CSE 231 **Spring 2016**

Computer Project #07

Assignment Overview

This assignment focuses on the implementation of Python programs to read files and process data by using lists and functions.

It is worth 50 points (5% of course grade) and must be completed no later than 11:59 PM on Monday, March 21.

Assignment Deliverable

The deliverable for this assignment is the following file:

`proj07.py` – the source code for your Python program

Be sure to use the specified file name and to submit it for grading via the **handin system** before the project deadline.

Assignment Background

One commonly hears reference to "the one percent" referring to the people whose income is in the top 1% of incomes. What is the data behind that number and where do others fall? Using the National Average Wage Index (AWI), an index used by the Social Security Administration to gauge a individual's earnings for the purpose of calculating their retirement benefit, we can answer such questions.

In this project, you will process AWI data. Example data for 2014 is provided in the file `year2014.txt` (2014 is the most recent year of complete data). The data is a table with the first row as the title and the second row defining the data fields; remaining rows are data. The URL for the data is: https://www.ssa.gov/cgi-bin/netcomp.cgi?year=2014

Here is the second line of data from the file followed by descriptions of the data. Notice that some data are ints and some are floats:

```
5,000.00 - 9,999.99  13,848,841  36,423,281  23.02549  102,586,913,092.61  7,407.62
```

Column 0 is bottom of this income range.
Column 1 is the dash separating the bottom of the range from the top.
Column 2 is the top of this income range.
Column 3 is the number of individuals in the income range.
Column 4 is the cumulative number of individuals in this income range and all lower ranges.
Column 5 is the Column 4 value represented as a cumulative percentage of all individuals.
Column 6 is the combined income of all the individuals in this range of income.
Column 7 is the average income of individuals in this range of income.

Assignment Specifications

1. The program must provide following functions to extract some statistics. Note that the `data_list` parameter specified in these functions may be the same for all functions or different for different functions—that is your choice.
   a) `open_file()` prompts the user to enter a year number for the data file. The program will check whether the year is between 1990 and 2014 (both inclusive). If year number is valid, the program will try to open data file with file name 'yearXXXX.txt', where XXXX is the year. Appropriate error message should be shown if the data file cannot be opened or if the year number is invalid. This function will loop until it receives proper input and successfully opens the file. It returns a file pointer and year.
      i. **Hint**: use string concatenation to construct the file name
   b) `read_file()` calls the `open_file()` function and uses the returned file pointer to read the data file. This function returns a list of your choosing containing data you need for other parts of this project, and the year (either separately or as part of your data structure).
   c) `get_range(data_list, percent)` takes a list of data (of some organization of your choosing) and a percent and returns the salary range (Columns 0 and 2) for the data line whose cumulative percentage (Column 5) is greater than or equal to the `percent` parameter, the cumulative percentage value (Column 5) and the average income (Column 7).
      i. For testing using the 2014 data and a percent value of 90 your function will return `([90000.0, 94999.99], 90.80624, 92420.5)`
   d) `get_percent(data_list, income)` takes a list of data (of some organization of your choosing) and an income and returns the cumulative percentage (Column 5) for the data line that the specified income is in the income range (Columns 0 and 2), income range (Columns 0 and 2) and the average income (Column 7).
      i. For testing using the 2014 data and an income value of 150,000 your function will return `([150000.0, 154999.99], 96.87301, 152393.84)`
   e) `find_average(data_list)` takes a list of data (of some organization of your choosing) and returns the average salary. ***Hints***:
      i. This is NOT (!) the average of the last column of data. It is not mathematically valid to find an average by finding the average of averages—for example, in this case there are many more in the lowest category than in the highest category.
      ii. How many wage earners are considered in finding the average (denominator)? There are a couple of ways to determine this. I think the easiest uses the "cumulative number" column (Column 4), but using Column 3 is not hard and may make more sense to some students.
      iii. How does one find the total dollar value of income (numerator)? Notice that "Column 6 is the combined income of all the individuals in this range of income."
      iv. For testing your function notice that for the 2014 data the average should be $44,569.20. That value is listed on the web page referenced above.
   f) `find_median(data_list)` takes a list of data (of some organization of your choosing) and returns the median income. Unfortunately, this file of data is not sufficient to find the true median so we need to approximate it.
      i. Here is the rule we will use: *find the data line whose cumulative percentage (Column 5) is closest to 50% and return its average income (Column 7).* If both data lines are equally close, return either one.
      ii. **Hint**: Python's `abs()` function (absolute value) is useful (necessary!) here.

   iii. **Hint:** your `get_range()` function should be useful here.
   iv. For testing your function, using our rule the median income for the 2014 data is $27,457.00

g) `do_plot(x_vals,y_vals,year)` provided by us takes two equal-length lists of numbers and plots them. You need to change the label strings in the function so the plot is properly labeled. Include the `year` in the title (Hint: use string concatenation). Note that if you plot the whole file of data, the income ranges are so skewed that the result is a nearly vertical plot at the leftmost edge so close to the edge that you cannot see it in the plot—it looks like nothing was plotted. Plotting the lowest 40 income ranges results in a more easily readable plot.

2. After opening the data file your program must
   a) Print the year.
   b) Print the average income.
   c) Print the median income.
   d) Plot the data: cumulative percentage (Column 5) vs. income (Column 0).
   e) Loop, prompting for either “r” for range , “p” for percent, or nothing
      i. r: prompt for a percent and output the income that is below that percent
      ii. p: prompt for an income and output the percent that earn more
      iii. if only a carriage-return is entered, halt the program

Assignment Notes

1. Items 1-9 of the Coding Standard will be enforced for this project.

2. Files for `year2000.txt` and `year2014.txt` are provided so that you can test your program.
3. Note that most data has commas. I wrote functions that converted a string with commas into a number without commas. I wrote separate functions for int and float, but you may find that one combined function suits your needs. I used a `try-except` statement in case the string wasn’t really a number.
4. For output you need to insert commas. There is a format specification, e.g. if you might have formatted a floating-point value without commas as `{:<12.2f}` you can simply insert a comma before the dot as in `{:<12,.2f}`.
5. There are multiple ways to handle the “`and over`” wording in the last line of the input files. One way you might not have thought of uses the special value `float(“inf”)` which represents infinity in the sense of a value bigger than all others.

Suggested Procedure

- *Solve the problem using pencil and paper first.* You cannot write a program until you have figured out how to solve the problem. This first step may be done collaboratively with another student. However, once the discussion turns to Python specifics and the subsequent writing of Python statements, you must work on your own.

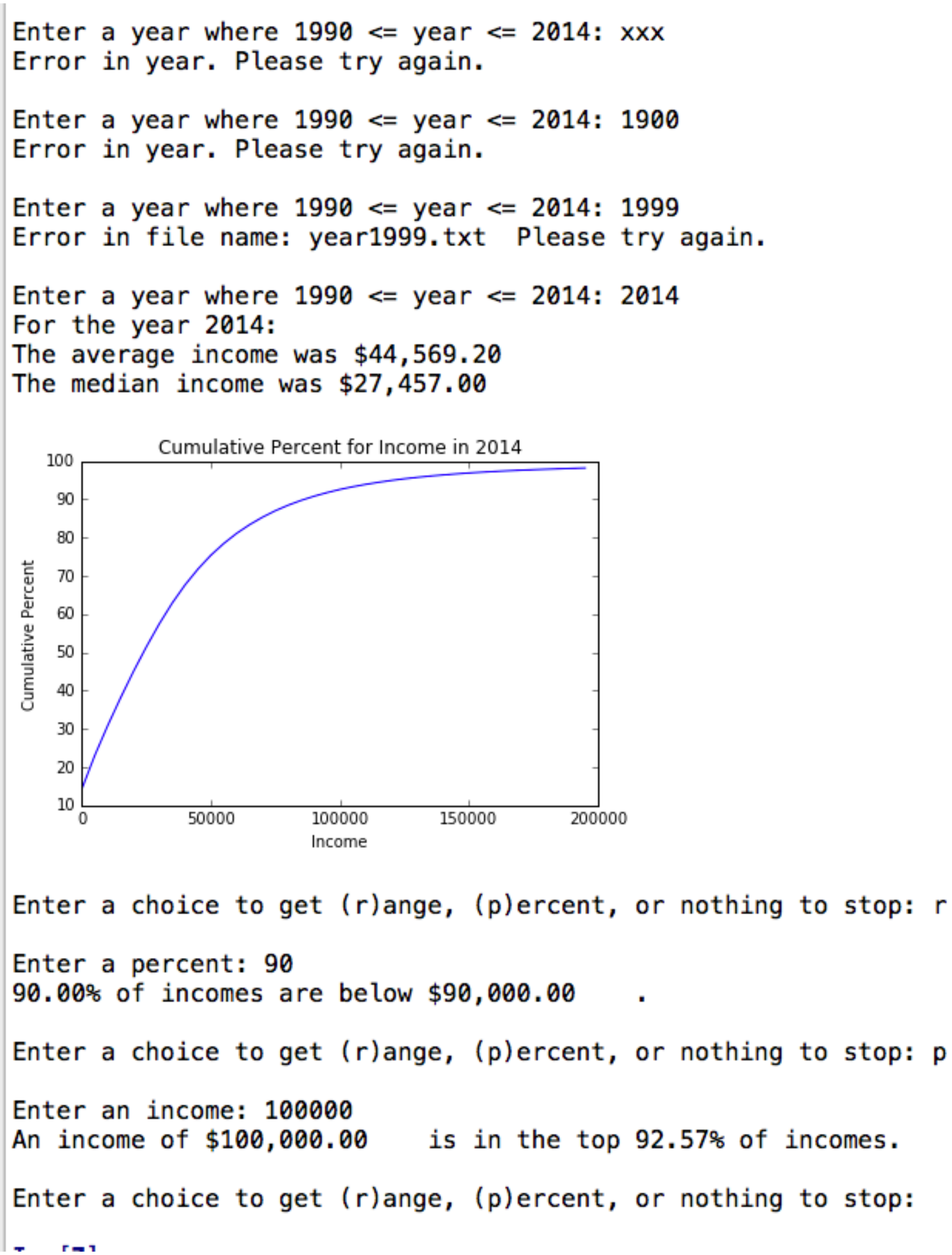
- Construct the program one function at a time—testing before moving on.

- Use the **handin system** to turn in the first version of your solution.
  Cycle through the steps to incrementally develop your program:

- Edit your program to add new capabilities.
- Run the program and fix any errors.
- Use the **handin system** to submit the current version of your solution.

- Be sure to log out when you leave the room, if you’re working in a public lab.

Sample Output



```
Enter a year where 1990 <= year <= 2014: xxx
Error in year. Please try again.

Enter a year where 1990 <= year <= 2014: 1900
Error in year. Please try again.

Enter a year where 1990 <= year <= 2014: 1999
Error in file name: year1999.txt  Please try again.

Enter a year where 1990 <= year <= 2014: 2014
For the year 2014:
The average income was $44,569.20
The median income was $27,457.00
```

```
Enter a choice to get (r)ange, (p)ercent, or nothing to stop: r

Enter a percent: 90
90.00% of incomes are below $90,000.00    .

Enter a choice to get (r)ange, (p)ercent, or nothing to stop: p

Enter an income: 100000
An income of $100,000.00    is in the top 92.57% of incomes.

Enter a choice to get (r)ange, (p)ercent, or nothing to stop:
```