Python Programming

Prepared by

Dr. Ramesh Saha

Assistant professor

Computer Science and Engineering

Indian Institute of Information Technology (IIIT), Sonepat

UNIT III Python

Lists: Values and Accessing Elements, Lists are mutable, traversing a List, Deleting elements from List, Built-in List Operators, Concatenation, Repetition, In Operator, Built-in List functions and methods Tuples and Dictionaries: Tuples, Accessing values in Tuples, Tuple Assignment, Tuples as return values, Variable-length argument tuples, Basic tuples operations, Concatenation, Repetition, in Operator, Iteration, Built-in Tuple Functions Creating a **Dictionary**, Accessing Values in a dictionary, Updating Dictionary, Deleting Elements from Dictionary, Properties of Dictionary keys, Operations in Dictionary, Built-In Dictionary Functions, Built-in Dictionary Method

Python Dictionary

❖ **WHAT IS DICTIONARY?**

A dictionary is **like a list**, but more in general. In a list, index value is an integer, while in a dictionary index value can be any other data type and are called keys. The key will be used as a string as it is easy to recall. A dictionary is an extremely useful data storage construct for storing and retrieving all key-value pairs, where each element is accessed (or indexed) by a unique key. However, dictionary keys are not in sequences and hence maintain no left-to-right order.

❖ **KEY VALYE PAIR**

We can refer to a dictionary as a mapping between a **set of indices** (which are called keys) and a set of values. Each key maps a value. The association of a key and a value is called a key-value pair.

Syntax:

```
my_dict = {'key1': 'value1','key2': 'value2','key3': 'value3'…'keyn': 'valuen'}
```

Python Dictionary

❖ DICTIONARY PROPERTY

- ✓ Curley brackets are used to represent a dictionary.
- ✓ Each pair in the dictionary is represented by a key and value separated by a colon.
- ✓ Multiple pairs are separated by v comas.
- ✓ A dictionary is an unordered collection of key value pairs.
- ✓ A dictionary has a length, specifically the number of key value pairs.
- ✓ A dictionary provides fast look-up by key.
- ✓ The keys must be immutable object types.

```
[1]  A = {1:"one", 2:"two", 3: "three"}

[2]  A

     {1: 'one', 2: 'two', 3: 'three'}
```

Python Dictionary

- CREATING A DICTIONARY – dict()

The function dict ( ) is used to create a new dictionary with no items. This function is called built-in function. We can also create dictionary using {}.

```
[4]  D = dict()

     print(D)

     {}
```

```
[10] device ={'Four': 'scanner', 'three': 'printer', 'two': 'Mouse', 'one': 'keyboard'}

[12] device

     {'Four': 'scanner', 'three': 'printer', 'two': 'Mouse', 'one': 'keyboard'}
```

Python Dictionary

- CREATING AND TRAVERSING DICTIONARAY

```
[21] def Creating_Dictionary():
       device = {'Four': 'scanner', 'three': 'printer', 'two': 'Mouse', 'one': 'keyboard'}
       for i in device:
         print(device[i])
     Creating_Dictionary()
```

```
scanner
printer
Mouse
keyboard
```

```
def create_dict():
  D = dict ()
  D['one'] = "C++"
  D['two'] = "Java"
  D['three']="Python"
  D['four']="pascal"
  for i in D:
    print(D[i])
create_dict()
```

```
C++
Java
Python
pascal
```

Python Dictionary

- **DICTIONARAY – BUILT IN METHODS**

Dictionary Method	Meaning
dict.keys()	returns list of dictionary dict's keys
dict.setdefault key, default=None	similar to get(), but will set dict[key]=default if key is not already in dict
dict.update(dict2)	adds dictionary dict2's key-values pairs to dict
dict.values()	returns list of dictionary dict's values
dict.pop()	returns list of dictionary dict's keys
dict.popitem()	similar to get(), but will set dict[key]=default if key is not already in dict

Python Dictionary

❖ **Key Points**

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

```
[1] dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'};

[9] print (dict['Name'])

    Zara

[10] print (dict['Alice'])

    ---------------------------------------------------------------------------
    KeyError                                  Traceback (most recent call last)
    <ipython-input-10-6aa6c95f2c6e> in <cell line: 1>()
    ----> 1 print (dict['Alice'])

    KeyError: 'Alice'

    SEARCH STACK OVERFLOW
```

Python Dictionary

❖ **Key Points**

- You can update a dictionary by adding a new entry or item (i.e., a key-value pair), modifying an existing entry, or deleting an existing entry
- 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.
- 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.

Python Dictionary

❖ **Key Points**

➢ There are two important points to remember about dictionary keys:

(a) 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.

```
[11] dict = {'Name': 'Zara', 'Age': 7, 'Name': 'Manni'};

[13] print (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.

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

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-15-a2591b016699> in <cell line: 1>()
----> 1 dict = {['Name']: 'Zara', 'Age': 7}

TypeError: unhashable type: 'list'
```

Python Dictionary

❖ **Key Points**

Dictionaries are like lists except that they use keys instead of numbers to look up values

```
0s   lst = list()

[23] lst.append(21)

[24] lst.append(183)

[27] print(lst)
     [21, 21, 183]

[29] lst[0] = 23

[30] print(lst)
     [23, 21, 183]
```

```
[1] ddd= dict()

[2] ddd['age'] = 21

[3] ddd['course'] = 182

[4] print(ddd)
    {'age': 21, 'course': 182}

[6] ddd['age'] = 23

[7] print(ddd)
    {'age': 23, 'course': 182}
```

Python Dictionary

- dict.clear() METHOD

```
Python 3.4.0: dict3.py - C:/Python34/dict3.py
File  Edit  Format  Run  Options  Windows  Help
def create_dict():
    birthday={
        'Newton':1642,
        'Darwin':1809,
        'Turing':1912
        }
    print("Before Clearing dictionary")
    print(birthday)
    birthday.clear()
    print("After use of clear functi e content of birthday dictionary")
    print(birthday)
create_dict()
```

clear method

```
Python 3.4.0 Shell
File  Edit  Shell  Debug  Options  Windows  Help
>>>
Before Clearing dictionary
{'Newton': 1642, 'Darwin': 1809, 'Turing': 1912}
After use of clear function the content of birthday dictionary
{}
>>>
Ln: 14 Col: 4
```

Birthday dictionary cleared

Python Dictionary

- dict.copy() METHOD

Python 3.4.0: dict3.py - C:/Python34/dict3.py

File Edit Format Run Options Windows Help

```
def create_dict():
    birthday={
        'Newton':1642,
        'Darwin':1809,
        'Turing':1912
        }
    print("Birthdy dictionary contains")
    print(birthday)
    b2=birthday.copy()
    print("After use of copy method the content of b2 dictionary")
    print(b2)
create_dict()
```

Ln: 13 Col: 0

copy method

Python 3.4.0 Shell

File Edit Shell Debug Options Windows Help

```
>>>
Birthdy dictionary contains
{'Newton': 1642, 'Darwin': 1809, 'Turing': 1912}
After use of copy method the content of b2 dictionary
{'Newton': 1642, 'Darwin': 1809, 'Turing': 1912}
>>>
```

Ln: 18 Col: 4

copy method creates b2 dictionary

Python Dictionary

- dict.get() METHOD

Python 3.4.0: dict4.py - C:/Python34/dict4.py

File Edit Format Run Options Windows Help

```
def create_dict():
    birthday={
        'Newton':1642,
        'Darwin':1809,
        'Turing':1912
        }
    print("Birthdy dictionary contains")
    print(birthday)
    b2=birthday.get('Newton')
    print("After use of get method the content of b2 dictionary")
    print(b2)
create_dict()
```

Ln: 13 Col: 0

get method

Python 3.4.0 Shell

File Edit Shell Debug Options Windows Help

```
>>>
Birthdy dictionary contains
{'Turing': 1912, 'Newton': 1642, 'Darwin': 1809}
After use of get method the content of b2 dictionary
1642
>>>
```

Ln: 10 Col: 0

Creating a b2 dictionary using get method

Python Dictionary

- Dict.items() METHOD

Python 3.4.0: dict4.py - C:/Python34/dict4.py

File Edit Format Run Options Windows Help

```
def create_dict():
    birthday={
        'Newton':1642,
        'Darwin':1809,
        'Turing':1912
        }
    print("Birthdy dictionary contains")
    print(birthday)
    b2=birthday.items()
    print("After use of items method the content of b2 dictionary")
    print(b2)
create_dict()
```

Ln: 12 Col: 13

items method

Python 3.4.0 Shell

File Edit Shell Debug Options Windows Help

```
>>>
Birthdy dictionary contains
{'Newton': 1642, 'Darwin': 1809, 'Turing': 1912}
After use of items method the content of b2 dictionary
dict_items([('Newton', 1642), ('Darwin', 1809), ('Turing', 1912)])
>>>
```

Ln: 18 Col: 4

items method returns dictionary content

Python Dictionary

- dict.keys() METHOD

keys method

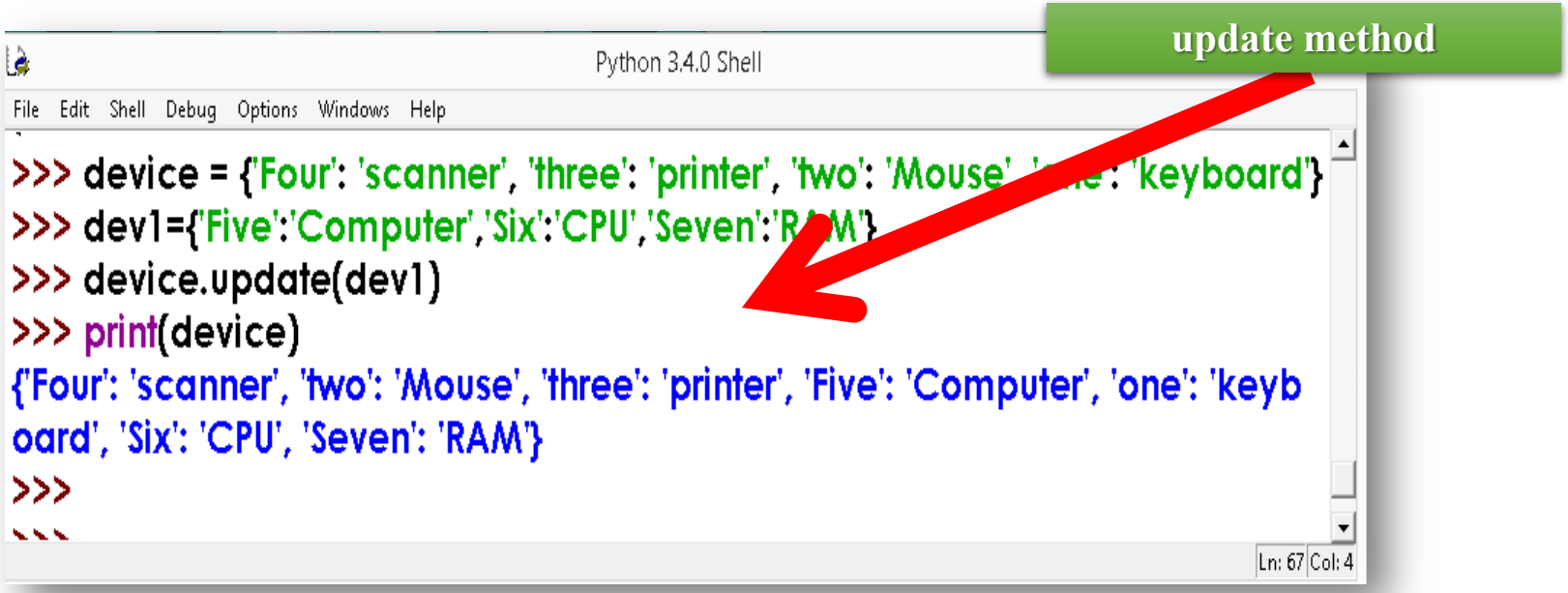
```
def create_dict():
    birthday={
        'Newton':1642,
        'Darwin':1809,
        'Turing':1912
        }
    print('dictionary keys are:',birthday.keys())
create_dict()
```

```
>>>
dictionary keys are: dict_keys(['Darwin', 'Turing', 'Newton'])
>>>
```

keys method returns dictionary keys

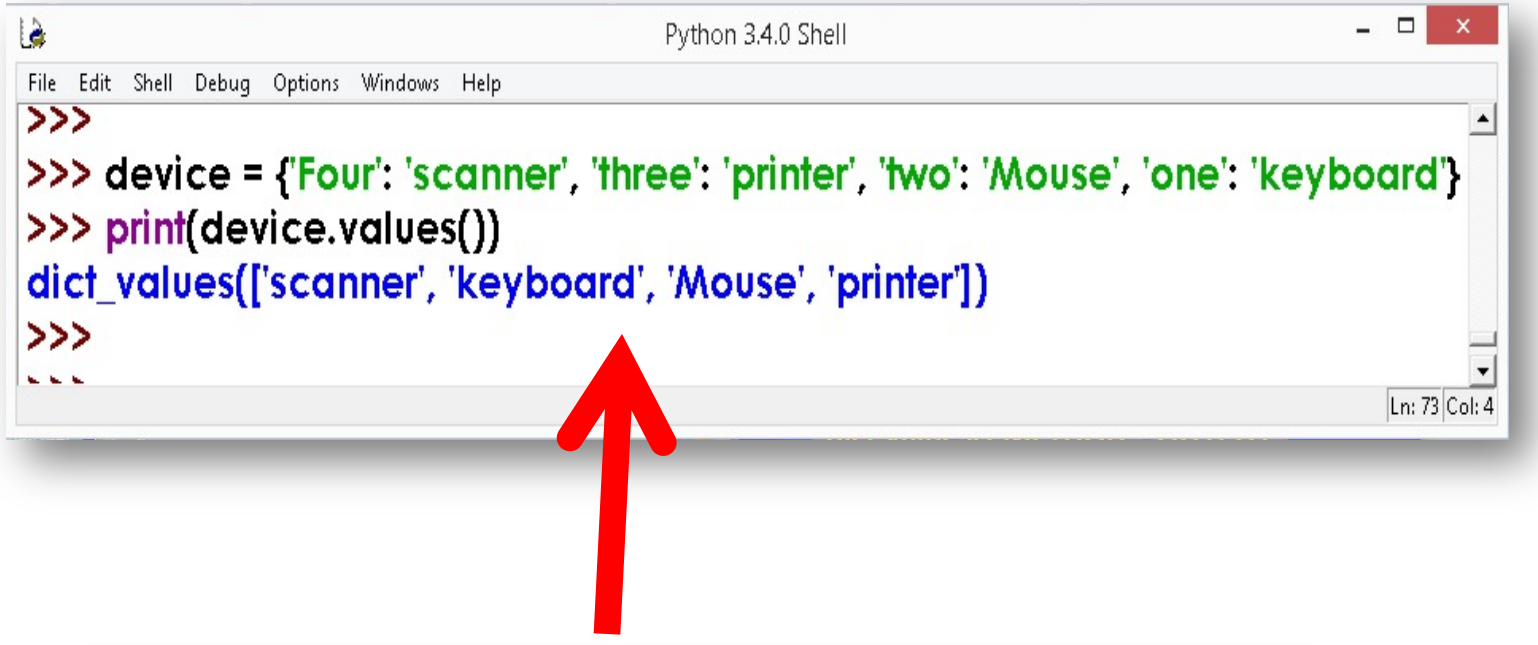
Python Dictionary

- dict.update() METHOD



Python Dictionary

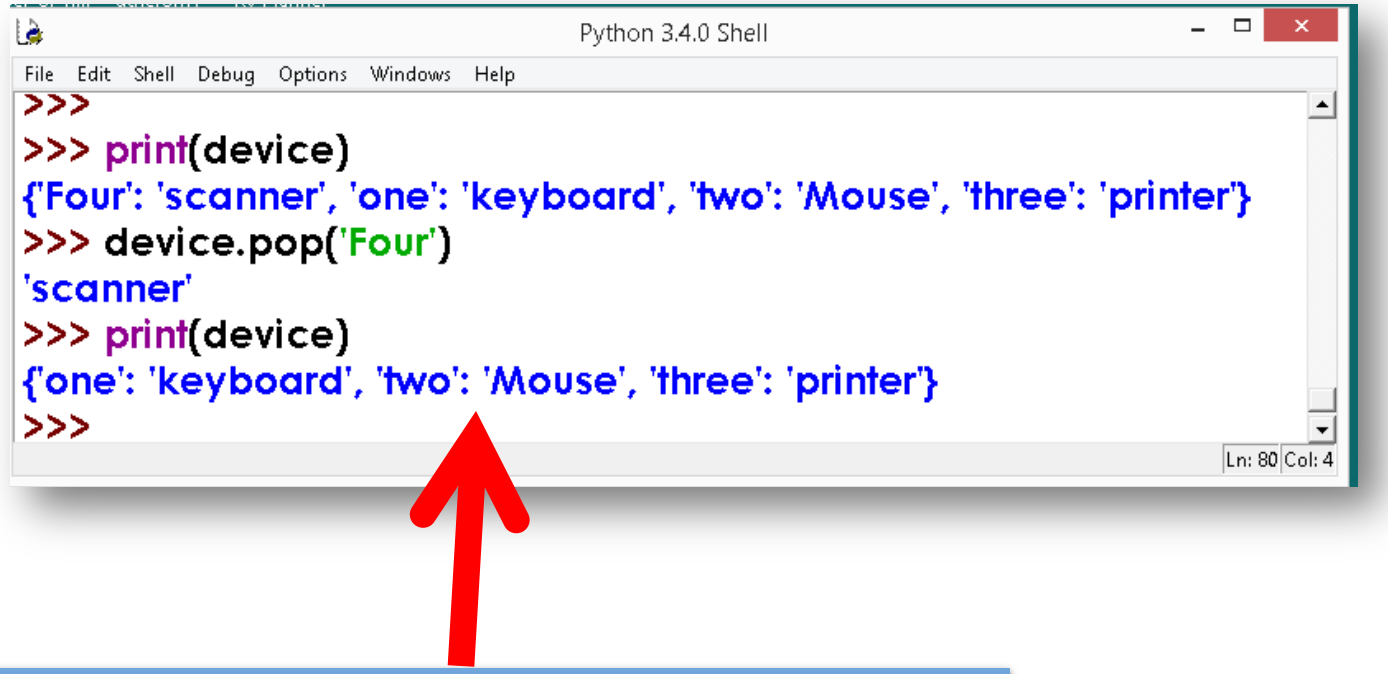
- dict.values() METHOD



values method returns dictionary values

Python Dictionary

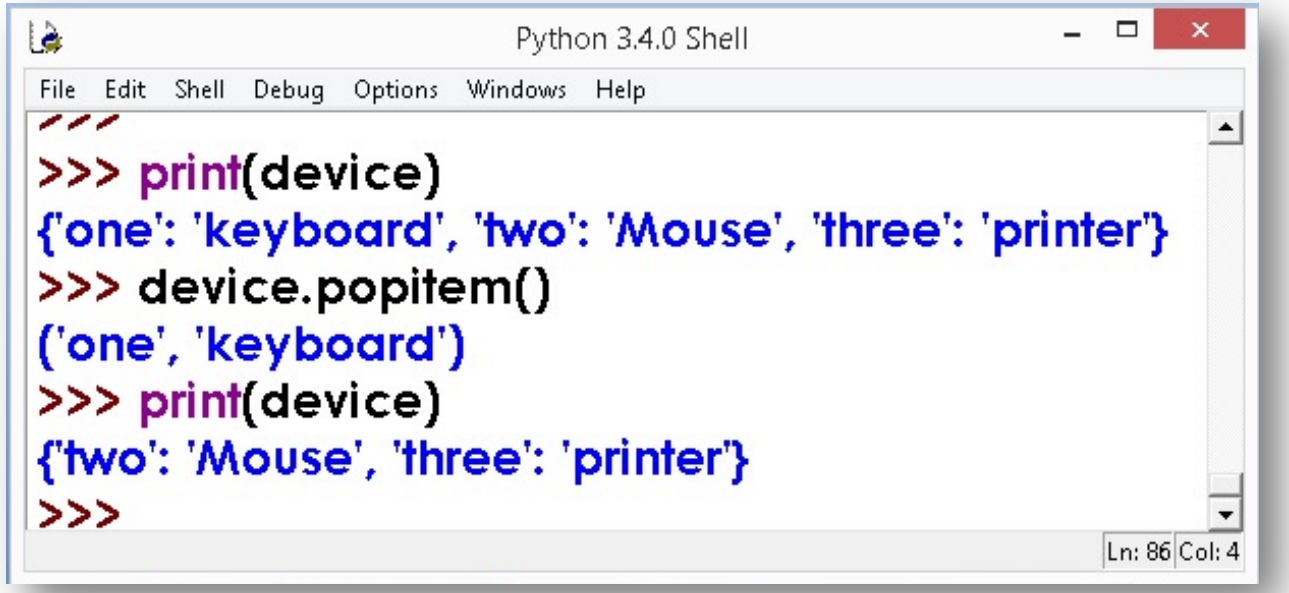
- Dct.pop() METHOD



pop method removes specified key values from the dictionary

Python Dictionary

- dict.popitem() METHOD



popitem method removes values/items from the dictionary

Python Dictionary

❖Accessing Dictionary

- Mentioning only the dictionary name without any key prints the entire dictionary.
- If the key in the square bracket is used along with the dictionary name produces the value that matches the key otherwise error is displayed.

```
[1] subjectandcode={"Physics":42,"Chemistry":43, "Mathem

[2] subjectandcode

    {'Physics': 42,
     'Chemistry': 43,
     'Mathematics': 41,
     'Biology': 44,
     'Computer Science': 83,
     'Informatics Practices': 65,
     'English': 101,
     'Hindi': 2}
```

```
[3] subjectandcode["Hindi"]

    2

[4] subjectandcode["Bengali"]

    ---------------------------------------------------------------------------
    KeyError                                  Traceback (most recent call last)
    <ipython-input-4-73f60601e3aa> in <cell line: 1>()
    ----> 1 subjectandcode["Bengali"]

    KeyError: 'Bengali'

    SEARCH STACK OVERFLOW
```

Python Dictionary

- A dictionary operation that takes a key and finds the corresponding value is called a **lookup**.
- To access a particular value of the dictionary the key is provided in a square bracket in double quote i.e., of string type.
- In Python the elements (**key: value pairs**) are unordered. It means one cannot access elements as per specific order.
- References of keys and values are stored in dictionaries.
- Dictionaries are unordered sets of elements, the printed order of elements may or may not be in the order in which we have stored them in the dictionary.

Python Dictionary

- Access Dictionary Keys : Value one by one through Loop

```
for subject in subjectandcode: print(subject,":",subjectandcode[subject])
```

```
Physics : 42
Chemistry : 43
Mathematics : 41
Biology : 44
Computer Science : 83
Informatics Practices : 65
English : 101
Hindi : 2
```

- Accessing Keys and Values

```
subjectandcode.keys()
```

```
dict_keys(['Physics', 'Chemistry', 'Mathematics', 'Biology', 'Com
Science', 'Informatics Practices', 'English', 'Hindi'])
```

```
subjectandcode.values()
```

```
dict_values([42, 43, 41, 44, 83, 65, 101, 2])
```

Python Dictionary

❖ Access Dictionary Keys

- It can be converted into the list as
- The keys of Dictionaries converted into a list can be stored in a list variable.

```
[13] Subject

     ['Physics',
      'Chemistry',
      'Mathematics',
      'Biology',
      'Computer Science',
      'Informatics Practices',
      'English',
      'Hindi']

[14] SubjectCode=list(subjectandcode.values())

     SubjectCode

     [42, 43, 41, 44, 83, 65, 101, 2]
```

```
list(subjectandcode.keys())

['Physics',
 'Chemistry',
 'Mathematics',
 'Biology',
 'Computer Science',
 'Informatics Practices',
 'English',
 'Hindi']

[11] list(subjectandcode.values())

     [42, 43, 41, 44, 83, 65, 101, 2]
```

Python Dictionary

❖Key Points

- Unlike string, list sand tuples a dictionary is not a sequence because it is unordered. The sequences are indexed by a range of ordinal numbers. Hence they are ordered but dictionaries are unordered collections.
- Dictionaries are indexed by keys. According to Python, a **key can be ” any non-mutable type”.** Since Strings and Numbers are non-mutable They can be used as keys. Tuple if containing immutable objects (integer sand strings), can be used as keys. **A value in a dictionary can be of any type and type can be mixed within one dictionary**.
- Each of the keys within a dictionary must be unique. Since keys are used to identify values in a dictionary, there cannot be duplicate keys in a dictionary. How ever two unique keys have same value.

Python Dictionary

❖Key Points

- Like lists, dictionaries are also mutable. We can change the value of a certain key “in place” using the assignment statement.
- **We can add a new key: value pair to the existing dictionary:** Using the following syntax a new Key: value pair can be added to the dictionary. {dictionary[“new”]=“a new pair is added}

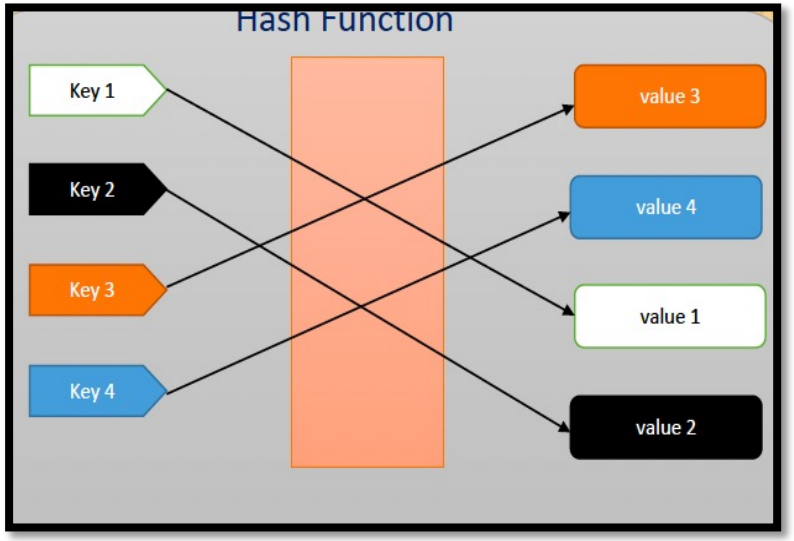
```
[8]  subjectandcode["Sanskrit"]=122

[9]  subjectandcode

     {'Physics': 42,
      'Chemistry': 43,
      'Mathematics': 41,
      'Biology': 44,
      'Computer Science': 83,
      'Informatics Practices': 65,
      'English': 101,
      'Hindi': 102,
      'Sanskrit': 122}
```

```
 'English': 101,
 'Hindi': 2}

subjectandcode["Hindi"]

2

subjectandcode["Hindi"]=102

subjectandcode

{'Physics': 42,
 'Chemistry': 43,
 'Mathematics': 41,
 'Biology': 44,
 'Computer Science': 83,
 'Informatics Practices': 65,
 'English': 101,
 'Hindi': 102}
```

Python Dictionary

❖Key Points

- Internally the key: value pairs of a dictionary are associated with one another with some internal function (called hash - function-algorithm to map and link a key with a stored value). This way of linking is called mapping.



Python Dictionary

❖Working with Dictionary

Multiple ways of creating a Dictionary : Various ways are as below

a) **Initializing a Dictionary:** In this method all the Keys: value pairs of a dictionary are written collectively, separated by commas and enclosed in curly braces.

```
[5]  Employee={ "Name":"BimlenduKumar","Department" :"ComputerScience","JoiningYear":"2007"}

[6]  Employee

     {'Name': 'BimlenduKumar',
      'Department': 'ComputerScience',
      'JoiningYear': '2007'}
```

Python Dictionary

❖Working with Dictionary

Multiple ways of creating a Dictionary : Various ways are as below

b) Adding Key: value pairs to an Empty Dictionary:

In this method first, we create an empty dictionary and then keys and values are added to it.

```
[7]  emp=dict()

[8]  emp
     {}

[9]  emp["ComputerScience"]="BimlenduKumar"

[10] emp
     {'ComputerScience': 'BimlenduKumar'}
```

Python Dictionary

❖Working with Dictionary

Multiple ways of creating a Dictionary : Various ways are as below

(c) Creating a dictionary from name and values pairs: Using the dict() constructor we can create a dictionary from any key: value pair.

Or
Specify comma-separated key: value pairs

```
[11] Emp1=dict(name="Prakash",Subject="Computer",School="HFCBarauni")

[12] Emp1

     {'name': 'Prakash', 'Subject': 'Computer', 'School': 'HFCBarauni'}
```


```
[11] Emp1=dict(name="Prakash",Subject="Computer",School="HFCBarauni")

     Emp1

     {'name': 'Prakash', 'Subject': 'Computer', 'School': 'HFCBarauni'}
```

Python Dictionary

❖Working with Dictionary

Multiple ways of creating a Dictionary : Various ways are as below

(d) Specify Keys Separately and corresponding values separately :

```
[15] EMP4=dict(zip(("name","Subject","School"),("RKTiwari","IP","KVKishanganj")))

[16] EMP4

     {'name': 'RKTiwari', 'Subject': 'IP', 'School': 'KVKishanganj'}
```

zip() function clubs the first key with the first value, the second key with the second value and soon.

Python Dictionary

❖Working with Dictionary

Multiple ways of creating a Dictionary : Various ways are as below

(**e)Specify Keys**: value pairs Separately in the form of sequence:

Using List

```
[17] Emp=dict([["name","Jaykank"],["Subject","ComputerScience"],["School","KVDanapurCantt"]])

[18] Emp

     {'name': 'Jaykank', 'Subject': 'ComputerScience', 'School': 'KVDanapurCantt'}
```

Using Tuple

```
[19] Emp1=dict((("name","SrwariBegum"),("Subject","ComputerScience"),("School","KVBaileyRoad")))

[20] Emp1

     {'name': 'SrwariBegum', 'Subject': 'ComputerScience', 'School': 'KVBaileyRoad'}
```

Python Dictionary

❖Adding elements to the Dictionary

- We can add new elements (kye: value) to a dictionary using the assignment as per the syntax given below:

<dictionary> [<key>] = <value>

❖Updating elements to the Dictionary

We can update the value of an existing element (kye: value) to a dictionary using the assignment as per the syntax given below:

<dictionary>[<existingkey>]=<newvalue>

Python Dictionary

❖Deleting elements to the Dictionary

We can delete element from a dictionary using following two ways
(a)Del <dictionary> [<key>]
(b)<dictionary> .pop(<key>)

❖Updating elements to the Dictionary

We can update the value of an existing element (kye: value) to a dictionary using the assignment as per the syntax given below:

<dictionary>[<existing key>]=<new value>

Python Dictionary

❖ Checking for existing of a key in dictionary

Usual Membership operator in and not in work with dictionary as well to check for the presence / absence of a key in the dictionary.

```
[4]  emp={'age':25,"Salary":10000,"name":"sanjay"}

     "age" in emp

     True
```

Python Dictionary

- Checking for existing of a key in dictionary

But if we want to check whether a value is present in a dictionary (called reverse lookup) we need to write proper code

```
dict1={0:"Zero",1:"One",2:"Two",3:"Three",4:"Four",5:"Five"}
ans='y'
while ans == 'y' or ans=='Y':
  val=input("Enter Value:")
  print("Value",val,end="")
  for k in dict1:
   if dict1[k]==val:
      print("Exists at",k)
      break;
   else:
      print("Not Found")
      ans=input("Want to check another value:(y/n)?:-")
```

```
Enter Value:Two
Value TwoNot Found
```