Think Python

10.11.2 Exercise

Dictionaries have a method called *get* that takes a key and a default value. If the key appears in the dictionary, *get* returns the corresponding value; otherwise it returns the default value. For example, here's a dictionary that maps from the letters in a string to the number of times they appear.

In [2]:
```
def value_counts(string):
    counter = {}
    for letter in string:
        if letter not in counter:
            counter[letter] = 1
        else:
            counter[letter] += 1
    return counter
```

In [3]:
```
counter = value_counts('brontosaurus')
```

If we look up a letter that appears in the word, get returns the number of times it appears.

In [4]:
```
counter.get('b', 0)
```

Out[4]:
```
1
```

If we look up a letter that doesn't appear, we get the default value, 0.

In [5]:
```
counter.get('c', 0)
```

Out[5]:
```
0
```

Use get to write a more concise version of value_counts. You should be able to eliminate the if statement.

In [8]:
```
def value_counts2(string):
    counter = {}
    for letter in string:
        counter[letter] = counter.get(letter, 0) + 1
    return counter

counter2 = value_counts2('brontosaurus')
counter2
```

Out[8]:
```
{'b': 1, 'r': 2, 'o': 2, 'n': 1, 't': 1, 's': 2, 'a': 1, 'u': 2}
```

10.11.4

Write a function called find_repeats that takes a dictionary that maps from each key to a counter, like the result from value_counts. It should loop through the dictionary and return a list of keys that have counts greater than 1. You can use the following outline to get started.

In [9]:
```
def find_repeats(counter):
    """Makes a list of keys with values greater than 1.

    counter: dictionary that maps from keys to counts

    returns: list of keys
    """
    key_list = []
    for key, value in counter.items():
        if value > 1:
            key_list.append(key)

    return key_list

find_repeats(counter)
```

Out[9]:
```
['r', 'o', 's', 'u']
```

In [ ]:
```

```