Final Project Instructions – Programming Focused

Introduction:

For this final project, you will use lottery data to create a user-interactive game. Data for winning lottery numbers and the corresponding prize(s) are provided in two accompanying csv files. The goal of this project is to use the skills you have learned to produce a game allowing a user to enter or randomly select five lotto numbers and play them from the period of October 27, 2006 through May 14, 2014. You will have more freedom with this project than the Data Analysis focused project. In lieu of an analysis report, you will turn in a single zip file containing two files:

1. Your python code
2. A short write-up containing:
    a. A brief description of how your code works (either a verbal description or using a flow chart) and
    b. A brief discussion of the prompts in the section of these instructions name **Analysis**

Relevant Files:

1. Cash5-WinningNumbers.csv
    a. This is a csv file containing the 5 winning lotto numbers for every play day from Oct 27, 2006 through May 14, 2014. The first few lines of the file are below. The order of the numbers is not relevant for our game, but each play day is associated with five non-repeating winning numbers.

Date	Ball 1	Ball 2	Ball 3	Ball 4	Ball 5
5/14/2014	4	27	1	34	36
5/13/2014	7	29	11	39	1
5/12/2014	15	33	31	12	5
5/11/2014	29	33	37	22	32

2. Cash5-Prizes.csv
    a. This file contains the associated prizes (in $) for correctly matching four (4) and five (5) numbers on every day from Oct 27, 2006 through May 14, 2014. It also contains the number of players who won prizes that day for matching 2, 3, 4, or 5 numbers. The first few lines are below:

date	prize_5	prize_4	winners_5	winners_4	winners_3	winners_2
10/27/2006	50000	250	0	37	1454	14674
10/28/2006	75000	250	0	42	1424	15029
10/29/2006	102914	187	1	32	886	8575
10/30/2006	50000	250	0	21	928	9699
10/31/2006	60000	250	0	32	1056	11922

<u>Rules for the Lotto Game:</u>

- **Each date is played only once, but every date is played**
- **Ball order does not matter, only matching**
- **Five winning numbers are selected every day**
- **It costs $1 to play per play**
- **All playable numbers are between 1 and 39, inclusively**

<u>Winners:</u>

The following are the Prizes for each class of "win"

Match 0 numbers = $0.00

Match 1 number = $0.00

Match 2 numbers = $1.00

Match 3 numbers = $5.00

Match 4 numbers = Determine correct prize using the data in Cash5-Prizes.csv column 'prize_4'

Match 5 numbers = Determine correct prize using the data in Cash5-Prizes.csv column 'prize_5'

<u>Outputs:</u>

Your code should output:

1. The numbers your player selected or an indication that numbers were assigned randomly
2. The total amount of money spent playing
3. The total amount of money won
4. The return on investment for playing everyday from October 27, 2006 to May 14, 2014
   a. Return on investment formula:

$$ROI = \frac{Total\ Winnings - Total\ Investment}{Total\ Investment} \times 100\%$$

<u>Analysis:</u>

1. Find the ROI for a user selected set of numbers
2. Find the ROI for a randomly selected set of numbers
3. Find the ROI for randomly reselecting numbers every day.
4. Note any scenario that resulted in a 4 or 5 number match.
5. Based on your simulations, does any one scenario (user selected, randomly selected, or randomly reselected) influence the final ROI? Why might that be (or not)?
6. **Optional Challenge:** Run each scenario (user selected, random selected, random reselected) a minimum of 5 times and report the average ROI for each scenario.

Hints:

- Since this is a game that, in theory, one might want to replay, work out of the editor. Consider making the game a callable function.

- You will need a user input for this game. In python, you can use:

  input('prompt for your user here ')

- Note the way the dates are formatted in the two csv files. Pandas has a built-in function called to_datetime to convert dates (and times, but that's not something we're worried about here) to a standard format. The documentation is here: https://pandas.pydata.org/docs/reference/api/pandas.to_datetime.html

- You may need both loc[] and iloc[] to access data from the reference csv's. loc[] uses labels to call data, and iloc[] uses integer rows and columns. An in-depth discussion of the differences between the two is here: https://towardsdatascience.com/how-to-use-loc-and-iloc-for-selecting-data-in-pandas-bd09cb4c3d79

- You will need to import random, a built-in python function to produce random numbers. I suggest random.sample(), but you may use whichever method you desire. Documentation is here:
  https://docs.python.org/3/library/random.html

IMPORTANT NOTE:

Some users may find that the version of Spyder returns an error when attempting to use input(). This is a known bug that had been fixed in later versions. However, the version in Anaconda may not be this updated version. Try a simple input() command before you begin. If you ***do not*** return an error, carry on with your project. If you ***do*** return an error, follow the instructions below.

What to do if input() is not working:

You will need to run something called a virtual environment in Anaconda. This will let you access the version of Spyder without the bug.

1. Close Spyder and Anaconda.
2. Open the Anaconda command

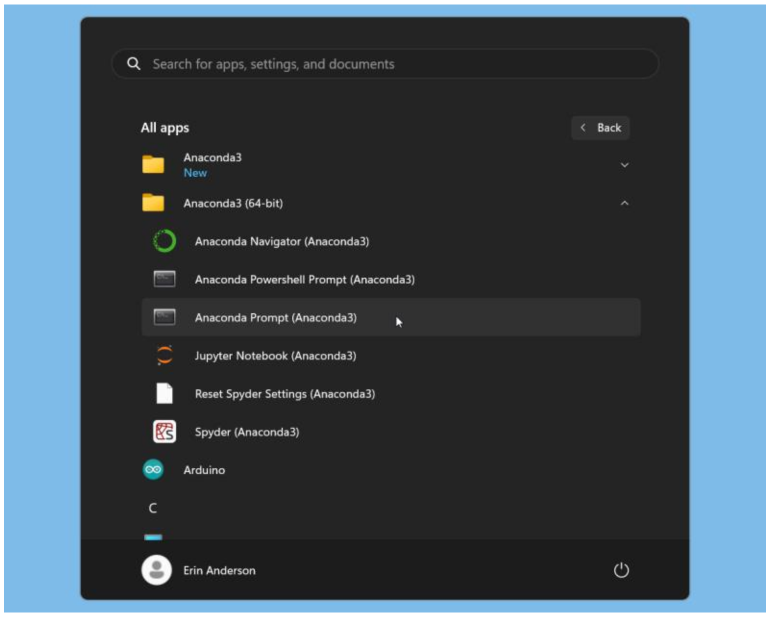
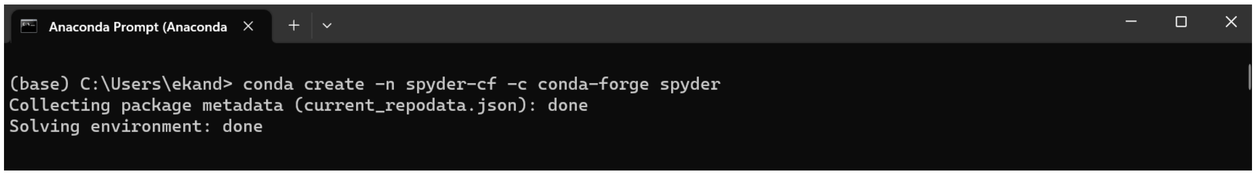

Search for apps, settings, and documents
All apps
Back
Anaconda3
New
Anaconda3 (64-bit)
Anaconda Navigator (Anaconda3)
Anaconda Powershell Prompt (Anaconda3)
Anaconda Prompt (Anaconda3)
Jupyter Notebook (Anaconda3)
Reset Spyder Settings (Anaconda3)
Spyder (Anaconda3)
Arduino
C
Erin Anderson

Figure 1 Open the Anaconda Command

3. Enter the following into your Anaconda Command:

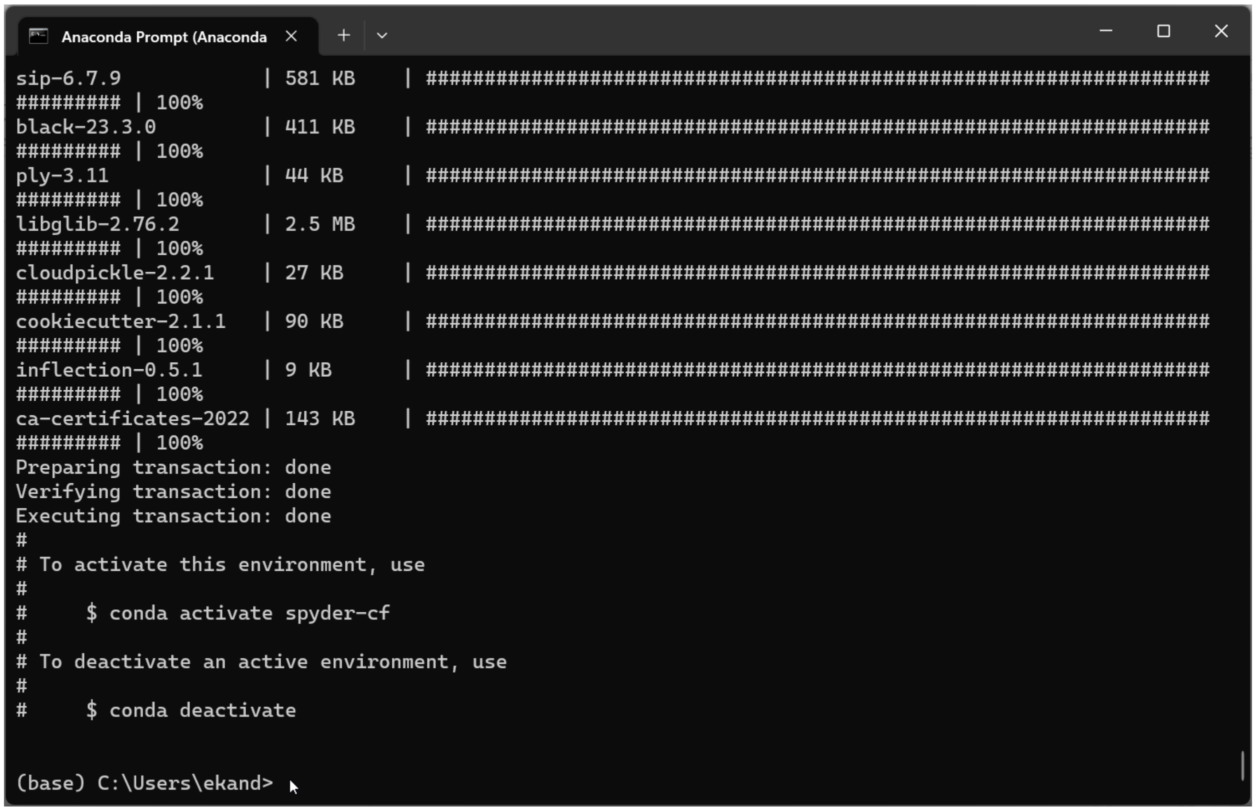
```
conda create -n spyder-cf -c conda-forge spyder
conda activate spyder-cf
spyder
```



```
(base) C:\Users\ekand> conda create -n spyder-cf -c conda-forge spyder
Collecting package metadata (current_repodata.json): done
Solving environment: done
```

Figure 2 Anaconda Command with inputs

4. Follow the prompts in the Anaconda Command.
5. When the process is completed, you should have a window that looks like this:



```
sip-6.7.9           | 581 KB    | ##########################################################################
######### | 100%
black-23.3.0        | 411 KB    | ##########################################################################
######### | 100%
ply-3.11            | 44 KB     | ##########################################################################
######### | 100%
libglib-2.76.2      | 2.5 MB    | ##########################################################################
######### | 100%
cloudpickle-2.2.1   | 27 KB     | ##########################################################################
######### | 100%
cookiecutter-2.1.1  | 90 KB     | ##########################################################################
######### | 100%
inflection-0.5.1    | 9 KB      | ##########################################################################
######### | 100%
ca-certificates-2022 | 143 KB   | ##########################################################################
######### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate spyder-cf
#
# To deactivate an active environment, use
#
#     $ conda deactivate


(base) C:\Users\ekand>
```

Figure 3 Completed Initiation of a Virtual Environment

6. You will now be able to open Spyder-cf, which should be available from the location of your Anaconda installation. It is a normal Spyder console, it's just using the latest version of Spyder not otherwise available from the current Anaconda Environment.
7. Open the Spyder-cf console
8. Reinstall numpy and pandas by entering the following into the Spyder console
    a. pip install numpy
    b. pip install pandas
9. Work as normal after this point!
10. Remember to deactivate the environment if you no longer wish to use it after this project. Otherwise, treat it the same as you would the usual Spyder console.
11. Email me with any questions.