BabyNames

by Juliette Woodrow and Kara Eng

~Logistics~

- Due Monday June 1st 1:30pm PDT
- 2 parts
  - Dictionaries
  - BabyNames

Assignment Overview

- Part 1: Dictionaries
  - Part A: Reading the file
  - Part B: Calculating the number of infections per day
- Part 2: Baby Names
  - Data Processing
  - Connecting Data to Graphics
  - Data Visualization

Data Analysis

Problem: Data Analysis

- Given a file with data in it. Each line of the file comes in the format:
  location, day1Total, day2Total, ... , day7Total
- Note: each line has a unique location, you don't need to worry about duplicates

Problem: Data Analysis Part 1

- implement **def load_data(filename)**
- Goal: build a dictionary where each key is the location and the value is a list of daily infection totals:

```
{location: [day1Total, day2Total, ... , day7Total]}
```

Problem: Data Analysis Part 2

- implement **def daily_cases(cumulative)**
- Goal: build a dictionary where each key is the location and the value is a list of new infections per day:

```
{location: [day1NewInfections, day2NewInfections, ... ,
                      day7NewInfections]}
```

Hint: you're building the new list based on the values of the old list, one by one. The first value for both lists is the same. Try doing it by hand!

BabyNames

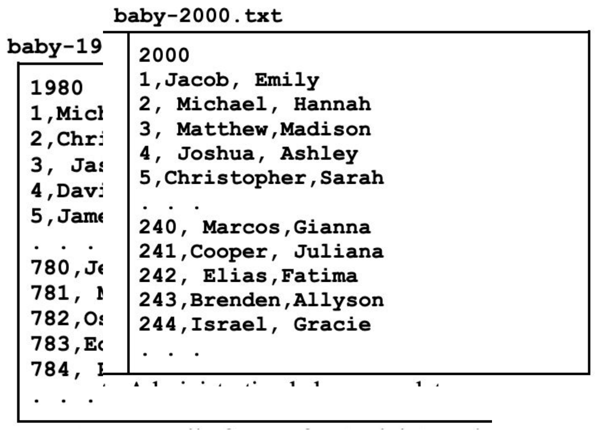
BabyNames - IMPORTANT NOTE

You should not change any of the function names or parameter requirements that we already provide to you in the starter code.

BabyNames Overview

BabyNames Overview

Social Security Administration baby name data in txt files



baby-2000.txt

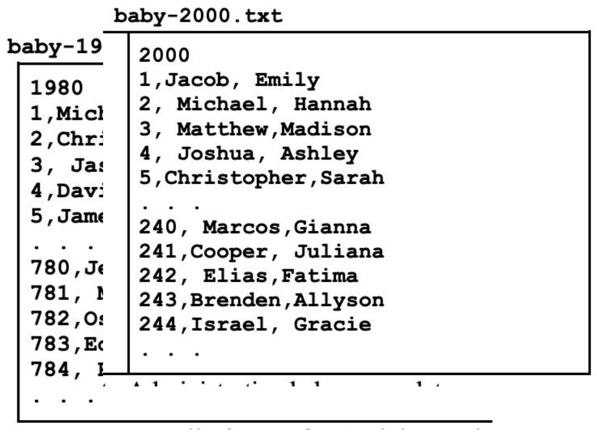
```
2000
1,Jacob, Emily
2, Michael, Hannah
3, Matthew,Madison
4, Joshua, Ashley
5,Christopher,Sarah
. . .
240, Marcos,Gianna
241,Cooper, Juliana
242, Elias,Fatima
243,Brenden,Allyson
244,Israel, Gracie
. . .
```

baby-19

```
1980
1,Mic
2,Chr
3, Ja
4,Dav
5,Jam
. . .
780,J
781,
782,O
783,E
784,
. . .
```

BabyNames Overview

Social Security Administration baby name data in txt files



baby-2000.txt

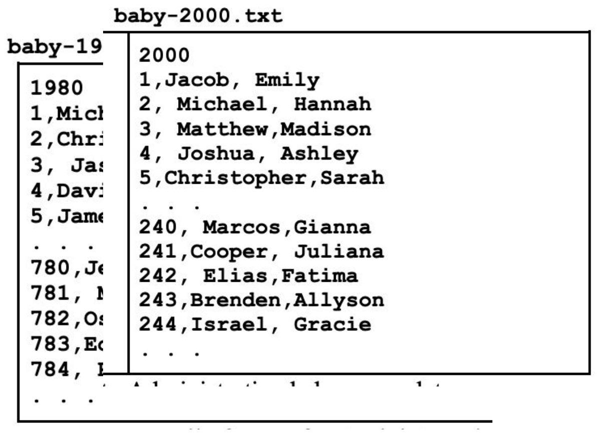
```
2000
1,Jacob, Emily
2, Michael, Hannah
3, Matthew,Madison
4, Joshua, Ashley
5,Christopher,Sarah
. . .
240, Marcos,Gianna
241,Cooper, Juliana
242, Elias,Fatima
243,Brenden,Allyson
244,Israel, Gracie
. . .
```

baby-19

```
1980
1,Mich
2,Chri
3, Jas
4,Davi
5,Jame
. . .
780,Je
781, M
782,Os
783,Ed
784, L
. . .
```

BabyNames Overview

Social Security Administration baby name data in txt files

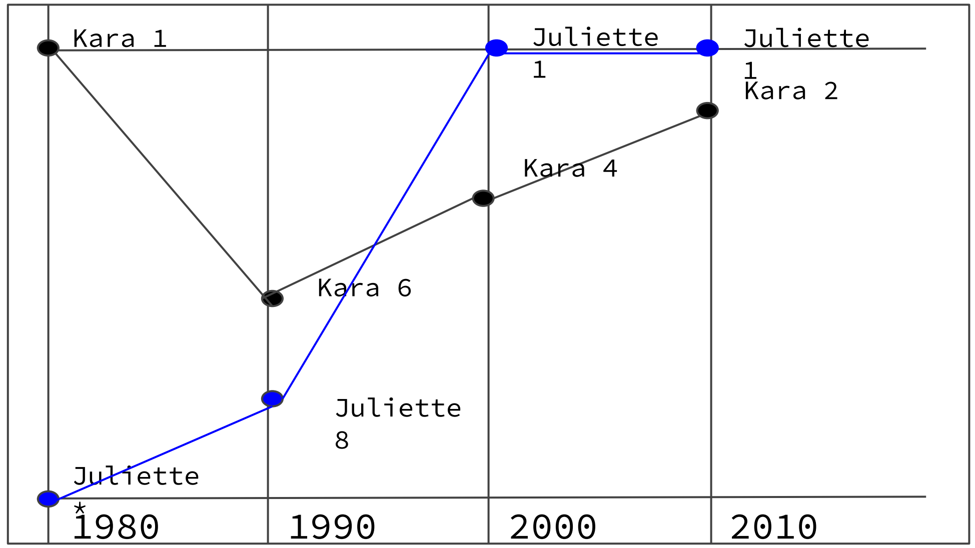


baby-2000.txt

```
2000
1,Jacob, Emily
2, Michael, Hannah
3, Matthew,Madison
4, Joshua, Ashley
5,Christopher,Sarah
. . .
240, Marcos,Gianna
241,Cooper, Juliana
242, Elias,Fatima
243,Brenden,Allyson
244,Israel, Gracie
. . .
```

baby-19

```
1980
1,Mic
2,Chr
3, Ja
4,Dav
5,Jam
. . .
780,J
781, 
782,O
783,E
784, 
. . .
```

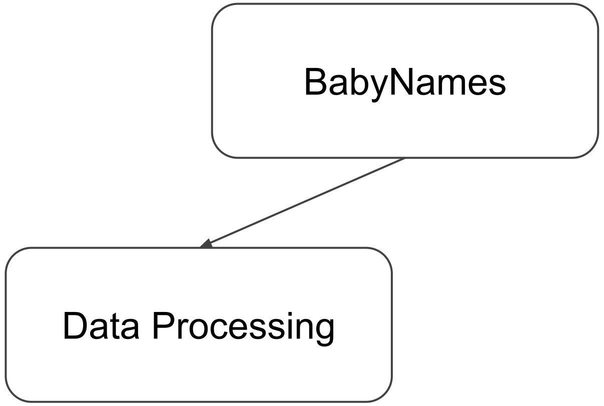


This super cool visualization of the data showing how name popularity varies over time.

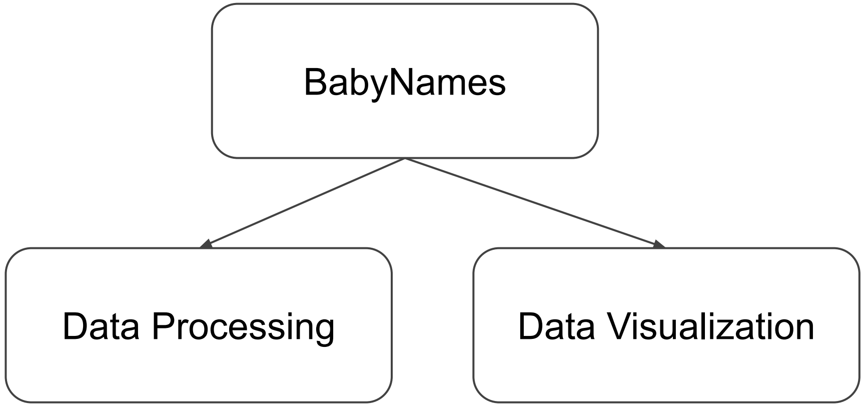
That seems like a lot… Let’s break it down

That seems like a lot... Let’s break it down

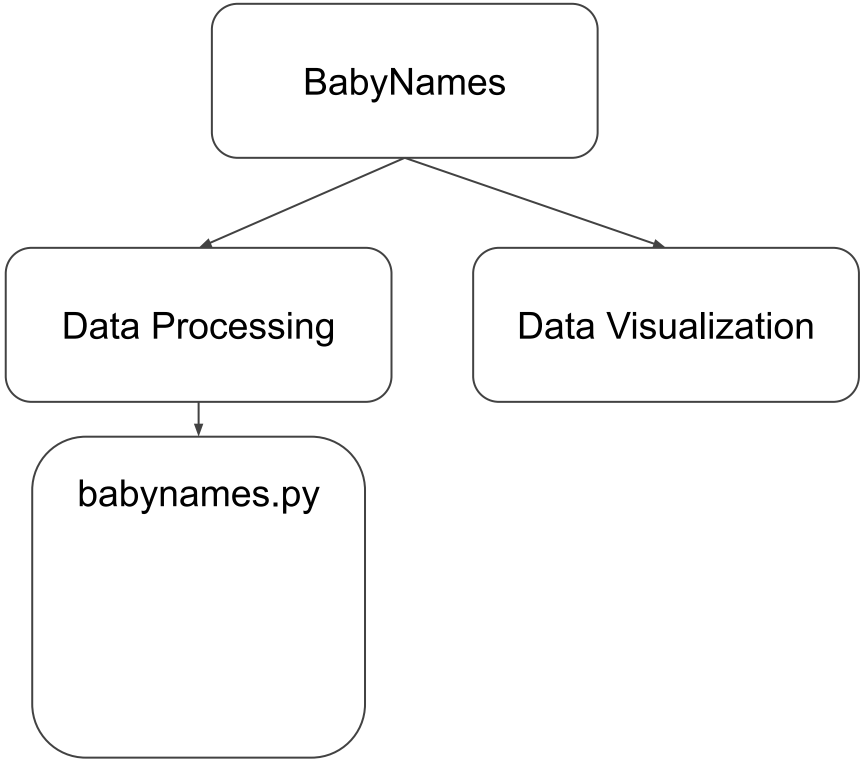
That seems like a lot… Let’s break it down



That seems like a lot... Let’s break it down



That seems like a lot... Let’s break it down



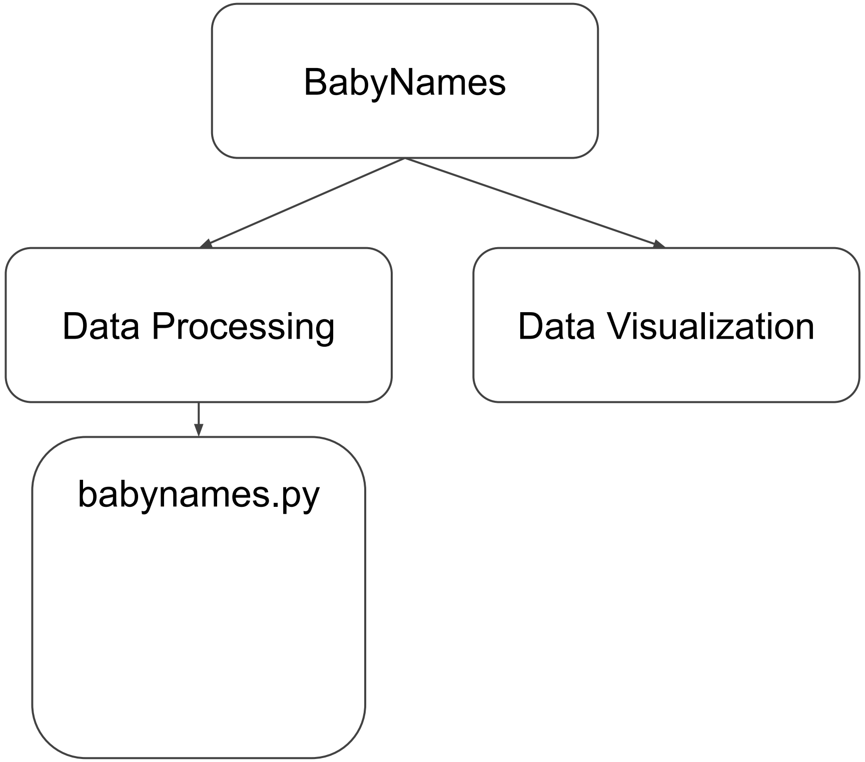
That seems like a lot... Let's break it down

Milestones 1-3:

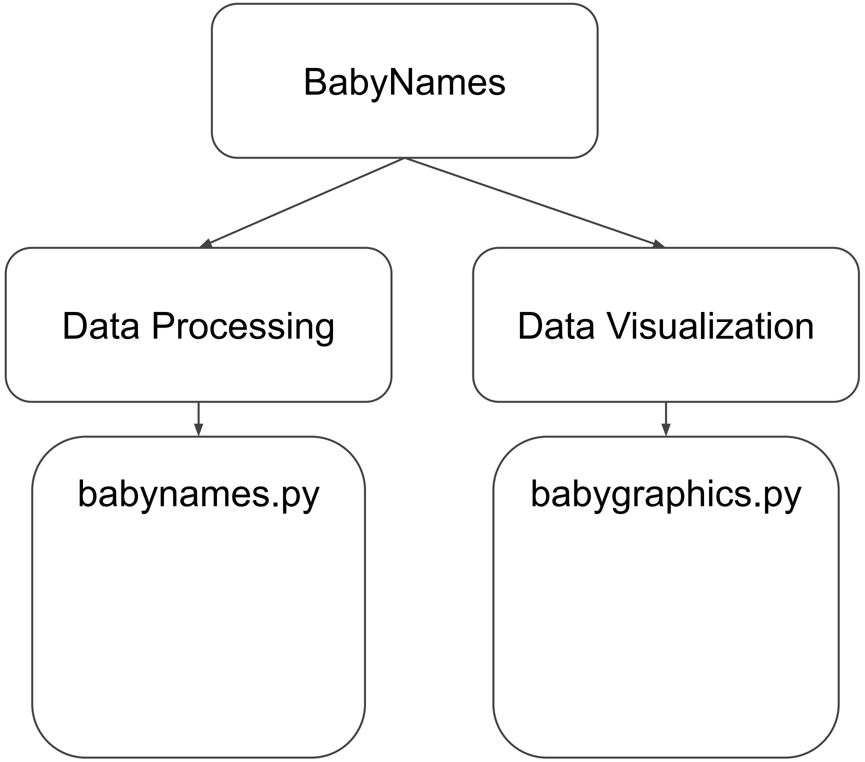
1. Add a single name

2. Processing a whole file

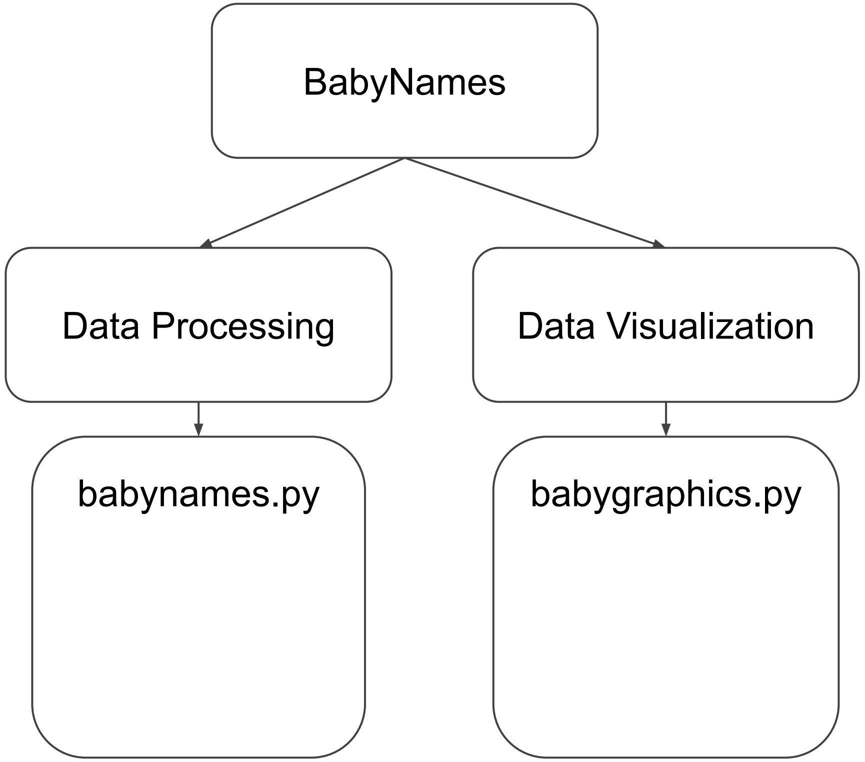
3. Processing many files and enabling search



That seems like a lot… Let’s break it down



That seems like a lot... Let's break it down

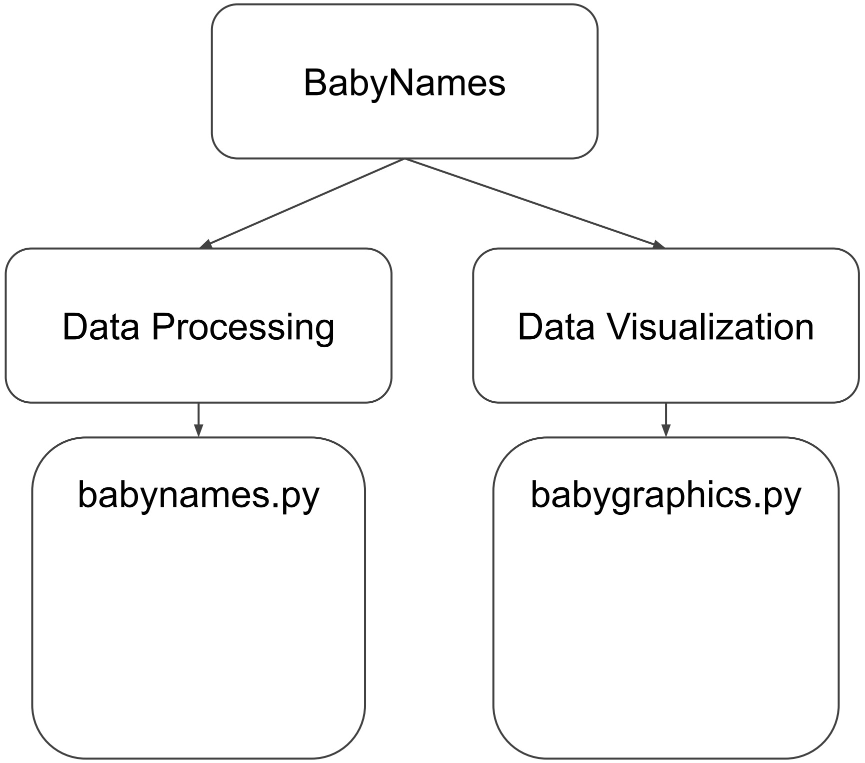


```
Milestones 4-6:

1.  Run provided
    graphics
    code

2.  Draw the
    background
    grid

3.  Plot the
    baby name
    data
```

That seems like a lot... Let’s break it down

Milestones 1-3:

1. Add a single name
2. Processing a whole file
3. Processing many files and enabling search



Milestones 4-6:

1. Run provided graphics code
2. Draw the background grid
3. Plot the baby name data

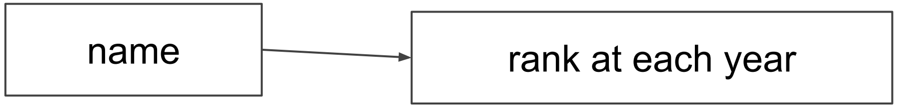
Let’s start with Data Processing

Let’s start with Data Processing

How can we efficiently store the data?

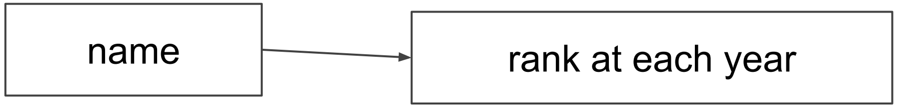
Let’s start with Data Processing

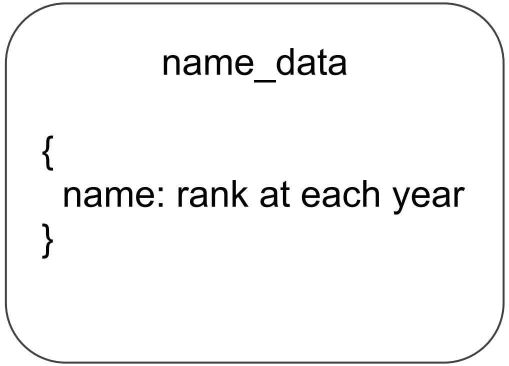
How can we efficiently store the data?



Let’s start with Data Processing

How can we efficiently store the data?

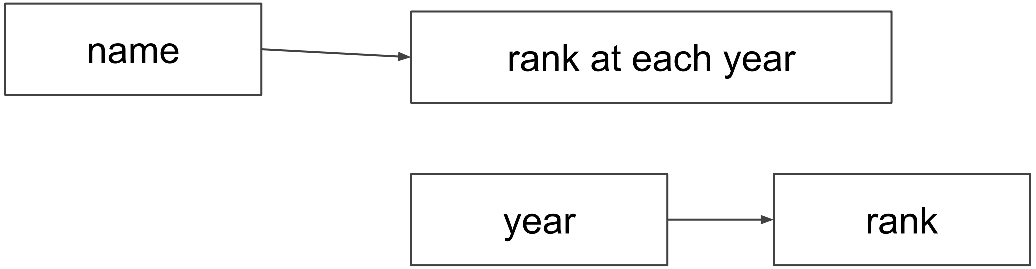




```
{
  name: rank at each year
}
```

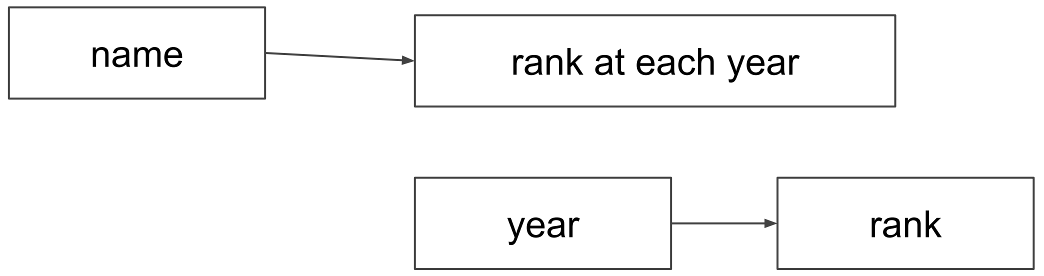
Let’s start with Data Processing

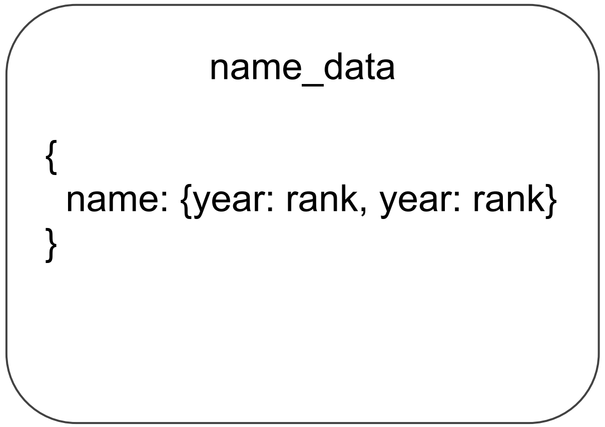
How can we efficiently store the data?



Let’s start with Data Processing

How can we efficiently store the data?

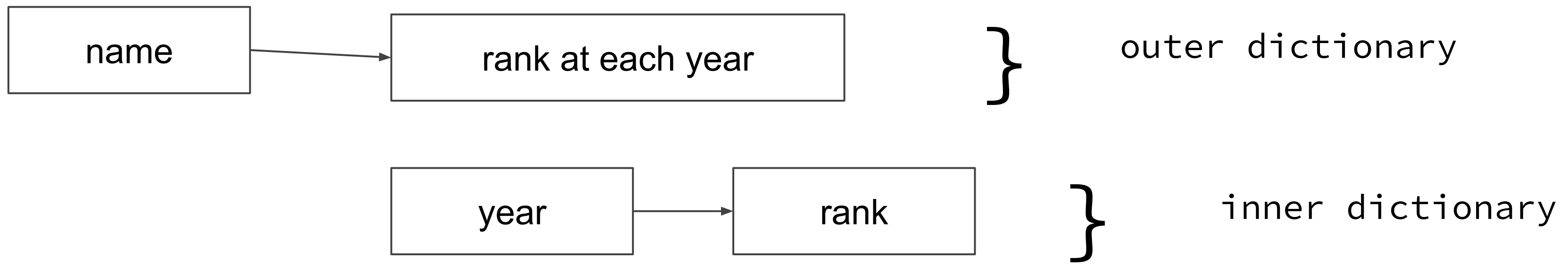




```
{
  name: {year: rank, year: rank}
}
```

Let's start with Data Processing

How can we efficiently store the data?



DataProcessing - Milestone 1

DataProcessing - Milestone 1

1. **Add a single name:** Write a function in babynames.py for adding some partial name/year/count data to a passed in dictionary.

DataProcessing - Milestone 1

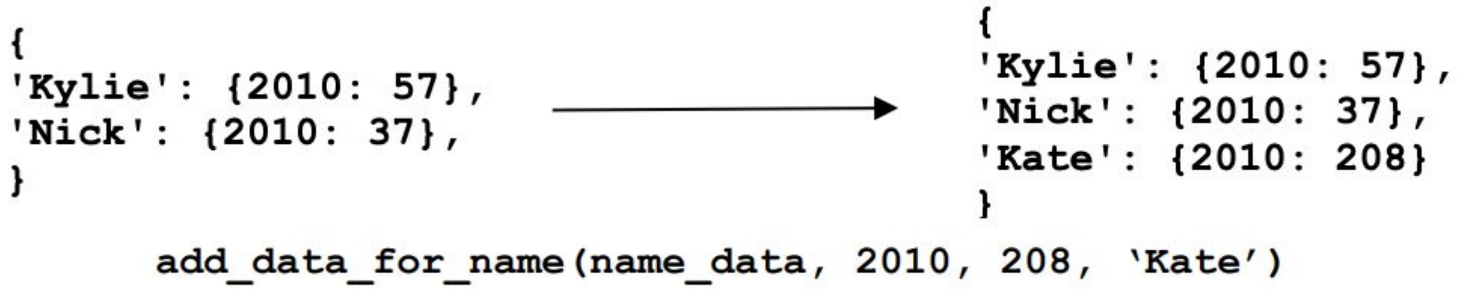
1. **Add a single name:** Write a function in babynames.py for adding some partial name/year/count data to a passed in dictionary.

```
def add_data_for_name(name_data, year, rank, name):
“””
Adds the given year and rank to the associated name in the name_data dictionary.
“””
```

DataProcessing - Milestone 1

1. **Add a single name:** Write a function in babynames.py for adding some partial name/year/count data to a passed in dictionary.

```
def add_data_for_name(name_data, year, rank, name):
“””
Adds the given year and rank to the associated name in the name_data dictionary.
“””
```



```
{
'Kylie': {2010: 57},
'Nick': {2010: 37},
}
```

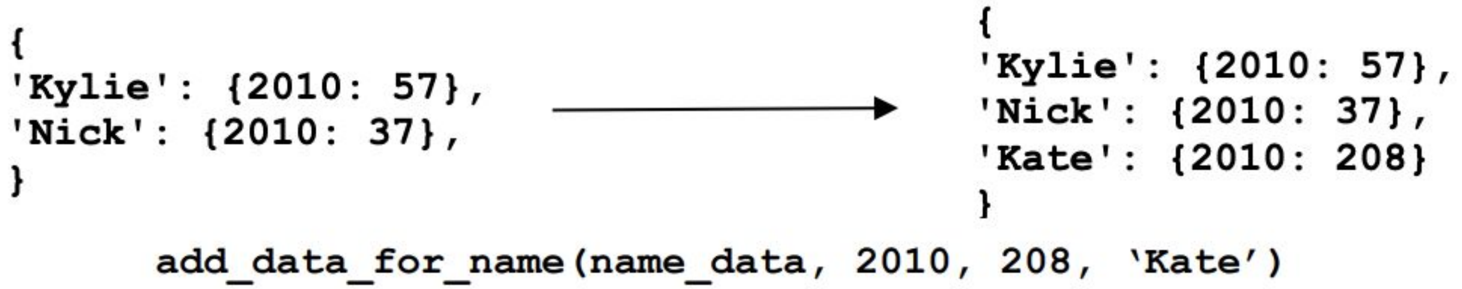
```
{
'Kylie': {2010: 57},
'Nick': {2010: 37},
'Kate': {2010: 208}
}
```

```
add_data_for_name(name_data, 2010, 208, ‘Kate’)
```

DataProcessing - Milestone 1

1. **Add a single name:** Write a function in babynames.py for adding some partial name/year/count data to a passed in dictionary.

```
def add_data_for_name(name_data, year, rank, name):
“””
Adds the given year and rank to the associated name in the name_data dictionary.
“””
```



```
{
'Kylie': {2010: 57},
'Nick': {2010: 37},
}
```

```
{
'Kylie': {2010: 57},
'Nick': {2010: 37},
'Kate': {2010: 208}
}
```

```
add_data_for_name(name_data, 2010, 208, ‘Kate’)
```

DataProcessing - Milestone 1 - The “Sammy issue”

DataProcessing - Milestone 1 - The “Sammy issue”

In some cases, a name shows up twice in one year. Once for a male name and once for a female name.

DataProcessing - Milestone 1 - The “Sammy issue”

In some cases, a name shows up twice in one year. Once for a male name and once for a female name.

To handle this, store whichever rank number **is smaller**

DataProcessing - Milestone 1 - The “Sammy issue”

In some cases, a name shows up twice in one year. Once for a male name and once for a female name.

To handle this, store whichever rank number **is smaller**

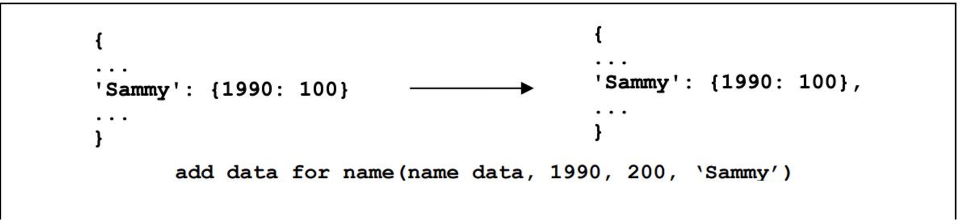
Example: If ‘Sammy’ shows up as both rank 100 (from male data) and 200 (from female data) in 1990, you should only store ‘Sammy’ as having rank 100 for year 1990.

DataProcessing - Milestone 1 - The “Sammy issue”

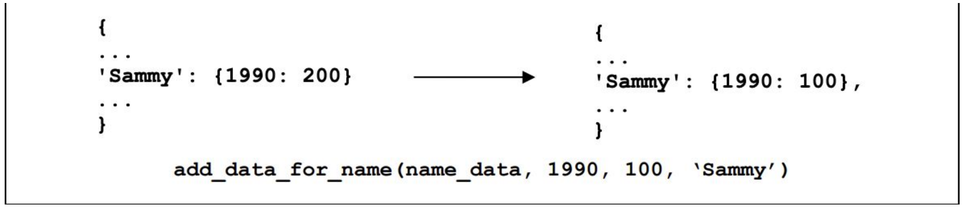
In some cases, a name shows up twice in one year. Once for a male name and once for a female name.

To handle this, store whichever rank number **is smaller**

Example: If ‘Sammy’ shows up as both rank 100 (from male data) and 200 (from female data) in 1990, you should only store ‘Sammy’ as having rank 100 for year 1990.



```
{                                         {
...                                       ...
'Sammy': {1990: 100}   ------>            'Sammy': {1990: 100},
...                                       ...
}                                         }
      add data for name(name data, 1990, 200, 'Sammy')
```



```
{                                         {
...                                       ...
'Sammy': {1990: 200}   ------>            'Sammy': {1990: 100},
...                                       ...
}                                         }
      add_data_for_name(name_data, 1990, 100, 'Sammy')
```

DataProcessing - Milestone 1 - TESTING

DataProcessing - Milestone 1 - TESTING

We provided you with two doctest for this function.

DataProcessing - Milestone 1 - TESTING

We provided you with two doctest for this function.

You should write more doctests to test other cases before moving on to the next milestone

DataProcessing - Milestone 1 - TESTING

We provided you with two doctest for this function.

You should write more doctests to test other cases before moving on to the next milestone

One idea: add a doctest for the “Sammy issue”

DataProcessing - Milestone 2

DataProcessing - Milestone 2

2. **Processing a whole file:** Write a function for processing an entire data file and adding its data to a dictionary.

DataProcessing - Milestone 2

2. **Processing a whole file:** Write a function for processing an entire data file and adding its data to a dictionary.

```
def add_file(name_data, filename):
“””
Reads the information from the specified file and populates the name_data
dictionary with the data found in the file.
“””
```

DataProcessing - Milestone 2

2. **Processing a whole file:** Write a function for processing an entire data file and adding its data to a dictionary.

```
def add_file(name_data, filename):
“””
Reads the information from the specified file and populates the name_data
dictionary with the data found in the file.
“””
```

We can use the helpful add_data_for_name function that we just wrote!

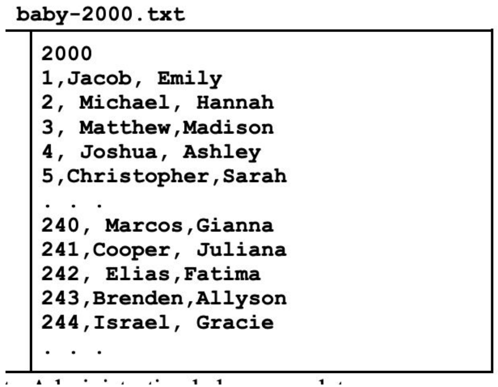
DataProcessing - Milestone 2

2. **Processing a whole file:** Write a function for processing an entire data file and adding its data to a dictionary.

```
def add_file(name_data, filename):
“””
Reads the information from the specified file and populates the name_data
dictionary with the data found in the file.
“””
```

We can use the helpful add_data_for_name function that we just wrote!

DataProcessing - Milestone 2 - File Format

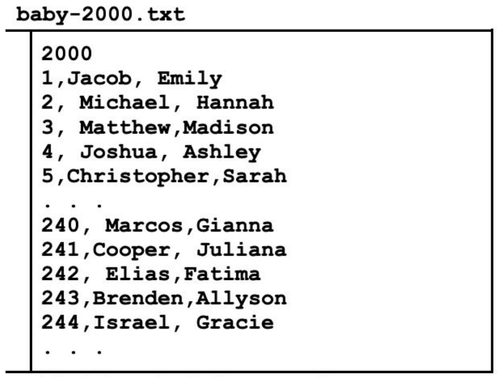


baby-2000.txt

```
2000
1,Jacob, Emily
2, Michael, Hannah
3, Matthew,Madison
4, Joshua, Ashley
5,Christopher,Sarah
. . .
240, Marcos,Gianna
241,Cooper, Juliana
242, Elias,Fatima
243,Brenden,Allyson
244,Israel, Gracie
. . .
```

DataProcessing - Milestone 2 - File Format

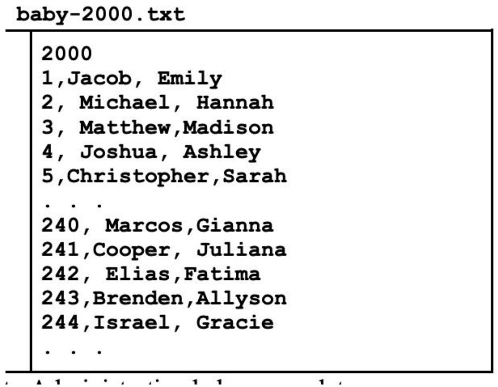
The first line of each file is the year



baby-2000.txt

```
2000
1,Jacob, Emily
2, Michael, Hannah
3, Matthew,Madison
4, Joshua, Ashley
5,Christopher,Sarah
. . .
240, Marcos,Gianna
241,Cooper, Juliana
242, Elias,Fatima
243,Brenden,Allyson
244,Israel, Gracie
. . .
```

DataProcessing - Milestone 2 - File Format



baby-2000.txt

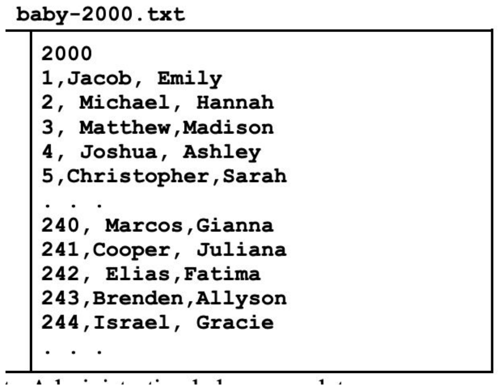
```
2000
1,Jacob, Emily
2, Michael, Hannah
3, Matthew,Madison
4, Joshua, Ashley
5,Christopher,Sarah
. . .
240, Marcos,Gianna
241,Cooper, Juliana
242, Elias,Fatima
243,Brenden,Allyson
244,Israel, Gracie
. . .
```

The first line of each file is the year

Following lines in file have format:

rank, male name, female name

DataProcessing - Milestone 2 - File Format



baby-2000.txt

```
2000
1,Jacob, Emily
2, Michael, Hannah
3, Matthew,Madison
4, Joshua, Ashley
5,Christopher,Sarah
. . .
240, Marcos,Gianna
241,Cooper, Juliana
242, Elias,Fatima
243,Brenden,Allyson
244,Israel, Gracie
. . .
```

The first line of each file is the year

Following lines in file have format:

rank, male name, female name

There may be some extra whitespace chars separating data we care about that you will need to remove

DataProcessing - Milestone 2 - TESTING

DataProcessing - Milestone 2 - TESTING

Tests are provided for this function using the small test files small-2000.txt and small-2010.txt

These will build up a rudimentary name_data dictionary

DataProcessing - Milestone 3

3. **Processing many files and enabling search:** Write one function for processing multiple data files and one function for interacting with our data (searching for data around a specific name).

DataProcessing - Milestone 3

3. **Processing many files and enabling search:** Write one function for **processing multiple data files** and one function for interacting with our data (searching for data around a specific name).

DataProcessing - Milestone 3

3. **Processing many files and enabling search:** Write one function for **processing multiple data files** and one function for interacting with our data (searching for data around a specific name).

```
def read_files(filenames):
“””
Reads the data from all files specified in the provided list
into a single name_data dictionary and then returns that dictionary.
“””
```

DataProcessing - Milestone 3

3. **Processing many files and enabling search:** Write one function for **processing multiple data files** and one function for interacting with our data (searching for data around a specific name).

```
def read_files(filenames):
“””
Reads the data from all files specified in the provided list
into a single name_data dictionary and then returns that dictionary.
“””
```

Input = a list of filenames containing baby name data

DataProcessing - Milestone 3

3. **Processing many files and enabling search:** Write one function for **processing multiple data files** and one function for interacting with our data (searching for data around a specific name).

```
def read_files(filenames):
“””
Reads the data from all files specified in the provided list
into a single name_data dictionary and then returns that dictionary.
“””
```

Input = a list of filenames containing baby name data

Output = name_data (dictionary) storing all baby name data in an effective manner

DataProcessing - Milestone 3

3. **Processing many files and enabling search:** Write one function for processing multiple data files and one function for **interacting with our data (searching for data around a specific name)**.

DataProcessing - Milestone 3

3. **Processing many files and enabling search:** Write one function for processing multiple data files and one function for **interacting with our data (searching for data around a specific name).**

```
def search_names(name_data, target):
“””
Given a name_data dictionary that stores baby name information and a target string,
returns a list of all names in the dictionary that contain the target string.
“””
```

DataProcessing - Milestone 3

3. **Processing many files and enabling search:** Write one function for processing multiple data files and one function for **interacting with our data (searching for data around a specific name).**

```
def search_names(name_data, target):
“””
Given a name_data dictionary that stores baby name information and a target string,
returns a list of all names in the dictionary that contain the target string.
“””
```

Should be case insensitive: ‘aa’ and ‘AA’ should both return ‘Aaliyah’

‘A’ should return both ‘Kara’ and ‘Brahm’

DataProcessing - Milestones 1-3 - TESTING

DataProcessing - Milestones 1-3 - TESTING

`main()` in `babynames.py` to help you test these milestones

DataProcessing - Milestones 1-3 - TESTING

main() in babynames.py to help you test these milestones

1. Testing read_files(filenames)
    a. Provide one or more baby data file arguments, all of which will be passed into the read_files() function you have written
    b. This data is then printed to the console by the print_names() function we have provided, which prints the names in alphabetical order, along with their ranking data.

Examples: (If you are using a mac, use python3 instead of py)

```
> py babynames.py data/small/small-2000.txt data/small/small-2010.txt
A [(2000, 1), (2010, 2)]
B [(2000, 1)]
C [(2000, 2), (2010, 1)]
D [(2010, 1)]
E [(2010, 2)]
```

```
> py babynames.py data/full/baby-1980.txt data/full/baby-1990.txt
data/full/baby-2000.txt data/full/baby-2010.txt
...lots of output...
```

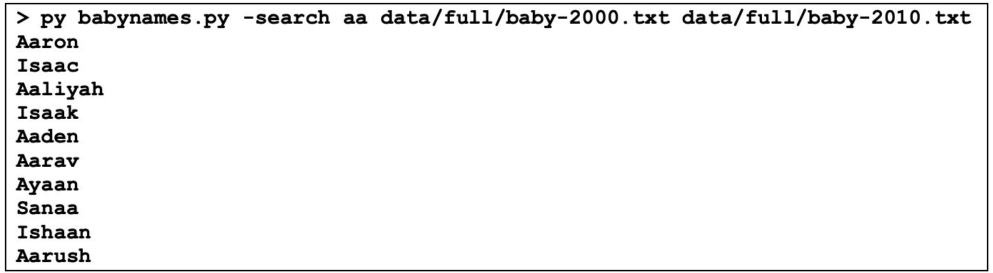
DataProcessing - Milestones 1-3 - TESTING

---

main() in babynames.py to help you test these milestones

2. Testing search_names(name_data, target)

If the first 2 command line arguments are "-search target", then main() reads in all the data, calls your search_names() function to find names that have matches with the target string, and prints those names.



```
> py babynames.py -search aa data/full/baby-2000.txt data/full/baby-2010.txt
Aaron
Isaac
Aaliyah
Isaak
Aaden
Aarav
Ayaan
Sanaa
Ishaan
Aarush
```

Connecting the Data to the Graphics - Milestone 4

Connecting the Data to the Graphics - Milestone 4

There is provided code in babygraphics.py to set up a drawing canvas and the ability to enter names and target strings.

Connecting the Data to the Graphics - Milestone 4

There is provided code in babygraphics.py to set up a drawing canvas and the ability to enter names and target strings.

In babygraphics.py, the provided main() function takes care of calling your babynames.read_files() function to read in the baby name data and populate the name_data dictionary.

Connecting the Data to the Graphics - Milestone 4

There is provided code in babygraphics.py to set up a drawing canvas and the ability to enter names and target strings.

In babygraphics.py, the provided main() function takes care of calling your babynames.read_files() function to read in the baby name data and populate the name_data dictionary.

Your job: figuring out how to write functions to graph the contents of the name_data dictionary.

Connecting the Data to the Graphics - Milestone 4

First, run the command: (use python3 for macs)

```
> py babygraphics.py
```

That will pop up a blank baby name graphical window

Connecting the Data to the Graphics - Milestone 4

First, run the command: (use python3 for macs)

```
> py babygraphics.py
```

That will pop up a blank baby name graphical window

To test this (and to test out the search_name()) function you wrote in Milestone 3, type a search string into the text field at the bottom of the window and then hit enter.

Connecting the Data to the Graphics - Milestone 4

First, run the command: (use python3 for macs)

```
> py babygraphics.py
```

That will pop up a blank baby name graphical window

To test this (and to test out the search_name()) function you wrote in Milestone 3, type a search string into the text field at the bottom of the window and then hit enter.

You should see a text field pop up in the bottom of the screen showing all names in the data set that match the search string.

Connecting the Data to the Graphics - Milestone 4

First, run the command: (use python3 for macs)

```
> py babygraphics.py
```

That will pop up a blank baby name graphical window

To test this (and to test out the search_name()) function you wrote in Milestone 3, type a search string into the text field at the bottom of the window and then hit enter.

You should see a text field pop up in the bottom of the screen showing all names in the data set that match the search string.

Once you see that this works correctly, you have completed this milestone !!!

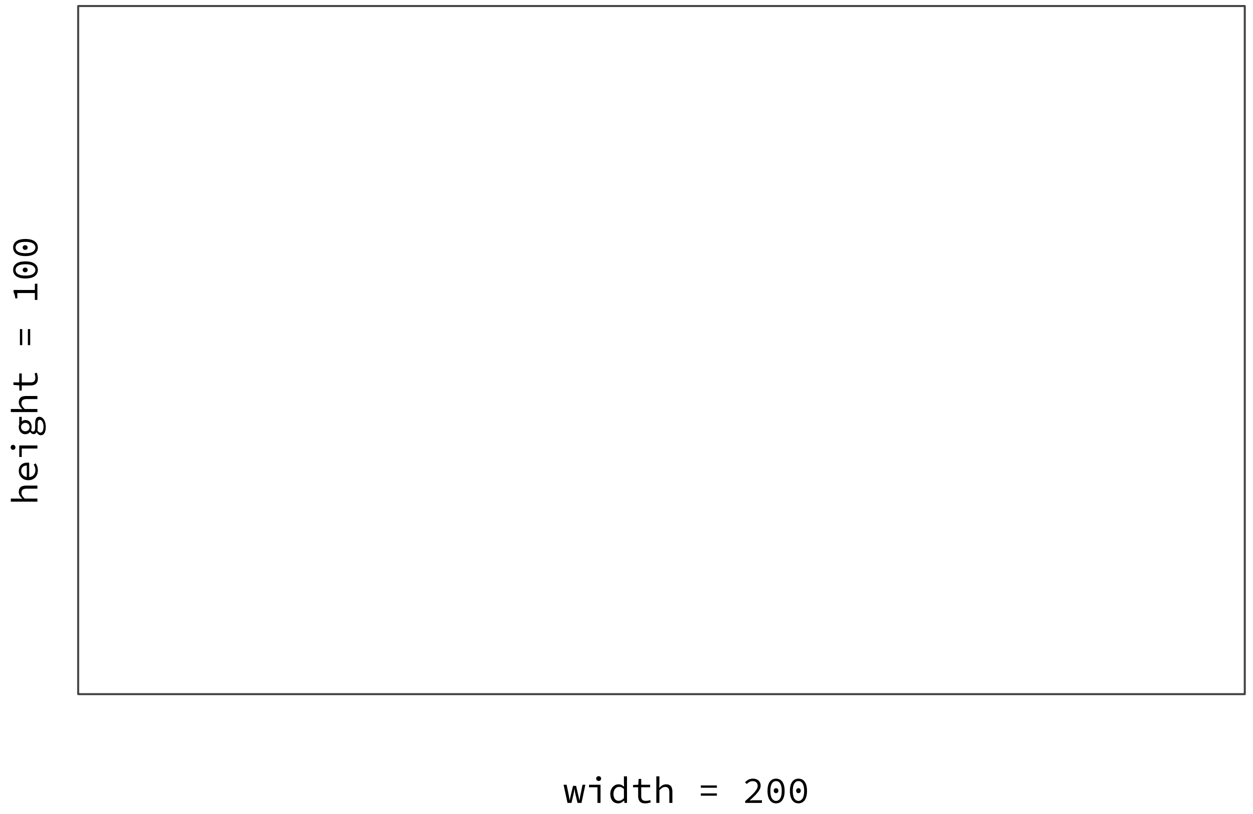
Data Visualization

- Milestone 5: Drawing the background grid
- Milestone 6: Plot the baby name data
- Let's do an example

Let's see a smaller version

- GRAPH_MARGIN_SIZE = 10
- Instead of 1000 ranks, there are only 10
- Instead of 11 decades, there are only 4

Milestone 5: Draw the background grid

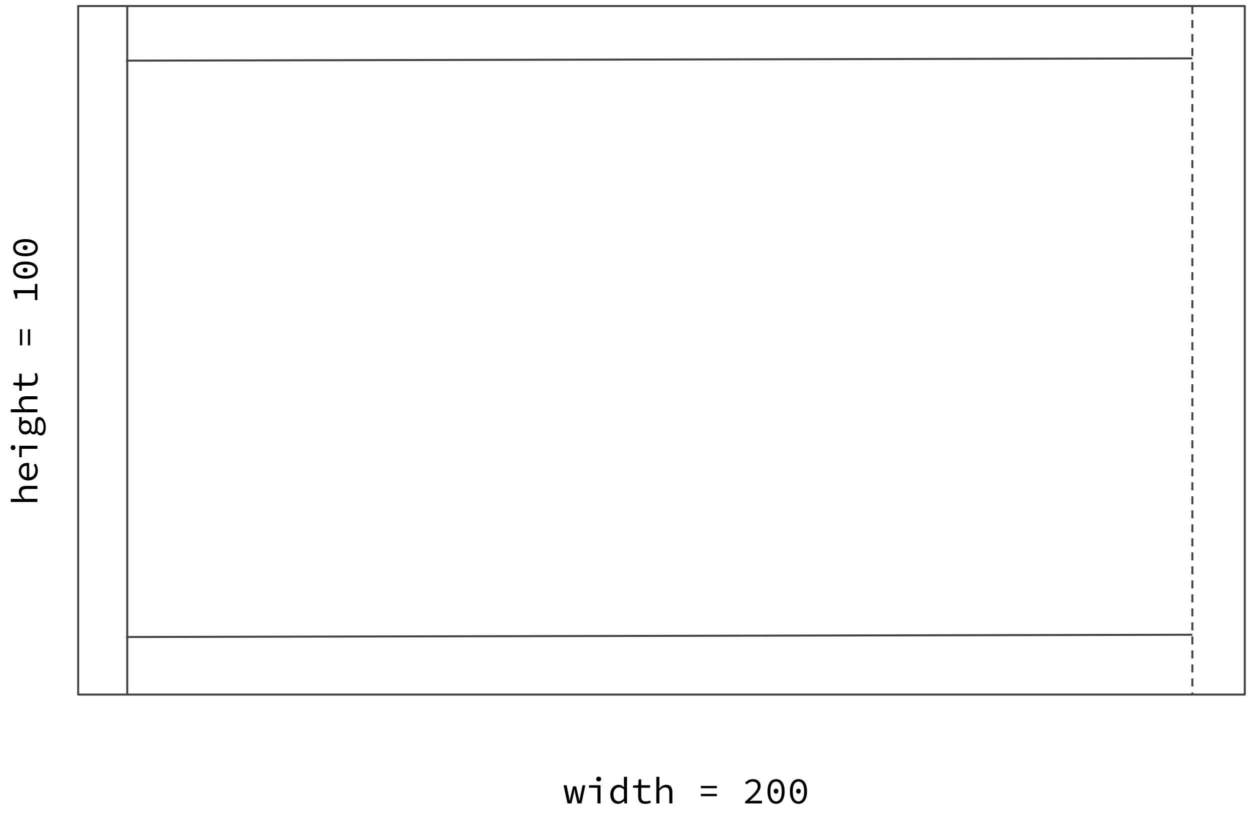


Let's see a smaller version

- `GRAPH_MARGIN_SIZE = 10`
- Instead of 1000 ranks, there are only 10
- Instead of 11 decades, there are only 4

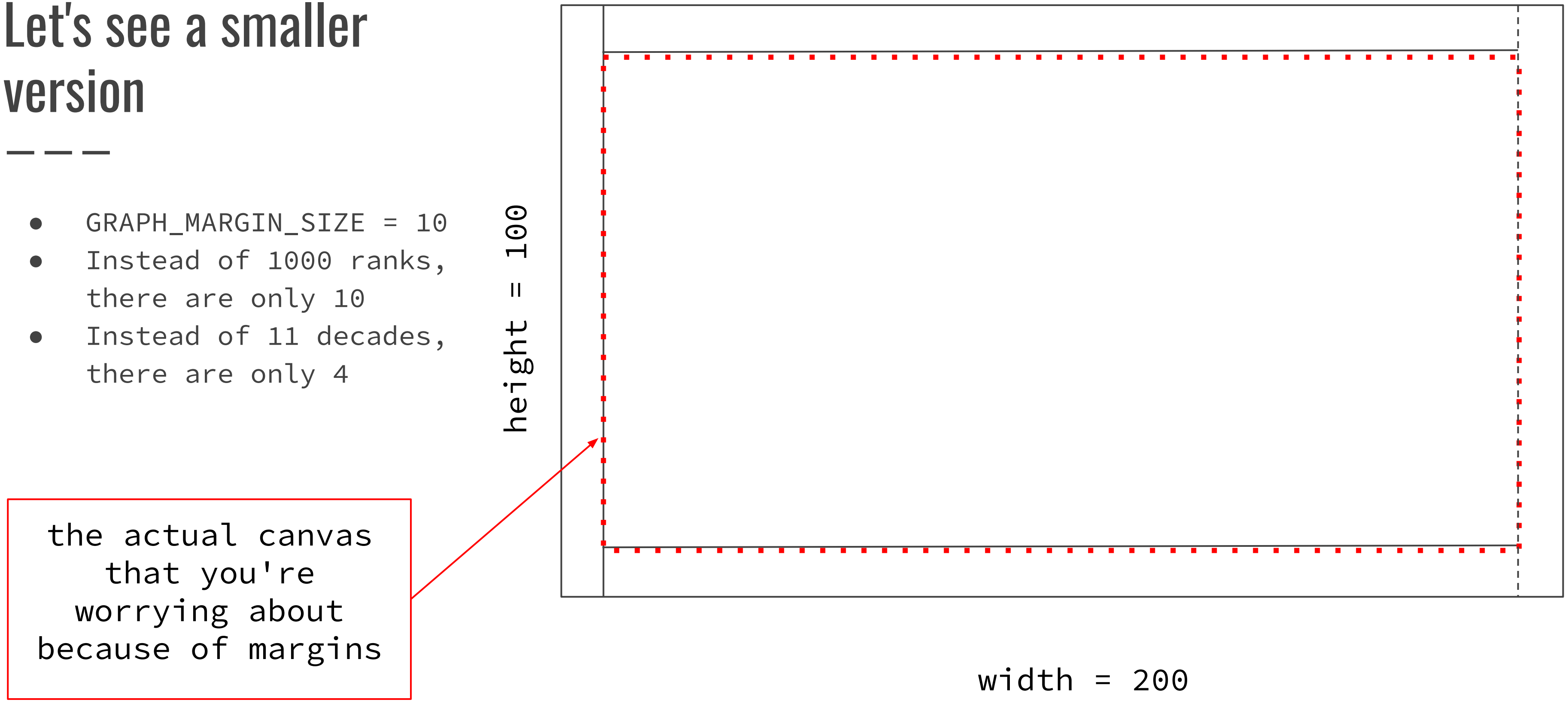
Draw in the margins

(not the one on the right)



Let's see a smaller version

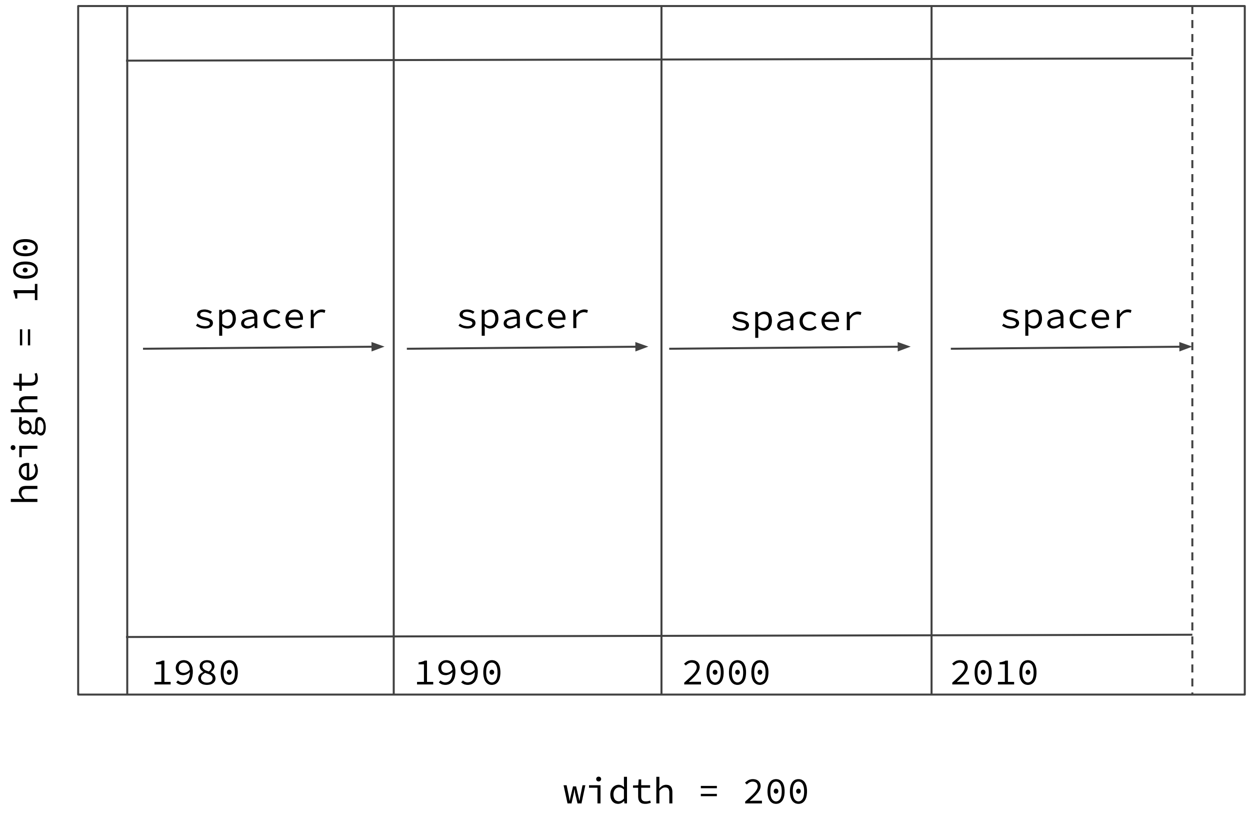
- GRAPH_MARGIN_SIZE = 10
- Instead of 1000 ranks, there are only 10
- Instead of 11 decades, there are only 4


the actual canvas that you're worrying about because of margins
height = 100
width = 200

Let's see a smaller version

- `GRAPH_MARGIN_SIZE = 10`
- `Instead of 1000 ranks, there are only 10`
- `Instead of 11 decades, there are only 4`

Draw the decade lines and labels

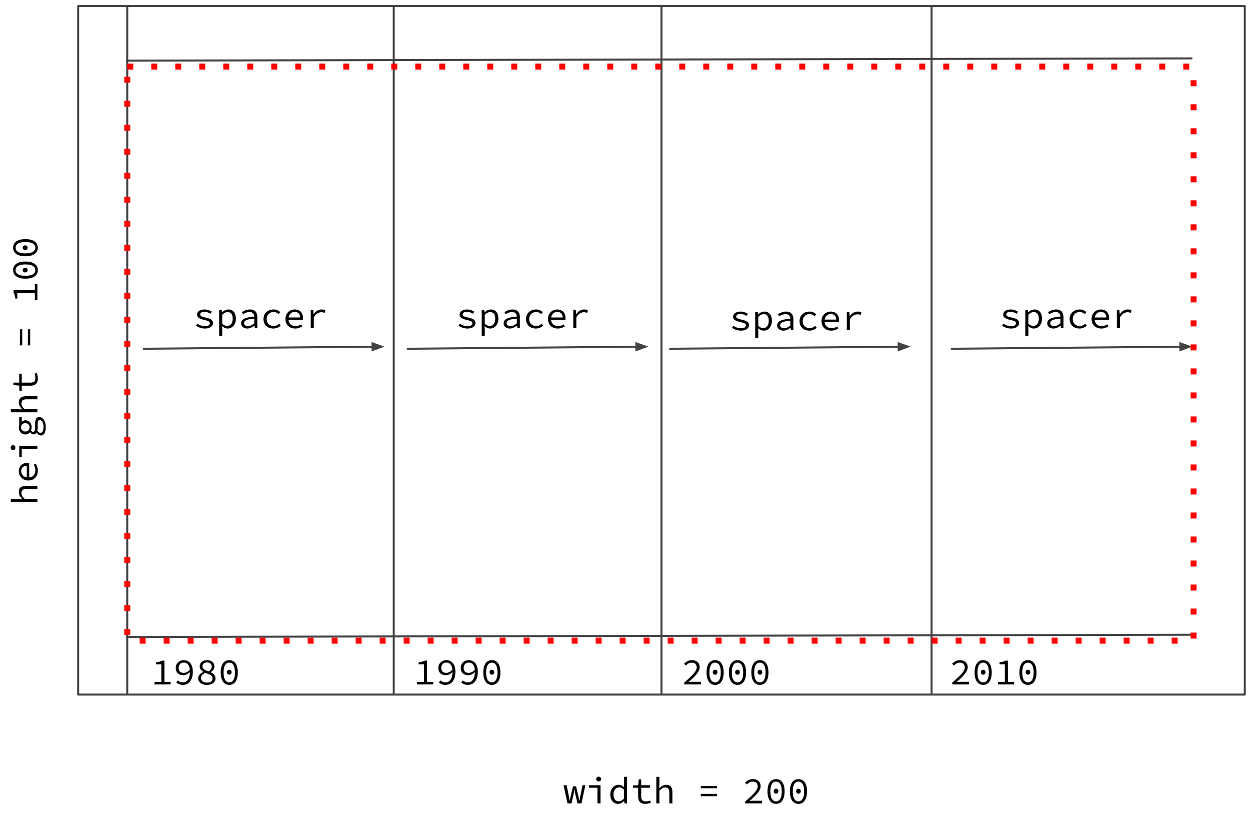


Let's see a smaller version

- GRAPH_MARGIN_SIZE = 10
- Instead of 1000 ranks, there are only 10
- Instead of 11 decades, there are only 4

Draw the decade lines

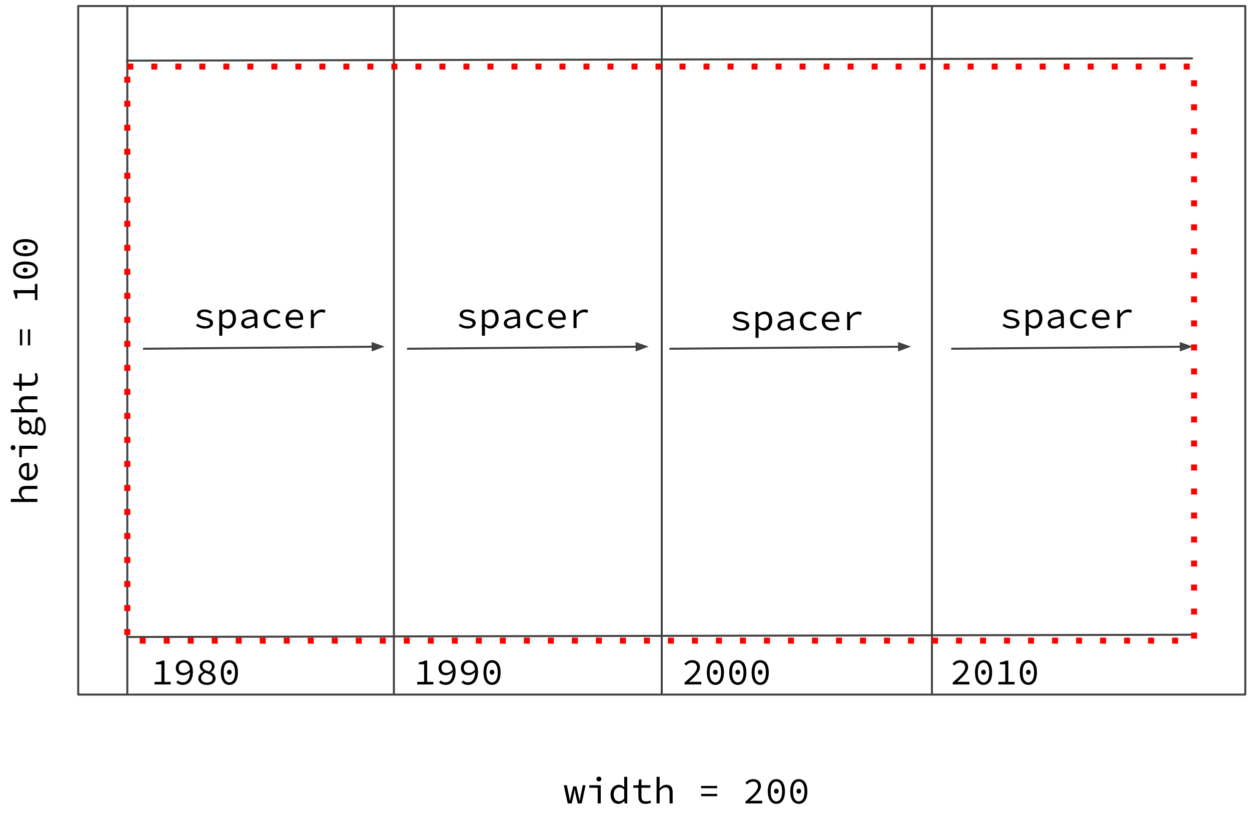
They'll evenly divide the actual_width



Let's see a smaller version

- `GRAPH_MARGIN_SIZE = 10`
- Instead of 1000 ranks, there are only 10
- Instead of 11 decades, there are only 4

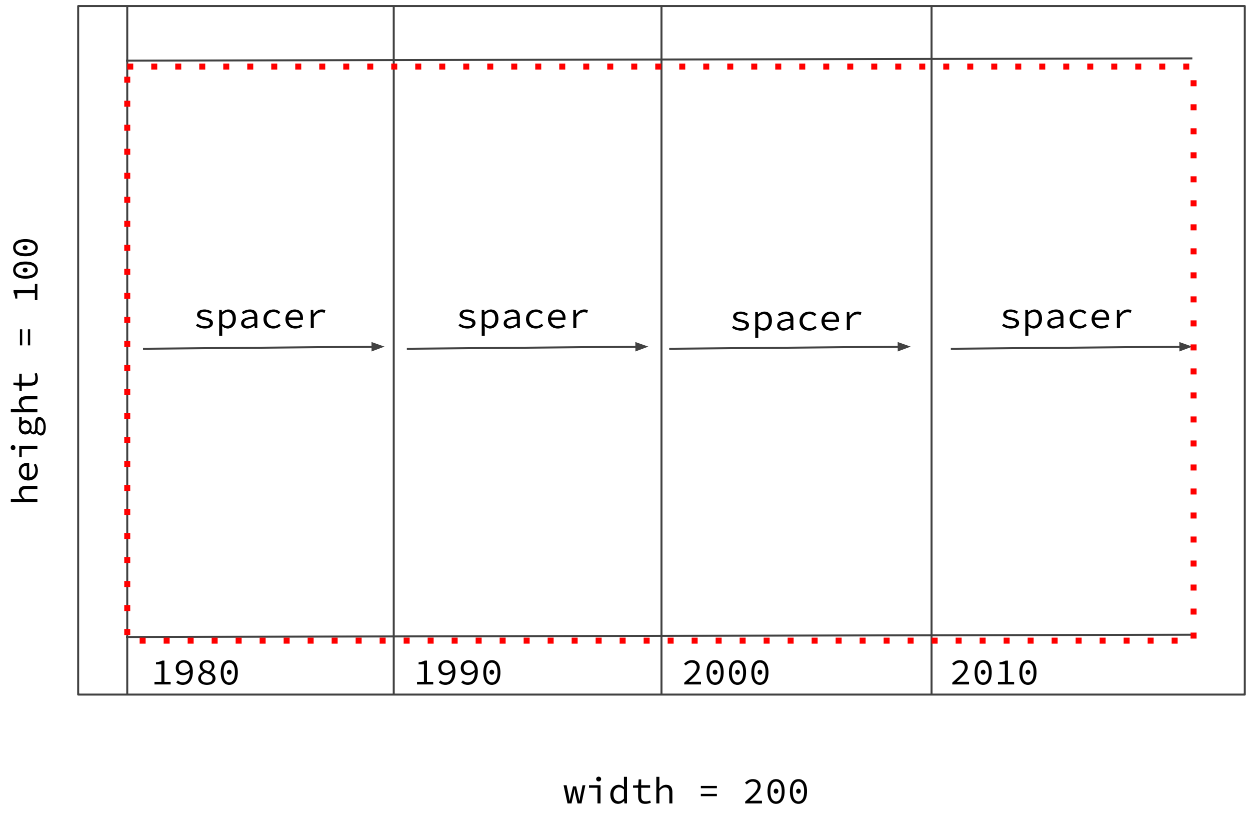
Done with Milestone 5



Let's see a smaller version

- `GRAPH_MARGIN_SIZE = 10`
- Instead of 1000 ranks, there are only 10
- Instead of 11 decades, there are only 4

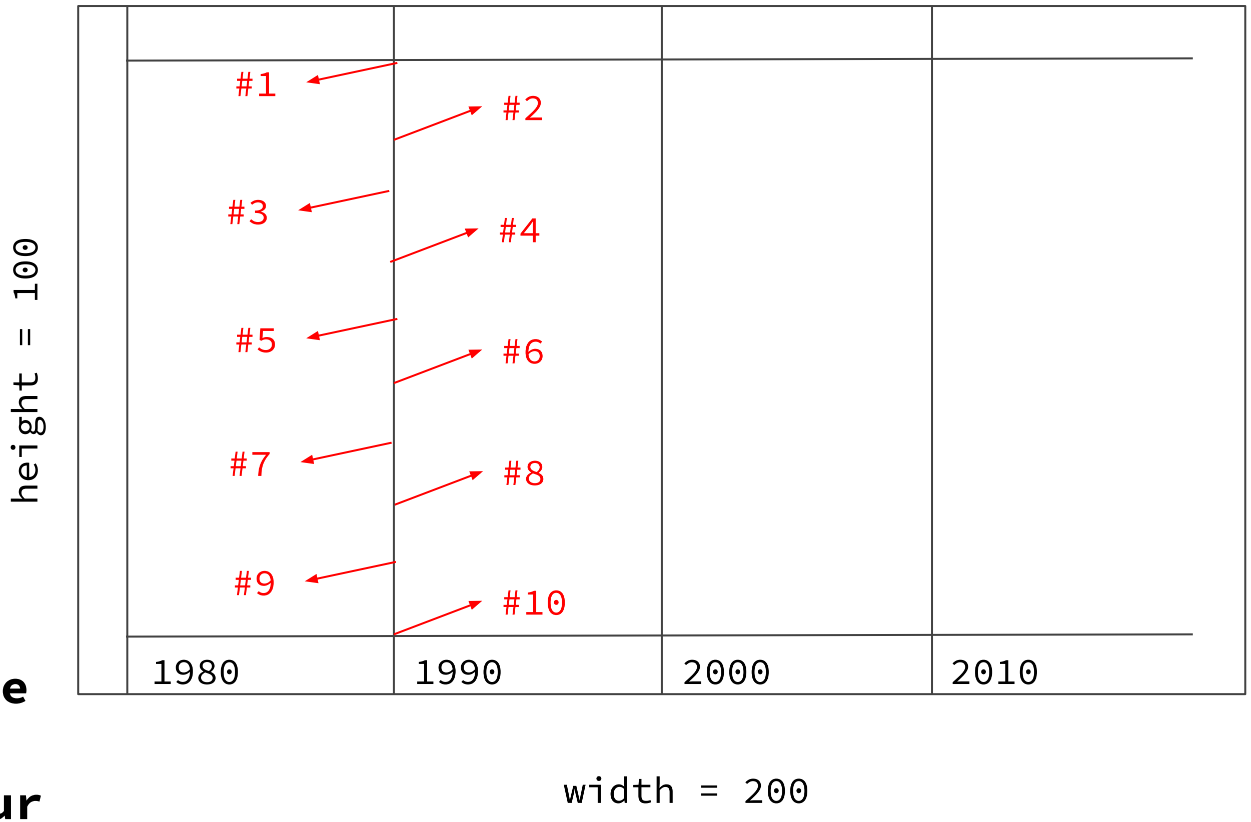
Starting Milestone 6



Let's see a smaller version

- GRAPH_MARGIN_SIZE = 10
- Instead of 1000 ranks, there are only 10
- Instead of 11 decades, there are only 4

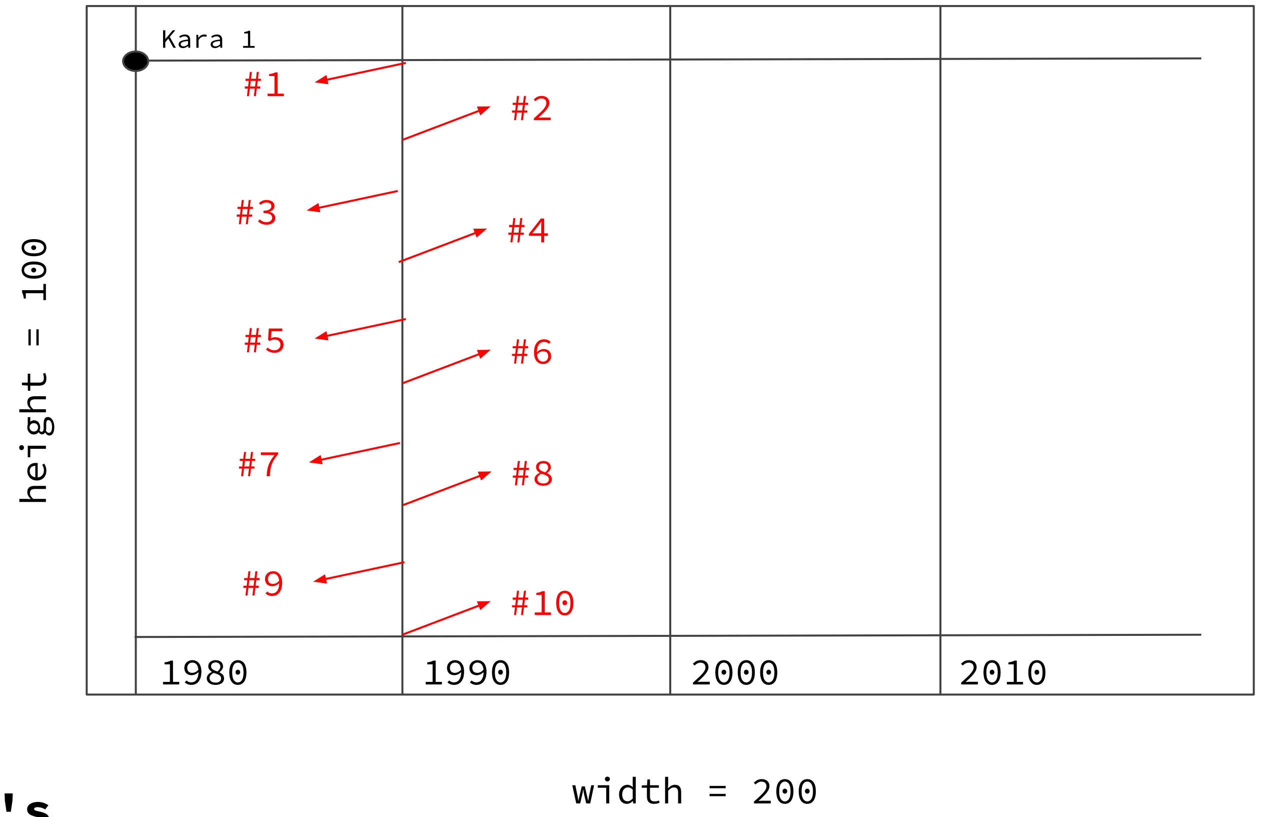
Divide the <span>actual_height</span> you're working with to be proportional with our ranks



Let's see a smaller version

- GRAPH_MARGIN_SIZE = 10
- Instead of 1000 ranks, there are only 10
- Instead of 11 decades, there are only 4

```
{'Kara': {1980: 1, 1990: 6,
2000: 4, 2010: 2},
'Juliette':{1990: 8, 2000: 1,
2010: 1}}
```



Graph a single name's rank for one decade

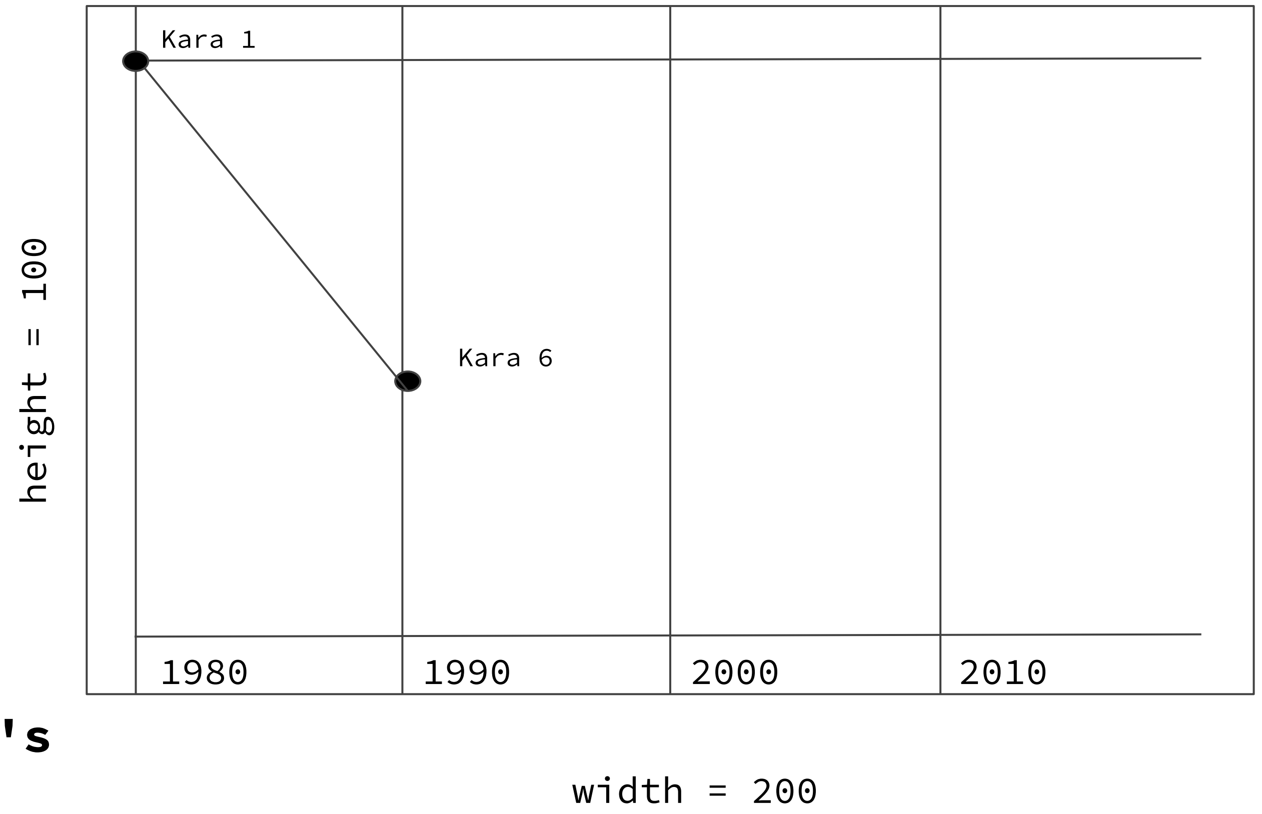
Note: you don't actually draw the points, you just draw the lines connecting them

Let's see a smaller version

- GRAPH_MARGIN_SIZE = 10
- Instead of 1000 ranks, there are only 10
- Instead of 11 decades, there are only 4

```
{'Kara': {1980: 1, 1990: 6,
2000: 4, 2010: 2},
'Juliette':{1990: 8, 2000: 1,
2010: 1}}
```

Graph a single name's rank for the next decade and connect the points



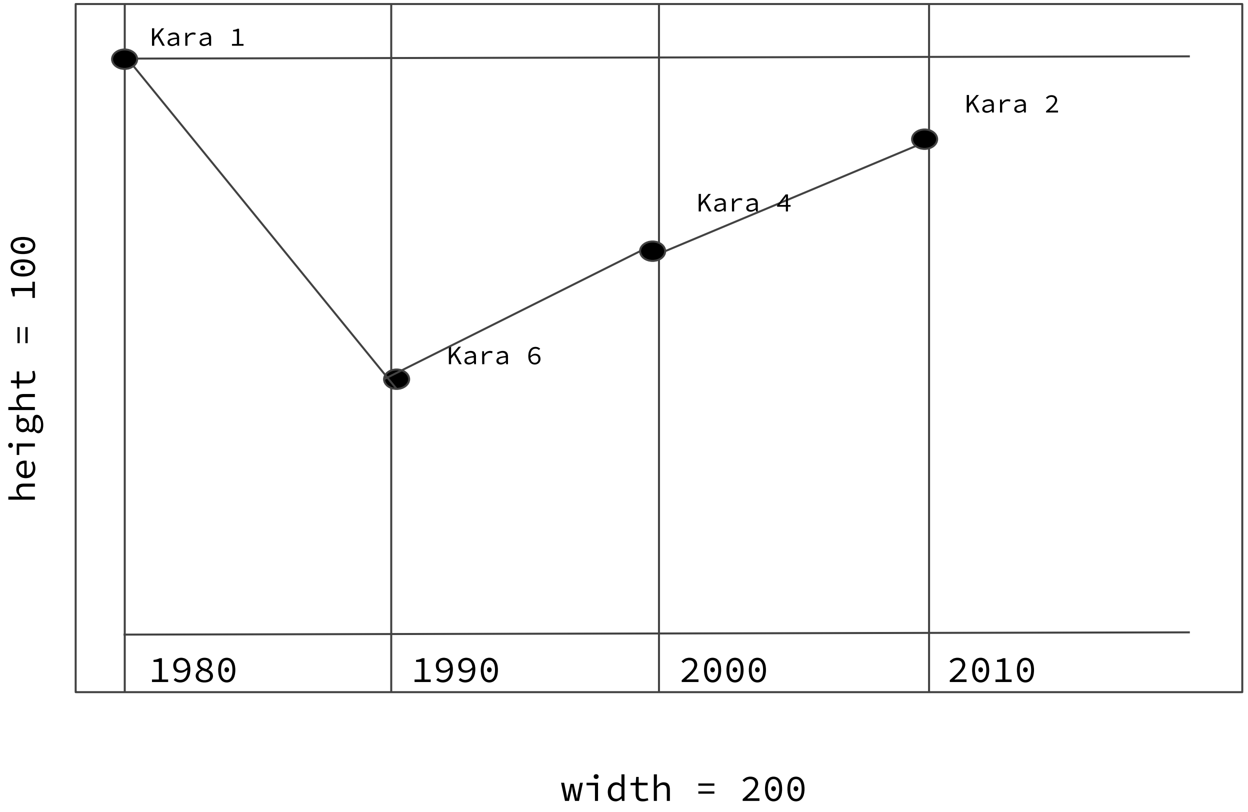
Note: you don't actually draw the points, you just draw the lines connecting them

Let's see a smaller version

- GRAPH_MARGIN_SIZE = 10
- Instead of 1000 ranks, there are only 10
- Instead of 11 decades, there are only 4

```
{'Kara': {1980: 1, 1990: 6,
2000: 4, 2010: 2},
'Juliette':{1990: 8, 2000: 1,
2010: 1}}
```

Repeat for all decades

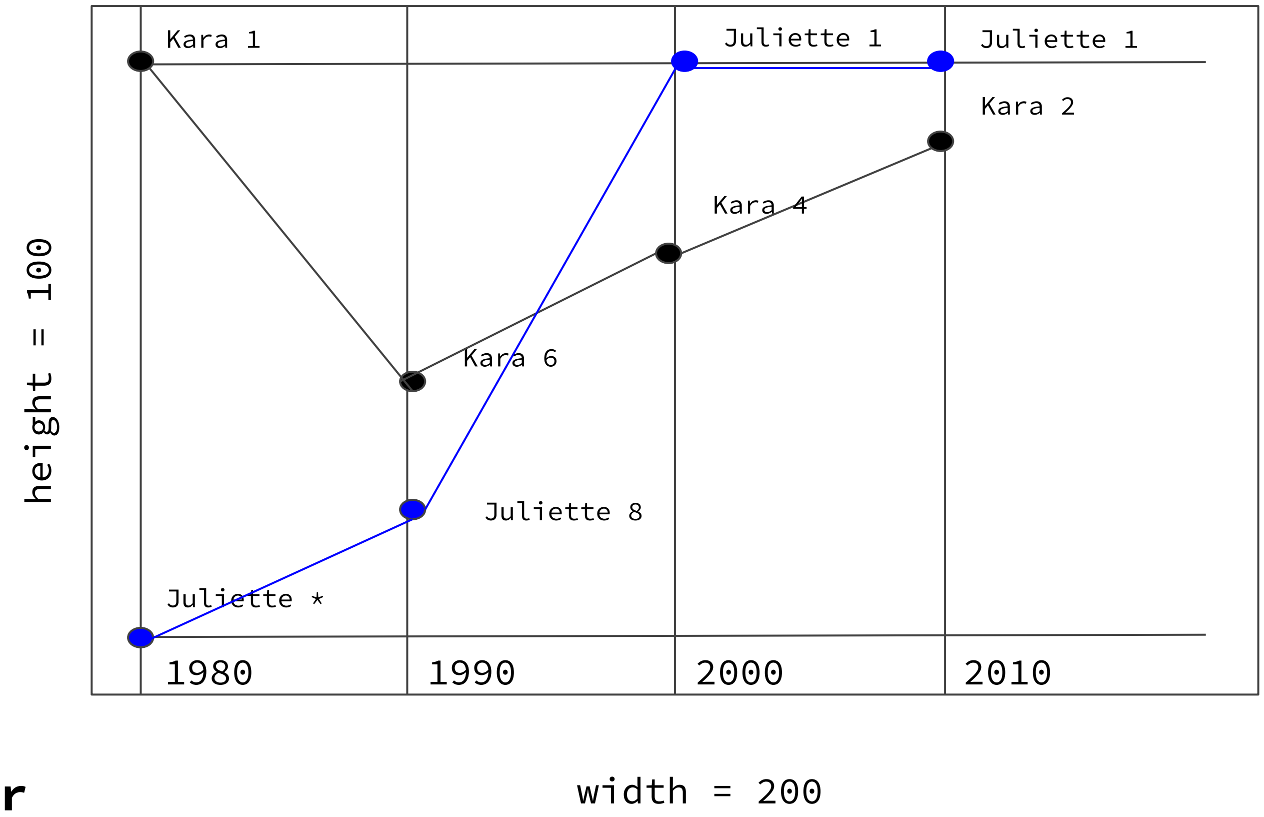


Note: you don't actually draw the points, you just draw the lines connecting them

Let's see a smaller version

- GRAPH_MARGIN_SIZE = 10
- Instead of 1000 ranks, there are only 10
- Instead of 11 decades, there are only 4

```
{'Kara': {1980: 1, 1990: 6,
2000: 4, 2010: 2},
'Juliette':{1990: 8, 2000: 1,
2010: 1}}
```



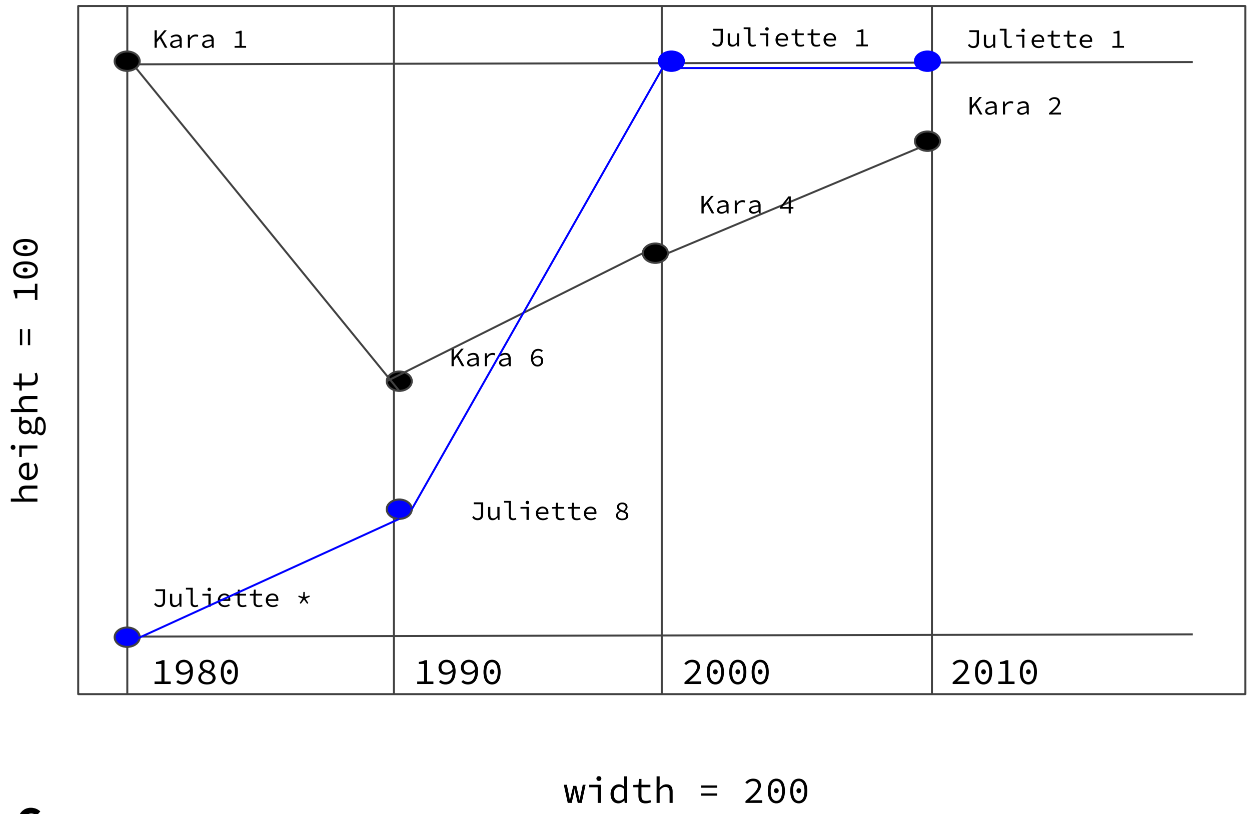
Do it for all other specified names

Note: you don't actually draw the points, you just draw the lines connecting them

Let's see a smaller version

- GRAPH_MARGIN_SIZE = 10
- Instead of 1000 ranks, there are only 10
- Instead of 11 decades, there are only 4

```
{'Kara': {1980: 1, 1990: 6,
2000: 4, 2010: 2},
'Juliette':{1990: 8, 2000: 1,
2010: 1}}
```



Done with Milestone 6

Note: you don't actually draw the points, you just draw the lines connecting them

Check out this week's section problems with Big Tweet Data for help

Good luck!