**Task 4. Use various data types, List, Tuples and Dictionary in python programming**

1. **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:**
2. **Add Elements**: Add elements to the list.
3. **Remove Elements**: Remove specific elements from the list.
4. **Sort Elements**: Sort the list in ascending and descending order.
5. **Find Minimum and Maximum**: Find the minimum and maximum elements in the list.
6. **Calculate Sum and Average**: Calculate the sum and average of the elements in the list.
7. **insert:** Insert integer at position.
8. **print:** Print the list.
9. **reverse:** Reverse the list.
10. **Nested List:** create nested list, access the element in the nested list and modify the same.

**Sample Input and 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

[30, [1, 2]]

[[1, 2], 30]

2

**PROGRAM**

#Add Elements: Add elements to the list.

list=[10,20]

a=30

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 thelist.

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))))

list.insert(2,[1,2])

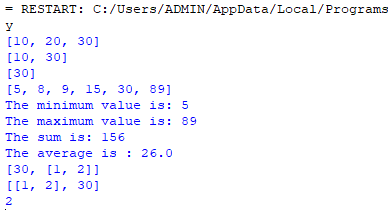
print(list)

list.reverse()

print(list)

print(list[0][1])

OUTPUT



1. **Given a list of integers, write a Python program using list comprehension to:**

Create a list containing the squares of all even numbers from the original list.

Create another list containing formatted string descriptions for each even number in the format:

"Square of x is y",

where x is the even number and y is its square.

**Sample Test Case**

**Input**

numbers = [1, 2, 3, 4, 5, 6]

**Output**

Even Squares: [4, 16, 36]

Descriptions: ['Square of 2 is 4', 'Square of 4 is 16', 'Square of 6 is 36']

**Program:**

# Input list of integers

numbers = [1, 2, 3, 4, 5, 6]

# List comprehension to get squares of even numbers

even\_squares = [x\*\*2 for x in numbers if x % 2 == 0]

# List comprehension with formatted descriptions

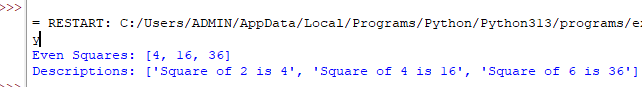
descriptions = [f"Square of {x} is {x\*\*2}" for x in numbers if x % 2 == 0]

# Display the results

print("Even Squares:", even\_squares)

print("Descriptions:", descriptions)

**Output:**



1. **You are tasked with creating a Python program that showcases 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:**
2. **Create a Tuple**: Define a tuple with elements of different data types(10, 'hello', 3.14, 'world')
3. **Access Elements**: Access individual elements and slices of the tuple.
4. **Concatenate Tuples**: Combine two tuples to create a new tuple.
5. **Immutable Nature**: Attempt to modify elements of the tuple and handle the resulting error.
6. **Tuple Unpacking:** Unpack elements of a tuple into individual variables.
7. **Membership Testing:** Check whether a value exists in the tuple using the in keyword.
8. **Tuple Length and Built-in Functions:** Use functions like len(), max(), min() and sum() (where applicable).

**Sample Test Case**

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

10

hello

3.14

world

('hello', 3.14)

(10, 'hello', 3.14)

(10, 'hello', 3.14, 'world', 5, 0.5)

Unpacked Values: a=10, b=hello, c=3.14, d=world

Is 'hello' in tuple? True

Length of Tuple: 4

Max: 15

Min: 5

Sum: 30

**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(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)

# Tuple Unpacking

a, b, c, d = tuple

print(f"Unpacked Values: a={a}, b={b}, c={c}, d={d}")

#Membership Testing

print("Is 'hello' in tuple?", 'hello' in tuple)

#Tuple Length

print("Length of Tuple:", len(tuple))

# Using functions where applicable

numeric\_tuple = (5, 10, 15)

print("Max:", max(numeric\_tuple))

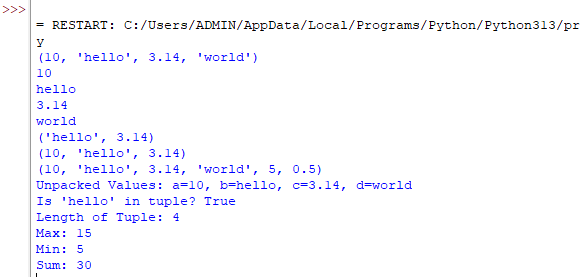
print("Min:", min(numeric\_tuple))

print("Sum:", sum(numeric\_tuple))

#Immutable Nature: Attempt to modify elements of the tuple and handle the resulting error.

tuple(3)="PI" #ERROR

**OUTPUT**



1. **You are tasked with creating a Python program that showcases 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:**
2. **Create a Dictionary**: Define a dictionary with key-value pairs of different data types.({'name': 'Alice', 'age': 30, 'city': 'New York'}
3. **Access Values**: Access values using keys.
4. **Modify Dictionary**: Update values, add new key-value pairs, and remove existing pairs.
5. **Iterate Over Dictionary**: Use loops to iterate over keys or values.

**Sample Test Case**

Original Dictionary: {'name': 'Alice', 'age': 30, 'city': 'New York'}

Accessed name: Alice

Accessed age: 30

Updated Dictionary: {'name': 'James', 'age': 30, 'city': 'New York'}

After Removing 'city': {'name': 'James', 'age': 30}

KEY: name

KEY: age

Items in dictionary: dict\_items([('name', 'James'), ('age', 30)])

**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)

dictionary.pop('city')

print(dictionary)

#Iterate Over Dictionary: Use loops to iterate over keys or values.

for k in dictionary:

print("KEY:",k)

print(dictionary.items())

**OUTPUT**

