

A.YUVAN CHARAN

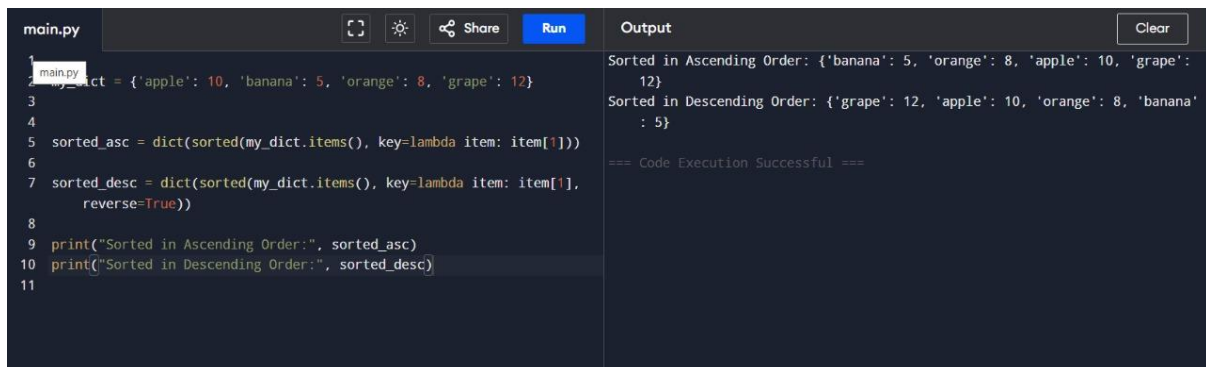
192424383

CSA0815

PYTHON PROGRAMMING

SLOT B

1. Write a Python script to sort (ascending and descending) a dictionary by value.



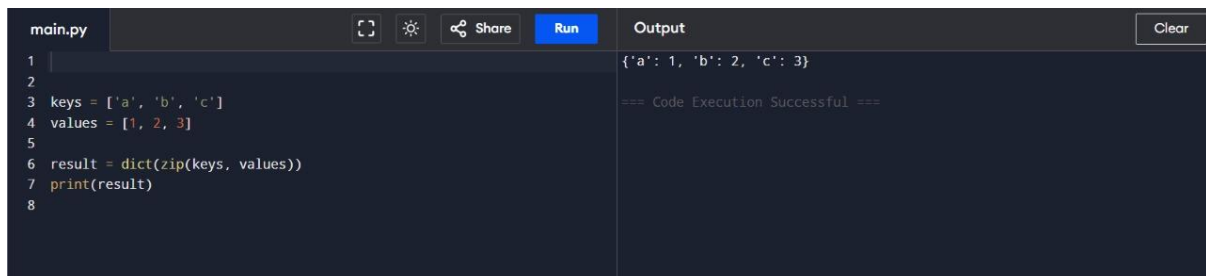
The screenshot shows a Python IDE with a file named 'main.py'. The code defines a dictionary 'my_dict' with fruit names as keys and their counts as values. It then uses the 'sorted' function with a lambda key to sort the items by value in ascending order, and another 'sorted' function with 'reverse=True' for descending order. The results are printed. The output window shows the sorted dictionaries and a success message.

```
1 my_dict = {'apple': 10, 'banana': 5, 'orange': 8, 'grape': 12}
2
3
4 sorted_asc = dict(sorted(my_dict.items(), key=lambda item: item[1]))
5
6 sorted_desc = dict(sorted(my_dict.items(), key=lambda item: item[1],
7                           reverse=True))
8
9 print("Sorted in Ascending Order:", sorted_asc)
10 print("Sorted in Descending Order:", sorted_desc)
11
```

Output

```
Sorted in Ascending Order: {'banana': 5, 'orange': 8, 'apple': 10, 'grape': 12}
Sorted in Descending Order: {'grape': 12, 'apple': 10, 'orange': 8, 'banana': 5}
=== Code Execution Successful ===
```

2. Write a Python program to map two lists into a dictionary.



The screenshot shows a Python IDE with a file named 'main.py'. The code creates two lists, 'keys' and 'values', and uses the 'zip' function to pair them. A dictionary is then created from the zipped pairs and printed. The output window shows the resulting dictionary and a success message.

```
1
2
3 keys = ['a', 'b', 'c']
4 values = [1, 2, 3]
5
6 result = dict(zip(keys, values))
7 print(result)
8
```

Output

```
{'a': 1, 'b': 2, 'c': 3}
=== Code Execution Successful ===
```

3. Write a Python program to combine two dictionary adding values for common keys.

d1 = {'a': 100, 'b': 200, 'c': 300}

d2 = {'a': 300, 'b': 200, 'd': 400}

Sample output: Counter({'a': 400, 'b': 400, 'd': 400, 'c': 300})

```
main.py
1
2
3 def combine_dicts(dict1, dict2):
4     combined_dict = dict1.copy()
5     for key, value in dict2.items():
6         if key in combined_dict:
7             combined_dict[key] += value
8         else:
9             combined_dict[key] = value
10    return combined_dict
11
12 dict_a = {'a': 1, 'b': 2, 'c': 3}
13 dict_b = {'b': 3, 'c': 4, 'd': 5}
14
15 result = combine_dicts(dict_a, dict_b)
16 print(result)
17
```

Output

```
{'a': 1, 'b': 5, 'c': 7, 'd': 5}
=== Code Execution Successful ===
```

4. Using a list comprehension, create a new list that contains only the even numbers from an existing list of integers.

```
main.py
1 existing_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2 even_numbers = [num for num in existing_list if num % 2 == 0]
3 print(even_numbers)
4
```

Output

```
[2, 4, 6, 8, 10]
=== Code Execution Successful ===
```

5. Using a list comprehension, create a new list that contains the squares of all the numbers in an existing list.

```
main.py
1 existing_list = [1, 2, 3, 4, 5]
2 squares = [num ** 2 for num in existing_list]
3 print(squares)
4
```

Output

```
[1, 4, 9, 16, 25]
=== Code Execution Successful ===
```

6. Using a list comprehension, create a new set that contains the unique words from an existing list of strings.

```
main.py
1 existing_list = ["apple", "banana", "apple", "orange", "banana",
2                 "grape"]
3 unique_words = {word for word in existing_list}
4 print(unique_words)
```

Output

```
{'apple', 'grape', 'orange', 'banana'}
=== Code Execution Successful ===
```

7. Using a list comprehension, create a new dictionary that contains the frequency of each word in an existing list of strings.

```
main.py
1 existing_list = ["apple", "banana", "apple", "orange", "banana",
2   "apple"]
3 word_frequency = {word: existing_list.count(word) for word in set
4   (existing_list)}
5 print(word_frequency)
```

Output

```
{'orange': 1, 'banana': 2, 'apple': 3}
=== Code Execution Successful ===
```

8. Using a list comprehension, create a new list that contains only the elements that are common to two existing lists.

```
main.py
1 list1 = [1, 2, 3, 4, 5]
2 list2 = [4, 5, 6, 7, 8]
3 common_elements = [element for element in list1 if element in list2]
4 print(common_elements)
5
```

Output

```
[4, 5]
=== Code Execution Successful ===
```

9. Using a list comprehension, create a new list that contains the uppercase versions of all the elements in an existing list of strings.

```
main.py
1 existing_list = ["apple", "banana", "cherry", "date"]
2 uppercase_list = [word.upper() for word in existing_list]
3 print(uppercase_list)
4
```

Output

```
['APPLE', 'BANANA', 'CHERRY', 'DATE']
=== Code Execution Successful ===
```

10. Using a list comprehension, create a new tuple that contains the elements of an existing list in reverse order.

```
main.py
1 existing_list = [1, 2, 3, 4, 5]
2 reversed_tuple = tuple([element for element in reversed(existing_list)]
3   )
4 print(reversed_tuple)
```

Output

```
(5, 4, 3, 2, 1)
=== Code Execution Successful ===
```

11. Using a list comprehension, create a new list that contains the elements of an existing list, but with duplicates removed.

main.py	Output
<pre>1 existing_list = [1, 2, 2, 3, 4, 4, 5] 2 unique_list = [] 3 [unique_list.append(item) for item in existing_list if item not in unique_list] 4 print(unique_list) 5</pre>	<pre>[1, 2, 3, 4, 5] === Code Execution Successful ===</pre>

12. Using a list comprehension, create a new dictionary that maps the elements of an existing list to their corresponding indices.

main.py	Output
<pre>1 existing_list = ["apple", "banana", "cherry", "date"] 2 index_map = {element: index for index, element in enumerate (existing_list)} 3 print(index_map) 4</pre>	<pre>{'apple': 0, 'banana': 1, 'cherry': 2, 'date': 3} === Code Execution Successful ===</pre>

13. Using a list comprehension, create a new list that contains the Cartesian product of two existing lists.

main.py	Output
<pre>1 list1 = [1, 2, 3] 2 list2 = ['a', 'b', 'c'] 3 cartesian_product = [(x, y) for x in list1 for y in list2] 4 print(cartesian_product) 5</pre>	<pre>[(1, 'a'), (1, 'b'), (1, 'c'), (2, 'a'), (2, 'b'), (2, 'c'), (3, 'a'), (3, 'b'), (3, 'c')] === Code Execution Successful ===</pre>

14. How can you use list comprehension to add two matrices:

main.py	Output
<pre>1 A = [2 [1, 2, 3], 3 [4, 5, 6], 4 [7, 8, 9] 5] 6 7 B = [8 [9, 8, 7], 9 [6, 5, 4], 10 [3, 2, 1] 11] 12 13 result = [[A[i][j] + B[i][j] for j in range(len(A[0]))] for i in range (len(A))] 14 15 print(result) 16</pre>	<pre>[[10, 10, 10], [10, 10, 10], [10, 10, 10]] === Code Execution Successful ===</pre>

