Lab 12 – CSE 101 (Spring 2021)

Objectives

The primary objectives of this lab assignment are:

- To get practice using regular expressions
- To get practice working with comma-separated values (CSV) files

1. Creating Regular Expressions (3 points)

For the problems below, you will create a text file called `Lab12.txt` and submit the regular expression answers in this text file. You should use a site like https://regex101.com/ to test your answers.

A. Create a regular expression that matches every instance of the word `"grey"` or `"gray"`. It should not match any other words.

B. Create a regular expression that matches any date in the format:

```
YYYY.MM.DD
```

The expression should match any digits in this specified format and does not need to check that the date is actually valid.

Below are some examples of **valid** instances:

```
2021.05.15
1988.12.30
```

Some examples of **invalid** instances:

```
2021.5.15
2000-03-15
2050.25.a1
```

C. Create a regular expression that will check whether an email address is a valid `.com` or `.edu` email address.

Assume every email address will be made with all lowercase letters and will only be in the format of:

```
<some text - any letters or numbers>@<some_domain>.com  OR
<some text - any letters or numbers>@<some_domain>.edu
```

The program should **not** match any email addresses that do not end with `.com` or `.edu`

Some examples of **valid** addresses are:

```
test123@stonybrook.edu
some.name@gmail.com
a@b.com
```

Some examples of **invalid** addresses are:

```
shouldfail@home.org
no spaces allowed @gmail.com
```

2. Using CSV Files

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.

Create a file `lab12.py` and submit your programs for problem 2 in this file.

Part A: Read a CSV file and print content (2 points)

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

Refer to the section on "**Reading into a dictionary"** from the tutorial at https://python-adv-web-apps.readthedocs.io/en/latest/csv.html to complete this problem. You will be reading each row in the CSV file into a dictionary. Using the code given in the section "**Reading into a dictionary**" as an example, write your own function defined below:

```
def read_statistics(year):
```

This takes a string parameter `year` and reads the `global_development.csv` file and goes through the CSV file for all rows matching the given `year` and prints out
1) the country name
2) the population in that year, and
3) The number of mobile cellular subscriptions per 100 people.

For example, given the following test case:

```
read_statistics("2013")
```

The first few lines of output should look like:

```
Name, 2013 Population, 2013 Cellular Subscriptions per 100 people
Canada, 34754312, 80
Sao Tome and Principe, 188098, 65
Lao PDR, 6645827, 65
Arab World, 362466629, 105
... (many more lines for each additional country)
```

Note that the values should be rounded to the nearest integer, as shown above. Include the provided test case in your program.

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.

The keys you will use for this problem are: "`Year`", "`Country`", "`Data.Health.Total Population`", and "`Data.Infrastructure.Mobile Cellular Subscriptions per 100 People`".

Part B: Process data and write content to other CSV file (2 points extra credit):

For this problem you will implement the following function:

```
def write_statistics_change(initial_year, later_year):
```

This function will take in two parameters, a string for an initial year and a later year and calculate some statistics between the years. Then you will write the output to a new CSV file named "`output.csv`".

You can write to the file similar to printing with the following code:

```
f = open("output.csv", "w") # Opens a file for writing named "output.csv"

f.write("Whatever you place here will appear in the file\n")
f.write("Another line")       # Write as many times as needed
f.close() # When you are done with all your writing, close the file
```

The data you will calculate and write to the CSV file, in order, is:

1) The country name
2) The population in `later_year`
3) The population ratio of `later_year` from `initial_year` (divide the value from `later_year` by `initial_year`)
4) The number of cellular subscribers per 100 people in `later_year`
5) The increase in cellular subscribers per 100 people from `initial_year` to `later_year`

For example, in this test case:

```
write_statistics_change("2003", "2013")
```

The first few lines of output in the CSV file would be:

```
Name, 2013 Population, Population Growth since 2003, 2013 Cellular Subscriptions
per 100 people, Cellular increase since 2003
"Canada", 34754312, 1.11, 80, 42
"Sao Tome and Principe", 188098, 1.3, 65, 64
"Lao PDR", 6645827, 1.2, 65, 64
"Arab World", 362466629, 1.25, 105, 97
```

An example of the full expected CSV file for the above test case is output.csv – include this test case in your program.

You can assume the `initial_year` will always be valid and will be earlier than the `later_year`. Note that in the CSV file the earlier years are shown first, so you will need to store this information for later use when you read the later year's information. Then when you see the later year's information, you can write to the file the calculated information for that country.

Also note that some countries have names with a comma in them, which can mess up a CSV file's formatting. So you should write all the country names with quotes (") around them that are saved into the CSV file, as shown in the example output file.

3. Submission

Submit the following programs on blackboard:

1) Completed `lab12.txt`
2) Completed `lab12.py`