**Dictionaries** are Python’s implementation of a data structure that is more generally known as an associative array. A dictionary consists of a collection of key-value pairs. Each key-value pair maps the key to its associated value.

You can define a dictionary by enclosing a comma-separated list of key-value pairs in curly braces ({}). A colon (:) separates each key from its associated value:

**What is a Dictionary in Python?**

A **Dictionary in Python** is the unordered and changeable collection of data values that holds key-value pairs. Each key-value pair in the dictionary maps the key to its associated value making it more optimized. A Dictionary in python is declared by enclosing a comma-separated list of key-value pairs using curly braces({}). Python Dictionary is classified into two elements: Keys and Values.

* Keys will be a single element
* Values can be a list or list within a list, numbers, etc.
* **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.
* **Note –**Keys in a dictionary doesn’t allows Polymorphism.

## Creating a Dictionary

* 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.
* **Note –**Dictionary keys are case sensitive, same name but different cases of Key will be treated distinctly.

|  |
| --- |
| # Creating a Dictionary  # with Integer Keys  Dict = {1: 'World', 2: 'For', 3: 'World'}  print("\nDictionary with the use of Integer Keys: ")  print(Dict)    # Creating a Dictionary  # with Mixed keys  Dict = {'Name': 'World', 1: [1, 2, 3, 4]}  print("\nDictionary with the use of Mixed Keys: ")  print(Dict) |

* **Output:**

Dictionary with the use of Mixed Keys:

{'Name': 'World', 1: [1, 2, 3, 4]}

* Dictionary with the use of Integer Keys:
* {1: 'World', 2: 'For', 3: 'World'}
* Dictionary with the use of Mixed Keys:
* {1: [1, 2, 3, 4], 'Name': 'World'}

**Properties of Dictionary Keys**

There are two important points while using dictionary keys

* More than one entry per key is not allowed ( no duplicate key is allowed)
* The values in the dictionary can be of any type, while the keys must be immutable like numbers, tuples, or strings.
* Dictionary keys are case sensitive- Same key name but with the different cases are treated as different keys in Python dictionaries.
* **Python dictionary** is an unordered collection of items. Each item of a dictionary has a key/value pair.
* Dictionaries are optimized to retrieve values when the key is known.
* **Creating Python Dictionary**
* Creating a dictionary is as simple as placing items inside curly braces {} separated by commas.
* An item has a key and a corresponding value that is expressed as a pair (**key: value**).
* While the values can be of any data type and can repeat, keys must be of immutable type ([string](https://www.programiz.com/python-programming/string), [number](https://www.programiz.com/python-programming/numbers) or [tuple](https://www.programiz.com/python-programming/tuple) with immutable elements) and must be unique.
* Example-
* # empty dictionary
* my\_dict = {}
* # dictionary with integer keys
* my\_dict = {1: 'apple', 2: 'ball'}
* # dictionary with mixed keys
* my\_dict = {'name': 'John', 1: [2, 4, 3]}
* # using dict()
* my\_dict = dict({1:'apple', 2:'ball'})
* # from sequence having each item as a pair
* my\_dict = dict([(1,'apple'), (2,'ball')])
* As you can see from above, we can also create a dictionary using the built-in dict() function.

## Accessing Elements from Dictionary

* While indexing is used with other data types to access values, a dictionary uses keys. Keys can be used either inside square brackets [] or with the get() method.
* If we use the square brackets [], KeyError is raised in case a key is not found in the dictionary. On the other hand, the get() method returns None if the key is not found.
* # get vs [] for retrieving elements
* my\_dict = {'name': 'Jack', 'age': 26}
* # Output: Jack
* print(my\_dict['name'])
* # Output: 26
* print(my\_dict.get('age'))
* # Trying to access keys which doesn't exist throws error
* # Output None
* print(my\_dict.get('address'))
* # KeyError
* print(my\_dict['address'])
* **Output**
* Jack
* 26
* None
* Traceback (most recent call last):
* File "<string>", line 15, in <module>
* print(my\_dict['address'])
* KeyError: 'address'

**Properties of Dictionary Keys**

There are two important points while using dictionary keys

* More than one entry per key is not allowed ( no duplicate key is allowed)
* The values in the dictionary can be of any type, while the keys must be immutable like numbers, tuples, or strings.
* Dictionary keys are case sensitive- Same key name but with the different cases are treated as different keys in Python dictionaries.
* Dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}
* print (Dict['Tiffany'])

output – 22

## Python Dictionary Methods

### Copying dictionary

You can also copy the entire dictionary to a new dictionary. For example, here we have copied our original dictionary to the new dictionary name “Boys” and “Girls”.

Dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}

Boys = {'Tim': 18,'Charlie':12,'Robert':25}

Girls = {'Tiffany':22}

studentX=Boys.copy()

studentY=Girls.copy()

print(studentX)

print(studentY)

## Updating Dictionary

You can also update a dictionary by adding a new entry or a key-value pair to an existing entry or by deleting an existing entry. Here in the example, we will add another name, “Sarah” to our existing dictionary.

Dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}

Dict.update({"Sarah":9})

print Dict

output-

{'Tim': 18, 'Charlie': 12, 'Tiffany': 22, 'Robert': 25, 'Sarah': 9}

### Delete Keys from the dictionary

Python dictionary gives you the liberty to delete any element from the dictionary list. Suppose you don’t want the name Charlie in the list, so you can remove the key element by the following code.

Dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}

del Dict ['Charlie']

print Dict

output-

{'Tim': 18, 'Tiffany': 22, 'Robert': 25}

### Dictionary items() Method

The items() method returns a list of tuple pairs (Keys, Value) in the dictionary.

Dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}

print "Students Name: %s" % Dict.items()

## Check if a given key already exists in a dictionary

For a given list, you can also check whether our child dictionary exists in the main dictionary or not. Here we have two sub-dictionaries “Boys” and “Girls”, now we want to check whether our dictionary Boys exist in our main “Dict” or not. For that, we use the for loop method with else if method.

Dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}

Boys = {'Tim': 18,'Charlie':12,'Robert':25}

Girls = {'Tiffany':22}

for key in Boys.keys():

if key in Dict.keys():

print True

else:

print False

OUTPUT-

True

True

True

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-

### Sorting the Dictionary

In the dictionary, you can also sort the elements. For example, if we want to print the name of the elements of our dictionary alphabetically, we have to use the for a loop. It will sort each element of the dictionary accordingly.

Dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}

Boys = {'Tim': 18,'Charlie':12,'Robert':25}

Girls = {'Tiffany':22}

Students = list(Dict.keys())

Students.sort()

for S in Students:

print":".join((S,str(Dict[S])))

output-

Charlie:12

Robert:25

Tiffany:22

Tim:18

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 |

## Python Dictionary in-built Functions

### Dictionary len() Method

The len() function gives the number of pairs in the dictionary.

Dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}

print "Length : %d" % len (Dict)

OUTPUT-

Length : %d 4

## Variable Types

Python does not require to explicitly declare the reserve memory space; it happens automatically. The assign values to variable “=” equal sign are used. The code to determine the variable type is ” %type (Dict).”

Dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}

print "variable Type: %s" %type (Dict)

**Here is the list of all Dictionary Methods**

| **Method** | **Description** | **Syntax** |
| --- | --- | --- |
| copy() | Copy the entire dictionary to new dictionary | dict.copy() |
| update() | Update a dictionary by adding a new entry or a key-value pair to anexisting entry or by deleting an existing entry. | Dict.update([other]) |
| items() | Returns a list of tuple pairs (Keys, Value) in the dictionary. | dictionary.items() |
| sort() | You can sort the elements | dictionary.sort() |
| len() | Gives the number of pairs in the dictionary. | len(dict) |
| cmp() | Compare the values and keys of two dictionaries | cmp(dict1, dict2) |
| Str() | Make a dictionary into a printable string format | Str(dict) |

### Summary:

* Dictionaries in a programming language is a type of data-structure used to store information connected in some way.
* Python Dictionary are defined into two elements Keys and Values.
* Dictionaries do not store their information in any particular order, so you may not get your information back in the same order you entered it.
* Keys will be a single element
* Values can be a list or list within a list, numbers, etc.
* More than one entry per key is not allowed ( no duplicate key is allowed)
* The values in the dictionary can be of any type, while the keys must be immutable like numbers, tuples, or strings.
* Dictionary keys are case sensitive- Same key name but with the different cases are treated as different keys in Python dictionaries.