CSCA08 Exercise 5

Due: October 29, 2017. 5:00pm
Pre-Run: October 27, 2017. 5:00pm

This week's exercise will revolve around reading files. Since you don't know the names or locations of the files, we will open/close them for you, and just pass in the `file handle` to your functions. As usual, your functions should all be in a file called `ex5.py`, that doesn't use `input`, `import` or `print`. Since you really have no way of using our `.test` files to test file I/O, you don't need to submit your testing files. But that doesn't mean you shouldn't be testing, just that we won't be evaluating your testing *this time*.

Finding Function Names

Write a function called `function_names` that takes as a parameter a file handle open for reading, and returns a list[1] of all of the function names in that file. Remember that function definitions have a very specific format: `def function_name(parameters)`. You can assume that all functions are exactly correctly formatted according to PEP-8 standards. e.g., 1 space after the `def`, no space before the `(`.

Calling `function_names` on `ex4.py` would (hopefully) return the result:

```
['insert', 'up_to_first', 'cut_list']
```

Hint: look through the `str` methods, some of them will be quite helpful, such as `startswith` or `find`

Justified

Write a function called `justified` that takes as a parameter a file handle open for reading, and returns a boolean which is true if and only if every line in that file is left-justified (the first character is something other than a space[2]). If any lines start with a space, the program should return False.

Challenge: Ensure that your code works efficiently on a very long file with one of the first lines being non-left-justified

Bonus: Section Average

Write a function called `section_average`[3] that takes two parameters: The first parameter is an open file of midterm marks (no, they're not real marks). Each line represents a single student and consists of a student number, a name, a section code and a midterm grade, all separated by whitespace. An example file has been uploaded as `ex5_grade_file.txt`. The second parameter is a section code. Return the average midterm mark for all students in that section, or return None if the section code does not appear in the marks file for any students. As before, this will be run through the auto-marker, but the marks won't count. It's just for your own enjoyment.

Hint: the `split` method might be particularly useful here.
Hint: Notice that not everyone has the same number of names, so we can't just assume that the mark and section code will be at a particular index... at least not reading from the left.

[1]The type contract for this would be `(io.TextIOWrapper) -> list of str`

[2]blank lines count as left justified

[3]As with previous bonus questions, this question will be marked for your own information, but your final mark for the exercise will be solely decided by the previous 2 questions.