Task 4: Use various data types, list, Tuples and Dictionary in Python Programming

Aim:

To use various data types, list, Tuples and Dictionary in the Python Programming.

ⓐ You are working on a python Project that requires you to manage and manipulate a list of numbers. Your task is to create a python program that demonstrates the following list operations:

Algorithm:

1. Start
2. For adding elements to a list first create a list with name "List" and assign the values within[] brackets, inorder to add a new value use the function append().
3. For removing a specific element use "pop(index value)" or "remove (itemname)".
4. For sorting the elements use "sorted(list)" function.
5. For finding minimum value use "min(list)" and for maximum use "max(list)".
6. For Sum use function "sum(list)" and for average use the formula "sum(list)/len(list)".
7. Print the output.
8. End.

Program

```
#Add Elements: Add elements to the list.
list = [10,20]
a=30
```

Output:

```
[10, 20, 30]
[10, 30]
[30]
[5, 8, 9, 15, 30, 89]
The minimum value is: 5
The maximum value is: 89
The sum is: 156
The average is: 26.0
```

```
list.append(a)
Print(list)
#Remove Elements: Remove specific elements from the list.
list.pop(1) #by index value.
Print(list)
list.remove(10) #by itemname
Print(list)
#Sort Elements: Sort the list in ascending and descending order.
l=[5,8,9,15, 30,89]
Print(sorted(l))
#Find Minimum and maximum: Find the minimum and maximum
elements in the list.
Print("The minimum value is:", min(l))
Print("The maximum value is:", max(l))
#calculate Sum and Average
Print("The sum is:", sum(l))
Print("The average is:", ((sum(l)/len(l))))
```

(6) You are tasked with creating a Python Program that show-cases operations on tuples. Tuples are immutable sequences, similar to lists but with the key difference that they cannot be changed after creation. Your Program should illustrate the following tuple operations.

Algorithm:

1. Start
2. To create a tuple use "tuple_name=(values)".
3. To access the elements of a tuple either use the index

Output:

```
(10, 'hello', 3.14, 'world')
10
hello
3.14
world
('hello', 3.14)
(10, 'hello', 3.14)
```

values(tuple_name (~~values~~ index_value)) or the tuple slicing (tuple_name[start:end])

4. To concatenate tuples use the operator "+" (tuple1 "+" tuple2)
5. Try to modify the tuple elements by assigning the values directly like; tuple(index)=new_value, will result in an error as it immutable.
6. Print the output.
7. End.

Program:

```
#create a tuple: Define a tuple with elements of different data
types (10, 'hello', 3.14, 'world').
tuple = (10, 'hello', 3.14, 'world')
Print(tuple)
# Access Elements: Access individual elements and slices of the
tuple
for i in tuple:
~~Print(i)~~ Print(i)
Print(tuple[1:3])
Print(tuple[:-1])
#Concatenate Tuples: Combine two tuples to create a new tuple.
t2 = (5, 0.5)
t3 = tuple + t2
Print(t3)
# Immutable Nature: Attempt to modify elements of the tuple
and handle the resulting error
tuple(1) = "Pi" #ERROR
```

ⓒ You are tasked with creating a Python program that showcases operations on dictionaries. Dictionaries in Python are unordered collections of items. Each items is a pair consisting of a key and a value.

Algorithm:

1. Start the program.
2. Define a dictionary with key value pairs of different data types.
3. Retrieve values from the dictionary using their corresponding keys.
4. Modify Dictionary.
5. Iterate over Dictionary.
6. Stop the program.

Program:

```
# Create a Dictionary: Define a dictionary with key-value
Pairs of different data types. ({`name': `Alice', `age': 30, `city':
`New york'})
dictionary = {`name': `Alice', `age': 30, `city': `New York'}
Print(dictionary)
# Access values: Access values using keys.
Print(dictionary[`name'])
Print(dictionary[`age'])
# Modify Dictionary: update values, add new key-value Pairs, and
remove existing Pairs.
dictionary[`name'] = "James".
Print(dictionary)
```

Output:

```
{'name': 'Alice', 'age': 30, 'city': 'New York'}
Alice
30
{'name': 'James', 'age': 30, 'city': 'New York'}
{'name': 'James, 'age': 30}
KEY: name
KEY: age
dict_items([('name', 'James'), ('age', 30)])
```

```
dictionary.Pop('city')
Print(dictionary)
#Iterate over Dictionary: Use loops to iterate over keys or
the values.
for k in dictionary:
~~Print("KEY:", k)~~ Print("KEY:", k)
Print(dictionary.items())
```

VEL TECH	
EX No.	4
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
VITH DATE	[illegible] 20/8/25

Result:

Thus, various data types, list, Tuples and Dictionary in Python Programming was used and verified successfully.