Assignment P4

Due Date: June 14

Purpose

Summer is the best time write programs. The grass is green, the air is humid, and the temperature is hot. Why not stay in air-conditioned space and write a Python program? You now have sufficient Python knowledge to write very useful software. In this case, you will draw a graph based on data found on the MA COVID-19 website. To draw these graphs, you will continue to practice writing Python programs that use files, loops, and selection statements, as well as string functions.

Problem

There are myriad websites and newspapers that show graphs of the daily number of deaths due to COVID-19. The data is gathered from various local, state, and federal sources, and put together in spreadsheets and stored in csv (*column separated values*) format. Each day is stored on a separate line, with various data points, such as the date, the number of deaths, and other values, separated by a comma. Your job is to read one of these files and produce a bar graph showing the number of COVID-19 deaths in MA using the Python graphics package.

Input

Your program should first prompt for a filename. This file will be in csv format that contains the number of daily COVID-19 deaths. The first line of the file contains the header information; that is, the name of each column:

```
Date,DeathsConfTotal,DeathsConfNew,DeathsProbTotal,DeathsProbNew
```

This means the first column is the date, the second column is the total number of confirmed deaths, the third column is newly reported deaths (on that day), etc. We need the data in the third column. The file then has any number of lines with each column separated by a comma. Some columns may have no data. For example:

```
6/1/2020,6894,48,,
6/2/2020,6944,50,,
6/3/2020,7012,68,,
6/4/2020,7062,50,,
6/5/2020,7097,35,,
etc.
```

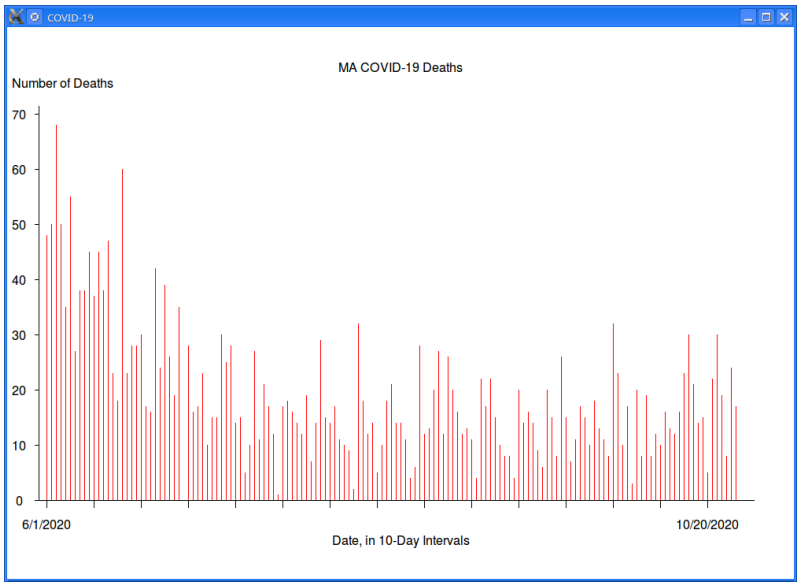
The first line shows a data of June 1, with 6894 total deaths to date, and 48 new deaths on that day. The last two columns have no data. If you investigate further, you can see the total deaths are added in a day later.

To determine when you've reached the end of the file (EOF) in Python, first read a line:
`dataLine = filePointer.readline()`
Then check that it has data:
`while (len (dataLine) > 0):`
If the length is 0, then that means there's no data and you've reached the end of the file.

Finally, a mouse click anywhere on the graph should close the graphics window.

Output

The program should display either a bar chart of the data. The bar chart for the sample data (available on the course web page) should look similar to the figure below:



The graph should have all appropriate labels: hash marks on the x-axis every 10 days including the date on the first and last hash, and hash marks on the y-axis for every 10 deaths. These should be scaled to match the data correctly.

Specifics

- Make the graphics window approximately 1000×700 pixels in size.
- **You can not assume the number of data points contained in the file!** The given test case is just that: **one** test file. Your program should work for files with **any number** of data points. This means you have to scale everything properly to fit within your window. You may have more points than the sample, for example. This means that your vertical bars must be closer together, which also means you have to adjust your x-axis hash marks. The number of deaths might also be higher or lower than the sample; this means this has to be scaled appropriately as well. **The data should be shown as clearly as possible; that is, as big as you can fit into your window.** For example, if you have only 20 data points, you do not want a graph that has 20 bars scrunched over on the left side of the graph.
- It is acceptable to have bars that are just one pixel wide in all cases. With a small number of data points, this will not look ideal. In the general case, the graph should look similar to the sample. However, there is a limit to how many bars can fit into your window. You should do an error check and display an error message in the shell if the graph can not fit in the window.

Notes

Once again, I urge you to start small and work your way to the final program. I suggest working on the input first, since if the input doesn't work, then nothing else will, either. You need to take

the input strings apart to get the useful data. Be certain that one part of the program works before going on to the next. In any case, you must have some sort of plan (algorithm) before trying to code any part of the problem! Work things out with a very small sample input file.

While you are coding and debugging, call the `print()` function liberally so that you know that your values are correct up to that point. You can always delete extra statements later.

As before, email your source code to me, using the usual naming conventions.

If you think a little bit harder, you'll understand.
– Professor Moorthy (a grad school professor of mine)