Modules & Visualization

CSE 160 Summer 2026

Questions?

Sli.do #cse160

Announcements

- **Midterm Scores** available on Gradescope
  - Accepting Regrade Requests until Friday, July 31 at 11:59pm
- **Programming Practice 5** **due Sunday, August 2 at 11:59 PM**
- **Homework Assignment 4** **due Monday, August 3 at 11:59 PM**
- **Resubmission Cycle 2** now available, **due Wednesday July 29th at 11:59pm**
  - Homework 1 and Homework 2 are eligible
  - Fill out the Google Form!

Things You Might Come Across

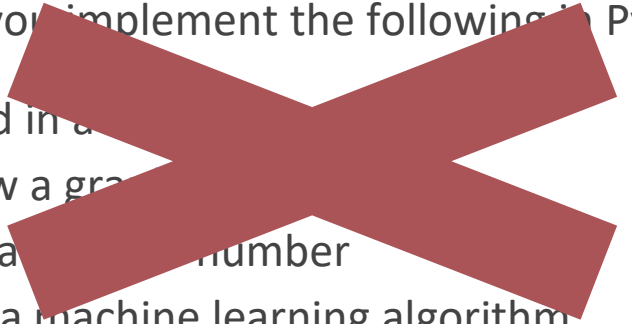
How would you implement the following in Python?

- Read in a CSV
- Draw a graph
- Get a random number
- Use a machine learning algorithm

Things You Might Come Across

How do you implement the following in Python?

- Read in a
- Draw a gra
- Get a number
- Use a machine learning algorithm



Don't worry about it!

Someone's already written in and we can reuse it!

Introducing Modules!

What are the benefits of using what other people have created?

Sli.do #cse160

Modules

- Allow us to reuse code other people have written
- Don’t need to reimplement common functions
- Simplify your code
- Make debugging easier
- Make reading code easier

Using a Module

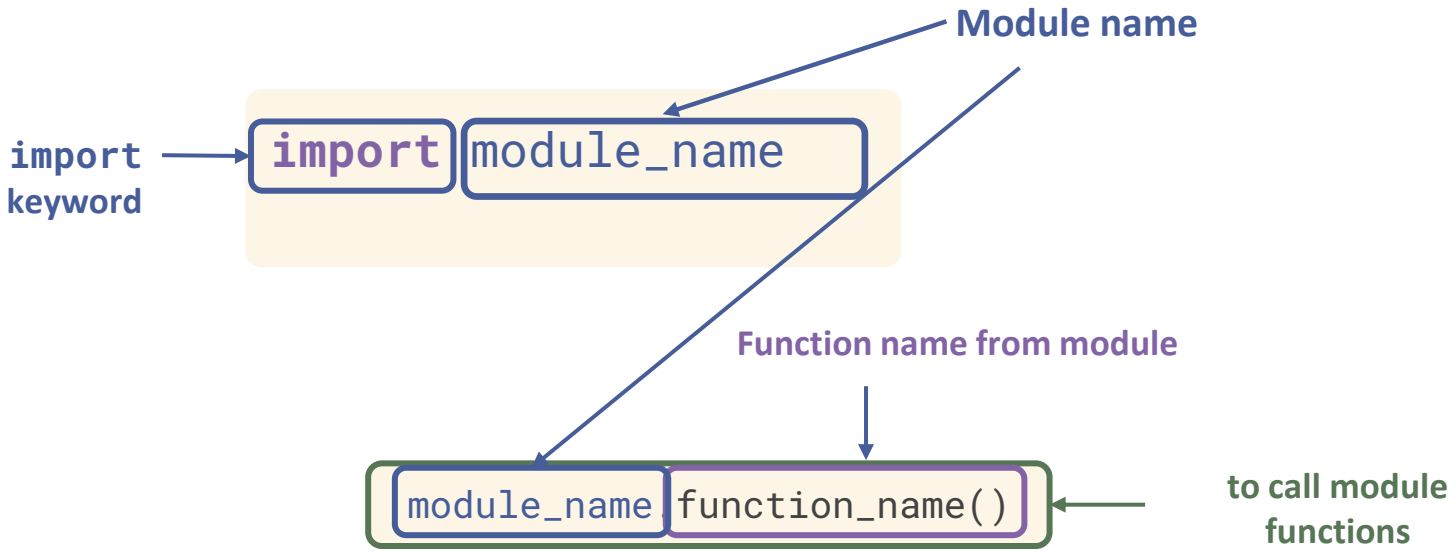
Import

```
# Tell Python to use the math module
import math
```

Calling a Module

```
# Calculates the square root of 9
print(math.sqrt(9))
```

Anatomy of Calling a Function from a Module



Example of a Module

my_sum.py

```
# Return the sum of numbers in a list
def list_sum(my_list):
        sum = 0
        for num in my_list:
                sum += num
        return sum
```

Using my_sum

```
import my_sum
list_1 = [1,2,3,4,5]
print(my_sum.list_sum(list_1))
```

Module Example 1: Random

```
import random
# How can we print a
random number between 1
and 10?
```

A. `print(random(1,10))`

B. `print(random.randint(1,10))`

C. `print(randint(1,10))`

Popular Packages

- A Python package is a collection of available code you can import and use in your program
- Packages are a collection of module(s)

Using a Module - From HW3

In a separate utils.py file

Import

```
# Tell Python what to import and from where
from utils import (read_image, write_image)
```

Function

```
# Reads the image file at file_path
def read_image(file_path):
    ...
    return result
```

Calling a Module

```
# Calls the read_image function from utils
input_grid = read_image(input_file)
```

Module Example 2:

```
from utils import (plot_2d)
# Given this import
statement, how would we use
plot_2d?
```

A. `utils.plot_2d()`

B. `plot_2d()`

C. `plot_2d.utils()`

Comparing the Different Ways

Import

```
# Tell Python to use the math module
import math
```

- Import the entire universal module

Import

```
# Tell Python which function(s) to import
from utils import (read_image, write_image)
```

- Import specific functions from an existing file

Comparing the Different Ways

Calling a Module

```
# Calculates the square root of 9
print(math.sqrt(9))
```

- module_name.function

Calling a Module

```
# Calls the read_image function from utils
input_grid = read_image(input_file)
```

- function_name(parameter)

Intro to Visualization (what's the big deal?)

ms.csv | spreadspoke_scores.csv

dspoke_scores.csv > data

```
schedule_date,schedule_season,schedule_week,schedule_playoff,team_home,score_home,
1/4/2026,2025,18,FALSE,Cincinnati Bengals,18,20,Cleveland Browns,CIN,-8,45.5,Payc
1/4/2026,2025,18,FALSE,Denver Broncos,19,3,Los Angeles Chargers,DEN,-14.5,37.5,Em
1/4/2026,2025,18,FALSE,Houston Texans,38,30,Indianapolis Colts,HOU,-11,39,NRG Sta
1/4/2026,2025,18,FALSE,Jacksonville Jaguars,41,7,Tennessee Titans,JAX,-13.5,47.5,
1/4/2026,2025,18,FALSE,Las Vegas Raiders,14,12,Kansas City Chiefs,KC,-4.5,36,Alle
1/4/2026,2025,18,FALSE,Los Angeles Rams,37,20,Arizona Cardinals,LAR,-13.5,47.5,So
1/4/2026,2025,18,FALSE,Minnesota Vikings,16,3,Green Bay Packers,MIN,-10.5,36.5,U.
1/4/2026,2025,18,FALSE,New England Patriots,38,10,Miami Dolphins,NE,-11.5,45.5,Gi
1/4/2026,2025,18,FALSE,New York Giants,34,17,Dallas Cowboys,DAL,-3.5,49,MetLife S
1/4/2026,2025,18,FALSE,Philadelphia Eagles,17,24,Washington Commanders,PHI,-4.5,3
1/4/2026,2025,18,FALSE,Pittsburgh Steelers,26,24,Baltimore Ravens,BAL,-3.5,40.5,A
1/10/2026,2025,Wildcard,TRUE,Carolina Panthers,31,34,Los Angeles Rams,LAR,-10.5,4
1/10/2026,2025,Wildcard,TRUE,Chicago Bears,31,27,Green Bay Packers,GB,-1,44.5,Sol
1/11/2026,2025,Wildcard,TRUE,Jacksonville Jaguars,24,27,Buffalo Bills,JAX,-1.5,51
1/11/2026,2025,Wildcard,TRUE,New England Patriots,16,3,Los Angeles Chargers,NE,-3
1/11/2026,2025,Wildcard,TRUE,Philadelphia Eagles,19,23,San Francisco 49ers,PHI,-6
1/12/2026,2025,Wildcard,TRUE,Pittsburgh Steelers,6,30,Houston Texans,HOU,-3,38.5,
1/17/2026,2025,Division,TRUE,Denver Broncos,33,30,Buffalo Bills,DEN,-1.5,46,Empowe
1/17/2026,2025,Division,TRUE,Seattle Seahawks,41,6,San Francisco 49ers,SEA,-7.5,4
1/18/2026,2025,Division,TRUE,Chicago Bears,17,20,Los Angeles Rams,LAR,-3.5,48.5,S
1/18/2026,2025,Division,TRUE,New England Patriots,28,16,Houston Texans,NE,-3.5,41
1/25/2026,2025,Conference,TRUE,Denver Broncos,7,10,New England Patriots,NE,-4,43,
1/25/2026,2025,Conference,TRUE,Seattle Seahawks,31,27,Los Angeles Rams,SEA,-2.5,4
2/8/2026,2025,Superbowl,TRUE,New England Patriots,13,29,Seattle Seahawks,SEA,-5,4
```

Source: https://www.kaggle.com/datasets/tobycrabtree/nfl-scores-and-betting-data?resource=download

~/Downloads/archive/nfl_teams.csv

```
Month,Questions
"2008-07-01 00:00:00","4"
"2008-08-01 00:00:00","3748"
"2008-09-01 00:00:00","14033"
"2008-10-01 00:00:00","14578"
"2008-11-01 00:00:00","12715"
"2008-12-01 00:00:00","12080"
"2009-01-01 00:00:00","15833"
"2009-02-01 00:00:00","17580"
"2009-03-01 00:00:00","20482"
"2009-04-01 00:00:00","21336"
"2009-05-01 00:00:00","25812"
"2009-06-01 00:00:00","28325"
"2009-07-01 00:00:00","32481"
"2009-08-01 00:00:00","32773"
"2009-09-01 00:00:00","33053"
"2009-10-01 00:00:00","36329"
"2009-11-01 00:00:00","38514"
"2009-12-01 00:00:00","37792"
"2010-01-01 00:00:00","44936"
"2010-02-01 00:00:00","44810"
"2010-03-01 00:00:00","52234"
"2010-04-01 00:00:00","50083"
"2010-05-01 00:00:00","51996"
"2010-06-01 00:00:00","55747"
"2010-07-01 00:00:00","60846"
"2010-08-01 00:00:00","63657"
```

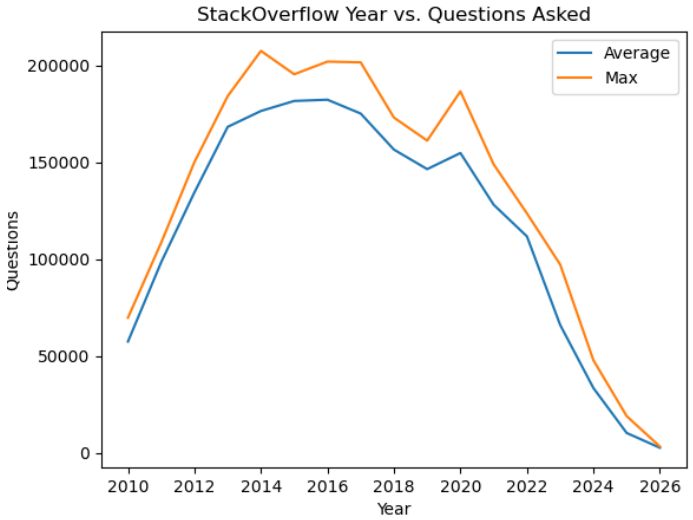
Source: https://data.stackexchange.com/stackoverflow/query/1926661#resultSets

Intro to Visualization (what's the big deal?)

Graphing data allows us to visually represent trends and notice patterns more easily

```
Month,Questions
"2008-07-01 00:00:00","4"
"2008-08-01 00:00:00","3748"
"2008-09-01 00:00:00","14033"
"2008-10-01 00:00:00","14578"
"2008-11-01 00:00:00","12715"
"2008-12-01 00:00:00","12080"
"2009-01-01 00:00:00","15833"
"2009-02-01 00:00:00","17580"
"2009-03-01 00:00:00","20482"
"2009-04-01 00:00:00","21336"
"2009-05-01 00:00:00","25812"
"2009-06-01 00:00:00","28325"
"2009-07-01 00:00:00","32481"
"2009-08-01 00:00:00","32773"
"2009-09-01 00:00:00","33053"
"2009-10-01 00:00:00","36329"
"2009-11-01 00:00:00","38514"
```

VS

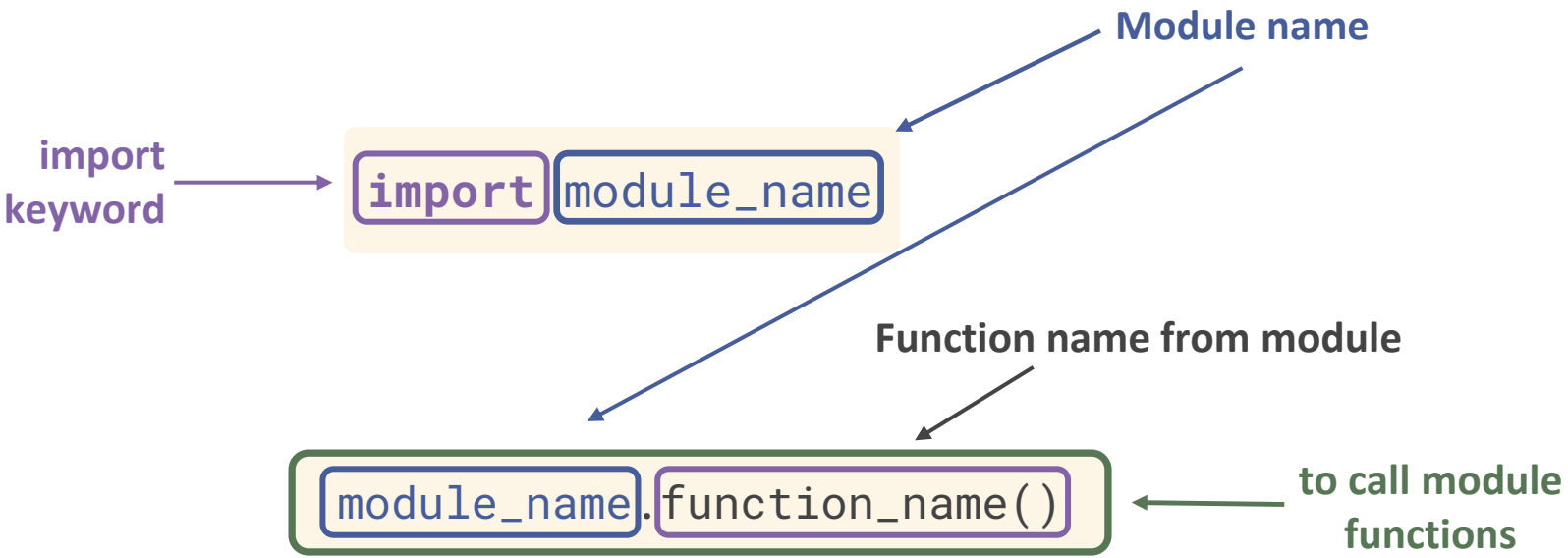


Matplotlib

- Matplotlib is a Python package with functions that can be used to produce graphs
- Matplotlib works well when you have two lists (of the same size)
  - List 1 acts as the x (independent) variable
  - List 2 acts as the y (dependent) variable
- To use the functions we have to import the module!

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

Anatomy of Calling a Function from a Module



Comparing the Different Ways

Import

```
# Tell Python to use the math
# module
import math
```

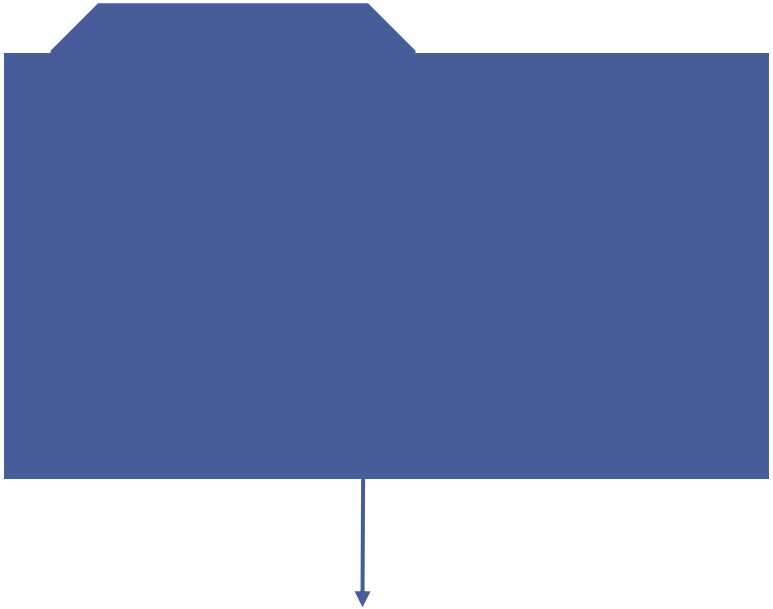
Calling a Module

```
# Calculates the square root of 9
print(math.sqrt(9))
```

```
# Tell Python which function(s) to
# import from the utils module
from utils import (read_image, write_image)
```

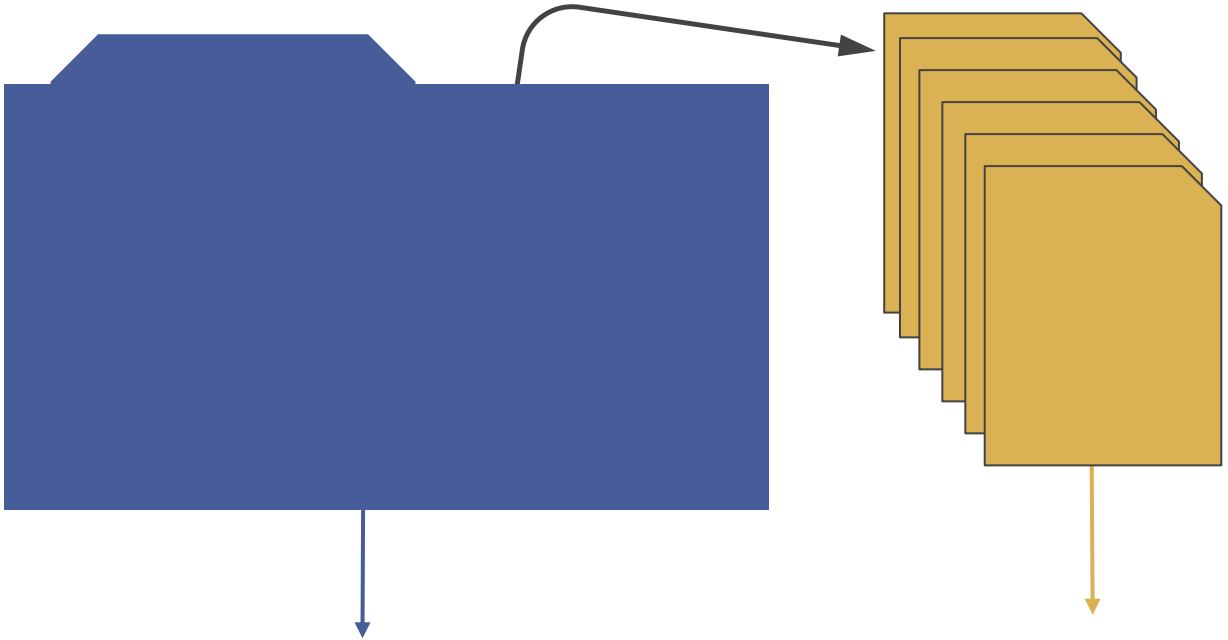
```
# Calls the read_image function from
# utils
input_grid = read_image(input_file)
```

Packages vs Modules vs Functions



Package: a folder that contains multiple modules

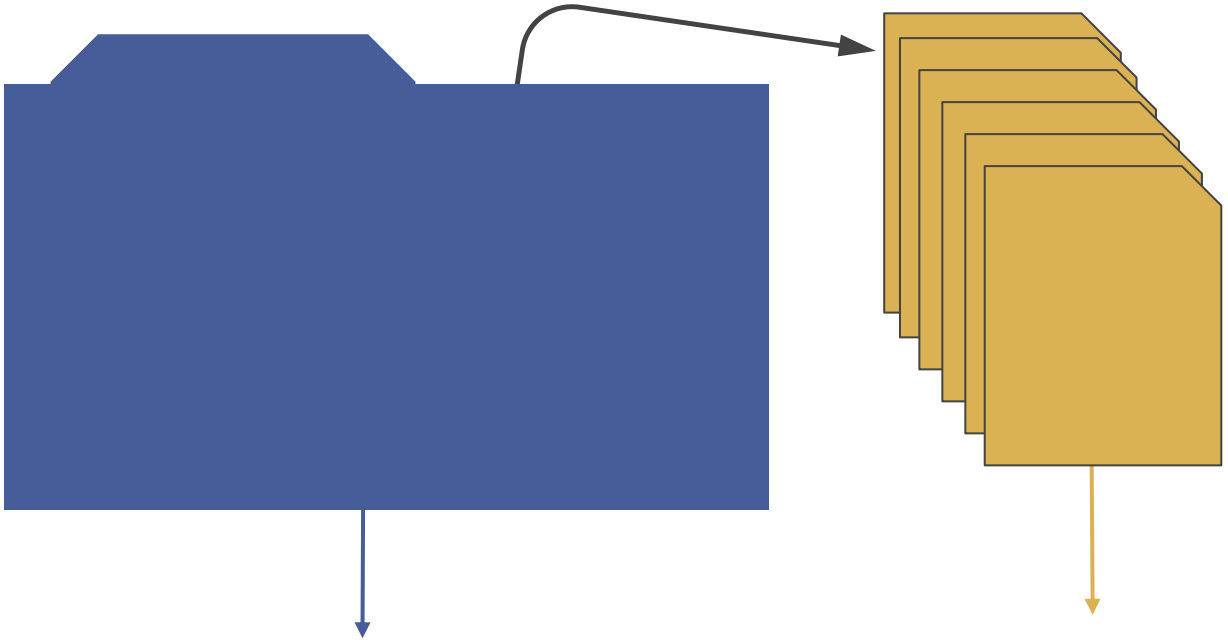
Packages vs Modules vs Functions



Package: a folder that contains multiple modules

Module: a .py file

Packages vs Modules vs Functions



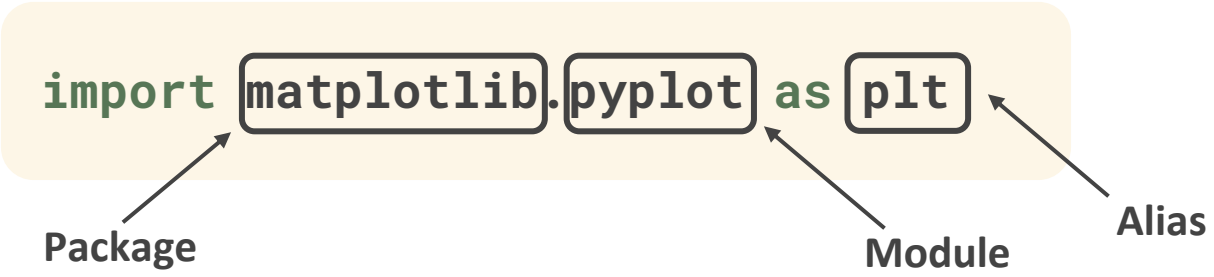
Package: a folder that contains multiple modules

Module: a .py file

Function: compute calculations

Matplotlib

- Matplotlib is a Python package with functions that can be used to produce graphs
- Matplotlib works well when you have two lists (of the same size)
  - List 1 acts as the x (independent) variable
  - List 2 acts as the y (dependent) variable
- To use the functions we have to import the module!



Quick Example!

Matplotlib Functions

```
plt.plot(x_data, y_data)
```

- Pass in two lists of data, with the x values before the y values
- Values at the same index are corresponding to one another
- Connects points in the order they appear in your list - so x_data should usually be in ascending order

```
plt.title("Title")
```

- Pass in a string to title the graph

```
plt.xlabel("X Label")
```

- Pass in a string to label the x axis

```
plt.ylabel("Y Label")
```

- Pass in a string to label the y axis

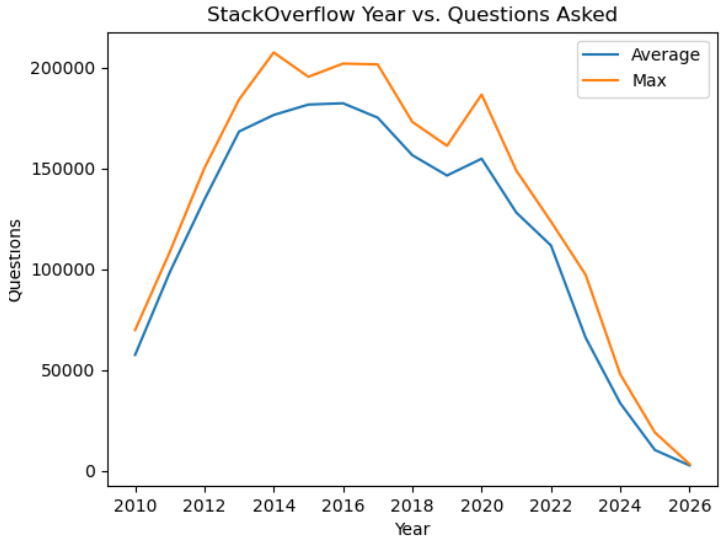
Matplotlib Functions - Data Labels & Legends

```
plt.plot(x_data, y_data,
label = “Your Label”)
```

- Pass in two lists and a string label

```
plt.legend()
```

- Adds a legend to your graph; you can also modify its location



Matplotlib Functions

```
plt.savefig("your_filename")
```

- Pass in a string for the filename
- Will save the figures into a .png file on your computer

```
plt.clf()
```

- Will reset your plot so you are not unintentionally adding new data points/lines each time you run your code

Matplotlib Functions

```
plt.axhline(val)
plt.axvline(val)
```

- Draws axis lines at the specified value
- .axhline() is for a horizontal line
- .axvline() is for a vertical line

```
plt.xlim(low, high)
plt.ylim(low, high)
```

- These functions limit the range of the axes
- Both numbers are inclusive (unlike range)

Example 1: Stackoverflow

```
years = [2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017,
2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026]

avg_questions = [60295, 101322, 137951, 170793, 174519,
182860, 181881, 173728, 155569, 146152, 154152, 126301,
109828, 61857, 30926, 8938, 1894]

max_questions = [79909, 115431, 157836, 188754, 207441,
195373, 201905, 201518, 172998, 161134, 186547, 148842,
123576, 87478, 46149, 18897, 1894]
```

How can we graph this data to see the pattern between time & questions posted?

Example 1: Stackoverflow - Answers

```
plt.plot(years, avg_questions, label="Average")
plt.plot(years, max_questions, label="Max")

plt.title("Stackoverflow Years vs. Questions Asked")
plt.xlabel("Years")
plt.ylabel("Questions Asked")
plt.legend()

plt.savefig("stackoverflow")
plt.clf()
```

Think Pair Share

What happens when this program is run?

```
x_vals = [1, 2, 3, 4]
y_vals = [10, 20, 30]

plt.plot(x_vals, y_vals)
plt.savefig("plot_1")
```

A A graph is drawn using only the first three values of **x_vals**

B A graph is drawn with missing y-values filled in automatically

C The program crashes with an error

D A blank graph appears

Sli.do #cse160

Think Pair Share

What is the difference between the output of the two code blocks on the right?

A There is no difference in the output.

B The top code block will have multiple lines on the graph while the bottom one will have two graphs with one line each.

C There will be an error for the bottom code block for calling `plt.clf()` too many times. The top code block will run normally.

Sli.do #cse160

```
x_vals = [1, 2, 3, 4]
y1_vals = [10, 20, 30, 70]
y2_vals = [20, 50, 40, 90]
plt.plot(x_vals, y1_vals)
plt.plot(x_vals, y2_vals)
plt.savefig("image")
plt.clf()
```

```
x_vals = [1, 2, 3, 4]
y1_vals = [10, 20, 30, 70]
y2_vals = [20, 50, 40, 90]
plt.plot(x_vals, y1_vals)
plt.savefig("image_1")
plt.clf()
plt.plot(x_vals, y2_vals)
plt.savefig("image_2")
plt.clf()
```

NFL

Example 2: NFL

```
schedule_season,schedule_date,team_home.....
2018,2019-01-06,Chicago Bears.....
2022,2023-01-08,Atlanta Falcons.....
1981,1981-10-11,San Francisco 49ers.....
2014,2014-10-12,Oakland Raiders.....
2012,2012-10-14,Arizona Cardinals.....
```

games_data.csv

```
def load_games(csv_file):
    ...
    with open(csv...
        ...
        ...
```

parse_data.py

```
def get_avgs_and_seasons(games):
        ...
    season_totals = {}
    season_counts = {}
    ...
```

plot_nfl.py

Example 2: NFL

```
schedule_season,schedule_date,team_home.....
2018,2019-01-06,Chicago Bears.....
2022,2023-01-08,Atlanta Falcons.....
1981,1981-10-11,San Francisco 49ers.....
2014,2014-10-12,Oakland Raiders.....
2012,2012-10-14,Arizona Cardinals.....
```

games_data.csv

```
def load_games(csv_file):
    ...
    with open(csv...
        ...
        ...
```

parse_data.py

CSV file containing the data

```
def get_avgs_and_seasons(games):
        ...
    season_totals = {}
    season_counts = {}
    ...
```

plot_nfl.py

Example 2: NFL

```
schedule_season,schedule_date,team_home.....
2018,2019-01-06,Chicago Bears.....
2022,2023-01-08,Atlanta Falcons.....
1981,1981-10-11,San Francisco 49ers.....
2014,2014-10-12,Oakland Raiders.....
2012,2012-10-14,Arizona Cardinals.....
```

games_data.csv

```
def load_games(csv_file):
    ...
    with open(csv...
        ...
        ...
```

parse_data.py

```
def get_avgs_and_seasons(games):
        ...
    season_totals = {}
    season_counts = {}
    ...
```

plot_nfl.py

Python file that contains a function load_games, which parses csv data and returns a dictionary representation of the data

Example 2: NFL

```
schedule_season,schedule_date,team_home.....
2018,2019-01-06,Chicago Bears.....
2022,2023-01-08,Atlanta Falcons.....
1981,1981-10-11,San Francisco 49ers.....
2014,2014-10-12,Oakland Raiders.....
2012,2012-10-14,Arizona Cardinals.....
```

games_data.csv

```
def load_games(csv_file):
    ...
    with open(csv...
        ...
        ...
```

parse_data.py

A file containing a function that parses dictionary data into two lists we can use to graph! Also the place we'll be putting graphing code

```
def get_avgs_and_seasons(games):
        ...
    season_totals = {}
    season_counts = {}
    ...
```

plot_nfl.py

Example 2: NFL

```
schedule_season,schedule_date…
2018,2019-01-06,Chicago Bears…..
2022,2023-01-08,Atlanta Falcons…..
1981,1981-10-11,San Francisco 49ers…..
2014,2014-10-12,Oakland Raiders…..
2012,2012-10-14,Arizona Cardinals…..
```

games_data.csv

```
def load_games(csv_file):
    ...
    with open(csv...
        ...
        ...
```

parse_data.py

```
def get_avgs_and_seasons(games):
        ...
    season_totals = {}
    season_counts = {}
    ...
```

plot_nfl.py

Your Task

Within the **plot_nfl.py** file, write code that calls the previously defined functions and plots the data returned from **get_avgs_and_seasons** on a graph (with a title and axis labels)

Example 2: NFL - Answers

```
import matplotlib.pyplot as plt
from parse_data import load_games
```

```
games = load_games("games_data.csv")
seasons, avg_totals = get_avgs_and_seasons(games)

plt.plot(seasons, avg_totals)

plt.title("Average Total Points per Season")
plt.xlabel("Season")
plt.ylabel("Average Total Points")
plt.savefig("nfl")
plt.clf()
```