module_ListFunction.py U | setoperation.py U | secondprogram.py U | fifthprogram.py U | LibraryManager.py U

module_ListFunction.py > compute_average

```python
def find_max(lst):
    if not lst:
        return None
    return max(lst)

def find_min(lst):
    if not lst:
        return None
    return min(lst)

def calculate_sum(lst):
    return sum(lst)

def compute_average(lst):
    if not lst:
        return None
    return sum(lst) / len(lst)

def determine_median(lst):
    if not lst:
        return None
    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]

if __name__ == "__main__":
    list1 = [x for x in range(10)]
    list2 = [x**2 for x in range(10)]
    list3 = [x for x in range(1, 21) if x % 2 == 0]

    print("List1:", list1)
    print("Max value in List1:", find_max(list1))
    print("Min value in List1:", find_min(list1))
```

Ln 15, Col 1

new.py > ...

```python
import module_ListFunction as mlf

list1 = [x for x in range(10)]
list2 = [x**2 for x in range(10)]
list3 = [x for x in range(1, 21) if x % 2 == 0]

print("List1:", list1)
print("Max value in List1:", mlf.find_max(list1))
print("Min value in List1:", mlf.find_min(list1))
print("Sum of List1:", mlf.calculate_sum(list1))
print("Average of List1:", mlf.compute_average(list1))
print("Median of List1:", mlf.determine_median(list1))

print("\nList2:", list2)
print("Max value in List2:", mlf.find_max(list2))
print("Min value in List2:", mlf.find_min(list2))
print("Sum of List2:", mlf.calculate_sum(list2))
print("Average of List2:", mlf.compute_average(list2))
print("Median of List2:", mlf.determine_median(list2))

print("\nList3:", list3)
print("Max value in List3:", mlf.find_max(list3))
print("Min value in List3:", mlf.find_min(list3))
print("Sum of List3:", mlf.calculate_sum(list3))
print("Average of List3:", mlf.compute_average(list3))
print("Median of List3:", mlf.determine_median(list3))

```

```
Median of List3: 11.0
PS C:\Users\Prath\Desktop\Python 1> & C:/Users/Prath/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/Prath/Desktop/Python 1/new.py"
List1: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Max value in List1: 9
Min value in List1: 0
Sum of List1: 45
Average of List1: 4.5
Median of List1: 4.5

List2: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
Max value in List2: 81
Min value in List2: 0
Sum of List2: 285
Average of List2: 28.5
Median of List2: 20.5

List3: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
Max value in List3: 20
Min value in List3: 2
Sum of List3: 110
Average of List3: 11.0
Median of List3: 11.0
PS C:\Users\Prath\Desktop\Python 1>
```

firstprogram2.py > ...

```python
import module_ListFunction as mlf


list1 = [x for x in range(10)]
list2 = [x**2 for x in range(10)]
list3 = [x for x in range(1, 21) if x % 2 == 0]


print("List1:", list1)
print("Max value in List1:", mlf.find_max(list1))
print("Min value in List1:", mlf.find_min(list1))
print("Sum of List1:", mlf.calculate_sum(list1))
print("Average of List1:", mlf.compute_average(list1))
print("Median of List1:", mlf.determine_median(list1))

print("\nList2:", list2)
print("Max value in List2:", mlf.find_max(list2))
print("Min value in List2:", mlf.find_min(list2))
print("Sum of List2:", mlf.calculate_sum(list2))
print("Average of List2:", mlf.compute_average(list2))
print("Median of List2:", mlf.determine_median(list2))

print("\nList3:", list3)
print("Max value in List3:", mlf.find_max(list3))
print("Min value in List3:", mlf.find_min(list3))
print("Sum of List3:", mlf.calculate_sum(list3))
print("Average of List3:", mlf.compute_average(list3))
print("Median of List3:", mlf.determine_median(list3))

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Max value in List3: 20
Min value in List3: 2
Sum of List3: 110
Average of List3: 11.0
Median of List3: 11.0
PS C:\Users\Prath\Desktop\Python 1>
```

3)

thirdprogram.py > ...

```python
def merging_dict(*args):
    merged_dict = {}
    for d in args:
        if isinstance(d, dict):
            merged_dict.update(d)
        else:
            raise ValueError("All arguments must be dictionaries")
    return merged_dict

def common_keys(*args):
    if not args:
        return set()

    common_keys_set = set(args[0].keys())
    for d in args[1:]:
        common_keys_set.intersection_update(d.keys())

    return common_keys_set

def invert_dict(d):
    inverted_dict = {}
    for key, value in d.items():
        if value in inverted_dict:
            if isinstance(inverted_dict[value], list):
                inverted_dict[value].append(key)
            else:
                inverted_dict[value] = [inverted_dict[value], key]
        else:
            inverted_dict[value] = key
    return inverted_dict

def common_key_value_pairs(*args):
    if not args:
        return {}

    common_pairs = set(args[0].items())
    for d in args[1:]:
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Prath\Desktop\Python 1> & C:/Users/Prath/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/Prath/Desktop/Python 1/thirdprogram.py"
Merged Dictionary: {'a': 1, 'b': 2, 'c': 3, 'd': 5, 'e': 6}
Common Keys: {'c'}
Inverted Dictionary: {1: 'a', 2: 'b', 3: 'c'}
Inverted Dictionary: {1: 'a', 2: 'b', 3: 'c'}
Common Key-Value Pairs: {}
PS C:\Users\Prath\Desktop\Python 1>
```

LibraryManager.py > LibraryManager > add_book

```python
class LibraryManager:
    def __init__(self):
        self.library = {}

    def add_book(self, title, author, publisher, volume, year, isbn):
        if isbn in self.library:
            print("Book with this ISBN already exists.")
        else:
            self.library[isbn] = {
                'Title': title,
                'Author': author,
                'Publisher': publisher,
                'Volume': volume,
                'Year': year,
                'ISBN': isbn
            }
            print("Book added successfully.")

    def remove_book(self, isbn):
        if isbn in self.library:
            del self.library[isbn]
            print("Book removed successfully.")
        else:
            print("Book with this ISBN not found.")

    def retrieve_book(self, isbn):
        book = self.library.get(isbn)
        if book:
            return book
        else:
            print("Book with this ISBN not found.")
            return None

    def search_books(self, title=None, author=None):
        results = []
        for book in self.library.values():
            if (title and title.lower() in book['Title'].lower()) or \
```

```
Book details updated successfully.
Book Availability: True
PS C:\Users\Prath\Desktop\Python 1> & C:/Users/Prath/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/Prath/Desktop/Python 1/LibraryManager.py"
Book added successfully.
Book added successfully.
Book added successfully.
Book removed successfully.
Book Details: {'Title': 'Introduction to Operating Systems', 'Author': 'John Doe', 'Publisher': 'Tech Publisher', 'Volume': '1', 'Year': 2022, 'ISBN': '978-1234567890'}
Search Results by Title: [{'Title': 'Machine Learning Basics', 'Author': 'Alice Johnson', 'Publisher': 'AI Publisher', 'Volume': '1', 'Year': 2023, 'ISBN': '978-1112131415'}]
Search Results by Author: [{'Title': 'Introduction to Operating Systems', 'Author': 'John Doe', 'Publisher': 'Tech Publisher', 'Volume': '1', 'Year': 2022, 'ISBN': '978-1234567890'}]
All Books: [{'Title': 'Introduction to Operating Systems', 'Author': 'John Doe', 'Publisher': 'Tech Publisher', 'Volume': '1', 'Year': 2022, 'ISBN': '978-1234567890'}, {'Title': 'Machine Learning Basic
s', 'Author': 'Alice Johnson', 'Publisher': 'AI Publisher', 'Volume': '1', 'Year': 2023, 'ISBN': '978-1112131415'}]
Book details updated successfully.
Book Availability: True
Book added successfully.
Book added successfully.
Book added successfully.
Book removed successfully.
Book Details: {'Title': 'Introduction to Operating Systems', 'Author': 'John Doe', 'Publisher': 'Tech Publisher', 'Volume': '1', 'Year': 2022, 'ISBN': '978-1234567890'}
Search Results by Title: [{'Title': 'Machine Learning Basics', 'Author': 'Alice Johnson', 'Publisher': 'AI Publisher', 'Volume': '1', 'Year': 2023, 'ISBN': '978-1112131415'}]
Search Results by Author: [{'Title': 'Introduction to Operating Systems', 'Author': 'John Doe', 'Publisher': 'Tech Publisher', 'Volume': '1', 'Year': 2022, 'ISBN': '978-1234567890'}]
All Books: [{'Title': 'Introduction to Operating Systems', 'Author': 'John Doe', 'Publisher': 'Tech Publisher', 'Volume': '1', 'Year': 2022, 'ISBN': '978-1234567890'}, {'Title': 'Machine Learning Basic
s', 'Author': 'Alice Johnson', 'Publisher': 'AI Publisher', 'Volume': '1', 'Year': 2023, 'ISBN': '978-1112131415'}]
Book details updated successfully.
Book Availability: True
PS C:\Users\Prath\Desktop\Python 1>
```

fifthprogram.py > ...

```python
weather_data = [
    {'Date': '2024-07-01', 'Max Temp': 32, 'Min Temp': 22, 'Humidity': 60},
    {'Date': '2024-07-02', 'Max Temp': 30, 'Min Temp': 21, 'Humidity': 65},
    {'Date': '2024-07-03', 'Max Temp': 35, 'Min Temp': 24, 'Humidity': 70},
    {'Date': '2024-07-04', 'Max Temp': 33, 'Min Temp': 23, 'Humidity': 68},
    {'Date': '2024-07-05', 'Max Temp': 29, 'Min Temp': 20, 'Humidity': 75},
    {'Date': '2024-07-06', 'Max Temp': 31, 'Min Temp': 22, 'Humidity': 64},
    {'Date': '2024-07-07', 'Max Temp': 28, 'Min Temp': 19, 'Humidity': 80}
]


def find_highest_lowest_temperatures(data):
    max_temps = [entry['Max Temp'] for entry in data]
    min_temps = [entry['Min Temp'] for entry in data]

    highest_temp = max(max_temps)
    lowest_temp = min(min_temps)

    return highest_temp, lowest_temp

def count_days_above_30(data):
    count = sum(1 for entry in data if entry['Max Temp'] > 30)
    return count

def compute_average_humidity(data):
    humidities = [entry['Humidity'] for entry in data]
    average_humidity = sum(humidities) / len(humidities)
    return average_humidity

highest_temp, lowest_temp = find_highest_lowest_temperatures(weather_data)
print(f"Highest Temperature: {highest_temp}°C")
print(f"Lowest Temperature: {lowest_temp}°C")
```

```
Highest Temperature: 35°C
Lowest Temperature: 19°C
Number of Days Above 30°C: 4
Average Humidity: 68.85714285714286%
PS C:\Users\Prath\Desktop\Python 1>
```

5)

fifthprogram.py > find_highest_lowest_temperatures

```python
weather_data = [
    {'Date': '2024-07-01', 'Max Temp': 32, 'Min Temp': 22, 'Humidity': 60},
    {'Date': '2024-07-02', 'Max Temp': 30, 'Min Temp': 21, 'Humidity': 65},
    {'Date': '2024-07-03', 'Max Temp': 35, 'Min Temp': 24, 'Humidity': 70},
    {'Date': '2024-07-04', 'Max Temp': 33, 'Min Temp': 23, 'Humidity': 68},
    {'Date': '2024-07-05', 'Max Temp': 29, 'Min Temp': 20, 'Humidity': 75},
    {'Date': '2024-07-06', 'Max Temp': 31, 'Min Temp': 22, 'Humidity': 64},
    {'Date': '2024-07-07', 'Max Temp': 28, 'Min Temp': 19, 'Humidity': 80}
]


def find_highest_lowest_temperatures(data):
    max_temps = [entry['Max Temp'] for entry in data]
    min_temps = [entry['Min Temp'] for entry in data]

    highest_temp = max(max_temps)
    lowest_temp = min(min_temps)

    return highest_temp, lowest_temp

def count_days_above_30(data):
    count = sum(1 for entry in data if entry['Max Temp'] > 30)
    return count

def compute_average_humidity(data):
    humidities = [entry['Humidity'] for entry in data]
    average_humidity = sum(humidities) / len(humidities)
    return average_humidity


highest_temp, lowest_temp = find_highest_lowest_temperatures(weather_data)
print(f"Highest Temperature: {highest_temp}°C")
print(f"Lowest Temperature: {lowest_temp}°C")

days_above_30 = count_days_above_30(weather_data)
print(f"Number of Days Above 30°C: {days_above_30}")

```

```
PS C:\Users\Prath\Desktop\Python 1> & C:/Users/Prath/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/Prath/Desktop/Python 1/fifthprogram.py"
Highest Temperature: 35°C
Lowest Temperature: 19°C
Number of Days Above 30°C: 4
Average Humidity: 68.85714285714286%
PS C:\Users\Prath\Desktop\Python 1>
```