Learn Python: Dictionary Operations

Abrar Hussain

March 2 2015

Python dictionaries are like an unordered set of (key, value) pairs, with the requirement that the keys are unique. Dictionaries are indexed by keys, we can always change the values of the (key, value) pairs.

Basic Operations

Basics operations include inserting into a dictionary, looking up values for a dictionary, and deleting from a dictionary. We can also update the values of specific keys.

```
>>> tel = {"Mary": 4165, "Jane": 4444, "Thomas": 5343}
>>> tel["John"] = 3434 # insert a new (key, value) pair
{'Mary': 4165, 'John': 3434, 'Thomas': 5343, 'Jane': 4444}
>>> tel["John"] = 3333 # update the value of an existing key
>>> tel["John"] # lookup a specific key to get its value
3333
>>> del tel["Thomas"]
>>> tel
{'Mary': 4165, 'John': 3333, 'Jane': 4444}
```

Traversing a Dictionary

We may want to traverse over an entire dictionary. This can be useful for situations where we want to do something with the entire data set. The built-in .items() function (tuple pair of keys, values) helps with traversal.

```
>>> tel = {"Mary": 4165, "Jane": 4444, "Thomas": 5343}
>>> for k, v in tel.items():
    print(k,v)

Mary 4165
John 3333
Jane 4444
```

Built-In Functions

dict.clear()
Removes all elements of dictionary dict
copy()
Returns a shallow copy of dictionary dict
dict.get(key, Default=None)
For *key* key, returns value or default if key not in dictionary
dict.items()
Returns a list of dict's (key, value) tuple pairs
dict.kekys()
Returns a list of dictionary dict's keys
dict.values()
Returns a list of dictionary dict's values