13. Python 3 – Dictionary

Each key is separated from its value by a colon (:), the items are separated by commas, and the whole thing is enclosed in curly braces. An empty dictionary without any items is written with just two curly braces, like this: {}.

Keys are unique within a dictionary while values may not be. The values of a dictionary can be of any type, but the keys must be of an immutable data type such as strings, numbers, or tuples.

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.

```
#!/usr/bin/python3

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
```

If we attempt to access a data item with a key, which is not a part of the dictionary, we get an error as follows-

```
#!/usr/bin/python3

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

print "dict['Alice']: ", dict['Alice']
```

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

```
dict['Zara']:
Traceback (most recent call last):
  File "test.py", line 4, in <module>
    print "dict['Alice']: ", dict['Alice'];
KeyError: 'Alice'
```

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.

```
#!/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-

```
#!/usr/bin/python3

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'])
```

This produces the following result.

Note: An exception is raised because after **del dict,** the dictionary does not exist anymore.

```
dict['Age']:
Traceback (most recent call last):
  File "test.py", line 8, in <module>
```

```
    print "dict['Age']: ", dict['Age'];
TypeError: 'type' object is unsubscriptable
```

Note: The del() method is discussed in subsequent section.

Properties of Dictionary Keys

Dictionary values have no restrictions. They can be any arbitrary Python object, either standard objects or user-defined objects. However, same is not true for the keys.

There are two important points to remember about dictionary keys-

(a) More than one entry per key is not allowed. This means no duplicate key is allowed. When duplicate keys are encountered during assignment, the last assignment wins. For example-

```
#!/usr/bin/python3

dict = {'Name': 'Zara', 'Age': 7, 'Name': 'Manni'}

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

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

```
dict['Name']:  Manni
```

(b) Keys must be immutable. This means you can use strings, numbers or tuples as dictionary keys but something like ['key'] is not allowed. Following is a simple example-

```
#!/usr/bin/python3

dict = {['Name']: 'Zara', 'Age': 7}

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

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

```
Traceback (most recent call last):
  File "test.py", line 3, in <module>
    dict = {['Name']: 'Zara', 'Age': 7}
TypeError: list objects are unhashable
```

Built-in Dictionary Functions & Methods

Python includes the following dictionary functions-

SN	Functions with Description
1	**cmp(dict1, dict2)** No longer available in Python 3.
2	**len(dict)** Gives the total length of the dictionary. This would be equal to the number of items in the dictionary.
3	**str(dict)** Produces a printable string representation of a dictionary.
4	**type(variable)** Returns the type of the passed variable. If passed variable is dictionary, then it would return a dictionary type.

Dictionary len() Method

DescriptionThe method len() gives the total length of the dictionary. This would be equal to the number of items in the dictionary.

Syntax

Following is the syntax for len() method-

```
len(dict)
```

Parameters

dict - This is the dictionary, whose length needs to be calculated.

Return Value

This method returns the length.

Example

The following example shows the usage of len() method.

```
#!/usr/bin/python3
```

```
dict = {'Name': 'Manni', 'Age': 7, 'Class': 'First'}
print ("Length : %d" % len (dict))
```

When we run the above program, it produces the following result-

```
Length : 3
```

Dictionary str() Method

Description

The method **str()** produces a printable string representation of a dictionary.

Syntax

Following is the syntax for str() method −

```
str(dict)
```

Parameters

dict - This is the dictionary.

Return Value

This method returns string representation.

Example

The following example shows the usage of str() method.

```
#!/usr/bin/python3
dict = {'Name': 'Manni', 'Age': 7, 'Class': 'First'}
print ("Equivalent String : %s" % str (dict))
```

When we run the above program, it produces the following result-

```
Equivalent String : {'Name': 'Manni', 'Age': 7, 'Class': 'First'}
```

Dictionary type() Method

Description

The method **type()** returns the type of the passed variable. If passed variable is dictionary then it would return a dictionary type.

Syntax

Following is the syntax for type() method-

```
type(dict)
```

Parameters

dict - This is the dictionary.

Return Value

This method returns the type of the passed variable.

Example

The following example shows the usage of type() method.

```
#!/usr/bin/python3
dict = {'Name': 'Manni', 'Age': 7, 'Class': 'First'}
print ("Variable Type : %s" %  type (dict))
```

When we run the above program, it produces the following result-

```
Variable Type : <type 'dict'>
```

Python includes the following dictionary methods-

SN	Methods with Description
1	**dict.clear()** Removes all elements of dictionary *dict.*
2	**dict.copy()** Returns a shallow copy of dictionary *dict.*
3	**dict.fromkeys()** Create a new dictionary with keys from seq and values *set* to *value*.
4	**dict.get(key, default=None)** For *key* key, returns value or default if key not in dictionary.

5	**dict.has_key(key)** Removed, use the **in** operation instead.
6	**dict.items()** Returns a list of *dict*'s (key, value) tuple pairs.
7	**dict.keys()** Returns list of dictionary dict's keys.
8	**dict.setdefault(key, default=None)** Similar to get(), but will set dict[key]=default if *key* is not already in dict.
9	**dict.update(dict2)** Adds dictionary *dict2*'s key-values pairs to *dict.*
10	**dict.values()** Returns list of dictionary *dict*'s values.

Dictionary clear() Method

Description

The method **clear()** removes all items from the dictionary.

Syntax

Following is the syntax for clear() method-

```
dict.clear()
```

Parameters

NA

Return Value

This method does not return any value.

Example

The following example shows the usage of clear() method.

```
#!/usr/bin/python3
```

```
dict = {'Name': 'Zara', 'Age': 7}
print ("Start Len : %d" % len(dict))
dict.clear()
print ("End Len : %d" %  len(dict))
```

When we run the above program, it produces the following result-

```
Start Len : 2
End Len : 0
```

Dictionary copy() Method

Description

The method **copy()** returns a shallow copy of the dictionary.

Syntax

Following is the syntax for copy() method-

```
dict.copy()
```

Parameters

NA

Return Value

This method returns a shallow copy of the dictionary.

Example

The following example shows the usage of copy() method.

```
#!/usr/bin/python3
dict1 = {'Name': 'Manni', 'Age': 7, 'Class': 'First'}
dict2 = dict1.copy()
print ("New Dictionary : ",dict2)
```

When we run the above program, it produces following result-

```
New dictionary :  {'Name': 'Manni', 'Age': 7, 'Class': 'First'}
```

Dictionary fromkeys() Method

Description

The method fromkeys() creates a new dictionary with keys from seq and values set to value.

Syntax

Following is the syntax for fromkeys() method-

```
dict.fromkeys(seq[, value]))
```

Parameters

- **seq** - This is the list of values which would be used for dictionary keys preparation.
- **value** - This is optional, if provided then value would be set to this value

Return Value

This method returns the list.

Example

The following example shows the usage of fromkeys() method.

```
#!/usr/bin/python3
seq = ('name', 'age', 'sex')
dict = dict.fromkeys(seq)
print ("New Dictionary : %s" % str(dict))
dict = dict.fromkeys(seq, 10)
print ("New Dictionary : %s" %  str(dict))
```

When we run the above program, it produces the following result-

```
New Dictionary : {'age': None, 'name': None, 'sex': None}
New Dictionary : {'age': 10, 'name': 10, 'sex': 10}
```

Dictionary get() Method

Description

The method **get()** returns a value for the given key. If the key is not available then returns default value None.

Syntax

Following is the syntax for get() method-

```
dict.get(key, default=None)
```

Parameters

- **key** - This is the Key to be searched in the dictionary.
- **default** - This is the Value to be returned in case key does not exist.

Return Value

This method returns a value for the given key. If the key is not available, then returns default value as None.

Example

The following example shows the usage of get() method.

```
#!/usr/bin/python3
dict = {'Name': 'Zara', 'Age': 27}
print ("Value : %s" %  dict.get('Age'))
print ("Value : %s" %  dict.get('Sex', "NA"))
```

When we run the above program, it produces the following result-

```
Value : 27
Value : NA
```

Dictionary items() Method

Description

The method items() returns a list of dict's (key, value) tuple pairs.

Syntax

Following is the syntax for items() method-

```
dict.items()
```

Parameters

NA

Return Value

This method returns a list of tuple pairs.

Example

The following example shows the usage of items() method.

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7}
print ("Value : %s" %  dict.items())
```

When we run the above program, it produces the following result-

```
Value : [('Age', 7), ('Name', 'Zara')]
```

Dictionary keys() Method

Description

The method **keys()** returns a list of all the available keys in the dictionary.

Syntax

Following is the syntax for keys() method-

```
dict.keys()
```

Parameters

NA

Return Value

This method returns a list of all the available keys in the dictionary.

Example

The following example shows the usage of keys() method.

```
#!/usr/bin/python3
dict = {'Name': 'Zara', 'Age': 7}
print ("Value : %s" %  dict.keys())
```

When we run the above program, it produces the following result-

```
Value : ['Age', 'Name']
```

Dictionary setdefault() Method

Description

The method setdefault() is similar to get(), but will set dict[key]=default if the key is not already in dict.

Syntax

Following is the syntax for setdefault() method-

```
dict.setdefault(key, default=None)
```

Parameters

- **key** - This is the key to be searched.
- **default** - This is the Value to be returned in case key is not found.

Return Value

This method returns the key value available in the dictionary and if given key is not available then it will return provided default value.

Example

The following example shows the usage of setdefault() method.

```
#!/usr/bin/python3
dict = {'Name': 'Zara', 'Age': 7}
print ("Value : %s" % dict.setdefault('Age', None))
print ("Value : %s" %  dict.setdefault('Sex', None))
print (dict)
```

When we run the above program, it produces the following result-

```
Value : 7
Value : None
{'Name': 'Zara', 'Sex': None, 'Age': 7}
```

Dictionary update() Method

Description

The method **update()** adds dictionary dict2's key-values pairs in to dict. This function does not return anything.

Syntax

Following is the syntax for update() method-

```
dict.update(dict2)
```

Parameters

dict2 - This is the dictionary to be added into dict.

Return Value

This method does not return any value.

Example

The following example shows the usage of update() method.

```
#!/usr/bin/python3
dict = {'Name': 'Zara', 'Age': 7}
dict2 = {'Sex': 'female' }
dict.update(dict2)
print ("updated dict : ", dict)
```

When we run the above program, it produces the following result-

```
updated dict :  {'Sex': 'female', 'Age': 7, 'Name': 'Zara'}
```

Dictionary values() Method

Description

The method **values()** returns a list of all the values available in a given dictionary.

Syntax

Following is the syntax for values() method-

```
dict.values()
```

Parameters

NA

Return Value

This method returns a list of all the values available in a given dictionary.

Example

The following example shows the usage of values() method.

```
#!/usr/bin/python3
dict = {'Sex': 'female', 'Age': 7, 'Name': 'Zara'}
```

```
print ("Values : ",  list(dict.values()))
```

When we run above program, it produces following result-

```
Values :  ['female', 7, 'Zara']
```