexError("Cannot pop an item from the empty stack.")**

**def is\_empty(self):**

**return len(self.items) == 0** *#Function to check the stack is empty*

**def display(self):**

**print("Stack items:", self.items)**

**StackStruc = StackStruc()** *# Create an instance of the Stack class*

**StackStruc.pushval(300)**

**StackStruc.pushval(400)**

**StackStruc.pushval(500)** *# Push items into the stack*

**StackStruc.pushval(600)**

**StackStruc.pushval(700)**

**StackStruc.display()**

**popped\_item = StackStruc.popval(-1)**

**print("Popped item is:", popped\_item)** *#Pop item from the stack(default value is -1, which returns the last item)*

**popped\_item = StackStruc.popval(-2)** *#A number specifying the position of the element you want to remove.*

**print("Popped item is:", popped\_item)**

**StackStruc.display()**

Output will be:  
Stack items: [300, 400, 500, 600, 700]

Popped item is: 700

Popped item is: 600

Stack items: [300, 400, 500]

3.Write a Python program to create a class representing a linked list data structure. Include methods for displaying linked list data, inserting and deleting nodes.

**class Node:**

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

**self.data = data** *# Initialize the node with data and set the next pointer to None*

**self.next = None**

**class LinkedList:**

**def \_\_init\_\_(self):** *# Initialize the linked list with an empty head node*

**self.head = None**

**def display(self):**

**current = self.head** *# Function to display the elements***.**

**while current:**

**print(current.data, end=" ")**

**current = current.next**

**print()**

**def insert(self, data):**

**new\_node = Node(data)**

**if not self.head:** *#Function to insert a new node with the given data at the end of the linked list*

**self.head = new\_node**

**else:**

**current = self.head**

**while current.next:**

**current = current.next**

**current.next = new\_node**

**def delete(self, data):**

**if not self.head:** *# Function to delete a node with the given data from the linked list*

**return**

**if self.head.data == data:**

**self.head = self.head.next**

**return**

**current = self.head**

**prev = None**

**while current and current.data != data:**

**prev = current**

**current = current.next**

**if current:**

**prev.next = current.next**

**linked\_list = LinkedList()**

**linked\_list.insert(10)**

**linked\_list.insert(20)**

**linked\_list.insert(30)**

**linked\_list.insert(40)**

**print("The Linked List is:")**

**linked\_list.display()** *# Called the display function*

**linked\_list.insert(100)**

**print("Linked list after inserting a new node:") #Call the insert the function add new new node**

**linked\_list.display()**

**linked\_list.delete(40)**

**print("Linked list after deleting an existing node:") #Call the function to delete the node**

**linked\_list.display()**

**Output will be:**

The Linked List is:

10 20 30 40

Linked list after inserting a new node:

10 20 30 40 100

Linked list after deleting an existing node:

1. 0 30 100

3. Write a Python program to create a class representing a shopping cart. Include methods for adding and removing items, and calculating the total price.

**Answer)**

**class Cart:**

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

**self.items = []**

**def add\_item(self, item\_name, Amount):**

**item = (item\_name, Amount)**

**self.items.append(item)**

**def remove\_item(self, item\_name):**

**for item in self.items:**

**if item[0] == item\_name:** *#Function to remove item from cart*

**self.items.remove(item)**

**break**

**def calculate\_total(self):**

**total = 0**

**for item in self.items:** *#Function calculate the total amount in cart*

**total += item[1]**

**return total**

**Purchase = Cart()**

**Purchase.add\_item("Apple", 120)** *# Add items to the shopping cart*

**Purchase.add\_item("Orange", 130)**

**Purchase.add\_item("Banana", 150)**

**Purchase.add\_item("Cucumber", 160)**

**Purchase.add\_item("Carrot", 180)**

**print("Current Items in Cart:")** *# Display the current items in the cart*

**for item in Purchase.items:**

**print(item[0], "-", item[1])**

**Total = Purchase.calculate\_total()**

**print("Total Quantity:", Total)**

**Purchase.remove\_item("Orange")**

**print("\nUpdated Items in Cart after removing Orange:")** *# Remove an item from the cart.*

**for item in Purchase.items:**

**print(item[0], "-", item[1])**

**Total = Purchase.calculate\_total()**

**print("Total Quantity:", Total)** *#Recalculate the total quantity*

**Output will be:**  
Current Items in Cart:

Apple - 120

Orange - 130

Banana - 150

Cucumber - 160

Carrot - 180

Total Quantity: 740

Updated Items in Cart after removing Orange:

Apple - 120

Banana - 150

Cucumber - 160

Carrot - 180

Total Quantity: 610