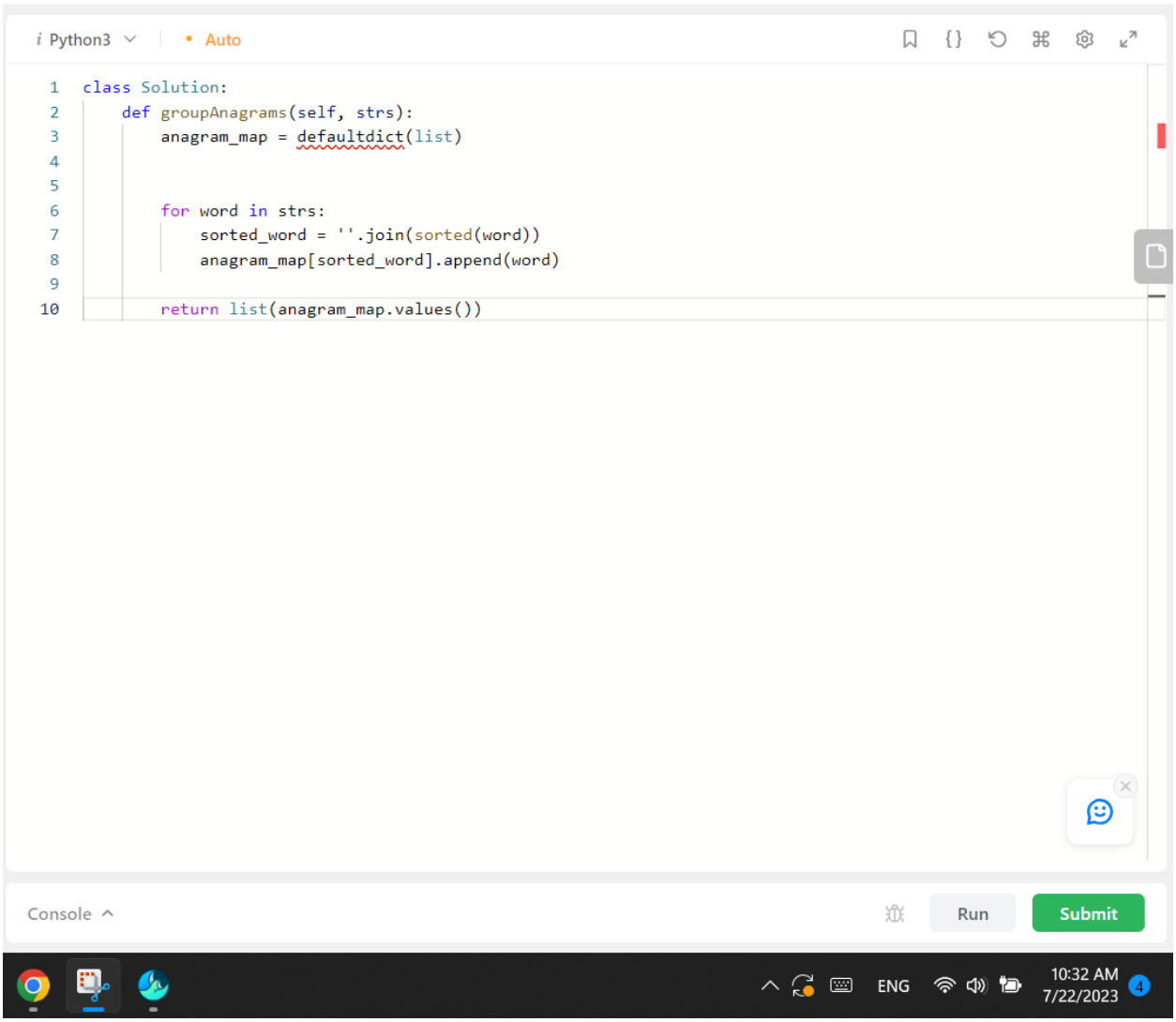
Step 1: Study 49. Group Anagrams (local copy)



Step 2: Following the examples shown on Hash Table to manually solve the problem

You can use this hint to manually solve the Example 1 of the "Description" of the LeetCode question:

Input: strs = ["eat", "tea", "tan", "ate", "nat", "bat"]

Output: [["bat"], ["nat","tan"], ["ate","eat","tea"]]

1) sort each element in strs and get a sorted strs and get only unique element in strs_sorted

strs_sorted = [“aet”, “aet”, “ant”, “aet”, “ant”, “abt”]

str_sorted_unique = [“aet”, “ant”, “abt”]

2) create a dictionary for elements in the strs and group them if their sorted values are the same. The element in strs_sorted_unique serves as key, and the elements in strs serve as values.

{“aet”: [“eat”, “tea”, “ate”],

“ant”: [“tan”, “nat”],

“abt”: [“bat”]}

3) Output the values.

[[“eat”, “tea”, “ate”], [“tan”, “nat”], [“bat”]]

Step 3: Ask ChatGPT: "How to use Hash Table approach to implement a Python code to solve the LeetCode question: '49. Group Anagrams', with the test data: 'Input: strs = ["eat", "tea", "tan", "ate", "nat", "bat"]; Output: [["bat"], ["nat","tan"], ["ate","eat","tea"]]'"

To solve the LeetCode question "49. Group Anagrams" using a Hash Table approach in Python, we can group words that are anagrams together by creating a dictionary where the keys are the sorted characters of each word (anagrams will have the same sorted characters), and the values are lists of words that have the same sorted characters.

Here's a step-by-step implementation of the solution:

python Copy code

```python
def group_anagrams(strs):
    # Initialize an empty dictionary to store the groups of anagrams
    anagram_groups = {}

    # Iterate through each word in the input list
    for word in strs:
        # Sort the characters of the word to get a unique key for anagrams
        sorted_word = "".join(sorted(word))

        # If the sorted_word is not already in the dictionary, add it with a
        if sorted_word not in anagram_groups:
            anagram_groups[sorted_word] = []
```

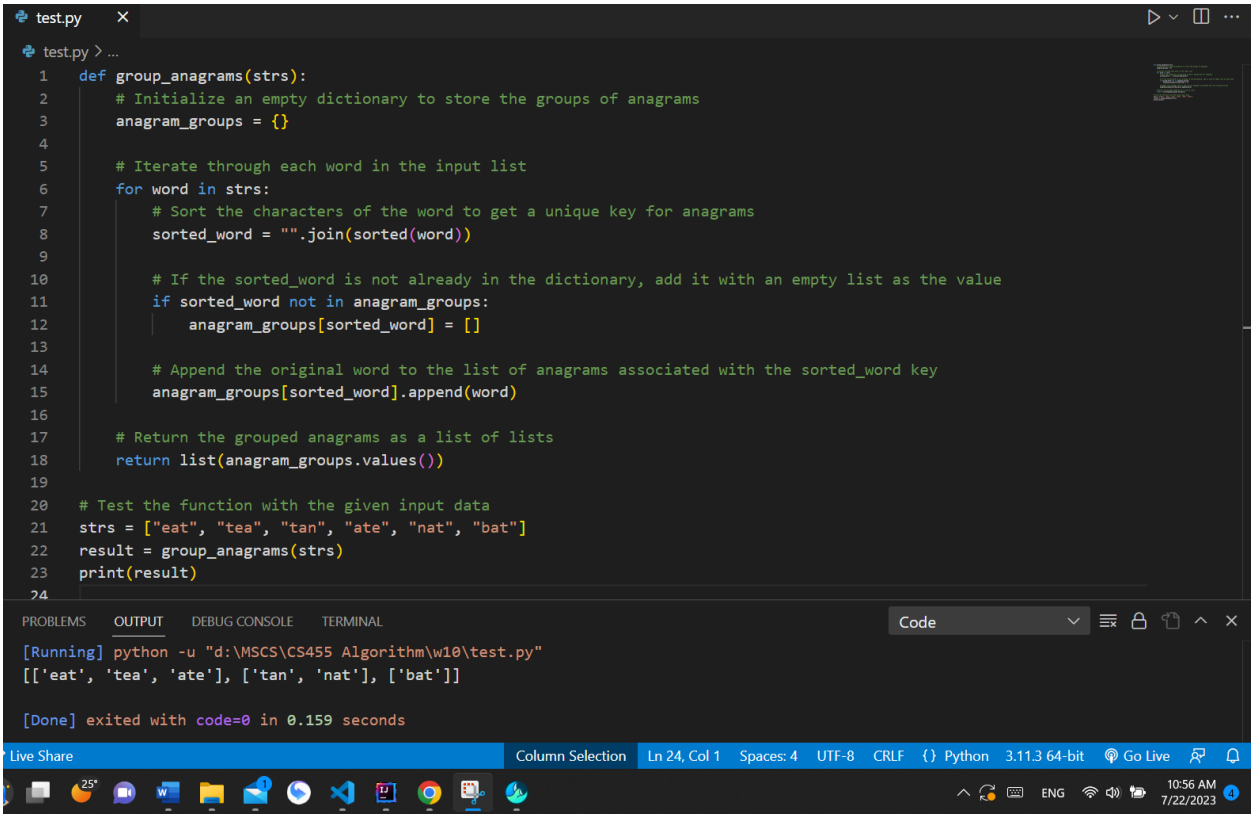
```
        # Append the original word to the list of anagrams associated with t
        anagram_groups[sorted_word].append(word)

    # Return the grouped anagrams as a list of lists
    return list(anagram_groups.values())

# Test the function with the given input data
strs = ["eat", "tea", "tan", "ate", "nat", "bat"]
result = group_anagrams(strs)
print(result)
```

Output:

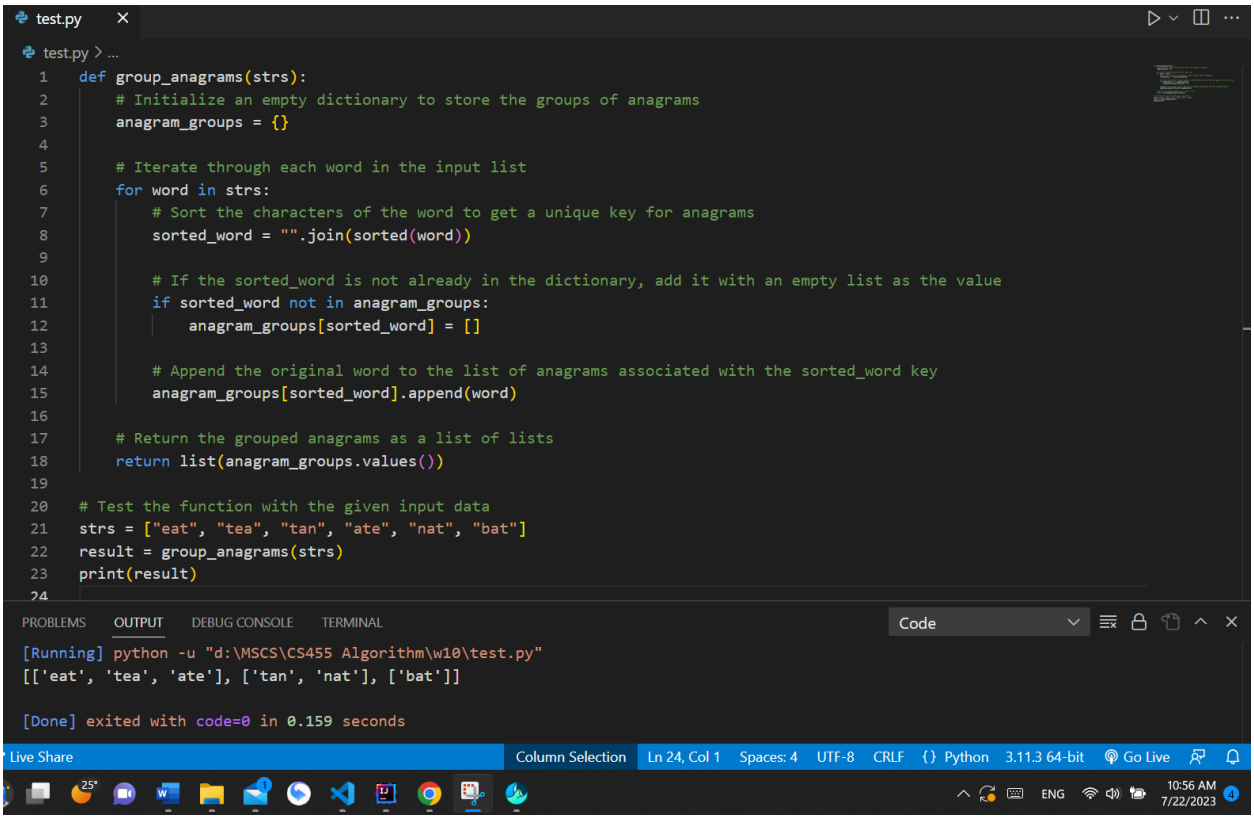
```
css                                                    Copy code

[['eat', 'tea', 'ate'], ['tan', 'nat'], ['bat']]
```

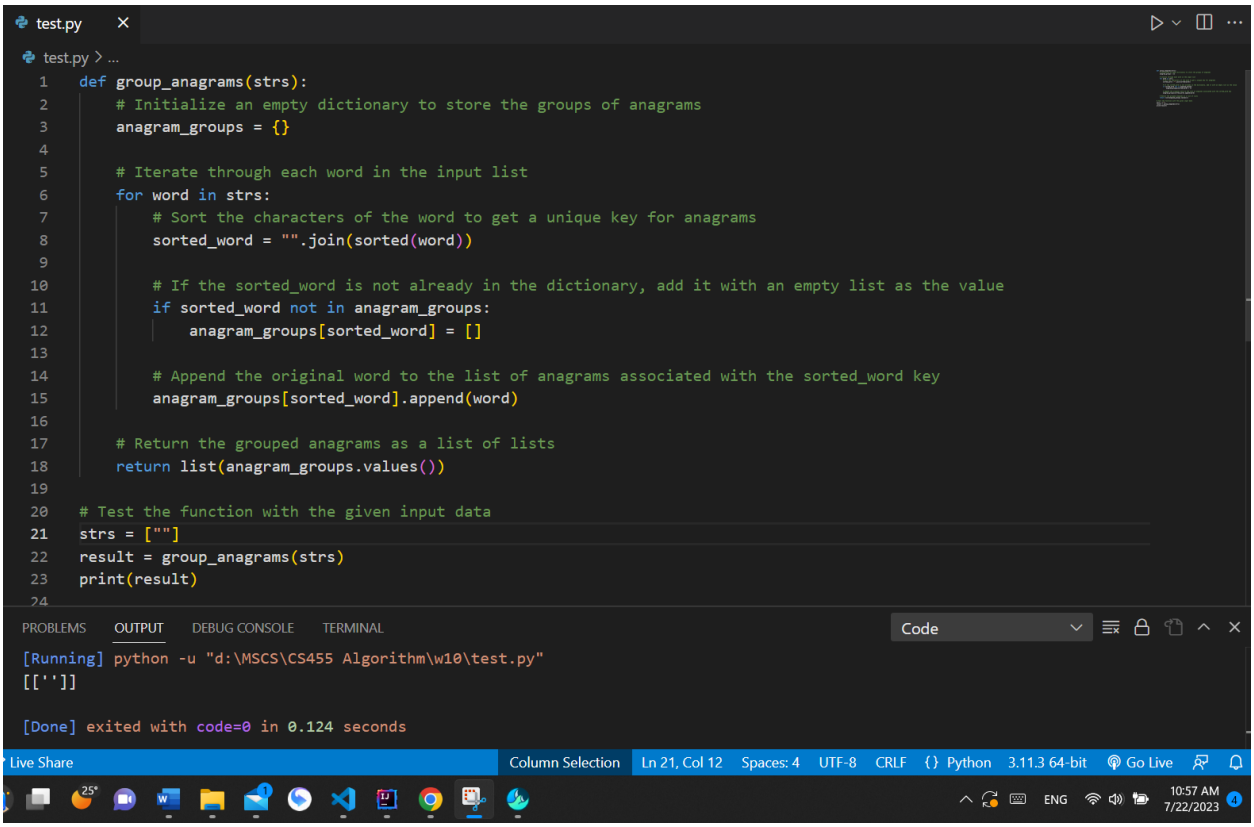
The function `group_anagrams()` takes the list of words `strs` as input and returns a list of lists, where each inner list represents a group of anagrams. In the output, you can see that "eat", "tea", and "ate" are anagrams, "tan" and "nat" are anagrams, and "bat" is not an anagram of any other word in the list. The order of the groups may vary as it depends on the internal hashing order of the keys in the dictionary.

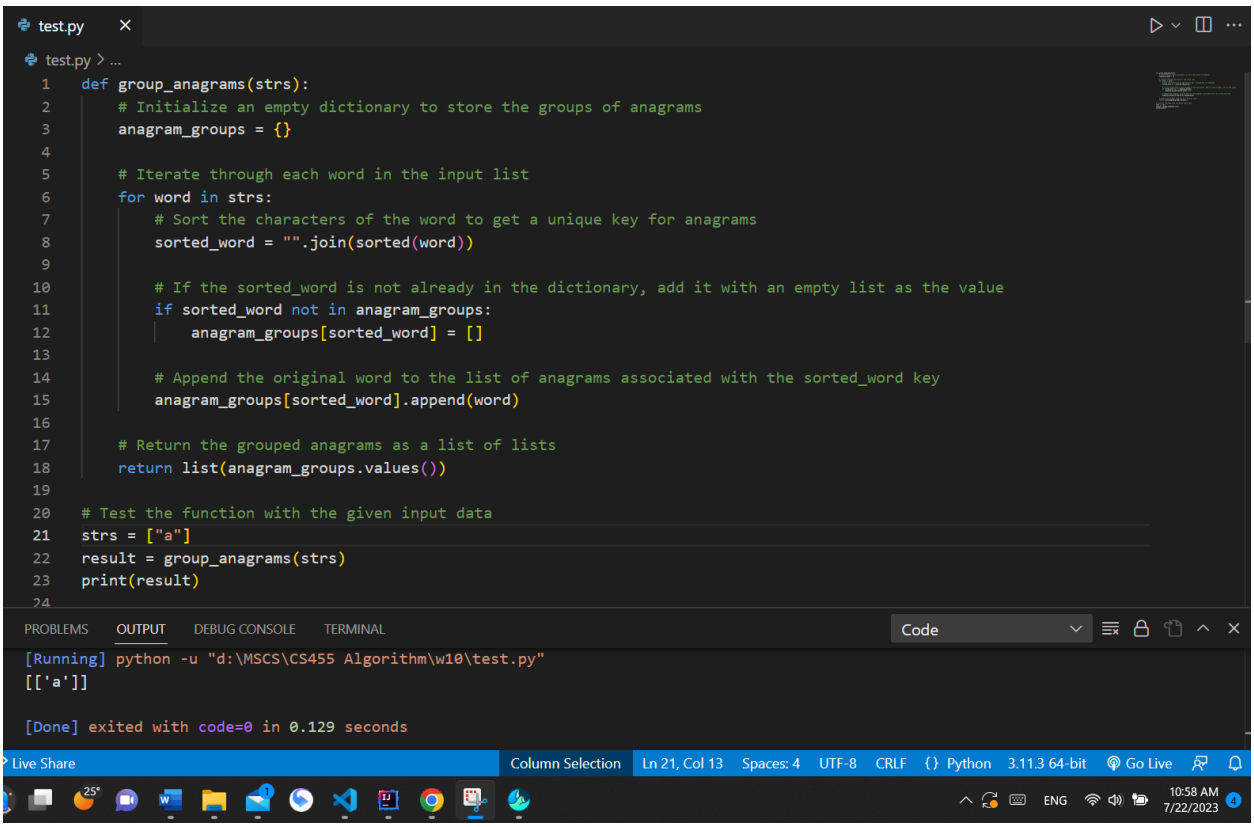
Step 4: Implement the Python code created by ChatGPT



Step 5: Test the Python code with all the test cases provided by 49. Group Anagrams (local copy)




test.py
test.py > ...
1 def group_anagrams(strs):
2 # Initialize an empty dictionary to store the groups of anagrams
3 anagram_groups = {}
4
5 # Iterate through each word in the input list
6 for word in strs:
7 # Sort the characters of the word to get a unique key for anagrams
8 sorted_word = "".join(sorted(word))
9
10 # If the sorted_word is not already in the dictionary, add it with an empty list as the value
11 if sorted_word not in anagram_groups:
12 anagram_groups[sorted_word] = []
13
14 # Append the original word to the list of anagrams associated with the sorted_word key
15 anagram_groups[sorted_word].append(word)
16
17 # Return the grouped anagrams as a list of lists
18 return list(anagram_groups.values())
19
20 # Test the function with the given input data
21 strs = [""]
22 result = group_anagrams(strs)
23 print(result)
24
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Code
[Running] python -u "d:\MSCS\CS455 Algorithm\w10\test.py"
[['']]
[Done] exited with code=0 in 0.124 seconds
Live Share
Column Selection
Ln 21, Col 12
Spaces: 4
UTF-8
CRLF
{} Python
3.11.3 64-bit
Go Live
ENG
10:57 AM
7/22/2023


test.py
test.py > ...
1 def group_anagrams(strs):
2 # Initialize an empty dictionary to store the groups of anagrams
3 anagram_groups = {}
4
5 # Iterate through each word in the input list
6 for word in strs:
7 # Sort the characters of the word to get a unique key for anagrams
8 sorted_word = "".join(sorted(word))
9
10 # If the sorted_word is not already in the dictionary, add it with an empty list as the value
11 if sorted_word not in anagram_groups:
12 anagram_groups[sorted_word] = []
13
14 # Append the original word to the list of anagrams associated with the sorted_word key
15 anagram_groups[sorted_word].append(word)
16
17 # Return the grouped anagrams as a list of lists
18 return list(anagram_groups.values())
19
20 # Test the function with the given input data
21 strs = ["a"]
22 result = group_anagrams(strs)
23 print(result)
24
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Code
[Running] python -u "d:\MSCS\CS455 Algorithm\w10\test.py"
[['a']]
[Done] exited with code=0 in 0.129 seconds
Live Share
Column Selection
Ln 21, Col 13
Spaces: 4
UTF-8
CRLF
{} Python
3.11.3 64-bit
Go Live
ENG
10:58 AM
7/22/2023