

Python Dictionaries

By Dr.Bijoy Kumar Mandal

- **Dictionary**

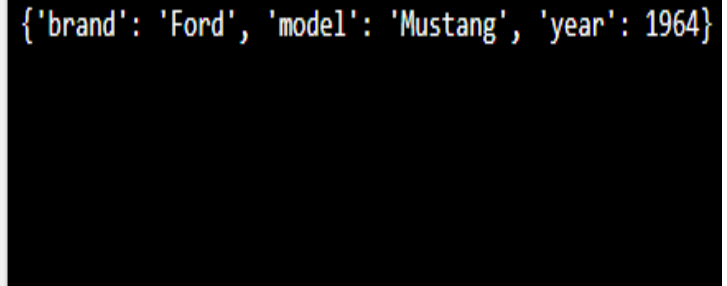
A dictionary is a collection which is unordered, changeable and indexed. In Python dictionaries are written with curly brackets, and they have keys and values.

- **Example**

- **Create and print a dictionary:**

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

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

A screenshot of a terminal window with a black background. The text displayed is a Python dictionary: {'brand': 'Ford', 'model': 'Mustang', 'year': 1964}. The text is rendered in a monospaced font with syntax highlighting: single quotes are red, strings are green, and integers are blue.

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

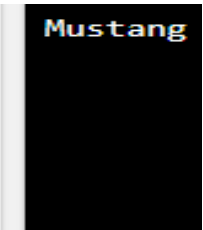
- # Accessing Items

You can access the items of a dictionary by referring to its key name, inside square brackets:

- Example
- Get the value of the "model" key:

```
x = thisdict["model"]
```

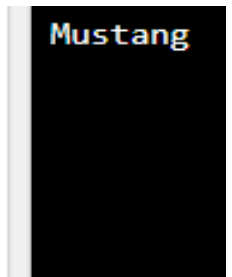
```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
x = thisdict["model"]  
print(x)
```



- There is also a method called `get()` that will give you the same result:
- Example
- Get the value of the "model" key:

```
x = thisdict.get("model")
```

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



- **Change Values**

You can change the value of a specific item by referring to its key name

- **Example**

- **Change the "year" to 2018:**

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

```
thisdict["year"] = 2018
```

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

```
thisdict["year"] = 2018
```

```
print(thisdict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 2018}
```

- **Loop Through a Dictionary**

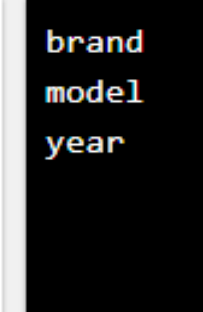
You can loop through a dictionary by using a for loop. When looping through a dictionary, the return value are the keys of the dictionary, but there are methods to return the values as well.

- **Example**

- **Print all key names in the dictionary, one by one:**

```
for x in thisdict:  
    print(x)
```

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
for x in thisdict:  
    print(x)
```



brand
model
year

- **Example**
- **Print all values in the dictionary, one by one:**

for x in thisdict:

print(thisdict[x])

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

Ford
Mustang
1964

- **Example**
- **You can also use the values() method to return values of a dictionary:**

for x in thisdict.values():

print(x)

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
for x in thisdict.values():  
    print(x)
```

Ford
Mustang
1964

- Dictionary Length

To determine how many items (key-value pairs) a dictionary has, use the `len()` function.

- **Example**

- **Print the number of items in the dictionary:**

`print(len(thisdict))`

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

- **Check if Key Exists**

To determine if a specified key is present in a dictionary use the in keyword:

- **Example**

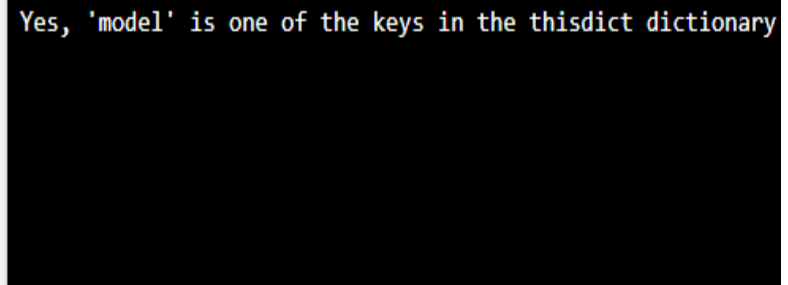
- **Check if "model" is present in the dictionary:**

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

if "model" in thisdict:

print("Yes, 'model' is one of the keys in the thisdict dictionary")

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
if "model" in thisdict:  
    print("Yes, 'model' is one of the keys in the thisdict dictionary")
```



```
Yes, 'model' is one of the keys in the thisdict dictionary
```


- Adding Items

Adding an item to the dictionary is done by using a new index key and assigning a value to it:

- Example

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

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

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'red'}
```

- **Removing Items**

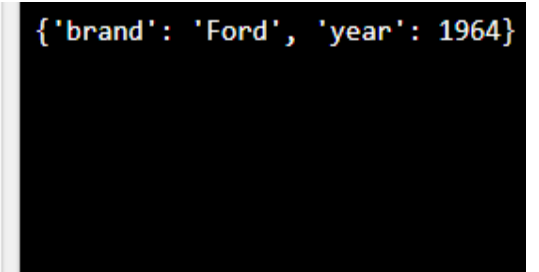
There are several methods to remove items from a dictionary:

- **Example**

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

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

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



```
{'brand': 'Ford', 'year': 1964}
```

- **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()  
print(thisdict)
```

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

```
{'brand': 'Ford', 'model': 'Mustang'}
```

- **Example**
- **The del keyword removes the item with the specified key name:**

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

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
del thisdict  
print(thisdict) #this will cause an error because "thisdict" no longer exists.
```

```
Traceback (most recent call last):  
  File "demo_dictionary_del3.py", line 7, in <module>  
    print(thisdict) #this will cause an error because "thisdict" no longer ex  
NameError: name 'thisdict' is not defined
```

- **Copy a Dictionary**

You cannot copy a dictionary simply by typing `dict2 = dict1`, because: `dict2` will only be a reference to `dict1`, and changes made in `dict1` will automatically also be made in `dict2`.

There are ways to make a copy, one way is to use the built-in Dictionary method `copy()`.

- **Example**

- **Make a copy of a dictionary with the `copy()` method:**

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

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

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

- **Nested Dictionaries**

A dictionary can also contain many dictionaries, this is called nested dictionaries.

- **Example**

- **Create a dictionary that contain three dictionaries:**

```
myfamily = {  
    "child1" : {  
        "name" : "Emil",  
        "year" : 2004  
    },  
    "child2" : {  
        "name" : "Tobias",  
        "year" : 2007  
    },  
    "child3" : {  
        "name" : "Linus",  
        "year" : 2011  
    }  
}
```

THANK YOU