ECE 364 Prelab Assignment 04

Python Collections II

Passing this lab will satisfy course objectives CO2, CO3, CO7

Instructions

- You must meet all *base requirements* in the syllabus to receive any credit.
- Work in your Prelab04 directory, and copy all files from ∼ee364/DataFolder/Prelab04 into your working directory:

```
cp -r ∼ee364/DataFolder/Prelab04/* ./
```

- Remember to add and commit all **required** files to SVN. **We will grade the version of the file that is in SVN!**
- Do *not* add any file that is *not* required. **You will lose points if your repository contains more files than required!**
- Make sure you file compiles. **You will not receive any credit if your file does not compile.**
- Name and spell the file, and the functions, exactly as instructed. Your scripts will be graded by an automated process. **You will lose some points, per our discretion, for any function that does not match the expected name.**
- Make sure your output from all functions match the given examples. **You will not receive points for a question whose output mismatches the expected result.**
- Unless otherwise specified, you cannot use any external library, but you can use any module in the **Python Standard Library** to solve this lab, i.e. anything under:

```
https://docs.python.org/3.7/library/index.html
```

- Make sure you are using Python 3.7 for your Prelab.

Python Collections

Required File `collection2Tasks.py`

Description

Create a Python file named `collection2Tasks.py`, and do all of your work in that file. This is the only file you need to submit. You can write any number of additional helper functions in this file, but **DO NOT CREATE ANY MODULE VARIABLES**. Refer to the **Python File Structure** section below for a reminder on the requirements. Note that the use of type annotation is *not* required, but it is recommended for better compile-time static analysis.

You are given some files in a sub-folder called `"maps"`, indicating that these files contain mapping information between two collections. These files are:

- `"technicians.dat"` containing technician names and their IDs. Note that, throughout this assignment, the format of the name should match what is used in this file.
- `"viruses.dat"` containing a list of virus names, their IDs, and the price of each virus unit.

In addition, you are given another sub-folder called `"reports"` containing a set of activity reports. Each report has been created by a lab technician, whose ID is in the first line of the report. The report contains a log of experiments, or trials, run by that technician. In each experiment, the tech logs the specific virus investigated and how many units were used. The format of each file name in the folder is `"report_<reportID>.dat"`.

Note the following:

- Some technicians in `"technicians.dat"` may have no reports present.
- Some viruses in `"viruses.dat"` may have not been used.
- Each report is generated by a single technician, but he/she could have submitted multiple reports.
- While each experiment is conducted on a single virus, a report can have several experiments on the same virus.

Requirements

1. [**15 pts**] Write a function called `getTechWork(techName)` that takes in a technician name and analyzes the work of that technician over the viruses. This function returns a `{string: int}` dictionary, where the key is the virus name, and the value is the total number of units of that virus that were used across all experiments by the technician. (Include only the viruses that were used by that technician.)
2. [**15 pts**] Write a function called `getStrainConsumption(virusName)` that takes in a virus name and analyzes the utilization of that virus by all technicians. This function returns a `{string: int}` dictionary, where the key is the technician name, and the value is the total number of units of that virus that were used across all experiments by that technician. (Include only the technicians that used that virus.)
3. [**15 pts**] Write a function called `getTechSpending()` that returns a `{string: float}` dictionary, where the key is the technician name, and the value is the total amount of money spent by that technician over all experiments, rounded to two decimals. (Include only the technicians that have performed experiments.)
4. [**15 pts**] Write a function called `getStrainCost()` that returns a `{string: float}` dictionary, where the key is the virus name, and the value is the total cost of using that virus by all technicians in all experiments, rounded to two decimals. (Include only the viruses used in any experiment.)

5. [**10 pts**] Write a function called `getAbsentTechs()` that returns a set of names of all technicians who have not done any experiment. Note that the result can be an empty set.

6. [**10 pts**] Write a function called `getUnusedStrains()` that returns a set of names of all the viruses that have not been used in any experiment. Note that the result can be an empty set.

Final Check!

You are given a file that checks for the exact spelling of the required functions. Make sure you run this file before your final commit. Remember that **you will lose some points for any function that does not match the expected name.**

Python File Structure

The following is the expected file structure that you need to conform to:

```
######################################################
#   Author:     <Your Full Name>
#   email:      <Your Email>
#   ID:         <Your course ID, e.g. ee364j20>
#   Date:       <Start Date>
######################################################

import os     # List of module import statements
import sys    # Each one on a line


######################################################
# No Module-Level Variables or Statements!
# ONLY FUNCTIONS BEYOND THIS POINT!
######################################################

def functionName1(a: float, b: float) -> float:
    return 0.


def functionName2(c: str, d: str) -> int:
    return 1


# This block is optional and can be used for testing.
# We will NOT look into its content.
######################################################
if __name__ == "__main__":
    # Write anything here to test your code.
    z = functionName1(1, 2)
    print(z)
```