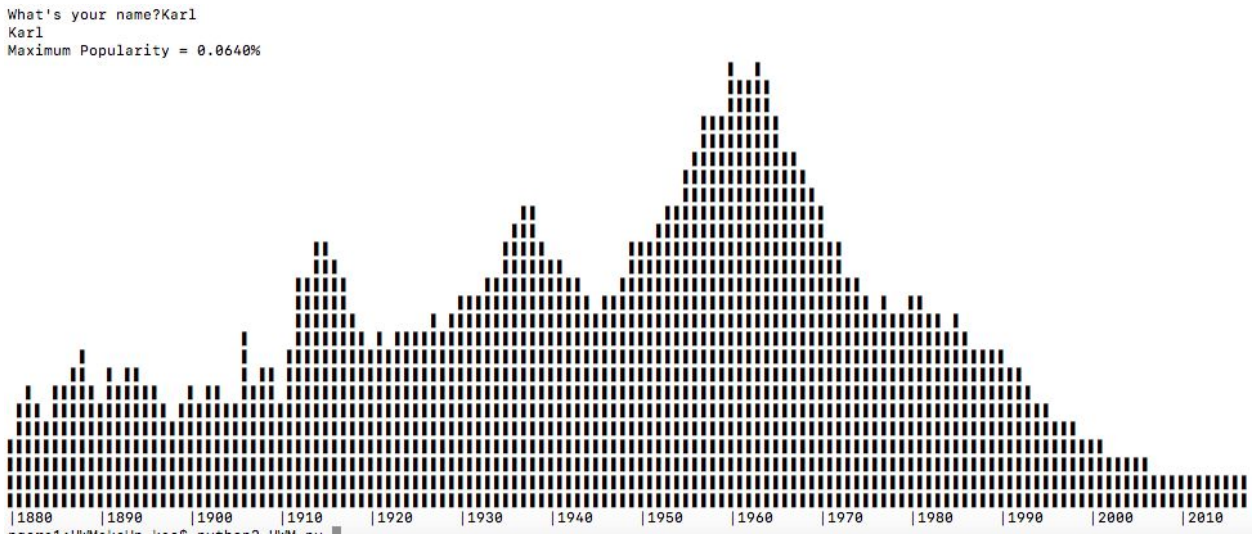
ECS32A Makeup Assignment (HW7 in Piazza)

This assignment is optional. It can improve your grade if you received less than full credit on any of your programming assignments. If completed, it will replace your lowest assignment grade. As an incentive, there is no penalty if this is your assignment grade.

The files for this assignment are under Files/Code/MakeupHomework on Canvas.

The Popularity of Baby Names

In this project we will analyze American baby name data taken from the Social Security Administration Website. We will determine the most popular baby name for a given year. We will also answer repeated queries about the popularity of baby names over time using a dictionary of lists datastructure. The graph below plots the frequency of babies named Karl by year.



As an interesting side note, this plot is very similar to the plots that can be obtained from Googles N-gram viewer, with the exception that this is a much more manageable set of data.

Primary Data File

The primary data file countsByName.csv contains 97,310 names compiled from Social Security card applications after 1879. We have also provided a smaller file `countsByName.test.csv` that can be used for all parts instead of the large one. This is the file gradescope will be using. The first line describes the columns. The leftmost column is the name. This is followed by the number of applications with that name from 1880 to 2017.

Here are the first lines from the file countsByName.csv:

```
Name,1880,1881,1882,1883,1884,1885,1886,1887,1888,1889,1890,1891,1892,1893
Mary,7092,6948,8178,8044,9253,9166,9921,9935,11804,11689,12113,11742,13222,12839
Anna,2616,2714,3143,3322,3880,4014,4298,4240,5008,5085,5253,5114,5562,5712
Emma,2013,2043,2310,2374,2596,2742,2775,2661,3104,2894,2996,2897,3140,2982
Elizabeth,1948,1852,2193,2268,2565,2591,2691,2695,3236,3074,3124,3065,3469,3372
Minnie,1755,1661,2014,2049,2243,2184,2380,2226,2668,2637,2666,2440,2617,2528
Margaret,1578,1667,1828,1894,2152,2215,2283,2432,2914,2930,3115,3077,3447,3579
Ida,1480,1444,1678,1639,1890,1860,2057,1938,2242,2130,2188,2002,2269,2256
Alice,1414,1315,1542,1494,1732,1690,1821,1829,2207,2154,2281,2024,2381,2445
```

Graphing module

This assignment includes an introduction to using a graphing module that outputs results in text to the screen. We will be using the plainchart.py module which is not covered in your primary textbook. Download the module here from Canvas and put it in the same directory as your Python code. You will need to know three lines of code to use the plainchart.py module:

```
# import the plainchart module at the top of your program
import plainchart

# create and print a text chart that is 25 lines high
# values is a list of integer counts by year or
# a list of floating point percentages by year
chart = plainchart.PlainChart(values, height=25)
print(chart.render())
```

You will need to create a list of numbers to give to the charting module. The module works with both integers and floating point values. Regardless of the range of values, the module will automatically adjust the height of the chart to 25 lines.

Part 1: (30 Points) **Names of the Year**

The first step of this assignment will be to determine the top names for a given query year.

In the program `names_of_the_year.py` complete the function stub `print_top_names_for_year` so that the program reads through the input file and returns the 10 names with the highest counts for the given year. You can locate the column in the file to analyze by determining an offset from the year 1880. To get the top 10 you should create a list of (count,name) tuples then sort this list from large to small just as we did for getting the top 10 words for the word counting program. The program only needs to answer this query once.

Below is the output if the user enters the year 2012.

Enter filename:countsByName.csv
Enter year:2012
Sophia 22322
Emma 20954
Isabella 19111
Jacob 19095
Mason 19047
Ethan 17690
Noah 17424
Olivia 17328
Jayden 16944
William 16899

Part 2 (30 Points) **Frequency over Time Plots**

Open and save the file `baby_names_stubs.py` as `baby_names1.py`

Now we will read in the data, store it in a dictionary of lists data structure, and then let the user type in names. If a name appears in our data, we will draw a graph, like the one below, showing how the popularity of the name evolved since 1880.

You will begin by completing the function `make_name_dict` that reads in the counts file and populates a dictionary that can be used to look up a list of counts using the person's name. The counts should be stored as a list of integers, not strings. The function returns the dictionary.

Next complete the function `get_counts_for_name`. This function contains a loop that keeps asking the user to enter a name until it finds it in the dictionary. When "Name" is found it simply prints out “Found Name” on a single line followed by the Python list of counts. The exact format for the prompt and output is shown here:

What's your name?spam
What's your name?porsche
What's your name?Porsche
Found Porsche
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 10, 14, 9, 7, 19, 48, 64, 51, 45, 46, 57, 66, 82, 127, 182, 172, 159, 179, 123, 123, 70, 68, 45, 30, 35, 27, 12, 25, 32, 15, 20, 9, 15, 18, 17, 16, 11, 9, 14, 9, 5, 0, 13, 0, 5]

Strategy hints:

- The split method will give you a list of counts as strings, but you will need to convert them to integers before storing them in the dictionary.
- The format for the printed list is exactly what you get if your give a list to the print command. (e.g. print(counts)) We will improve on this output in the next part.

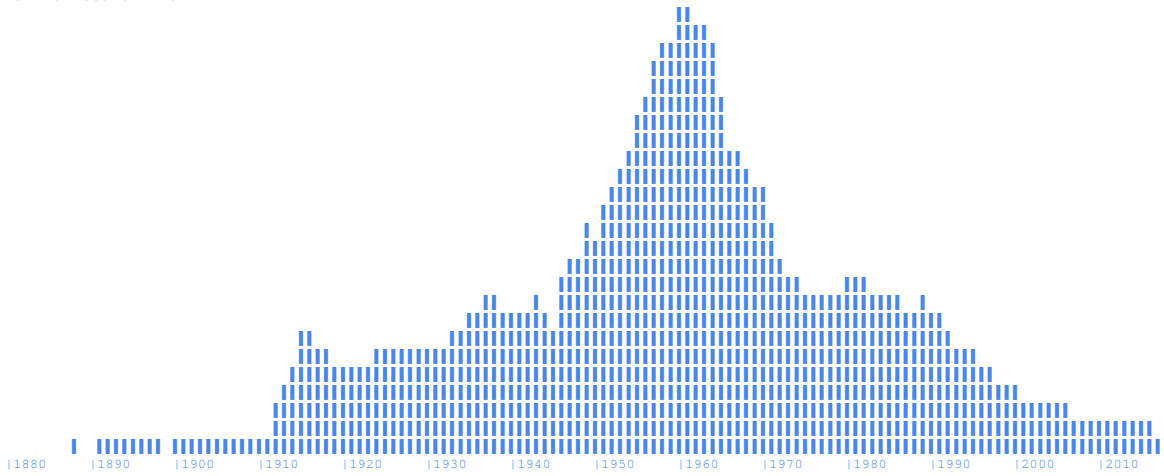
Submit this to gradescope as `baby_names1.py`

Part 3 (20 Points) **Plotting the output**

Save and submit this part as `baby_names2.py` and make sure the `plainchart.py` module is in the same directory.

Uncomment the `make_graph` function and complete it so that it plots the graph exactly as shown below. The function will first print the maximum count, the chart, and then call the function we provided for printing the x-axis. The output for the name Karl should look like this:

```
What's your name?Karl
Found Karl
Maximum Count = 2612
```



Part 4 (20 Points) **Plotting percentages**

Save and submit this part as `baby_names3.py`.

To account for the fact that the population of the United States has been increasing it makes sense to plot the percentage of applications instead of the total number of applications for a specific name. To get the percentage of applications for a specific name and year, you need to divide the count for that name by the total number of applications that year. You have the counts by year from the file above. We are also providing the totals by year in a file named totalsByYear.csv:

```
Year,Total
1880,201484
1881,192696
1882,221533
…
2014,3696311
2015,3688687
```

```
2016,3652968
2017,3546301
```

Complete the function `get_totals_by_year` to return the total number of people in the database for each year as integers. This number increases over time, and our plots will be more accurate representation of the name's popularity if we plot the percentage over time.

Next complete the function `percentages` that takes the lists of counts and totals and returns a list of percentages, (count/total)*100 for each year. Pass the percentages to the `make_graph` function instead of the counts.

Finally, modify the `make_graph` function so that it prints the maximum popularity as a percentage formatted to four decimal places using the format method. The output for the name Karl should look like this:

```
What's your name?Karl
Found Karl
Maximum Popularity = 0.0640%
```