# Python - Maps

Python Maps also called ChainMap is a type of data structure to manage multiple dictionaries together as one unit. The combined dictionary contains the key and value pairs in a specific sequence eliminating any duplicate keys. The best use of ChainMap is to search through multiple dictionaries at a time and get the proper key-value pair mapping. We also see that these ChainMaps behave as stack data structure.

## Creating a ChainMap

We create two dictionaries and club them using the ChainMap method from the collections library. Then we print the keys and values of the result of the combination of the dictionaries. If there are duplicate keys, then only the value from the first key is preserved.

### Example

```
import collections

dict1 = {'day1': 'Mon', 'day2': 'Tue'}
dict2 = {'day3': 'Wed', 'day1': 'Thu'}

res = collections.ChainMap(dict1, dict2)

# Creating a single dictionary
print(res.maps,'\n')
```

```
print('Keys = {}'.format(list(res.keys())))
print('Values = {}'.format(list(res.values())))
print()

# Print all the elements from the result
print('elements:')
for key, val in res.items():
   print('{} = {}'.format(key, val))
print()

# Find a specific value in the result
print('day3 in res: {}'.format(('day1' in res)))
print('day4 in res: {}'.format(('day4' in res)))
```

## Output

When the above code is executed, it produces the following result −

```
[{'day1': 'Mon', 'day2': 'Tue'}, {'day1': 'Thu', 'day3': 'Wed'}]

Keys = ['day1', 'day3', 'day2']
Values = ['Mon', 'Wed', 'Tue']

elements:
day1 = Mon
day3 = Wed
day2 = Tue

day3 in res: True
day4 in res: False
```

## Map Reordering

If we change the order the dictionaries while clubbing them in the above example we see that the position of the elements get interchanged as if they are in a continuous chain. This again shows the behavior of Maps as stacks.

### Example

```
import collections

dict1 = {'day1': 'Mon', 'day2': 'Tue'}
dict2 = {'day3': 'Wed', 'day4': 'Thu'}

res1 = collections.ChainMap(dict1, dict2)
print(res1.maps,'\n')

res2 = collections.ChainMap(dict2, dict1)
print(res2.maps,'\n')
```

### Output

When the above code is executed, it produces the following result −

```
[{'day1': 'Mon', 'day2': 'Tue'}, {'day3': 'Wed', 'day4': 'Thu'}]

[{'day3': 'Wed', 'day4': 'Thu'}, {'day1': 'Mon', 'day2': 'Tue'}]
```

## Updating Map

When the element of the dictionary is updated, the result is instantly updated in the result of the ChainMap. In the below example we see that the new updated value reflects in the result

without explicitly applying the ChainMap method again.

## Example

```
import collections

dict1 = {'day1': 'Mon', 'day2': 'Tue'}
dict2 = {'day3': 'Wed', 'day4': 'Thu'}

res = collections.ChainMap(dict1, dict2)
print(res.maps,'\n')

dict2['day4'] = 'Fri'
print(res.maps,'\n')
```

## Output

When the above code is executed, it produces the following result −

```
[{'day1': 'Mon', 'day2': 'Tue'}, {'day3': 'Wed', 'day4': 'Thu'}]

[{'day1': 'Mon', 'day2': 'Tue'}, {'day3': 'Wed', 'day4': 'Fri'}]
```