Assignment 2: File I/O, String Parsing
Due: September 19, 2019 07:00:00 PM

Description

Megacorporation IDN wants to clean up some old data they have sitting around. As a first cut, they want you to quickly estimate how many lines of data they have that are not properly formatted. Unfortunately, the data is about 30 years old and formatted in a way some engineer made up on their first day on the job.

You must deal with these poorly made decisions and handle parsing files with silly formatting.

Tasks

For this assignment, you must:

1. Read in a schema file line by line.
    a. Discard commented lines (lines starting with the '#' character).
    b. Parse all other lines to identify the WIDTH_INTEGER (see format rules below).
        i. Sum all of the WIDTH_INTEGER values contained in the schema file to calculate a total width value LINE_WIDTH.
2. After parsing the schema file and calculating LINE_WIDTH:
    a. Read through the provided data file line by line.
        i. Calculate the number of lines that have total with equal to LINE_WIDTH (do not count the trailing newline character). These are the valid lines in the data file.
        ii. Calculate the number of lines that have a width not equal to LINE_WIDTH. These are the invalid lines in the data file.
3. Print the number of valid lines followed by the number of invalid lines to stdout. The values should be separated by a single space and there should be no trailing whitespace.

Parsing Rules

Each line in the schema file is freeform text containing two or more text tokens. Each token is separated by one or more tabs or spaces.

The only guaranteed information about each line in the schema file is that the keyword ‘width’ will appear somewhere on the line, and the very next non-whitespace token will be an integer value (this is the WIDTH_INTEGER mentioned in the Tasks section).

Program Input

The program you submit must be named ‘assignment2.py’ and it must accept two positional arguments. Your program must be able to be invoked by:

```
    python3 assignment2.py SCHEMA_FILE DATA_FILE
```

For this assignment, read in command line arguments with the sys module. Look up sys.argv examples for assistance reading in the schema and data file names.

Program Output

Your program must output two integer values separated by a single space and no trailing whitespace:

```
VALID_LINES INVALID_LINES
```

Allowable Imports

You are allowed to import the sys module in order to access the command line arguments passed to the program (for the schema and data files). No other imports are allowed. All other work must be done using Python built-in functions and data types.

Example Inputs and Outputs

Example inputs and outputs are available at: https://github.com/mtsargent/eecs-297-397-examples/tree/master/assignment-002. These can be used for testing, but keep in mind that assignments will be graded with additional test cases not covered by the examples. You should identify cases not covered by the examples and make sure you test your code with these additional cases.

Special Cases

Your code should use exception handling to gracefully handle cases when the schema file or data file does not exist. In the case of one or both files missing, your program should output a count of 0 for both valid and invalid lines.

Code Submission

Code must be uploaded to Canvas. Make sure your file is named assignment2.py. Please include your name and Case ID in a comment at the top of the file.

Grading

- 90% - Passing test cases. To receive full credit, your code must follow the input and output formats described above.

- 10% - Code style. See https://www.python.org/dev/peps/pep-0008/ for guidelines. Grading will focus on things mentioned in class. These include, but are not limited to:
  - Variable and function naming style.
  - Code readability
    - Variable/function names
    - Line length
    - Whitespace formatting
  - Comments where appropriate.
  - Code reuse (functions/constants where appropriate).
  - Closing files appropriately.

Hey! That was too easy! I want more to do.

If you are a real go-getter, make sure you first turn the assignment in as normal.

Here are some things you can try to add to the program that will not earn you extra credit, but will reward you with knowledge.

- Add in extra error handling. Possible errors:
  - ‘width’ token not found on a schema line.
  - No integer follows the ‘width’ token on a schema line.
  - Other?
- Track the individual widths from the schema file.
  - Use the widths to parse out individual values from the data file.
    - Ex. with widths [1, 2, 3] and data “ABCDEF”, the parsed data values would be [“A”, “BC”, “DEF”].
    - When parsing values, make sure leading/trailing whitespace is removed.
  - Once values are parsed, try converting the data to a more sensible format. An easier example would be to convert to a csv (either by hand or look at using the csv module). Try to make sure your csv conforms to the csv spec (for example, what do you do if the data itself has the ‘,’ character in it).

    - When writing out “converted” columns, discard the lines that were not of the correct width.
- Add a ‘type’ field to each line in the schema file (ex: type:int, type:float, or type:string). When parsing out individual values from each line in the data file, make sure that the values are the correct type. If not, discard those lines from the converted output.
- Try adding new keywords to each line in the schema (perhaps a column header or other information). Parse out this information and do something sensible with it.