

Task-4 Use Various data types, List, Tuples and Dictionary in Python programming

Aim:- To use various data types, List, Tuples and Dictionary in 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:
- Add elements:- Add element to the list.
- Remove elements:- Remove specific elements from the list.
- Sort elements:- Sort the list in ascending and descending order.
- Find minimum and maximum:- Find the min and max elements in the list.
- Calculate sum and average:- Calculate the sum and average of the elements in the list.

Algorithm:-

- Start
- For adding elements to a list first create a list with name 'lst' and assign the values.
- For removing a specific element use 'pop(index value)' (or) 'remove(elementname)'.
- For Sorting the elements use 'sorted(list)' function.
- For finding minimum value use 'min(list)' and for maximum use 'max(list)'.
- For sum use function 'sum(list)' and for average use the formula 'sum(list) / len(list)'.
- Print the Output.
- Output.

*Program:-

```

list = [10, 20]
a = 30
list.append(a)
print(list)

list.pop(1)
print(list)

l = [5, 8, 9, 15, 30, 89]
print(sorted(l))
print("The minimum value is:", min(l))
print("The maximum value is:", max(l))
print("The sum is:", sum(l))
print("The average is:", (sum(l)/len(l)))
    
```

Output:-

```

[10, 20, 30]
[10, 20]
[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

Sl. No.	NAME	DATE
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

* Sequence

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

```
print(tuple)
```

~~the tuple~~

```
for i in tuple:
```

```
    print(i)
```

```
print(tuple[0:3])
```

```
print(tuple[1:4])
```

```
t2 = (5, 0.5)
```

```
t3 = tuple + t2
```

```
print(t3)
```

```
tuple[3] = "PI" # error.
```

Output

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

```
10
```

```
hello
```

```
3.14
```

```
world
```

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

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

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

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

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

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

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

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

- 1) Create a Tuple :- Define a tuple with elements of different data types (10, 'hello', 3.14, 'world')
- 2) Access elements :- Access individual elements and slices of the tuple.
- 3) Concatenate tuples :- Combine two tuples to create a new tuple.
- 4) Immutable Nature :- Attempt to modify elements of the tuple and handle the resulting error.

Algorithm:-

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

(tuple_name[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 error as it is immutable.
6. Print the output
7. End.

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


* Program:

```
dictionary = {'name': 'Alice', 'age': 30, 'city': 'New York'}
```

```
print(dictionary)
```

```
print(dictionary['age'])
```

```
print(dictionary['name'])
```

```
dictionary['name'] = "James"
```

```
print(dictionary)
```

```
dictionary.pop('city')
```

```
print(dictionary)
```

```
for k in dictionary:
```

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

```
: print(dictionary.items())
```

Output:

```
{'name': 'James', '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)])
```

C) You are tasked, with creating a Python program that show cases operations on dictionaries. Dictionaries in Python are unordered collections of items.

Each item is a pair consisting of a key and a value. Your program should illustrate the following dictionary operations:

1. Create a dictionary: Define a dictionary with key-value pairs of different data types. (name: 'Alice', age: 30, city: 'New York').

2. Access values: Access values using keys.

3. Modify dictionary: Update values, add new key-value pairs, and remove existing pairs.

4. Iterate over dictionary: Use loops to iterate over keys (or) values.

Algorithm:-

1. Start

2. Define a dictionary with key-value pairs of different data types.

3. Retrieve values from the dictionary using their corresponding keys.

4. Iterate over dictionary.

5. Stop the program.

Result: Thus, various data types, list, tuples, and dictionary in Python programming are used and verified successfully.

Signature: _____

Date: _____

Page: _____

Section: _____

Roll No: _____

Branch: _____

Subject: _____

Teacher: _____

Head of Institution: _____

Principal: _____

Director: _____

Manager: _____

Coordinator: _____

Officer: _____

EX No.	PERFORMANCE	REMARKS
1	85	Good
2	78	Fair
3	92	Excellent
4	88	Good
5	75	Fair
6	82	Good
7	79	Fair
8	86	Good
9	81	Good
10	84	Good
11	87	Good
12	83	Good
13	80	Good
14	85	Good
15	82	Good
16	86	Good
17	81	Good
18	84	Good
19	87	Good
20	83	Good
21	80	Good
22	85	Good
23	82	Good
24	86	Good
25	81	Good
26	84	Good
27	87	Good
28	83	Good
29	80	Good
30	85	Good
31	82	Good
32	86	Good
33	81	Good
34	84	Good
35	87	Good
36	83	Good
37	80	Good
38	85	Good
39	82	Good
40	86	Good
41	81	Good
42	84	Good
43	87	Good
44	83	Good
45	80	Good
46	85	Good
47	82	Good
48	86	Good
49	81	Good
50	84	Good
51	87	Good
52	83	Good
53	80	Good
54	85	Good
55	82	Good
56	86	Good
57	81	Good
58	84	Good
59	87	Good
60	83	Good
61	80	Good
62	85	Good
63	82	Good
64	86	Good
65	81	Good
66	84	Good
67	87	Good
68	83	Good
69	80	Good
70	85	Good
71	82	Good
72	86	Good
73	81	Good
74	84	Good
75	87	Good
76	83	Good
77	80	Good
78	85	Good
79	82	Good
80	86	Good
81	81	Good
82	84	Good
83	87	Good
84	83	Good
85	80	Good
86	85	Good
87	82	Good
88	86	Good
89	81	Good
90	84	Good
91	87	Good
92	83	Good
93	80	Good
94	85	Good
95	82	Good
96	86	Good
97	81	Good
98	84	Good
99	87	Good
100	83	Good