

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

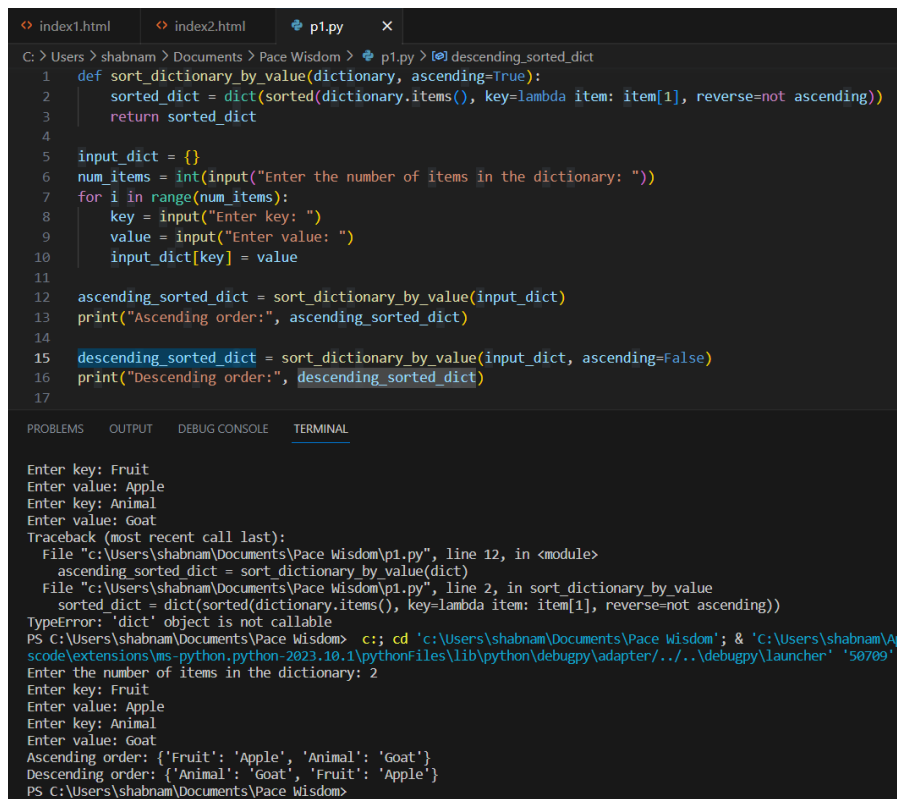
ANS:

```
def sort_dictionary_by_value(dictionary, ascending=True):
    sorted_dict = dict(sorted(dictionary.items(), key=lambda item: item[1], reverse=not ascending))
    return sorted_dict
```

```
input_dict = {}
num_items = int(input("Enter the number of items in the dictionary: "))
for i in range(num_items):
    key = input("Enter key: ")
    value = input("Enter value: ")
    input_dict[key] = value
```

```
ascending_sorted_dict = sort_dictionary_by_value(input_dict)
print("Ascending order:", ascending_sorted_dict)
```

```
descending_sorted_dict = sort_dictionary_by_value(input_dict, ascending=False)
print("Descending order:", descending_sorted_dict)
```



The screenshot shows a Python IDE with a file named `p1.py` open. The code in the file is as follows:

```
1 def sort_dictionary_by_value(dictionary, ascending=True):
2     sorted_dict = dict(sorted(dictionary.items(), key=lambda item: item[1], reverse=not ascending))
3     return sorted_dict
4
5 input_dict = {}
6 num_items = int(input("Enter the number of items in the dictionary: "))
7 for i in range(num_items):
8     key = input("Enter key: ")
9     value = input("Enter value: ")
10    input_dict[key] = value
11
12 ascending_sorted_dict = sort_dictionary_by_value(input_dict)
13 print("Ascending order:", ascending_sorted_dict)
14
15 descending_sorted_dict = sort_dictionary_by_value(input_dict, ascending=False)
16 print("Descending order:", descending_sorted_dict)
17
```

The IDE's output window shows the following text:

```
Enter key: Fruit
Enter value: Apple
Enter key: Animal
Enter value: Goat
Traceback (most recent call last):
  File "c:\Users\shabnam\Documents\Pace Wisdom\p1.py", line 12, in <module>
    ascending_sorted_dict = sort_dictionary_by_value(dict)
  File "c:\Users\shabnam\Documents\Pace Wisdom\p1.py", line 2, in sort_dictionary_by_value
    sorted_dict = dict(sorted(dictionary.items(), key=lambda item: item[1], reverse=not ascending))
TypeError: 'dict' object is not callable
PS C:\Users\shabnam\Documents\Pace Wisdom> c:: cd 'c:\Users\shabnam\Documents\Pace Wisdom'; & 'C:\Users\shabnam\AppData\Local\Microsoft\WindowsApps\python-2023.10.1\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '50709'
Enter the number of items in the dictionary: 2
Enter key: Fruit
Enter value: Apple
Enter key: Animal
Enter value: Goat
Ascending order: {'Fruit': 'Apple', 'Animal': 'Goat'}
Descending order: {'Animal': 'Goat', 'Fruit': 'Apple'}
PS C:\Users\shabnam\Documents\Pace Wisdom>
```

The error message indicates a `TypeError: 'dict' object is not callable`, which occurs because the `dict` constructor is being used incorrectly in the `sorted_dict` assignment in the `sort_dictionary_by_value` function.

2. Write a Python script to add a key to a dictionary.

Sample Dictionary : {0: 10, 1: 20}

Expected Result : {0: 10, 1: 20, 2: 30}

ANS

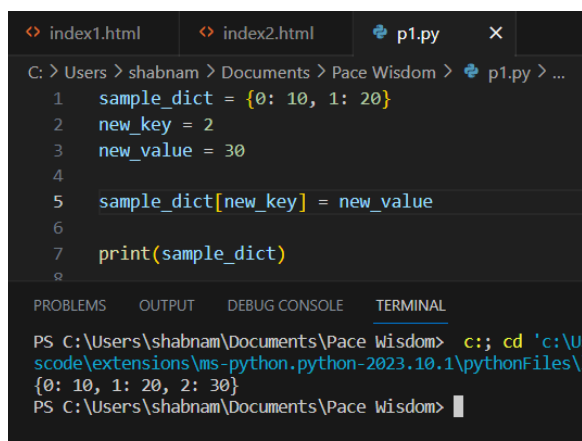
```
sample_dict = {0: 10, 1: 20}
```

```
new_key = 2
```

```
new_value = 30
```

```
sample_dict[new_key] = new_value
```

```
print(sample_dict)
```

A screenshot of a code editor with a dark theme. The top bar shows three tabs: 'index1.html', 'index2.html', and 'p1.py'. The main editor area displays a Python script with the following code:

```
1 sample_dict = {0: 10, 1: 20}
2 new_key = 2
3 new_value = 30
4
5 sample_dict[new_key] = new_value
6
7 print(sample_dict)
```

The bottom panel of the editor is divided into 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL' tabs. The 'TERMINAL' tab is active, showing the command prompt output:

```
PS C:\Users\shabnam\Documents\Pace Wisdom> c;; cd 'c:\Us
scode\extensions\ms-python.python-2023.10.1\pythonFiles\
{0: 10, 1: 20, 2: 30}
PS C:\Users\shabnam\Documents\Pace Wisdom>
```

3. Write a Python script to concatenate following dictionaries to create a new one.

Sample Dictionary :

```
dic1={1:10, 2:20}
```

```
dic2={3:30, 4:40}
```

```
dic3={5:50,6:60}
```

Expected Result : {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

ANS:

```
dic1 = {1: 10, 2: 20}
```

```
dic2 = {3: 30, 4: 40}
```

```
dic3 = {5: 50, 6: 60}
```

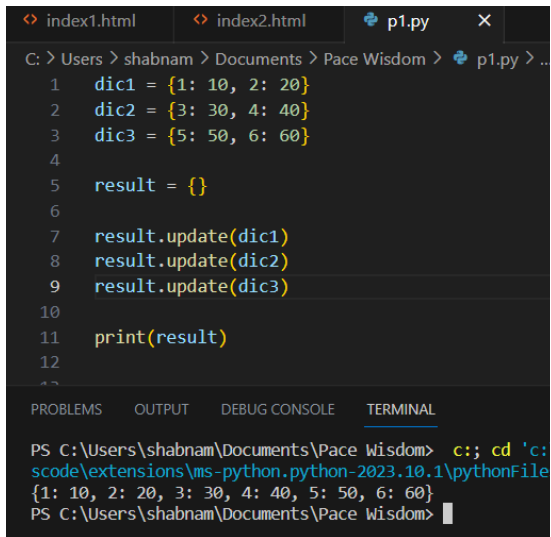
```
result = {}
```

```
result.update(dic1)
```

```
result.update(dic2)
```

```
result.update(dic3)
```

```
print(result)
```



```
index1.html index2.html p1.py x
C: > Users > shabnam > Documents > Pace Wisdom > p1.py > ..
1  dic1 = {1: 10, 2: 20}
2  dic2 = {3: 30, 4: 40}
3  dic3 = {5: 50, 6: 60}
4
5  result = {}
6
7  result.update(dic1)
8  result.update(dic2)
9  result.update(dic3)
10
11 print(result)
12
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\shabnam\Documents\Pace Wisdom> c::; cd 'c:
scode\extensions\ms-python.python-2023.10.1\pythonFile
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
PS C:\Users\shabnam\Documents\Pace Wisdom> |
```

4. Write a Python script to check if a given key already exists in a dictionary.

ANS:

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

```
def check_key_existence(key, dictionary):
```

```
    if key in dictionary:
```

```
        return True
```

```
    else:
```

```
        return False
```

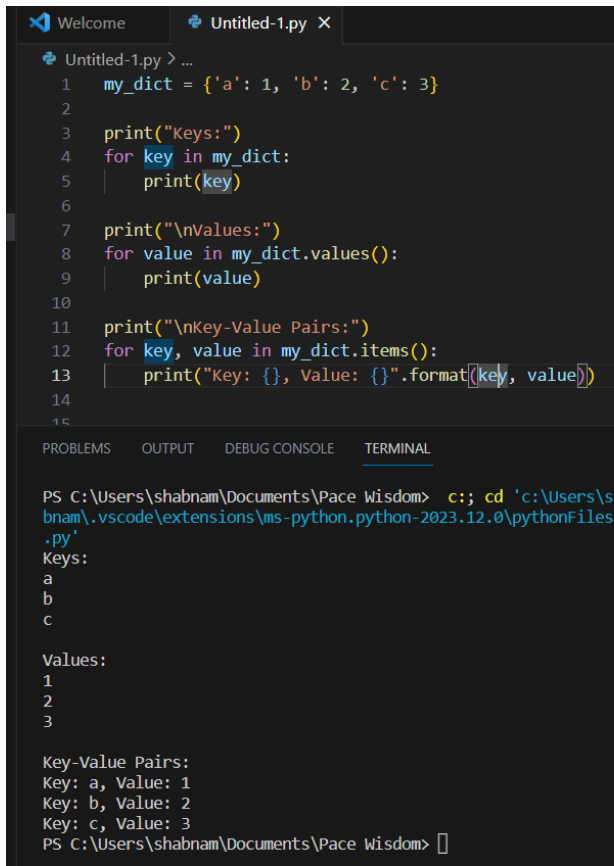
```
key_to_check = input("Enter the key to check: ")
```

```
if check_key_existence(key_to_check, my_dict):
```

```
    print("The key '{}' exists in the dictionary.".format(key_to_check))
```

```
else:
```

```
    print("The key '{}' does not exist in the dictionary.".format(key_to_check))
```

The screenshot shows a VS Code editor with a file named 'Untitled-1.py'. The code defines a dictionary 'my_dict' with keys 'a', 'b', and 'c' and values 1, 2, and 3. It then prints the keys, values, and key-value pairs. The terminal output shows the execution of the script, displaying the keys, values, and key-value pairs as expected.

```
1 my_dict = {'a': 1, 'b': 2, 'c': 3}
2
3 print("Keys:")
4 for key in my_dict:
5     print(key)
6
7 print("\nValues:")
8 for value in my_dict.values():
9     print(value)
10
11 print("\nKey-Value Pairs:")
12 for key, value in my_dict.items():
13     print("Key: {}, Value: {}".format(key, value))
14
15
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\shabnam\Documents\Peace Wisdom> c::; cd 'c:\Users\shabnam\.vscode\extensions\ms-python.python-2023.12.0\pythonFiles\py'

Keys:
a
b
c

Values:
1
2
3

Key-Value Pairs:
Key: a, Value: 1
Key: b, Value: 2
Key: c, Value: 3
PS C:\Users\shabnam\Documents\Peace Wisdom> □

6. Write a Python script to generate and print a dictionary that contains a number (between 1 and n) in the form (x, x*x).

Sample Dictionary (n = 5) :

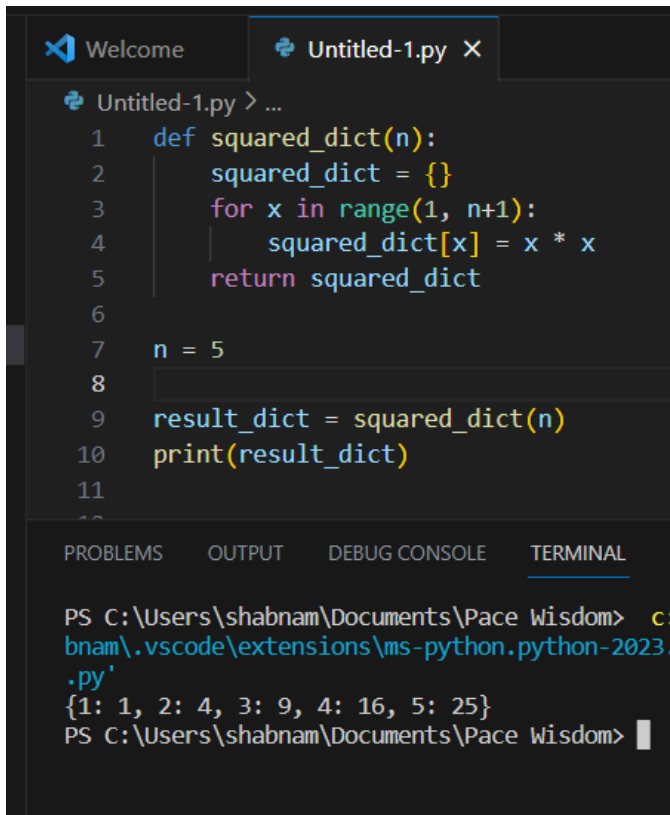
Expected Output : {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

ANS:

```
def squared_dict(n):  
    squared_dict = {}  
    for x in range(1, n+1):  
        squared_dict[x] = x * x  
    return squared_dict
```

n = 5

```
result_dict = squared_dict(n)  
print(result_dict)
```



The screenshot shows a VS Code editor with a file named 'Untitled-1.py'. The code defines a function 'squared_dict(n)' that creates a dictionary with squares of numbers from 1 to n. It then calls this function with n=5 and prints the result. The terminal at the bottom shows the command to run the script and the output: '{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}'.

```
def squared_dict(n):
    squared_dict = {}
    for x in range(1, n+1):
        squared_dict[x] = x * x
    return squared_dict

n = 5

result_dict = squared_dict(n)
print(result_dict)
```

```
PS C:\Users\shabnam\Documents\Pace Wisdom> c:\
shabnam\.vscode\extensions\ms-python.python-2023.
.py'
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
PS C:\Users\shabnam\Documents\Pace Wisdom>
```

7. Write a Python script to merge two Python dictionaries.

ANS:

```
def merge_dicts(dict1, dict2):
    result_dict = dict1.copy()
    result_dict.update(dict2)
    return result_dict
```

```
dict1_input = input("Enter the first dictionary in the format {key1: value1, key2: value2, ...}: ")
dict1 = eval(dict1_input)
```

```
dict2_input = input("Enter the second dictionary in the format {key1: value1, key2: value2, ...}: ")
dict2 = eval(dict2_input)
```

```
merged_dict = merge_dicts(dict1, dict2)
```

```
print("Merged Dictionary:")
print(merged_dict)
```

```
Welcome  Untitled-1.py X
Untitled-1.py > ...
1  def merge_dicts(dict1, dict2):
2      result_dict = dict1.copy()
3      result_dict.update(dict2)
4      return result_dict
5
6  dict1_input = input("Enter the first dictionary in the format {key1: value1, key2: value2, ...}: ")
7  dict1 = eval(dict1_input)
8
9  dict2_input = input("Enter the second dictionary in the format {key1: value1, key2: value2, ...}: ")
10 dict2 = eval(dict2_input)
11
12 merged_dict = merge_dicts(dict1, dict2)
13
14 print("Merged Dictionary:")
15 print(merged_dict)
16

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS C:\Users\shabnam\Documents\Pace Wisdom> c:: cd 'c:\Users\shabnam\Documents\Pace Wisdom'; & 'C:\Users\shabnam\AppData\Local\Microsoft\Windows\apps\python\python-2023.12.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '58952' '-c' 'c:\Users\shabnam\Documents\Pace Wisdom\Untitled-1.py'
Enter the first dictionary in the format {key1: value1, key2: value2, ...}: {1: 10, 2: 20, 3: 30}
Enter the second dictionary in the format {key1: value1, key2: value2, ...}: {'a': 'apple', 'b': 'banana', 'c': 'cherry'}
Merged Dictionary:
{1: 10, 2: 20, 3: 30, 'a': 'apple', 'b': 'banana', 'c': 'cherry'}
PS C:\Users\shabnam\Documents\Pace Wisdom>
```

8. Write a Python program to sum all the items in a dictionary.

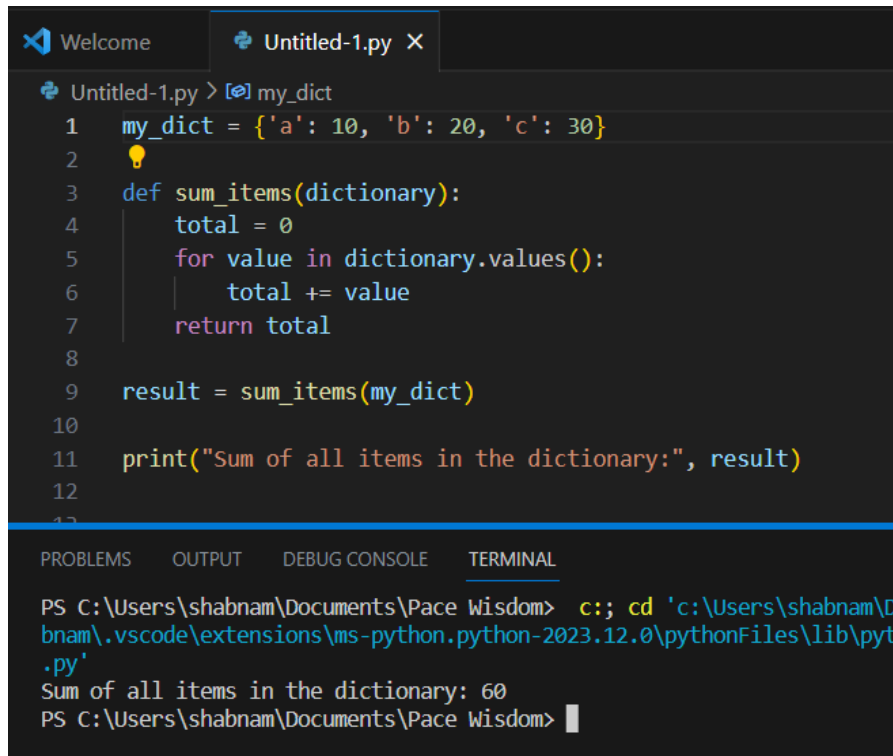
ANS:

```
my_dict = {'a': 10, 'b': 20, 'c': 30}
```

```
def sum_items(dictionary):
    total = 0
    for value in dictionary.values():
        total += value
    return total
```

```
result = sum_items(my_dict)
```

```
print("Sum of all items in the dictionary:", result)
```



```
Untitled-1.py > [🔍] my_dict
1 my_dict = {'a': 10, 'b': 20, 'c': 30}
2
3 def sum_items(dictionary):
4     total = 0
5     for value in dictionary.values():
6         total += value
7     return total
8
9 result = sum_items(my_dict)
10
11 print("Sum of all items in the dictionary:", result)
12
13
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\shabnam\Documents\Pace Wisdom> c::; cd 'c:\Users\shabnam\Documents\Pace Wisdom\.vscode\extensions\ms-python.python-2023.12.0\pythonFiles\lib\python\python.exe'
Sum of all items in the dictionary: 60
PS C:\Users\shabnam\Documents\Pace Wisdom>
```

9. Write a Python program to multiply all the items in a dictionary.

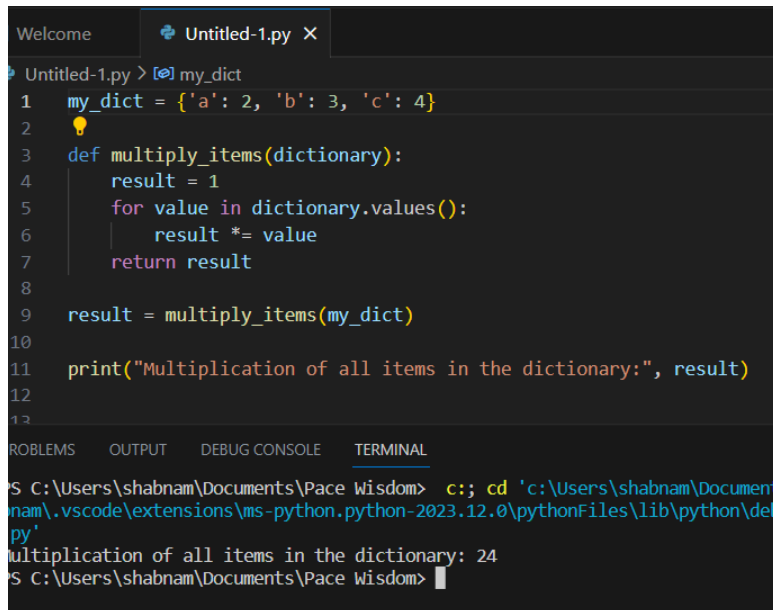
ANS:

```
my_dict = {'a': 2, 'b': 3, 'c': 4}
```

```
def multiply_items(dictionary):
    result = 1
    for value in dictionary.values():
        result *= value
    return result
```

```
result = multiply_items(my_dict)
```

```
print("Multiplication of all items in the dictionary:", result)
```

The screenshot shows a VS Code editor with a file named 'Untitled-1.py'. The code defines a dictionary 'my_dict' with values 2, 3, and 4 for keys 'a', 'b', and 'c' respectively. A function 'multiply_items' is defined to calculate the product of all values in a dictionary. The script then calls this function on 'my_dict' and prints the result. The terminal at the bottom shows the command to run the script, which outputs 'Multiplication of all items in the dictionary: 24'.

```
1 my_dict = {'a': 2, 'b': 3, 'c': 4}
2
3 def multiply_items(dictionary):
4     result = 1
5     for value in dictionary.values():
6         result *= value
7     return result
8
9 result = multiply_items(my_dict)
10
11 print("Multiplication of all items in the dictionary:", result)
12
```

Terminal output:

```
C:\Users\shabnam\Documents\Pace Wisdom> c:: cd 'c:\Users\shabnam\Documents\Pace Wisdom\vscode\extensions\ms-python.python-2023.12.0\pythonFiles\lib\python\debugpy'
Multiplication of all items in the dictionary: 24
C:\Users\shabnam\Documents\Pace Wisdom>
```

10. Write a Python program to remove a key from a dictionary.

ANS:

```
dict1_input = input("Enter the dictionary in the format {key1: value1, key2: value2, ...}: ")
```

```
dict1 = eval(dict1_input)
```

```
key_to_remove = input("Enter the key to remove from the dictionary: ")
```

```
if key_to_remove in dict1:
```

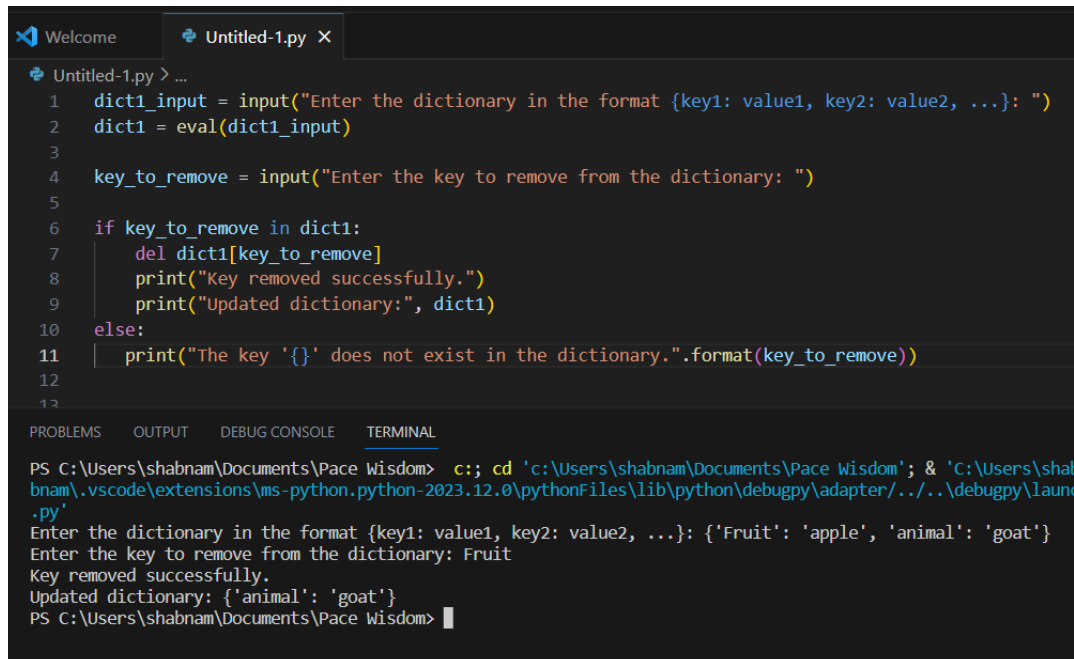
```
    del dict1[key_to_remove]
```

```
    print("Key removed successfully.")
```

```
    print("Updated dictionary:", dict1)
```

```
else:
```

```
    print("The key '{}' does not exist in the dictionary.".format(key_to_remove))
```



```
1 dict1_input = input("Enter the dictionary in the format {key1: value1, key2: value2, ...}: ")
2 dict1 = eval(dict1_input)
3
4 key_to_remove = input("Enter the key to remove from the dictionary: ")
5
6 if key_to_remove in dict1:
7     del dict1[key_to_remove]
8     print("Key removed successfully.")
9     print("Updated dictionary:", dict1)
10 else:
11     print("The key '{}' does not exist in the dictionary.".format(key_to_remove))
12
13
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\shabnam\Documents\Peace Wisdom> c:: cd 'c:\Users\shabnam\Documents\Peace Wisdom'; & 'C:\Users\shabnam\.vscode\extensions\ms-python.python-2023.12.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher.py'
Enter the dictionary in the format {key1: value1, key2: value2, ...}: {'Fruit': 'apple', 'animal': 'goat'}
Enter the key to remove from the dictionary: Fruit
Key removed successfully.
Updated dictionary: {'animal': 'goat'}
PS C:\Users\shabnam\Documents\Peace Wisdom>
```

11. Write a Python program to sort a dictionary by key.

ANS:

```
def sort_dict_by_key(dictionary):
```

```
    sorted_dict = dict(sorted(dictionary.items(), key=lambda item: item[0]))
```

```
    return sorted_dict
```

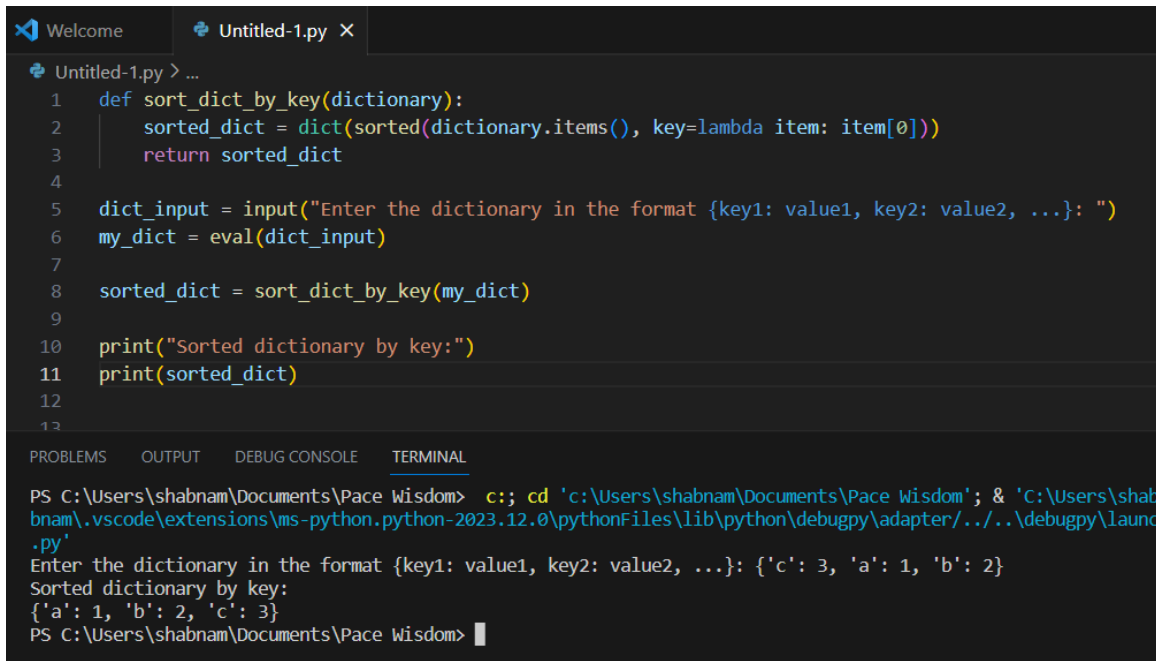
```
dict_input = input("Enter the dictionary in the format {key1: value1, key2: value2, ...}: ")
```

```
my_dict = eval(dict_input)
```

```
sorted_dict = sort_dict_by_key(my_dict)
```

```
print("Sorted dictionary by key:")
```

```
print(sorted_dict)
```



The screenshot shows a VS Code editor with a file named 'Untitled-1.py'. The code defines a function 'sort_dict_by_key' that sorts a dictionary by its keys. It then takes user input for a dictionary, evaluates it, sorts it using the defined function, and prints the result. The terminal at the bottom shows the command prompt running the script, entering the dictionary '{c: 3, a: 1, b: 2}', and displaying the sorted output '{a: 1, b: 2, c: 3}'.

```
1 def sort_dict_by_key(dictionary):
2     sorted_dict = dict(sorted(dictionary.items(), key=lambda item: item[0]))
3     return sorted_dict
4
5 dict_input = input("Enter the dictionary in the format {key1: value1, key2: value2, ...}: ")
6 my_dict = eval(dict_input)
7
8 sorted_dict = sort_dict_by_key(my_dict)
9
10 print("Sorted dictionary by key:")
11 print(sorted_dict)
12
13
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\shabnam\Documents\Peace Wisdom> c++; cd 'c:\Users\shabnam\Documents\Peace Wisdom'; & 'C:\Users\shabnam\.vscode\extensions\ms-python.python-2023.12.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher.py'

Enter the dictionary in the format {key1: value1, key2: value2, ...}: {'c': 3, 'a': 1, 'b': 2}

Sorted dictionary by key:

{'a': 1, 'b': 2, 'c': 3}

PS C:\Users\shabnam\Documents\Peace Wisdom> █

12. Write a Python program to get the maximum and minimum value in a dictionary.

ANS:

```
def get_max_min_values(dictionary):
```

```
    max_value = max(dictionary.values(), default=None)
```

```
    min_value = min(dictionary.values(), default=None)
```

```
    return max_value, min_value
```

```
dict_input = input("Enter the dictionary in the format {key1: value1, key2: value2, ...}: ")
```

```
my_dict = eval(dict_input)
```

```
max_value, min_value = get_max_min_values(my_dict)
```

```
print("Maximum value in the dictionary:", max_value)
```

```
print("Minimum value in the dictionary:", min_value)
```

```
Untitled-1.py > ...
1 def get_max_min_values(dictionary):
2     max_value = max(dictionary.values(), default=None)
3     min_value = min(dictionary.values(), default=None)
4     return max_value, min_value
5
6 dict_input = input("Enter the dictionary in the format {key1: value1, key2: value2, ...}:")
7 my_dict = eval(dict_input)
8
9 max_value, min_value = get_max_min_values(my_dict)
10
11 print("Maximum value in the dictionary:", max_value)
12 print("Minimum value in the dictionary:", min_value)
13
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\shabnam\Documents\Pace Wisdom> c::; cd 'c:\Users\shabnam\Documents\Pace Wisdom'; & 'C:\Users\shabnam\Documents\Pace Wisdom\.vscode\extensions\ms-python.python-2023.12.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher'
Enter the dictionary in the format {key1: value1, key2: value2, ...}: {'a': 10, 'b': 20, 'c': 5, 'd': 30}
Maximum value in the dictionary: 30
Minimum value in the dictionary: 5
PS C:\Users\shabnam\Documents\Pace Wisdom>
```

13. Write a Python program to remove duplicates from Dictionary.

ANS:

```
def remove_duplicates(dictionary):
    unique_dict = {}
    for key, value in dictionary.items():
        if value not in unique_dict.values():
            unique_dict[key] = value
    return unique_dict
```

```
dict_input = input("Enter the dictionary in the format {key1: value1, key2: value2, ...}: ")
my_dict = eval(dict_input)
```

```
unique_dict = remove_duplicates(my_dict)
```

```
print("Dictionary without duplicates:")
print(unique_dict)
```

```
Welcome  Untitled-1.py X

Untitled-1.py > ...
1 def remove_duplicates(dictionary):
2     unique_dict = {}
3     for key, value in dictionary.items():
4         if value not in unique_dict.values():
5             unique_dict[key] = value
6     return unique_dict
7
8 dict_input = input("Enter the dictionary in the format {key1: value1, key2: value2, ...}: ")
9 my_dict = eval(dict_input)
10
11 unique_dict = remove_duplicates(my_dict)
12
13 print("Dictionary without duplicates:")
14 print(unique_dict)
15
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\shabnam\Documents\Pace Wisdom> c:: cd 'c:\Users\shabnam\Documents\Pace Wisdom'; & 'c:\Users\shabnam\AppData\Local\Programs\Microsoft VS Code\python\python-2023.12.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '59413' '--' 'c:\Users\shabnam\Documents\Pace Wisdom\Untitled-1.py'
Enter the dictionary in the format {key1: value1, key2: value2, ...}: {'c': 5, 'a': 10, 'b': 20, 'c': 5, 'd': 30, 'a': 10}
Dictionary without duplicates:
{'c': 5, 'a': 10, 'b': 20, 'd': 30}
PS C:\Users\shabnam\Documents\Pace Wisdom>
```

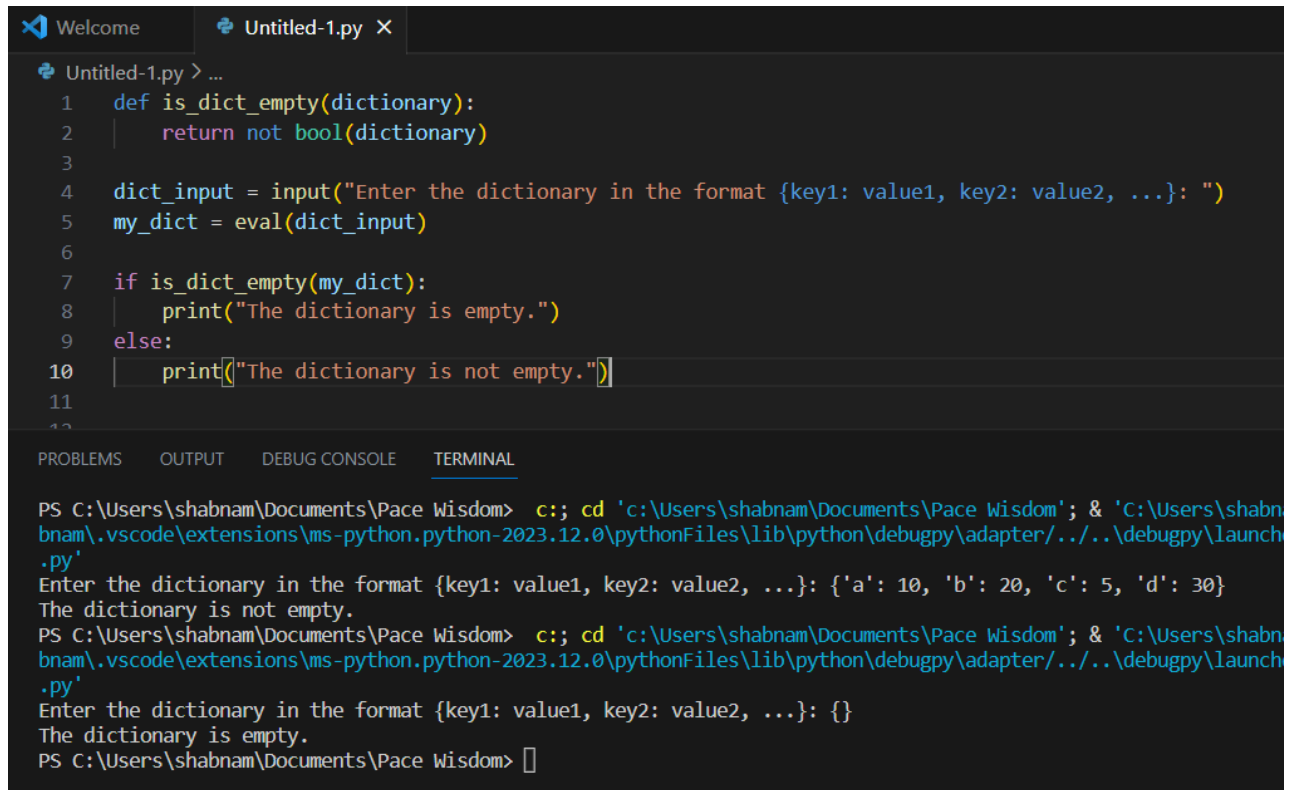
14. Write a Python program to check a dictionary is empty or not.

ANS:

```
def is_dict_empty(dictionary):
    return not bool(dictionary)
```

```
dict_input = input("Enter the dictionary in the format {key1: value1, key2: value2, ...}: ")
my_dict = eval(dict_input)
```

```
if is_dict_empty(my_dict):
    print("The dictionary is empty.")
else:
    print("The dictionary is not empty.")
```



The screenshot shows a VS Code editor with a file named 'Untitled-1.py'. The code defines a function `is_dict_empty` that returns `not bool(dictionary)`. It then prompts the user to enter a dictionary in the format `{key1: value1, key2: value2, ...}`. The program checks if the entered dictionary is empty and prints a message accordingly.

```
1 def is_dict_empty(dictionary):
2     return not bool(dictionary)
3
4 dict_input = input("Enter the dictionary in the format {key1: value1, key2: value2, ...}: ")
5 my_dict = eval(dict_input)
6
7 if is_dict_empty(my_dict):
8     print("The dictionary is empty.")
9 else:
10    print("The dictionary is not empty.")
11
```

The terminal output shows two runs of the program. In the first run, the user enters `{'a': 10, 'b': 20, 'c': 5, 'd': 30}`, and the program outputs "The dictionary is not empty." In the second run, the user enters `{}`, and the program outputs "The dictionary is empty."

15. 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})

16. Write a Python program to find the highest 3 values in a dictionary.

17. Write a Python program to match key values in two dictionaries.

Sample dictionary: `{'key1': 1, 'key2': 3, 'key3': 2}`, `{'key1': 1, 'key2': 2}`

Expected output: key1: 1 is present in both x and y

18. Write a Python program to check if all dictionaries in a list are empty or not.

Sample list : `[{}, {}, {}]`

Return value : True

Sample list : `[{1, 2}, {}, {}]`

Return value : False

19. Write a Python program to remove duplicates from a list of lists.

Sample list : `[[10, 20], [40], [30, 56, 25], [10, 20], [33], [40]]`

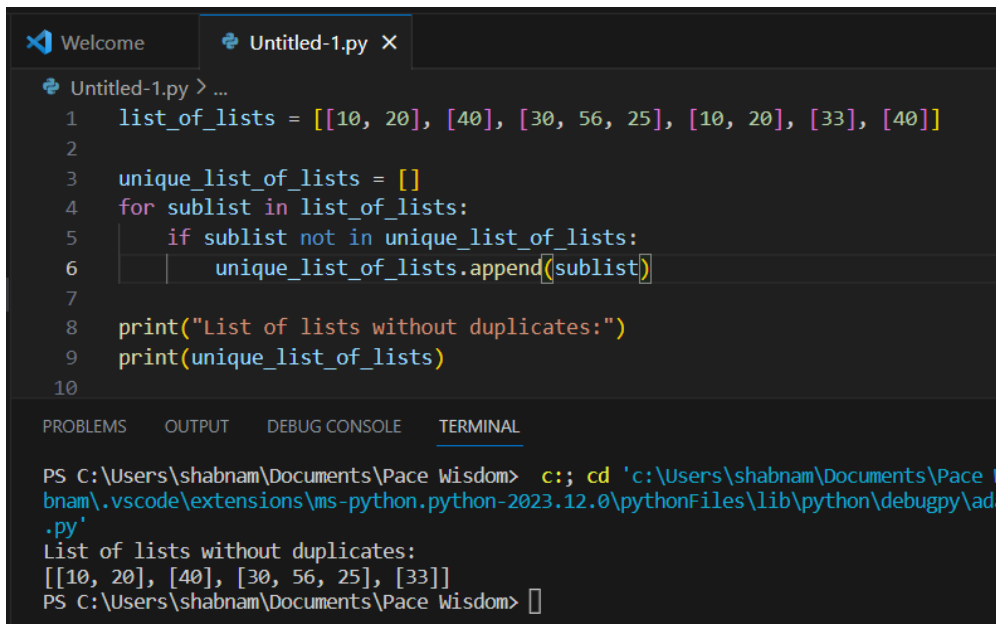
New List : [[10, 20], [30, 56, 25], [33], [40]]

ANS:

```
list_of_lists = [[10, 20], [40], [30, 56, 25], [10, 20], [33], [40]]
```

```
unique_list_of_lists = []
for sublist in list_of_lists:
    if sublist not in unique_list_of_lists:
        unique_list_of_lists.append(sublist)
```

```
print("List of lists without duplicates:")
print(unique_list_of_lists)
```

A screenshot of a Visual Studio Code editor window. The top bar shows 'Welcome' and 'Untitled-1.py'. The editor area contains a Python script with 10 lines of code. The code defines a list of lists, iterates through it, and builds a new list without duplicates. The bottom panel shows the 'TERMINAL' tab with the command prompt output. The command prompt shows the directory path and the execution of the script, followed by the printed output of the unique list of lists.

```
1 list_of_lists = [[10, 20], [40], [30, 56, 25], [10, 20], [33], [40]]
2
3 unique_list_of_lists = []
4 for sublist in list_of_lists:
5     if sublist not in unique_list_of_lists:
6         unique_list_of_lists.append(sublist)
7
8 print("List of lists without duplicates:")
9 print(unique_list_of_lists)
10
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\shabnam\Documents\Pace Wisdom> c:; cd 'c:\Users\shabnam\Documents\Pace Wisdom\vscode\extensions\ms-python.python-2023.12.0\pythonFiles\lib\python\debugpy\ada...
List of lists without duplicates:
[[10, 20], [40], [30, 56, 25], [33]]
PS C:\Users\shabnam\Documents\Pace Wisdom>

20. Write a Python program to extend a list without append.

Sample data: [10, 20, 30]

[40, 50, 60]

Expected output : [40, 50, 60, 10, 20, 30]

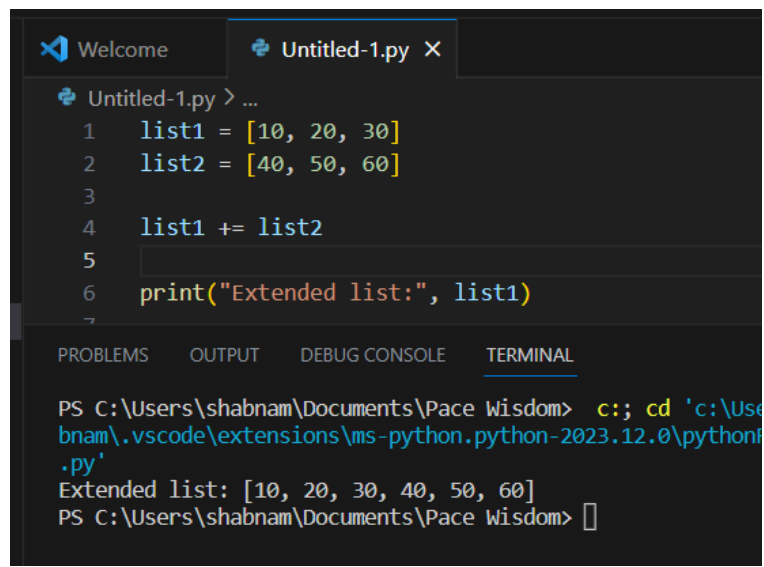
ANS:

```
list1 = [10, 20, 30]
```

```
list2 = [40, 50, 60]
```

```
list1 += list2
```

```
print("Extended list:", list1)
```



The image shows a Visual Studio Code (VS Code) editor window with a dark theme. The top bar displays two tabs: 'Welcome' and 'Untitled-1.py X'. The main editor area shows a Python script in 'Untitled-1.py' with the following code:

```
1 list1 = [10, 20, 30]
2 list2 = [40, 50, 60]
3
4 list1 += list2
5
6 print("Extended list:", list1)
```

Below the editor, the 'TERMINAL' tab is active, showing the command prompt output. The prompt is 'PS C:\Users\shabnam\Documents\Peace Wisdom>'. The command executed is 'c:; cd 'c:\Users\shabnam\.vscode\extensions\ms-python.python-2023.12.0\pythonf.py'', which results in the output 'Extended list: [10, 20, 30, 40, 50, 60]'. The prompt returns to 'PS C:\Users\shabnam\Documents\Peace Wisdom>'.

```
PS C:\Users\shabnam\Documents\Peace Wisdom> c:; cd 'c:\Users\shabnam\.vscode\extensions\ms-python.python-2023.12.0\pythonf.py'
Extended list: [10, 20, 30, 40, 50, 60]
PS C:\Users\shabnam\Documents\Peace Wisdom>
```