Python Dictionary Methods

Python Dictionary Methods

Python **dict** class provides many methods that transform or operate on the items (key:value pairs) of Python Dictionary. In addition to these, we have builtin methods that operate on list objects and transform them.

In this tutorial, we will learn about all the available methods of Python Dictionary. Following is the list of methods.

1. dict.clear()
2. dict.copy()
3. dict.get()
4. dict.items()
5. dict.keys()
6. dict.values()
7. dict.fromkeys()
8. dict.pop()
9. dict.popitem()
10. dict.setdefault()
11. dict.update()

dict.clear()

Python Dictionary clear() method removes all the items from the dictionary.

clear() method operates inplace, hence modifies the original dictionary.

In the following program, we will create a dictionary with a few key:value pairs and call clear() method on the dictionary. The items in the dictionary should be removed.

Example Program

```
dictionary = {'a': 54, 'b': 87, 'c': 61}
dictionary.clear()
print(dictionary)
```

Program Output

```

```

More about Python Dictionary clear() method.

dict.copy()

Python Dictionary copy() method returns a shallow copy of the dictionary. Any modifications to the copy of

dictionary does not affect the original dictionary.

In the following program, we will initialize a dictionary with a few key:value pairs and make a copy of this dictionary. After that we shall make some modifications to the copy and print the two dictionaries.

Example Program

```
dictionary1 = {'a': 54, 'b': 87, 'c': 61}
dictionary2 = dictionary1.copy()
dictionary2['b'] = 11
print("dictionary1 :", dictionary1)
print("dictionary2 :", dictionary2)
```

Program Output

```
dictionary1 : {'a': 54, 'b': 87, 'c': 61}
dictionary2 : {'a': 54, 'b': 11, 'c': 61}
```

More about Python Dictionary copy() method.

dict.get(key[, default])

Python Dictionary get(key[, default]) method returns the value for the key in dictionary. We are passing key as argument.

You can also pass a default value as second argument. If there is no key:value pair for the specified key, get() returns the default value. If there is no key:value pair for the given key and no default value is passed as argument, get() returns None.

In the following program, we will take a dictionary and get the value for key 'b'.

Example Program

```
dictionary = {'a': 54, 'b': 87, 'c': 61}
value = dictionary.get('b')
print(value)
```

Program Output

```

```

In the following program, we will call get() method with key not present in the dictionary and default value passed as argument.

Example Program

```
dictionary = {'a': 54, 'b': 87, 'c': 61}
value = dictionary.get('f', 0)
print(value)
```

Program Output

```
0
```

In the following program, we will call get() method with key not present in the dictionary and default value not passed as argument.

Example Program

```
dictionary = {'a': 54, 'b': 87, 'c': 61}
value = dictionary.get('f')
print(value)
```

Program Output

```
None
```

More about Python Dictionary get() method.

dict.items()

Python Dictionary items() method returns the view object for the dictionary items (key:value pairs).

The view object provides a dynamic view on the dictionary's entries, which means that when the dictionary changes, the view reflects these changes.

In the following program, we will take a dictionary and iterate over the dictionary items using the view object dict.items().

Example Program

```
dictionary = {'a': 54, 'b': 87, 'c': 61}
print(dictionary.items())

for key, value in dictionary.items():
    print(key, '-', value)
```

Program Output

```
dict_items([('a', 54), ('b', 87), ('c', 61)])
a - 54
b - 87
c - 61
```

More about Python Dictionary items() method.

dict.keys()

Python Dictionary keys() method returns the view object for the dictionary keys.

The view object returned by keys() method is similar to that of returned by items() method. The only difference between these methods is that items() return key:value pairs while keys() return only keys.

In the following program, we will take a dictionary and iterate over the dictionary keys using the view object dict.keys().

Example Program

```
dictionary = {'a': 54, 'b': 87, 'c': 61}
print(dictionary.keys())

for key in dictionary.keys():
    print(key)
```

Program Output

```
dict_keys(['a', 'b', 'c'])
a
b
c
```

More about Python Dictionary keys() method.

dict.values()

Python Dictionary values() method returns the view object for the dictionary values.

The view object returned by values() method is similar to that of returned by items() method. The only difference between these methods is that items() return key:value pairs while values() return only values.

In the following program, we will take a dictionary and iterate over the dictionary values using the view object dict.values().

Example Program

```
dictionary = {'a': 54, 'b': 87, 'c': 61}
print(dictionary.values())

for value in dictionary.values():
    print(value)
```

Program Output

```
dict_values([54, 87, 61])
54
87
61
```

More about Python Dictionary values() method.

dict.fromkeys(sequence[, value])

Python Dictionary fromkeys(sequence[, value]) method returns a new dictionary created using the items of sequence as keys and the value for each key.

The second argument, value, is optional to fromkeys() method. If this value is not provided, then None would be the default value for each key in the newly created dictionary.

In the following program, we will create a dictionary from a list of strings as keys, and `0` as the default value for each of the keys.

Example Program

```
dictionary = dict.fromkeys(['a', 'b', 'c'], 0)
print(dictionary)
```

Program Output

```
{'a': 0, 'b': 0, 'c': 0}
```

In the following program, we will call dict.fromkeys() method with only the sequence provided. Value is not given as argument.

Example Program

```
dictionary = dict.fromkeys(['a', 'b', 'c'])
print(dictionary)
```

Program Output

```
{'a': None, 'b': None, 'c': None}
```

dict.pop(key[, default])

Python Dictionary pop(key[, default]) method removes the key:value pair corresponding to the given key, and returns the value of removed key:value pair.

If key is not present in the dictionary, pop() throws KeyError. You can override this behavior by passing a default value as second argument. When key is not present in the dictionary, pop() returns the default value.

In the following program, we will take a dictionary and pop() key `'b'` from the dictionary.

Example Program

```
dictionary = {'a': 54, 'b': 87, 'c': 61}
value = dictionary.pop('b')
print(value)
```

Program Output

In the following program, we will take a dictionary and pop() key `'f'` from the dictionary, but the key is not present in the dictionary.

Example Program

```
dictionary = {'a': 54, 'b': 87, 'c': 61}
value = dictionary.pop('f')
print(value)
```

Program Output

```
Traceback (most recent call last):
  File "d:/workspace/python/example.py", line 2, in <module>
    value = dictionary.pop('f')
KeyError: 'f'
```

In the following program, we will take a dictionary and pop() key `'f'` from the dictionary and the key is not present in dictionary. But, we will give a default value to pop() method.

Example Program

```
dictionary = {'a': 54, 'b': 87, 'c': 61}
value = dictionary.pop('f', 0)
print(value)
```

Program Output

```
0
```

More about Python Dictionary pop() method.

dict.popitem()

Python Dictionary popitem() method removes and returns a key:value pair from the dictionary.

popitem() throws KeyError if called on an empty dictionary.

In the following program, we will take a dictionary and popitem() method iteratively on a dictionary.

Example Program

```
dictionary = {'a': 54, 'b': 87, 'c': 61}

while dictionary:
    print(dictionary.popitem())
```

Program Output

```
('c', 61)
('b', 87)
('a', 54)
```

popitem() method returns key:value pairs in Last-In-First-Out order.

More about Python Dictionary popitem() method.

dict.setdefault(key[, default])

Python Dictionary setdefault() method

- returns value corresponding to the key from dictionary, if key is present in the dictionary.
- adds key:value pair, with value=default, to the dictionary if key is not present in the dictionary. The value of **default** by default is None.

In the following program, we will take a dictionary and call setdefault() method with a key present in the dictionary.

Example Program

```
dictionary = {'a': 54, 'b': 87, 'c': 61}
value = dictionary.setdefault('b')
print(value)
```

Program Output

```

```

In the following program, we will take a dictionary and call setdefault() method with a key not present in the dictionary, and a default value of `0`.

Example Program

```
dictionary = {'a': 54, 'b': 87, 'c': 61}
value = dictionary.setdefault('f', 0)
print(value)
print(dictionary)
```

Program Output

```
0
{'a': 54, 'b': 87, 'c': 61, 'f': 0}
```

More about Python Dictionary setdefault() method.

dict.update([other])

Python Dictionary update([other]) method updates this dictionary's values with the matching keys from the other dictionary. And keys that are not present in this dictionary are added.

update() method modifies the original dictionary and returns None.

In the following program, we will take a dictionary and update its values from the items of another dictionary.

Example Program

```
dictionary1 = {'a': 54, 'b': 87, 'c': 61}
dictionary2 = {'a': 77, 'b': 44, 'p': 1}
dictionary1.update(dictionary2)
print(dictionary1)
```

Program Output

```
{'a': 77, 'b': 44, 'c': 61, 'p': 1}
```

More about Python Dictionary update() method.

Conclusion

In this Python Tutorial, we learned about Python Dictionary Methods and their usage with explanation and example programs.

Python Programming

- Python Tutorial
- Install Python
- Install Anaconda Python
- Python HelloWorld Program
- Python Variables
- Python Variable Data Type Conversion
- Python Comments

Control Statements

- Python If
- Python If Else
- Python While Loop
- Python For Loop

Python String

- Python String Methods
- Python String Length
- Python String Replace
- Python Split String
- Python Count Occurrences of Sub-String
- Python Sort List of Strings

Functions

- Python Functions

Python Collections

- Python List
- Python Dictionary

Advanced

- Python Multithreading

Useful Resources

- Python Interview Questions