CS21 Assignment #10 Submit three files: *myfns.py, electoral_dictionaries.py, electoral_sets.py*
Please use the given name for each file.

Step 0 Before you begin, read this background information

1. You are using a new approach for packaging user-defined functions

   - For this assignment, you will create a separate file, *myfns.py*, to hold your user-defined functions.
   - A file serving this purpose is called a **module**.
   - You will import your module and use these functions in *electoral_dictionaries.py* and *electoral_sets.py.*

   Note: You have done this in earlier activities with the modules *math* and *random*. Do not include the .py extension!

   This time: `import myfns`

   - When you invoke (call) your own functions, use the same syntax as you've done thus far, i.e., state the name of the module, followed by a dot, then the name of the function you want to use.

   Example: `states = myfns.make_elector_dictionary('electoralvotes.txt')`

2. About *electoralvotes.txt*

   You will use this plain text file to populate a *states* dictionary. The file contains information about each U.S. state plus the District of Columbia, including a) name, b) population (2010 U.S. Census), and c) number of electoral votes that state casts in the Electoral College (used to determine the outcome of the U.S. presidential election).

Step 1 Create your functions module

(*myfns.py*) Define each of the following functions.

Here is the information you need to define each of your functions:

	I	P	O
function	**Input argument(s)**	**function tasks (Processing)**	**return (Output)**
`make_elector_dictionary`	Name of text file providing census and electoral votes data	Create an empty dictionary. Try opening the file name. If `IOError`, display message. Otherwise, read each line from the file and create a dictionary entry for each locale. key = name value = 2-element list [popn,votes]	dictionary

	I	P	O
function	**Input argument(s)**	**function tasks (Processing)**	**return (Output)**
`make_category_sets`	The states dictionary	Create six empty sets. For each key in the dictionary, compute the ratio of population to number of votes. Using this ratio, add each state to its correct set: s1: ratio < 250,000 s2: 250,000 ≤ ratio < 350,000 s3: 350,000 ≤ ratio < 450,000 s4: 450,000 ≤ ratio < 550,000 s5: 550,000 ≤ ratio < 650,000 s6: ratio ≥ 650,000	Six sets of state names

Notes:

- To return multiple values from a function:

```
return s1,s2,s3,s4,s5,s6
```

- To call a function that returns more than one variable:

```
lt250K,lt350K,lt450K,lt550K,lt650K,gt650K = myfns.make_category_sets(states)
```

	I	P	O
function	**Input argument(s)**	**function tasks (Processing)**	**return (Output)**
`print_set`	The set whose contents you wish to display	Print each member of the set on a separate line	This function does not return anything

(*Still in Step 1*) **Test your functions – this is NOT submitted – you will delete this testing code after you know your functions are working properly.**

In `myfns.py`, define your `main()`, the test driver that uses the strategies described in the table.

Remember, you are only using this particular main TEMPORARILY, in order to TEST your functions.

Once you've verified all three functions, you should **delete main** altogether (both its definition and call).

To test this function:	Do this in main():
`make_elector_dictionary`	a) Call the function. b) Use one print statement (no loop) to display the dictionary returned by the function.
`make_category_sets`	a) Call this function with the dictionary you just created. b) Use one print statement (no loop) to display one of the sets returned by the function.
`print_sets`	a) Call this function with one of the sets you just created.

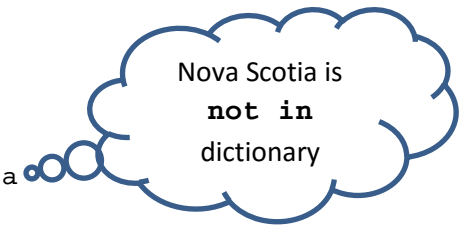
Step 2 Write your first script

(*elector_dictionaries.py*) In this script, in `main()`, you will create a states dictionary by calling `make_elector_dictionary` as described above. Once you've done this, loop through the dictionary to compute the total number of electoral votes (half the total votes + 1). Then compute the number needed to win, and display the following output:

```
The total number of electoral votes is 538
The total number of electoral votes needed to win is 270
```

Finally, allow the user to query for state information until done, while you produce the output as shown.

```
Choose a state or just hit Enter when done:  Vermont
Population:  630337   Electoral votes:  3

Choose a state or just hit Enter when done:  Nova Scotia
Try again.  This is not a valid state name.

Choose a state or just hit Enter when done:  New Mexico
Population:  2067273   Electoral votes:  5

Choose a state or just hit Enter when done:  Montana
Population:  994416   Electoral votes:  3

Choose a state or just hit Enter when done:

Thanks for using my program.
```



Step 3 Write your second script

(*elector_sets.py*) In this script, you will group states into different category sets, based on how many people it takes to earn one vote in the Electoral College. For example, using the numbers shown above,

Vermont = 630,337/3 = 210,112 people per vote, while New Mexico = 2,067,273/5 = 413,455 people per vote. Therefore, Vermont falls into the "less than 250,000" (LT250K) category while New Mexico falls into "less than 450,000" (LT450K) category.

In your script, start by calling `make_elector_dictionary` and `make_category_sets` as described above.

Then define a list of strings: 'LT250K', 'LT350K', 'LT450K', LT550K, LT650K, GT650. You will use a loop to display this list to the user (see next page for sample run).

Allow the user to select a category until done. For each selection, you should display the states in that set.

Once you've done this, allow the user to name a state until done. Determine which set the state is in and display the appropriate message.

SAMPLE RUN PROVIDED ON NEXT PAGE

Sample run for *elector_sets.py*:

```
1: LT250K
2: LT350K
3: LT450K
4: LT550K
5: LT650K
6: GT650K

Choose category 1-6, or just hit Enter when done: 3

Idaho
Nebraska
West Virginia
New Mexico

Choose category 1-6, or just hit Enter when done: 1

DC
North Dakota
Vermont
Alaska
Wyoming

Choose category 1-6, or just hit Enter when done:

Thank you for inspecting the sets.

Now check your state(s).

Name a state or just hit Enter to quit:  Vermont
Vermont has one vote per 250,000 people.


Name of your state or just Enter when done:  Nova Scotia
Try again.  This is not a valid state.

Name of your state or just Enter when done:  New Mexico
New Mexico has one vote per 450,000 people.


Name of your state or just Enter when done:
>>>
```

Reminders (not following will result in point deductions):

- It is expected that you will use exception handling and error handing to ensure your programs are robust and will not crash.
- If a function's task doesn't include output to the user, do not do it!
- Use constants! No magic numbers! No global variables!
- It is expected that you will complete the same process of development that we use in class. When you reach the point of having an algorithm (pseudocode), this will become the comments of your program as a starting point for writing code. Comment first, then code!
- Be sure to include comments at the top of the program that include your name, class and a short description of the program.
- Each function should begin with a comment describing the task the function will perform.
- Be sure all output is formatted. Unless otherwise, specified, displays non-integer values with 2 digits after the decimal point.

Any work you submit for this assignment should be authored entirely by yourself. Assistance is permitted from the instructor or teaching assistants only. All submitted programming assignments are subject to originality verification through software designed and used for the Measure Of Software Similarity (MOSS).