CSE101 – Spring 2020
Programming Assignment #8
(13 points + 3 points extra credit) Due: 8 June 2020

Instructions

For each of the following problems, create an error-free Python program.

- Each program should be submitted in a separate Python file that follows a particular naming convention: Submit the answer for problem 1 as "Assign8Answer1.py" and for problem 2 as "Assign8Answer2.py".
- These programs should execute properly in PyCharm using the setup we created in lab.
- At the top of every file add your name and Stony Brook email address in a comment.

Problems

Problem 1: Wolfie Numerals

For this problem you will be writing some functions to convert between two number representations: the familiar Arabic numerals we use every day and a new scheme we call *Wolfie Numerals*.

Download this initial code to get started.

Part 1. Arabic Numerals to Wolfie Numerals Converter (4 points)

You've heard of Roman numerals, but have you heard of Wolfie numerals? Probably not because some CS people at SBU have invented them and we are using it for this assignment. Wolfie numerals are similar to Roman numerals in that numbers are formed by combining symbols and adding the values. In Wolfie numerals, each numeral has a fixed value representing a power of 8 (rather than a power of 10 or half of a power of 10, as in Roman numerals):

Symbol:	I	E	S
Value:	1	8	64

In addition, there are also symbols representing the halved powers of 8. A complete list of Wolfie numerals is given below:

Symbol:	I	F	E	T	S
Value:	1	4	8	32	64

Symbols are placed from left to right in order of value, starting with the largest. For example, `EFII` is 14 : $(8 \times 1) + (4 \times 1) + (1 \times 2) = 8 + 4 + 2 = 14$.

Note: You may want to review (or learn) the rules for the Roman numerals to better understand this assignment. In a few specific cases, to avoid having three or more characters being repeated in succession (such as `III` or `EEE`), subtractive notation is used, meaning that a smaller number is listed before the bigger value, as in this table:

Number:	3 (4-1)	7 (8-1)	24 (32-8)	56 (64-8)
Notation:	IF	IE	ET	ES

For example, with these subtractive notations, instead of writing 7 as 4 + 1 + 1 + 1 (`FIII`), you must write it as 8 − 1 (`IE`), to shorten the numeral sequence.

Thus, there must **never be more than two** instances of a single symbol next to each other in a Wolfie numeral. Also, only the numerals that are powers of 8 can be repeated. Moreover, it is disallowed for a symbol to be used in an additive manner *after* it has been used in a subtractive manner. To see why this is true, consider Roman numerals for a moment. `CXC` (190) is valid because `C` is used first in an additive way and *then* in a subtractive way. But `XCXX` is invalid because `X` is first used in a subtractive way and *then* in an additive way. In Wolfie numerals, examples of similar invalid cases would be expressions like `IFII`, `ETE`, and `ESEE`.

Based on the rules described above, an updated table of the numerals can be described as below:

Symbol:	I	IF	F	IE	E	ET	T	ES	S
Value:	1	3	4	7	8	24	32	56	64
Max # of Appearances:	2	1	1	1	2	1	1	1	2

In `Assign8Answer1.py` complete the function `arabic2wolfie` that takes one parameter, `num`, which is an **Arabic** numeral that you will need to convert to the Wolfie numerals. Your function should return the Wolfie numeral as a string.

Note: You will only be given `num` in the range `[1, 63]` (inclusive of both).

Below are some examples to help you understand. Keep in mind that you want the sequence to be *as short as possible* while following the numeral rules.

Example #1: `num = 10`

Look from the larger Wolfie numerals to the smaller numerals. What is first value you can subtract? It's 8. Now, you have 2 left to worry about, and the only applicable value is 1. Since 1 can appear twice, we have two 1's.

Therefore, 10 = 8 + 1 + 1, and this translates to `EII`.

Example #2: `num = 14`
Look from the larger Wolfie numerals to the smaller numerals. What is first value you can subtract? It's 8. Now, you have 6 left to worry about.

Applying the same technique, 6 can be translated to 4 + 1 + 1. Therefore, 14 = 8 + 4 + 1 + 1, which translates to `EFII`.

Example #3: `num = 22`
Applying the same technique, first we can subtract two 8's because `E` can be repeated. Then, we have 6 to worry about. From the last example we know that it can be expressed by 4+1+1. Therefore, 22 = 8+8+4+1+1, which translates to `EEFII`.

Example #4: `num = 28`
Applying the same technique as above, 28 = 24 + 4, which translates to `ETF`.

Example #5: `num = 30`
Applying the same technique as above, 30 = 24 + 4 + 1 + 1, which translates to `ETFII`.

Example #6: `num = 54`
Applying the same technique as above, 54 = 32 + 8 + 8 + 4 + 1 + 1, which translates to `TEEFII`.

Test Cases:

```
arabic2wolfie(10) returns 'EII'
arabic2wolfie(14) returns 'EFII'
arabic2wolfie(22) returns 'EEFII'
arabic2wolfie(28) returns 'ETF'
arabic2wolfie(30) returns 'ETFII'
arabic2wolfie(54) returns 'TEEFII'
```

Part 2. Wolfie Numerals to Arabic Numerals Converter (4 points)

In `Assign8Answer1.py` complete the function `wolfie2arabic` that takes one string parameter, `numerals`, consisting of properly-formatted Wolfie numerals. Your function should return the integer represented by the Wolfie numerals. The converted integer is guaranteed to be in the range `[1, 63]`.

Process the input string from left-to-right, looking for combinations of numerals (possibly indicating subtraction) or single numerals that are not involved in subtraction. Add the value of the numerals to a running total as you process the string.

To find combinations of numerals, consider two characters simultaneously (e.g., `IE` or `ES`), taking care not to overrun the end of the string while doing this (an "index out of range" error). Proceed in this manner until the rightmost numeral has been processed.

Example Test Cases:

```
wolfie2arabic('EII')    returns 10
wolfie2arabic('EFII')   returns 14
wolfie2arabic('EEFII')  returns 22
wolfie2arabic('ETF')    returns 28
wolfie2arabic('ETFII')  returns 30
wolfie2arabic('TEEFII') returns 54
```

Problem 2: CSV File Processing

A comma-separated values (CSV) file is a text file that uses a comma to separate values. A CSV file stores tabular data, such as numbers or text in a spreadsheet, in plain text. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. Thus, the file format gets its name from using the comma to separate different values.

Refer to the section on "Parsing CSV Files With Python's Built-in CSV Library" from the tutorial at this link: https://realpython.com/python-csv/#writing-csv-file-from-a-dictionary-with-csv to complete this problem.

Part 1. Read a CSV file and print content (2 points)

For this problem you need to download and add the worldpopulation.csv file to your project.

Using the code given in the section 'Reading CSV Files Into a Dictionary With csv' as an example, write your own function called `readcsv()` which reads in the CSV file at the path 'worldpopulation.csv' and prints the following output for each country:

```
GI Gibraltar has a population of 34733 in 2018, and will have a
population of 35897 in 2030.
TC Turks and Caicos Islands has a population of 35963 in 2018, and
will have a population of 41528 in 2030.
```

For reading the CSV file, please note that each row read by the reader is a dictionary, and you will need to update the tutorial code to read the dictionary with the appropriate keys for the worldpopulation.csv file. The keys are the names in the first row (e.g. "Flag", "Country") of the CSV file.

Part 2. Process data and write content to other CSV file (3 points)

Refer to the section called 'Writing CSV File From a Dictionary With csv'. Create your own function called `writecsv()` in which you will copy and extend the same code that you wrote for the `readcsv()` function to perform the following:

1) Calculate the ratio of the 2030 population to the 2018 population for each country. For example, for South Korea, this ratio is 1.0300.
2) Write this ratio along with the other parameter values to a new file named `'worldpopulationchange.csv'`. The first few lines in this file should look like the following:

Flag	Country	Population2018	Population2030	Ratio
CN	China	1415045928	1441181813	1.018469991
IN	India	1354051854	1512985207	1.11737612
US	United States	326766748	354711670	1.085519479
ID	Indonesia	266794980	295595234	1.10794901

For solving part 2, one approach is before reading data in the worldpopulation.csv file, create an output CSV file and set its headers using the following code:

```
output_file = open('worldpopulationchange.csv', mode='w')
fieldnames = ['Flag', 'Country', 'Population2018', 'Population2030',
'ratio']
output_writer = csv.DictWriter(output_file, fieldnames=fieldnames)
output_writer.writeheader()
```

Then when you are reading each dictionary `row`, you can create a new key 'ratio' and assign it a value using the following logic:

```
row['ratio'] = float(row["Population2030"])/ float(row["Population2018"])
```

Part 3. Use Matplotlib library for graph rendering Extra Credit (3 points)

This part of the problem is extra credit and in it you will plot the data from the CSV file.

Matplotlib library (https://matplotlib.org/) in Python can be used to render a variety of charts. Your tasks are the following:

a. Install Matplotlib library on your system. See installation instructions at: https://matplotlib.org/users/installing.html – if the commands in the instructions does not work, you may want to try typing "pip3 install matplotlib" in your Command Prompt / Terminal.
b. Go through examples of bar charts given at: https://matplotlib.org/gallery/index.html
c. Extend your earlier code to display two separate bar charts for the following (See example outputs):
    i. Top 20 countries with the greatest population growth (in percent) from 2018 to 2030; and
    ii. Top 20 countries with the greatest population loss (in percent) from 2018 to 2030. Note that you will need to convert the ratio from before into a percentage for the growth/loss of the population of a country.

You may want to try with different colors or different formats for plotting these graphs. Example graphs are below:

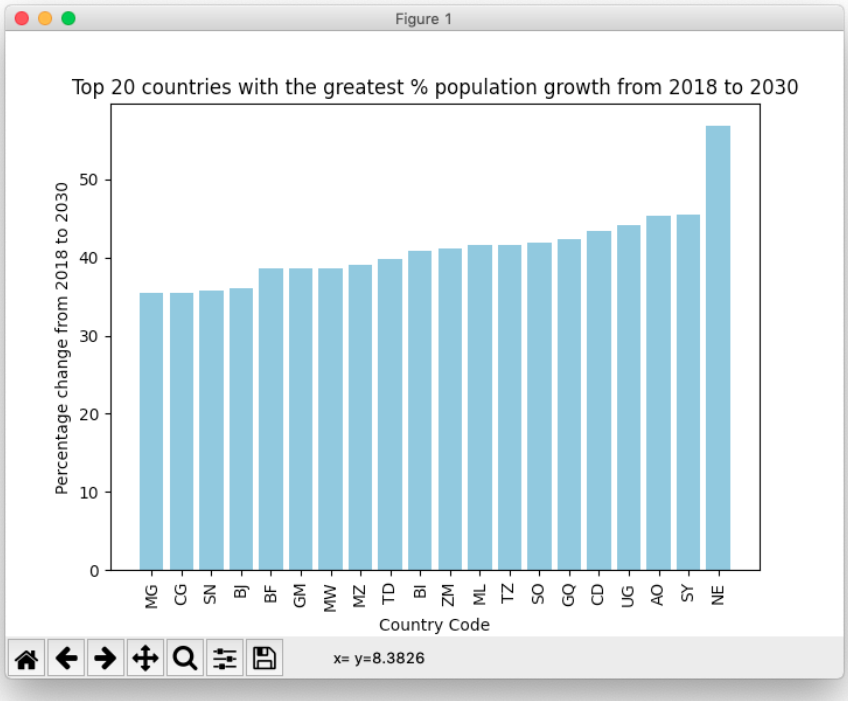
Figure 1
Top 20 countries with the greatest % population growth from 2018 to 2030
Percentage change from 2018 to 2030
0
10
20
30
40
50
MG
CG
SN
BJ
BF
GM
MW
MZ
TD
BI
ZM
ML
TZ
SO
GQ
CD
UG
AO
SY
NE
Country Code
x= y=8.3826

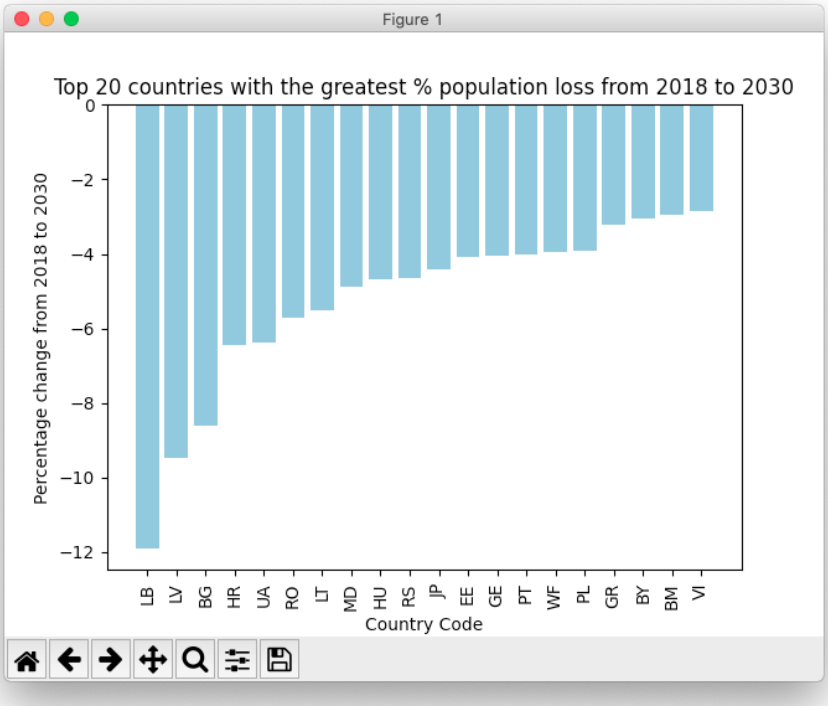
Figure 1
Top 20 countries with the greatest % population loss from 2018 to 2030
Percentage change from 2018 to 2030
0
−2
−4
−6
−8
−10
−12
LB
LV
BG
HR
UA
RO
LT
MD
HU
RS
JP
EE
GE
PT
WF
PL
GR
BY
BM
VI
Country Code