**9.5. DICTIONARY**

Dictionary is an **unordered set of key and value pair**. It is a container that contains data, enclosed within curly braces.

The pair i.e., **key** and **value** is known as **item**. The **key** passed in the item must be **unique**.

**Word : meaning1,meaning2,meaning3**

**Key : value**

The key and the value is separated by a **colon(:).** This pair is known as item. Items are separated from each other by a comma(,). Different items are enclosed within a curly brace and this forms 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 {}

1. **Keys** are **unique** within a dictionary while **values may not be**.
2. **keys** must be of an **immutable** data type such as **number , bool, string, or tuple**.

The **values** of a dictionary can be of any type,

Dictionary is **mutable,** why because values can be updated.

**Python Dictionary Example:**

**1. CREATE : gmail id registration UserName Password MObileNo emailed**

Data = {100:'Vijay', 101:'Rahul', 100 :’Madhu’}

Print ( data[‘edetails’][‘sal’] ) details[3][4]

**print**(data)

{100: 'Ravi', 101: 'Vijay', 102: 'Rahul'}

plant={}

plant[1]='Ravi'  {1:’Ravi’}

plant[2]='Manoj'  {1:’Ravi’, 2:’Manoj’,’name’:’Hari’}

plant['name']='Hari'

plant[4]='Om'

**print**(plant[2])

**print**(plant['name'])

**print**(plant[1])

**print**(plant)

 Manoj

Hari

Ravi

{1: 'Ravi', 2: 'Manoj', 4: 'Om', 'name': 'Hari'}

## 2. RETRIEVE : Accessing Dictionary Values

## Since Index is not defined, a Dictionary values can be accessed by their keys only. It means, to access dictionary elements we need to pass key, associated to the value.

## 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,'grade':'First'}

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

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

dict['Name']: Zara

dict['Age']: 7

data1={'Id':100, 'name':'Suresh', 'profession':'Developer'}

data2={'Id':101, 'name':'Ramesh', 'profession':'Trainer'}

**print** ("Id of 1st employer is",data1['Id'])

**print** ("Id of 2nd employer is",data2['Id'] )

**print** ("Name of 1st employer:",data1['name'] )

**print** ("Profession of 2nd employer:",data2['profession'] )

Id of 1st employer is 100

Id of 2nd employer is 101

Name of 1st employer is Suresh

Profession of 2nd employer is Trainer

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

dict={'name':'Zara','age':7,'grade':'First'}

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

dict['Alice']:

Traceback (most recent call last):

File "test.py", line 4, in <module>

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

KeyError: 'Alice'

## 3. UPDATE:Updating Python Dictionary Elements

## The item i.e., key-value pair can be updated. Updating means new item can be added. The values can be modified.

data1={'Id':100, 'name':'Suresh', 'profession':*'Developer',*

*’ADDRESSES’:{‘cur\_addr’:{} ,*

*‘pem\_addr’:{} ,*

*‘office\_addr’: {}*

}

list1 = [100,’Suresh’, ‘Software Developer’,’[[102,],[],[]] ]

data2={'Id':101, 'name':'Ramesh', 'profession':'Trainer'}

data1['profession']='Manager'

data2['salary']=20000

data1['salary']=15000

**print(**data1)

**print(**data2)

{'Salary': 15000, 'Profession': 'Manager','Id': 100, 'Name': 'Suresh'}

{'Salary': 20000, 'Profession': 'Trainer', 'Id': 101, 'Name': 'Ramesh'}

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 below in the simple example –

dict={'name':'Zara','age':7,'grade':'First'}

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

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

print"dict['age']: ",dict['age']

print"dict['school']: ",dict['school']

dict['Age']: 8

dict['School']: DPS School

## 4. DELETE: Deleting Python 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.

**del** statement is used for performing deletion operation.

An item can be deleted from a dictionary using the key only

Following is a simple example –

Dict1 = {'name':'Zara','age':7,'grade':'First'}

del dict1['name'] # remove entry with key 'name'

dict1.clear() # remove all entries in dict

del dict1 # delete entire dictionary

x = 10

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

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

an exception is raised because after **del dict** dictionary does not exist any more

dict['age']:

Traceback (most recent call last):

File "test.py", line 8, in <module>

print "dict['age']: ", dict['age'];

TypeError: 'type' object is unsubscriptable

data={100:'Ram', 101:'Suraj', 102:'Alok'}

**del** data[102]

**print**(data)

**del**(data)

**print**(data)   #will show an error since dictionary is deleted.

{100: 'Ram', 101: 'Suraj'}

Traceback (most recent call last):

File "C:/Python27/dict.py", line 5, in

Print(data)

NameError: name 'data' is not defined

## 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 −

**i.** **Keys must be unique:**  More than one entry per key not allowed. Which means **no duplicate key** is allowed. When duplicate keys encountered during assignment, the last assignment wins.

For example:

Course = “Python”

Age = 10

dict={'name':'Zara','age':7,'name':‘Madhu’}

print"dict['name']: ",dict['name']

|  |  |
| --- | --- |
| Name | Madhu |
| Age | 7 |
|  |  |

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

dict['name']: Manni

**ii.** Keys must be **immutable**. Which means you can use strings, numbers or tuples as dictionary keys but something like ['key'] is not allowed.

Following is a simple example:

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

## Python Dictionary Functions and Methods

|  |  |
| --- | --- |
| **Functions** | **Description** |
| **len**(dictionary) | It returns number of items in a dictionary. |
| **cmp**(dictionary1,dictionary2) | It compares the two dictionaries. |
| **str**(dictionary) | It gives the string representation of a dictionary. |
| [**type(**variable**)**](https://www.tutorialspoint.com/python/dictionary_type.htm)  **dict()** | Returns the type of the passed variable |

### **len(dictionary) :** It returns length of the dictionary

data={100:'Ram', 101:'Suraj', 102:'Alok'}

**print**(data)

**print**(len(data))

{100: 'Ram', 101: 'Suraj', 102: 'Alok'}

3

### **cmp(dictionary1,dictionary2) :** comparison is done on the basis of key and value

If, dictionary1 == dictionary2, returns 0

      dictionary1 < dictionary2, returns -1

     dictionary1 > dictionary2, returns 1

data1={100:'Ram', 101:'Suraj', 102:'Alok'}

data2={103:'abc', 104:'xyz', 105:'mno'}

data3={'id':10, 'first':'Aman','second':'Sharma'}

data4={100:'Ram', 101:'Suraj', 102:'Alok'}

**print**(cmp(data1,data2))   **-1**

**print**(cmp(data1,data4))  **0**

**print**(cmp(data3,data2))  **1**

### **str(dictionary) :** This method returns string formation of the value.

data1={100:'Ram', 101:'Suraj', 102:'Alok'}

**print**(str(data1))

{100: 'Ram', 101: 'Suraj', 102: 'Alok'}

**Python Dictionary Functions**

|  |  |
| --- | --- |
| **Methods** | **Description** |
| 1. **keys()** | It returns all the keys element of a dictionary. |
| 1. **values()** | It returns all the values element of a dictionary. |
| 1. **items()** | It returns all the items(key-value pair) of a dictionary. |
| 1. update(dictionary2) | It is used to add items of dictionary2 to first dictionary. |
| 1. clear() | It is used to remove all items of a dictionary.  It returns an empty dictionary. |
| 1. fromkeys(sequence,value1)/ fromkeys(sequence) | It is used to create a new dictionary from the sequence where  sequence elements forms the key and all keys share  the values ?value1?.  In case value1 is not give, it set the values of keys to be none. |
| 1. copy() | It returns an ordered copy of the data. |
| 1. **has\_key(key)** | It returns a boolean value.  True in case if key is present in the dictionary ,else false. |
| 1. **get(key)** | It returns the value of the given key.  If key is not present it returns None. |

### **keys():**This method returns all the keys element of a dictionary

data1={100:'Ram', 101:'Suraj', 102:'Alok'}  data1[120] *data1.get(120,’Madhu’) data1[100]*

***print****(data1.keys())*

[100, 101, 102]

### **values():**This method returns all the values element of a dictionary

data1={100:'Ram', 101:'Suraj', 102:'Alok'}

**print**( data1.values() )

['Ram', 'Suraj', 'Alok']

### **items() :**This method returns all the items(key-value pair) of a dictionary.

data1={100:'Ram', 101:'Suraj', 102:'Alok'}

**print**(data1.items() )

[(100, 'Ram'), (101, 'Suraj'), (102, 'Alok')]

### **update(dictionary2):**This method is used to add items of dictionary2 to first dictionary

data1={100:'Ram', 101:'Suraj', 102:'Alok'}

data2={103:'Sanjay'}

data1.update(data2)

**print**(data1)

**print**(data2)

{100: 'Ram', 101: 'Suraj', 102: 'Alok', 103: 'Sanjay'}

{103: 'Sanjay'}

### **clear() :**Removes all elements of dictionary dict

data1={100:'Ram', 101:'Suraj', 102:'Alok'}

**print**(data1)

data1.clear()

**print**(data1)

{100: 'Ram', 101: 'Suraj', 102: 'Alok'}

{}

### **fromkeys(sequence)/ fromkeys(seq,value)  :**

This method is used to create a new dictionary from the sequence where sequence elements forms the key and all keys share the values ?value1?. In case value1 is not give, it set the values of keys to be none.

Create a new dictionary with keys from seq and values set to value

Sequence=('id' , 'number' , 'email')

data={}

data=data.fromkeys(sequence)

**print**(data)

data1={}

data1=data1.fromkeys(sequence,100)

**print**(data1)

{'email': None, 'id': None, 'number': None}

{'email': 100, 'id': 100, 'number': 100}

### **copy() :** This method returns an ordered copy of the data

Returns a **shallow** copy of dictionary dict

data={'id':100 , 'name':'Aakash' , 'age':23}

data1=data.copy()

**print**(data1)

{'age': 23, 'id': 100, 'name': 'Aakash'}

### **has\_key(key) :** It returns a boolean value

### True in case if key is present in the dictionary, else False

data={'id':100 , 'name':'Aakash' , 'age':23}

**print**(data.has\_key('age'))

**print**(data.has\_key('email'))

True

False

1. **get(key) (OR)** [**get(key, default=None)**](https://www.tutorialspoint.com/python/dictionary_get.htm)

### This method returns the value of the given key.

### If key is not present it returns none.

For *key* key, returns value or default if key not in dictionary

data={'id':100 , 'name':'Aakash' , 'age':23}

**print** ( data.get('age') )

**print** ( data.get('email'))

23

None

1. [dict**.setdefault(key, default=None)**](https://www.tutorialspoint.com/python/dictionary_setdefault.htm) **:**

Similar to get(), but will set dict[key]=default if *key* is not already in dict

dict1 = {'id':10,'name':'Madhu'}

dict1.setdefault('age',100)

{'id': 10, 'name': 'Madhu', 'age': 100}