Administrivia

- PS4 is due on Wednesday (October 8)
- deadline for quiz regrade requests is 3pm next Friday (October 10)

Dictionaries

- Key, value pairs
- Keys don't have to be integers (unlike list indices)
- Keys can be integers, strings, tuples, etc. (anything immutable)
- Keys can't be of mutable types (e.g. lists and dictionaries can't be keys)
- Keys are unique, values don't have to be
- Values can be mutable

rec4.py has an example showing common/useful dictionary functions.

Python documentation: [http://docs.python.org/2/library/stdtypes.html#mapping-types-dict](http://docs.python.org/2/library/stdtypes.html#mapping-types-dict)

Quiz Question 3

(When printed, lists are enclosed in square brackets, and they contain commas. Strings inside lists are printed in quotation marks.)

```python
fruits = ['apple', 'banana']
desserts = ['pie', 'split']
L = len(fruits)
combo = fruits + desserts
print combo
['apple', 'banana', 'pie', 'split']
(We can use the plus sign to concatenate two lists together.)

for f in range(L):
    combo[f] = fruits[f] + desserts[len(desserts)-1-f]
print combo
['applesplit', 'bananapie', 'pie', 'split']
(Here, we mutate combo but reassign the values of just combo[0] and combo[1] because L = 2. The right-hand side concatenates fruits[0] + desserts[1] during the first iteration of the loop and fruits[1] + desserts[0] during the second iteration.)

meal = [fruits]
print meal
[['apple', 'banana']]  
(We put fruits inside a list, creating a list that contains a list.)
veggies = fruits
veggies[1] = 'tomato'
print fruits
['apple', 'tomato']
(This is an example of aliasing. veggies and fruits point to the same list, so mutating veggies will mutate fruits.)

veggies = fruits[:]
veggies[0] = 'melon'
print fruits
['apple', 'tomato']
(Now veggies is a copy of fruits because we can use list slicing to create a copy. So when we mutate veggies, fruits will stay the same as before.)

veggies = ['broccoli', 'spinach']
meal.append(veggies)
print meal
[['apple', 'tomato'], ['broccoli', 'spinach']]
(Remember that fruits was mutated earlier, so before the appending, meal is now ['apple', 'tomato']. The append function puts the whole veggies list into the end of meal. Make sure you know the difference between append and extend!)

meal[0][1][0] = 'p'
print meal
ERROR
[['apple', 'tomato'], ['broccoli', 'spinach']]  
(Strings are immutable, so meal[0][1][0] = 'p' will throw an error. But we asked in the instructions for you to keep executing the code after an error.)
Quiz Question 4
(The sort list function sorts the elements in alphabetical order when all of the elements are lowercase strings.)

def build_dict(words):
    word_dict = {}
    for w in words:
        key_list = []
        for c in w:
            key_list.append(c)
        key_list.sort()
        key = ''
        # turns list into string, i.e. same as key = ''.join(key_list)
        for e in key_list:
            key = key + e
        if key not in word_dict:
            word_dict[key] = [w]
            # the list that is the value at word_dict[key] won't have more
            # than one of the same word
        elif w not in word_dict[key]:
            word_dict[key].append(w)
    return word_dict

words = ['dog', 'good', 'god', 'nan', 'ann', 'god']
new_dict = build_dict(words)
(new_dict is: {'dgoo': ['good'], 'ann': ['nan', 'ann'], 'dgo': ['dog', 'god']})
keys = new_dict.keys()
keys.sort()
(keys, after it's been sorted, is: ['ann', 'dgo', 'dgoo'])
for k in keys:
    print new_dict[k]
['nan', 'ann']
['dog', 'god']
['good']
L = new_dict['ann']
(L is ['nan', 'ann'])
L.insert(1, 'dog')
print new_dict['ann']
['nan', 'dog', 'ann']
(We insert 'dog' at index 1.)
print new_dict[0]
ERROR
(This results in a KeyError because 0 is not a key in new_dict.)