CHAPTER -9
PYTHON DICTIONARY

Bimlendu Kumar
PGT Computer Sc.
Kendriya Vidyalaya Garhara

INTRODUCTION

- **Dictionary** are mutable, unordered collection with elements in the form of a **key:value** pairs that associate keys to values.
- Rather than index associated with lists, tuples and strings, a dictionary has key associated with values.
- It is just like English Dictionary where meaning is associated with a word (Key).
- Dictionaries are containers that associate keys to values.
- With list, tuples and strings we need to know the index of individual element to access but with Dictionaries user need to search value based on key.

CREATING AND ACCESSING LISTS

>>>Month={“January":31,"February":28,"March":31,"April":30,"May":31,"June":30,"July":31,"August":31,"September":30,"October":31,"November":30, "December":31}

>>> teacher={'Bimlendu':'9470010804','Prakash':'8604774180'}

Notice that:

1. Curly braces mark the beginning and end of the dictionary
2. Each entry (key:value) consists of a pair separated by a colon. The key and corresponding value is given by writing colon(:) between them.
3. The key:value pairs are separated by comma.

Internally dictionaries are indexed (i.e.) arranged on the basis of keys.

SOME MORE DICTIONARY DECLRATION

Emptydisctionary={ }

```
>>>daynumberofweek={"Monday":1, "Tuesday":2, "Wednesday":3,
    "Thursday":4,"Friday":5,"Saturday":6,"Sunday":7}
>>> subjectandcode={"Physics":42,"Chemistry":43,
    "Mathematics":41, "Biology":44,"Computer
    Science":83,"Informatics Practices":65,"English":101,"Hindi":2}
```

Note:

- Dictionaries are also called **Associative Arrays** or **mappings** or **hashes**.
- **Keys of a dictionaries must be of immutable type** such as Python string, Number, a tuple (containing only immutable entry) but list which is mutable can not be used as keys of a dictionary.

```
>>>dict2={[2,3]:"abc"}      #TypeError: Un-sharable Type 'list'
```

ACCESSING OF A DICTIONARY

Its general syntax is
dictionaryname<“key">

EXAMPLE:

```
>>> subjectandcode["Hindi"]
2
>>> subjectandcode["Computer Science"]
83
>>> subjectandcode["Informatics Practices"]
65
```

ACCESSING ENTIRE DICTIONARY

>>>dictionaryname

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

Example:

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

ACCESSING ENTIRE DICTIONARY

- A dictionary operation that takes a key and finds the corresponding value is called **lookup.**
- To access a particular value of the dictionary the key is provided in square bracket is in double quote i.e. of string type.
- In python the elements (key:value pairs) are unordered. It means one can not access elements as per specific order.
- References of keys and values are stored in dictionaries.
- Dictionaries are unordered set of elements, the printed order of elements may or may not be in the order in which we have stored them in the 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', 'Computer Science', 'Informatics Practices', 'English', 'Hindi'])
>>> subjectandcode.values()
dict_values([42, 43, 41, 44, 83, 65, 101, 2])
```

It can be converted in list as below

```
>>> list(subjectandcode.keys())
['Physics', 'Chemistry', 'Mathematics', 'Biology', 'Computer Science', 'Informatics Practices', 'English', 'Hindi']
>>> list(subjectandcode.values())
[42, 43, 41, 44, 83, 65, 101, 2]
```

Accessing Keys and Values

The keys of Dictionaries converted into list can be stored in a list variable.

```
>>> Subject=list(subjectandcode.keys())
>>>Subject
['Physics', 'Chemistry', 'Mathematics', 'Biology', 'Computer Science', 'Informatics Practices', 'English', 'Hindi']
>>> SubjectCode=list(subjectandcode.values())
>>>SubjectCode
[42, 43, 41, 44, 83, 65, 101, 2]
```

CHARACTERISTICS OF DICTIONARY

- Dictionary is mutable like list. Except this there is no similarity with list. It has following attributes:

1. **Unordered set:** Dictionary is an unordered set of **keys:value** pairs. Its value can contain references to any type of object.
2. **Not a sequence:** Unlike string,, lists and tuples **a dictionary is not a sequence** because it is unordered. The sequence are indexed by a range or ordinal numbers. Hence they are ordered but dictionaries are unordered collection.
3. **Indexed by Keys and not Numbers:** Dictionaries are indexed by keys. According to Python, a key can be ”any non-mutable type”. Since Strings and Numbers are non mutable it can be used as keys. Tuple if containing immutable objects (integers and strings), can be used as keys. A value in a dictionary can be of any type and type can be mixed within one dictionary.

CHARACTERISTICS OF DICTIONARY

4. **Keys must be unique**: Each of the keys within a dictionary must be unique. Since keys are used to identify values in a dictionary, there can not be duplicate keys in a dictionary. However two unique keys have same value.
5. **Mutable:** Like list, dictionaries are also mutable. We can change the value of a certain key "in place" using the assignment statement as per following syntax

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

```
>>> 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}
```

CHARACTERISTICS OF DICTIONARY

6. **We can add new key:value pair to existing dictionary**:Using following syntax a new Key:value pair can be added to dictionary.

dictionary[“new”]=“a new pair is added”

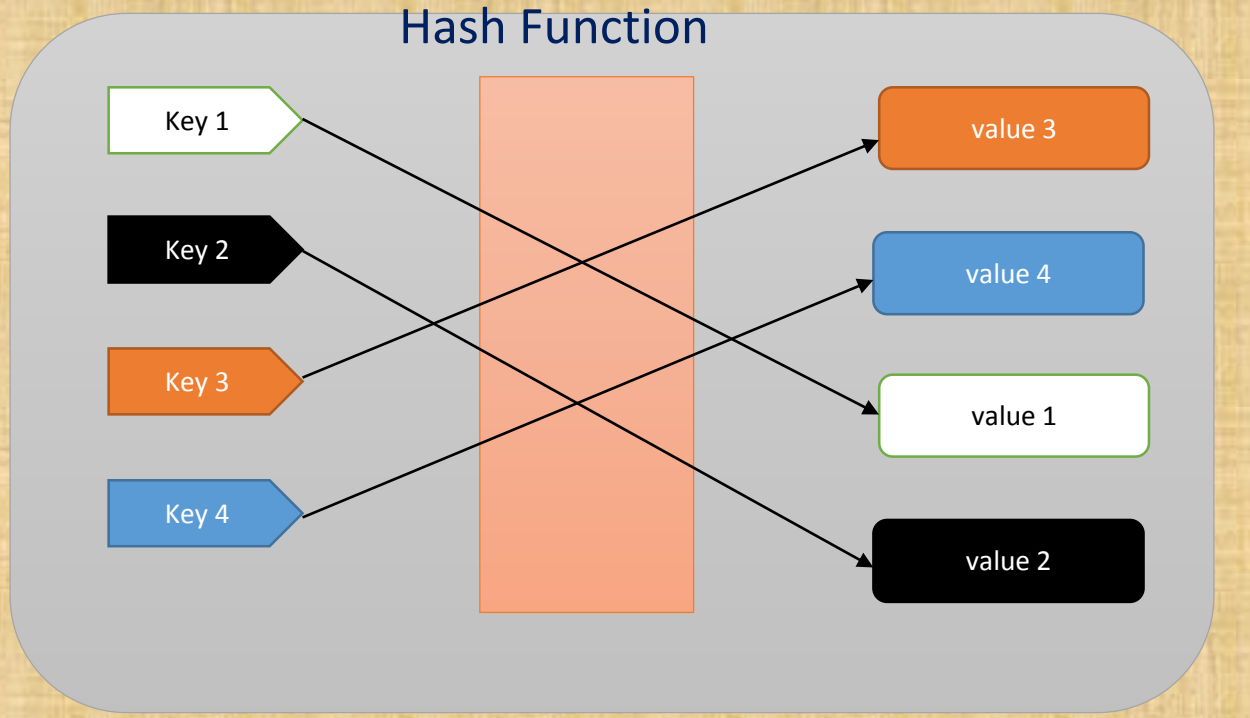
```
>>> subjectandcode["Sanskrit"]=122

>>> subjectandcode

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

CHARACTERISTICS OF DICTIONARY

7. Internally Dictionary is stored as mapping: 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.



Internally Keys are mapped with values using hash function

WORKING WITH DICTIONARY

1. **Multiple ways of creating Dictionary:** Various ways are as below:

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

   Employee={"Name":"Bimlendu Kumar", "Department": "Computer Science", "Joining Year":2007}

   (b) Adding Key:value pairs to an Empty Dictionary : In this method first we create a empty dictionary and then keys and values are added to it.

```
>>> emp=dict()              #or emp={ }
>>> emp
{}
>>> emp["Computer Science"]="Bimlendu Kumar"
>>> emp
{'Computer Science': 'Bimlendu Kumar'}
```

WORKING WITH DICTIONARY

1. **Multiple ways of creating Dictionary:** Various ways are as below:

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

```
>>> Emp1=dict(name="Prakash",Subject="Computer",School="HFC Barauni")
>>> Emp1
{'name': 'Prakash', 'Subject': 'Computer', 'School': 'HFC Barauni'}
```

or specify comma separated key:value pairs

```
>>> Emp2=dict({"name":"Rishi","Subject":"Informatics Practices","School":"KV Muzaffarpur FS"})
>>> Emp2
{'name': 'Rishi', 'Subject': 'Informatics Practices', 'School': 'KV Muzaffarpur FS'}
```

(d) Specify Keys Separately and corresponding values separately:

```
>>> EMP4=dict(zip(("name","Subject","School"),("R K Tiwari","IP","KV Kishanganj")))
>>> EMP4
{'name': 'R K Tiwari', 'Subject': 'IP', 'School': 'KV Kishanganj'}
```

zip() function clubs first key with first value, second key with second value and so on.

WORKING WITH DICTIONARY

1. **Multiple ways of creating Dictionary:** Various ways are as below:

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

Using List

```
>>> Emp=dict([ ["name","Jaykank"], ["Subject","Computer Science"], ["School","KV Danapur Cantt"]])
>>> Emp
{'name': 'Jaykank', 'Subject': 'Computer Science', 'School': 'KV Danapur Cantt'}
```

Using Tuples

```
>>> Emp1=dict((("name","Srwari Begum"),("Subject","Computer Science"),("School","KV Bailey Road")))
>>> Emp1
{'name': 'Srwari Begum', 'Subject': 'Computer Science', 'School': 'KV Bailey Road'}
```

Note: Using dict() method to create a dictionary takes longer time compared to traditional method of enclosing values in { }. Thus it should be avoided until it becomes a necessity.

ADDING ELEMENTS TO DICTIONARY

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

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

```
>>> Subject={"Computer":"083","Informatics Practices":"065"}
>>> Subject
{'Computer': '083', 'Informatics Practices': '065'}
>>> Subject["English"]="101"
>>> Subject
{'Computer': '083', 'Informatics Practices': '065', 'English': '101'}
```

UPDATING ELEMENTS OF DICTIONARY

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

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

```
>>> Subject
{'Computer': '083', 'Informatics Practices': '065', 'English': '101'}
>>> Subject["Enlsih"]="001"
>>> Subject
{'Computer': '083', 'Informatics Practices': '065', 'English': ‘001',}
```

Note: To add and to update the syntax is exactly same but in case adding the Key must be a new unique key and for updating the key must be an existing key and the value will be a new value by which the key value have to be changed.

DELETING ELEMENTS FROM DICTIONARY

We can delete element from a dictionary using following two ways

(a) del <dictionary>[<key>]

(b) <dictionary>.pop(<key>)

```
>>> Subject
{'Computer': '083', 'Informatics Practices': '065', 'English': ‘001',}
>>> del Subject["English"]
>>> Subject
{'Computer': '083', 'Informatics Practices': '065'}
>>> Subject.pop("Computer")
'083‘   # popped key value is returned by pop() function which is displayed
>>> Subject
{'Informatics Practices': '065', 'English': '001'}
>>> del Subject["IP"]
Traceback (most recent call last):
  File "<pyshell#20>", line 1, in <module>
    del Subject["IP"]
KeyError: 'IP'
```

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.

```
>>> emp={'age':25,"Salary":10000,"name":"sanjay"}
>>> "age“ in emp
True
>>> "name" in emp
True
>>> "basic" in emp
False
>>> "basic" not in emp
True
```

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

CHECKING FOR VALUE OF EXISTING KEY IN DICTIONARY

The following code can be used to check a value in the dictionary.

```
dict1={0:"Zero",1:"One",2:"Two",3:"Three",4:"Four",5:"Five"}
ans='y'
while ans=='y' or ans=='Y':
   val=input("Enter Vlue:  ")
   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)? :- ")
```

Output

```
 Enter Vlue:  Two
 Value   Two Not Found
 Want to check another value:(y/n)? :- y
 Enter Vlue:  Four
 Value   Four Exists at  4
 Want to check another value:(y/n)? :- n
```

PRETTY PRINTING A DICTIONARY

```
>>> Winners={"Nihar":3,"Rohan":5,"Madhu":3,"Roshan":1,"James":5}
>>> print(Winners)
{'Nihar': 3, 'Rohan': 5, 'Madhu': 3, 'Roshan': 1, 'James':
```

Look the output is not formatted rather it is being displayed in the same manner as it was being displayed earlier.

```
>>> import json
>>> print(json.dumps(Winners, indent=2))
{
  "Nihar": 3,
  "Rohan": 5,
  "Madhu": 3,
  "Roshan": 1,
  "James": 5
}
```

COUNTING FREQUENCY OF ELEMENTS IN A LIST USING DICTIONARY

Do the following

1. Create an empty dictionary
2. Take up an element from the list List (First element 1st time, 2nd element 2nd time and so on)
3. Check if the element exists as a key in the dictionary
   if not then add {key:value} to dictionary in the form of {List element: Countof List element in the list}

```
import json
sentence="This is a super idea \
idea will change the idea of learning"
words=sentence.split()
d={}
for one in words:
    key=one
    if key not in d:
        count = words.count(key)
        d[key]=count
print("Counting frequencies in list\n",words)
print(json.dumps(d,indent=2))
```

```
Counting frequencies in list
 ['This', 'is', 'a', 'super', 'idea', 'idea', 'will',
'change', 'the', 'idea', 'of', 'learning']
{
  "This": 1,
  "is": 1,
  "a": 1,
  "super": 1,
  "idea": 3,
  "will": 1,
  "change": 1,
  "the": 1,
  "of": 1,
  "learning": 1
}
```

DICTIONARY FUNCTIONS AND METHODS

SN	METHODS	PURPOSE	SYNTAX AND EXAMPLE

01.	len()	Returns length of dictionary	**len<dictionary>**
			Example
			>>> employee={"Name":"B Kumar","Salary":10000,"Age":45}
			>>> employee
			{'Name': 'B Kumar', 'Salary': 10000, 'Age': 45}
			>>> len(employee)
			3

02.	clear()	Removes all elements of a list	**<dictionary>.clear()**
			>>> employee={"Name":"B Kumar","Salary":10000,"Age":45}
			>>> employee
			{'Name': 'B Kumar', 'Salary': 10000, 'Age': 45}
			>>> employee.clear()
			>>> employee
			{}

clear() function removes all the elements of the dictionary but the dictinary object still remains in memory.

del command used with dictionary delete dictionary object also,

DICTIONARY FUNCTIONS AND METHODS

SN	METHODS	PURPOSE	SYNTAX AND EXAMPLE

03.	get( )	We can get the item with the given key similar to **dictionary[key]**	**<dictionary>.get()**
			Example
			>>> Subject={"Computer":"083","Informatics Practices":"065"}
			>>> Subject.get("Computer")
			'083'

04.	items()	Returns all of the items in the dictionary as a sequence	**<dictionary>.items()**
			>>> Subject={"Computer":"083","Informatics Practices":"065"}
			>>>subject
			>>> Subject
			{'Computer': '083', 'Informatics Practices': '065'}
			>>> Mylist=Subject.items()
			>>> Mylist
			dict_items([('Computer', '083'), ('Informatics Practices', '065')])

DICTIONARY FUNCTIONS AND METHODS

SN	METHODS	PURPOSE	SYNTAX AND EXAMPLE

05.	keys()	This method returns all the keys in the dictionary as a sequence (in the form of list)	**<dictionary>.keys()**
			Example
			>>> Subject={"Computer":'083',"IP":'065',"Math":'041'}
			>>> Subject
			{'Computer': '083', 'IP': '065', 'Math': '041'}
			>>> Subject.keys()
			dict_keys(['Computer', 'IP', 'Math'])

06.	values()	This method returns all the values in the dictionary as a sequence (in the form of list)	>>> Subject={"Computer":'083',"IP":'065',"Math":'041'}
			>>> Subject
			{'Computer': '083', 'IP': '065', 'Math': '041'}
			>>> Subject.values()
			dict_values(['083', '065', '041'])

DICTIONARY FUNCTIONS AND METHODS

SN	METHODS	PURPOSE	SYNTAX AND EXAMPLE

07.	update()	Merges {key:value} pairs from the new dictionary into the existing dictionary, adding or replacing as needed. The items in the new dictionary are added to the old one override any items already there with the same keys.	**dictionary_to_be_updated.update(dictionary_which_to_be_updated)**
			>>> Subject={"Computer":'083',"IP":'065',"Math":'041'}
			>>> Subject
			{'Computer': '083', 'IP': '065', 'Math': '041'}
			>>>
			Science={"Physics":"042","Chemistry":"043","Math":"041"}
			>>> Science
			{'Physics': '042', 'Chemistry': '043', 'Math': '041'}
			>>> Subject.update(Science)
			>>> Subject
			{'Computer': '083', 'IP': '065', 'Math': '041', 'Physics': '042', 'Chemistry': '043'}

Thanks for Watching This Presentation