Programming Principles in Python (CSCI 503/490)

Introduction

Dr. David Koop

Python Experience?

Programming Principles?

Why Python?

Productivity

Libraries, Libraries, Libraries

What about speed?

Administrivia

- Course Web Site
- TA: Mohammed Abdul Moyeed (Blackboard Collaborate)
- Syllabus
  - Plagiarism
  - Accommodations
- Assignments
- Tests: 2 (Feb. 23, April 6) and Final (May 9)
- Course is offered to both undergraduates (CS 490) and graduates (CS 503)
  - Grad students have extra topics, exam questions, assignment tasks

Academic Honesty

- **Do not cheat!**
- You will receive a **zero** for any assignment/exam/etc. where cheating has occurred
- You will **fail** the course if you cheat more than once
- Misconduct is reported through the university's system
- You **may** discuss problems and approaches with other students
- You **may not** copy or transcribe code from another source

In-Person Course

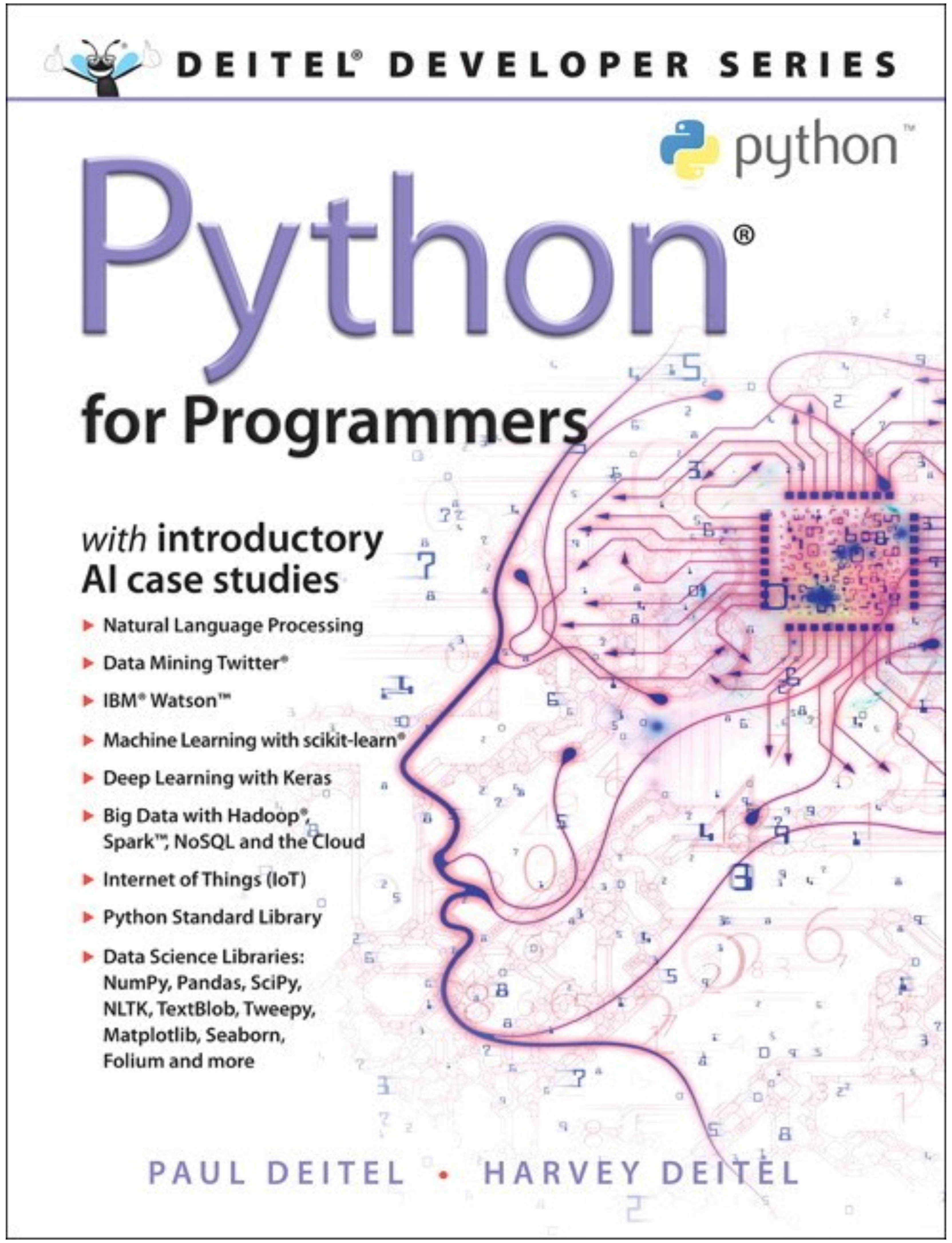
- Plan is for lectures to be 2:00-3:15pm MW in PM 153
  - Better for learning if you are engaged
  - **Ask questions**
  - Please advise me of any issues, including those related to your health
- Due to the ongoing pandemic, plans may be modified by the university
  - Any changes will be announced as soon as possible
- Slides will be posted to the course website

Office Hours & Email

- Moyeed's office hours will be held via Blackboard Collaborate
  - TuTh: 11:30am-2:30pm
- Prof. Koop's office hours will be held in person
  - MW: 10:45am-12pm, or by appointment (can be Zoom)
- You do not need an appointment to stop by during scheduled office hours, but please adhere to university regulations (Protecting the Pack)
- If you wish to meet virtually, please schedule an appointment
- If you need an appointment, please email me with **details** about what you wish to discuss and times that would work for you
- Many questions can be answered via email. **Please consider writing an email before scheduling a meeting.**

Course Material

- Textbook:
  - Recommended: Python for Programmers
  - Good overview + data science examples
- Many other resources are available:
  - https://wiki.python.org/moin/BeginnersGuide
  - https://wiki.python.org/moin/IntroductoryBooks
  - http://www.pythontutor.com
  - https://www.python-course.eu
  - https://software-carpentry.org/lessons/



DEITEL® DEVELOPER SERIES

python™

Python®

for Programmers

with introductory AI case studies

- Natural Language Processing
- Data Mining Twitter®
- IBM® Watson™
- Machine Learning with scikit-learn®
- Deep Learning with Keras
- Big Data with Hadoop®, Spark™, NoSQL and the Cloud
- Internet of Things (IoT)
- Python Standard Library
- Data Science Libraries: NumPy, Pandas, SciPy, NLTK, TextBlob, Tweepy, Matplotlib, Seaborn, Folium and more

PAUL DEITEL • HARVEY DEITEL

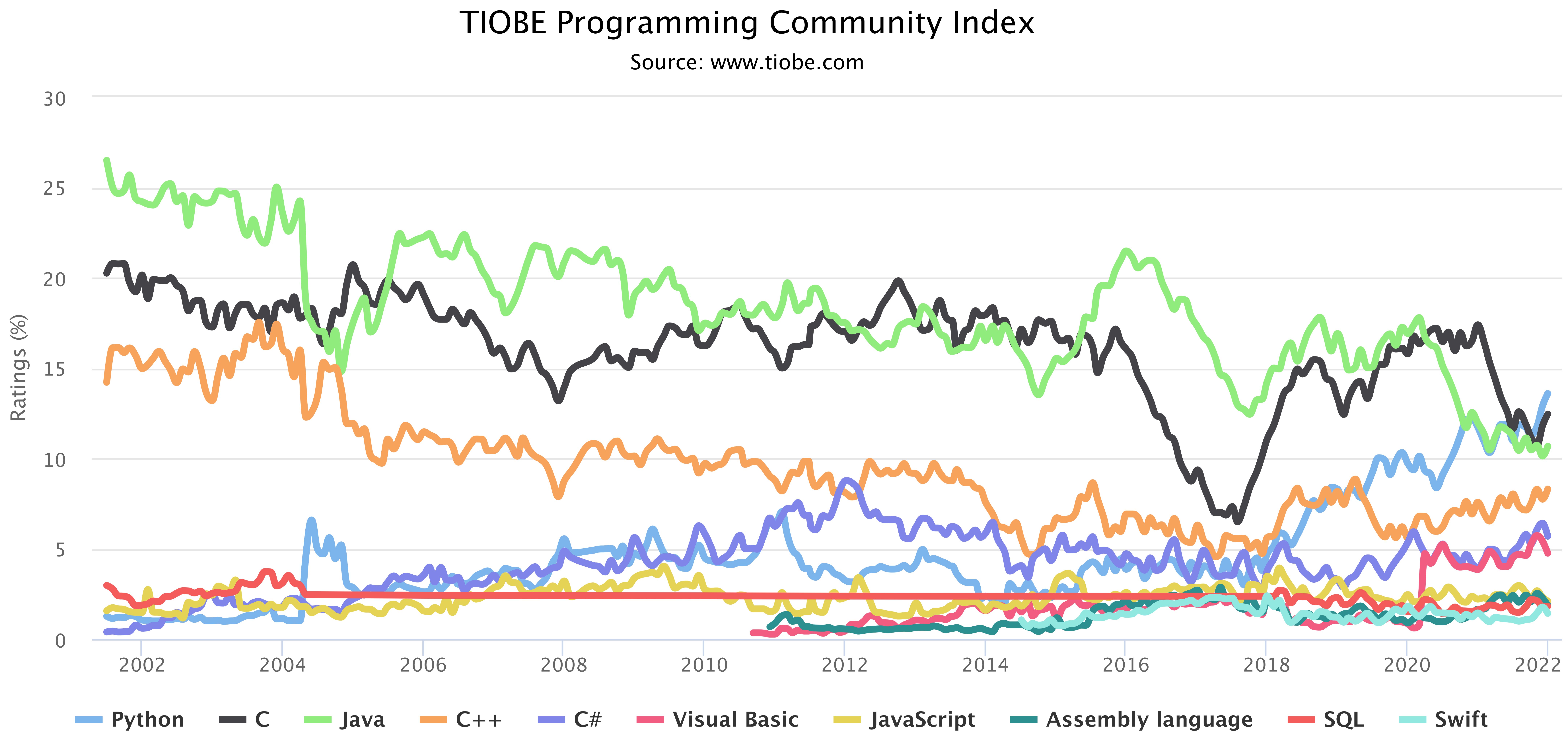
Course Material

- Software:
  - Anaconda Python Distribution (https://www.continuum.io/downloads): makes installing python packages easier
  - Jupyter Notebook: Web-based interface for interactively writing & executing Python code
  - JupyterLab: An updated web-based interface that includes the notebook and other cool features
  - JupyterHub: Access everything through a server

Python

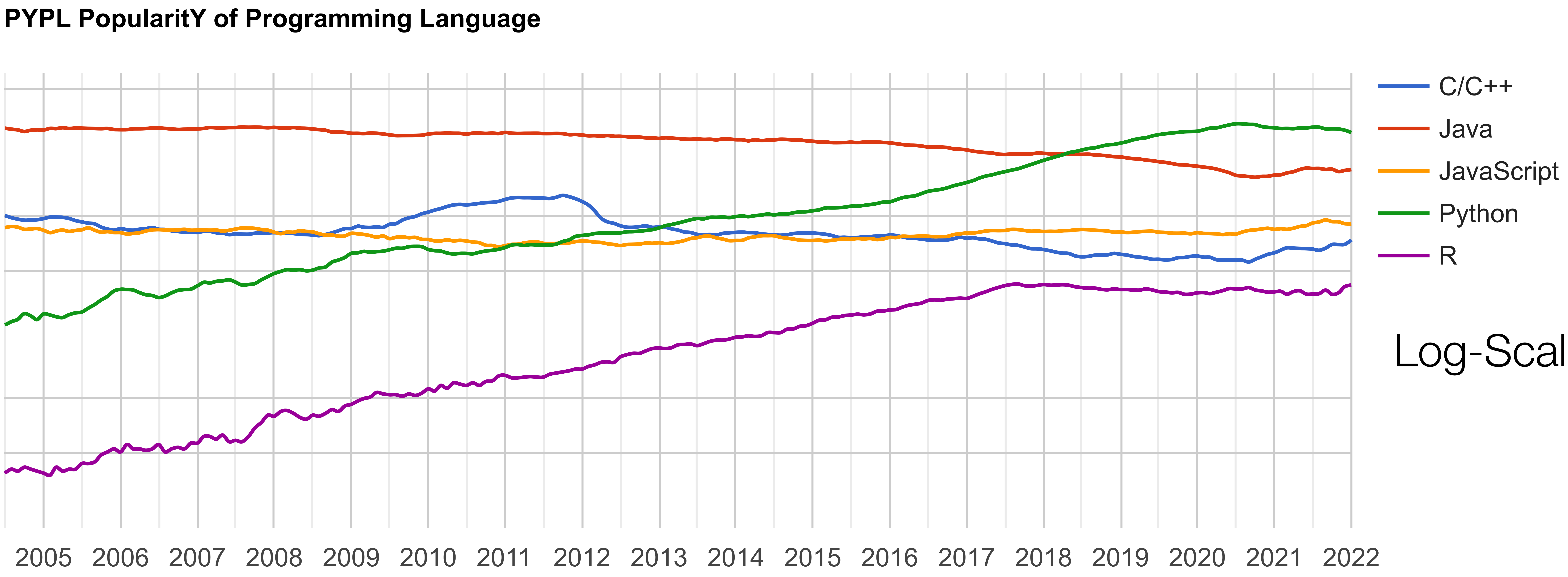
- Started in December 1989 by Guido van Rossum
- "Python has surpassed Java as the top language used to introduce U.S. students to programming…" (ComputerWorld, 2014)
- Python is also a top language for data science
- High-level, interpreted language
- Supports multiple paradigms (OOP, procedural, functional)
- Help programmers write **readable** code, use less code to do more
- Lots of libraries for python
- Designed to be extensible, easy to wrap code from other languages like C/C++
- Open-source with a large, passionate community

Python was the #1 Programming Language in 2021



[TIOBE]

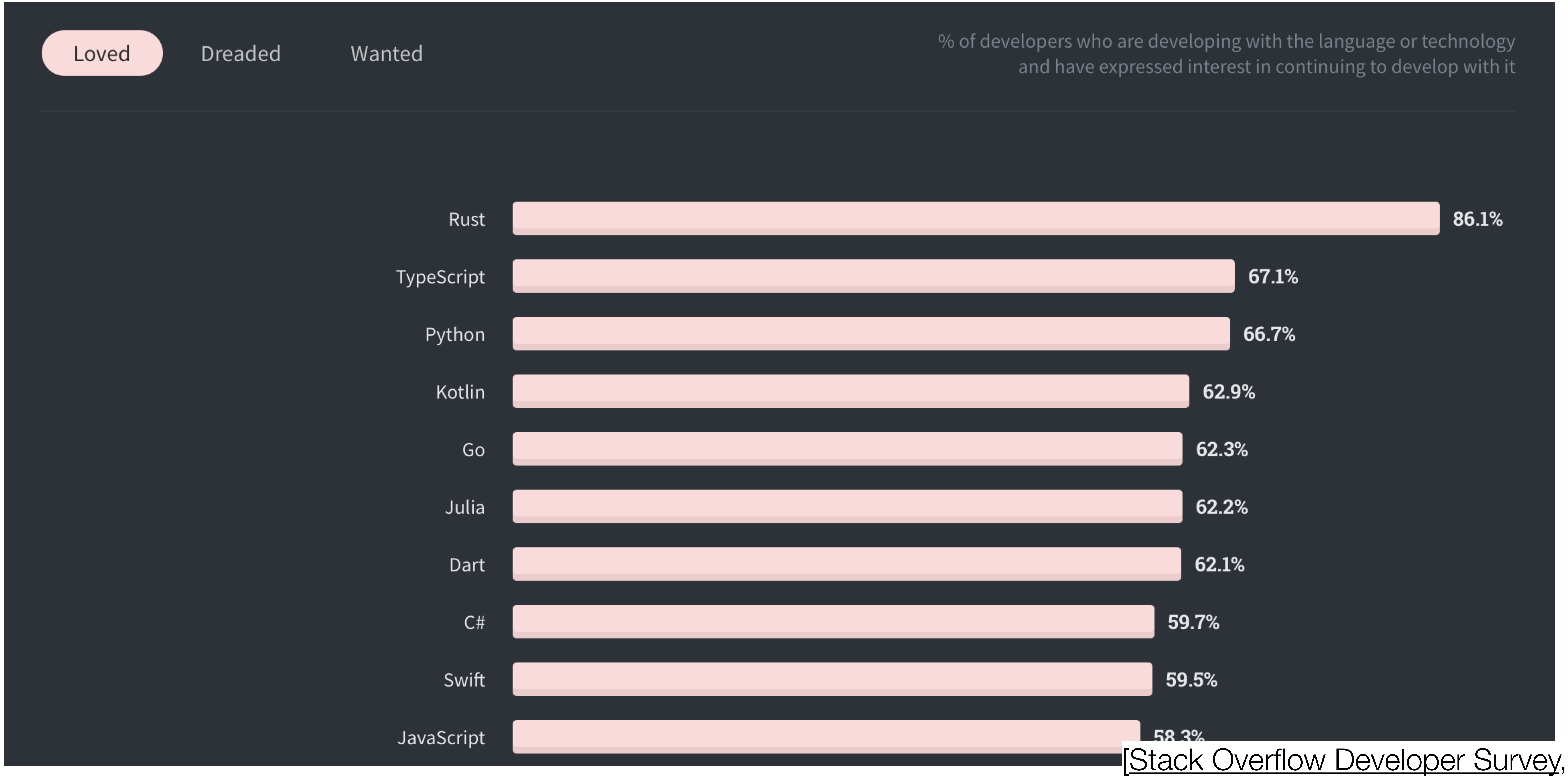
Even Wider Gap in Google Tutorial Searches



Log-Scale

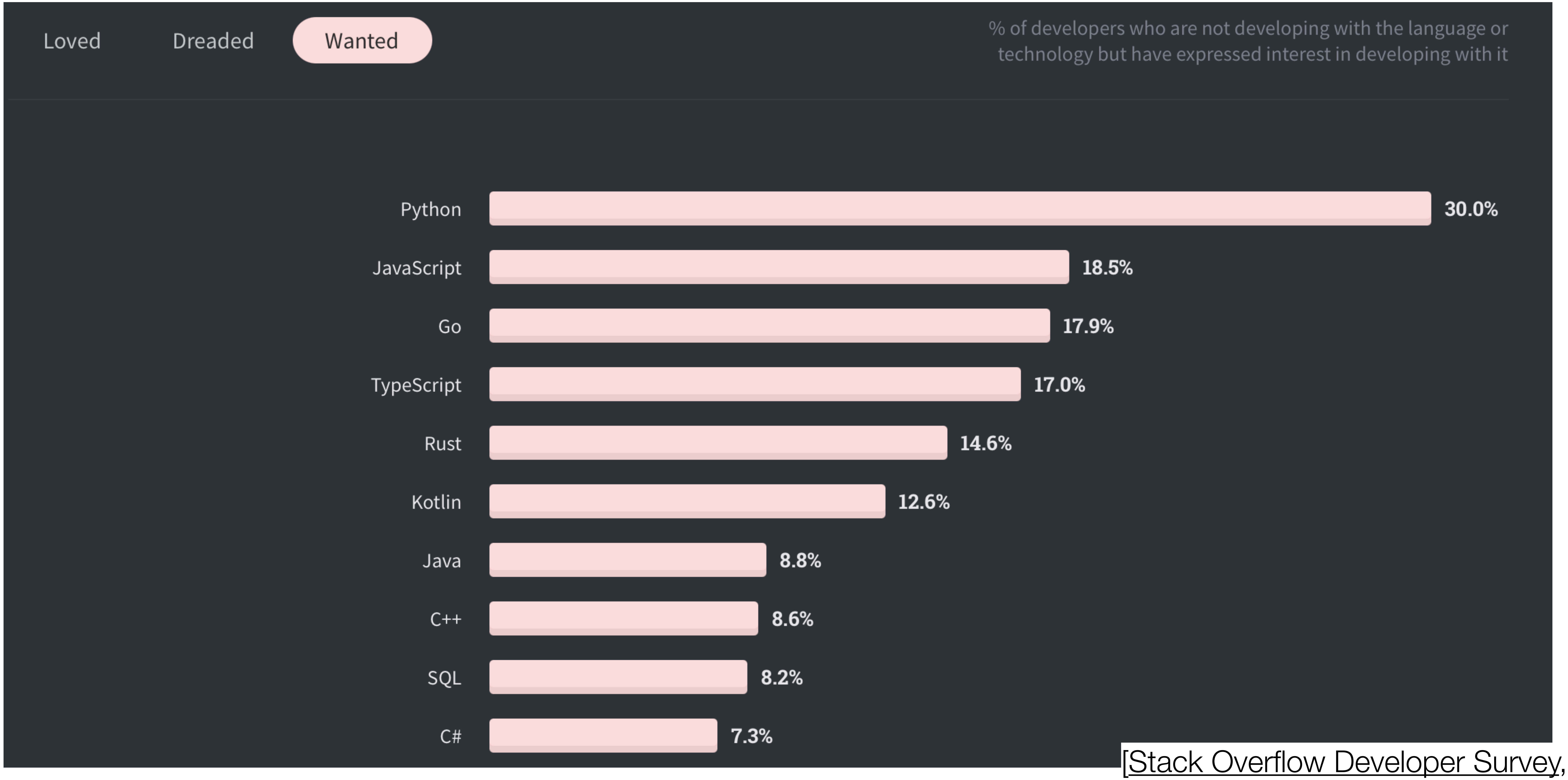
[PopularitY of Programming Language]

StackOverflow Languages



[Stack Overflow Developer Survey, 2020]

StackOverflow Languages



[Stack Overflow Developer Survey, 2020]

Modes of Computation

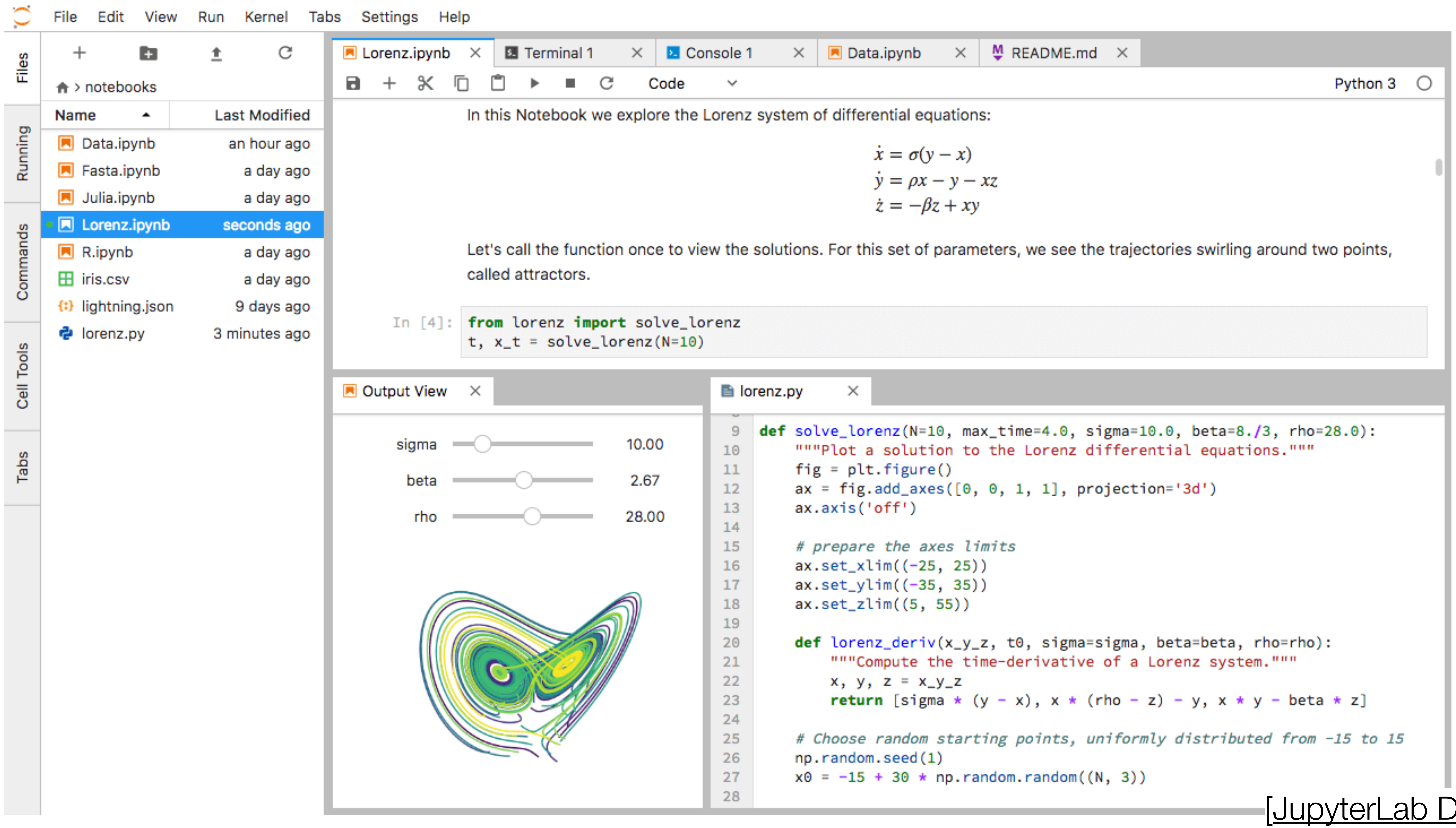
- Python is **interpreted**: you can run one line at a line without compiling
- Interpreter in the Shell
  - Execute line by line
  - Hard to structure loops
  - Usually execute whole files (called scripts) and edit those files
- Notebook
  - Richer results (e.g. images, tables)
  - Can more easily edit past code
  - Re-execute any cell, whenever

Python Differences

- Dynamic Typing
  - A variable does not have a fixed type
  - Example: `a = 1; a = "abc"`
- Indentation
  - Braces define blocks in Java, good style is to indent but not required
  - Indentation is critical in Python

```
z = 20
if x > 0:
    if y > 0:
        z = 100
else:
    z = 10
```

JupyterLab and Jupyter Notebooks



Name	Last Modified
Data.ipynb	an hour ago
Fasta.ipynb	a day ago
Julia.ipynb	a day ago
Lorenz.ipynb	seconds ago
R.ipynb	a day ago
iris.csv	a day ago
lightning.json	9 days ago
lorenz.py	3 minutes ago

In this Notebook we explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

Let's call the function once to view the solutions. For this set of parameters, we see the trajectories swirling around two points, called attractors.

In [4]:
```
from lorenz import solve_lorenz
t, x_t = solve_lorenz(N=10)
```

sigma 10.00

beta 2.67

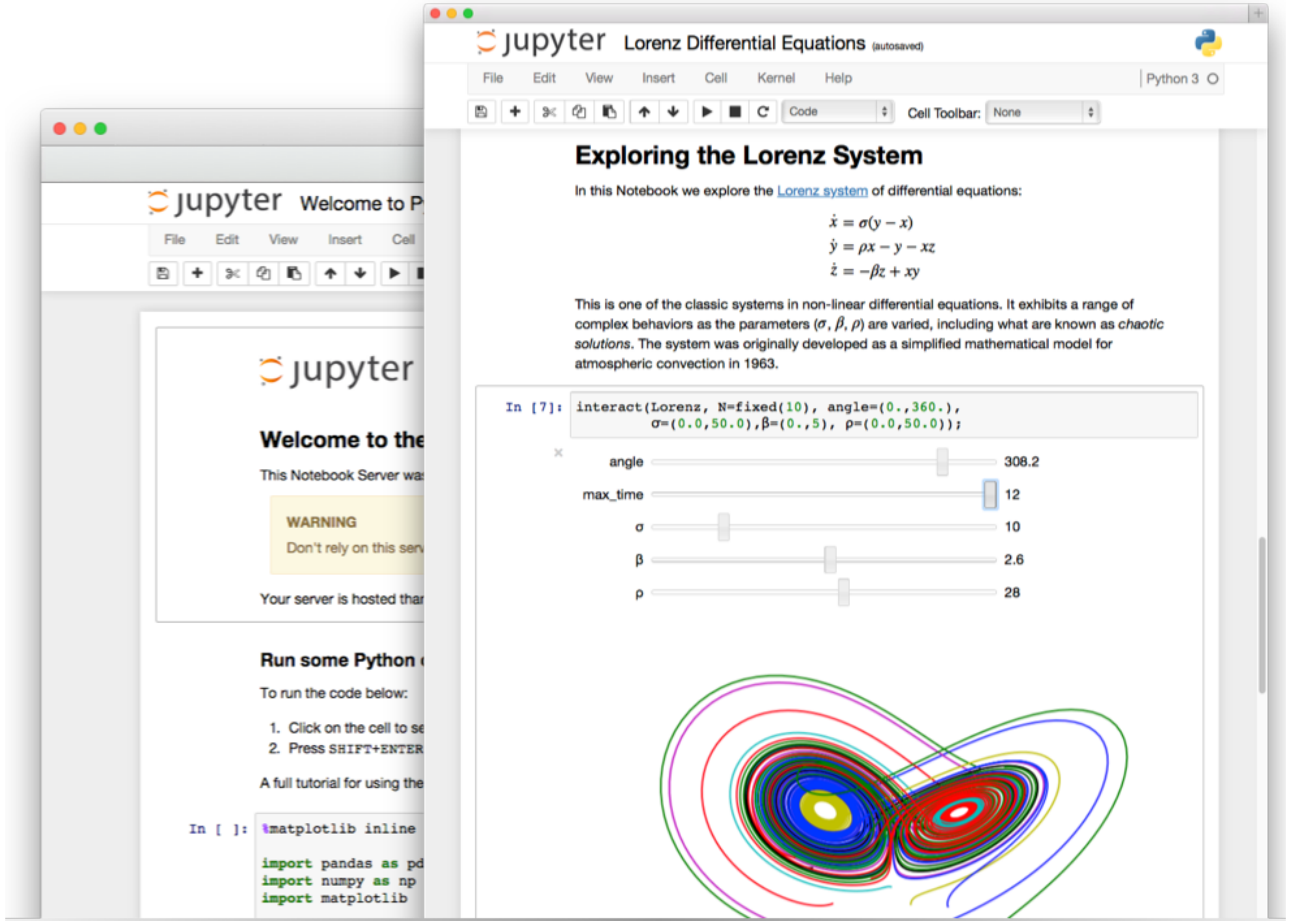
rho 28.00

```
def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=8./3, rho=28.0):
    """Plot a solution to the Lorenz differential equations."""
    fig = plt.figure()
    ax = fig.add_axes([0, 0, 1, 1], projection='3d')
    ax.axis('off')

    # prepare the axes limits
    ax.set_xlim((-25, 25))
    ax.set_ylim((-35, 35))
    ax.set_zlim((5, 55))

    def lorenz_deriv(x_y_z, t0, sigma=sigma, beta=beta, rho=rho):
        """Compute the time-derivative of a Lorenz system."""
        x, y, z = x_y_z
        return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]

    # Choose random starting points, uniformly distributed from -15 to 15
    np.random.seed(1)
    x0 = -15 + 30 * np.random.random((N, 3))

```

[JupyterLab Documentation]

Jupyter Notebooks

- Display rich representations and text
- Uses Web technology
- Cell-based
- Built-in editor
- GitHub displays notebooks



Welcome to the

This Notebook Server wa

WARNING

Don't rely on this serv

Your server is hosted thar

Run some Python

To run the code below:

1. Click on the cell to se
2. Press SHIFT+ENTER

A full tutorial for using the

```
In [ ]: %matplotlib inline

        import pandas as pd
        import numpy as np
        import matplotlib
```

Exploring the Lorenz System

In this Notebook we explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

This is one of the classic systems in non-linear differential equations. It exhibits a range of complex behaviors as the parameters (σ, β, ρ) are varied, including what are known as *chaotic solutions*. The system was originally developed as a simplified mathematical model for atmospheric convection in 1963.

```
In [7]: interact(Lorenz, N=fixed(10), angle=(0.,360.),
                 σ=(0.0,50.0),β=(0.,5), ρ=(0.0,50.0));
```

angle 308.2
max_time 12
σ 10
β 2.6
ρ 28

[Jupyter]

Jupyter Notebooks

- An interactive programming environment
- Runs in your web browser
- Displays results (even interactive maps) inline
- Originally designed for Python
- Supports other languages, too
- You decide how to divide code into executable cells
- Shift+Enter (or the "play" button) to execute a cell

Notebooks in JupyterLab

- Directory view on left
- Create new notebooks using "+" button, "New" from the File menu, or Launcher window
  - Notebook originally has name "Untitled"
  - Click on "Untitled" to change the name (do this!)
- Save a notebook using the command under the File menu
- Shutting down the notebook — use Close and Shutdown Kernel
  - Web browser is **interface** to display code and results
  - **Kernel** actually runs the code: usually see messages in a console/terminal window

Notebooks in JupyterLab

- Open a notebook by going back to the file browser and clicking on it like you would in a desktop view
- Past results are displayed—does not mean they are loaded in memory
- Use "Run All" or "Run All Above" to re-execute past work
  - If you shut down the kernel, all of the data and variables you defined need to be redefined (so you need to re-run all)
  - **Watch Out—Order Matters**: If you went back and re-executed cells in a different order than they are shown, doing "Run All" may not produce the same results!
- Edit mode (green) versus Command mode (blue == **Be Careful**)
- Learn keyboard shortcuts

Notebooks in JupyterLab

- Can write code or plain text (can be styled Markdown)
  - Choose the type of cell using the dropdown menu
- Cells break up your code, but all data is **global**
  - Defining a variable `a` in one cell means that variable is accessible in **any** other cell
  - This includes cells **above** the cell `a` was defined in!
- Remember **Shift+Enter** to execute
- Enter just adds a new line
- Use ?<function_name> for help
- Use Tab for **auto-complete** or suggestions

JupyterLab

- More than just notebooks:
  - Text editor
  - Console
  - Custom components (Many extensions)
- Arrange multiple documents and views
- JupyterLab Documentation

Using Python & JupyterLab Locally

- www.anaconda.com/download/
- Anaconda has JupyterLab
- Use Python 3.9
- Anaconda Navigator
  - GUI application for managing Python environment
  - Can install packages
  - Can start JupyterLab
- Can also use the shell to do this:
  - `$ jupyter lab`
  - `$ conda install <pkg_name>`

Using Python & JupyterLab on Course Server

- Stay tuned…

Chicago Food Inspections

- Data: Information about food facility inspections in Chicago
- Data Source: https://data.cityofchicago.org/Health-Human-Services/Food-Inspections/4ijn-s7e5/data
- Fields: Name, Facility Type, Risk, Violations, Location, etc.

Chicago Food Inspections Exploration

- Based on David Beazley's PyData Chicago talk
- YouTube video: https://www.youtube.com/watch?v=j6VSAsKAj98
- Our in-class exploration:
  - Don't focus on the syntax
  - Focus on how interactive Python makes this exploration work well