

Unit I Lab Exercises II
MCA171 Python Programming

Department of Computer Science, Christ University Central Campus

TOJIN VARKEY SIMSON

2447253 MCA-B

(1)Q: Write a Python program to create a module that performs basic list operations. Follow the instructions given below to write the program: (a) Create a module, named 'module_ListFunction' which has the following functions:

- A function that provides the maximum value in the defined list.
- A function that provides the minimum value.
- A function that provides the sum of all elements in a list.
- A function that provides the average of the list.
- A function that finds the median of a list.

(b) Create another script named 'main_ListOperations.py' and Imports the 'module_ListFunction' to it.

(c) Demonstrate the execution of each function with suitable examples.

A:

```
def find_max(lst):
```

```
return max(lst)
```

```
def find_min(lst):
```

```
    return min(lst)
```

```
def calculate_sum(lst):
```

```
    return sum(lst)
```

```
def calculate_average(lst):
```

```
    return sum(lst) / len(lst) if lst else 0
```

```
def find_median(lst):
```

```
    sorted_lst = sorted(lst)
```

```
    n = len(sorted_lst)
```

```
mid = n // 2
```

```
if n % 2 == 0:
```

```
    return (sorted_lst[mid - 1] + sorted_lst[mid]) / 2
```

```
else:
```

```
    return sorted_lst[mid]
```

```
from module_ListFunction import find_max, find_min, calculate_sum,  
    calculate_average, find_median
```

```
# Example lists
```

```
list1 = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
```

```
list2 = [10, 20, 30, 40, 50]
```

```
list3 = [7, 5, 3, 1, 2, 8, 9, 6, 4]
```

```
print("List 1:", list1)
```

```
print("Max:", find_max(list1))
```

```
print("Min:", find_min(list1))
```

```
print("Sum:", calculate_sum(list1))
```

```
print("Average:", calculate_average(list1))
```

```
print("Median:", find_median(list1))
```

```
print("\nList 2:", list2)
```

```
print("Max:", find_max(list2))
```

```
print("Min:", find_min(list2))
```

```
print("Sum:", calculate_sum(list2))
```

```
print("Average:", calculate_average(list2))
```

```
print("Median:", find_median(list2))
```

```
print("\nList 3:", list3)
```

```
print("Max:", find_max(list3))
```

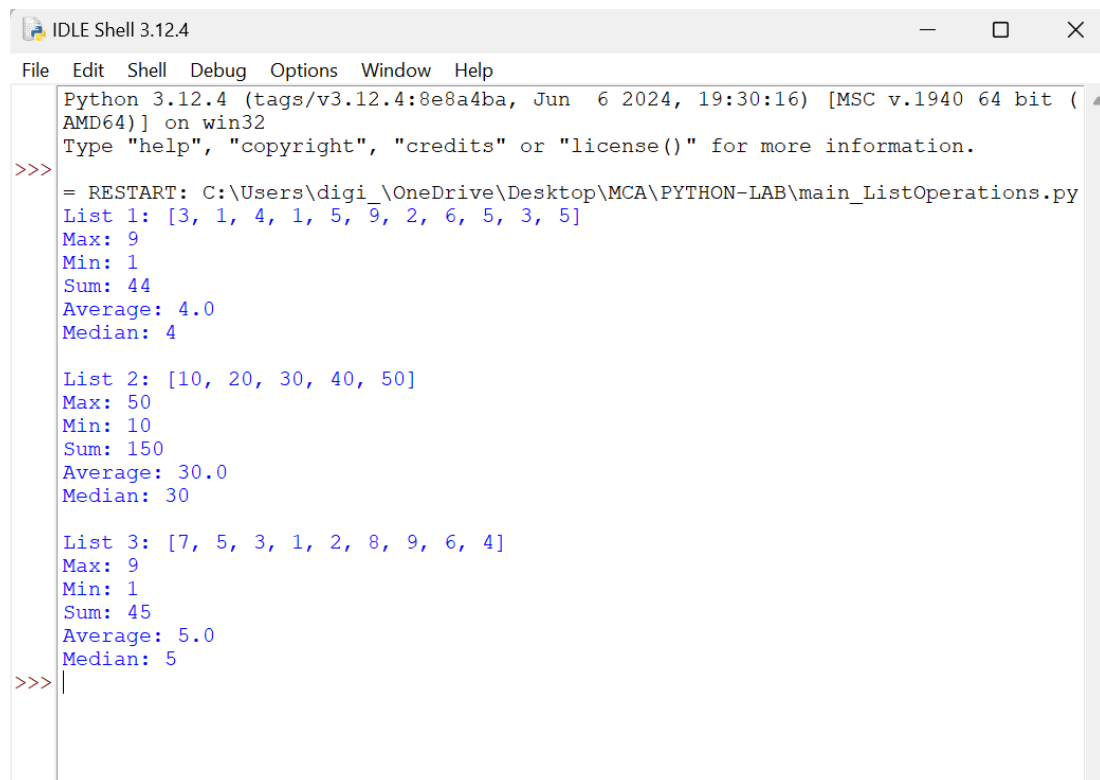
```
print("Min:", find_min(list3))
```

```
print("Sum:", calculate_sum(list3))
```

```
print("Average:", calculate_average(list3))
```

```
print("Median:", find_median(list3))
```

OUTPUT:



```
IDLE Shell 3.12.4
File Edit Shell Debug Options Window Help
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun 6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\digi_\OneDrive\Desktop\MCA\PYTHON-LAB\main_ListOperations.py
List 1: [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
Max: 9
Min: 1
Sum: 44
Average: 4.0
Median: 4

List 2: [10, 20, 30, 40, 50]
Max: 50
Min: 10
Sum: 150
Average: 30.0
Median: 30

List 3: [7, 5, 3, 1, 2, 8, 9, 6, 4]
Max: 9
Min: 1
Sum: 45
Average: 5.0
Median: 5
>>> |
```

(2)Q: Write a Python program to create a module that performs various set operations.

(a) Write a function that adds an element to a set.

(b) Write a function that removes an element from a set.

(c) Write a function that delivers the union and intersection of two sets. (d) Write a function that returns the difference of two sets say, S_1 and S_2 . (e) For any arbitrary sets S_1 and S_2 , write a function that checks if set S_1 is a subset of set S_2 .

(f) Write a function that finds the length of a given set without using the 'len()' function.

1

(g) Write a function to get all unique subsets of a given set.

Implement this module and demonstrate it by using adequate examples.

A: `def add_element(s, element):`

`s.add(element)`

`return s`

`def remove_element(s, element):`

`s.discard(element)`

`return s`

`def union_sets(s1, s2):`

`return s1 | s2`

`def intersection_sets(s1, s2):`

`return s1 & s2`

`def difference_sets(s1, s2):`

```
return s1 - s2
```

```
def is_subset(s1, s2):  
    return s1 <= s2
```

```
def set_length(s):  
    length = 0  
    for _ in s:  
        length += 1  
    return length
```

```
def symmetric_difference(s1, s2):  
    return s1 ^ s2
```

```
def power_set(s):  
    from itertools import chain, combinations  
    return set(chain.from_iterable(combinations(s, r) for r in range(len(s)+1)))
```

```
def unique_subsets(s):  
    from itertools import combinations  
    subs = set()  
    for i in range(len(s) + 1):  
        for combo in combinations(s, i):  
            subs.add(frozenset(combo))  
    return subs
```

```
from set_operations import *
```

```
# Example sets
```

```
s1 = {1, 2, 3}
```

```
s2 = {2, 3, 4}
```

```
# Adding an element
```

```
print("Add Element:")
```

```
print("Before:", s1)
```

```
s1 = add_element(s1, 5)
```

```
print("After:", s1)
```

```
# Removing an element
```

```
print("\nRemove Element:")
```

```
print("Before:", s1)
```

```
s1 = remove_element(s1, 5)
```

```
print("After:", s1)
```

```
# Union of sets
```

```
print("\nUnion of Sets:")
```

```
print("s1:", s1)
```

```
print("s2:", s2)
```

```
print("Union:", union_sets(s1, s2))
```

```
# Intersection of sets
```

```
print("\nIntersection of Sets:")
```

```
print("s1:", s1)
```

```
print("s2:", s2)
```

```
print("Intersection:", intersection_sets(s1, s2))
```

```
# Difference of sets
```

```
print("\nDifference of Sets (s1 - s2):")
```



```
print("s1:", s1)
print("s2:", s2)
print("Difference:", difference_sets(s1, s2))
```

```
# Check if s1 is a subset of s2
print("\nIs Subset (s1 <= s2):")
print("s1:", s1)
print("s2:", s2)
print("Is Subset:", is_subset(s1, s2))
```

```
# Length of set
print("\nLength of Set s1:")
print("s1:", s1)
print("Length:", set_length(s1))
```

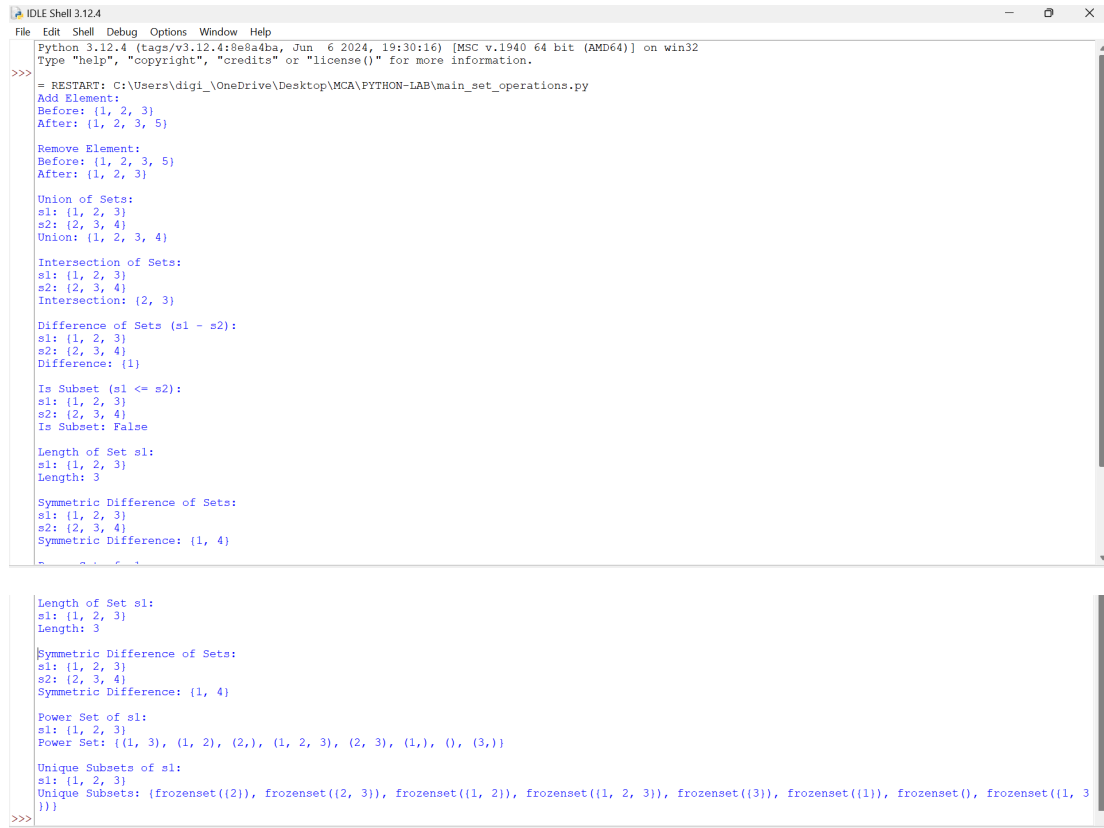
```
# Symmetric difference of sets
print("\nSymmetric Difference of Sets:")
print("s1:", s1)
print("s2:", s2)
print("Symmetric Difference:", symmetric_difference(s1, s2))
```

```
# Power set of s1
print("\nPower Set of s1:")
print("s1:", s1)
print("Power Set:", power_set(s1))
```

```
# Unique subsets of s1
print("\nUnique Subsets of s1:")
print("s1:", s1)
```

```
print("Unique Subsets:", unique_subsets(s1))
```

OUTPUT:



```
IDLE Shell 3.12.4
File Edit Shell Debug Options Window Help
Python 3.12.4 (tags/v3.12.4:9e8a4ba, Jun 6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\digi_OneDrive\Desktop\MCA\PYTHON-LAB\main_set_operations.py
Add Element:
Before: {1, 2, 3}
After: {1, 2, 3, 5}

Remove Element:
Before: {1, 2, 3, 5}
After: {1, 2, 3}

Union of Sets:
s1: {1, 2, 3}
s2: {2, 3, 4}
Union: {1, 2, 3, 4}

Intersection of Sets:
s1: {1, 2, 3}
s2: {2, 3, 4}
Intersection: {2, 3}

Difference of Sets (s1 - s2):
s1: {1, 2, 3}
s2: {2, 3, 4}
Difference: {1}

Is Subset (s1 <= s2):
s1: {1, 2, 3}
s2: {2, 3, 4}
Is Subset: False

Length of Set s1:
s1: {1, 2, 3}
Length: 3

Symmetric Difference of Sets:
s1: {1, 2, 3}
s2: {2, 3, 4}
Symmetric Difference: {1, 4}

Length of Set s1:
s1: {1, 2, 3}
Length: 3

Symmetric Difference of Sets:
s1: {1, 2, 3}
s2: {2, 3, 4}
Symmetric Difference: {1, 4}

Power Set of s1:
s1: {1, 2, 3}
Power Set: {(1, 3), (1, 2), (2, ), (1, 2, 3), (2, 3), (1, ), (), (3, )}

Unique Subsets of s1:
s1: {1, 2, 3}
Unique Subsets: {frozenset({2}), frozenset({2, 3}), frozenset({1, 2}), frozenset({1, 2, 3}), frozenset({3}), frozenset({1}), frozenset(), frozenset({1, 3})}
>>>
```

- (3)Q: Write a program to create functions that can accept multiple dictionaries as arguments using ‘*args’, and perform various operations on them. (a) Write a function, say, ‘merging_Dict(*args)’ that takes multiple dictio naries and merge them.**
- (b) Write a function which can find common keys in multiple dictionaries. (c) Create a function that inverts a dictionary by swapping its keys and values. If multiple keys have the same value in the original dictionary, they should be grouped in a list as the value in the inverted dictionary. Implement this program**

demonstrate it by using adequate examples.

A:

(a) Function to merge multiple dictionaries

```
def merging_Dict(*args):  
    merged_dict = {}  
    for dictionary in args:  
        merged_dict.update(dictionary)  
    return merged_dict
```

(b) Function to find common keys in multiple dictionaries

```
def common_keys(*args):  
    if not args:  
        return set()  
    common_keys = set(args[0].keys())  
    for dictionary in args[1:]:  
        common_keys &= set(dictionary.keys())  
    return common_keys
```

(c) Function to invert a dictionary, swapping keys and values

```
def invert_dict(d):  
    inverted_dict = {}  
    for key, value in d.items():  
        if value not in inverted_dict:  
            inverted_dict[value] = key  
        else:  
            if isinstance(inverted_dict[value], list):  
                inverted_dict[value].append(key)
```

```
        else:
            inverted_dict[value] = [inverted_dict[value], key]
    return inverted_dict
```

(d) Function to find common key-value pairs across multiple dictionaries

```
def common_key_value_pairs(*args):
    if not args:
        return {}
    common_pairs = set(args[0].items())
    for dictionary in args[1:]:
        common_pairs &= set(dictionary.items())
    return dict(common_pairs)
```

Demonstrating the functions with examples

```
dict1 = {'a': 1, 'b': 2, 'c': 3}
```

```
dict2 = {'b': 2, 'c': 4, 'd': 5}
```

```
dict3 = {'b': 2, 'c': 3, 'e': 6}
```

(a) Merging dictionaries

```
merged = merging_Dict(dict1, dict2, dict3)
```

```
print("Merged Dictionary:", merged)
```

(b) Finding common keys

```
common_keys_result = common_keys(dict1, dict2, dict3)
```

```
print("Common Keys:", common_keys_result)
```

(c) Inverting a dictionary

```
inverted = invert_dict(dict1)
```

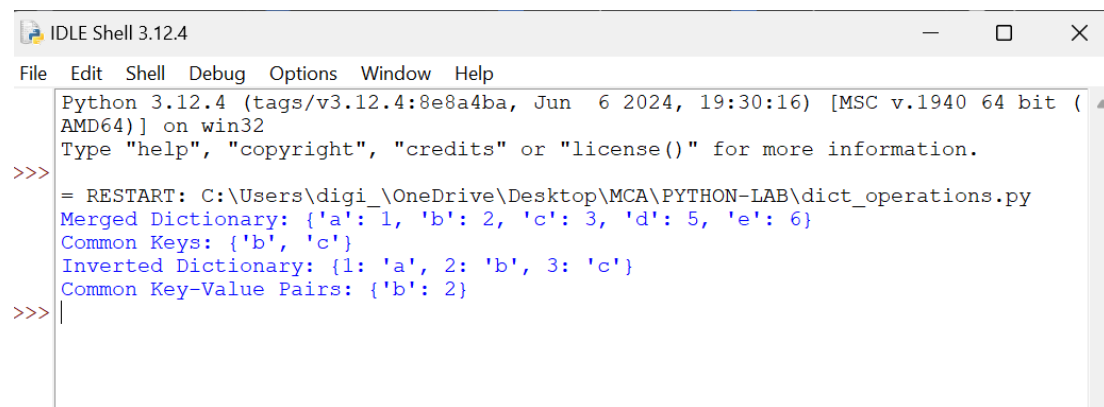
```
print("Inverted Dictionary:", inverted)
```

(d) Finding common key-value pairs

```
common_pairs = common_key_value_pairs(dict1, dict2, dict3)
```

```
print("Common Key-Value Pairs:", common_pairs)
```

OUTPUT:



```
IDLE Shell 3.12.4
File Edit Shell Debug Options Window Help
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun 6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\digi_\OneDrive\Desktop\MCA\PYTHON-LAB\dict_operations.py
Merged Dictionary: {'a': 1, 'b': 2, 'c': 3, 'd': 5, 'e': 6}
Common Keys: {'b', 'c'}
Inverted Dictionary: {1: 'a', 2: 'b', 3: 'c'}
Common Key-Value Pairs: {'b': 2}
>>>
```

(4)Q: Write a Python program to efficiently handle and manage the books in a library and this program should have functions to add a book, remove a book, and get the details of a book. Each book is represented with the following details as title, author, publisher, volume, year of publication, and ISBN (International Standard Book Number).

Follow the instructions given below to write the program:

- (a) Create a module named 'LibraryBooks.py' to manage books.**
- (b) Collect the information from web resources for 25 newly published books on subjects, namely Operating Systems, Data structures, and**

Machine Learning using Python, between the years 2020 and 2024 [The collected information may be stored in a Dictionary as key: value pairs]. (c) Create the functions for adding a book, removing a book, and getting the book details, and then place those functions in the created module, 'LibraryBooks.py'.

3

(d) Create another module named. 'mainLibraryManagement.py' to use these functions.

A:

```
# LibraryManager.py
```

```
class LibraryManager:
```

```
    def __init__(self):
```

```
        self.books = {}
```

```
    def add_book(self, isbn, title, author, publisher, volume, year, isbn_num):
```

```
        if isbn in self.books:
```

```
            print(f"Book with ISBN {isbn} already exists.")
```

```
            return
```

```
        self.books[isbn] = {
```

```
            'title': title,
```

```
        'author': author,

        'publisher': publisher,

        'volume': volume,

        'year': year,

        'isbn_num': isbn_num

    }

    print(f"Book '{title}' added successfully.")
```

```
def remove_book(self, isbn):

    if isbn in self.books:

        del self.books[isbn]

        print(f"Book with ISBN {isbn} removed successfully.")

    else:

        print(f"Book with ISBN {isbn} not found.")
```

```
def get_book(self, isbn):

    return self.books.get(isbn, "Book not found.")
```

```
def search_books(self, keyword, by="title"):
```

```
        results = [book for book in self.books.values() if keyword.lower() in  
                    book[by].lower()]
```

```
    return results
```

```
def list_books(self):
```

```
    return list(self.books.values())
```

```
def update_book(self, isbn, **kwargs):
```

```
    if isbn in self.books:
```

```
        self.books[isbn].update(kwargs)
```

```
        print(f"Book with ISBN {isbn} updated successfully.")
```

```
    else:
```

```
        print(f"Book with ISBN {isbn} not found.")
```

```
def is_available(self, isbn):
```

```
    return isbn in self.books
```

```
# Demonstration
```

```
if __name__ == "__main__":
```



```
library = LibraryManager()
```

```
# Adding sample books
```

```
library.add_book("978-3-16-148410-0", "Operating Systems  
Concepts", "Abraham Silberschatz", "Wiley", "10th", 2021,  
"978-3-16-148410-0")
```

```
library.add_book("978-0-13-409341-3", "Data Structures and  
Algorithms in Python", "Michael T. Goodrich", "Wiley", "1st", 2020,  
"978-0-13-409341-3")
```

```
library.add_book("978-0-262-03384-8", "Machine Learning", "Tom M.  
Mitchell", "McGraw Hill", "1st", 2021, "978-0-262-03384-8")
```

```
# Removing a book
```

```
library.remove_book("978-0-13-409341-3")
```

```
# Retrieving book details
```

```
print(library.get_book("978-3-16-148410-0"))
```

```
# Searching for books by title
```

```
print(library.search_books("Machine Learning"))
```

```
# Listing all books
```

```
print(library.list_books())
```

```
# Updating book details
```

```
library.update_book("978-3-16-148410-0", title="Operating Systems  
Concepts Updated Edition")
```

```
# Checking if a book is available
```

```
print(library.is_available("978-0-262-0384-8"))
```

```
print(library.is_available("978-0-13-409341-3"))
```

OUTPUT:

```
Python Shell 3.12.4
File Edit Shell Debug Options Window Help
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun 6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\digi_OneDrive\Desktop\MCA\PYTHON-LAB\LibraryManager.py
Book 'Operating Systems Concepts' added successfully.
Book 'Data Structures and Algorithms in Python' added successfully.
Book 'Machine Learning' added successfully.
Book with ISBN 978-0-13-409341-3 removed successfully.
[{'title': 'Operating Systems Concepts', 'author': 'Abraham Silberschatz', 'publisher': 'Wiley', 'volume': '10th', 'year': 2021, 'isbn_num': '978-3-16-148410-0'}]
[{'title': 'Machine Learning', 'author': 'Tom M. Mitchell', 'publisher': 'McGraw Hill', 'volume': '1st', 'year': 2021, 'isbn_num': '978-0-262-0384-8'}]
[{'title': 'Operating Systems Concepts', 'author': 'Abraham Silberschatz', 'publisher': 'Wiley', 'volume': '10th', 'year': 2021, 'isbn_num': '978-3-16-148410-0'}, {'title': 'Machine Learning', 'author': 'Tom M. Mitchell', 'publisher': 'McGraw Hill', 'volume': '1st', 'year': 2021, 'isbn_num': '978-0-262-0384-8'}]
Book with ISBN 978-3-16-148410-0 updated successfully.
False
False
>>>
```

(5) Q:Write a Python program for a wether analyzing system to process and analyze weather data for Bangalore City from 20th July to 26th July in 2024. (a) Each day's data includes:

- **Date**
- **Maximum temperature (in Celsius)**
- **Minimum temperature (in Celsius)**
- **Humidity (in percentage)**

[Hint: The data may be stored in a list of dictionaries]

(b) Write a function that finds the average maximum and minimum temper atures from the weather data.

(c) Write a function that calculates the average humidity over the given period

A:

#a

```
weather=[{"Date":"20th July","Maximum temperature (in
Celsius)":28,"Minimum temperature (in Celsius)":25,"Humidity (in
percentage)":77},
{"Date":"21th July","Maximum temperature (in
Celsius)":27,"Minimum temperature (in Celsius)":24,"Humidity (in
percentage)":81},
{"Date":"22th July","Maximum temperature (in
Celsius)":27,"Minimum temperature (in Celsius)":22,"Humidity (in
percentage)":84},
{"Date":"23th July","Maximum temperature (in
Celsius)":28,"Minimum temperature (in Celsius)":25,"Humidity (in
percentage)":76},
{"Date":"24th July","Maximum temperature (in
Celsius)":29,"Minimum temperature (in Celsius)":26,"Humidity (in
```

```
percentage)":64},  
    {"Date":"25th July","Maximum temperature (in  
Celsius)":28,"Minimum temperature (in Celsius)":22,"Humidity (in  
percentage)":82},  
    {"Date":"26th July","Maximum temperature (in  
Celsius)":27,"Minimum temperature (in Celsius)":24,"Humidity (in  
percentage)":81}]
```

#b

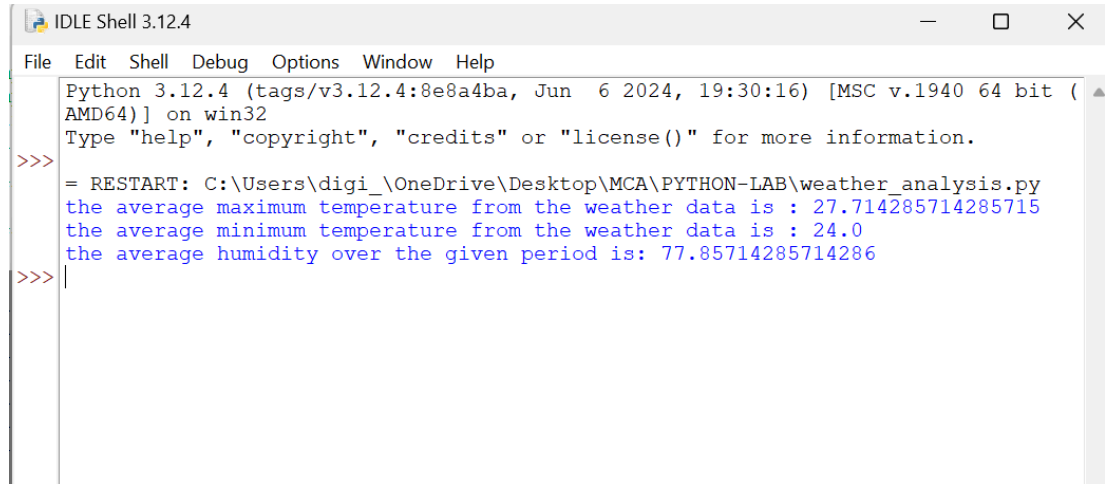
```
def avg_temp(weather):  
    sum1=sum2=0  
    for i in weather:  
        sum1+=i["Maximum temperature (in Celsius)"]  
        sum2+=i["Minimum temperature (in Celsius)"]  
    avg1=sum1/7  
    avg2=sum2/7  
    print("the average maximum temperature from the weather data is  
: ",avg1)  
    print("the average minimum temperature from the weather data is  
: ",avg2)  
avg_temp(weather)
```

#c

```
def avg_humid(weather):  
    sum=0  
    for i in weather:  
        sum+=i["Humidity (in percentage)"]  
    avg=sum/7  
    print("the average humidity over the given period is:",avg)
```

avg_humid(weather)

OUTPUT:



```
IDLE Shell 3.12.4
File Edit Shell Debug Options Window Help
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun 6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\digi_\OneDrive\Desktop\MCA\PYTHON-LAB\weather_analysis.py
the average maximum temperature from the weather data is : 27.714285714285715
the average minimum temperature from the weather data is : 24.0
the average humidity over the given period is: 77.85714285714286
>>>
```