Python Dictionary

**Dictionary**in Python is an unordered collection of data values, used to store data values like a map, which unlike other Data Types that hold only single value as an element, Dictionary holds **key:value** pair. Key value is provided in the dictionary to make it more optimized.

In Python, a Dictionary can be created by placing sequence of elements within curly **{}** braces, separated by ‘comma’. Dictionary holds a pair of values, one being the Key and the other corresponding pair element being its **Key:value**. Values in a dictionary can be of any datatype and can be duplicated, whereas keys can’t be repeated and must be immutable.

Accessing Values in Dictionary

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value. Following is a simple example −

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

print ("dict['Name']: ", dict['Name'])

print ("dict['Age']: ", dict['Age'])

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

dict['Name']: Zara

dict['Age']: 7

Updating Dictionary

You can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry as shown in a simple example given below.

[Live Demo](http://tpcg.io/cXjIA7)

#!/usr/bin/python3

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

dict['Age'] = 8; # update existing entry

dict['School'] = "DPS School" # Add new entry

print ("dict['Age']: ", dict['Age'])

print ("dict['School']: ", dict['School'])

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

dict['Age']: 8

dict['School']: DPS School

Delete Dictionary Elements

You can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete entire dictionary in a single operation.

To explicitly remove an entire dictionary, just use the **del** statement. Following is a simple example −

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

del dict['Name'] # remove entry with key 'Name'

dict.clear() # remove all entries in dict

del dict # delete entire dictionary

print ("dict['Age']: ", dict['Age'])

print ("dict['School']: ", dict['School']

### Using Methods to Access Elements

In addition to using keys to access values, we can also work with some built-in methods:

* dict.keys() isolates keys
* dict.values() isolates values
* dict.items() returns items in a list format of (key, value) tuple pairs

The pop() method removes the specified item from the dictionary.

### **Example**

Remove "model" from the dictionary:

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

### **Example**

Remove the last item from the dictionary:

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

### **Example**

Return the keys:

car = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
  
x = car.keys()  
  
print(x)

### **Example**

Return the values:

car = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
  
x = car.values()  
  
print(x)

### **Example**

Copy the car dictionary:

car = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
  
x = car.copy()  
  
print(x)

**7. When we iterate through a dictionary using a for loop, we actually iterate over the keys:**

**d = { "key1":1, "key2":2, "key3":1, "key4":3, "key5":1, "key6":4, "key7":2 }**

**for k in d :**

**print("key=", k, " value=", d[k], sep="")**

## 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)**

### **Example**

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

**for x in thisdict:  
  print(thisdict[x])**

### **Example**

**You can also use the values() function to return values of a dictionary:**

**for x in thisdict.values():  
  print(x)**

### **Example**

**Loop through both keys and values, by using the items() function:**

**for x, y in thisdict.items():  
  print(x, y)**

## 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")**

## Dictionary Length

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

### **Example**

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

**print(len(thisdict))**

## 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)**

## 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)**

### **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)**

### **Example**

**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**

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)**

**Another way to make a copy is to use the built-in method dict().**

**Example**

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

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

**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  
  }  
}**

**Or, if you want to nest three dictionaries that already exists as dictionaries:**

**Example**

**Create three dictionaries, than create one dictionary that will contain the other three dictionaries:**

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

**The dict() Constructor**

**It is also possible to use the dict() constructor to make a new dictionary:**

**Example**

**thisdict = dict(brand="Ford", model="Mustang", year=1964)  
# note that keywords are not string literals  
# note the use of equals rather than colon for the assignment  
print(thisdict)**

# **Python Dictionary Comprehension**

A dictionary comprehension takes the form **{key: value for (key, value) in iterable}**

Let’s see a example,lets assume we have two lists named keys and value now,

**# Python code to demonstrate dictionary**

**# comprehension**

**# Lists to represent keys and values**

**keys = ['a','b','c','d','e']**

**values = [1,2,3,4,5]**

**# but this line shows dict comprehension here**

**myDict = { k:v for (k,v) in zip(keys, values)}**

**# We can use below too**

**# myDict = dict(zip(keys, values))**

**print (myDict)**

**output**

**{'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}**

**# Python code to demonstrate dictionary**

**# creation using list comprehension**

**myDict = {x: x\*\*2 for x in [1,2,3,4,5]}**

**print (myDict)**

**Output :**

**{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}**

**sDict = {x.upper(): x\*3 for x in 'coding '}**

**print (sDict)**

**Output :**

**{'O': 'ooo', 'N': 'nnn', 'I': 'iii', 'C': 'ccc', 'D': 'ddd', 'G': 'ggg'}**

**# Python code to demonstrate dictionary**

**# comprehension using if.**

**newdict = {x: x\*\*3 for x in range(10) if x\*\*3 % 4 == 0}**

**print(newdict)**

**Output :**

**{0: 0, 8: 512, 2: 8, 4: 64, 6: 216}**