[12]Removing Items from a Dictionary

There are several methods to remove items from a dictionary:

Example

The pop() method removes the item with the specified key name:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.pop("model")
print(thisdict)
```

Example

The popitem() method removes the last inserted item (in versions before 3.7, a random item is removed instead):

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.popitem()
```

Work Cited

[1] XXX

[2] dddd

zz, xx yy. “title1.” *https://w3school.com/*, 05/09/2024 sep 2024, https://w3school.com/. Accessed 05/09/2024 sep 2024.

```
print(thisdict)
```

Del Keyword

The del keyword removes the item with the specified key name:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
del thisdict["model"]
print(thisdict)
```

Example

The del keyword can also delete the dictionary completely:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
```

```
del thisdict
print(thisdict) #this will cause an error because "thisdict" no longer exists.
```

Example

The clear() keyword empties the dictionary:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.clear()
print(thisdict)
```

mnnnn

➢ Dictionary Method

Python has a set of built-in methods that you can use on dictionaries.

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary

fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

➢ Clear() :-

The clear() keyword empties the dictionary:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}

print(thisdict)
```

➢ Copy ()

They copy() method returns a shallow copy of the dictionary.

```
dict.copy()
```

copy() Parameters :-copy() method doesn't take any parameters.

Return Value from copy ()

This method returns a shallow copy of the dictionary. It doesn't modify the original dictionary.

Example 1: How copy works for dictionaries?

```
original = {1:'one', 2:'two'}
new = original.copy()

print('Orignal: ', original)
print('New: ', new)
```

Output

```
Orignal:  {1: 'one', 2: 'two'}
New:  {1: 'one', 2: 'two'}
```

➢ get()

The get() method returns the value for the specified key if key is in dictionary.

The syntax of get() is:

```
dict.get(key[, value])
```

get() Parameters

get() method takes maximum of two parameters:

- **key** - key to be **searched** in the dictionary
- **Value** (optional) - Value to be returned if the key is not found. The default value is None.

Return Value from get()

get() method returns:

- the value for the specified key if key is in dictionary.
- None if the key is **not** found and value is not specified.
- value if the key is not found and value is specified.

Example :

```
Person  = {'name': 'Phill', 'age': 22}

print('Name: ', person.get('name'))
print('Age: ‘ person.get('age'))

# Value is not provided
print('Salary: ‘ person.get('salary'))

# value is provided
print('Salary: ‘ person.get('salary', 0.0))
```

Output

```
Name:  Phill
Age:  22
Salary:  None
Salary:  0.0
```

➢ **items()**

The items() method returns a view object that displays a list of dictionary's (key, value) tuple pairs.

The syntax of items() method is:

```
dictionary.items()
```

items() Parameters

items() method doesn't take any parameters.

Return value from items()

items() method returns a view object that displays a list of a given dictionary's (key, value) tuple pair.

Example 1: Get all items of a dictionary with items ()

```
# random sales dictionary
sales = { 'apple': 2, 'orange': 3, 'grapes': 4}

print(sales.items())
```

Output

```
dict_items([('apple', 2), ('orange', 3), ('grapes', 4)])
```

➢ keys ()

The keys () method returns a view object that displays a list of all the keys in the dictionary

The syntax of keys() is:

```
dict.keys()
```

keys() Parameters

keys() doesn't take any parameters.

Return Value from keys ()

Keys () returns a view object that displays a list of all the keys. When the dictionary is changed, the view object also reflects these changes.

Example 1: How keys () works?

```
person = {'name': 'Phill', 'age': 22, 'salary': 3500.0}
print (person.keys ())

empty_dict = {}
print (empty_dict. Keys())
```

Output

```
dict_keys(['name', 'salary', 'age'])
dict_keys([])
```

➢ fromkeys() Method

The fromkeys() method returns a dictionary with the specified keys and the specified value.

Syntax

dict.fromkeys(*keys, value*)

Parameter Values

Parameter	Description
keys	Required. An iterable specifying the keys of the new dictionary
value	Optional. The value for all keys. Default value is None

Example

Create a dictionary with 3 keys, all with the value 0:

```
x = ('key1', 'key2', 'key3')
y = 0

thisdict = dict.fromkeys(x, y)
```

```
print(thisdict)
```