

CSc 110, Spring 2017

Programming Assignment #6: Baby Names (20 points)

Due: Tuesday, February 28, 2017, 7:00 PM

Thanks to Nick Parlante from Stanford for the assignment concept and to Marty Stepp and Stuart Reges for parts of this document.

This assignment focuses on reading input files. Turn in a file named `baby_names.py`. You will need `drawingpanel.py`, which you used on previous assignments, to write this program.

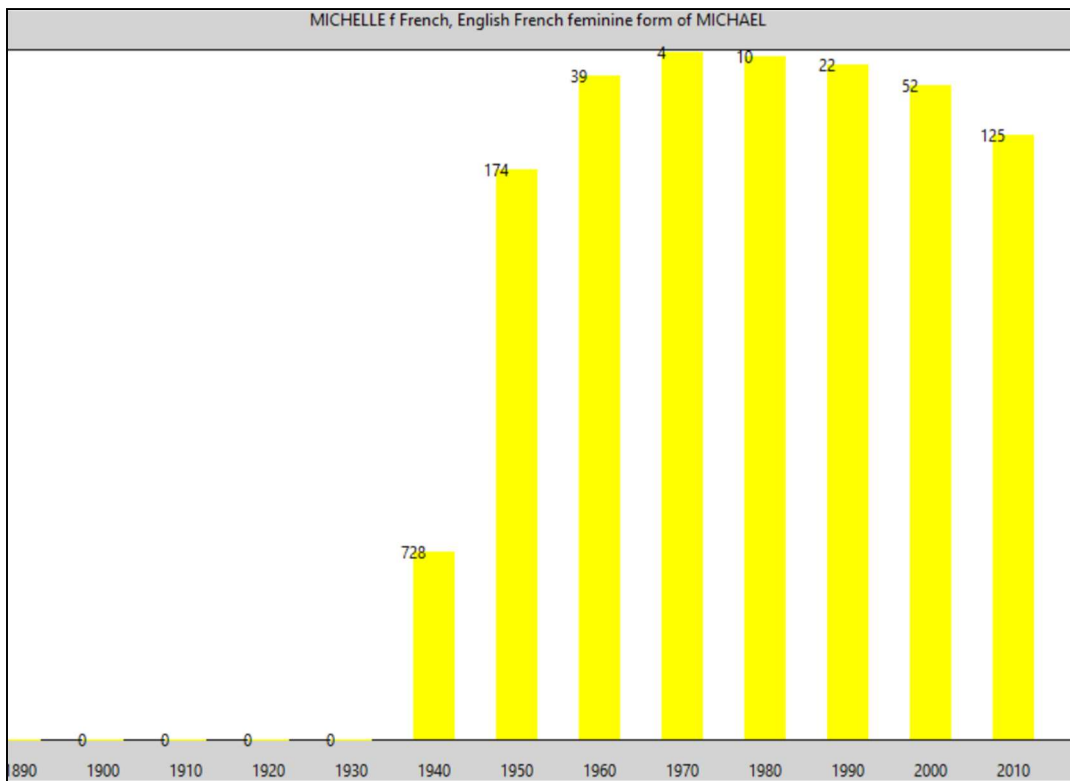
Program Description:

The Social Security Administration has published the 1000 most popular boy and girl names for children born in the US for all years after 1879 (see <http://www.ssa.gov/OACT/babynames/>). For this project, you will prompt the user for a name and gender, and then display the name's meaning and popularity as console text and as a graphical bar chart on a `DrawingPanel`. The input data about names' popularity rankings and meanings comes from two input files provided on the course web site.

Your program should give an introduction and then prompt the user for a first name and gender. It should then read the name rank data file searching for that name and first letter of the gender, case-insensitively (that is, you should find the name and gender regardless of the capitalization the user uses when typing them in). If the name and gender combination is found in the file, your program should print a line of statistics about that name's popularity in each decade, the name's meaning and display information about the name graphically.


This program allows you to search through the data from the Social Security Administration to see how popular a particular name has been since 1890.

```
Name: michelle
Gender: female
Michelle f 0 0 0 0 0 728 174 39 4 10 22 52 125
MICHELLE f French, English French feminine form of MICHAEL
```



Input Data and Files:

Your program reads data from two files. Download them from our web site to the same folder as your program.

1.  `names.txt`: *popularity rankings for each name 1890-2010*

Each line of `names.txt` contains a name followed by that name's rank in 1890, 1900, 1910, etc. The default file has 13 numbers/line, so the last represents the ranking in 2010. Rank #1 was the most popular that year, while rank #999 was not popular. Rank 0 means the name did not appear in the top 1000. For example:

```
Michelle f 0 0 0 0 0 728 174 39 4 10 22 52 125
Michelle m 0 0 0 0 0 0 0 0 736 897 0 0 0
Michial m 0 0 0 0 0 0 987 0 0 0 0 0 0
```

"Michelle" as a female name first made the list in 1940 and peaked in 1970 at #4. It has been on a steady decline in popularity since. "Michial" only made the top-1000 in 1950.


Once the user types a name and gender, search each line of `names.txt` to see if it contains data for that name and gender. If the name and gender combination is found, output its data line to the console, then construct a `DrawingPanel` to graph the data (see next page). **Your code should not assume that the file is sorted alphabetically.**

If the name and gender combination is not found, output a "not found" message and don't show any data. No `DrawingPanel` should appear.

```
This program allows you to search through the
data from the Social Security Administration
to see how popular a particular name has been
since 1890.
```

```
Name: haYdEn
Gender: uncertain
"haYdEn" not found.
```

Though the data shown above has 13 decades' worth of rankings, your program should work properly with any number of decades of data (at least 1). Since there is a limit to the size of the `DrawingPanel`, you'd only be able to see data from 13 decades, but your code should process as many decades of data as it finds in the line. **Do not assume that there will be exactly 13 decades when writing this program.** On the course website is a file named `names2.txt` with 8 decades of data to help you test this behavior.

2.  `meanings.txt`: *descriptions of the meanings of each name*

If the name and gender combination is found in `names.txt`, you should also read `meanings.txt` to find its meaning. The line containing the name's meaning should be printed to the console and also drawn on the `DrawingPanel`. Every name in `names.txt` is also in `meanings.txt`, so you do not need to worry about a name having rankings but no meaning data. Some names have long meanings that may stretch past the edges of the `DrawingPanel`. Don't be concerned by that.

Each line of `meanings.txt` contains a name in upper case, followed by the name's gender and meaning. For example:

```
MICHELLE f French, English French feminine form of MICHAEL
MICHELYNE f English (Modern) Pet form of MICHELLE
MACHI f Japanese Means "pathway" in Japanese.
MICHIAL m (no meaning found)
```

Though the two input files contain different data, the task of searching for a name and gender in `names.txt` is very similar to the task of searching for a name and gender in `meanings.txt`. Your program should take advantage of this fact and follow the Implementation Guidelines.

Graphical Output:

The panel's overall size is 780x560 pixels. Its background is white. It has light gray ("light gray") filled rectangles along its top and bottom with a black line at their bottom and top, respectively. The two rectangles are each 30 pixels tall and span across the entire panel, leaving an open area of 780x500 pixels in the middle. The line of data about the name's meaning appears in the top gray rectangle at (390, 16).

Each decade is represented by a width of 60 pixels. The bottom light gray rectangle contains black labels for each decade, 8px from the bottom of the `DrawingPanel`. For example, with default constant values (see style guidelines), the text "1890" is at (15, 552) and "1910" is at (75, 552).

Starting at the same x-coordinate, a bar shows the name ranking data for each year. The bars are light green for male names and yellow for female names. Bars are half as wide as each decade (30px). The table at right shows the mapping between rankings and y-values of the tops of bars. Y-values start at 30 (below the top gray rectangle), and there is a vertical scaling factor of 2 between pixels and rankings. With a legend height of 30, the y-coordinate is 30 plus the rank divided by 2.

Rank	Top y
1	30
2, 3	31
4, 5	32
6, 7	33
...	...
996, 997	528
998, 999	529
0	530

At the same coordinate as the top-left of each bar, black text shows the name's rank for that decade. For example, Michelle was #4 in 1970, so "4" appears at (480, 32). A 0 rank means the name was not in the top 1000. No bar should appear in such a case, and "0" should be drawn right above the bottom gray bar. For example, in the screenshot above, Michelle's 0 in 1920 is drawn at (180, 530).

Implementation Guidelines:

You are required to implement the three functions below, using the names specified for the functions and their corresponding parameters.

1. `find_name(fname, name, gender)` – where `fname`, `name`, and `gender` are strings.

This function opens the file `fname` and then uses `readlines()` to produce all the lines of the file. It then loops over the lines of the file, looking for the first line that contains the specified name and gender combination. (Hint: think about using `split()` to break up the lines of the file as in Lecture 17. If the function finds the name and gender, it immediately returns that line. (Note that the line is a string.) If the name and gender combination is not found, the function returns an empty string. The function should find the name regardless of the capitalization the user uses to type the name. If the user types "LiSa" or "lisa", you should find it even though the input files have it as "Lisa" and "LISA".

2. `draw_basics(p, meaning_line)` – where `p` is a `DrawingPanel` and `meaning_line` is a string.

This function draws the top and bottom rectangles of the graph and displays the meaning of the name, specified by the parameter `meaning_line`, centered in the top rectangle.

3. `draw_histogram(p, name_line)` – where `p` is a `DrawingPanel` and `name_line` is a string.

This function draws the remaining portions of the graph (the dates, ranks, rectangles of the histogram) as specified above in the Graphical Output section.

Stylistic Guidelines:

You should have at least these **three constants**. If the constant values are changed, your output should adapt.

- The **starting year** of the input data, as an integer (default of 1890)
e.g. If you change the start year to 1825, the program should assume the data comes from 1825, 1835, etc.
- The **width** of each decade on the `DrawingPanel`, as an integer (default of 60)
e.g. If you change the width from 60 to 50, the decade bars would be 50px apart and 25px wide.
- The **height** of the legend rectangles, as an integer (default of 30)
e.g. If you change the legend height from 30 to 20, the gray rectangles at top and bottom would be 20px tall. The panel would be 540px tall, the open area in the middle would be 500px tall, and the decade labels adjust to $y=532$.

For this assignment you are limited to the language features in lectures 1 - 18. Follow past stylistic guidelines about indentation, line lengths, identifier names, and localizing variables, and commenting at the beginning of your program, at the start of each function, and on complex sections of code. You may not have global variables or nest functions in one another.