Lecture 26:
List Comprehensions and matplotlib

Elyse Cornwall

List Comprehensions

Goal: double a list of numbers

```
[1, 2, 3, 4, 5] -> [2, 4, 6, 8, 10]
```

Goal: double a list of numbers

Function def

```
def double(nums):
    doubles = []
    for n in nums:
        doubles.append(n * 2)
    return doubles
```

okay…

Goal: double a list of numbers

Map with lambda

oh, nice :)

```
map(lambda n: n * 2, nums)
```

Goal: double a list of numbers

List comprehension

beautiful :')

```
[n * 2 for n in nums]
```

1, 2, 3 of List Comprehensions

1, 2, 3 of List Comprehensions

[ ]

1. Square brackets

1, 2, 3 of List Comprehensions

```
[      for n in nums]
```

1. Square brackets
2. For-each over input list

1, 2, 3 of List Comprehensions

```
[n * 2 for n in nums]
```

1. Square brackets
2. For-each over input list
3. Output expression to produce

Let’s try some more

Given a list of positive numbers, like:

```
nums = [1, 2, 3, 4, 5]
```

- Make a list of these numbers’ squares
  - `[1, 4, 9, 16, 25]`
- Make a list of these numbers, but negative
  - `[-1, -2, -3, -4, -5]`

Now, you try list comprehensions!

Given a list of positive numbers, like:

```
nums = [1, 2, 3, 4, 5]
```

- Make a list of these numbers with 10 added to them
  - `[11, 12, 13, 14, 15]`
- Make a list of these numbers cast to strings with “!!” appended on the end
  - `['1!!', '2!!', '3!!', '4!!', '5!!']`

List Comprehensions

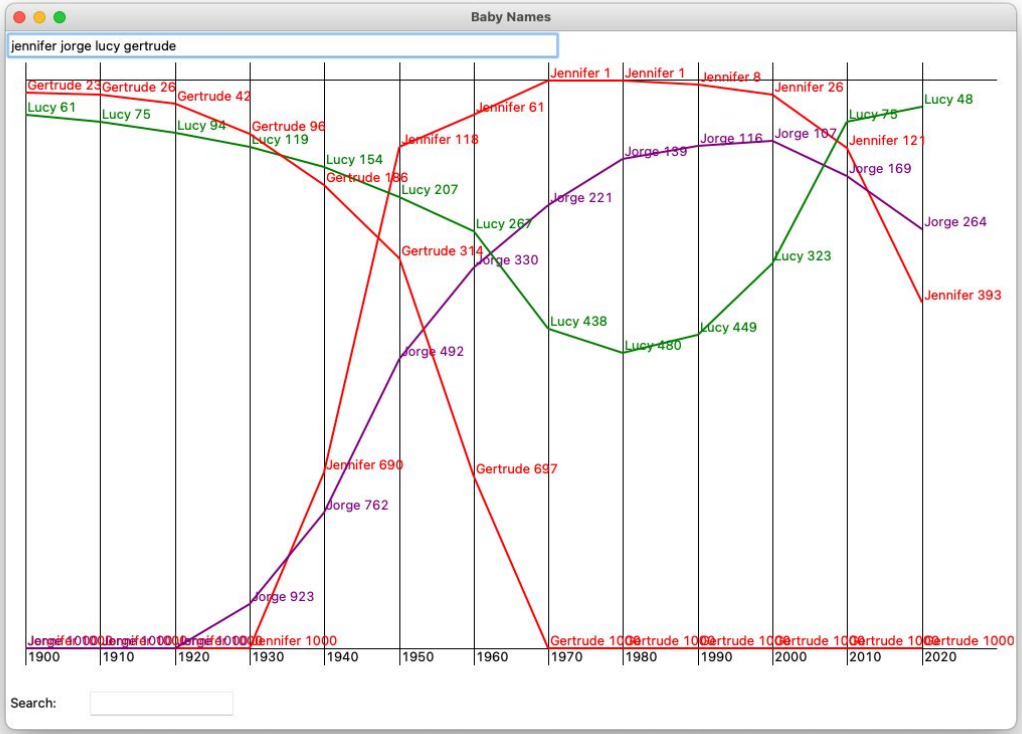
very nice

```
[n * 2 for n in nums]
```

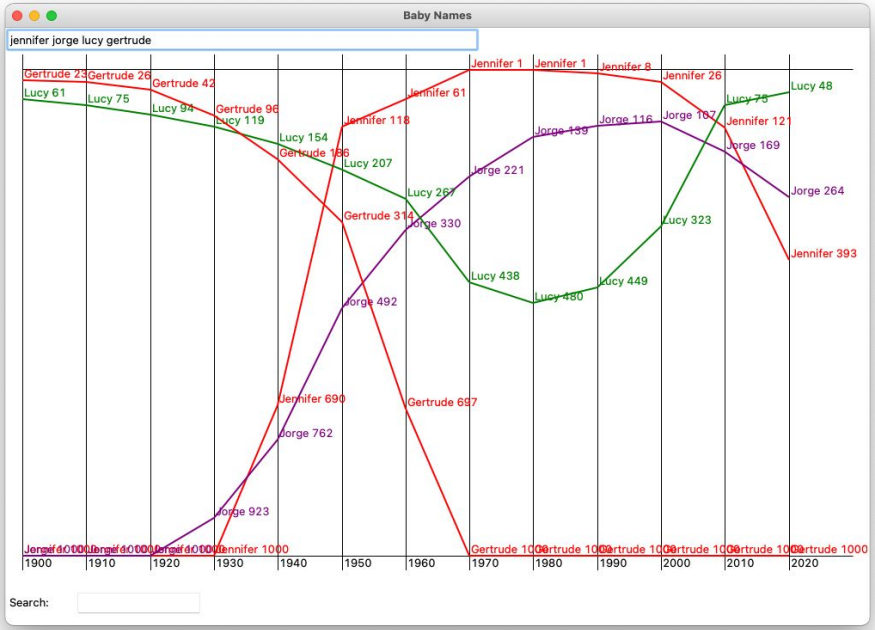
- Like `map()`, but a bit more readable
- Cool? Elegant?
- Fits problems that are in a specific form
  - We have list XXX and want it in form YYY

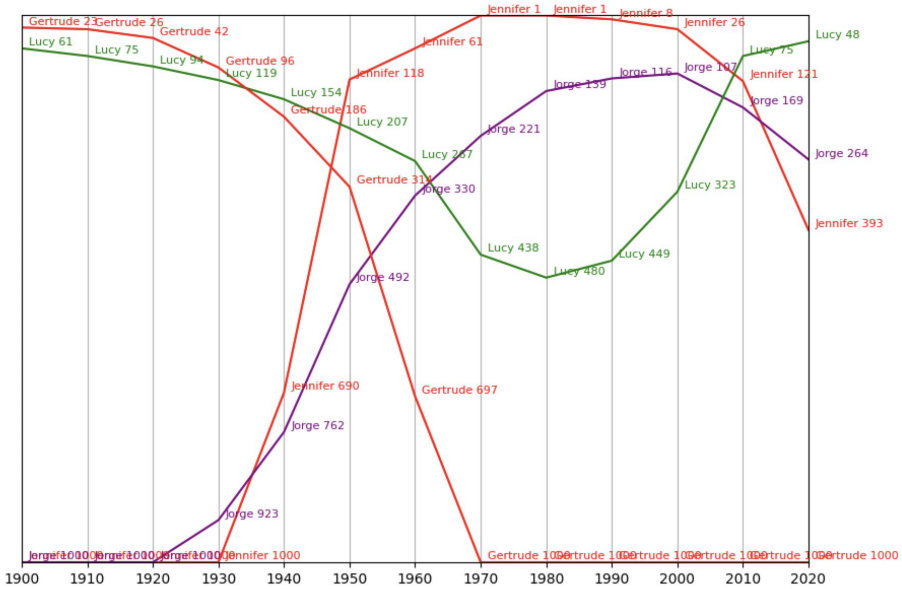
matplotlib

Remember this?



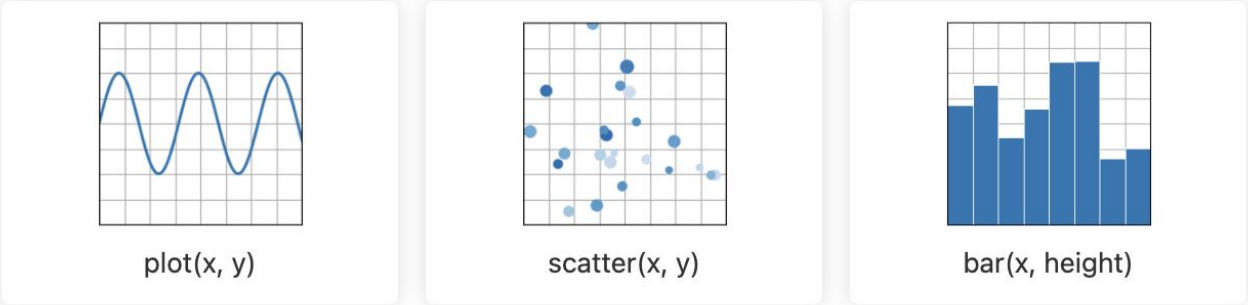
`matplotlib` can do it, too





Generating Visualizations IRL

- It’s tedious to draw all the shapes on a graph
- Python module for making visualizations more easily - so many options!



To install: `python3 -m pip install matplotlib`

Example: CS Course Enrollment

Let’s say we want to visualize this enrollment data:

- CS109 has 198 students
- CS124 has 338 students
- CS182 has 145 students

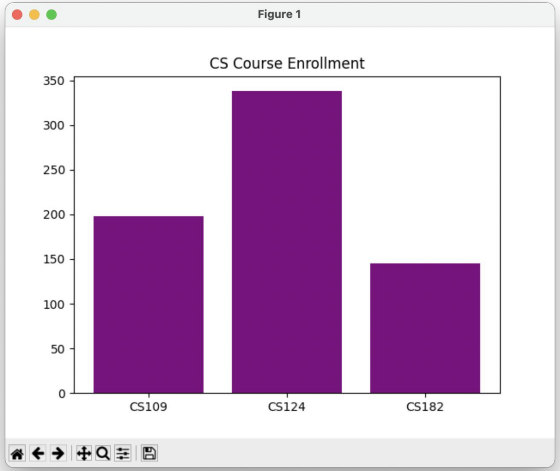
Which is stored as a dictionary:

```
enrolls = {'CS109': 198, 'CS124': 338, 'CS182': 145}
```

Example: CS Course Enrollment

Let's say we want to visualize this enrollment data:

- CS109 has 198 students
- CS124 has 338 students
- CS182 has 145 students



Which is stored as a dictionary:

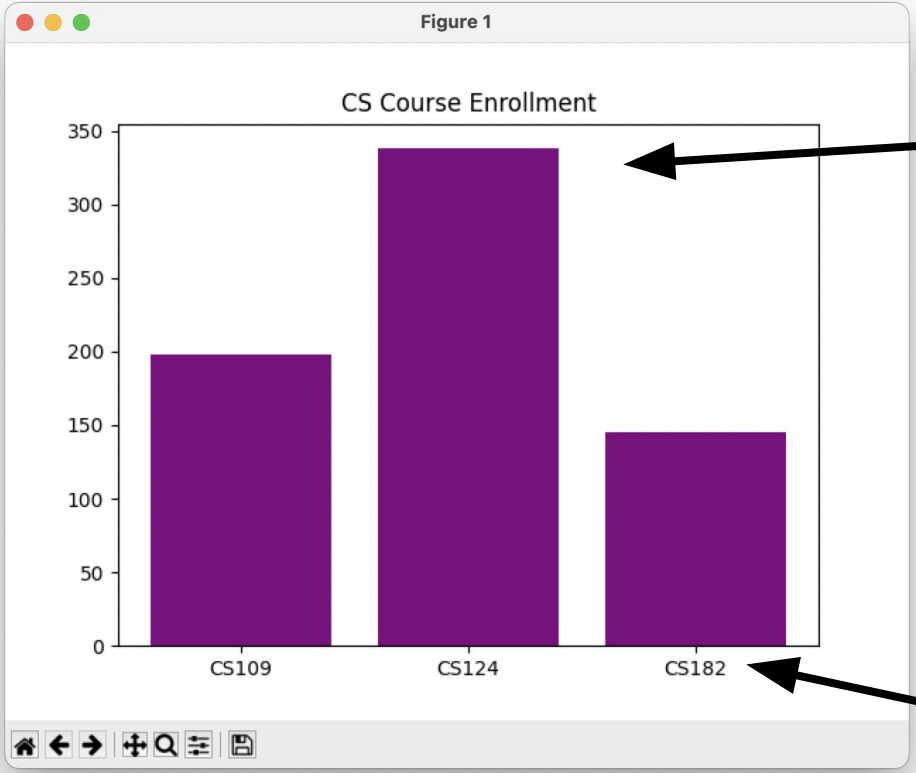
```
enrolls = {'CS109': 198, 'CS124': 338, 'CS182': 145}
```

Example: CS Course Enrollment

```
import matplotlib.pyplot as plt       # at the top of your file
```

Example: CS Course Enrollment

```
import matplotlib.pyplot as plt       # at the top of your file


x_vals = ???

y_vals = ???
```



y values
(enrollments)

x values
(course names)

Example: CS Course Enrollment

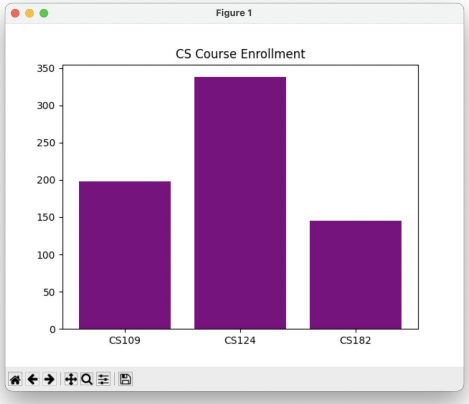
```
import matplotlib.pyplot as plt       # at the top of your file


x_vals = ['CS109', 'CS124', 'CS182']

y_vals = [enrolls['CS109'], enrolls['CS124'], enrolls['CS182']]

# y_vals in this example are [198, 338, 145]
```

Example: CS Course Enrollment

```
import matplotlib.pyplot as plt       # at the top of your file


x_vals = ['CS109', 'CS124', 'CS182']

y_vals = [enrolls['CS109'], enrolls['CS124'], enrolls['CS182']]

plt.bar(x_vals, y_vals, color='purple')
```

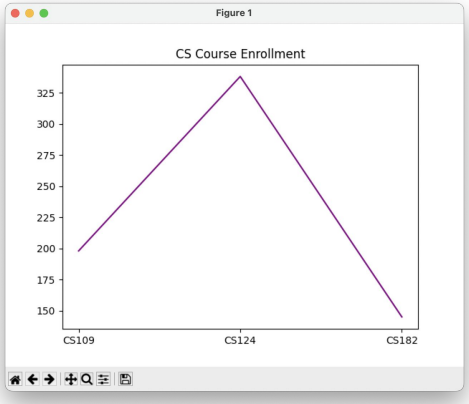
Example: CS Course Enrollment

```
import matplotlib.pyplot as plt       # at the top of your file


x_vals = ['CS109', 'CS124', 'CS182']

y_vals = [enrolls['CS109'], enrolls['CS124'], enrolls['CS182']]

plt.bar(x_vals, y_vals, color='purple')


plt.title('CS Course Enrollment')

plt.show()
```



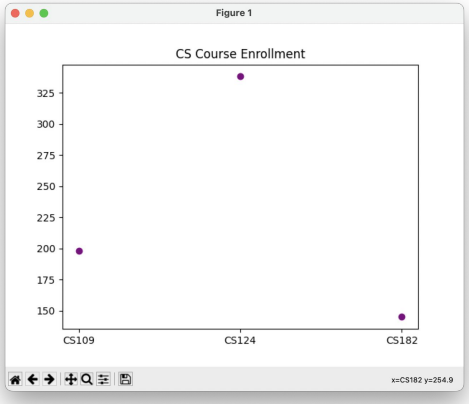
Example: CS Course Enrollment

```
import matplotlib.pyplot as plt       # at the top of your file


x_vals = ['CS109', 'CS124', 'CS182']

y_vals = [enrolls['CS109'], enrolls['CS124'], enrolls['CS182']]

plt.plot(x_vals, y_vals, color='purple')


plt.title('CS Course Enrollment')

plt.show()
```



Example: CS Course Enrollment

```
import matplotlib.pyplot as plt       # at the top of your file


x_vals = ['CS109', 'CS124', 'CS182']

y_vals = [enrolls['CS109'], enrolls['CS124'], enrolls['CS182']]

plt.scatter(x_vals, y_vals, color='purple')


plt.title('CS Course Enrollment')

plt.show()
```



Example: CS Course Enrollment

What if my dataset was a little larger:

CS109 has 198 students

CS124 has 338 students

CS182 has 145 students

CS103 has 419 students

CS106A has 463 students

CS106B has 318 students

CS107 has 285 students

CS108 has 62 students

…

Example: CS Course Enrollment

```
enrolls = {'CS109': 198, 'CS124': 338, 'CS182': 145,
'CS103': 419, 'CS106A': 463, 'CS106B': 318, 'CS107': 285,
'CS108': 62, ...}
```

Example: CS Course Enrollment

```
enrolls = {'CS109': 198, 'CS124': 338, 'CS182': 145,
'CS103': 419, 'CS106A': 463, 'CS106B': 318, 'CS107': 285,
'CS108': 62, ...}
```

```
x_vals = ['CS109', 'CS124', 'CS182', 'CS103', ...]
```

```
y_vals = [enrolls['CS109'], enrolls['CS124'], enrolls['CS182'], ...]
```

Wait a minute, I don’t want to type all that!

PyCharm Demo (zip file)

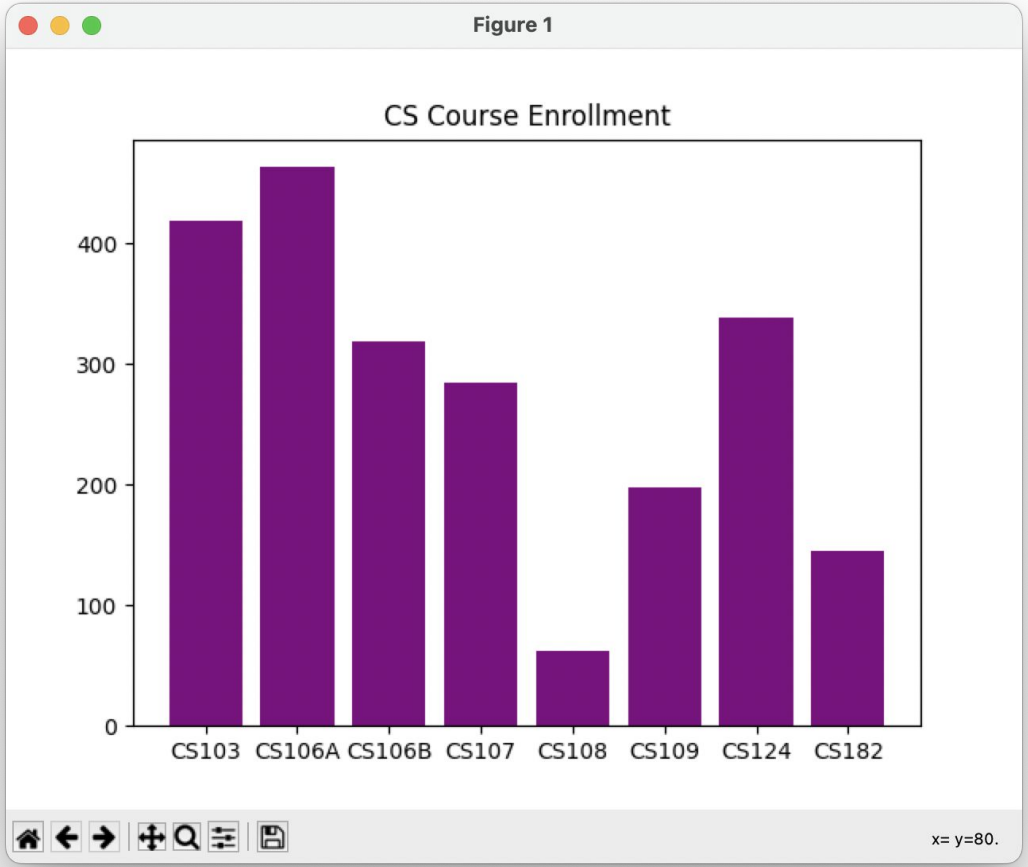


Figure 1
CS Course Enrollment
400
300
200
100
0
CS103
CS106A
CS106B
CS107
CS108
CS109
CS124
CS182
x= y=80.

Example: CS Course Enrollment

```
enrolls = {'CS103': 419, 'CS106A': 463, 'CS106B': 318,
'CS107': 285, 'CS108': 62, 'CS109': 198, 'CS124': 338,
'CS182': 145, ...}
```

```
x_vals = ['CS103', 'CS106A', 'CS106B', ...]
```

```
y_vals = [enrolls[course] for course in x_vals]
```

List comprehensions to the rescue!