CS3101-1 Python, Fall 2014: Problem Set 2

Daniel Bauer

Total points: 20
Due date: Sep 23, 11:59pm EST

Submission instructions:
Put the code for each part in its own file (`part1.py` etc.).

Place the files for all problems in a directory named `[your_uni]_week[X]`, where X is the number of the problem set. For instance if your uni is xy1234 and you are submitting the problem set for the first week, the directory should be called `xy1234_week1`. Either zip or tar and gzip the directory (using `tar -c xy1234_week1 | gzip > xy1234_week1.tgz`) and upload it to the directory for problem set X on the Courseworks page for this class.

Please pay attention to the general guidelines/homework policy on the course website.

Part0 (0pt) - Files

Read the lecture notes on reading and writing files (slide 21 to 27, as marked on the slides). You will need to read from a file for Part 3.

Part 1 (4) - List Comprehensions

In the following list, every tuple describes a part-time employee in the format (`name, hours_worked_this_week, hourly_wage`).

```
employees = [('Bob',40,18.25), ('Mary',10,20.00), ('John',0,100.90), \
             ('Carl',19,17.21),('Meg', 60, 22.10)]
```

Write **a single list comprehension** that produces a list of tuples in the format (`name, total_pay`), where `total_pay` is the product of hours worked and hourly wage, but only includes employees who worked this week.

E.g. for the list above the result should be

```
[('Bob', 730.0), ('Mary', 200.0), ('Carl', 326.99), ('Meg', 1326.0)]
```

Part 2 (4) - Dictionaries

Assume we have the following dictionary:

```
    fruit_to_color = {'banana':'yellow',
                      'blueberry':'blue',
                      'cherry':'red',
                      'lemon':'yellow',
                      'kiwi':'green',
                      'strawberry':'red',
                      'tomato':'red'}
```

Write a program that creates a dictionary that maps colors to lists of fruits that have this color, e.g. `color_to_fruits['yellow']` should produce `['banana','lemon']`.

Hint: Before you can add an item to a list it has to be initialize. This has to be done the first time you add any fruit of a color (i.e. when `color_to_fruits` does not yet have a key for this color).

Do **not** use any external modules (such as the `collections` module which contains the class `defaultdict`).

Part 3 (12 points) - Functions, Files, and String Processing

Download the file `markets.tsv`, which contains geographic information for over 7000 farmers markets[1] in the US. The file is in a tabular form, where each line/row lists data for a farmers market and fields are separated by a single tab (tab-separated-value format). The fields are (in this order): state, market name, street address, city, zip code, longitude, latitude.

Your task is to write a tool that allows users to search for farmers markets in their town or zip code.

(a) Write a function that, given a filename, opens the file in the format described and reads in the data. Each farmers market should be represented as a tuple of strings. The function should return two objects: A dictionary mapping zip codes to lists of such tuples and a dictionary mapping towns to sets of zip codes. Note that you can use `return a, b` to return two values and `result_a, result_b = function(args)` to capture the return values when you call the function.

(b) Using Python string formatting, write a function that, given the parameters `state, name, address, town, zip`, returns a formatted string. For example:

```
Columbia University Greenmarket
E. Side of Broadway between 114th & 115th Streets
New York, New York 10027
```

Do not simply use '+' to concatenate strings, but string formatting as described in class [2].

(c) Write a program that first reads in the data file once (using the function from part (a)), and then asks the user repeatedly to enter a zip code or a town name (in a while loop until the user types "quit"). For each request, the program prints all farmers markets for this town or zip code (using the function from part (b)). If town names are ambiguous (because the town name exists in multiple US states), all entries should be printed.

[1] Source: data.gov, `http://explore.data.gov/d/wfna-38ey`

[2] and in PEP3101, `http://www.python.org/dev/peps/pep-3101/`