Assignment 6

Objectives: Practice working using lists and files, as well as their methods. Practice the Pythonic way to make `for` loops. Learn basic procedures to use csv files. Apply new tools to old problems.

Note: Include DocStrings in each script you submit analogous to the following:

```
"""Retrieves name and phone# of members of the house of representatives


Submitted by Mauricio Arias. NetID: ma6918
This script scans a csv document for the correct information. It
processes it using lists and strings
"""
```

Part 1. Re-making towers with many colors.
For this part we will modify the solution I provided for the tower problem: we will use a list to store 10 different colors of your choice. Notice that in the template there are 2 TODO sections: the first one to add the input sections and the second one to add the code for the tower. Use pixel arithmetic.

Regarding the input, the user will provide a single height for each of the rows (for example 100 pixels) and the width of the tower (for example 200 pixels). (Using the values in the example, the tower will be 200 pixels wide by 1000 pixels tall.) Make sure you define the two size variables that are passed to the header. Notice that padding is already provided as `padding`; it should be added to each and every horizontal line of pixels.

Submit your script as `[NetID]_tower_v2.py` and the results as a bmp file `[NetID]_tower_v2.bmp`. (2 pts)

Part 2. How much?
A multidimensional list allows handling of multiple sets of related data together. For example, we can represent prices of some goods for different months in a year. One way to represent this is as follows:

Horiz_index →	0	1	2
Vert index ↓	Month	Price of Lemons	Price of Milk
0	Jan	0.76	3.99
1	Feb	0.75	3.99
2	Mar	0.78	4.09
3	Apr	0.80	4.09
4	May	0.77	4.19
5	Jun	0.70	4.29

Notice that indices are not stored per se. The values are. We access the information in the first column as `prices[month_index][0]`. (Putting the row or vertical index first is a convention.) That information corresponds to the name of the month in this case. It will be ‘May’ if `month_index` is 4. The third

column will be accessed by setting the column index to 2; if `month_index` is 4, `prices[month_index][2]` will be 4.19.

For this exercise, you will read a csv file and make a 2D list with all the information present. You will then ask the user to select a subset of columns and then generate a new file with only those columns. The script should then go back to asking for another selection of columns from the same file. An empty string ends the program. The following subtasks should guide you through the process. Make sure you test your code at the end of each subtask and for any difficult part of the code.

Task 2.1. Request the name of the csv file. Start reading it and obtain a clean version of the labels. (1 pt)

a) Write code that asks for the file name and opens the text file.
b) Process the header line by removing any special characters at the end of the line.
c) Generate a list with all the column labels in the header line. Use the split method for strings with "`\t`" as the delimiter.

Task 2.2. Make a list and populate it with all the information in the table. (1 pt)

a) Make a new empty list to contain the table and call it `price_table`.
b) Split the information in each row to generate a list.
c) Append the list generated by each row to the table. Notice that you are appending a list at each position generating therefore a table with rows and columns. (Take this opportunity to explore what this "table" is like.)

Task 2.3. Request the columns the user is interested in. (1 pt)

a) Request a series of column numbers the user wants to keep; the user should input these numbers separated by commas. Column 0 should always be displayed even if the user does not request it. The row with the labels too. Repetitions are allowed but each column is displayed only once.
b) Use a set comprehension to clean up the substrings: they might come with extra spaces.
c) Use a list comprehension to generate a curated list containing only numbers within range: between 1 and the total number of columns. Add 0 to this list.
d) Sort the list for presentation purposes.

Task 2.4. Write the desired output to a file. (1 pt)

a) Open a file for writing and name it "subset [index] of [original filename]". Where [original filename] is the filename of the original file including the extension and [index] is an index that increases for every iteration of the selection cycle. It starts at 1.
b) Make a list with the column labels for the columns to be saved.
c) Write in the file the column labels corresponding to the columns to be saved. Separate the names with a tab character: "`\t`". Use the join method of the string "`\t`" and give it the list of labels: `"\t".join(labels_saved)`.
d) For each row, make a list with the values to be saved.
e) Write the output in the file. Separate the different columns with a tab character and join it all into a string with the join method as done with the column labels.
f) Close the file and go back to request a new selection.

Submit your script as `[NetID]_prices_abridged.py`.

Part 3. House members.
In the file House_of_representatives_117.csv, a listing of all the members of the current house of representatives (Congress 117) is provided. Ask the user for a state and a district number and find the information for the representative by scanning the full file. Provide the full name in the format "given name" "family name" and the phone number of the representative. For example for the 12th district of New York (where NYU is located), the information is:

```
The representative for district 12 in the state of New York is Carolyn
Maloney. The phone number is (202) 225-7944.
```

If no match was found, report that:

```
No representative was found for district 5 in the state of Iowa.
```

Task 3.1. Initial processing steps. (To avoid issues due to capitalization, all strings will be converted to lowercase letters.) (1 pt)

a) Write code that asks for the file name and opens the text file.
b) Process the header line by removing any special characters at the end of the line.
c) Generate a list with all the column labels in the header line. Use the split method with "," as the delimiter.
d) Use a list comprehension to trim any spaces flanking these labels. Take the opportunity to put the labels in lowercase.
e) Record the position in the file.

Task 3.2. Identifying the columns for our purposes. (1 pt)

a) Write code to figure out which column contains the family name,
b) which column contains the given name,
c) which column contains the information for the district and
d) which column contains the phone number.

Task 3.2. Set up an infinite loop to perform searches.

a) Ask for the state of interest. If the answer is an empty string exit the script.
b) To handle capitalization better, store two copies of this query: the original one and another one in lowercase.
c) Ask for the district ID. This ID can be a word or a number: see below.

Task 3.3. Obtaining the string in the district column. (1 pt)
The district column contains both the state information as well as the district information within the state in a single string. For the "state" the possibilities are the 50 states, American Samoa, District of Columbia, Guam, Northern Mariana Islands, Puerto Rico and Virgin islands. There are 4 possibilities for the district ID: the district number (e.g., 4th), "At Large," "Resident" or "Delegate." From this column the state and district information should be extracted. Write code that selects the district column as follows:

a) Read the next row of information in the file.
b) Separate the information into a list using the split method.
c) Use a list comprehension to trim each element of any "invisible" characters.
d) Obtain the information about the district.

Task 3.4. Obtaining the state and district ID. (1 pt)
Processing unstructured strings is complex. We will handle it in cases but first separate the string into a list of words.

a) Use the split method to obtain the words. (The delimiter is a space: my_string.split(“ ”))
b) If the last word is Large, the last two words are “At Large.” The rest is the state name. Join them using the join method for strings.
c) If the last word is either Delegate or Representative, the rest is the state name. Join it!
d) Otherwise it is a numbered district. The number is contained in the last word. Remove all characters which are non-numeric. (Look for an appropriate method in the list provided by PyCharm or using dir(str).) The state is the rest.

Task 3.5. Finishing up. Use the original queries (1 pt)

a) Compare the state and the district.
b) If it is a match to the query print the results and break the loop.
c) Use the `else` clause of the loop if no match was found to report that.
d) Reset the file to allow processing a new query as described in Task 3.2 above.

Submit your script as `[NetID]_house_directory.py`.