Dictionary in Python

Python Dictionaries

- Dictionaries are used to store data values in key:value pairs.
- A dictionary is a collection which is ordered*, changeable and do not allow duplicates.
- Dictionaries are also called associative arrays or mappings or hashes.
- Dictionaries are written with curly brackets, and have keys and values, each key and value are separated with colon where each item of dictionary separated with commas.

Creating a Dictionary

- Syntax:

  <dic_name>={key:value,key:value,….}

- Example 1:

  Marks={‘Hindi’:60, ‘English’:60, ‘Maths’: 90}

- Other Examples:
  - Dict1={} # Empty Dictionary
  - Dict2={‘hindi’:’Rajesh Sir’,’english’:’Amit Sir’,’maths’:’Mohit Sir’}
  - Dict3={‘even’:[4,6,7,8],’odd’:[3,5,7,9]}

Creating Dictionary using dict() Constructor

- Using the dict() constructor of dictionary, we can also create a new dictionary initialized from specified set of keys and values.
- There are multiple way to provide keys and values to dict() constructor.
- Specify Key : Value pairs as Keyword argument to dict() function, keys as argument and values as their values.

```
>>> emp=dict(name='Ravi',dep='sales',age=30)
>>> emp
{'name': 'Ravi', 'dep': 'sales', 'age': 30}
```

- keys separately and corresponding values separately.

```
>>> emp=dict(zip(('name','salary','age'),('ravi',1000,23)))
>>> emp
{'name': 'ravi', 'salary': 1000, 'age': 23}
```

Accessing Elements of a Dictionary

- A dictionary operation that takes a key and find the corresponding value, is called lookup.

- Syntax:
  - <dic_name>[<key name>]
- Example:
  - Marks={‘Hindi’:60, ‘English’:60, ‘Maths’: 90}
  - Print(Marks[‘Maths’])

Traversing a Dictionary

- Traversal of a collection means accessing and processing each element of it.
- Syntax: for <item> in <Dictionary>:
  Process each item hear
- Example:

```
>>> marks={'hindi':50,'english':89,'maths':99}
>>> for a in marks:
        print(a," : ",marks[a])


hindi  :  50
english  :  89
maths  :  99
```

Adding Element to Dictionary

- You can add new element (key: value pair) to a dictionary but the key being added must not exist in dictionary and must be unique.
- If the key already exists, then it will change (update) the value of existing key and no new entry will be added.
- Syntax: <dictionary>[<key>]= <value>
- Example:

```
>>> emp={'name':'ravi','age':23}
>>> emp
{'name': 'ravi', 'age': 23}
>>> emp['salary']=30000
>>> emp
{'name': 'ravi', 'age': 23, 'salary': 30000}
```

Update Existing Element in a Dictionary

- We can change the value of an existing key but if the key is not present the it will add a new item (key : value) in dictionary.
- Syntax: <dictionary>[<Key>]=<value>
- Example:

```
>>> emp={'name': 'ravi', 'age': 23, 'salary': 30000}
>>> emp['salary']=50000
>>> emp
{'name': 'ravi', 'age': 23, 'salary': 50000}
```

Deleting elements from a Dictionary

- There are two methods for deleting elements from a dictionary.
- Using del Command:

  Syntax: del <dictionary>[<key>]

```
>>> marks={'hindi':55,'english':80,'maths':77}
>>> del marks['english']
>>> marks
{'hindi': 55, 'maths': 77}
```

- Using pop() function:

  Syntax: <dictionary>.pop(<key>)

```
>>> marks={'hindi':55,'english':80,'maths':77}
>>> marks.pop('hindi')
55
>>> marks
{'english': 80, 'maths': 77}
```

Checking for existence of key

- The membership operator in and not in also work with dictionary, which can be used to check the existence of key in dictionary.
- Syntax: <key> in <dictionary>
  <key> not in <dictionary>
- Example:

```
>>> marks={'hindi':55,'english':80,'maths':77}
>>> 'hindi' in marks
True
>>> 'english' not in marks
False
>>> 'art' in marks
False
>>> 'art' not in marks
True
```

Dictionary Functions and Methods

- len () : it returns length of the dictionary i.e. the count of elements (key: value pair) in the dictionary.
- Syntax: len(<dictionary>)

```
>>> marks={'hindi':55,'english':80,'maths':77}
>>> len(marks)
3
```

- Clear() : it removes all items from the dictionary.
  - Syntax: <dictionary>.clear()

```
>>> marks={'hindi':55,'english':80,'maths':77}
>>> marks.clear()
>>> marks
{}
```

Dictionary Functions and Methods (Cont..)

- get() : it return the value of given key from dictionary.
- Syntax: <dictionary>.get(<key>)

```
>>> marks={'hindi':55,'english':80,'maths':77}
>>> marks.get('maths')
77
```

- pop() : it gives the value corresponding to the given key, and then removes the key: value pair from the dictionary.

```
>>> marks={'hindi':89,'english':88,'maths':88}
>>> marks.pop('hindi')
89
>>> marks
{'english': 88, 'maths': 88}
```

Dictionary Functions and Methods (Cont..)

- keys(): it returns the list of all the keys used in the dictionary.

Syntax: <dictionary>.keys( )

```
>>> marks={'hindi':44,'english':67,'maths':89}
>>> marks.keys()
dict_keys(['hindi', 'english', 'maths'])
```

- values(): it returns the list of all the values in the dictionary.

Syntax: <dictionary>.values( )

```
>>> marks={'hindi':44,'english':67,'maths':89}
>>> marks.values()
dict_values([44, 67, 89])
```

Dictionary Functions and Methods (Cont..)

- update ( ) : Two dictionaries can be merged in to one by using update ( ) method. It merges the keys and values of one dictionary into another and overwrites values of the same key.
- Syntax: Dic_name1.update (dic_name2)

```
>>> marks={'hindi':44,'english':67,'maths':89}
>>> test={'hindi':84,'maths':87,'computer':99}
>>> test.update(marks)
>>> print(test)
{'hindi': 44, 'maths': 89, 'computer': 99, 'english': 67}
```

THANK YOU

Address: G. R. Complex Preetam Nagar, Dhoomanganj
Prayagraj Uttar Pradesh 211011

Mobile No.: 9598948810,8874588766,9532878816
Website : www.infomaxacademy.com

