CIS 41A - Lab 3: containers, files

In the *spirit* (pun intended) of the season, write a program to let the user look up the most popular Halloween candies.

**Input file**

Download the 2 files: candies.txt and state.csv

candies.txt is a CSV file, where each line has the 3 most popular candies for one US state, in the following format:   
 *State abbreviation,#1 Candy,#1 pounds,#2 Candy,#2 pounds,#3 Candy,#3 pounds*

Example of one line: AL,Skittles,105263,Starburst,101928,Hershey's Mini Bars,93728

[candies.txt source: the State by State Data Table at <https://www.candystore.com/blogs/facts-trivia/halloween-candy-map-popular?y=2022>]

state.csv is a CSV file, where each line has the state name and abbreviation for one US state, in the following format: *State name,State abbreviation*   
Example of one line: Alabama,AL

[state.csv is downloaded from <http://worldpopulationreview.com/states/state-abbreviations/> ]

**Lab requirement**

A. The program has 3 global constants that are defined for: the 2 input filenames and 1 output filename.   
Set the output filename constant to *lab3out.txt*  
When accessing the files, make sure you use the constants and not the actual filename.

B. The program also has the following 5 functions. And it’s recommended that you write the 5 functions in the following ordered steps.

1. readFile(): read both input files into containers.

* Read in data from state.csv and store in a dictionary, where the state name is the key, and the state abbreviation is the value.  
  *[1 pt extra credit (EC) if you use comprehension to read data into the dictionary]*
* Read in data from candies.txt and store in another dictionary. For each line of data:
* Read in the abbreviation as the key
* Loop to get one candy name and its number of pounds, then store the pair of data into a tuple.  
  Then store the tuple into a list, which is the value for the state abbreviation (the key).   
  --The resulting dictionary will be made of key:value pairs as followed:

state abbreviation : [ (No.1 candy, pounds), (No.2 candy, pounds), (No.3 candy, pounds) ]

-- Repeat the loop until you've stored all 3 tuples in the same list.  
*[Challenge: don't hard code the 3 for the number of tuples, your code should be able to work with any number of candy-pounds pairs]*

* In the same loop that stores the candy data in the dictionary above, create another container to store each unique candy name and the total number of pounds of that candy for *all* states.  
  -- This container is your choice, choose the correct one and your code will be shorter.
* After readFile() is done, you should not open the input files again since all data should have been saved in memory already.

1. printState(): print all candies and their number of pounds for one given state.

* Accept a state name as input argument
* Use the state name:abbreviation dictionary from step 1 to convert the name to an abbreviation
* Use the abbreviation to find and print the matching state name and the state's candies and numbers of pounds
* The candies must be printed in their ranking order (1, 2, and 3), and all data are lined up in column format.   
  Use string formatting to create columns and to add commas to the number of pounds. See sample output.
* If a state can't be found, print an error message.
* The input state name is case *insensitive*, so any combination of uppercase and lowercase should work.

See sample output

*[1.5pts EC if you can demo your code with the 2 function calls in green above by Thu Oct 20, end of office hour time. See sample output to check your output data. The pseudocode to demo your code is:*

*call readFile -- don’t need to have code for the comprehension EC*

*call printState and enter a valid state name to see the state name and all 3 candies and pounds*

*call printState again and enter a different valid state name to see the state name and all 3 candies and pounds]*

1. printCandiesV1(): print all candies and the total number of pounds of each candy.

This is to test your code’s calculation and storing of the total number of pounds of each candy.

1. printCandies(): print all candies and the total number of pounds of each candy, sorted in order from highest total weight (pounds) to lowest total weight.

* Add more code to the printCandiesV1() function from step 3, now that you know the data storage works.
* printCandies() accept a limit of the top candies by weight, where the limit is guaranteed to be at least 1.
* Sort the total pounds of each candy from highest to lowest weight and print the top ‘limit’ number of candies and their weights. Example of using the limit input argument:  
  - If limit is 2: print the 2 candies and total pounds that are the top 2 total pounds   
  - If limit is 15: print the 15 candies and total pounds that are the top 15 total pounds  
  - If limit is larger than the total number of candies: print all the candies and their total pounds
* Format the printing so it’s 1 candy and its total pounds per line, in columns, and with commas for the total pounds. The candies and pounds are in descending sorted order by pounds. See sample output.
* Please write your own code for this sorted print, it is both a problem solving exercise and an exercise in working with containers. Don’t look up online and use a lambda function that we haven’t talked about, lambdas are covered in Adv Python.

1. Additional code for printState() : save the user’s unique search results to an output file.

* Each time the user searches for a state, store the user’s chosen state and its candies (but not the pounds) in a container:
  + The same container is used for all searches within one run of the program.
  + The container only stores unique searches, for example:  
     The user chooses California: California and its 3 candies are stored in the container.   
     Then the user chooses Colorado: Colorado and its 3 candies are stored in the container.  
     Then the user chooses California again: nothing is stored in the container.
  + Each data in the container is a string in the format: “state name: candy1, candy2, candy3”
  + Choose the container wisely and your code can be simpler and shorter.
  + This container is used for step 6.

1. saveFile(): save to file all unique searches by state data that the user did, when the user quits from the program

* Write to the output file (use the constant defined above) the strings in the container described in step 5.
* The output file has the format: state name: candy1, candy2, candy3  
  where each line is for one state, and there should not be any duplicates of the state name. The state names can be listed in any order you choose (alphabetical, random, …)

1. main(): coordinate all the function calls and provide the user interface  
   Pseudocode for the main function:

* Call readFile
* Loop to prompt the user with a menu:

c. search by candy

s. search by state

q. quit

* Re-prompt until there is a valid choice
* If the choice is 'c': Prompt for a number  
   Check that the number at least 1 or print an error message  
   Check that the number is not larger than the max number of candies or set it to the max  
   Call printCandies
* If the choice is 's': Prompt for a state name  
   Check that it’s a valid name (case insensitive) or print an error message  
   Call printState
* If the choice is 'q': Call saveFile  
   Print a message to let the user know about the output file or that there’s no output file

Then the program should end

Additional notes:

* You can have more functions than the 5 functions described above. But you should have at least these 5, and they should work as described.
* You can use break and continue when necessary.

C. Documentation

- Create a beginning comment block with your name and lab number

- Write a docstring for each function except for main

**Test**

Sample program output below. Use it as a sample test case for your code. The user input is in green.

Search data by

s. state

c. candy

q. quit

Enter your choice: California # wrong choice

s. state

c. candy

q. quit

Enter your choice: s

Enter state name: califORNIa # case insensitive

1. Reese's Cups 1,231,675 # column format with commas for numbers

2. M&M's 1,219,928

3. Skittles 1,157,821

Search data by

s. state

c. candy

q. quit

Enter your choice: s

Enter state name: washington

1. Tootsie Pops 167,289

2. Salt Water Taffy 120,982

3. M&M's 113,728

Search data by

s. state

c. candy

q. quit

Enter your choice: s

Enter state name: California # search same state again

1. Reese's Cups 1,231,675

2. M&M's 1,219,928

3. Skittles 1,157,821

Search data by

s. state

c. candy

q. quit

Enter your choice: s

Enter state name: DELAWARE

1. Sour Patch Kids 20,083

2. Skittles 16,728

3. Life Savers 11,292

Search data by

s. state

c. candy

q. quit

Enter your choice: s

Enter state name: phoenix # invalid state name

not a valid state

Search data by

s. state

c. candy

q. quit

Enter your choice: c

Enter top limit: 0 # invalid limit

minimum limit is 1

Search data by

s. state

c. candy

q. quit

Enter your choice: c

Enter top limit: 2

Reese's Cups : 3,548,476 # print in columns, with commas in numbers

M&M's : 2,813,637

Search data by

s. state

c. candy

q. quit

Enter your choice: c

Enter top limit: 100 # above max number of candies

printing max limit 24 candies

Reese's Cups : 3,548,476

M&M's : 2,813,637

Skittles : 2,568,337

Hot Tamales : 2,293,393

Starburst : 1,770,779

Sour Patch Kids : 1,420,289

Hershey Kisses : 1,157,419

Snickers : 1,129,797

Tootsie Pops : 1,086,056

Candy Corn : 836,092

Hershey's Mini Bars : 832,061

Butterfinger : 688,862

Jolly Ranchers : 577,616

Blow Pops : 303,214

Salt Water Taffy : 277,596

Twix : 209,693

Swedish Fish : 177,654

Milky Way : 174,405

Kit Kat : 135,672

Dubble Bubble Gum : 110,271

Lemonheads : 109,826

3 Musketeers : 91,820

Almond Joy : 44,883

Life Savers : 11,292

Search data by

s. state

c. candy

q. quit

Enter your choice: q

Your state candy data is in lab3out.txt

Screen capture of lab3out.txt from the search of states above, which include:

California, Washington, California, Delaware

Text

Description automatically generated