CSE 8A Programming Assignment 3

Fall 2019

If you have any questions about what this assignment is asking or about how to approach any part of this assignment, check the PA3 FAQ on Piazza.

Learning goals:

- Loop over the elements in a list
- Use conditional statements to process data from a list

Submission:

- The submission template can be found here

Part 0: Data Ingestion

Choose a data set that seems interesting to you. In this project, we will be taking a data set and analyzing it to find out something interesting about the data.

Part 0 will help you accomplish the setup tasks for this assignment. These are:

1. Find a dataset online that is represented in csv (comma separated value) format. Download this file into a folder for PA3 on your computer or lab account.
2. Create a new python file to contain your PA3 code. Copy the code on the next page into this file. Save this file in the same directory that contains the data file you downloaded.

More details of each step follow.

Finding a data set (step 1 above)

Data sets can be found in many places online, such as this website. Make sure the data set you choose is in a csv (comma separated value) file format so it is easy to ingest, and that it has at least two fields (columns). For example, here is a data set with information about baby names in the US between 1880 and 2014. The first line in the file tells you what the "fields" are: 'Id', 'Name', 'Year', 'Gender', 'Count'. Then each following line in the file is an entry that provides data for each field.

Once you have chosen a data set, we need to ingest the data and turn it into a list. We provide you with a function called **`ingest_data`**. This function takes a file name **`filename`** (which is the name of the csv file you have chosen) and a field name **`fieldname`**. The function **`ingest_data`** returns a list of strings containing the first 2,000 data entries in the given file for the given field name. For example, if we pass in 'Name' for **`fieldname`**, we will get a list of the first 2,000 names. To get meaningful results with a full set of data, try to find a file with at most

2,000 entries for this project. (But if your file is larger it's OK. You'll just only work with the first 2,000 entries)

Ingesting the data (step 2 above)

Create a new Python file in IDLE and copy and paste the code below for `ingest_data` in this file. You need to copy and paste from the "`# BEGIN PROVIDED CODE`" until "`# END PROVIDED CODE`".

```
# BEGIN PROVIDED CODE
# ingest_data: given a file name and a field name, returns a list containing
# all the data in the given file for the given field name
import csv
def ingest_data(filename, fieldname):
    file_object = open(filename, newline='')
    rows = csv.reader(file_object, delimiter=',')
    headers = next(rows)
    try:
        field_idx = headers.index(fieldname)
    except ValueError:
        print('The field name', fieldname, 'does not exist in the headers.')
        print('Here are the value field names in this file:')
        for h in headers:
            print(h)
        return
    data_list = []
    count = 0
    limit = 2000 # CHANGE LIMIT
    for line in rows:
        if (count >= limit):
            print('Too many entries, returning first', limit, 'entries.')
            return data_list
        try:
            field_value = line[field_idx]
        except IndexError:
            print('Skipping row #', count, 'because field does not exist')
            continue
        data_list.append(field_value)
        count = count + 1
    return data_list
# END PROVIDED CODE
```

Using the provided code (finishing step 2)

Let's say that your csv file is called "**`data.csv`**". **Make sure this file exists in the same directory as the Python file above which contains the code for `ingest_data`.** In IDLE run the file that contains **`ingest_data`** (press F5, or click on "Run", then "Run Module").

Now go to the IDLE command prompt and enter the following:

```
>>> ingest_data("data.csv", "Blah")
```

The above call will return the data for all the "Blah" fields in the csv file. If the **`ingest_data`** function does not find a field called "Blah", it will list all the valid field names. Let's assume that we are using the data set here with information about baby names in the US between 1880 and 2014. Here is what will happen:

```
>>> ingest_data("data.csv", "Blah")
The field name Blah does not exist in the headers.
Here are the value field names in this file:
Id
Name
Year
Gender
Count
```

This tells you that there are 5 valid field names in this file: "**`Id`**", "**`Name`**", "**`Year`**", "**`Gender`**", and "**`Count`**". Now you can try:

```
>>> data = ingest_data("data.csv", "Name")
```

The result will be:

```
>>> data = ingest_data("data.csv", "Name")
Too many entries, returning first 2000 entries.
```

The message above tells you the file has too many entries, and so only the first 2,000 entries will be returned. Now the **`data`** variable will be a list of the first 2,000 values from the "**`Name`**" field in the **`data.csv`** file. If your data file has fewer than 2,000 entries, all values will be included in the returned list.

Part 1: Data Investigation

By calling **`ingest_data`** we get a list of data that we can process. You can call **`ingest_data`** multiple times, and then get multiple lists of data. Think of a meaningful way of analyzing these lists of data. For example, if you have a csv file containing election data, you could count the number of votes for various states, and then return the name of the state with the largest number of votes for a given candidate.

Once you have found a meaningful way of analyzing the data, write a function called **`analyze_data`** that will take at least two lists of data (from two different fields from the file) and perform this analysis. The function you will write must:

- Take at least two parameters that are lists, which can be the data read from the csv file using **`ingest_data.`** You can (and are encouraged to) have more than two parameters.
- Use at least one loop
- Use at least one if statement in the loop and use at least one if-else statement, either in the loop or outside the loop. If you use an if-else statement in the loop, that is sufficient to meet both criteria for this requirement.
- Use data from both lists in the analysis
- Return a meaningful result

Then write a program that (1) reads a csv file using at least two calls to **`ingest_data`** (2) calls **`analyze_data`** to analyze the data, and (3) prints some output to show the results.

For example, let's assume we use the data set here with information about baby names in the US between 1880 and 2014. If you execute the following:

```
names = ingest_data("data.csv", "Name")
counts = ingest_data("data.csv", "Count")
```

You would get two lists: the **`names`** and **`counts`** of the first 2,000 entries in the file. You could then for example write an **`analyze_data(names, counts, n)`** which would return the top **`n`** names that appear in the data.

Part 2: Testing

You will be testing the function for the data investigation you wrote in Part 1 here. Test your function by passing it two lists and printing out the output to see if it is what you expect.
As in the previous assignments, please provide the following in your submission:

- The output of three different calls to your function with three different sets of input. Note that for these tests you are calling *only* the **`analyze_data`** function. You should *not*

take input from the user or read data from a file. You should call the function directly on hard-coded values, e.g.:

```
>>> analyze_data([3, 20, 1], ["Sally", "Adaline", "Leah"])
```
Result will be shown here

Include in your writeup the line of code that performs the call and the resulting output. Make sure you consider all the possibilities of your data when making these tests.

For each of these three calls to analyze_data these three calls, explain why they were chosen, how you know the outputs are correct, and why they illustrate the scope of your function's functionality.

- Then show the result of running your full program **once**.

  Include the program's output in your writeup. Explain what data is in the data set you chose, what the program is calculating, and explain why the answer the program produces is correct. You do not have to include the full data set.

Finally, include any bugs or issues with your program. If you think you have none, justify why.

OPTIONAL note for after you have finished your implementation and tested it: If you find that you want more than 2,000 entries (to get more meaningful results), you can change the code for `ingest_data` in the following simple way. Find the line with the comment "`# CHANGE LIMIT`" and change the 2000 inside "`limit = 2000`" to a larger number, for example: "`limit = 2000000`". Note however that this gives you MUCH larger data, which can be hard to process. In particular, you need to make sure not to print anything inside of a loop, otherwise your code will take too long to run. And you also need to make sure not to print any of the lists that you are processing, since it will take too long to display them. Finally, you might want to start increasing the limit little by little to see how your code works with larger data, instead of shooting up from 2000 to 2000000 in one shot.

Part 3: Reflection

Once you have finished and submitted your assignment, fill out the reflection form here.