ENGR 102 – Fall 2021
Lab Assignment #10b

Deliverables:

There are several deliverables for this individual assignment. Please submit the following files to Mimir:

- `Lab10b_Act1.py`
- `Lab10b_Act2.py`
- `Lab10b_Act3.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 to accomplish.

Activity #1: *Reading from and writing to a file – to be done individually*

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 +25 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 Lab10b_Act1.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. You do not have to submit your coins.txt file to Mimir.

✓ Example output:

```
Total coins: ???
```

✓ Example coins.txt file created by your program:

```
29
0
…
15
-38
```

Activity #2: *Calculating and writing to a file – to be done individually*

This activity provides practice with what is called a CSV (Comma Separated Value) file. CSV files are one of the most common formats for storing tabular data (e.g. spreadsheets). Each line represents a row of the table, and the cells in each column are separated by commas. CSV files can usually be read by spreadsheet programs (such as Microsoft Excel), and most spreadsheet programs can output their data in the CSV format. These files are often given the .csv extension. In this activity, you will practice reading and writing data in the CSV format using Python.

Lab Assignment #10b

Write a program named Lab10b_Act2.py that will write to a file a list of amortized values for a loan. Specifically, the program should:

1. Prompt the user for an output file name for the data
2. Prompt the user for the principal amount of the loan (P), the number of months (N) over which the loan will be repaid, and the annual interest rate (i). (Note that i should be a decimal number, not a percentage: 0.025 **not** 2.5%)
3. Calculate the monthly payment (M) as shown below. Note: $J = i/12$

$$M = \frac{P \cdot J}{1 - \left(\frac{1}{1+J}\right)^N}$$

   a. Note: You should verify your program's calculations: e.g. for $P = \$100000, N = 60,$ $i = 0.025$ the monthly payment should be $M = \$1774.74$
4. Open the output file for writing
5. Write to the output file column headers for the table as comma-separated strings: `Month,Total Accrued Interest,Loan Balance`
6. Write to the output file the initial values for the month number, the total amount of interest accrued so far, and the amount remaining on the loan, separated by commas: `0,$0.00,$P.pp` where `P.pp` is the initial value of the amount of the loan)
7. For each month, starting with month 1 and ending when the amount remaining on the loan is less than $0.01:
   a. Calculate the accrued interest by multiplying the balance at the beginning of the month by $J = i/12$
   b. Calculate the balance at the end of the month by adding the accrued interest to the beginning balance and subtracting the monthly payment
   c. Write to the output file the month number, the **total** amount of interest accrued so far (to the nearest penny), and the amount remaining on the loan (to the nearest penny), separated by commas (see output format below)

Note: If you write your CSV file correctly, you should be able to open it in a spreadsheet program that can read CSV files (like Excel). You can also check the values themselves using the `PMT()` function in Excel, e.g. `PMT(0.025/12, 60, 100000) = ($1774.74)`. You do not have to submit this file to Mimir.

✓ Example output (using inputs **`out.csv`**, **`100000`**, **`60`**, **`0.025`**):

```
Please enter the output filename: out.csv
Please enter the principal amount: 100000
Please enter the term length (months): 60
Please enter the annual interest rate: 0.025
```

✓ Example file created by your program:

```
Month,Total Accrued Interest,Loan Balance
0,$0.00,$100000.00
1,$208.33,$98433.60
…
59,$6480.48,$1771.05
60,$6484.17,$0.00
```

Lab Assignment #10b

Activity #3: *Reading in and processing data – to be done individually*

On Canvas, there is a CSV file posted with this assignment named "CLLWeatherData.csv" that contains weather data from Easterwood Airport (in College Station) for 3 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 PyCharm. The first line of the file contains the column headers explaining what each column is.

Download the file and write a program named Lab10b_Act3.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 3-year period
    b. the minimum temperature seen over the 3-year period
    c. the average daily precipitation over the 3-year period (use 3 decimal places)
3. Output the results to the console using the format below
4. Perform the following three 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 percentage of days when the average daily wind speed was above 10 mph (use 1 decimal place)
    c. Calculate the mean of the daily precipitation levels (use 4 decimal places)

✓ Example Output (using inputs **January**, **2018**, but with made-up numbers):

```
3-year maximum temperature: 101 F
3-year minimum temperature: 19 F
3-year average precipitation: 0.241 inches

Please enter a month: January
Please enter a year: 2018

For January 2018:
Mean daily temperature: 45.6 F
Percentage of days with average wind speed above 10 mph: 31.8%
Mean daily precipitation: 0.0279 inches
```

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