Dictionaries

References

- tutorialspoint:Python Dictionaries

Dictionaries

- In a nutshell, **dictionaries** are lists in which the elements are accessed by a string rather than index number.
    - Dictionaries are also called associative arrays, maps, or hashes.
- A dictionary **entry** is composed of a **key/value pair**.
    - The string used to access the element is called the **key**.
    - The element is called the **value**.

```
# make an empty dictionary
# use curly braces to distinguish from standard list
words = {}

# add a key / value pair to words
words[ 'python' ] = "A scary snake"

# print the whole thing
print( words )

# print one value by key
val = words['python']
print(val)
```

Safe access

- The **in** operator can be used to check if a key exists in a dictionary.
- alternatively, you could attempt the access in a **try** statement and catch the error if one occurs.
- Or you could use the **get** method of dict

```
key = 'python'
```

```python
# method 1: ask for permission
if key in words:
    val = words[key]
    print( val )
else:
    print( 'key', key, 'not found' )

# method 2: ask for forgiveness
try:
    val = words[key]
    print( val )
except KeyError:
    print( 'key',key,'not found' )

# method 3: use get method, returns None if key not present
val = words.get(key)
if val:
  print(val)
else:
  print( 'key not found' )
```

Iterating

- You can iterate using keys, values, or both.

```python
# iterate with keys, 2 ways
#for key in words.keys():
for key in words:
    val = words[ key ]
    print( key, ':', val, end=', ' )
print()

# iterate with values
for value in words.values():
    print( value, end=', ' )
print()

# iterate with keys and values
for key, value in words.items():
    print( key, ':', value, end=', ' )
print()
```

Updating

- the **update** method can be used to:
  - merge two dictionaries
  - add multiple entries at once
  - change the value for an existing key
- The **pop** method is used to remove a key/value pair.
  - pop also returns the value

```python
words = {}

# define another dict
words2 = {
    'dictionary': 'a heavy book.',
    'class': 'a group of students.'
}

# add items from one dict to another
words.update( words2 )

# add another key/item
words.update( { 'object': 'something'} )

# change a definition
words.update( { 'class': 'an echelon of society'} )

# remove a key/value pair
value = words.pop('dictionary')
```

Key sorting / random access

- to get a list of the keys, you can just cast the dict (or the keyset) to a list.
- dictionaries are unsorted
    - to get a sorted list of keys, use the **sorted** function.
- to get a random key, you need to cast the dictionary keyset to a list
    - (need to **import random)**

```python
# getting keys list
a_dict = { 'a':'apple', 'c':'cherry',
'b':'banana' }
print( a_dict )
print( list(a_dict) )
print( sorted(a_dict) )
# now you can iterate over sorted keys

# random key / value from dict
a_dict = { 'a':'apple', 'b':'banana',
'c':'cherry' }
keys = list(a_dict)
key = random.choice(keys)
val = a_dict[key]
print(val)
```

```
{'a': 'apple', 'c': 'cherry', 'b':
'banana'}
['a', 'c', 'b']
['a', 'b', 'c']
apple
```

```python
# sorting by value  (e.g. descending order frequency)
for w in sorted(d, key=d.get, reverse=True):
   print(w, d[w])
```