Practicum 7 - CSV files!

DS2001 - Computer Science Practicum
Fall 2021
October 20/21th, 2021
Practicum Deadline: October 22nd, 2021 at 12:00pm (that's noon!) Boston time

Project groups + initial topics are due on Friday, October 22nd at 9pm on Gradescope.
Note! You may find many of the functions that you write in practicum today to be very helpful when implementing your final project. Pay attention to them! :)

What we're practicing today:

- functions
- lists of lists
- functions + lists

Handouts:

- Writing Functions (from DS 2000)
- Functions + Scope (from DS 2000)
- Functions + Testing (from DS 2000)
- 2D Lists (from DS 2000)

What to do if you miss practicum this week:

- Fill out the "I'm missing class" form ASAP
- Follow up in office hours with Felix (find office hours links on the course website). **Note that this is required.**
- Come to practicum next week if you are well again. **Do not come to class if you are not feeling well.**

How far we expect you to get this week:

- We expect you to complete **Task 2**, which means writing the `get_column` function with the correct parameters and return value. We *highly* recommend attempting task 3 as well.

Everyone should create **one** file for the whole practicum called **p7.py**.

We have a few problems to tackle today; get through as many as you can, and submit whatever you have done at the end of your section. Each question in this write-up will describe a problem to work on, then might ask you a question that you'll answer with your group.

At the end, everyone should make sure to turn in your code to gradescope **with your group's answers to the questions written as comments in your file!**

Set-up: background of your data

This week, we'll be working with data from Analyze Boston's crime incident reports, filtered just for March from 2020.

We can think of a list of lists as corresponding to a table of data where each sublist is a row in our data set. Columns of data are represented by values corresponding to the same index in multiple rows (sublists).

Here is an example, looking at the 1st, 30th, and 59th row in your big data file (so that you can see how the values vary).

	INCIDENT_NUMBER data[row][0]	OFFENSE_DESCRIPTION data[row][1]	SHOOTING data[row][2]	OCCURRED_ON_DATE data[row][3]	DAY_OF_WEEK data[row][4]	HOUR data[row][5]	STREET data[row][6]
data[0]	I20219259	LARCENY ALL OTHERS	0	2020-03-10 11:49:00	Tuesday	11	BOWDOIN ST
data[29]	I20203469	TOWED MOTOR VEHICLE	0	2020-03-28 00:00:00	Saturday	0	TREMONT ST
data[58]	I20203113	ASSAULT - SIMPLE	0	2020-03-25 22:00:00	Wednesday	22	BLUE HILL AVE

Here, if this were a list of lists saved in the variable `data`, then we would access rows (sublists) with the first index given and columns with the second index given as in `data[row][column]`.

Task 0: Write your file comment, your `if __name__ == "__main__":` and import the `p7_utils` and `matplotlib.pyplot` modules

First, write down your name in a comment at the top of your file and a small description of what the file is, like we did last time. This is a good habit to get into for all your programs!

Next, import the `p7_utils` and `matplotlib.pyplot` modules so that you can access the functions in them.

Task 1: Read in the example data, define a `FILENAME` constant

For all of this practicum, test each task with the file called `"boston-incident-reports-march2020-tiny.csv"`, then, **after** it's working, run your code on `"boston-incident-reports-march2020.csv"`. The **only** code that you should need to modify for this is the value of your `FILENAME` constant.

Using the `p7_utils.read_csv()` function, read in the data to a new list of lists. Once you've read it in, print out the length and the first three elements of the list. Notice that this function is different from the `read_csv` function that you're implementing for your DS 2000 homework :).

Warning! This is **A LOT** of data! (Even for the "tiny" file!) We do **NOT** recommend printing out the entire list of lists. :)

Example output:

```
Number of rows: 208687
[['I20219259', 'LARCENY ALL OTHERS', 'C11', '340', '0', '2020-03-10 11:49:00', '3',
'Tuesday', '11', 'BOWDOIN ST'], ['I20219259', 'LARCENY ALL OTHERS', 'C11', '340', '0',
```

```
'2020-03-10 11:49:00', '3', 'Tuesday', '11', 'BOWDOIN ST'], ['I20219259', 'LARCENY ALL
OTHERS', 'C11', '340', '0', '2020-03-10 11:49:00', '3', 'Tuesday', '11', 'BOWDOIN ST']]
```

Task 2: Get the data from one column

Write a function with the following signature.

```
"""
name: get_column
This function takes a list of lists and an integer and gives the user the list of values
from the indicated column. Does not change the passed in list of lists.
data - list of lists
index - integer indicating which column values to access
return:
List of values where the first value is the value at position "index" from the first row
in data, the second value is the value at position "index" from the second row in data,
etc for all rows in data
"""
```

Use this function to get the data from the column containing information about the day of the week that the incident occurred on. Make sure that the length of the list you get back matches the number of rows in your data!

Example output (for tiny file):

```
length of column list (should be 21277 ): 21277
```

Example output (for "regular" file):

```
length of column list (should be 208687 ): 208687
```

Group Question: how should the length of a column relate to the dimensions of your dataset? If we gave you a new dataset with 1,000,000 rows and 25 columns, is there anything that you would have to update about your code? What would you expect to be the length of the list returned from this function?

Task 3: Graph some histograms

Use the functions that you've written so far to plot two different histograms of columns. A histogram is a special kind of graph on which one axis is always frequency/count. Matplotlib conveniently has some built-in functionality for you here!

Make sure that one of the graphs represents the day of the week. Columns that are good options for histograms in general are columns where there is a set number of categories that the different incidents are categorized into. Columns like incident id are not good options for histograms.

Use the `plt.hist(values)` function to create this histogram.

Hint: if you end up with overlapping labels on the x-axis, you can rotate them like this:

```
plt.hist(values)
plt.xticks(rotation=90)
plt.show()
```

Group Question 1: Take a look at the day of the week graph that you created. What about this graph could be improved?

Group Question 2: Compare the graph from the tiny data file with the one using the full-size data set. Do you see the same trends here? What may have caused differences in what you see?

Task 4: Filter rows, make a new graph

Write a function with the following signature.

```
"""
name: exact_match
This function takes a list of lists of values and returns a new list of lists containing
only rows from the original data where the value at column index matches the provided
target value.
parameters:
data - list of lists (your table of data)
index - column to look in
target_value - value to match
return:
a new list of lists containing all target rows
"""
```

Then, use this function to help produce a bar graph of day of the week vs. count of incidents for that day. You'll need to build a list of days and a corresponding list of counts for each day. Print out these lists to verify your counts.

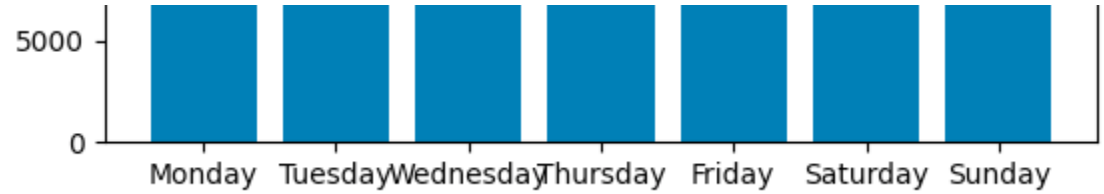
Note that this is not the output of the `exact_match` function! You'll write this function and then use it alongside other code to achieve your goals here.

Example output (for the tiny data):

```
['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
[2917, 2880, 2238, 2071, 2310, 6294, 2567]
```

Use the `plt.bar(x_values, y_values)` function to do this, which is a more general version of what you saw with `plt.hist(values)`.

Your graph should be "in order" — starting with either Sunday or Monday, the x-axis should look similar to (note that we're not showing the whole y-axis here):



Hint 1: you'll want to make a list that defines the order that you want your bars to be plotted in.

Hint 2: if you call your `exact_match` function appropriately, you won't need to re-write any code to loop through your overall data searching for matching rows. :)

Group Question: Looking at your `exact_match` function, what would you need to alter about it if you wanted to write another, similar function that looked for a specific range of values (e.g. "I want rows where the 2nd column's value is between 10 and 100") instead?

Task 5: Get unique values

Write a function with the following signature.

```
"""
name: unique_values
This function takes a list of values and returns a list containing only one copy of every
value in the given list, in sorted order.
parameters:
lst - list of values
return:
a new list of unique values, in sorted order
"""
```

Then, use this function to find what different values the offense description column might contain. Pick out one of these offenses and investigate it further. Does it follow the same overall pattern for frequency by day of the week that we see reflected in the data as a whole? Make sure to call the functions that you've already implemented to help figure this out!

Hint: first, you'll want to use your `exact_match` function to isolate the relevant part of the data. Next, you'll follow the same strategy as in task 4! (feel free to make this graphing code into a function that you re-use!)

Finished early? Do the following....

- Make constants for the index positions of the different columns that you access. This avoids having "magic numbers" and makes it so when you update your code, you'd only need to change one thing.

- Investigate the number of reports in the week(s) surrounding March 10th, when a state of emergency was declared in Boston due to the pandemic. Was there a dip in incidents reported at this time?
    - Hint: you'll need to do some string manipulation here to do what's called "parsing" the information in the date column. You can use the string split() function to first split on spaces, then split the first chunk based on "-" characters. Don't forget to convert any strings to integers if you want to compare them by their numeric ordering!