Introduction to PYTHON

Dictionary

By: Atul Kumar Uttam

Dictionary

- Python dictionary is an **unordered** collection of items.
- A dictionary has a **key : value** pair.
- Dictionaries are optimized to **retrieve values** when the key is known.
- Dictionary are **mutable**.
- **Key** must be an **immutable object** and unique
- **Value** can be mutable/immutable object

```
>>>a = {}
>>>b = {1: 'apple', 2: 'ball'}
>>>c = {'name': 'John',   1: [2, 4, 3]}

# from sequence having each item as a pair
>>>d = dict([(1, 'apple'), (2, 'ball')])
>>>d
{1 : 'apple‘ ,  2 : 'ball‘}
```

Accessing the value

```
>>>a = {'name' : 'Jack', 'age': 26}
>>>a['name']
'Jack'

>>>a.get('age'))
26

# Trying to access keys which doesn't exist throws error
# a.get('address')
# a['address']
```

Dictionary Update

- We can add new items or change the value of existing items **using assignment operator**.

- If the key is already present,
  - value gets updated,
- else
  - a new key: value pair is added to the dictionary.

change or add elements in a dictionary

```
>>>A = {'name': 'Jack', 'age': 26}
>>>A['age'] = 27
>>>A
{'age': 27, 'name': 'Jack‘}
```

change or add elements in a dictionary

```
>>>A = {'name': 'Jack', 'age': 26}
>>>A['address'] = 'Downtown'
>>>A
{'address': 'Downtown', 'age': 26, 'name': 'Jack‘}
```

Introduction to PYTHON

Dictionary's functions

By: Atul Kumar Uttam

Dictionary Function

clear(...)

D.clear() -> None. Remove all items from D

```
>>>D = {1: 2, 2: 4, 3: 10}
>>>D.clear()
>>>D
{}
```

copy()

D.copy() -> a shallow copy of D

```
>>>D = {1: 2, 2: 4, 3: 10}
>>>A = D.copy()
>>>A
{1: 2, 2: 4, 3: 10}
>>>A is B
False
```

fromkeys(iterable, value=None**)**

Returns a new dict with keys from iterable and values equal to value.

```
>>>A = {'a', 'e', 'i', 'o', 'u' }
>>>B = dict.fromkeys(A)
>>>B
{'a': None, 'u': None, 'o': None, 'e': None, 'i': None}
```

```
>>>A = {'a', 'e', 'i', 'o', 'u' }

>>>C = dict.fromkeys(A, 10)

>>>C
{'a': 10, 'u': 10, 'o': 10, 'e': 10, 'i': 10}
```

get()

D.get(k[,d])

D[k] if k in D, else d.

d defaults to None.

```
>>>D = {1: 20, 2: 40, 3: 10}
>>>D.get(2)
40
>>> D.get(6, 1000)
1000
```

items()

D.items() -> a set-like object providing a view on D's items

```
>>>d = {1: 10, 2: 20, 3: 30}
>>>d.items()
dict_items([(1, 10), (2, 20), (3, 30)])
```

keys()

D.keys() -> a set-like object providing a view on D's keys

```
>>>d={1:2, 3:22, 4:55}
>>>d.keys()
dict_keys([1, 3, 4])
```

values()

- D.values() -> an object providing a view on D's values

```
>>>d = {1: 12, 3: 4, 5: 7, 10: 2000, 100: 200}

>>>d.values()
dict_values([12, 4, 7, 2000, 200, 400, 700])
```

pop()

D.pop(k[,d]) -> v, remove specified key and return the corresponding value.

If key is not found, d is returned if given, otherwise KeyError is raised

```
>>>B = {1:1, 2:4, 3:9, 4:16, 5:25}
>>>B.pop(4)
16
>>>B
{1: 1, 2: 4, 3: 9, 5: 25}
```

popitem()

D.popitem() -> (k, v), remove and return some (key, value) pair as a

2-tuple; but raise KeyError if D is empty.

# remove an arbitrary item

>>>B = {1: 1, 2: 4, 3: 9, 5: 25}

>>>B.popitem()

(5, 25)

Introduction to PYTHON

Dictionary’s functions

By: Atul Kumar Uttam

setdefault()

- D.setdefault(k[,d]) ->
- D.get(k,d), also set D[k]=d if k not in D

```
>>>d={1:2, 3:4, 5:7}
>>>d.setdefault(1,1000)
2
>>>d.setdefault(10,1000)
1000
>>>d
{1: 2, 3: 4, 5: 7, 10: 1000}
```

update()

```
>>>d = {1:2, 3:4, 5:7}
>>>d.update({1:12})
>>>d
 {1: 12, 3: 4, 5: 7}
```

```
>>>d = {1: 12, 3: 4, 5: 7}

>>>d.update({10:100})

>>>d
{1: 12, 3: 4, 5: 7, 10: 100}
```

```
>>>d = {1: 12, 3: 4, 5: 7, 10: 100}

>>>d.update([(100,200), (300,400), (600,700)])

>>>d
{1: 12, 3: 4, 5: 7, 10: 100, 100: 200, 300: 400, 600: 700}

>>>d.update([(10,2000), (300,400), (600,700)])

>>>d
{1: 12, 3: 4, 5: 7, 10: 2000, 100: 200, 300: 400, 600: 700}
```

Dictionary Comprehension

```
>>>a={x:x*x  for x in range(10)}


>>>a
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}
```

```
>>>a = {x:x*x for x in range(10) if x%2==0}

>>>a
{0: 0, 2: 4, 4: 16, 6: 36, 8: 64}
```