Northeastern University
DS2001 – Programming with Data — Social Science Practicum
Fall 2019, Mattsson & McCabe

Practicum 7: Reading Files

In this week's practicum we will be asking Python to read in data for us, from two comma-separated files. We will then apply the functions from previous weeks to draw conclusions from these datasets.

1 Fatal Encounters with Police (US)

Following the fatal shooting of Michael Brown in Ferguson, Mo. on Aug. 9, 2014, there has been increased attention on collecting and analyzing data related to fatal police encounters in the US. You can read more about the dialogue around this issue and specific cases which have raised questions about officer conduct here: Fatal Police Shootings: Accounts Since Ferguson, New York Times
Since Jan. 1, 2015, The Washington Post has been compiling a database of every fatal shooting in the US by a police officer in the line of duty. It is difficult to find reliable data from before this period. The Washington Post is tracking details about each killing - including the race, age and gender of the deceased. They have gathered this information from law enforcement websites, local news reports, social media, independent databases, and their own reporting.
This description of the data, and the version we use, was compiled by Karolina Wullum. It is available here: Fatal Police Shootings in the US.

1.1 Download

Please download `pr07.zip` from the practicum website (or Blackboard), unzip it, and move the directory/folder to where you want it in your file system. Use Atom's `File > "Add Project Folder..."` to open this folder and view `fatal_encounters.py`, `data.py`, and `fatal_encounters.csv`.

1.2 How Python reads files

Atom shows you files in a way that is easy for *you* to read. Computers don't have eyes, so to understand the contents of a file they need some more specific information. So, let's explore how Python reads files.

Opening files Python makes it relatively straightforward to open files. To open a file named `"filename.csv"`, you would use the following syntax:

```
dataset_file = open("filename.csv", "r")
# you may need to include the full path to the file
```

Please open `"fatal_encounters.csv"` and assign this file to a variable. Then print out the following:

1. `your_file_variable`
2. `your_file_variable.read()`
3. `your_file_variable.read()` *again*

Raw text files Raw text files are stored in your computer with specific special characters that denote breaks in the text. The most important are the new line character (`\n`) and the tab character (`\t`).

1.3 Processing the first line of a file

Python's file object gives us several ways to process files that don't just turn the whole file into one long string. Please use `file.readline()` to get the first line of `"fatal_encounters.csv"`, and use string processing to turn this into a useful `header` variable. You will need to re-open the file. It's good practice to close a file when you're done with it, so let's close the file before reopening it.

```
dataset_file.close()
dataset_file = open("filename.csv", "r")
```

Strip and split The most common string funcitons for processing files are `strip` and `split`. The first strips away any white-space characters (ex. `\n` and (`\t`) from the ends of a string. The second splits a string based on the delimiter you give it (ex. `","`). You can stack the two string commands, like so:

```
line_list = line.strip().split(",")
```

1.4 Processing all the lines of a file

You can then loop over the contents of the rest of the file. Please write a for loop that builds up the full dataset as a list-of-lists.

1.5 View and summarize the dataset

Please use the `head` function and the `summarize` function from the `data` module to view and summarize the data in `"fatal_encounters.csv"`.

2 Disparities in Police-involved Fatalities

Summarizing a dataset is a good first step, but it does not tell us all we might want to know about something like police-involved fatalities. Here, we will look specifically at demographic disparities.

2.1 Automate loading a file

In previous weeks, we gave you the file `data.py` from which you could load the functions `data_header` and `data_as_list_of_lists`. You now have the tools to write these functions yourself! Please do so, and use them to load both the header and the dataset from `"state_population.csv"`.
Hint: use your code from Problem 1!

2.2 Choose a minority, and a state

This dataset on state populations is already summarized, giving the proportion of the population by several demographic categories. We would like to be able to compare the underlying demographics of a state to the proportion of fatal encounters with police where the person killed was of a particular race or ethnicity. Please choose a demographic category that you would like to consider.

To study disparities we will need the underlying population to be large enough. Please choose a state that has both (a) over 100 police-involved fatalities and (b) at least 10% underlying proportion of the group you have chosen. Hint: `data.py` also includes our `filter` functions from prior weeks.

2.3 Check for demographic disparity

Now compute the proportion of all of the people killed by police belonging to the demographic group you are considering. Please print this number alongside the underlying proportion of the population for that state. Are there disparities in police-involved fatalities in this state?

3 (extra) Automate this analysis

You chose a demographic group and quantified the demographic disparity in police-involved fatalities. Please write a function that does this for every state, and find the states with the largest disparities.

This handout was originally created by Carolina Mattsson and Stefan McCabe, Fall 2019. This exercise was originally created by Sarah Shugars, Spring 2019.