VINOD KUMAR VERMA, PGT(CS), KV OEF KANPUR & SACHIN BHARDWAJ, PGT(CS), KV NO.1 TEZPUR

DICTIONARIES

KEY-VALUE PAIR

What is Dictionary

- It is another collection in Python but with different in way of storing and accessing. Other collection like **list, tuple, string are having an index** associated with every element but **Python Dictionary have a "key"** associated with every element. That's why python dictionaries are known as **KEY:VALUE** pairs.

- Like with English dictionary we search any word for meaning associated with it, similarly in Python we search for "key" to get its associated value rather than searching for an index.

VINOD KUMAR VERMA, PGT(CS), KV OEF KANPUR &
SACHIN BHARDWAJ, PGT(CS), KV NO.1 TEZPUR

Creating a Dictionary

Syntax to create dictionary:

dictionary_name = {key1:value,key2:value,….}

Example

>>> emp = {"empno":1,"name":"Shahrukh","fee":1500000}

Here   Keys are : "empno", "name" and "fee"

Values are: 1, "Shahrukh", 1500000

Note:

1) Dictionary elements must be between curly brackets

2) Each value must be paired with key element

3) Each key-value pair must be separated by comma(,)

Creating a dictionary

□ Dict1 = {}   # empty dictionary

□ DaysInMonth={"Jan":31,"Feb":28,"Mar":31,"Apr":31
                  "May":31,"Jun":30,"Jul":31,"Aug":31
                  "Sep":30,"Oct":31,"Nov":30,"Dec":31}

Note: Keys of dictionary must of immutable type such as:

- A python string

- A number

- A tuple(containing only immutable entries)

- If we try to give mutable type as key, python will give an error

  - **>>>dict2 = {[2,3]:"abc"}   #Error**

Accessing elements of Dictionary

☐ To access Dictionary elements we need the "key"

>>>mydict={'empno':1,'name':'Shivam','dept':'sales','salary':25000}

>>> mydict['salary']

25000

Note: if you try to access "key" which is not in the dictionary, python will raise an error

>>>mydict['comm'] #Error

Traversing a Dictionary

□ Python allows to apply "for" loop to traverse every element of dictionary based on their "key". For loop will get every key of dictionary and we can access every element based on their key.

```
mydict={'empno':1,'name':'Shivam','dept':'sales','salary':25000}
for key in mydict:
    print(key,'=',mydict[key])
```

Accessing keys and values simultaneously

>>> mydict={'empno':1,'name':'Shivam','dept':'sales','salary':25000}

>>>mydict.keys()

dict_keys(['empno', 'name', 'dept', 'salary'])

>>>mydict.values()

dict_values([1, 'Shivam', 'sales', 25000])

We can convert the sequence returned by keys() and values() by using list() as shown below:

>>> list(mydict.keys())

['empno', 'name', 'dept', 'salary']

>>> list(mydict.values())

[1, 'Shivam', 'sales', 25000]

VINOD KUMAR VERMA, PGT(CS), KV OEF KANPUR &
SACHIN BHARDWAJ, PGT(CS), KV NO.1 TEZPUR
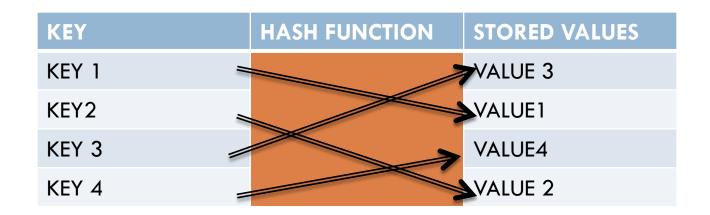
Characteristics of a Dictionary

- Unordered set
  - A dictionary is a unordered set of key:value pair

- Not a sequence
  - Unlike a string, tuple, and list, a dictionary is not a sequence because it is unordered set of elements. The sequences are indexed by a range of ordinal numbers. Hence they are ordered but a dictionary is an unordered collection

- Indexed by Keys, Not Numbers
  - Dictionaries are indexed by keys. Keys are immutable type

Characteristics of a Dictionary

- Keys must be unique
  - Each key within dictionary must be unique. However two unique keys can have same values.
  - >>> data={1:100, 2:**200**,3:300,4:**200**}
- Mutable
  - Like lists, dictionary are also mutable. We can change the value of a certain "key" in place
  - Data[3]=400
  - >>>Data
  - So, to change value of dictionary the format is :
    - DictionaryName["key" / key]=new_value
  - You can not only change but you can add new key:value pair :
    - DictionaryName["new key"]= value

Internally stored as Mappings

□ Internally, the key:value pair are associated with one another with some internal function(called hash function). This way of linking is called mapping

KEY	HASH FUNCTION	STORED VALUES
KEY 1		VALUE 3
KEY2		VALUE1
KEY 3		VALUE4
KEY 4		VALUE 2

Working with Dictionaries

- **Multiple ways of creating dictionaries**
  1. <u>**Initializing a Dictionary :**</u> in this method all the **key:value** pairs of dictionary are written collectively separated by commas and enclosed in curly braces
     Student={"roll":1,"name":"Scott","Per":90}

  2. <u>**Adding key:value pair to an empty Dictionary :**</u> in this method we first create **empty dictionary** and then **key:value** pair are added to it one pair at a time
     For example

     Alphabets={}   **#Empty dictionary**

     Or

     Alphabets = dict()

Working with Dictionaries

□ **Multiple ways of creating dictionaries**

Now we will add new pair to this empty dictionary one by one as:

Alphabets = {}

Alphabets["a"]="apple"

Alphabets["b"]="boy"

3. Creating dictionary from name and value pairs: using the dict() constructor of dictionary, you can also create dictionary initialized from specified set of keys and values. There are multiple ways to provide keys and value to dict()

(i) Specific key:value pairs as keyword argument to dict()

Student=dict(roll=1,name='scott',per=89)

Working with Dictionaries

☐ **Multiple ways of creating dictionaries**

(ii) **Specify comma-separated key:value pairs**

student = dict({'roll':1,'name':'scott','per':89})

(iii) **Specify keys separately and corresponding values separately:** in this method keys and values are enclosed separately in parenthesis and are given as arguments to the **zip()** inside **dict()**

Emp = dict(zip(('empno','name','dept'),(1,'Scott','HR')))

Working with Dictionaries

☐ **Multiple ways of creating dictionaries**

(iv) Specify key:value pairs separately in form of sequences : in this method one list of tuple argument is passed to dict(). These list or tuple contains individual key:value pair

Example:

Emp = dict(['name','Victor'],['dept','sales'])

Or

Emp = dict((('name','john'),('dept','it'),('sal',1200)))

Adding elements to Dictionary

- You can add new element to dictionary as :
  - **dictionaryName["key"] = value**

- **Nesting Dictionaries :** you can add dictionary as value inside a dictionary. This type of dictionary known as nested dictionary. For example:

   **Visitor = {'Name':'Scott','Address':{'hno':'11A/B','City':'Kanpur', 'PinCode':'208004'},
   'Name':'Peter','Address':{'hno':'11B/A','City':'Kanpur','PinCode':'208004'**

Adding elements to Dictionary

☐ To print elements of nested dictionary is as :

>>> Visitor = {'Name':'Scott','Address':{'hno':'11A/B','City':'Kanpur','PinCode' :'208004'}}

>>> Visitor

{'Name': 'Scott', 'Address': {'hno': '11A/B', 'City': 'Kanpur', 'PinCode': '2080

04'}}

>>> Visitor['Name']

'Scott'

>>> Visitor['Address']['City']   # to access nested elements

'Kanpur'

Updating elements in Dictionary

- Dictionaryname["key"]=value

>>> data={1:100, 2:200,3:300,4:200}

>>> data[3]=1500

>>> data[3]   # 1500

Deleting elements from Dictionary

del dictionaryName["Key"]

>>> D1 = {1:10,2:20,3:30,4:40}

>>> del D1[2]

>>> D1

1:10,3:20,4:40

- **If you try to remove the item whose key does not exists, the python runtime error occurs.**

- **Del D1[5]   #Error**

VINOD KUMAR VERMA, PGT(CS), KV OEF KANPUR &
SACHIN BHARDWAJ, PGT(CS), KV NO.1 TEZPUR

pop() elements from Dictionary

dictionaryName.pop(["Key"])

>>> D1 = {1:10,2:20,3:30,4:40}

>>> D1.pop(2)

1:10,3:20,4:40

Note: if key passed to pop() doesn't exists then python will raise an exception.

Pop() function allows us to customized the error message displayed by use of wrong key

VINOD KUMAR VERMA, PGT(CS), KV OEF KANPUR &
SACHIN BHARDWAJ, PGT(CS), KV NO.1 TEZPUR

pop() elements from Dictionary

>>> d1

{'a': 'apple', 'b': 'ball', 'c': 'caterpillar', 'd': 'dog'}

>>>d1.pop('a')

>>> d1.pop('d','Not found')

Not found

VINOD KUMAR VERMA, PGT(CS), KV OEF KANPUR &
SACHIN BHARDWAJ, PGT(CS), KV NO.1 TEZPUR

Checking the existence of key

□ We can check the existence of key in dictionary using "in" and "not in".

>>>alpha={"a":"apple","b":"boy","c":"cat","d":"dog"}

>>> 'a' in alpha

True

>>>'e' in alpha

False

>>>'e' not in alpha

True

Checking the existence of key

☐ If you pass "value" of dictionary to search using "in" it will return False

>>>'apple' in alpha

False

To search for a value we have to search in
dict.values()

>>>'apple' in alpha.values()

Pretty printing a Dictionary

□ We generally use print() to print the dictionary in python. For e.g.

>>>alpha={"a":"apple","b":"boy","c":"cat","d":"dog"}

>>>print(alpha)

{'a': 'apple', 'b': 'boy', 'c': 'cat', 'd': 'dog'}

To print dictionary in more readable form we use json module. i.e. **import json** and then call the function **dumps()**

Pretty printing a Dictionary

>>>alpha={"a":"apple","b":"boy","c":"cat","d":"dog"}

>>>import json

>>> print(json.dumps(alpha,indent=2))

```
{
  "a": "apple",
  "c": "cat",
  "b": "boy",
  "d": "dog"
}
```

Counting frequency of elements in a list using dictionary

- Create an empty dictionary

- Take up element from list "**listname**"

- Check if this element exists as a key in the dictionary:

  If not then add **{key:value}** to dictionary in the form

  {list-element:count of list element}

  Before we move on to this topic let us understand the function **split()**

VINOD KUMAR VERMA, PGT(CS), KV OEF KANPUR &
SACHIN BHARDWAJ, PGT(CS), KV NO.1 TEZPUR

split() function

- It is used to break up string into words and create a list out of it.

>>> message ="india is my country"

>>> message.split()

['india', 'is', 'my', 'country']

>>> mylist = message.split()

>>> mylist

['india', 'is', 'my', 'country']

Note: by default it splits the message based on the spaces between the words. However if the message to be break on any other delimeter we have to pass that delimeter

split() function

```
>>> message="ravi,vikas,dinesh,suresh"
>>> mylist = message.split(',')
>>> mylist
['ravi', 'vikas',
 'dinesh', 'suresh']
```

Program to count the frequency of list-element using a dictionary

```python
import json
sentence="Python learning is great fun \
Python is interpreted language"
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=1))
```

Dictionary functions and methods

len() : it return the length of dictionary i.e. the count of elements (key:value pairs) in dictionary

>>>alpha = {'a': 'apple', 'b': 'boy', 'c': 'cat', 'd': 'dog'}

>>> len(alpha)

4

clear() : this method removes all items from dictionary and dictionary becomes empty dictionary

>>>alpha.clear()

>>>alpha   # {}

Dictionary functions and methods

However if you use "del" to delete dictionary it will remove dictionary from memory

>>>alpha = {'a': 'apple', 'b': 'boy', 'c': 'cat', 'd': 'dog'}

>>>del alpha

>>>alpha   **#Error 'alpha' is not defined**

get() : this method is used value of given key, if key not found it raises an exception

>>>alpha.get('b')   # boy

>>>alpha.get('z')   #Error, nothing will print

Dictionary functions and methods

>>>alpha.get('z','not found')

Not found

items() : this method returns all the items in the dictionary s a sequence of (key,value) tuple

>>>alpha = {'a': 'apple', 'b': 'boy', 'c': 'cat', 'd': 'dog'}

>>> mytuple = alpha.items()

>>>for item in mytuple:

print(item)

Dictionary functions and methods

>>>alpha = {'a': 'apple', 'b': 'boy', 'c': 'cat', 'd': 'dog'}

>>> mytuple = alpha.items()

>>>for key,value in mytuple:

print(key,value)

keys() : this method return all the keys in the dictionary as a sequence of keys(not in list form)

>>> alpha.keys()

dict_keys(['a', 'b', 'c', 'd'])

Dictionary functions and methods

>>>alpha = {'a': 'apple', 'b': 'boy', 'c': 'cat', 'd': 'dog'}

values() : this method return all the values in the dictionary as a sequence of keys(a list form)

>>> alpha.values()

dict_values(['apple', 'boy', 'cat', 'dog'])

Update() method : this method merges the key:value pair from the new dictionary into original dictionary, adding or replacing as needed. The items in the new dictionary are added to the old one and override items already with the same keys.

Example of update

```
>>> d1={1:100,2:200,3:300,4:400}
>>> d2={1:111,2:222,5:555,4:444}
>>> d1.update(d2)
>>> d1
{1: 111, 2: 222, 3: 300, 4: 444, 5: 555}
>>>d2
{1: 111, 2: 222, 5: 555, 4: 444}
```

It is equivalent to:

```
for key in d2.keys():
    d1[key] = d2[key]
```

Dictionary functions and methods

fromkeys() : return new dictionary with the given set of elements as the keys of the dictionary.

```
>>> newkeys=('empcode','ename','post','sal')
>>> d1 = dict.fromkeys(newkeys)
>>> print(d1)
{'empcode': None, 'ename': None, 'post': None, 'sal': None}
```

Default value is None

```
>>> newkeys=('empcode','ename','post','sal')
>>> value='NA'
>>> d1 = dict.fromkeys(newkeys,value)
>>> print(d1)
{'empcode': 'NA', 'ename': 'NA', 'post': 'NA', 'sal': 'NA'}
```

Given value is assigned to each key

```
>>> vowels=('a','e','i','o','u')
>>> count=[1]
>>> d1 = dict.fromkeys(vowels,count)
>>> print(d1)
{'a': [1], 'e': [1], 'i': [1], 'o': [1], 'u': [1]}
>>> count.append(2)
>>> print(d1)
{'a': [1, 2], 'e': [1, 2], 'i': [1, 2], 'o': [1, 2], 'u': [1, 2]}
```

List is assigned to each key, as the list updated dictionary key values are automatically updated

Dictionary functions and methods

copy() : as the name suggest, it will create a copy of dictionary.

```
>>> d1={'empno':1,'empname':'Raj','dept':'sales','salary':8000}
>>> d2 = d1.copy()
>>> print(d2)
{'empno': 1, 'empname': 'Raj', 'dept': 'sales', 'salary': 8000}
>>> d2['salary']=10000
>>> print(d2)
{'empno': 1, 'empname': 'Raj', 'dept': 'sales', 'salary': 10000}
>>> print(d1)
{'empno': 1, 'empname': 'Raj', 'dept': 'sales', 'salary': 8000}
```

Popitem() : it will remove the last dictionary item are return key,value.

```
>>> d1={'empno':1,'empname':'Raj','dept':'sales','salary':8000}
>>> k,v = d1.popitem()
>>> print(k,v)
salary 8000
```

max() : this function return highest value in dictionary, this will work only if all the values in dictionary is of numeric type

VINOD KUMAR VERMA, PGT(CS), KV OEF KANPUR &
SACHIN BHARDWAJ, PGT(CS), KV NO.1 TEZPUR

Dictionary functions and methods

min() : this function return highest value in dictionary, this will work only if all the values in dictionary is of numeric type.

```
>>> d1={1:1000,2:2000,3:10000,4:9000}

>>> print(max(d1.values()))

10000
>>> print(min(d1.values()))

1000
```

sorted() : this function is used to sort the key or value of dictionary in either ascending or descending order. By default it will sort the keys.

```
>>> D1 = {1:4000,3:8000,5:3000,2:12000,4:7000}

>>> print(sorted(D1))

[1, 2, 3, 4, 5]
```

Sorting the keys

Dictionary functions and methods

```
>>> D1 = {1:4000,3:8000,5:3000,2:12000,4:7000}

>>> print(sorted(D1.values()))

[3000, 4000, 7000, 8000, 12000]
```

Sorting values in ascending

```
>>> D1 = {1:4000,3:8000,5:3000,2:12000,4:7000}

>>> print(sorted(D1.values(),reverse=True))

[12000, 8000, 7000, 4000, 3000]
```

Sorting values in descending

Program to count how many times characters appear in dictionary

```python
name = input("Enter any string :")
d={}
for ch in name:
        key = ch
        if key not in d:
                count = name.count(ch)
                d[key]=count

print(d)
```

```
Enter any string :aabbcccddde
{'a': 2, 'b': 2, 'c': 3, 'd': 3, 'e': 1}
```

VINOD KUMAR VERMA, PGT(CS), KV OEF KANPUR &
SACHIN BHARDWAJ, PGT(CS), KV NO.1 TEZPUR

Program to create dictionary for storing employee names and salary and access them

```python
Names =[]
Salary=[]
emp_dict = {'empname':Names,'salary':Salary}
choice=1
while choice!=0:
        print("1. Add Employee Detail ")
        print("2. Show All Employee Detail ")
        print("3. Search Employee ")
        print("0. Quit ")
        choice = int(input("Enter your choice :"))
        if choice==1:
                name = input("Enter name")
                salary = int(input("Enter Salary "))
                Names.append(name)
                Salary.append(salary)
        elif choice==2:
                print("*********** EMPLOYEE DETAILS ****************")
                print("%20s"%"NAME","%10s"%"SALARY")
                print("------------------------------------------------")
                n = emp_dict['empname']
                s = emp_dict['salary']
                for i in range(len(n)):
                        print("%20s"%n[i],"%10s"%s[i])
                print("------------------------------------------------")
```

VINOD KUMAR VERMA, PGT(CS), KV OEF KANPUR &
SACHIN BHARDWAJ, PGT(CS), KV NO.1 TEZPUR

Program to create dictionary for storing employee names and salary and access them

```python
elif choice==3:
        print("Enter name to search :")
        name = input("Enter name")
        n = emp_dict['empname']
        s = emp_dict['salary']
        try:
                pos = n.index(name)
                print("## Record Found ##")
                print("Salary :",s[pos])
        except:
                print("Name not in the record ")
```

VINOD KUMAR VERMA, PGT(CS), KV OEF KANPUR &
SACHIN BHARDWAJ, PGT(CS), KV NO.1 TEZPUR

Program to create dictionary for storing employee names and salary and access them

```
1. Add Employee Detail
2. Show All Employee Detail
3. Search Employee
0. Quit
Enter your choice :1
Enter nameAMIT
Enter Salary 9000
1. Add Employee Detail
2. Show All Employee Detail
3. Search Employee
0. Quit
Enter your choice :1
Enter nameSANJAY
Enter Salary 3000
1. Add Employee Detail
2. Show All Employee Detail
3. Search Employee
0. Quit
Enter your choice :1
Enter nameNITIN
Enter Salary 7000
```

```
1. Add Employee Detail
2. Show All Employee Detail
3. Search Employee
0. Quit
Enter your choice :2
********** EMPLOYEE DETAILS ***
                    NAME      SALARY
-------------------------------
                    AMIT        9000
                  SANJAY        3000
                   NITIN        7000
-------------------------------
1. Add Employee Detail
2. Show All Employee Detail
3. Search Employee
0. Quit
Enter your choice :3
Enter name to search :
Enter nameSANJAY
## Record Found ##
Salary : 3000
```

```
1. Add Employee Detail
2. Show All Employee Detail
3. Search Employee
0. Quit
Enter your choice :3
Enter name to search :
Enter nameSUMAN
Name not in the record
1. Add Employee Detail
2. Show All Employee Detail
3. Search Employee
0. Quit
Enter your choice :0
```

VINOD KUMAR VERMA, PGT(CS), KV OEF KANPUR &
SACHIN BHARDWAJ, PGT(CS), KV NO.1 TEZPUR