Assignment -1

Task Description:

Create a class named ListManipulator with the following methods:

\_\_init\_\_: Initializes an empty list.

Add\_elements: Takes a list of elements as a parameter and appends them to the internal list.

Remove\_duplicates: Removes duplicate elements from the internal list.

Reverse\_list: Reverses the order of elements in the internal list.

Sort\_list: Sorts the elements in the internal list in ascending order.

Get\_unique\_elements: Returns a new list containing only the unique elements from the internal list.

Remove\_element: Takes an element as a parameter and removes its first occurrence from the internal list.

Get\_list: Returns the current state of the internal list.

->

Here’s the implementation of the `ListManipulator` class in Python:

```python

Class ListManipulator:

Def \_\_init\_\_(self):

Self.internal\_list = []

Def add\_elements(self, elements):

Self.internal\_list.extend(elements)

Def remove\_duplicates(self):

Self.internal\_list = list(set(self.internal\_list))

Def reverse\_list(self):

Self.internal\_list.reverse()

Def sort\_list(self):

Self.internal\_list.sort()

Def get\_unique\_elements(self):

Return list(set(self.internal\_list))

Def remove\_element(self, element):

If element in self.internal\_list:

Self.internal\_list.remove(element)

Def get\_list(self):

Return self.internal\_list

```

You can use this class to manipulate lists according to the provided methods. For example:

```python

Lm = ListManipulator()

Lm.add\_elements([1, 2, 3, 4, 3, 2])

Lm.remove\_duplicates()

Print(lm.get\_list()) # Output: [1, 2, 3, 4]

Lm.reverse\_list()

Print(lm.get\_list()) # Output: [4, 3, 2, 1]

Lm.sort\_list()

Print(lm.get\_list()) # Output: [1, 2, 3, 4]

Print(lm.get\_unique\_elements()) # Output: [1, 2, 3, 4]

Lm.remove\_element(2)

Print(lm.get\_list()) # Output: [1, 3, 4]

```