Day 8 – Python Session – Accenture bootcamp

Name: Pothumulla Kankanamge Mewan Madhusha

Table of Contents

Task 01

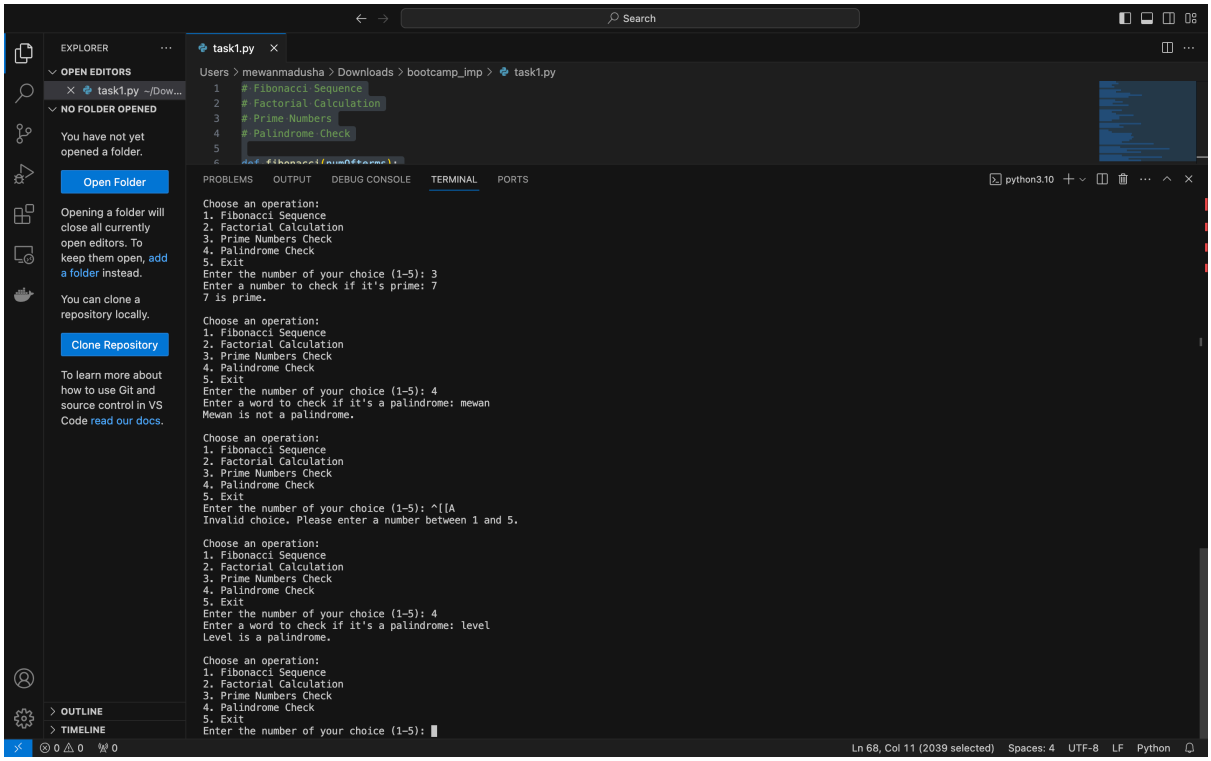
Please create a document, write a Python program, and share it in this chat.
Fibonacci Sequence
Factorial Calculation
Prime Numbers
Palindrome Check

Answer

```python
# Fibonacci Sequence
# Factorial Calculation
# Prime Numbers
# Palindrome Check

def fibonacci(numOfterms):
    fibo_seq = [0,1]
    while len(fibo_seq) < numOfterms:
       fibo_seq.append(fibo_seq[-1] + fibo_seq[-2])
    return fibo_seq[:numOfterms]

def factorial(num):
    if num == 0 or num == 1:
        return 1
    return num * factorial(num - 1)


def prime_numbercheck(num):
    if num < 2:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

# i have just used python reversed string function
def is_palindrome(word):
    return word == word[::-1]

def main():
    while True:
        print("\nChoose an operation:")
        print("1. Fibonacci Sequence")
        print("2. Factorial Calculation")
        print("3. Prime Numbers Check")
        print("4. Palindrome Check")
        print("5. Exit")
        choice = input("Enter the number of your choice (1-5): ")

        if choice == '1':
```

```
            n = int(input("Enter the number of terms for the Fibonacci sequence:
"))
            result = fibonacci(n)
            print(f"Fibonacci Sequence: {result}")

        elif choice == '2':
            num = int(input("Enter a number to calculate its factorial: "))
            result = factorial(num)
            print(f"Factorial of {num}: {result}")

        elif choice == '3':
            num = int(input("Enter a number to check if it's prime: "))
            result = prime_numbercheck(num)
            print(f"{num} is {'prime' if result else 'not prime'}.")

        elif choice == '4':
            word = input("Enter a word to check if it's a palindrome: ").lower()
            result = is_palindrome(word)
            print(f"{word.capitalize()} is {'a palindrome' if result else 'not a
palindrome'}.")

        elif choice == '5':
            print("Exiting the program. Goodbye!")
            break

        else:
            print("Invalid choice. Please enter a number between 1 and 5.")

if __name__ == "__main__":
    main()
```

Output answers


Search
EXPLORER
OPEN EDITORS
task1.py ~/Dow...
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder
Opening a folder will close all currently open editors. To keep them open, add a folder instead.
You can clone a repository locally.
Clone Repository
To learn more about how to use Git and source control in VS Code read our docs.
OUTLINE
TIMELINE
task1.py
Users > mewanmadusha > Downloads > bootcamp_imp > task1.py
1 # Fibonacci Sequence
2 # Factorial Calculation
3 # Prime Numbers
4 # Palindrome Check
5
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
python3.10
Choose an operation:
1. Fibonacci Sequence
2. Factorial Calculation
3. Prime Numbers Check
4. Palindrome Check
5. Exit
Enter the number of your choice (1-5): 3
Enter a number to check if it's prime: 7
7 is prime.
Choose an operation:
1. Fibonacci Sequence
2. Factorial Calculation
3. Prime Numbers Check
4. Palindrome Check
5. Exit
Enter the number of your choice (1-5): 4
Enter a word to check if it's a palindrome: mewan
Mewan is not a palindrome.
Choose an operation:
1. Fibonacci Sequence
2. Factorial Calculation
3. Prime Numbers Check
4. Palindrome Check
5. Exit
Enter the number of your choice (1-5): ^[[A
Invalid choice. Please enter a number between 1 and 5.
Choose an operation:
1. Fibonacci Sequence
2. Factorial Calculation
3. Prime Numbers Check
4. Palindrome Check
5. Exit
Enter the number of your choice (1-5): 4
Enter a word to check if it's a palindrome: level
Level is a palindrome.
Choose an operation:
1. Fibonacci Sequence
2. Factorial Calculation
3. Prime Numbers Check
4. Palindrome Check
5. Exit
Enter the number of your choice (1-5):
Ln 68, Col 11 (2039 selected) Spaces: 4 UTF-8 LF Python

Task 2

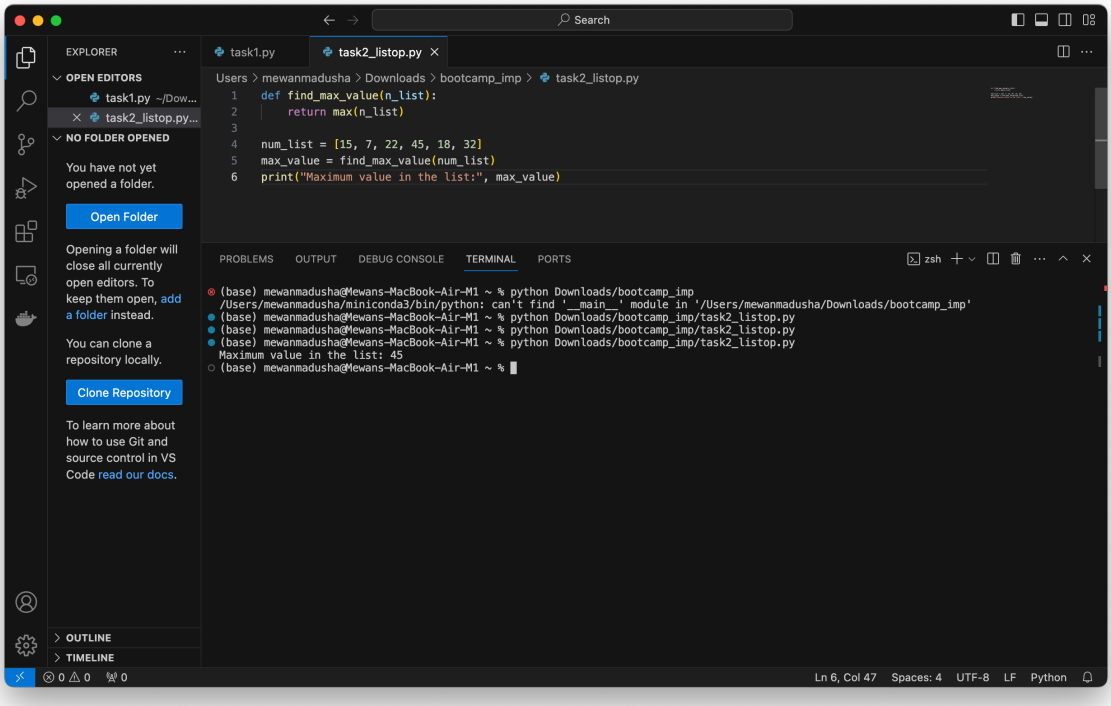
Complete this task. You can seek help online, but if you understand the process, feel free to proceed. Otherwise, note down your questions, and we will discuss them in the end-time session.

List Operations:

1. Write a Python function to find the maximum value in a list of numbers.

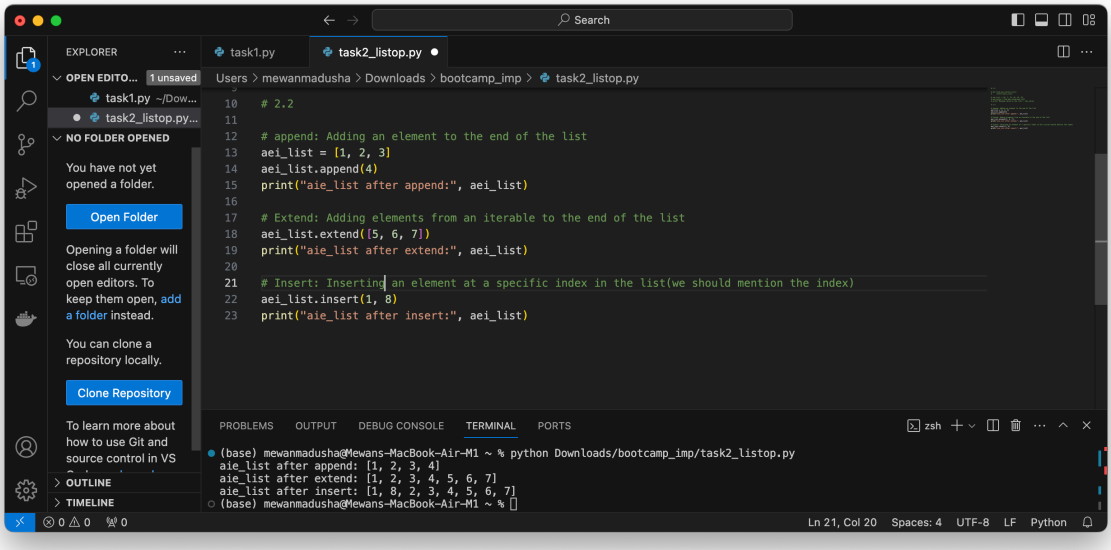
```python
def find_max_value(n_list):
    return max(n_list)

num_list = [15, 7, 22, 45, 18, 32]
max_value = find_max_value(num_list)
print("Maximum value in the list:", max_value)
```

Answer



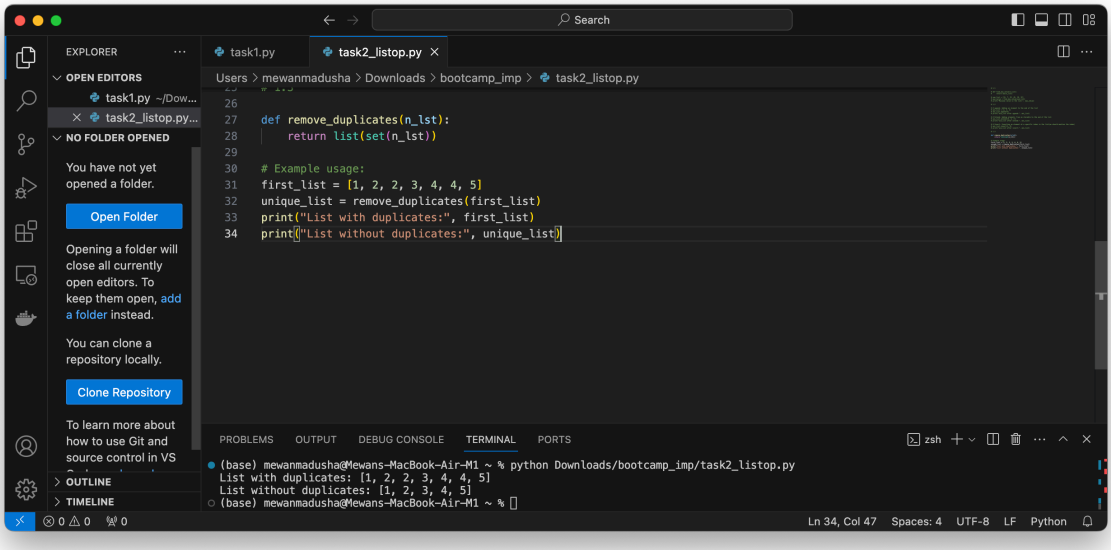
2. append, Extend and Insert works explain in code.

```python
# append: Adding an element to the end of the list
aei_list = [1, 2, 3]
aei_list.append(4)
print("aie_list after append:", aei_list)

# Extend: Adding elements from an iterable to the end of the list
aei_list.extend([5, 6, 7])
print("aie_list after extend:", aei_list)

# Insert: Inserting an element at a specific index in the list(we should mention
the index)
aei_list.insert(1, 8)
print("aie_list after insert:", aei_list)
```



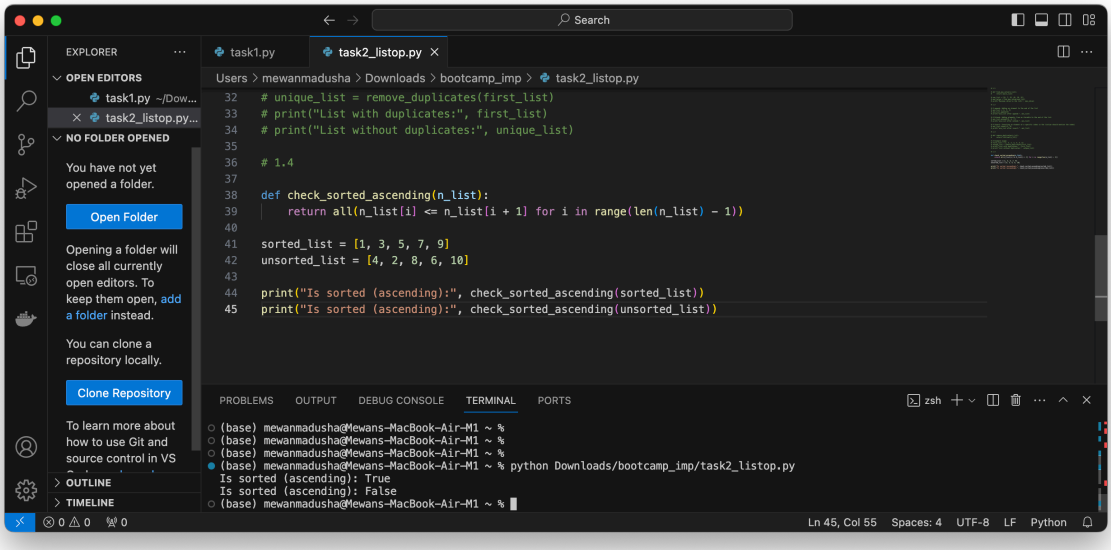
3. Implement a program to remove duplicates from a given list.

```python
def remove_duplicates(n_lst):
    return list(set(n_lst))

# Example usage:
first_list = [1, 2, 2, 3, 4, 4, 5]
unique_list = remove_duplicates(first_list)
print("List with duplicates:", first_list)
print("List without duplicates:", unique_list)
```



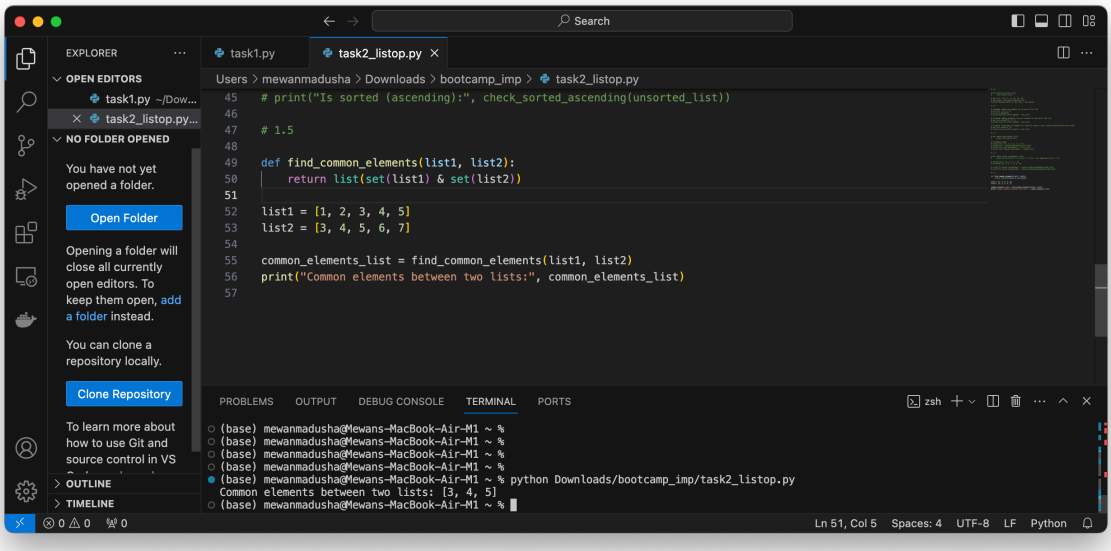
4. Create a function to check if a list is sorted in ascending order.

```python
def check_sorted_ascending(n_list):
    return all(n_list[i] <= n_list[i + 1] for i in range(len(n_list) - 1))

sorted_list = [1, 3, 5, 7, 9]
unsorted_list = [4, 2, 8, 6, 10]

print("Is sorted (ascending):", check_sorted_ascending(sorted_list))
print("Is sorted (ascending):", check_sorted_ascending(unsorted_list))
```



5. Write a program to find the common elements between two lists.

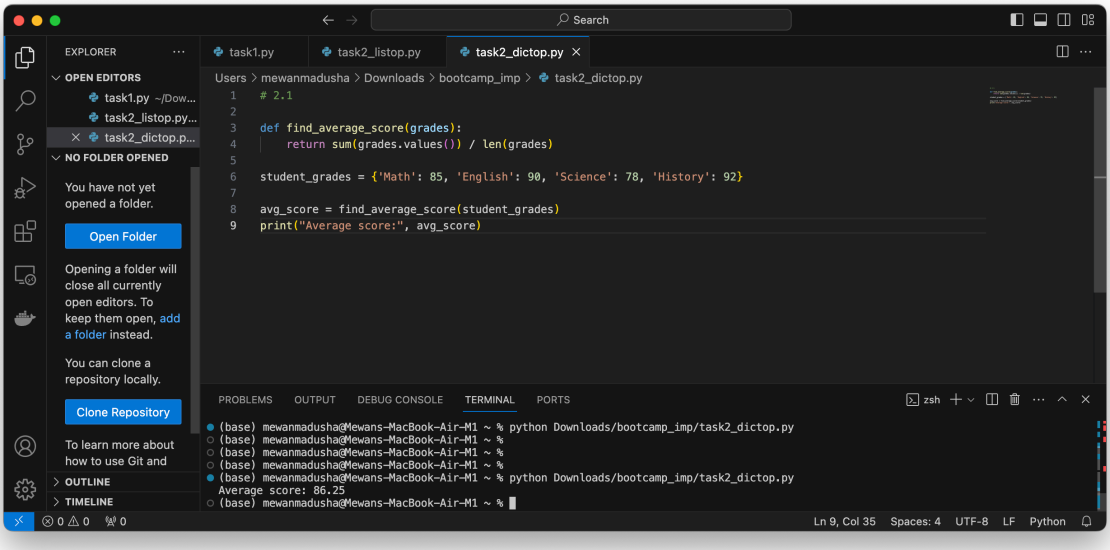
```
def find_common_elements(list1, list2):
    return list(set(list1) & set(list2))

list1 = [1, 2, 3, 4, 5]
list2 = [3, 4, 5, 6, 7]

common_elements_list = find_common_elements(list1, list2)
print("Common elements between two lists:", common_elements_list)
```



Dictionary Operations:

1. Create a Python dictionary representing a student's grades (subject: score). Write a function to find the average score.

```
def find_average_score(grades):
    return sum(grades.values()) / len(grades)

student_grades = {'Math': 85, 'English': 90, 'Science': 78, 'History': 92}

avg_score = find_average_score(student_grades)
print("Average score:", avg_score)
```



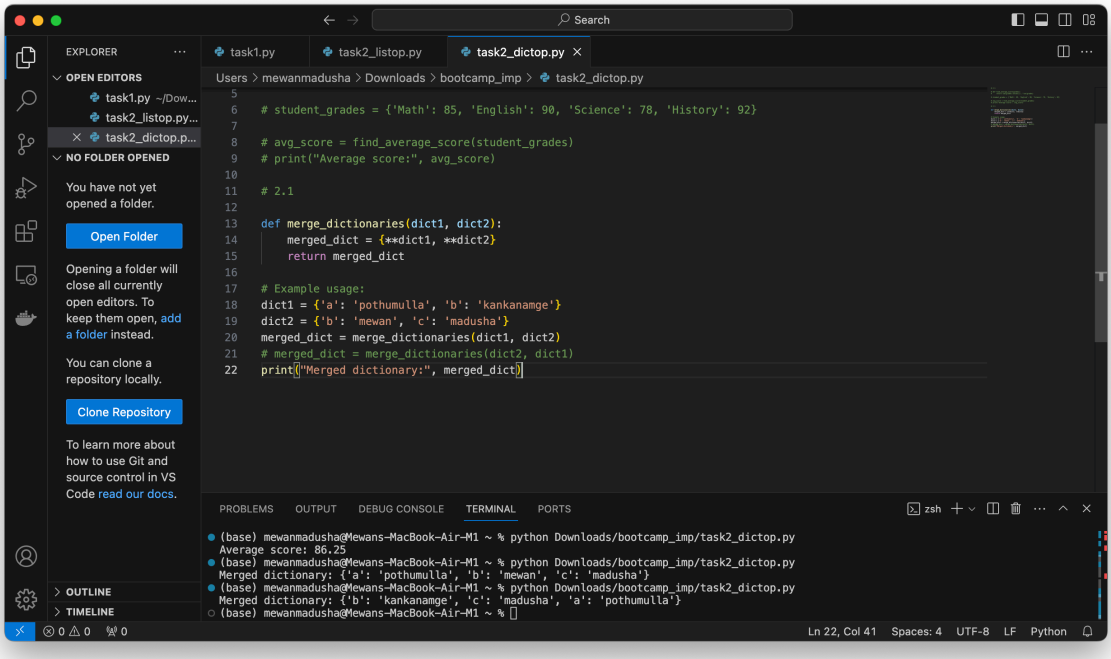
2. Implement a program to merge two dictionaries.

Hear I used dictionary unpacking syntax that available after python 3.5

```
def merge_dictionaries(dict1, dict2):
    merged_dict = {**dict1, **dict2}
    return merged_dict

# Example usage:
dict1 = {'a': 'pothumulla', 'b': 'kankanamge'}
dict2 = {'b': 'mewan', 'c': 'madusha'}
merged_dict = merge_dictionaries(dict1, dict2)
# merged_dict = merge_dictionaries(dict2, dict1)
print("Merged dictionary:", merged_dict)
```

Merged dictionary included all the keys but if there is any common keys, based on the unpacking second dictionary value will be the override value of the merged_dict.

In the below code I also shown both above scenarios.



3. Create a program to find the keys with the top N values in a dictionary.

Sorted list returns a new sorted list from the elements of the provided iterable

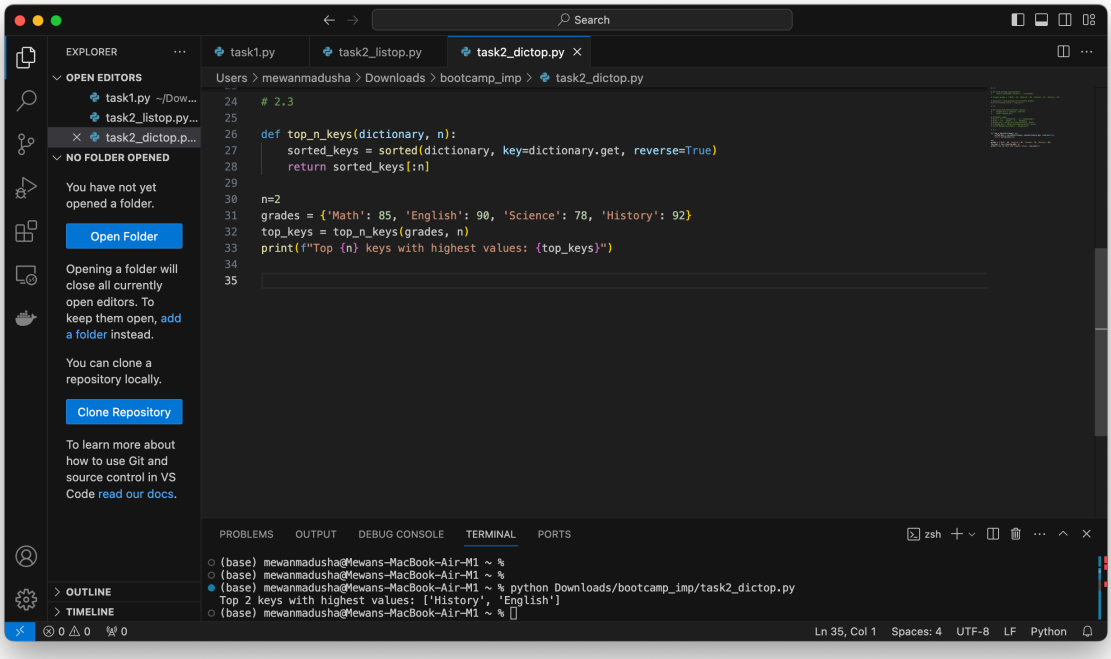
Iterable is values of dictionary, dictionary.get means sorting based on dictionary values

Reverse=true – sort in to descending order because in the slicing step we can cut out elements from start

[:n] – slicing the list and then get only first N number of elements

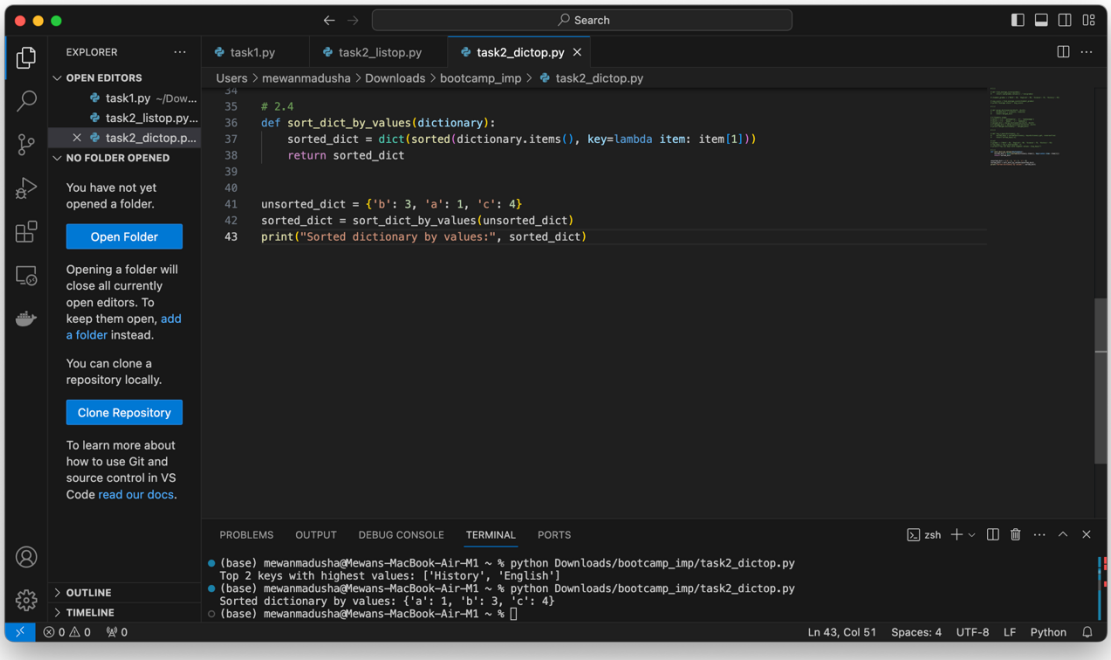
Assume n=2

```
def top_n_keys(dictionary, n):
    sorted_keys = sorted(dictionary, key=dictionary.get, reverse=True)
    return sorted_keys[:n]

n=2
grades = {'Math': 85, 'English': 90, 'Science': 78, 'History': 92}
top_keys = top_n_keys(grades, n)
print(f"Top {n} keys with highest values: {top_keys}")
```



4. Implement a function to sort a dictionary by its values.

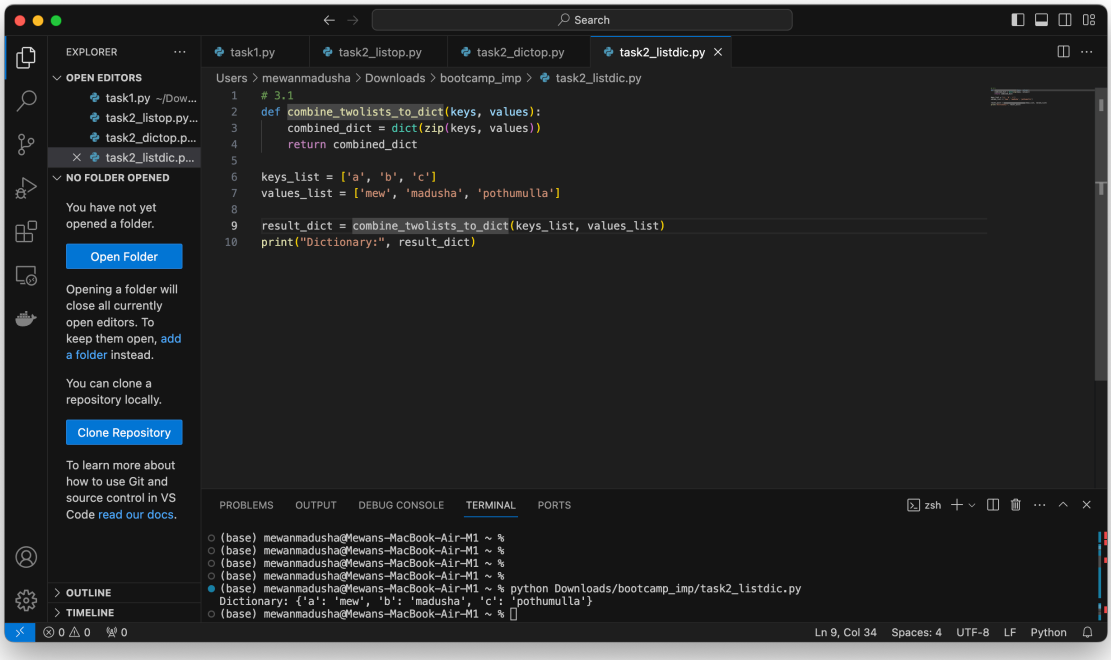
```
def sort_dict_by_values(dictionary):
    sorted_dict = dict(sorted(dictionary.items(), key=lambda item: item[1]))
    return sorted_dict


unsorted_dict = {'b': 3, 'a': 1, 'c': 4}
sorted_dict = sort_dict_by_values(unsorted_dict)
print("Sorted dictionary by values:", sorted_dict)
```



Combining Lists and Dictionaries:

1. Write a Python program that combines two lists into a dictionary where elements from the first list are keys and elements from the second list are values.

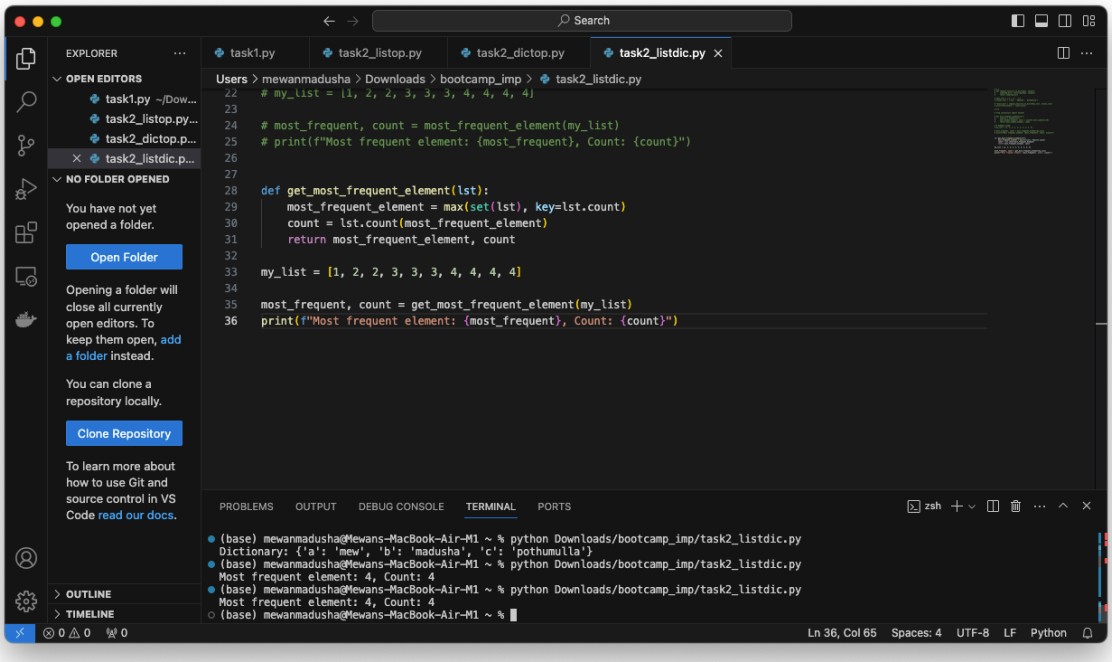
```python
def combine_twolists_to_dict(keys, values):
    combined_dict = dict(zip(keys, values))
    return combined_dict

keys_list = ['a', 'b', 'c']
values_list = ['mew', 'madusha', 'pothumulla']

result_dict = combine_twolists_to_dict(keys_list, values_list)
print("Dictionary:", result_dict)
```



2. Write a program to find the most frequent element in a list and its count.

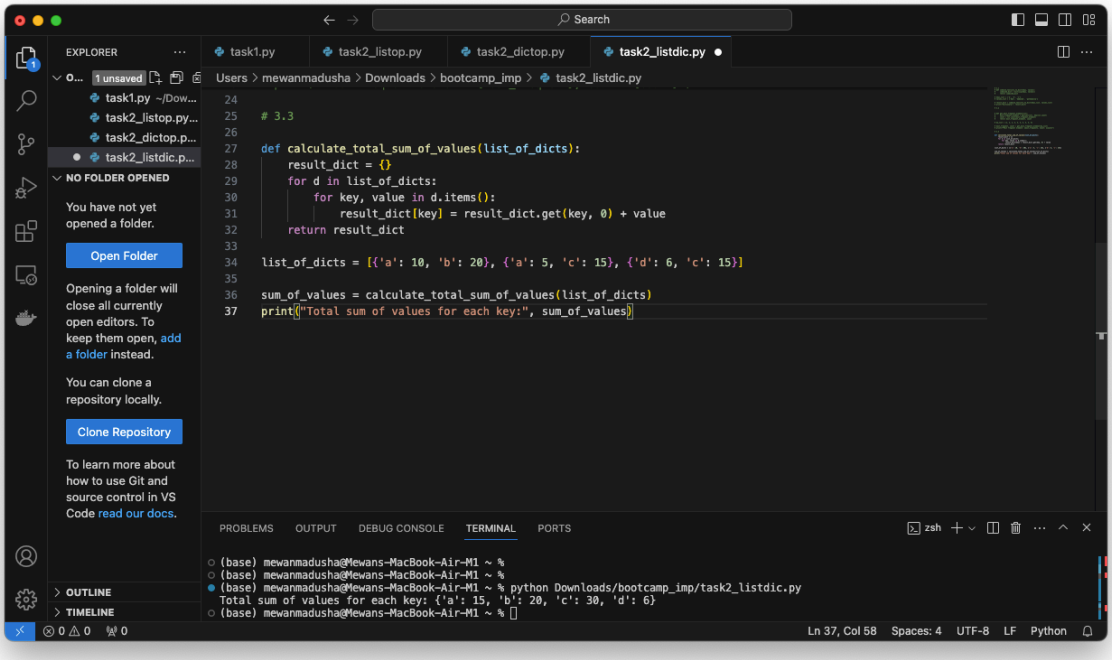
```python
def get_most_frequent_element(lst):
    most_frequent_element = max(set(lst), key=lst.count)
    count = lst.count(most_frequent_element)
    return most_frequent_element, count

my_list = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4]
```

```
most_frequent, count = get_most_frequent_element(my_list)
print(f"Most frequent element: {most_frequent}, Count: {count}")
```



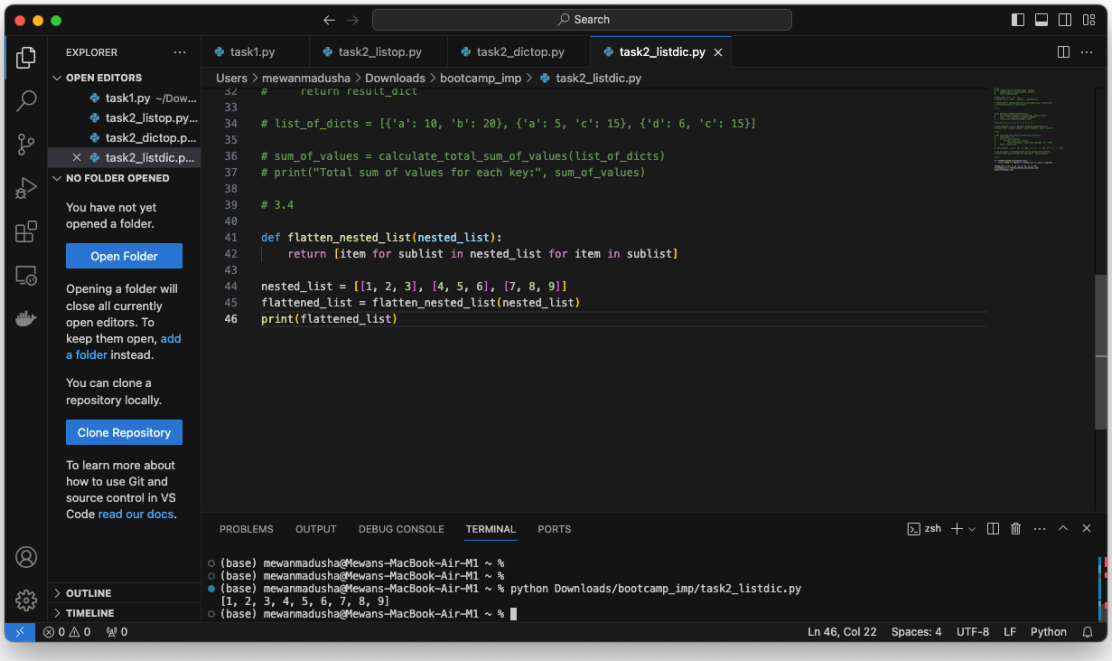
3. Create a program to calculate the total sum of values for each key in a list of dictionaries.

```
def calculate_total_sum_of_values(list_of_dicts):
    result_dict = {}
    for d in list_of_dicts:
        for key, value in d.items():
            result_dict[key] = result_dict.get(key, 0) + value
    return result_dict

list_of_dicts = [{'a': 10, 'b': 20}, {'a': 5, 'c': 15}, {'d': 6, 'c': 15}]

sum_of_values = calculate_total_sum_of_values(list_of_dicts)
print("Total sum of values for each key:", sum_of_values)
```



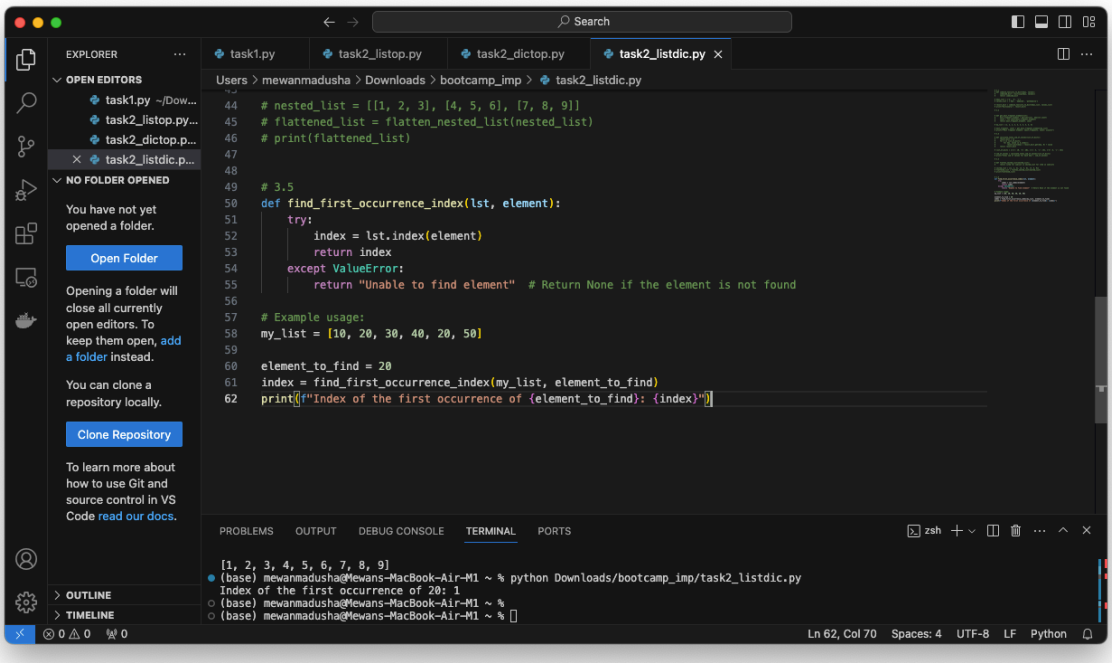
4. Write a Python program to flatten a nested list.

```python
def flatten_nested_list(nested_list):
    return [item for sublist in nested_list for item in sublist]

nested_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
flattened_list = flatten_nested_list(nested_list)
print(flattened_list)
```



5. Write a function to find the index of the first occurrence of a specific element in a list.

```python
def find_first_occurrence_index(lst, element):
    try:
        index = lst.index(element)
        return index
    except ValueError:
        return "Unable to find element"  # Return None if the element is not found

my_list = [10, 20, 30, 40, 20, 50]

element_to_find = 20
index = find_first_occurrence_index(my_list, element_to_find)
print(f"Index of the first occurrence of {element_to_find}: {index}")
```



Extra work(hands-on experience)

List

list.append(elem) -- adds a single element to the end of the list. Common error: does not return the new list, just modifies the original.
list.insert(index, elem) - inserts the element at the given index, shifting elements to the right.
list.extend(list2) adds the elements in list2 to the end of the list. Using + or += on a list is similar to using extend).

```
>>> a="mewan 'madusha'"
>>> print(a)
mewan 'madusha'
>>>
>>>
>>> cities = ['Riga','Colombo','Rome','Tokyo','WDC']
>>> print(cities)
['Riga', 'Colombo', 'Rome', 'Tokyo', 'WDC']
>>> cities.append('Beijing')
>>> print(cities)
['Riga', 'Colombo', 'Rome', 'Tokyo', 'WDC', 'Beijing']
>>> cities.insert(1,'Moscow')
>>> print(cities)
['Riga', 'Moscow', 'Colombo', 'Rome', 'Tokyo', 'WDC', 'Beijing']
>>>
>>> list2 = ['test1','test2']
>>> list2
['test1', 'test2']
>>>
>>> cities.extend(list2)
>>> print(cities)
['Riga', 'Moscow', 'Colombo', 'Rome', 'Tokyo', 'WDC', 'Beijing', 'test1', 'test2']
>>>
```

list.indexelem) -- searches for the given element from the start of the list and returns its index. Throws a ValueError if the element does not appear (use "in" to check without a ValueError).

```
>>> if 'Moscow' in cities:
...     print("Moscow available")
... else:
...     print("Moscow Not available")
...
Moscow available
```

list.remove(elem) -- searches for the first instance of the given element and removes it (throws ValueError if not present)

```
>>> try:
...     cities.remove("WDC")
... except ValueError:
...     print("element is not in the list")
...
>>> cities
['Riga', 'Moscow', 'Colombo', 'Rome', 'Tokyo', 'Beijing', 'test1', 'test2']
>>>
>>> try:
...     cities.remove("WDC")
... except:
...     print("no value")
...
no value
```

list.sort() -- sorts the list in place (does not return it). (The sorted) function shown later is preferred.)
list.reverse) -- reverses the list in place (does not return it)
list.pop(index) -- removes and returns the element at the given index. Returns the rightmost element if index is omitted (roughly the opposite of append()).

```
>>> cities.sort()
>>> cities
['Beijing', 'Colombo', 'Moscow', 'Riga', 'Rome', 'Tokyo', 'test1', 'test2']
>>> cities.reverse()
>>> cities
['test2', 'test1', 'Tokyo', 'Rome', 'Riga', 'Moscow', 'Colombo', 'Beijing']
>>>
>>>
>>> removing_elem = cities.pop(1)
>>> removing_elem
'test1'
>>> cities
['test2', 'Tokyo', 'Rome', 'Riga', 'Moscow', 'Colombo', 'Beijing']
>>>
```

Dictionary

Dictionary

```
>>> dict = {}
>>> dict['name'] = 'mewan'
>>> dict['lname'] = 'madusha'
>>>
>>> print(dict)
{'name': 'mewan', 'lname': 'madusha'}
>>> print(dict[name])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'name' is not defined
>>> print(dict['name'])
mewan
>>> dict['name']='pothumulla'
>>>
>>> 'name' in dict
True
>>> 'madusha' in dict
False
>>> dict
{'name': 'pothumulla', 'lname': 'madusha'}
```

```
>>> if 'name' in dict:
...     dict
...
{'name': 'pothumulla', 'lname': 'madusha'}
>>>
>>> try:
...     print(dict.get('name'))
... except:
...     print('no key')
...
pothumulla
```