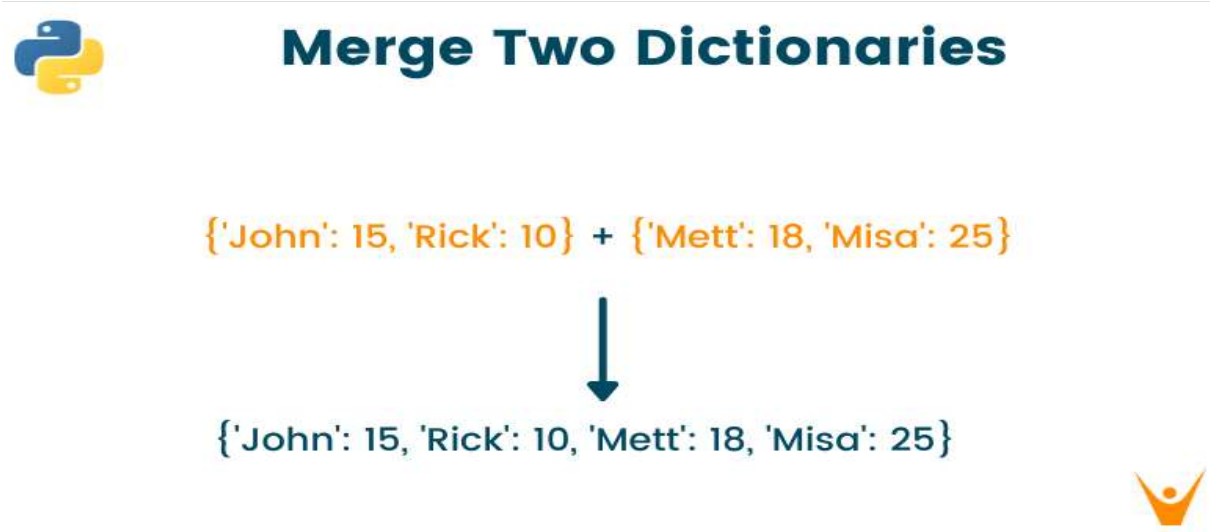
Merge Dictionaries in Python: 8 Standard Methods (with code)

Jan 14, 2023 7 Minutes Read



A dictionary is one of the fundamental and most used data structures in python programming. It is not an uncommon situation where you wish to combine two or more dictionaries. In this article, we will study various ways to merge two dictionaries in Python in different situations along with the code.

But before that, let us have a brief look at dictionaries in python below.

What is Dictionary in Python?

A dictionary is an unordered collection of data elements stored in key-value pairs, unlike any other data structure holding the data as a single value element. You can create a dictionary by placing the key-value pair inside the curly brackets({}) separated by a comma (,).

Moreover, the key and value elements are separated by placing a semi-colon(:) between them. Dictionary in python is ordered, changeable, and does not allow duplicates. That means the dictionary cannot have two items with the same key; hence, dictionary keys are immutable.

Example:

```
sample_dict = {
  "Language_1": "Python",
  "Langauge_2": "C++",
  "Language_3": "Java"
}
print(sample_dict)
```

Output:

```
{'Language_1': 'Python', 'Langauge_2': 'C++', 'Language_3': 'Java'}
```

Check how to create a python dictionary to learn more.

When is the need to merge dictionaries?

Dictionaries are used to store key-value pairs. They can prove exceptionally usef
indexed. For example, if you were to keep a record of all the passwords used on a

How to Merge Dictionaries in Python?

Below are the 8 unique methods by which you can concatenate two dictionaries in python:

1) Using update() method

You can merge two dictionaries in python using the update() method. **Using the update method, the first dictionary is merged with the other one by overwriting it.** Hence, no new dictionary is created, and it returns None. If both dictionaries contain the same key and different values, then the final output will overwrite the value of the latter dictionary.

Example:

```
dict_1 = {'John': 15, 'Rick': 10, 'Misa' : 12 }
dict_2 = {'Bonnie': 18,'Rick': 20,'Matt' : 16 }
dict_1.update(dict_2)
print('Updated dictionary:')
print(dict_1)
```

Output:

```
{'John': 15, 'Rick': 20, 'Misa': 12, 'Bonnie': 18, 'Matt': 16}
```

2) Using merge(|) operator

You can merge two dictionaries using the | operator. It is a very convenient method to merge dictionaries; however, it is only used in the python 3.9 version or more.

Example:

```
def Merge(dict_1, dict_2):
        result = dict_1 | dict_2
        return result

# Driver code
dict_1 = {'John': 15, 'Rick': 10, 'Misa' : 12 }
dict_2 = {'Bonnie': 18,'Rick': 20,'Matt' : 16 }
dict_3 = Merge(dict_1, dict_2)
print(dict_3)
```

Output:

```
{'John': 15, 'Rick': 20, 'Misa': 12, 'Bonnie': 18, 'Matt': 16}
```

3) Using ** operator

The simplest way to concatenate two dictionaries in python is by using the unpacking operator(**). By applying the "**" operator to the dictionary, it expands its content being the collection of key-value pairs.

For example, if you apply "**" to dict_1 as shown below, the output will collect key-value pairs stored inside dict_1.

Example:

```
dict_1 = {'John': 15, 'Rick': 10, 'Misa': 12}
print(dict(**dict_1))
```

Output:

```
{'John': 15, 'Rick': 10, 'Misa': 12}
```

Example:

```
dict_1 = {'John': 15, 'Rick': 10, 'Misa' : 12 }
dict_2 = {'Bonnie': 18,'Rick': 20,'Matt' : 16 }
dict_3 = {**dict_1,**dict_2}
print(dict_3)
```

Output:

```
{'John': 15, 'Rick': 20, 'Misa': 12, 'Bonnie': 18, 'Matt': 16}
```

You can also merge three dictionaries at the same time using the "**" operator, as shown in the below example.

Example:

```
dict_1 = {'John': 15, 'Rick': 10, 'Misa' : 12 }
dict_2 = {'Bonnie': 18,'Rick': 20,'Matt' : 16 }
dict_3 = {'Stefan': 19, 'Riya': 14, 'Lora': 17}
dict_4 = {**dict_1,**dict_2, **dict_3}
print (dict_4)
```

Output:

```
{'John': 15, 'Rick': 20, 'Misa': 12, 'Bonnie': 18, 'Matt': 16, 'Stefan': 19, 'Riya': 14, 'Lora': 17}
```

4) Unpacking the second dictionary

Unpacking the second dictionary using the "**" operator will help you merge two dictionaries and get your final output. However, this method is not highly used and recommended because it will only work if the keys of the second dictionary are compulsorily strings. If any int value is encountered, then it will raise the "TypeError" method.

Example:

```
dict_1={'John': 15, 'Rick': 10, 'Misa' : 12 }
dict_2={'Bonnie': 18,'Rick': 20,'Matt' : 16 }
dict_3=dict(dict_1,**dict_2)
print (dict_3)
```

Output:

```
{'John': 15, 'Rick': 20, 'Misa': 12, 'Bonnie': 18, 'Matt': 16}
```

5) Using collection.ChainMap() method

This is one of the least known methods to merge two dictionaries in python. **Using collection.ChainMap() method, you have to make use of the collection module from the ChainMap library which will help you to group multiple dictionaries in a single view.**

If both dictionaries contain the same key/s, then the value of the first dictionary is fetched in the final output. Note that we will make use of the "import..from.." syntax to import the collection module as shown in the below example:

Example:

```
from collections import ChainMap
dict_1={'John': 15, 'Rick': 10, 'Misa' : 12 }
dict_2={'Bonnie': 18,'Rick': 20,'Matt' : 16 }
dict_3 = ChainMap(dict_1, dict_2)
print(dict_3)
print(dict(dict_3))
```

6) Using itertools.chain()

The iterator returns the element from the first iterable until it's empty and then jumps to the next iterable. Basically, it will handle the consecutive sequence as a single sequence.

As the dictionary is also iterable, we can use the chain() function from itertools class and merge two dictionaries. The return type of this method will be an object, and hence, we can convert the dictionary using the dict() constructor.

Example:

```
import itertools
dict_1={'John': 15, 'Rick': 10, 'Misa': 12}
dict_2={'Bonnie': 18, 'Rick': 20, 'Matt': 16}
dict_3=itertools.chain(dict_1.items(),dict_2.items())
#Returns an iterator object
print (dict_3)
print(dict(dict_3))
```

Output:

```
<itertools.chain object at 0x000001EB8E312520>
{'John': 15, 'Rick': 20, 'Misa': 12, 'Bonnie': 18, 'Matt': 16}
```

7) Using dictionary comprehension

We can combine two dictionaries in python using dictionary comprehension. Here, we also use the for loop to iterate through the dictionary items and merge them to get the final output. If both dictionaries have common keys, then the final output using this method will contain the value of the second dictionary.

Example:

```
dict_1={'John': 15, 'Rick': 10, 'Misa': 12}
dict_2={'Bonnie': 18, 'Rick': 20, 'Matt': 16}
dict_3={k:v for d in (dict_1,dict_2) for k,v in d.items()}
print (dict_3)
```

Output:

```
{'John': 15, 'Rick': 20, 'Misa': 12, 'Bonnie': 18, 'Matt': 16}
```

8) Add values of common keys

In all of the above methods, if both the dictionary contains the different data values of the same keys, the values in the final output were getting overridden. What if you wish to preserve all the value in the final output? For this purpose, you can merge the dictionaries such that it adds the values of the common keys in the list and return the final output.

In this method, we will also make use of for loop to traverse through the keys and values in the dictionary inside the function. Later, we will call the function to get the final output as merged dictionaries as shown below:

Example:

```
dict_1 = {'John': 15, 'Rick': 10, 'Misa': 12}
dict_2 = {'Bonnie': 18, 'Rick': 20, 'Matt': 16}

def mergeDictionary(dict_1, dict_2):
   dict_3 = {**dict_1, **dict_2}
   for key, value in dict_3.items():
       if key in dict_1 and key in dict_2:
               dict_3[key] = [value , dict_1[key]]
   return dict_3
```