Computer Science 1 — CSCI 1100
Lab 7 — Egg Scavenger Hunt

Lab Overview

This lab uses an egg scavenger hunt to review the use of files. Please download the file `lab07_files.zip` from the Submitty Course Materials tab. This includes a set of large text files named `1.txt`, `2.txt`, etc. Inside these files are hidden Python programs, 9 in total. To hunt them down, you need to recover the individual lines and write them into a program file. Then, you must execute that file within Spyder. You will have to do a bit of file parsing to get there. Collect them all!

Checkpoint 0: Ink Shedding

Earlier in the semester, we had you all participate in an active learning exercise to get to know each another. We are going to try another exercise today. First break up into tables of 3-4 students. Each of you should write a quick few sentences on something you found either confusing or striking about the course and Python so far. You are welcome to put your name on the paper, or to remain anonymous. After a minute or two, rotate the papers. You pass your sheet to your neighbor and take one from a different neighbor. Read the sheet you received and comment on it in writing. Do you agree with their comment? Can you help clarify something or is there some other aspect of the issue that you also find striking or confusing? After a few minutes swap again and continue until you get your own paper back. Read the comments. Do they help clarify anything? Are there common issues or interests? Feel free to continue your discussion during the rest of the lab. When you are done with the writing, show your commented paper to the TA or mentor to complete Checkpoint 0 and get credit. Optionally, feel free to turn your sheet in to the TA so we can learn more about the confusions and interests of the class.

Now back to our regularly scheduled lab!

Checkpoint 1: More Text Processing Practice

Write a function called `parse_line()` that takes as input a line of text such as

```
Here is some random text, like 5/4=3./12/3/4
```

and converts this into a four tuple of the form:

```
(12, 3, 4,"Here is some random text, like 5/4=3.")
```

where the last three values are converted to integers, and the remainder of the text is a string. The last values are always assumed to be separated by a slash.

You must however do some error checking. If you do not have a line that has at least three `'/'` in this given format or if the values contained in the last three slots are not integers, then your function should return `None`. Here are some example runs of this function:

```
>>>print(parse_line("Here is some random text, like 5/4=3./12/3/4"))
(12, 3, 4, 'Here is some random text, like 5/4=3.')
>>>print(parse_line("Here is some random text, like 5/4=3./12/3/4as"))
None
>>>print(parse_line("Here is some random text, like 5/4=3./12/412/a/3/4"))
None
>>>print(parse_line("     Here is some spaces 12/32/4"))
None
>>>print(parse_line("    Again some spaces\n/12/12/12"))
(12, 12, 12, '    Again some spaces\n')
```

Some advice here. You can use a limited version of `split()` for this or use more complex processing using `find()`. We recommend using `split()`, appropriate slicing operations, and perhaps `join()`. This approach will save you time. This should be a pretty quick checkpoint.

To complete Checkpoint 1: Show your code and output once you are finished.

Checkpoint 2: Parsing some files

Copy your file from checkpoint 1 to a new file called `check2.py`. We will not use the function you have just implemented in this part, but it is good to keep it around for the next checkpoint. Make sure you save the program file into the same folder as the text files given in the ZIP folder.

This checkpoint is the second step to building your egg scavenger hunt code. This is probably the most complex part of your lab. You will write a new function:

```
get_line(fname, parno, lineno)
```

which takes as input a file name and two numbers. Basically, the two numbers tell you which paragraph and which line your egg scavenger hunt lies. Paragraphs are separated by any number of blank lines, containing nothing but the new line character and spaces.

You must open the given file, skip all the paragraphs up to the requested one (`parno`), skip all the lines until the given line (`lineno`), read and return the requested line. For example, suppose a file contains the following text:

```
I love deadlines. I love the whooshing
noise they make as they go by.


I refuse to answer
that question on the
grounds that
I don't know the answer.
```

If we are looking for paragraph 2 and line 3, your function should return the string `grounds that`. You should remove any trailing whitespace such as `'\n'`, but be sure

to preserve leading whitespace. Refer back to Lecture 13 and the `rstrip()` function if you are having problems with this part.

Write a program that asks the user a file number, a paragraph number and a line number. Then, call this function to find and print the line at this given location. For example, using the files given to you, here are some example runs:

```
Please enter the file number ==> 1
Please enter the paragraph number ==> 5
Please enter the line number ==> 4
import webbrowser/2/3/3

Please enter the file number ==> 2
Please enter the paragraph number ==> 3
Please enter the line number ==> 3
webbrowser.open("https://www.youtube.com/watch?v=IPjQQ5dTsz4")/2/4/4

Please enter the file number ==> 2
Please enter the paragraph number ==> 4
Please enter the line number ==> 4
END/0/0/0

Please enter the file number ==> 7
Please enter the paragraph number ==> 35
Please enter the line number ==> 4
class draw_lines:/10/372/2
```

To accomplish this, think of using two variables: one for keeping track of paragraph number and the second one for the line number in that paragraph. Now, keep updating these numbers as you read a new line from the file within the loop. When do you increment paragraph count and when do you increment line count?

One quick hint. When you open the file in `get_line()`, use command **open(fname, encoding='utf8')**. Some of the characters in one of the files are not standard `ASCII` characters and you may get the following error:

```
Traceback (most recent call last):
Python Shell, prompt 133, line 1
File "/Users/wes/miniconda3/lib/python3.6/encodings/ascii.py", line 26, in decode
return codecs.ascii_decode(input, self.errors)[0]
builtins.UnicodeDecodeError: 'ascii' codec can't decode byte 0xe2 in position  \
    1309: ordinal not in range(128)
```

To complete Checkpoint 2, show a TA or mentor your code once you have checked it for these test cases.

Now for the egg scavenger hunt!

If we look at the above examples, we can see the following. File 1, paragraph 5, line 4 contains the line: `import webbrowser/2/3/3`

This means two things:

- The first line of the program you are reading is: `import webbrowser`
- The next line of your program is at file `2.txt`, paragraph 3 and line 3.

The line at 2/3/3/ contains the content:
`webbrowser.open("https://www.youtube.com/watch?v=IPjQQ5dTsz4")`
This line points you to the next line, which is at 2/4/4. The line at this location has the content: `END/0/0/0`
This means, you are at the end of the program. By following these clues about where the next line is, you have read a program with the following lines:

```
import webbrowser
webbrowser.open("https://www.youtube.com/watch?v=IPjQQ5dTsz4")
```

Your egg scavenger hunt starts at a given line and continues to find the next line until you reach the end. All the lines you found are part of a small Python program.
Please come to lab for the last checkpoint.