Assignment (List, Tuple, Dictionary)

List Programs:

1. Find the Sum of Elements in a List

Solution:

```
    my_list = [1, 2, 3, 4, 5]

  sum_of_elements = sum(my_list)

  print("Sum of elements:", sum_of_elements)
```

2. Remove Duplicates from a List

Solution:

```
  my_list = [1, 2, 2, 3, 4, 4, 5]

  unique_list = list(set(my_list))

  print("List with duplicates removed:", unique_list)
```

3. Sort a List in Ascending Order

Solution:

```
  my_list = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]

  my_list.sort()

  print("Sorted list in ascending order:", my_list)
```

Using `eval()` for Input:

4. Find the Average of Numbers in a List using `eval()`

Solution:

```
   input_str = input("Enter a list of numbers separated by spaces: ")

   num_list = eval("[" + input_str + "]")

   average = sum(num_list) / len(num_list)
   print("Average:", average)
```

Dictionary Programs:

5. Create a Dictionary and Access Values

Solution:

```
  my_dict = {'name': 'John', 'age': 30, 'city': 'New York'}

  print("Name:", my_dict['name'])

  print("Age:", my_dict['age'])

  print("City:", my_dict['city'])
```

6. Add a Key-Value Pair to a Dictionary

Solution:

```
   my_dict = {'name': 'John', 'age': 30}

   my_dict['city'] = 'New York'

   print("Updated Dictionary:", my_dict)
```

7. Iterate Over a Dictionary

Solution:

```
   my_dict = {'name': 'John', 'age': 30, 'city': 'New York'}

   for key, value in my_dict.items():

       print(key, ":", value)
```

8. Create a Dictionary with User-Provided Key-Value Pairs using `eval()`

Solution:

```
   input_str = input("Enter a dictionary in the format {'key1': 'value1', 'key2': 'value2'}: ")

   my_dict = eval(input_str)
```

```
print("Dictionary:", my_dict)
```

Tuple Programs:

9. Create and Access Elements in a Tuple

Solution:

```
my_tuple = (1, 2, 3, 4, 5)

print("Element at index 2:", my_tuple[2])
```

10. Concatenate Tuples

Solution:

```
tuple1 = (1, 2, 3)

tuple2 = (4, 5, 6)

concatenated_tuple = tuple1 + tuple2

print("Concatenated Tuple:", concatenated_tuple)
```

11. Check if an Element Exists in a Tuple

Solution:

```
my_tuple = (1, 2, 3, 4, 5)

element_to_check = 3

if element_to_check in my_tuple:

    print(element_to_check, "exists in the tuple.")

else:

    print(element_to_check, "does not exist in the tuple.")
```

12. Perform Tuple Operations using `eval()`

Solution:

```
input_str1 = input("Enter a tuple (e.g., (1, 2, 3)): ")
```

```
input_str2 = input("Enter another tuple (e.g., (4, 5, 6)): ")

my_tuple1 = eval(input_str1)

my_tuple2 = eval(input_str2)

concatenated_tuple = my_tuple1 + my_tuple2

length = len(concatenated_tuple)

print("Concatenated Tuple:", concatenated_tuple)

print("Length of Concatenated Tuple:", length)
```