CS106A

Juliette Woodrow

Housekeeping

- Graphics contest is due Saturday at 11:59pm Anywhere on Earth

A little bit about me...

Let’s Jump Right In

Guiding Questions

1. How can we write better loops?

2. What tools do we have for developing and analyzing data?

3. Are there any trends in UFO data?

List Comprehensions

Problem: getting a list of squares

Problem: getting a list of squares

- Imagine you have a list of numbers, and you want a list of those same numbers squared

Problem: getting a list of squares

- Imagine you have a list of numbers, and you want a list of those same numbers squared

  [4, 6, 7, 8] → [16, 36, 49, 64]

Problem: getting a list of squares

- Imagine you have a list of numbers, and you want a list of those same numbers squared

  [4, 6, 7, 8] → [16, 36, 49, 64]

- How would you produce this output list?

Problem: getting a list of squares - Attempt #1

[4, 6, 7, 8] → [16, 36, 49, 64]

Problem: getting a list of squares - Attempt #1

```
# [4, 6, 7, 8] → [16, 36, 49, 64]

def get_squared(num_lst):
    squares = []
    for num in num_lst:
        squares.append(num**2)
    return squares
```

Problem: getting a list of squares - Attempt #1

```
# [4, 6, 7, 8] → [16, 36, 49, 64]

def get_squared(num_lst):
    squares = []
    for num in num_lst:
        squares.append(num**2)
    return squares
```

```
num_lst = [4,6,7,8]

squared_lst = get_squared(num_lst)

print(squared_lst)

# would print [16, 36, 49, 64]
```

Problem: getting a list of squares - Attempt #2

Problem: getting a list of squares - Attempt #2

num_lst = [4, 6, 7, 8]

Problem: getting a list of squares - Attempt #2

```
num_lst = [4, 6, 7, 8]

squared_lst = [num ** 2 for num in num_lst]
```

Problem: getting a list of squares - Attempt #2

```
num_lst = [4, 6, 7, 8]

squared_lst = [num ** 2 for num in num_lst]
```

this is a list comprehension!

List Comprehensions

[num ** 2 for num in num_lst]

List Comprehensions

[num ** 2 for num in num_lst]

- **Definition:** one way to make a new list based on the values of an existing list

List Comprehensions

[num ** 2 for num in num_lst]

- **Definition:** one way to make a new list based on the values of an existing list

- **Three Key Parts:**

List Comprehensions

[num ** 2 for num in num_lst]

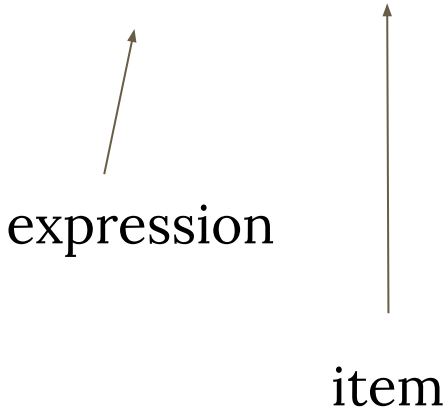
expression

- **Definition:** one way to make a new list based on the values of an existing list

- **Three Key Parts:**
  - Expression

List Comprehensions

```
[num ** 2 for num in num_lst]
```

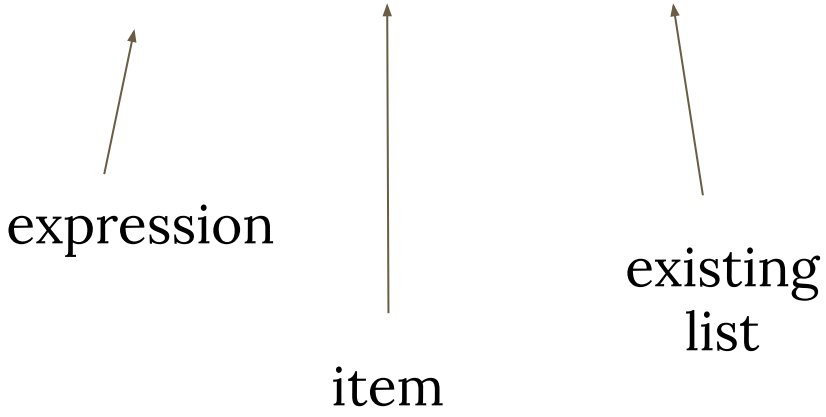


- **Definition:** one way to make a new list based on the values of an existing list

- **Three Key Parts:**
  - Expression
  - Item

List Comprehensions

[num ** 2 for num in num_lst]



- **Definition:** one way to make a new list based on the values of an existing list

- **Three Key Parts:**
  - Expression
  - Item From Existing List
  - Existing List

Hey, we already know some of that syntax!

[num ** 2 for num in num_lst]

Hey, we already know some of that syntax!

```
[num ** 2 for num in num_lst]
```

- [ ] -- that makes it a list

Hey, we already know some of that syntax!

```
[num ** 2 for num in num_lst]
```

- [ ] -- that makes it a list
- for num in num_list -- that’s just a for each loop

Hey, we already know some of that syntax!

```
[num ** 2 for num in num_lst]
```

- [ ] -- that makes it a list
- for num in num_list -- that's just a for each loop
- This is how we square a number

Hey, we already know some of that syntax!

[num ** 2 for num in num_lst]

- **[ ] -- that makes it a list**
- **for num in num_list -- that's just a for each loop**
- **This is how we square a number**

Let’s try it out!

Let's try it out!

- You have a list of strings with random casing and you want a list of strings that are all lowercase

Let’s try it out!

- You have a list of strings with random casing and you want a list of strings that are all lowercase

  [“Hi”, “mOm”, “aNd”, “DAD”] → [“hi”, “mom”, “and”, “dad”]

Let's try it out!

- You have a list of strings with random casing and you want a list of strings that are all lowercase

  ["Hi", "mOm", "aNd", "DAD"] → ["hi", "mom", "and", "dad"]

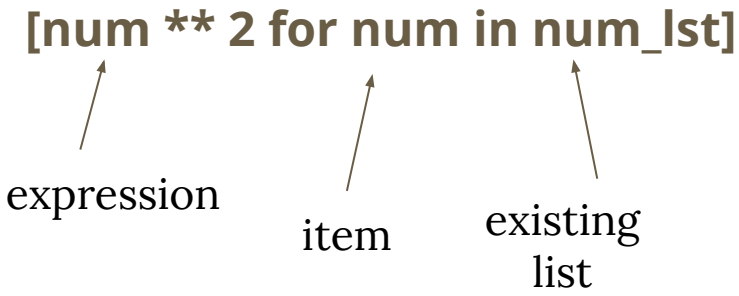
- How can we use a list comprehension to do this?

Let's try it out!

- You have a list of strings with random casing and you want a list of strings that are all lowercase

  ["Hi", "mOm", "aNd", "DAD"] → ["hi", "mom", "and", "dad"]

- How can we use a list comprehension to do this?



- **Definition:** one way to make a new list based on the values of an existing list

Problem: getting a list of lowercase strings

random_case = [“Hi”, “mOm”, “aNd”, “DAD”]

Problem: getting a list of lowercase strings

random_case = [“Hi”, “mOm”, “aNd”, “DAD”]

all_lower = [s.lower() for s in random_case]

print(all_lower)

would print [“hi”, “mom”, “and”, “dad”]

Problem: getting a list of tuples

Problem: getting a list of tuples

- Say you have a list of strings. Create a list of tuples where the first elem is the string and the second elem is the length of the string

Problem: getting a list of tuples

- Say you have a list of strings. Create a list of tuples where the first elem is the string and the second elem is the length of the string

  [“I”, “love”, “CS106A”] → [(“I”, 1), (“love”, 4), (“CS106A”, 6)]

Problem: getting a list of tuples

- Say you have a list of strings. Create a list of tuples where the first elem is the string and the second elem is the length of the string

  [“I”, “love”, “CS106A”] → [(“I”, 1), (“love”, 4), (“CS106A”, 6)]

- How can you solve this with a list comprehension?

Problem: getting a list of tuples

my_thoughts = [“I”, “love”, “CS106A”]

Problem: getting a list of tuples

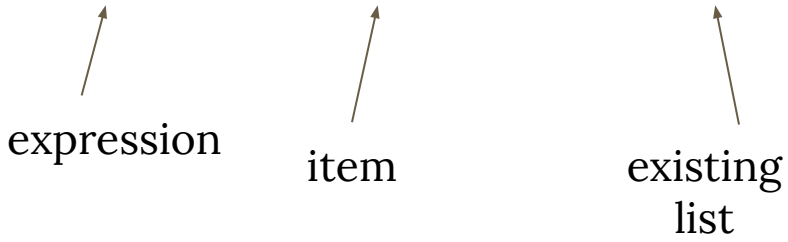
my_thoughts = [“I”, “love”, “CS106A”]

tuple_thoughts = [(s, len(s)) for s in my_thoughts]

Problem: getting a list of tuples

my_thoughts = [“I”, “love”, “CS106A”]

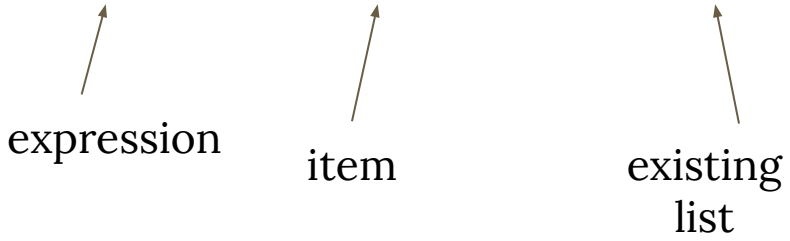
tuple_thoughts = [(s, len(s)) for s in my_thoughts]



Problem: getting a list of tuples

my_thoughts = [“I”, “love”, “CS106A”]

tuple_thoughts = [(s, len(s)) for s in my_thoughts]



print(tuple_thoughts)

#would print [(“I”, 1), (“love”, 4), (“CS106A”, 6)]

Problem: converting temperature to celsius

Problem: converting temperature to celsius

- List of temperatures in degrees celsius
  france_temps_c = [13, 14, 15, 16, 8, 9, 12]

Problem: converting temperature to celsius

- List of temperatures in degrees celsius
  france_temps_c = [13, 14, 15, 16, 8, 9, 12]

- Want a list of temperatures in degrees fahrenheit

Problem: converting temperature to celsius

- List of temperatures in degrees celsius
  france_temps_c = [13, 14, 15, 16, 8, 9, 12]

- Want a list of temperatures in degrees fahrenheit

- Can you think of a way to write a list comprehension to solve this?

Problem: converting temperature to celsius

- List of temperatures in degrees celsius
  france_temps_c = [13, 14, 15, 16, 8, 9, 12]

- Want a list of temperatures in degrees fahrenheit

- Can you think of a way to write a list comprehension to solve this?

Problem: converting temperature to celsius

Problem: converting temperature to celsius

france_temps_c = [13, 14, 15, 16, 8, 9, 12]

℃(9/5) + 32 = ℉

Problem: converting temperature to celsius

france_temps_c = [13, 14, 15, 16, 8, 9, 12]

℃(9/5) + 32 = ℉

france_temps_f = [ ]

Problem: converting temperature to celsius

france_temps_c = [13, 14, 15, 16, 8, 9, 12]

℃(9/5) + 32 = ℉

france_temps_f = [for t in france_temps_c]

Problem: converting temperature to celsius

france_temps_c = [13, 14, 15, 16, 8, 9, 12]

℃(9/5) + 32 = ℉

france_temps_f = [t*(9/5) + 32 for t in france_temps_c]

Problem: converting temperature to celsius

france_temps_c = [13, 14, 15, 16, 8, 9, 12]

℃(9/5) + 32 = ℉

france_temps_f = [t*(9/5) + 32 for t in france_temps_c]

print(france_temps_f)

would print [55.4, 57.2, 59.0, 46.4, 48.2, 53.6, 46.4]

Problem: converting temperature to celsius

france_temps_c = [13, 14, 15, 16, 8, 9, 12]

℃(9/5) + 32 = ℉

france_temps_f = [t*(9/5) + 32 for t in france_temps_c]

print(france_temps_f)

would print [55.4, 57.2, 59.0, 46.4, 48.2, 53.6, 46.4]

Problem: converting temperature to celsius

france_temps_c = [13, 14, 15, 16, 8, 9, 12]

℃(9/5) + 32 = ℉

france_temps_f = [t*(9/5) + 32 for t in france_temps_c]

print(france_temps_f)

would print [55.4, 57.2, 59.0, 46.4, 48.2, 53.6, 46.4]

Problem: converting temperature to celsius

france_temps_c = [13, 14, 15, 16, 8, 9, 12]

℃(9/5) + 32 = ℉

france_temps_f = [**t*(9/5) + 32** for t in france_temps_c]

Problem: converting temperature to celsius

france_temps_c = [13, 14, 15, 16, 8, 9, 12]

℃(9/5) + 32 = ℉

france_temps_f = [**t*(9/5) + 32** for t in france_temps_c]

- Can we decompose this?

Problem: converting temperature to celsius

france_temps_c = [13, 14, 15, 16, 8, 9, 12]

℃(9/5) + 32 = ℉

france_temps_f = [**t*(9/5) + 32** for t in france_temps_c]

- Can we decompose this? Yes !!

Problem: converting temperature to celsius

france_temps_c = [13, 14, 15, 16, 8, 9, 12]

℃(9/5) + 32 = ℉

france_temps_f = [**t*(9/5) + 32** for t in france_temps_c]

- Can we decompose this? Yes !!

```
def make_fahrenheit(c):
    return c * (9/5) + 32
```

Problem: converting temperature to celsius

france_temps_c = [13, 14, 15, 16, 8, 9, 12]

℃(9/5) + 32 = ℉

france_temps_f = [**make_fahrenheit(t)** for t in france_temps_c]

- Can we decompose this? Yes !!

```
def make_fahrenheit(c):
    return c * (9/5) + 32
```

Conditions in List Comprehensions

Conditions in List Comprehensions

nums = [12, 13, 14, 15, 15, 16, 18, 22, 22]

Conditions in List Comprehensions

nums = [12, 13, 14, 15, 15, 16, 18, 22, 22]

- What if we want only the even numbers?

Conditions in List Comprehensions

nums = [12, 13, 14, 15, 15, 16, 18, 22, 22]

- What if we want only the even numbers?

even_nums → [12, 14, 16, 18, 22, 22]

Conditions in List Comprehensions

nums = [12, 13, 14, 15, 15, 16, 18, 22, 22]

- What if we want only the even numbers?

even_nums → [12, 14, 16, 18, 22, 22]

even_nums = [n for n in nums if n % 2 == 0]

Conditions in List Comprehensions

- You can add a condition to a list comprehension for additional “filtering”

Conditions in List Comprehensions

- You can add a condition to a list comprehension for additional “filtering”

[expression for item in list if condition]

Conditions in List Comprehensions

- You can add a condition to a list comprehension for additional “filtering”

[expression for item in list if condition]

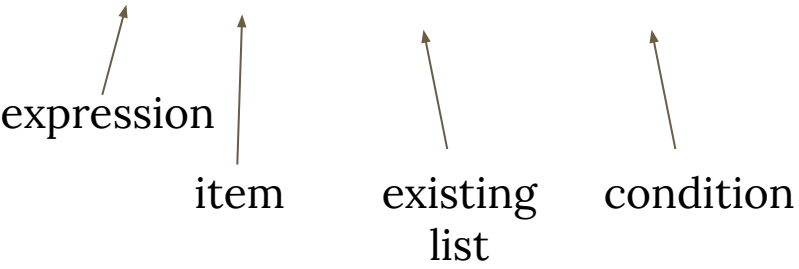
[n for n in nums if n % 2 == 0]

Conditions in List Comprehensions

- You can add a condition to a list comprehension for additional “filtering”

[expression for item in list if condition]

[n for n in nums if n % 2 == 0]



Let’s try it out

Let’s try it out

kids = [“jonathan”, “isabelle”, “henry”, “juliette”, “audrey”, “bailey”]

Let’s try it out

kids = [“jonathan”, “isabelle”, “henry”, “juliette”, “audrey”, “bailey”]

- Want only the names that end in “y”

Let’s try it out

kids = [“jonathan”, “isabelle”, “henry”, “juliette”, “audrey”, “bailey”]

- Want only the names that end in “y”

y_at_end_kids = [name for name in kids if name[-1] == ‘y’]

Let’s try it out

kids = [“jonathan”, “isabelle”, “henry”, “juliette”, “audrey”, “bailey”]

- Want only the names that end in “y”

y_at_end_kids = [name for name in kids if name[-1] == ‘y’]

y_at_end_kids is [“henry”, “audrey”, “bailey”]

Let's try it out

kids = [“jonathan”, “isabelle”, “henry”, “juliette”, “audrey”, “bailey”]

- Want only the names that end in “y”

```
y_at_end_kids = [name for name in kids if name[-1] == ‘y’]

# y_at_end_kids is [“henry”, “audrey”, “bailey”]
```

Note: a list comprehension makes a *new* list and does not modify the original one

Let's try it out

kids = [“jonathan”, “isabelle”, “henry”, “juliette”, “audrey”, “bailey”]

- Want only the names that end in “y”

y_at_end_kids = [name for name in kids if name[-1] == ‘y’]

y_at_end_kids is [“henry”, “audrey”, “bailey”]

- Want only the names that start with “b”

Note: a list comprehension makes a *new* list and does not modify the original one

Let's try it out

kids = ["jonathan", "isabelle", "henry", "juliette", "audrey", "bailey"]

- Want only the names that end in "y"

```
y_at_end_kids = [name for name in kids if name[-1] == 'y']

# y_at_end_kids is ["henry", "audrey", "bailey"]
```

- Want only the names that start with "b"

```
b_at_front_kids = [name for name in kids if name[0] == 'b']
```

Note: a list comprehension makes a *new* list and does not modify the original one

Let's try it out

kids = [“jonathan”, “isabelle”, “henry”, “juliette”, “audrey”, “bailey”]

- Want only the names that end in “y”

y_at_end_kids = [name for name in kids if name[-1] == ‘y’]

y_at_end_kids is [“henry”, “audrey”, “bailey”]

- Want only the names that start with “b”

b_at_front_kids = [name for name in kids if name[0] == ‘b’]

#b_at_front_kids is

Note: a list comprehension makes a *new* list and does not modify the original one

Let's try it out

kids = [“jonath liette”, “audrey”, “bailey”]

- Want only ”

y_at_end_kids if name[-1] == ‘y’]

y_at_end_ki ailey”]

- Want only n “b”

b_at_front_kid ls if name[0] == ‘b’]

#b_at_front_k

Note: a list comprehension makes a *new* list and does not modify the original one

Let's try it out

kids = [“jonathan”, “isabelle”, “henry”, “juliette”, “audrey”, “bailey”]

- Want only the names that end in “y”

y_at_end_kids = [name for name in kids if name[-1] == ‘y’]

y_at_end_kids is [“henry”, “audrey”, “bailey”]

- Want only the names that start with “b”

b_at_front_kids = [name for name in kids if name[0] == ‘b’]

#b_at_front_kids is [“bailey”]

Note: a list comprehension makes a *new* list and does not modify the original one

Why List Comprehensions?

Why List Comprehensions?

- They make me feel cool 😎

Why List Comprehensions?

- They make me feel cool 😎
- They are more concise

Why List Comprehensions?

- They make me feel cool 😎
- They are more concise
- They are faster than using a for loop

Why List Comprehensions?

- They make me feel cool 😎
- They are more concise
- They are faster than using a for loop
- They are *Pythonic*

Why List Comprehensions?

- They make me feel cool 😎
- They are more concise
- They are faster than using a for loop
- They are *Pythonic*

What does it mean to by *Pythonic*?

When to not use list comprehensions?

When to not use list comprehensions?

- When you need more than one condition

When to not use list comprehensions?

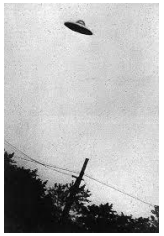
- When you need more than one condition
- When the expression is complex and you can’t easily decompose into a helper function

What tools do we have to develop and analyze data?

Let’s Analyze Some Data

Dataset Source

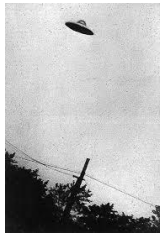
Let's Analyze Some Data



- Found this cool dataset with a lot of UFO sightings over the last century

Dataset Source

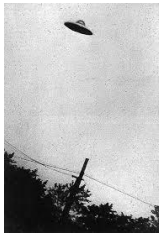
Let's Analyze Some Data



- Found this cool dataset with a lot of UFO sightings over the last century
- date/time, city, state, country, shape, duration, etc.

Dataset Source

Let's Analyze Some Data



- Found this cool dataset with a lot of UFO sightings over the last century
- date/time, city, state, country, shape, duration, etc.
- Made me wonder:
  - Where are the most UFO sightings?
  - How long do the sightings last for?
  - What other trends can we find from this data?

Dataset Source

Let’s check out the code

If only there was something else...

Jupyter Notebook

Jupyter Notebook

- Interactive “notebook” where you can run parts of your code at a time

Jupyter Notebook

- Interactive “notebook” where you can run parts of your code at a time
  - Can develop code step by step
  - Great for data analysis

Jupyter Notebook

- Interactive “notebook” where you can run parts of your code at a time
  - Can develop code step by step
  - Great for data analysis

- Built on top of regular python

Jupyter Notebook

- Interactive “notebook” where you can run parts of your code at a time
  - Can develop code step by step
  - Great for data analysis

- Built on top of regular python

- Kind of like a playground for you to work in
- Supplemental to Pycharm

Jupyter Notebook

- Interactive “notebook” where you can run parts of your code at a time
  - Can develop code step by step
  - Great for data analysis

- Built on top of regular python

- Kind of like a playground for you to work in
- Supplemental to Pycharm

- Good for collaboration !
  - In your CS life outside of this class, you will likely collaborate on the code that you write

Jupyter Notebook

- Jupyter Notebook Set Up

$ python3 -m pip install jupyter

Quick Review of Sorting

Sorting

num_lst = [3, 4, 5, 8, 1, 12.3, 0, -3]

Sorting

num_lst = [3, 4, 5, 8, 1, 12.3, 0, -3]

sorted_lst = sorted(num_lst)

Sorting

```
num_lst = [3, 4, 5, 8, 1, 12.3, 0, -3]

sorted_lst = sorted(num_lst)

# sorted_lst → [-3, 0, 1, 3, 4, 5, 8, 12.3]
```

Sorting

```
num_lst = [3, 4, 5, 8, 1, 12.3, 0, -3]

sorted_lst = sorted(num_lst)

# sorted_lst → [-3, 0, 1, 3, 4, 5, 8, 12.3]

rev_sorted_lst = sorted(num_lst, reverse=True)
```

Sorting

```
num_lst = [3, 4, 5, 8, 1, 12.3, 0, -3]

sorted_lst = sorted(num_lst)

# sorted_lst → [-3, 0, 1, 3, 4, 5, 8, 12.3]

rev_sorted_lst = sorted(num_lst, reverse=True)

# rev_sorted_lst → [12.3, 8, 5, 4, 3, 1, 0, -3]
```

Super Advanced Sorting

Super Advanced Sorting

tuple_thoughts = [(“I”, 1), (“love”, 4), (“CS106A”, 6)] # (s, len(s))

Super Advanced Sorting

tuple_thoughts = [(“I”, 1), (“love”, 4), (“CS106A”, 6)] # (s, len(s))

- How can we sort tuple thoughts by the length of the first element from largest to smallest?

Super Advanced Sorting

tuple_thoughts = [(“I”, 1), (“love”, 4), (“CS106A”, 6)] # (s, len(s))

- How can we sort tuple thoughts by the length of the first element from largest to smallest?

```
def second_elem(x):
    return x[1]  # returns second element of a tuple x
```

Super Advanced Sorting

tuple_thoughts = [(“I”, 1), (“love”, 4), (“CS106A”, 6)] # (s, len(s))

- How can we sort tuple thoughts by the length of the first element from largest to smallest?

```
def second_elem(x):
    return x[1]  # returns second element of a tuple x
```

sorted_tuple_thoughts = sorted(tuple_thoughts, **key=second_elem**, reverse=True)

Super Advanced Sorting

tuple_thoughts = [(“I”, 1), (“love”, 4), (“CS106A”, 6)] # (s, len(s))

- How can we sort tuple thoughts by the length of the first element from largest to smallest?

```
def second_elem(x):
    return x[1]  # returns second element of a tuple x
```

sorted_tuple_thoughts = sorted(tuple_thoughts, **key=second_elem**, reverse=True)

sorted_tuple_thoughts → [('CS106A', 6), ('love', 4), ('I', 1)]

Back to the code

How Can We Visualize Data?


Bias Bars
Word To Plot: class
11329
9711
8092
6474
4855
3237
1618
0
W
M
W
M
W
M
Low Reviews
Medium Reviews
High Reviews
Search:

You are all already experts on this!
But...
there is another way

Using Matplotlib

Using Matplotlib

- A library to create plots
  - Other people felt your pain and they created a library to help us all make graphs in python

Using Matplotlib

- A library to create plots
  - Other people felt your pain and they created a library to help us all make graphs in python

- To install
  - $ python3 -m pip install matplotlib

Using Matplotlib

Using Matplotlib

```
import matplotlib.pyplot as plt
```

Using Matplotlib

```
import matplotlib.pyplot as plt

# x = list of x vals;     y = list of y vals

plt.plot(x, y)   # line plot
```

Using Matplotlib

```
import matplotlib.pyplot as plt

# x = list of x vals;     y = list of y vals

plt.plot(x, y)   # line plot

plt.scatter(x, y)   # scatter plot
```

Using Matplotlib

```
import matplotlib.pyplot as plt

# x = list of x vals;     y = list of y vals

plt.plot(x, y)   # line plot

plt.scatter(x, y)   # scatter plot

plt.title(text)   #adds a title to the plot

plt.show()    #displays the plot
```

Using Matplotlib

- There are many more features !!
  - Read more about Matplotlib here
  - This is a useful Matplotlib tutorial tutorial

Let’s try it out