

You are to write the following program as described below. For all programs, include comments in your code that describe the purpose of individual blocks. Remember the appropriate header information.

Program 1: Temperatures

☑ *Analyze data from a file and output processed results to a file.*

For this program use the “`with open(<-->) as <-->:`” argument to open file(s), and do not use CSV files, a `csv.reader`, or a `csv.writer`. Write a program that reads a data set of temperatures given in Celsius (one temperature value per line of the file) from a file named `Celsius.dat` (not provided—you should create one for testing your program), and creates a file `Fahrenheit.dat` that contains the same temperatures in Fahrenheit (one per line, in the same order).

Program 2: Amortization in CSV

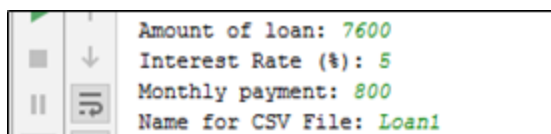
☑ *Analyze data from a file and output processed results to a file.*

For this program, use the “`FileID = open(<-->)`” argument to open and close file(s). Write a program to save a list of amortized values for a loan to a file. Specifically:

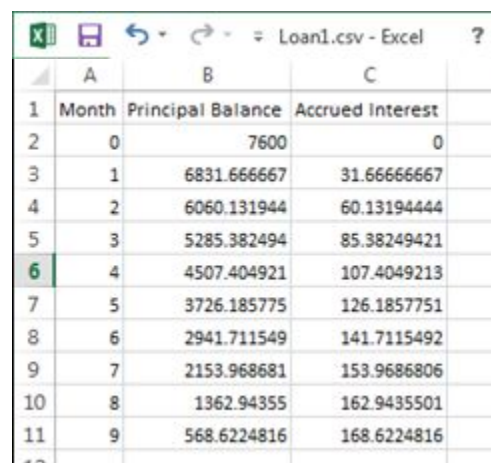
- Ask the user the amount of the loan, the annual interest rate, the amount paid monthly and for the name of the file in which to store the results.
- Your program should add a “.csv” extension to the file name, to indicate that it is a CSV file.
- Each month, calculate the interest (loan value times 1/12 of the annual interest rate). Then, find the amount remaining on the loan by adding the interest (increasing the loan value), and applying the monthly payment (reducing the loan value).
- For each month, write to the output file the month number, the total amount of interest accrued so far, and the amount remaining on the loan, separated by commas.
 - Start with month 0, when there is no payment and no interest, with month 1 being the first payment and first interest accumulation
 - If the loan eventually will be paid off (i.e. if the loan amount is decreasing), write out values until the loan amount is 0
 - If it will not be paid off (i.e. the loan amount increases or stays the same each month), then write 30 months’ worth of data.
- Create column headers for the table, indicating what each column is.

Note: if you write your .csv file correctly, you should be able to open it in a spreadsheet program that can read .csv files. Here is an example for the displayed inputs, opened in Excel:

Example output (note: formatting is individual style):



Amount of loan: 7600
Interest Rate (%): 5
Monthly payment: 800
Name for CSV File: Loan1



	A	B	C	D
1	Month	Principal Balance	Accrued Interest	
2	0	7600	0	
3	1	6831.666667	31.66666667	
4	2	6060.131944	60.13194444	
5	3	5285.382494	85.38249421	
6	4	4507.404921	107.4049213	
7	5	3726.185775	126.1857751	
8	6	2941.711549	141.7115492	
9	7	2153.968681	153.9686806	
10	8	1362.94355	162.9435501	
11	9	568.6224816	168.6224816	
12				

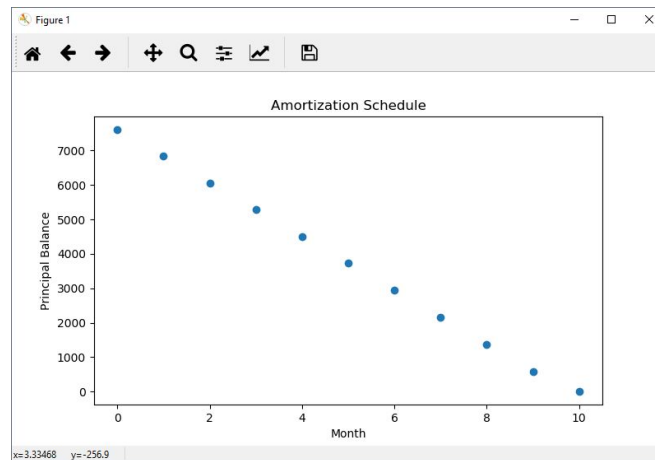
(continued, next page)

Program 3: Visualizing Amortization

☑ *Apply programming techniques to graphically plot data*

Write a program to read the data stored from Program 2, and plot a scatter plot of the loan balance for each month as saved in the file. If the loan balance is decreasing, the final point should be 0; if the balance is *not* decreasing, there should be 30 points.

Use matplotlib to create the plot. Include plot title, axis labels and any other information necessary to create a professionally styled plot. Aim to make yours look *better* than this:



Program 4: Weather

☑ *Create Python program to read user input, perform necessary data reformatting, and print the expected output.*

A CSV file containing weather data from Coulter Field in Bryan is available on our eCampus course page. This file contains 3 years of data (minus one day, for some reason!) from Weather Underground (wunderground.com). There are two versions of the file, one for Windows and another for Mac; the only difference is new-line and carriage return formatting (you don't need to worry about this).

Download the file to your system; **do not rename the file** to use. Note that the first line of the file contains the column headers explaining what each column is.

Write a program that will open the .csv file, and give descriptive output of the following information to the PyCharm console:

- The maximum and minimum temperature seen over the 3 year period
- The average daily precipitation
- Any two (2) other “interesting” data analysis questions of your choice*

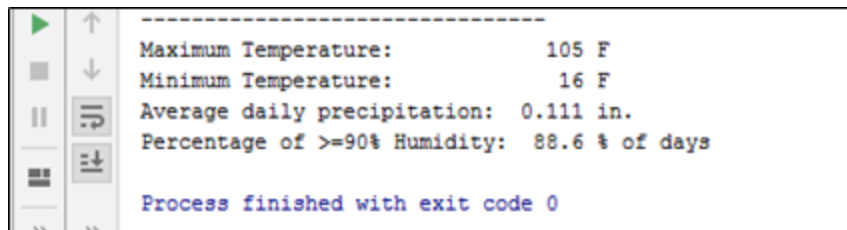
For at least one, use the date information in some way: here are some ideas if you're not feeling creative, but you can pick whatever you want:

- For some particular day, such as December 25, find the maximum and minimum temperatures reached among the 3 years of data.
- For some particular month, such as July 2015, calculate the average high temperature.
- Calculate how frequently the pressure increases from one day to another vs. how frequently it decreases.
- Calculate the percentage of days when the humidity was above some value, like 90%.
- Calculate the mean and standard deviation of precipitation levels.

The two analysis questions should be substantially different from each other. For example, you should not find the min/max temperature for three different dates, or find the min/max pressure all for December 25.

(continued, next page)

Example output (note: yours requires two 'interesting' data results, this example only contains one):

A screenshot of a terminal window. On the left is a vertical toolbar with icons for running (green play button), stopping (red square), pausing (two vertical bars), and stepping through code (arrows). The terminal output is as follows:

```
-----  
Maximum Temperature:      105 F  
Minimum Temperature:      16 F  
Average daily precipitation: 0.111 in.  
Percentage of >=90% Humidity: 88.6 % of days  
  
Process finished with exit code 0
```