CS134 Lecture 17:
Files & Plotting

Announcements & Logistics

- Almost Spring Break!
- Lab 5 due Friday noon (*after* midterm)
- **Midterm reminders:**
  - Review examples posted on course schedule
  - Sample Exam. Sample solutions posted (many possible ans)
  - **Exam Thurs 3/14** from 6-7:30pm OR 8-9:30pml in Bronfman
  - You need your anonymous ID!
- Instructor Help Hours Schedule: Wed 1-4 pm, Thurs 1-4 pm
- TA hours as usual this week

Do You Have Any Questions?

Last Time

- Discussed testing and debugging strategies
    - how to approach different types of errors
    - using print to debug loops and conditionals

Today

- Discuss a new "iterable" : files
    - How to loop over files
    - How to process the data, store it, manipulate it and plot it

Files

Working with Files in Python

- File I/O is a very common and important operation
- `open(filename)` is a built-in Python function for working with files
    - `filename` is a path to a file as a **string**
- Using `open()` within a `with … as` code block, we can **iterate** over the **lines of a text file** just as we iterated over strings and lists in previous lectures

Opening Files: `with … as`

Path to file on computer as a string

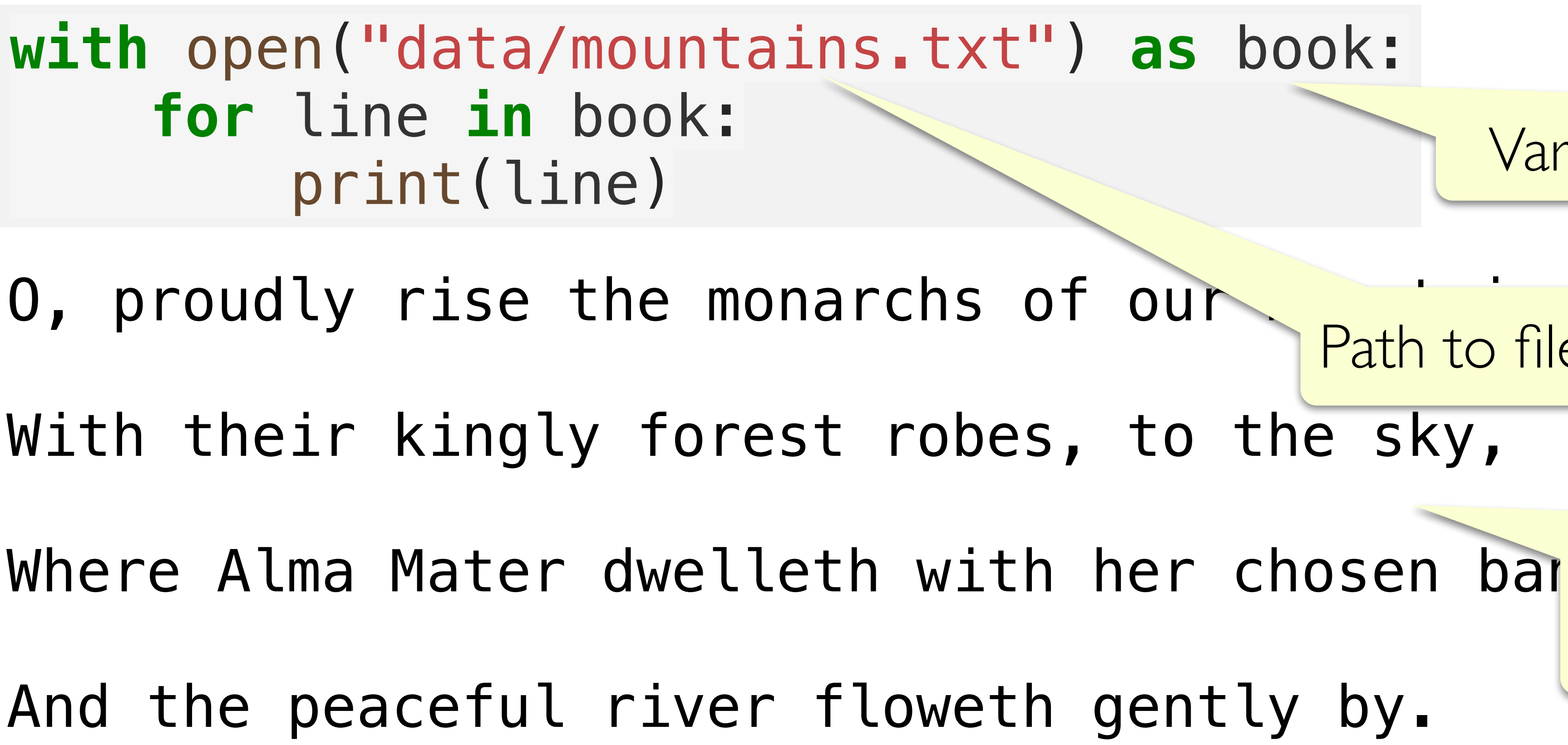
```
with open(filename) as input_file:
    # do something with file
```

Variable name for your file

Note. **(syntax)** Indentation defines the body of the with block where the file is open. File automatically closed after with…as block.

Iterating over Lines in a File

- Within a `with open(filename) as input_file:` block, we can iterate over the lines in the file just as we would iterate over any sequence such as lists, strings, or ranges
- The end of a line in the text file is determined by the special newline character `'\n'`
- Example: We have a text file `mountains.txt` within a directory `data`. We can iterate and print each line as follows:



```
with open("data/mountains.txt") as book:
    for line in book:
        print(line)
```

```
O, proudly rise the monarchs of our

With their kingly forest robes, to the sky,

Where Alma Mater dwelleth with her chosen ba

And the peaceful river floweth gently by.
```

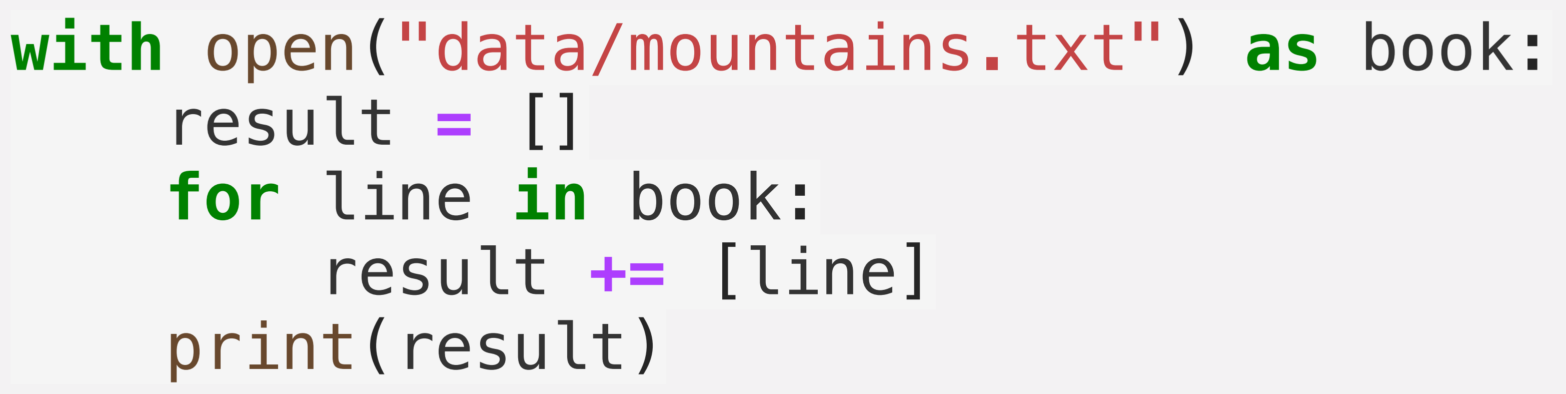
Variable name for your file

Path to file on computer as a string

`'\n'` between each line

Iterating over Lines in a File

- Because the end of the line in a file is a newline character `'\n'` and when we `print(a_string)` a newline character is added to the end...we end up with an empty newline between each printed line!



```python
with open("data/mountains.txt") as book:
    result = []
    for line in book:
        result += [line]
    print(result)
```

```
['O, proudly rise the monarchs of our mountain land,\n',
 'With their kingly forest robes, to the sky,\n',
 'Where Alma Mater dwelleth with her chosen band,\n',
 'And the peaceful river floweth gently by.\n']
```

Removing Leading/Trailing Whitespace from a String

- Because the end of the line in a file is a newline character `'\n'` and when we `print(a_string)` a newline character is added to the end...we end up with an empty newline between each printed line!
- Let's write a function that will remove leading and trailing whitespaces.

```
s = '\n  \t  String with\t different\nspaces.\r\n\t'
```

`'\n'` newline

`'\t'` newline

`'\r'` return

```
spaces = ['\n', '\t', '\r', ' ']
```

```
  >>> len('\n')
  1
```

`'\n'` is one character! The backslash *escapes* the character.

Removing Leading/Trailing Whitespace from a String

- Let's write a function that will remove leading and trailing whitespaces.

```
def strip(line):
   # handle empty line somehow
        # return line?
    spaces = ['\n', '\t', '\r', ' ']

    # find where the words start
    # look at each character
    # if it's a space...keep looking
        # keep track of indices looked at

    # find where the word ends
    # look at each character in reverse
    # if it's a space...keep looking
        # keep track of indices looked at

    # return the string between the start and end index
```

Removing Leading/Trailing Whitespace from a String

- Let's write a function that will remove leading and trailing whitespaces.

```python
def strip(line):
    if not line: # handle empty line
        return line
    spaces = ['\n', '\t', '\r', ' ']

    # find the first not-space
    start_index = 0
    while start_index<len(line) and line[start_index] in spaces:
        start_index += 1

    # find the last not-space
    end_index = len(line)-1
    while end_index>0 and line[end_index] in spaces:
        end_index -= 1

    return line[start_index:end_index+1]
```

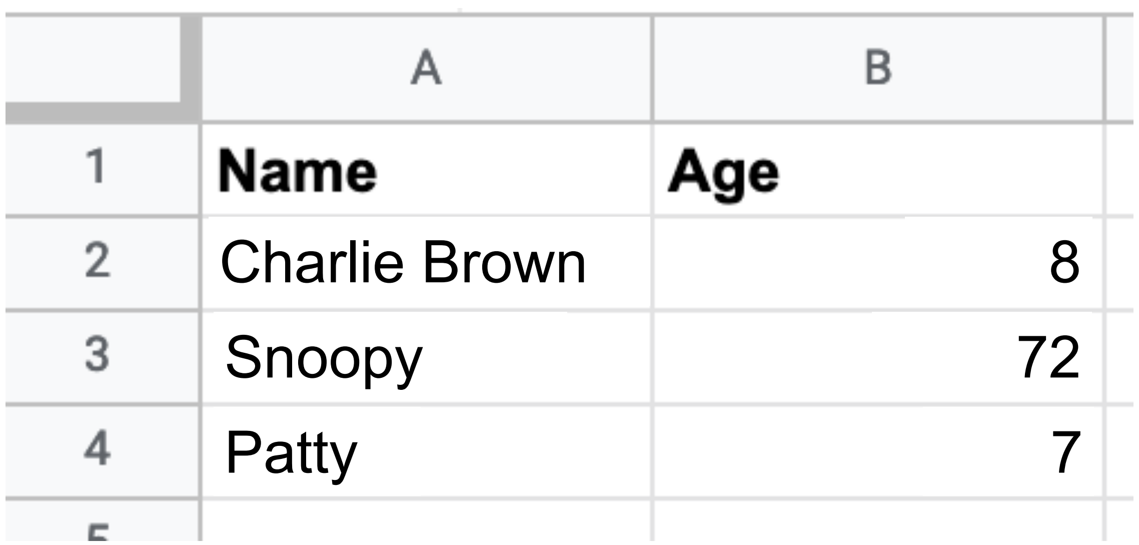
```
>>> s = '\n  \t  String with\t different\nspaces.\r\n\t'
>>> strip(s)
'String with\t different\nspaces.'
```

Useful String and List Functions in File Reading

- When reading files, we may need to use some common string and list operations to work with the data.
- We'll learn about the built-in features python has for these later in the semester, but we can write our own with iterating over strings and accumulator variables!
  - `strip(line):` Remove any leading/trailing white space or "\n"
  - `split(line, ','):` Separate a **comma-separated** sequence of words and create a list of strings
  - `join(' ', lines):` Create a single "big" string with words separated by spaces instead of commas
  - `count_appearances(ele, let):` Count the occurrence of various elements
  - …and so on!

Common File Type: CSVs

- A CSV (Comma Separated Values) file is a specific type of plain text file that stores "tabular" data
- Each row of a table is a line in the text file, with each column on the row separated by commas
- This format is a common import and export format for spreadsheets and databases



	A	B
1	**Name**	**Age**
2	Charlie Brown	8
3	Snoopy	72
4	Patty	7

CSV form:

```
Name,Age
Charlie Brown,8
Snoopy,72
Patty,7
```

Working with CSVs

- Since CSVs are just text files, we can process them in the same way
- Might require additional post-processing/splitting using string functions

```
with open("data/superheroes.csv") as roster:
    for line in roster:
        print(strip(line))
```

```
Wonderwoman,Strength,5
Superman,Strength,13
Spiderman,Spidey things,9
Black Panther,Technology,4
Captain Marvel,Strength,4
Starfire,Strength,1
Cyborg,Technology,1
Batman,Justice,23
Robin,Justice,2
Ms. Marvel,Light,0
Jean Grey,Telekenesis,7
Ironman,Technology,9
Forge,Technology,1
```

- `split(line, delimiter)` could be really useful here...

Data Analysis with CSVs

- Read the following CSV file into lists:

```
Wonderwoman,Strength,5
Superman,Strength,13
Spiderman,Spidey things,9
Black Panther,Technology,4
Captain Marvel,Strength,4
Starfire,Strength,1
Cyborg,Technology,1
Batman,Justice,23
Robin,Justice,2
Ms. Marvel,Light,0
Jean Grey,Telekenesis,7
Ironman,Technology,9
Forge,Technology,1
```

```
# accumulator variables
names = []
powers = []
movies = []

with open("data/superheroes.csv") as roster:
    for line in roster:
        line = split(trim(line), ',') # remove newline & split on commas
        names = names + [line[0]]
        powers = powers + [line[1]]
        movies = movies + [int(line[2])] # convert count to integer!
print(powers)
print(movies)
```

```
['Strength', 'Strength', 'Spidey things', 'Technology', 'Strength', 'Strength',
'Technology', 'Justice', 'Justice', 'Light', 'Telekenesis', 'Technology',
'Technology']
[5, 13, 9, 4, 4, 1, 1, 23, 2, 0, 7, 9, 1]
```

Data Analysis with CSVs

- Now let's count the appearance of different types of superpowers in our dataset. We'll need the `count_appearances(e,l)` function we've built previously.

A `set` to get us the unique list of superpowers!

```
unique_powers = list(set(powers))
count_list = []

for pwr in unique_powers:
    count_list += [count_appearances(pwr, powers)]

print(unique_powers)
print(count_list)
```

```
['Strength', 'Telekenesis', 'Light', 'Justice', 'Spidey things', 'Technology']
[4, 1, 1, 2, 1, 4]
```

- There are 4 superheroes with Strength power, 1 with Telekinesis, 1 with Light, etc. etc.

Plotting

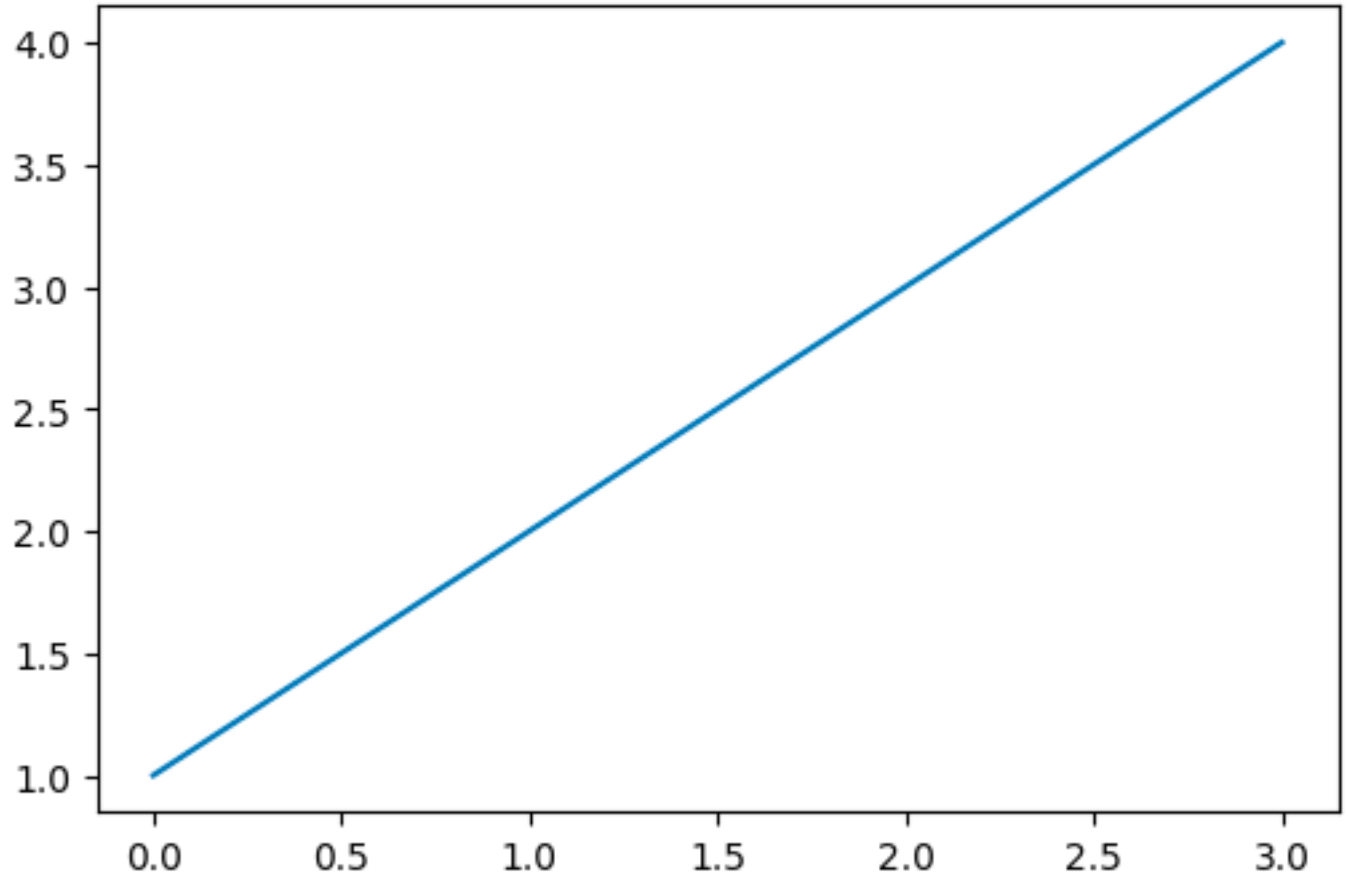
Plotting with `matplotlib`

- Suppose we want to a way to visualize our data (not just print it to the terminal)
- A plot is a graphical technique for representing a data set, usually as a graph showing the relationship between two or more variables
- We'll be using Python's **`matplotlib`** library to make plots/graphs
- The best way to learn how to plot different types of graphs is to read the documentation and see examples
- **Resources**
    - **matplotlib examples**: http://matplotlib.org/examples
    - **pyplot documentation**: http://matplotlib.org/api/pyplot_summary.html
    - **cool plots**: https://matplotlib.org/gallery.html

Plotting Basics: Plot function

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4])
plt.show()
```

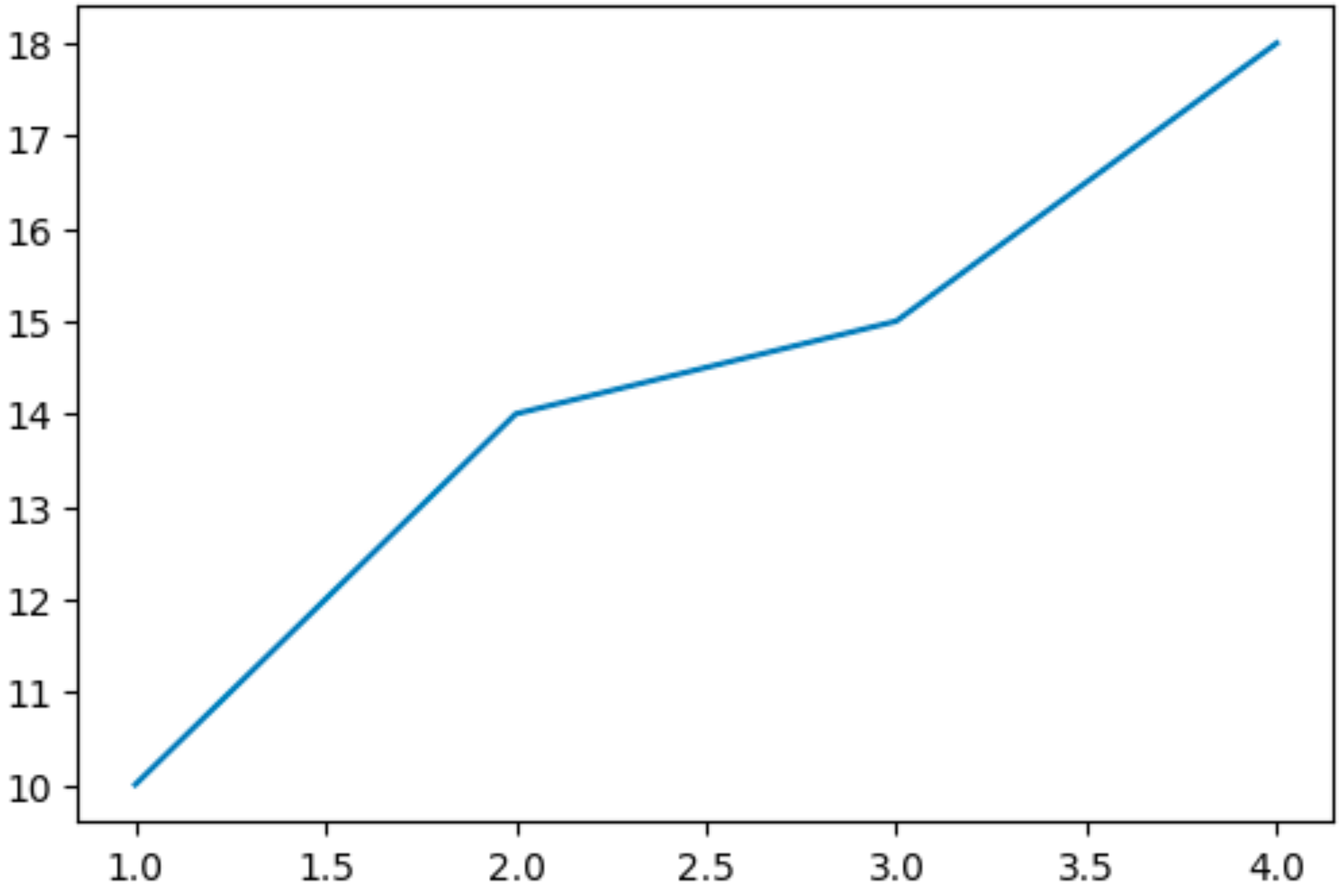
If only one list is provided, Python assumes it is as the points on the **y axis** (x values start at 0)



Plotting Basics: Plot function

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4], [10, 14, 15, 18])
plt.show()
```

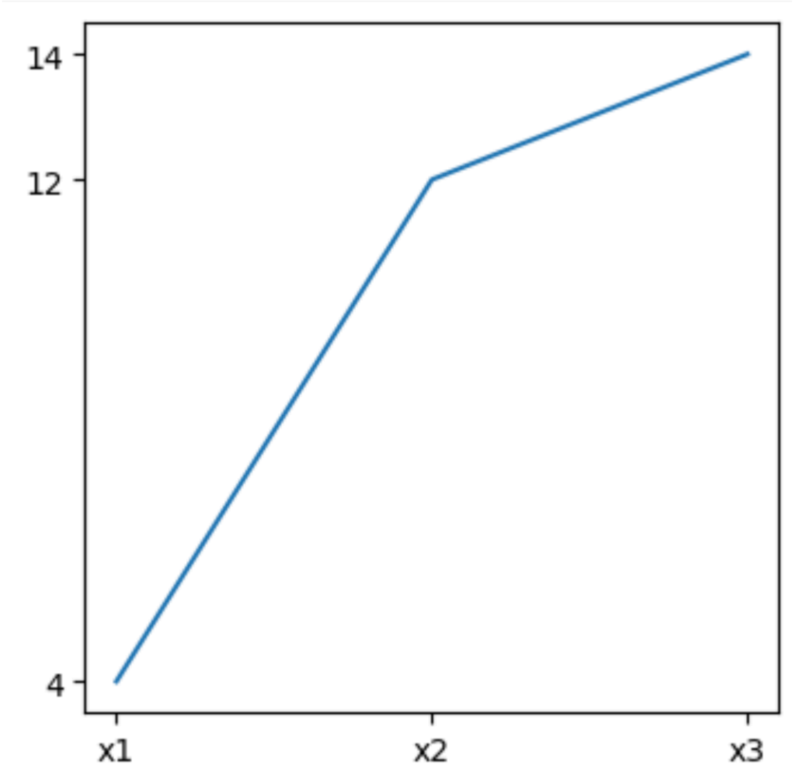
Equivalent to saying plot the points
`(1, 10), (2, 14), (3, 15), (4, 18)`



Plotting Basics: Plot function

- There are additional customizations we can add to our plot, such as the size of the plot, the location and names of the X and Y ticks.

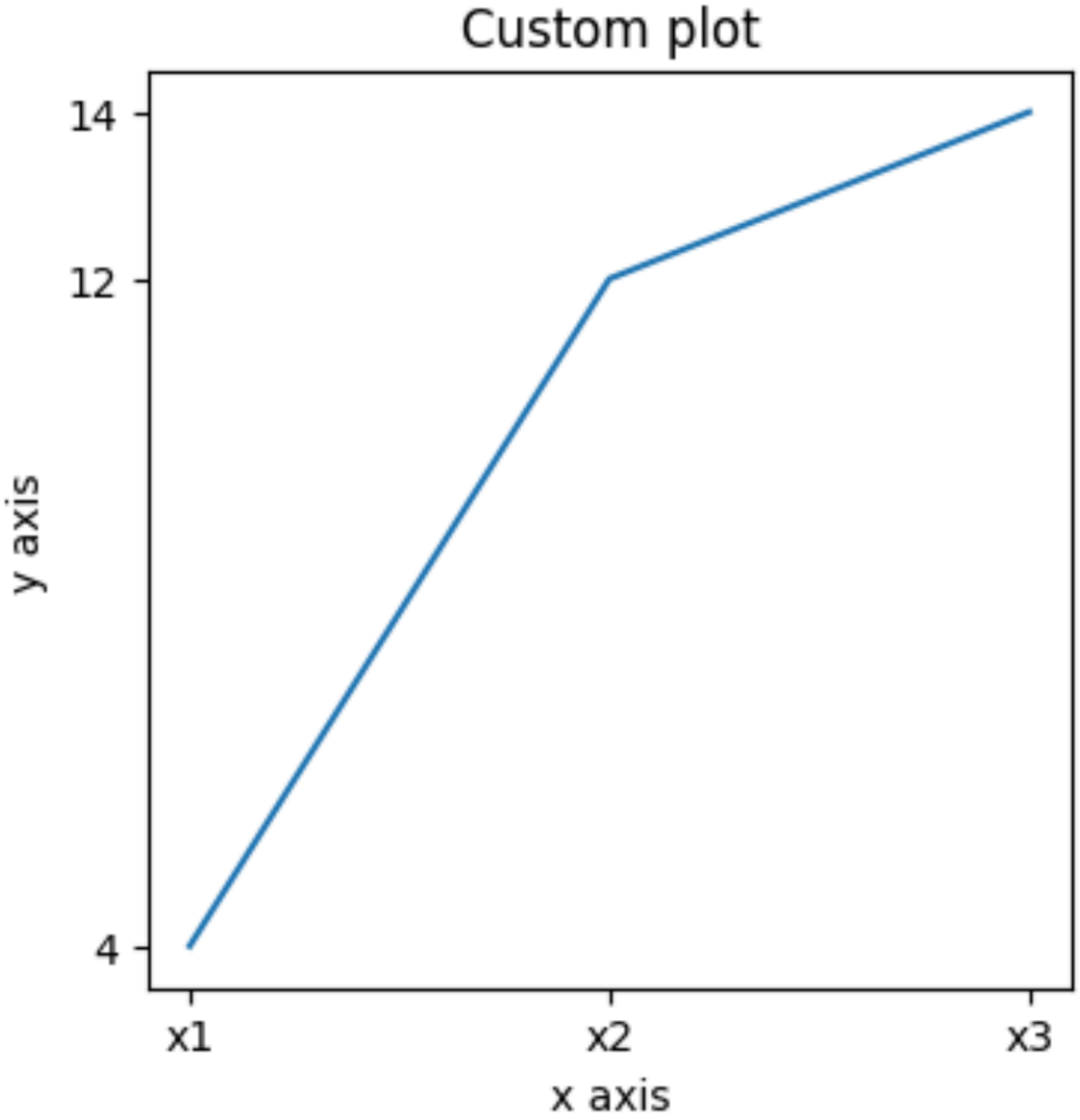
```
# a more advanced example where we customize the line plot
# create a 4 by 4 figure
plt.figure(figsize=(4, 4))
plt.plot([0, 5, 10], [4, 12, 14])
plt.xticks([0, 5, 10],                # x values of axis `ticks`
           ['x1', 'x2', 'x3'])  # values to show for `ticks`

# specify y-tick locations
plt.yticks([4, 12, 14])
```



Plotting Basics: Plot function

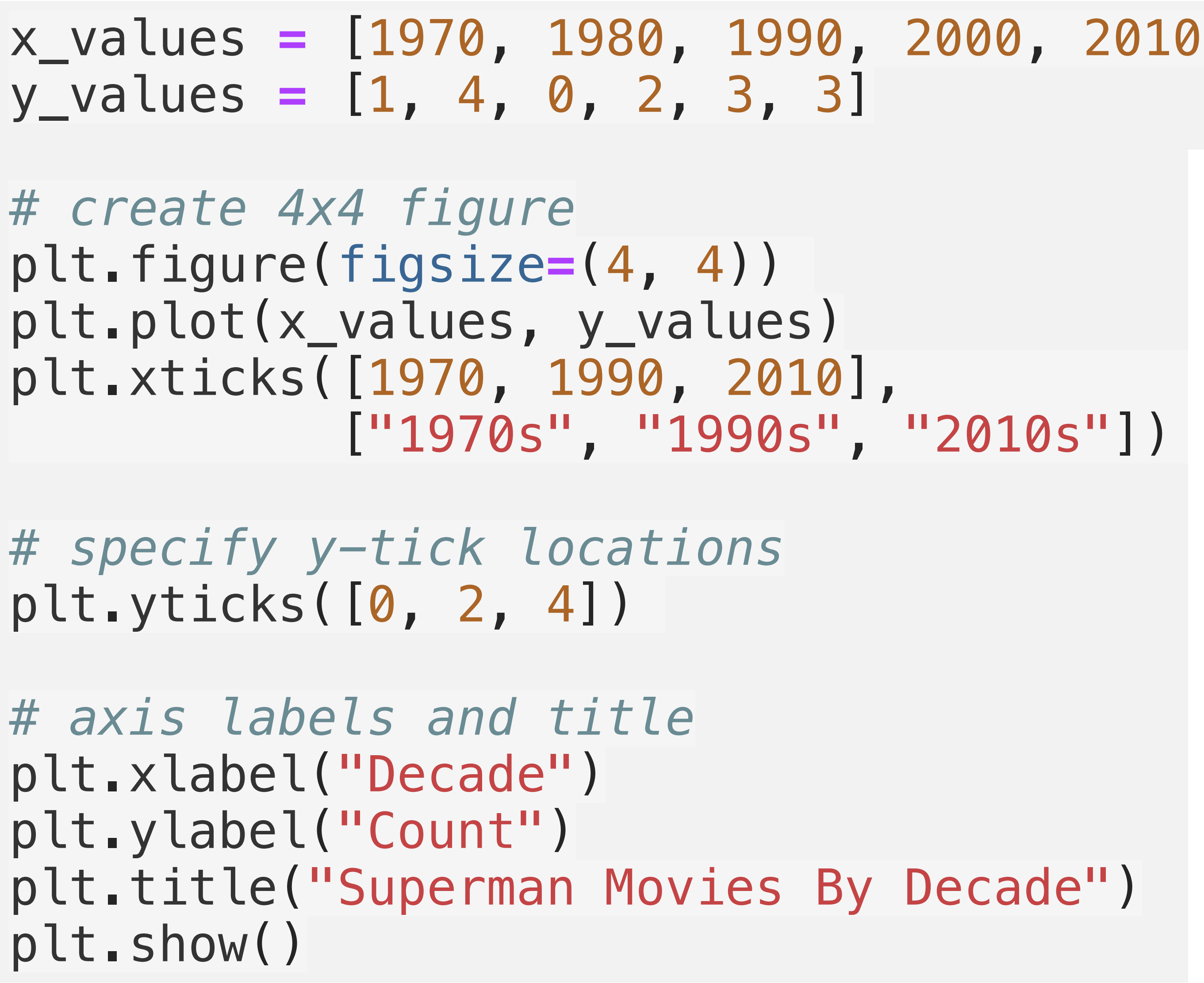
- There are additional customizations we can add to our plot, such as the X and Y Labels, as well as the title.

```
# axis labels and title
plt.xlabel("x axis")
plt.ylabel("y axis")
plt.title("Custom plot")
plt.show()
```

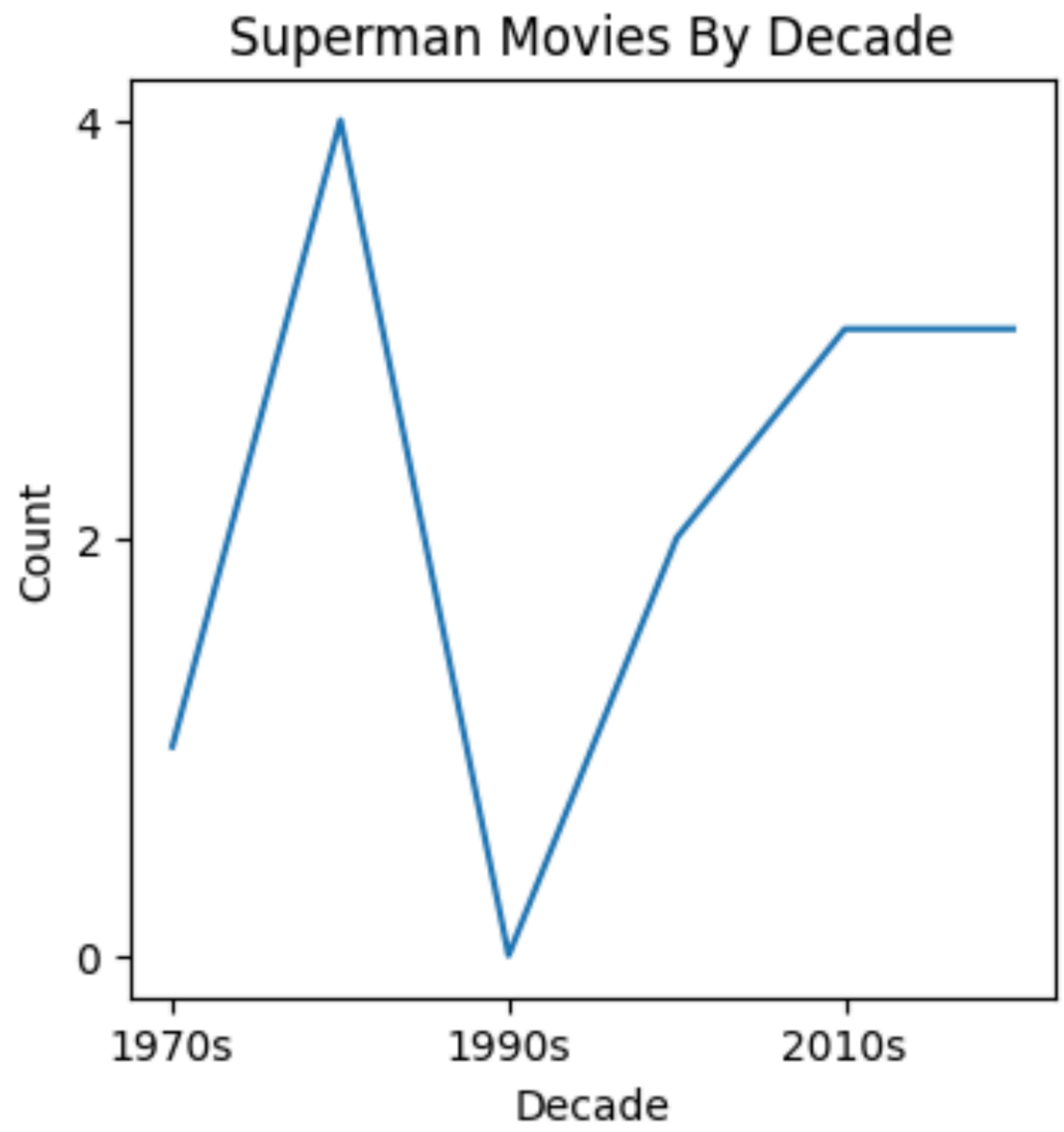


Data Analysis with CSVs

- We can use these features to make interesting plots for data analysis:

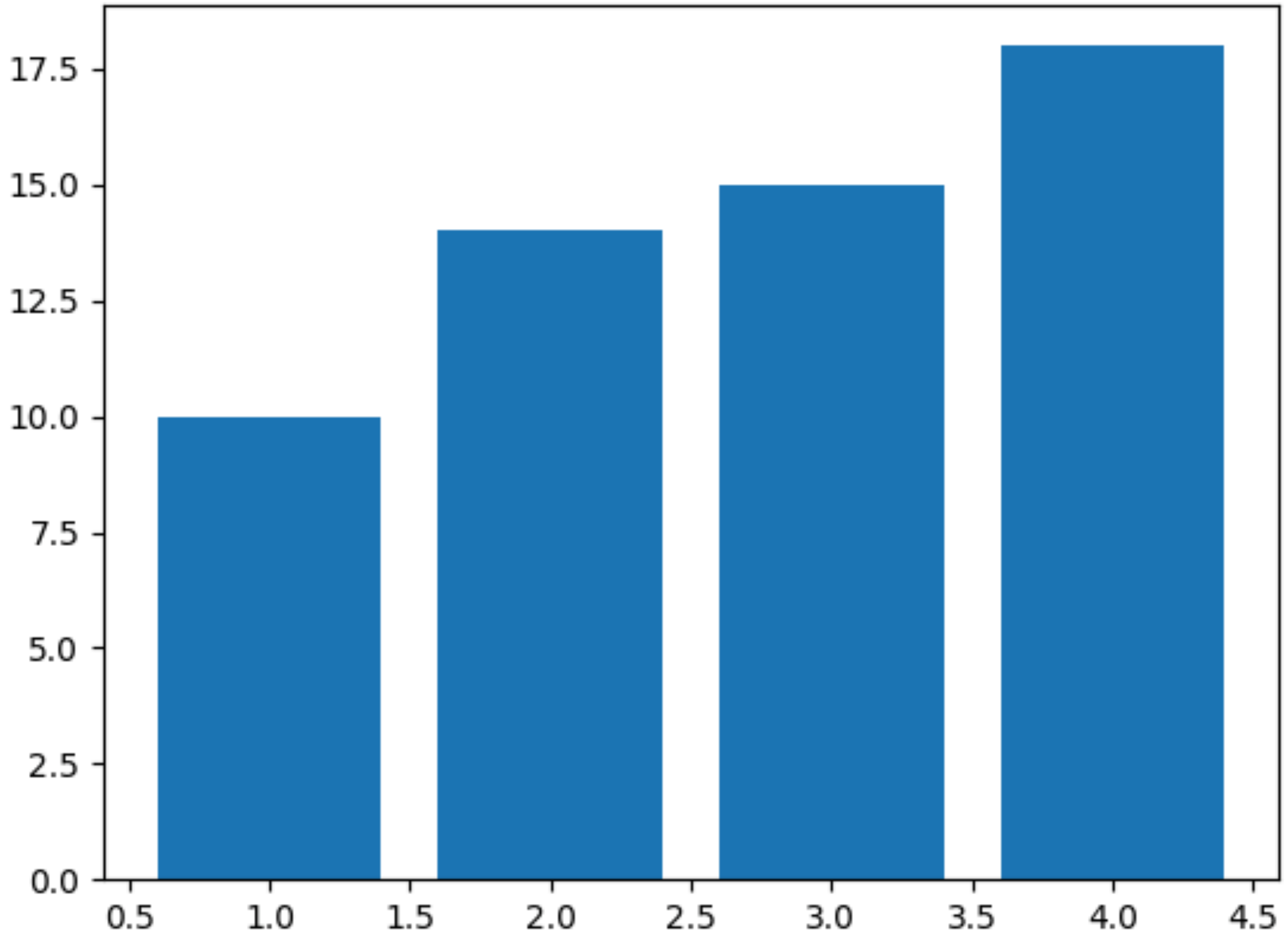


```
x_values = [1970, 1980, 1990, 2000, 2010, 2020]
y_values = [1, 4, 0, 2, 3, 3]

# create 4x4 figure
plt.figure(figsize=(4, 4))
plt.plot(x_values, y_values)
plt.xticks([1970, 1990, 2010],
           ["1970s", "1990s", "2010s"])

# specify y-tick locations
plt.yticks([0, 2, 4])

# axis labels and title
plt.xlabel("Decade")
plt.ylabel("Count")
plt.title("Superman Movies By Decade")
plt.show()
```



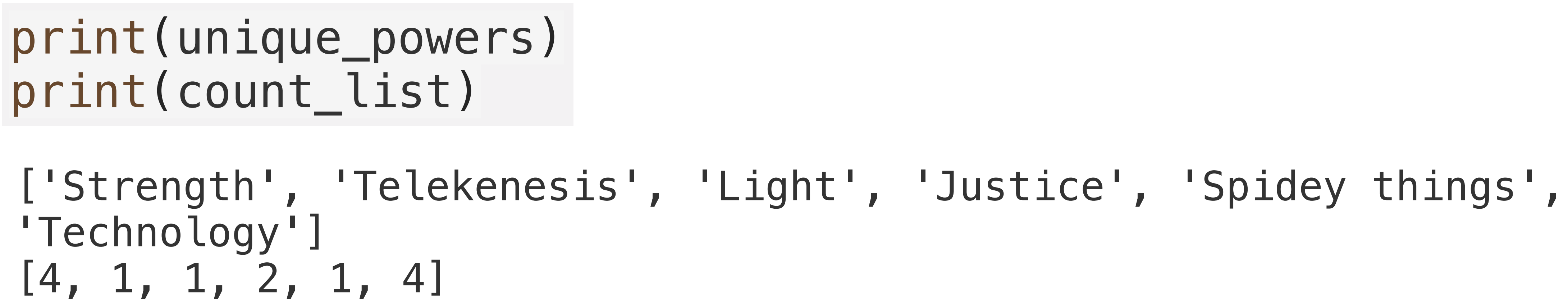
Plotting Basics: Bar function

```
import matplotlib.pyplot as plt
plt.bar([1, 2, 3, 4], [10, 14, 15, 18])
plt.show()
```



Data Analysis with CSVs

- Now let's get back to our Superhero Data Analysis. We had two lists, one of unique superpowers, and one of the counts of those superpowers:



```
print(unique_powers)
print(count_list)
```

```
['Strength', 'Telekenesis', 'Light', 'Justice', 'Spidey things',
'Technology']
[4, 1, 1, 2, 1, 4]
```

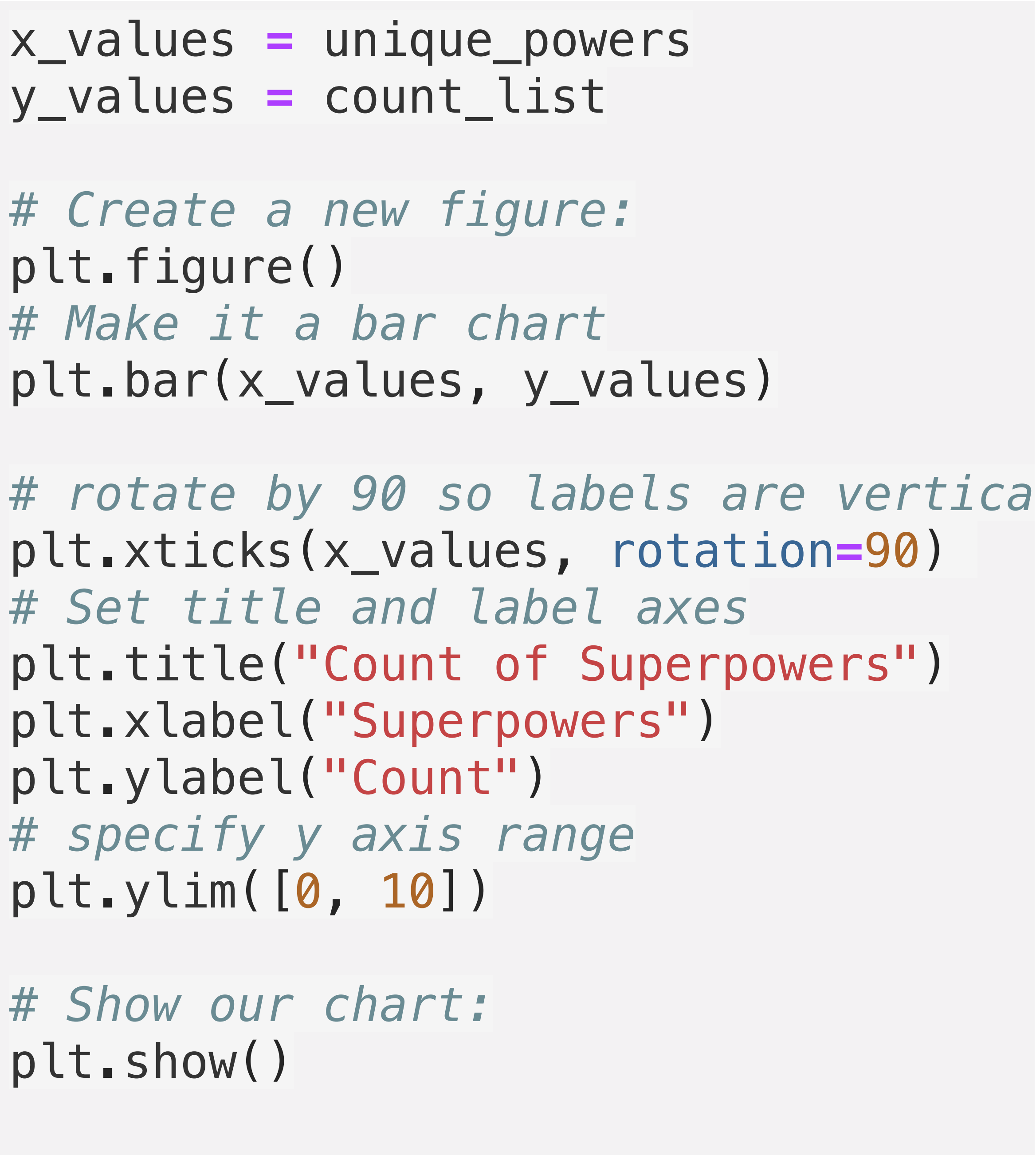
Data Analysis with CSVs

- Now let's get back to our Superhero Data Analysis. We had two lists, one of unique superpowers, and one of the counts of those superpowers:

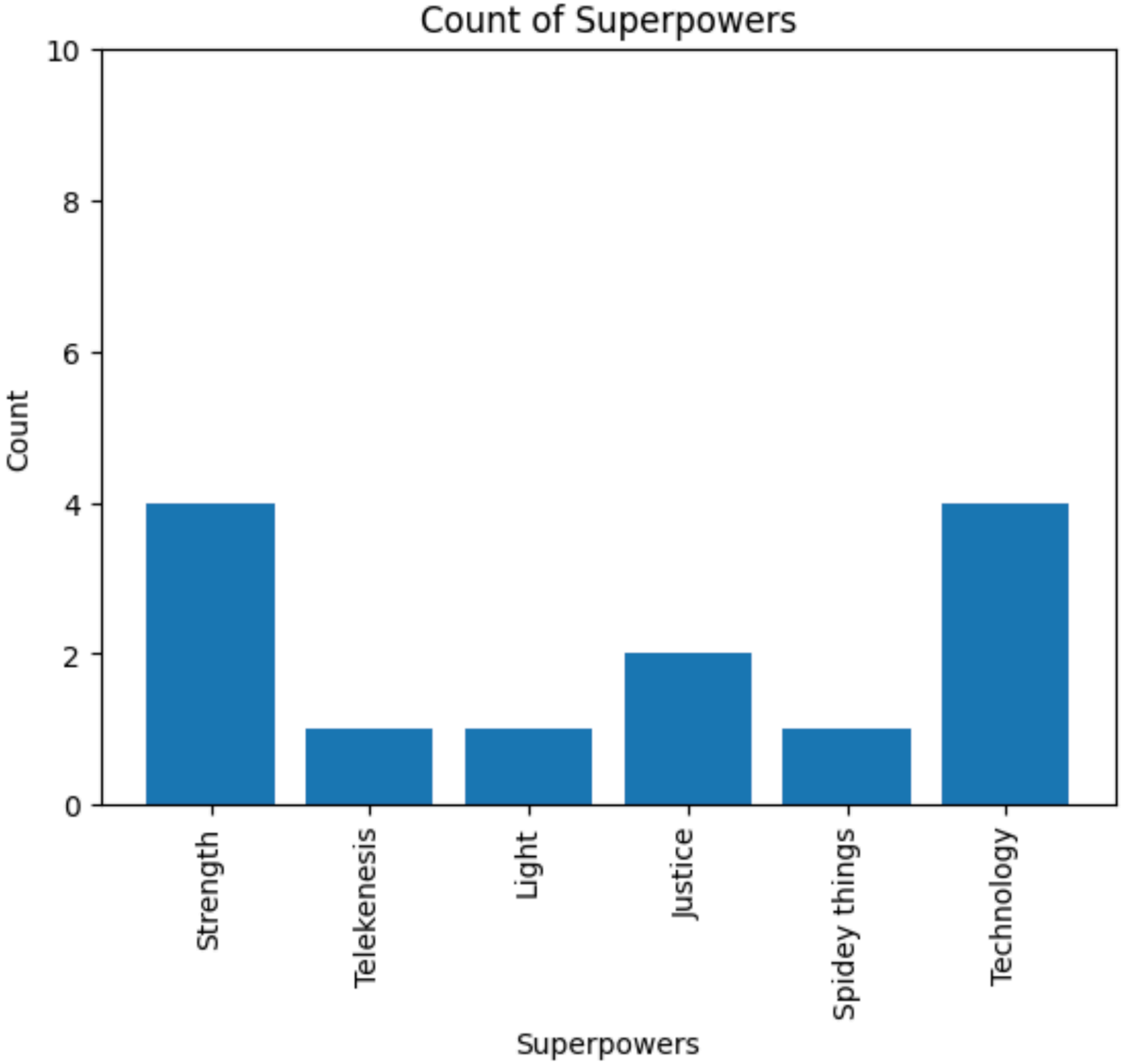
```
x_values = unique_powers
y_values = count_list

# Create a new figure:
plt.figure()
# Make it a bar chart
plt.bar(x_values, y_values)

# rotate by 90 so labels are vertical and do not overlap
plt.xticks(x_values, rotation=90)
# Set title and label axes
plt.title("Count of Superpowers")
plt.xlabel("Superpowers")
plt.ylabel("Count")
# specify y axis range
plt.ylim([0, 10])

# Show our chart:
plt.show()
```

Data Analysis with CSVs

- Now let's get back to our Superhero Data Analysis. We had two lists, one of unique superpowers, and one of the counts of those superpowers:



```
x_values = unique_powers
y_values = count_list

# Create a new figure:
plt.figure()
# Make it a bar chart
plt.bar(x_values, y_values)

# rotate by 90 so labels are vertica
plt.xticks(x_values, rotation=90)
# Set title and label axes
plt.title("Count of Superpowers")
plt.xlabel("Superpowers")
plt.ylabel("Count")
# specify y axis range
plt.ylim([0, 10])

# Show our chart:
plt.show()
```



Next Time: Data Wrangling

- New time we'll learn about a new data structure that will help us rearrange data even more, so we can produce more interesting plots!