ECE 364 Lab 4 Handout

Python Collections

September 11, 2017

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

Instructions

- Work in your Lab04 directory.
- Copy all files from ∼ee364/labfiles/Lab04 into your Lab04 directory. You may use the following command:
  `cp -r ∼ee364/labfiles/Lab04/* ./`
- Remember to add and commit all files to SVN. **We will grade the version of the file that is in SVN!**
- 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. Also, 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.**
- Make sure you are using Python 3.4 for your project. In PyCharm, go to:

  File Menu → Settings → Project Interpreter

  And make sure that Python 3.4 (`/usr/local/bin/python3.4`) is selected.

Python Collections

Description

Students in some university are collaborating on several projects, where in each projects they are building multiple circuits using four component types: Resistors, Inductors, Capacitors and Transistors. You are given a file called `students.txt` containing student names and IDs. You are also given a file called `projects.txt` containing project IDs and the circuits IDs that belong to each project. Finally, you are given a folder called `Circuits` containing multiple files, where each file holds information about a specific circuit.

The format of each file name in the folder is `circuit_<circuitID>.txt` and it contains two types of information:

- The students IDs that have collaborated in building that circuit, separated by commas.
- The components used in that circuit, where the format of each component is: `<Symbol><floatValue>`, e.g. `L411.320`. The symbols used are: `R`, `L`, `C` and `T` for Resistors, Inductors, Capacitors and Transistors, respectively. The components are separated by commas as well.

While the format of each circuit file is consistent, the number of students participating in building each circuit varies, and so does the number of components used in that circuit. Moreover, some circuits may not have uses all component types, i.e. they may have not used transistors, or inductors, etc. Examine all of the files before you begin, to make sure you are comfortable with their format.

Requirements

Create a Python file named `projectAnalytics.py`, and do all of your work in that file. This file should only consists of function blocks, and, optionally, a conditional main block, (i.e. `if __name__ == "__main__":`). Do not include any loose Python statements, but you can, however, write any number of additional utility/helper functions that you might need.

1. [**10 pts**] Write a function called `getComponentCountByProject(projectID)` that takes in a project ID, and returns a tuple of the form `(int, int, int, int)` where the tuple elements are the numbers of unique components used in all circuits within that project, ordered as: resistors, inductors, capcitors and transistors. If the project ID provided does not exists, return `None`.
2. [**5 pts**] Write a function called `getComponentCountByStudent(studentName)` that takes in the full student name, in the same format used in the file, and returns a tuple of the form `(int, int, int, int)` where the tuple elements are the numbers of unique components used in all circuits where that student has participated, ordered as: resistors, inductors, capcitors and transistors. If the student has not participated in any projects, return an empty `tuple`. If the student name passed does not exist, return `None`.
3. [**10 pts**] Write a function called `getParticipationByStudent(studentName)` that takes in the full student name, in the same format used in the file, and returns a `set` of the project IDs that the student has participated in. If the student has not participated in any projects, return an empty `set`. If the student name passed does not exist, return `None`.
4. [**5 pts**] Write a function called `getParticipationByProject(projectID)` that takes in a project ID, and returns a `set` of all the full names of the studnets who have participated in this project. If the project ID provided does not exists, return `None`.
5. [**10 pts**] Write a function called `getProjectByComponent(components)` that takes in a `set` of component strings, and returns a `{string: set of string}` dictionary, where the key is the component from the input set, and the value is the `set` of the project IDs where that component has been used.

6. [**5 pts**] Write a function called `getStudentByComponent(components)` that takes in a `set` of component strings, and returns a `{string: set of string}` dictionary, where the key is the component from the input set, and the value is the `set` of all the student names who have used that component.

7. [**10 pts**] Write a function called `getComponentByStudent(studentNames)` that takes in a `set` of student full names, and returns a `{string: set of string}` dictionary, where the key is the student name from the set given, and the value is the `set` of all components used by that student in all projects.

8. [**5 pts**] Write a function called `getCommonByProject(projectID1, projectID2)` that returns a sorted unique list of all the components that have been used in both of the projects passed. If there were no common components used between these projects, return an empty list, and if either of the project IDs passed does not exist, return `None`.

9. [**5 pts**] Write a function called `getCommonByStudent(studentName1, studentName2)` that returns a sorted unique list of all the components that have been used by both of the students whose full names are passed. If there were no common components found, or if one of the students did not work on any project, return an empty list, and if either of the names passed does not exist, return `None`.

10. [**5 pts**] Write a function called `getProjectByCircuit()` that returns a `{string: list of string}` dictionary, where the key is the circuit ID, and the value is a sorted unique list of all project IDs, where that circuit is part of.

11. [**5 pts**] Write a function called `getCircuitByStudent()` that returns a `{string: list of string}` dictionary, where the key is the full student name, and the value is a sorted unique list of all circuits IDs that the student has worked on.

12. [**5 pts**] Write a function called `getCircuitByStudentPartial(studentName)` that takes in a student's first or last name, and returns a `{string: list of string}` dictionary, where the key is a full student name whose first or last name matched the input, and the value is a sorted unique list of all circuits IDs that the student has worked on. If the student name passed did not work on any project, return an empty dictionary. If the student name passed does not exist as a first or a last name among all students, return `None`.