CIS 41B - Lab assignment 3: web scraping, data storage, review of regex, iterables, GUI

Write an application that extracts data of high tech jobs from the US Bureau of Labor Statistics, then displays the data for the user.

The application has 2 parts: lab3back.py (the backend to get data) and lab3front.py (the frontend with the GUI)

The 2 parts do *not* work with each other (no importing of data or methods from one file to the other).

Instead: - the lab3back.py will produce a JSON file and an SQL database file

- the lab3front.py will read from the database to display data to the user

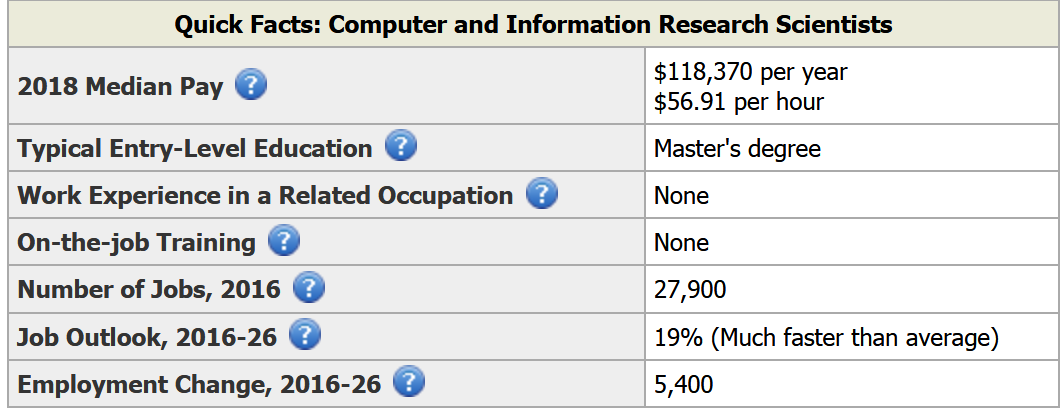
lab3back.py description

Part A

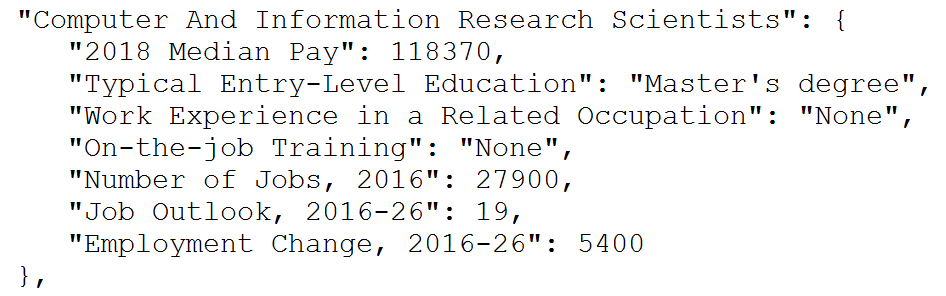
The URL for the US Bureau of Labor Statistics for high tech jobs is: <https://www.bls.gov/ooh/computer-and-information-technology/home.htm>

The lab3back.py code should do the following:

1. From the URL above, extract the links of each of the 10 Occupation names on the page.
2. From the links that you've extracted (do not hard code the links), go to each of the links and extract the occupation name and the 7 fields and data of the Quick Facts table.



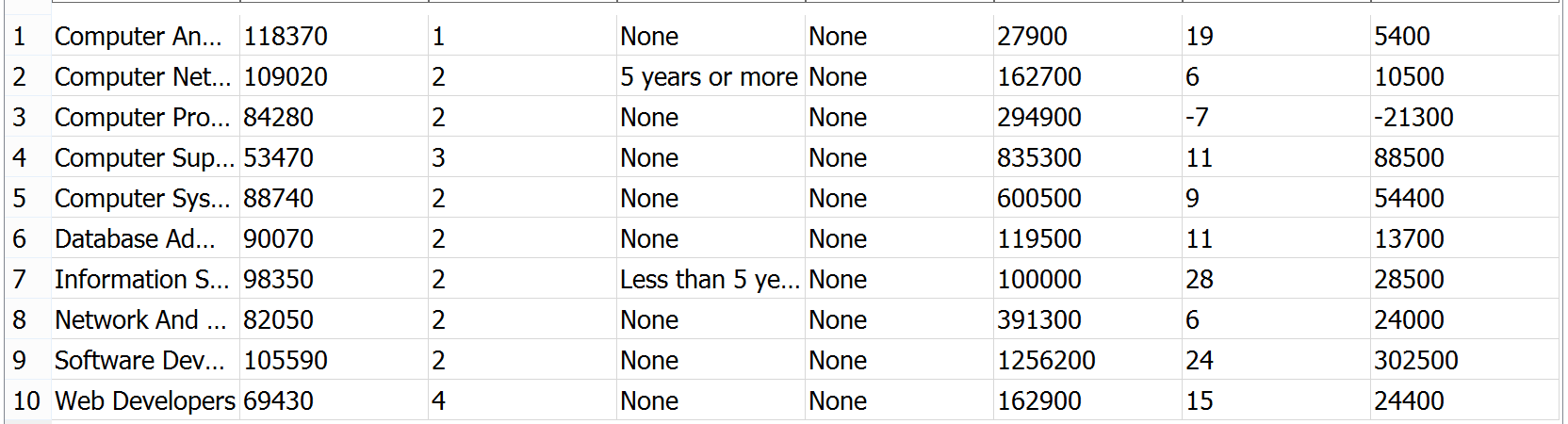
1. The 7 field names should be kept as shown in the table, for example, "2018 Median Pay" or "Job Outlook, 2016-2026".
2. For the Median Pay data, only keep the yearly salary as an integer.  
   For the Number of Jobs data, Job Outlook data, Employment Change data: change the value to an integer.  
   For the Entry-Level Education, if it's "See How To Become One", then change the wording to "Certification".
3. Write all extracted data to a JSON file.  
   Sample first record of JSON file:



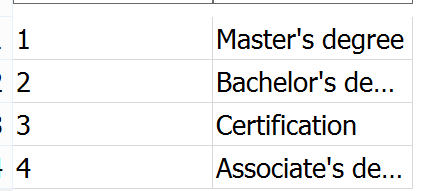
Part B

After you've created the JSON file, comment out the first part of lab3back.py. You don't need to fetch data from the website any more.

1. Read in from the JSON file to create an SQL database. The database has 3 tables, for the field names, job data, and education levels.
2. The field names table has 7 columns and one row. The row of data has actual names of the 7 data fields ("2018 Median Pay", "Typical Entry-Level Education", "Work Experience in a Related Occupation", etc.)
3. The job data table has 8 columns (not counting the id column or first column), and each row is one data record.  
   Sample view of the job data table from the DB Browser tool:



Note that the education column contains the foreign key into the education level table.

1. The education level has 4 rows for the types of degrees/certificates.  
   Sample education level table:  
    

lab3front.py description

1. Write a GUI with 4 window classes: main window, plot window, dialog window, display window.
2. The main window has 3 buttons for the user to view data by salary, by growth rate, or by degree.  
    

1. When the user clicks on the button to view jobs by salary, the plot window appears with a plot of all occupation names, sorted by salary.



Note that:

* The title of the plot comes from the field names table of the database. Do not hard code it.
* The occupation names come from the job data table of the database. Do not hard code them.

1. When the user clicks on the button to view by growth rate, the plot window appears with a plot of all occupation names and the percent growth rate.

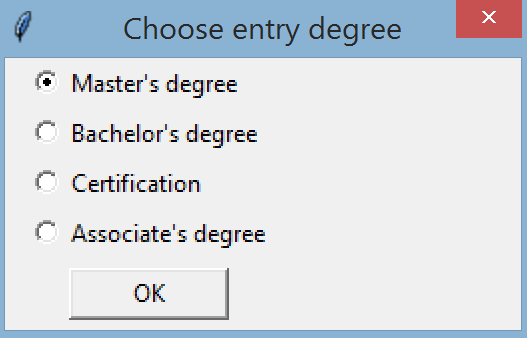
* The title of the plot and the occupation names must come from the database. Don't hard code them.
* Since the plot by salary and plot by growth rate are very similar, use only one callback function and one plot window class. The same code should be used for both plots, and the only if else statement is for the x-axis label (dollars vs. percentage)

1. When the user clicks on the button to view by degree:

* A dialog window with radio buttons lets the user choose an entry level education (degree)
* Then a display window with a listbox displays all the occupation names that require the chosen degree

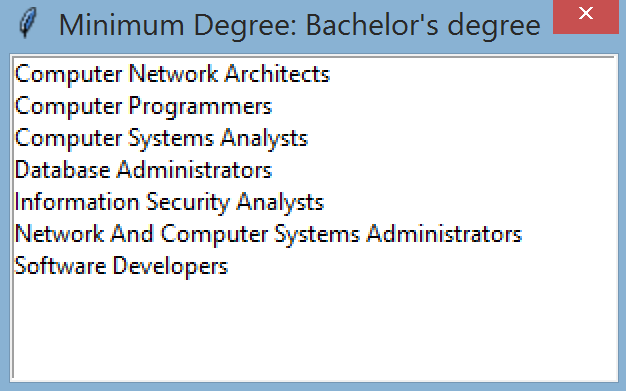
1. The dialog window has 4 radio buttons for the 4 education levels, and an OK button.

* The text for the degree / certification must come from the database. Do not hard code them.
* The 4 radio buttons should be created in a loop. Don't copy and paste 4 lines of code.
* While the dialog window is shown, all other windows are disabled.
* When the user clicks OK, the window closes and the user's choice is available for the main window.



1. The display window has:

* A listbox which can display 10 lines of text.
* The window should get the focus in the GUI, but the user can still go to the main window and select another choice.
* The occupation names are displayed in alphabetical order in the listbox.
* The title of the window should show the degree that the user chose, but don't hard code it.



Some tips on handling data efficiently:

* lab3front.py should get data from the database file. Do not go to the website or import anything from lab3back.py or use the JSON file.
* Don't extract all data from the database to store in memory. Only fetch the data that the user requests.
* lab3front.py should connect to the database and keep the connection open for all transactions. Close the database when the GUI is closed.

When done, turn in 4 files: lab3front.py, lab3back.py, the db file, and the json file.