Dictionaries were described in the previous handout. The focus of today is getting practice with using dictionaries. One new idea is the idea of *inverting* a dictionary.

Inverting a dictionary

Suppose we have a dictionary that maps friends to phone numbers.

```
friends_numbers = {'alice': '315-825-5555',
                   'matt': '315-824-7777',
                   'gerome': '413-835-5555'}
```

If we *invert* this dictionary, we get a dictionary that maps phone numbers to friends. In other words, the keys become values and vice versa. Here is how we might do that.

```
numbers_to_friends = {}
for friend in friends_numbers:
        number = friends_numbers[friend]
        numbers_to_friends[number] = friend
```

The resulting dictionary `numbers_to_friends` looks like this:

```
{'315-824-7777': 'matt', '315-825-5555': 'alice', '413-835-5555': 'gerome'}
```

Before inverting a dictionary, we must consider the possibility that the same value appears more than once. In this example, that would mean two different friends sharing the same phone number. That's pretty unlikely in this particular example. However, in Exercise 3 below, duplicate values are quite possible and we must handle them properly. See the solution for exercise 3 to see how we handle duplicate the values.

Exercises

Here are some exercises that will help you learn how to use dictionaries. Some solutions are presented in class and also included in the moodle version of this handout.

1. Given a list of bird sightings, write a function that returns a dictionary that maps bird names to their frequency of sightings.

   Solution:

   ```
   def new_sighting(bird_counts, sighting):
       '''(dict of {str:int}, str) -> NoneType
       Adds new sighting bird_counts dictionary.
       '''
       if sighting not in bird_counts:
   ```

```
        bird_counts[sighting] = 0
    bird_counts[sighting] = bird_counts[sighting] + 1

# recorded bird sightings
sightings = ['osprey', 'osprey', 'red-tailed hawk',
             'harrier falcon', 'osprey',
             'peregrine falcon', 'harrier falcon',
             'osprey', 'osprey', 'osprey',
             'harrier falcon', 'osprey',
             'harrier falcon', 'osprey',
             'harrier falcon', 'osprey',
             'red-tailed hawk', 'osprey',
             'osprey']

bird_counts = {}

# add the sightings to bird_counts
for sighting in sightings:
    new_sighting(bird_counts, sighting)

# print them out
for kind in bird_counts:
    print kind, bird_counts[kind]
```

2. Given a dictionary of bird sightings (key is bird kind, value is number of sightings), write a function that prints out the birds and their sighting frequency. The birds should be ordered alphabetically. Hint: there is no sort method on dictionaries but there is a sort method on lists.

Solution:

```
def print_sorted(bird_counts):
    '''(dict of {str:int}) -> NoneType
    Expects bird_counts to be a dictionary of
    bird counts (key is kind, value is count).
    Prints bird kind along with its count in
    alphabetical order.
    '''
    kinds = bird_counts.keys()
    kinds.sort()
    for kind in kinds:
        print kind, bird_counts[kind]
```

```
birds = {'peregrine falcon': 1, 'harrier falcon': 5,
         'red-tailed hawk': 2, 'osprey': 11}
print_sorted(birds)
```

3. Given a dictionary of bird sightings (key is bird kind, value is number of sightings), write a function that returns an *inverted* dictionary: the key is a number, and the value is a *list* of birds that were seen that number of times.

Solution:

```
def invert_dictionary(bird_counts):
    '''(dict of {str:int}) -> dict of {int: list of str}
    Expects bird_counts to be a dictionary of
    bird counts (key is kind, value is count).
    Returns a dictionary that maps a count to the
    list of birds having that count.
    >>> birds = {'robin': 2, 'tern':1, 'finch':2}
    >>> invert_dictionary(birds)
    {1: ['tern'], 2: ['robin', 'finch']}
    '''
    # make a new dictionary
    count_to_birds = {}

    # loop over kinds of birds (the keys of bird_counts)
    # for each kind of bird...
    for kind in bird_counts:

        # ...look up count for this kind
        count = bird_counts[kind]

        # ...if count is NOT already present in new
        #    dictionary, add the count
        if count not in count_to_birds:
            count_to_birds[count] = []

        # ...add this bird to list of birds having this
        #    count
        count_to_birds[count].append(kind)

    # return new dictionary
    return count_to_birds
```

```
birds = {'peregrine falcon': 1, 'harrier falcon': 2,
         'red-tailed hawk': 2, 'osprey': 5}
print invert_dictionary(birds)
```

4. Same as exercise 2 but print the results ordered by frequency from largest to smallest. Hint: you may need to first invert the dictionary.

Solution:

```
def invert_dictionary(bird_counts):
    count_to_birds = {}
    for kind in bird_counts:
        count = bird_counts[kind]
        if count not in count_to_birds:
            count_to_birds[count] = []
        count_to_birds[count].append(kind)
    return count_to_birds

def print_sorted2(bird_counts):
    '''(dict of {str:int}) -> NoneType
    Expects bird_counts to be a dictionary of
    bird counts (key is kind, value is count).
    Prints bird kind along with its count in
    alphabetical order.
    '''
    count_to_birds = invert_dictionary(bird_counts)
    counts = count_to_birds.keys()
    counts.sort()
    counts.reverse()
    for count in counts:
        for bird in count_to_birds[count]:
            print bird, count


birds = {'peregrine falcon': 1, 'harrier falcon': 5,
         'red-tailed hawk': 2, 'osprey': 11}


print_sorted2(birds)
```

5. Suppose two birders go out bird watching and then they want to combine their results. Write

a function that takes two bird count dictionaries and merges them. The function should return a new dictionary that contains every kind of bird that occurs in either dictionary. The count for that bird should be the total count (if the bird occurs in both input dictionaries, the new dictionary should have the sum of those counts). Keep in mind that some birds may only appear in one of the input dictionaries and not both.

Solution:

```
def merge(bird_counts1, bird_counts2):
    '''(dict of {str:int}, dict of {str:int}) -> dict of {str:int}
    Expects bird_counts1 and bird_counts2 to be
    dictionaries of bird counts (key is kind, value
    is count). Returns a dictionary that results
    from merging bird_counts1 and bird_counts2.
    >>> birds1 = {'robin': 2, 'tern':4}
    >>> birds2 = {'robin': 1, 'owl':1}
    >>> merge(birds1, birds2)
    {'tern': 4, 'robin': 3, 'owl': 1}
    '''
    # make a new dictionary
    merged_counts = {}

    # add all from first dictionary
    for kind in bird_counts1:

        # check if kind is already present in new
        # dictionary
        if kind not in merged_counts:
            merged_counts[kind] = 0

        merged_counts[kind] += bird_counts1[kind]

    # add any from second dictionary
    for kind in bird_counts2:
        if kind not in merged_counts:
            merged_counts[kind] = 0
        merged_counts[kind] += bird_counts2[kind]

    # return new dictionary
    return merged_counts
```

6. Same as previous question but the new dictionary should only contain the counts of birds that were seen by *both* birders. If a bird only appears in one of the input dictionaries, it should not be included in the final result.

Solution:

```
def merge2(bird_counts1, bird_counts2):
    '''(dict of {str:int}, dict of {str:int}) -> dict of {str:int}
    Expects bird_counts1 and bird_counts2 to be
    dictionaries of bird counts (key is kind, value
    is count). Returns a dictionary that results
    from merging bird_counts1 and bird_counts2 where
    only bird that appear in both are kept.
    >>> birds1 = {'robin': 2, 'tern':4}
    >>> birds2 = {'robin': 1, 'owl':1}
    >>> merge2(birds1, birds2)
    {'robin': 3}
    '''
    # make a new dictionary
    merged_counts = {}

    # add all from first dictionary
    for kind in bird_counts1:
        # only add if also in second dictionary
        if kind in bird_counts2:
            total = bird_counts1[kind] + bird_counts2[kind]
            merged_counts[kind] = total

    # return new dictionary
    return merged_counts
```