ENGR 102 – Fall 2024
Lab: Topic 11 (individual)

Deliverables:
There are three deliverables for this individual assignment. Please submit the following files to zyBooks:

- `counting_coins.py`
- `pixel_painter.py`
- `weather_data.py`

These activities are meant to help give you practice reading and writing files, as well as processing larger amounts of data, which is one of the common tasks that computer programs are written for.

Activity #1: Counting coins – individual
While digging through a box of very old handheld game consoles, you find one that piques your interest. Wondering how it works, you access the source code and find a text file named game.txt full of instructions, one per line. Each instruction consists of an operation (coin, jump, or none) and a signed number (like +28 or -3). You quickly figure out that coin increases or decreases a value that stores the number of coins earned by the player, jump will jump to a new instruction relative to itself, and none does absolutely nothing. After executing a coin or none operation, the instruction immediately below is executed next. However, jump +2 would continue to the instruction 2 lines below it, and jump -5 causes the instruction 5 lines above to be executed next. The program ends when it attempts to execute an instruction immediately after the last instruction in the file.

Write a program named count_coins.py that opens the game file (game.txt), executes the instructions, and creates a new file named coins.txt that contains only the numbers of coins gained or lost in the order the program is executed. Have your program output the total number of coins earned using the example output below.

Example output:
```
Total coins collected: ???
```

Example coins.txt file created by your program:
```
28
0
...
-87
4
8
```

Activity #2: Pixel painter – individual
Run length encoding is a form of data storage where runs of data are stored as a single data value. A run of data is a sequence of the same value that occurs in many consecutive elements. This type of data storage is efficient for simple graphic images such as icons, line drawings, and Conway's Game of Life. For this activity, consider a binary image where each pixel is either light (a space) or dark (a character). Each line of the image is represented by comma separated values that indicate the number of consecutive pixels that are light and dark. The first value in a line always corresponds to the number of light pixels, and following values alternate between dark and light. For example, the line `1,2,3,4` represents 1 light pixel, followed by 2 dark pixels, 3 light pixels, and finally 4 dark pixels. If a space is used to represent light pixels and the character `O` is used to represent dark pixels, the line would look like this (spaces underlined for clarity): `_OO___OOOO`

Write a program named pixel_painter.py that takes as input a filename and a character, converts the contents of the file to pixel art using a space for light pixels and the entered character for dark pixels, and writes the art to a new file of the same name but with the .txt extension.

[1]NOAA's NCEI, https://www.ncdc.noaa.gov/

Example pixel_triangle.csv file:

```
2,1,2
1,1,1,1,1
0,5
```

Example output using inputs **pixel_triangle.csv** and **A**:

```
Enter the filename: pixel_triangle.csv
Enter a character: A
pixel_triangle.txt created!
```

Example pixel_triangle.txt file created by your program:

```
  A
 A A
AAAAA
```

Activity #3: Weather data – individual

On Canvas, there is a CSV file posted with this assignment named WeatherDataCLL.csv that contains weather data from Easterwood Airport (in College Station) for 10 years. The data was taken from the National Oceanic and Atmospheric Administration's National Centers for Environmental Information[1]. You can view the data by opening the file in any text or spreadsheet editor (e.g. Notepad++, Excel) or in your IDE. The first line of the file contains the column headers explaining what each column is. Note: Excel does NOT display the date correctly. Open the file in your IDE to see the correct date format. Also, some days are missing data.

Download the file and write a program named weather_data.py that does the following:

1. Open the CSV file for reading
2. Read the CSV file and compute
    a. the maximum temperature seen over the 10-year period
    b. the minimum temperature seen over the 10-year period
3. Output the results to the console using the format below
4. Perform the following five data analysis exercises and output the results to the console. Take as input from the user a month and year, then for that month,
    a. Calculate the mean of the average temperatures (use 1 decimal place)
    b. Calculate the mean of the average dew point (use 1 decimal place)
    c. Calculate the mean relative humidity (use 1 decimal place)
    d. Calculate the mean daily wind speed (use 2 decimal places)
    e. Calculate the percentage of days with non-zero precipitation (use 1 decimal place)

Example Output (using inputs **July**, **2023**, but with made-up numbers):

```
10-year maximum temperature: 102 F
10-year minimum temperature: 7 F

Please enter a month: July
Please enter a year: 2023

For July 2023:
Mean average daily temperature: 92.5 F
Mean average daily dew point: 78.1 F
Mean relative humidity: 58.6%
Mean daily wind speed: 7.34 mph
Percentage of days with precipitation: 2.1%
```

[1]NOAA's NCEI, https://www.ncdc.noaa.gov/