CS 1120: Media Computation Spring 2017
Lab 10: Strings, lists and files

Description

In today's lab, you will work with strings and lists in the context of reading and writing csv files. CSV stands for comma-separated values; a csv file is a plain text file which contains data in csv format: rows separated by newline characters, with each row containing values separated by commas.

A csv file can be viewed and edited in any spreadsheet application (like Microsoft Excel, LibreOffice Calc, or Google Sheets). However, since it is nothing but text, it can be also viewed and edited in any text editor. Most importantly, it can be accessed programmatically: we can write a program to create, read, or modify such files - which is what you will do tin today's lab.

Download 2 files: lab10-cities.csv and lab10-states.csv. Open these files in a text editor of your choice, and in a spreadsheet program (e.g., Excel). As you will see, a spreadsheet program makes it easy to view the data; however, you should remember that this is just a visual representation of the data; and that data is nothing but rows of comma-separated values - i.e., what you see in your text editor.

We will refer to the data in your files as data sets (i.e., a set of data). Today you'll write 2 functions that will query the lab01-cities data set. You will also write a function that will create a new csv file, which will contain a modified version of the lab10-states data set.

Task 1

Write a function, **findCityPopulation(city, state)** that **returns the population** of the passed city. So your task is to query the lab10-cities data set to find a city's population.

As a reminder, here's how you read in the contents of a file:

```
f = open(getMediaPath('lab10-cities.csv'), "r")
lines = f.readlines()
f.close()
```

As a result, the variable **lines** contains a **list** of lines from your csv file. Don't forget to close the file as soon as you have read in its contents into a variable.

Now we can iterate over the lines, splitting each line and using index notation to access the value we need:

```
for line in lines:
  cells = line.split(",")
  if cells[7].lower() == city.lower() and cells[8].lower() == state.lower():
    return int(cells[14])
return -1
```

Pay attention that we convert the value and the passed argument to lowercase:this makes our function a little more flexible: we may not know or care how the city is recorded in the data set (lowercase or. title case or. uppercase, etc.).

Returning -1 if the value was not found is good practice: our code will expect an integer, and -1 will signal an error, which we can then handle (display an error message, ask for input again, etc.)

Test your function on several names. Pay attention to how cities are listed in the data set (e.g., "Waterloo" is "Waterloo city").

Task 2

Write a function, **findLargestCityInState(state)** that prints the name and population of the city with the largest population in a state. For example:

```
>>> findLargestCityInState('iowa')
The city with the largest population in iowa is Des Moines city with a population of 207510
>>> findLargestCityInState('new york')
The city with the largest population in new york is New York city with a population of 8405837
>>>
```

So your task is to query the lab10-cities data set to find the city with the largest population in a given state.

In this case, you will need to keep track of two variables, which you initialize before you start your iteration:

```
city = ""
city_population = 0
```

Use the last column for the population value: POPESTIMATE2013. Also, you will need to add one more test: the data set contains records for counties - we don't need those. So check that the name of the "city" does not contain the word "county" (otherwise your largest "city" in Iowa will be Polk county.

```
if cells[8].lower() == state.lower() and 'county' not in cells[7].lower():
  population = int(cells[14])
```

Remember: we are dealing with text - so the data we have read in from the file is strings. Therefore, before we can do any arithmetic operations on values that appear to be numbers, we should convert them to the appropriate data type - hence, we use `int(cells[14])`.

To find the largest population, use the same approach we used to find the largest sample value. However, you cannot use the max() function because you need to check explicitly if the value at the current step is greater than the running maximum, and if so - update the city name too!

Task 3

Write a function, **updateStateFile()**

In this function, you will create a new csv file that will contain a modified version of the lab10-states data set. YOu will add the following columns to the data set:

LARGEST_CITY: the name of the city with the largest population in the state
LARGEST_CITY_POPULATION: the population of that city
STATE_PERCENTAGE: the percentage of the state population in the country's total population
CITY_PERCENTAGE: the percentage of the city population in the country's total population

For example, according to our data, the population of the state of New York is 15,411,151. The largest city is "New York city" (casing as per data set) with a population of 8,405,837. The state's population accounts for which accounts for 6.23% of the total population, whereas the population of the city accounts for 2.66% of the total population (percentages are rounded to two decimal places).

Implementation. Here's our algorithm:
1. create an empty list (list A) to hold the new data set

2. calculate the total population and store it in a variable. It's best to write a helper function for this, so you can call it from your main function (see implementation tips below)

```
total = getTotalPopulation() * 1.0
```

2.1. We multiply the result by 1.0 to convert the data type to a float: we will be using this to calculate percentages, so we don't need integer division.

3. read in the current dataset into a list (list B)

4. iterate over the data (list B):

4.1. remove the newline character from the current line:

```
for line in lines:
  line = line.strip() #remove the new line character ("\n")
```

4.2. split the line into a list (just like you did in the previous functions)

```
cells = line.split(",")
```

4.3. check if we are at the header row; if so - append the new column titles:

```
if cells[0] == "SUMLEV":
  cells.append("LARGEST_CITY")
  cells.append("LARGEST_CITY_POPULATION")
  cells.append("STATE_PERCENTAGE")
  cells.append("CITY_PERCENTAGE")
else:
```

4.4. for all other rows:

4.4.1. Get the values you will need:

```
population = int(cells[5])
state = cells[4]
```

4.4.2. Inform the user what row you are processing:

```
print("Processing " + state)
```

4.4.3. Modify the function that finds the largest city. Our previous version printed a string - which is not very useful in this scenario. Let's make it into a more useful "building block": remove the print statement and instead add a return statement that returns a list the has 2 values: the city name and its population: (note the square brackets)

```
def findLargestCityInState(state):
  .....
  return [city, city_population]
```

4.4.4. Now use the modified function to get the data we need:

```
city_data = findLargestCityInState(state)
city = city_data[0]
city_population = int(city_data[1])
```

4.4.5. Calculate the percentages:

```
state_pop_percent = population / total * 100
city_pop_percent = city_population / total * 100
```

4.4.6. And append the new values, converting the integers and floats back to strings (we'll need them to be strings when we call the join() method)::

```
cells.append(city)
cells.append(str(city_population))
cells.append(str(state_pop_percent))
cells.append(str(city_pop_percent))
```

4.5. Append your current list of cells to the new data list (list A):

```
newdata.append(cells)
```

5. After the iteration, write the new data to a new file:

```
f = open(getMediaPath("lab10-newdata.csv"), "w")
for row in newdata:
  csv_line = ",".join(row)
  f.write(csv_line + "\n")
f.close()
```

Run your function! It should create a new file which you can then open in a Excel - it will be the same the same data set as in lab10-states, with 4 additional columns. You've just created a spreadsheet with new useful data - congratulations! Here's what the new columns of your modified data set may look like:

	H	I	J	K	L
T18	PCNT_POI	LARGEST_CITY	LARGEST_CITY_POPULATION	STATE_PERCENTAGE	CITY_PERCENTAGE
151	78.4	New York city	8405837	6.216176627	2.658990881
644	76.1	Los Angeles city	3884307	12.12560079	1.228710108
828	76.5	Chicago city	2718782	4.074963563	0.860023403
207	73.4	Houston city	2195914	8.366270247	0.694626282
156	78.7	Philadelphia city	1553165	4.040694623	0.491307596
810	75.6	Phoenix city	1513367	2.096178261	0.478718425
875	75.9	Indianapolis city	852866	2.078551903	0.269784308
186	79.4	Jacksonville city	842583	6.185092148	0.26653152
978	77.1	Columbus city	822553	3.660155789	0.260195496
455	76.8	Charlotte city	792862	3.115204557	0.250803439
421	77.3	Detroit city	688701	3.130249689	0.217854531
401	77	Nashville-Davidsoı	658602	2.05485144	0.208333413
611	77.1	Seattle city	652405	2.205242022	0.206373136
435	76.5	Denver city	649495	1.666525274	0.205452626
975	82.8	Washington city	646449	0.204489094	0.204489094
878	79.2	Boston city	645966	2.11711909	0.204336309
292	77.3	Baltimore city	622104	1.87544231	0.19678812
541	75.4	Oklahoma City city	610613	1.218037561	0.19315321
459	78.2	Portland city	609456	1.243184587	0.19278722

How to calculate the total population

A helper function will open the same file, create an accumulator variable, iterate over the data, adding the value of column POPESTIMATE2013 (sixth column) to the accumulator, and finally return the accumulator's value. Remember to check if you are not at the heading row; also remember to convert the value of the cell to an integer:

```
if cells[0] != "SUMLEV":
  total += int(cells[5])
```

Submit your work

And you're done! Submit your lab10.py file to eLearning.