CS112 Homework 6: File I/O – Groups Allowed

Due: Sun, Nov 10th, 11:59pm

As this is a **Homework**, you can read any and all materials, ask us questions, talk with other students, and learn however you best learn in order to solve the task. Just create your own solution from those experiences, and turn in your work.

The purpose of this homework is to provide additional practice with the use of files and input/output. Python can read/write both text and binary files; in this homework we will work with text files only.

Notes

- You cannot import any module. When a task has individual restrictions, note them, as we will remove points for any task that does not follow the requirements.
- A tester has been included with this assignment. **Grading is based on the fraction of passing tests.**

Turning It In

Add a comment at the top of the file that indicates your name, userID, G#, lab section, a description of your collaboration partners, as well as any other details you feel like sharing. Once you are done, run the testing script once more to make sure you didn't break things while adding these comments. If all is well, go ahead and turn in **just your .py file** you've been working on, named with our usual convention, over on BlackBoard. Please don't turn in the tester file or any other extra files.

Grading Rubric

Pass shared test cases	100 (zero points for hard-coding)
--	
TOTAL:	100

Tasks

Restrictions

- You're not allowed to make more than **one pass** reading a file
- You're not allowed to modify the input file in any way
- You're not allowed to use the **`with`** statement
- The only functions you're allowed to use are: `int`, `open`, `close`, `read`, `write`, `readline`, `range`, `len`, `append`, `next`, `split`, `strip`, `join`.
- Please **do not ask on piazza** whether you can use `readlines`, `index`, `count`, etc. 😡

In this homework we'll be working with GenBank[1] text files that look like the one in the link below. All the tasks require the parsing of the file and the extraction of some info. Familiarize yourself with the format before writing any code, check the files inside `test_inputs` directory. The GenBank format starts with several lines containing meta-data (e.g. title, journal, authors, etc.). Then, there is a series of lines that describe some features of the genetic sequence (e.g. source, gene, mRNA, etc.). Last, there is a series of lines that contain the actual sequences; each of these lines starts with a serial number followed by the sequences in chunks of 10 characters each (except the very last line which might not be complete).
https://www.ncbi.nlm.nih.gov/nuccore/AH011052.2

1. Extract title

Write a function `extract_title` that takes one parameter, the filename of a GenBank formatted file, and returns a **`string`** with the title of the dataset. The title is what follows the first occurrence of the tag "TITLE" and goes all the way before the tag "JOURNAL". Be aware that the title might extend in more than one lines; in this case you should replace in your return value the newline character with a single space character. Also, keep in mind that the string should not include any leading or trailing whitespace characters.

2. Extract organism

Write a function `extract_organism` that takes one parameter, the filename of a GenBank formatted file, and returns a **`list`** of the categories that are listed after tag "ORGANISM" in the same order as the one they're listed in the file. Be aware that the list of categories starts at the next line (the text after the tag is not a category name) and ends right before the tag "REFERENCE". The **`strings`** stored in the list should be clear of the characters **`;`** and **`.`** as well as any leading and trailing whitespace characters.

3. Extract sequences

Write a function `extract_sequences` that takes one parameter, the filename of a GenBank formatted file, and returns a Python **`dictionary`** containing all the sequences that are listed at the end of the file, starting after the tag "ORIGIN" and ending right before //). The first number in each line is a serial number and will become an **`integer`** key in the dictionary. The sequences that come after it should be put in a **`list`** of **`strings`** that will become the *value* for this *key*. Be aware that not all lines in the file contain 6 sequences.

[1] https://en.wikipedia.org/wiki/GenBank