

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: Gagana Aladhalli Ramegowda (Office: PM 356)
- Syllabus
 - Plagiarism
 - Accommodations
- Assignments
- Tests: 2 (Feb. 22, Apr. 5) and Final (May 10)
- 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

Schedule

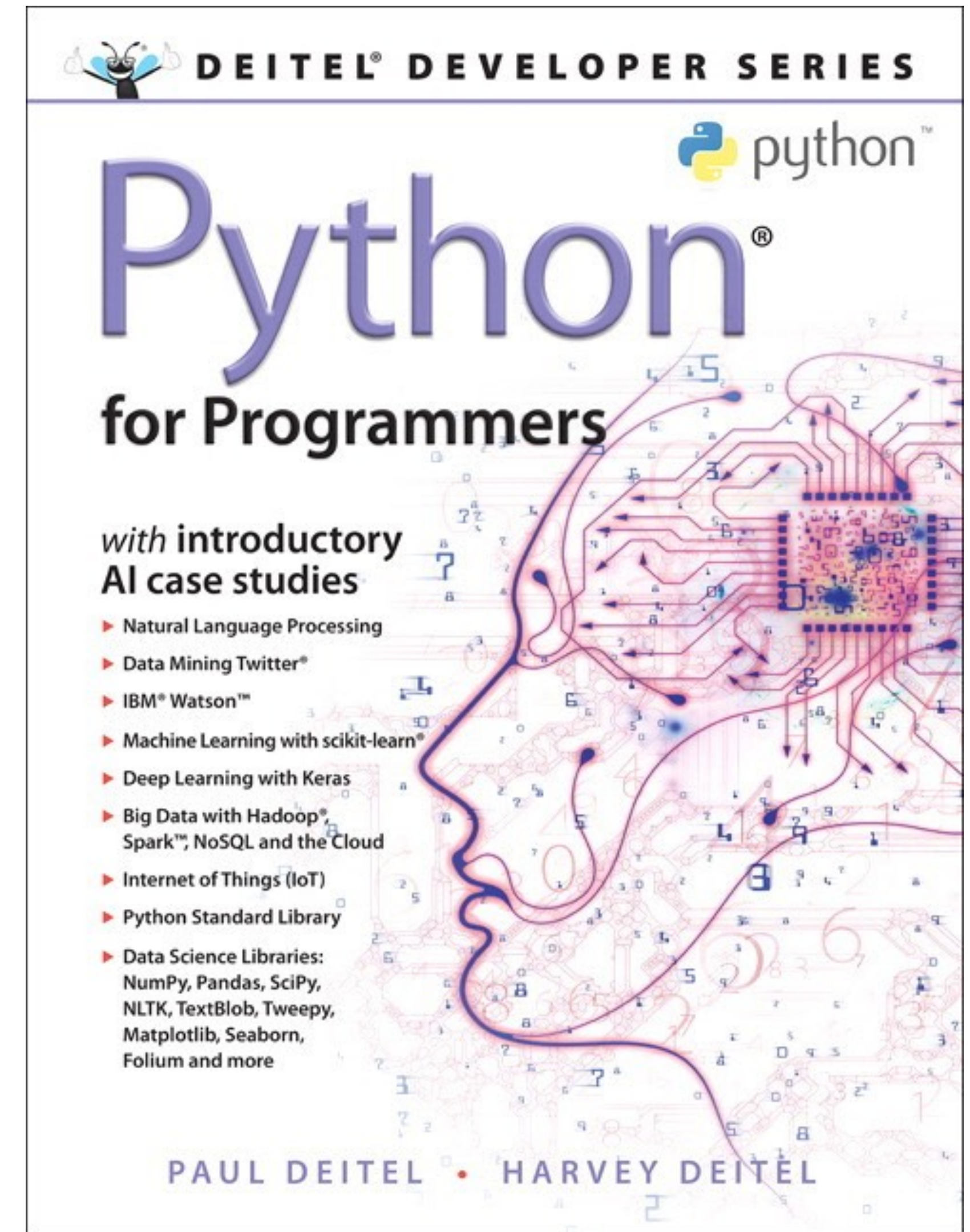
- Lectures are 11:00am-12:15pm MW in PM 110
 - Better for learning if you are engaged
 - **Ask questions**
 - Please advise me of any issues, including those related to your health
- Any changes will be announced as soon as possible
- Slides will be posted to the course website

Office Hours & Email

- Gagana's office hours will be held in person in PM 354
 - TuTh: 2:00-5:00pm
- Prof. Koop's office hours will be held in person in PM 461
 - M: 2:00-3:15pm, W: 1:00-2:15pm, or by appointment
 - You do not need an appointment to stop by during scheduled office hours,
 - 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/>



Course Material

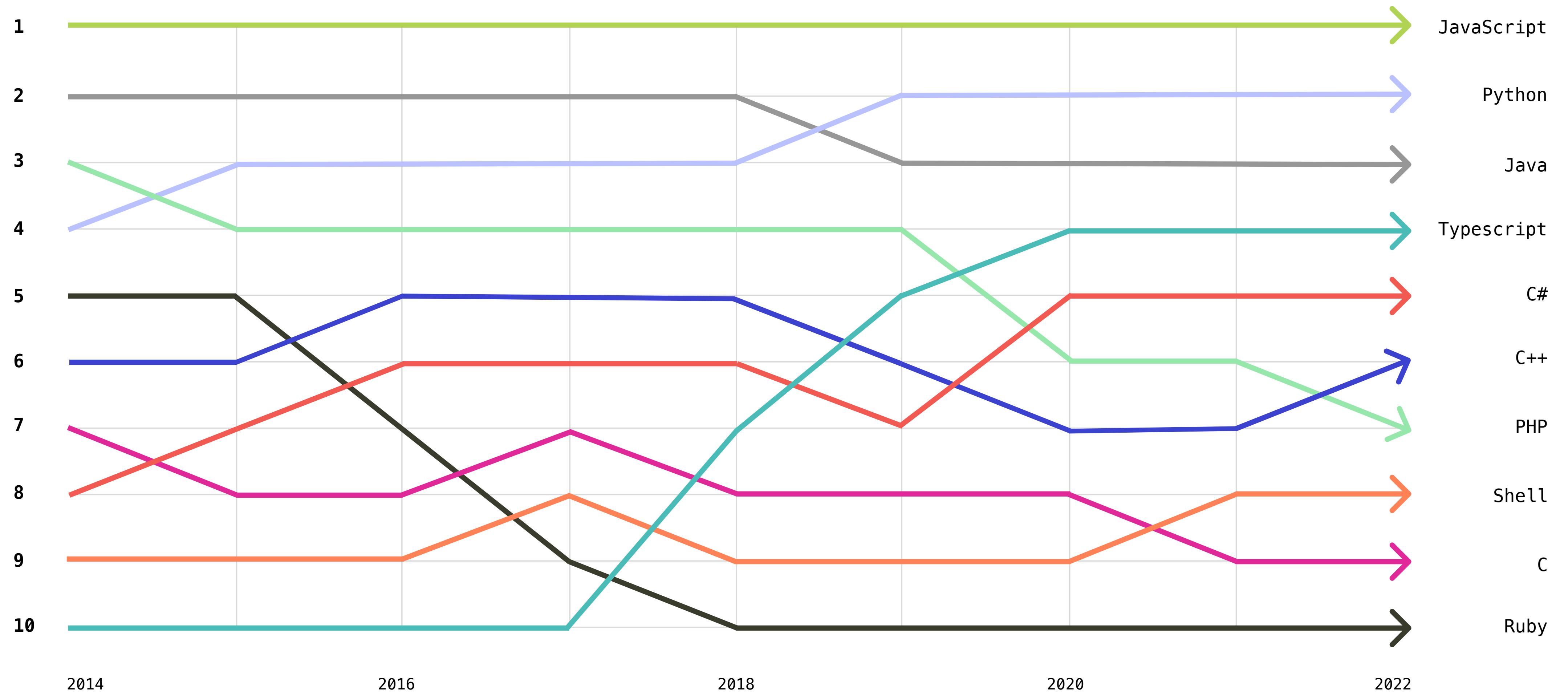


- Software:
 - Anaconda Python Distribution (<https://www.continuum.io/downloads>): makes installing python packages easier
 - Jupyter Notebook: Web-based interface for interactively writing & executing Python code
 - JupyterLab: An updated web-based interface that includes the notebook and other cool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

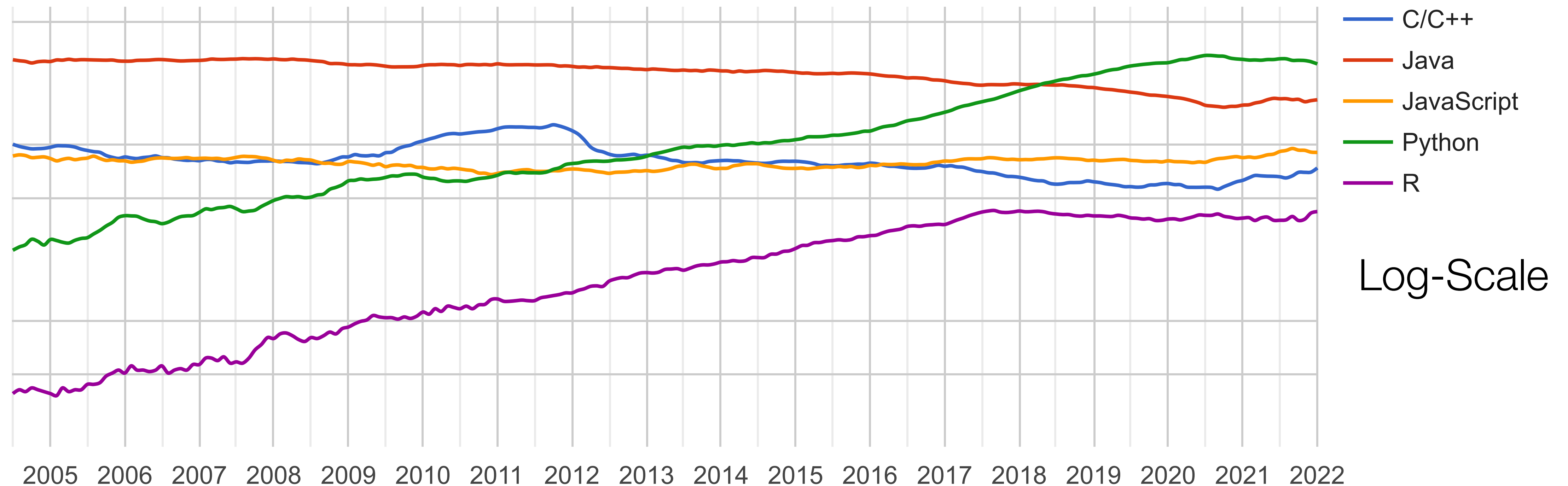
Python the #2 Programming Language in 2022



[GitHub]

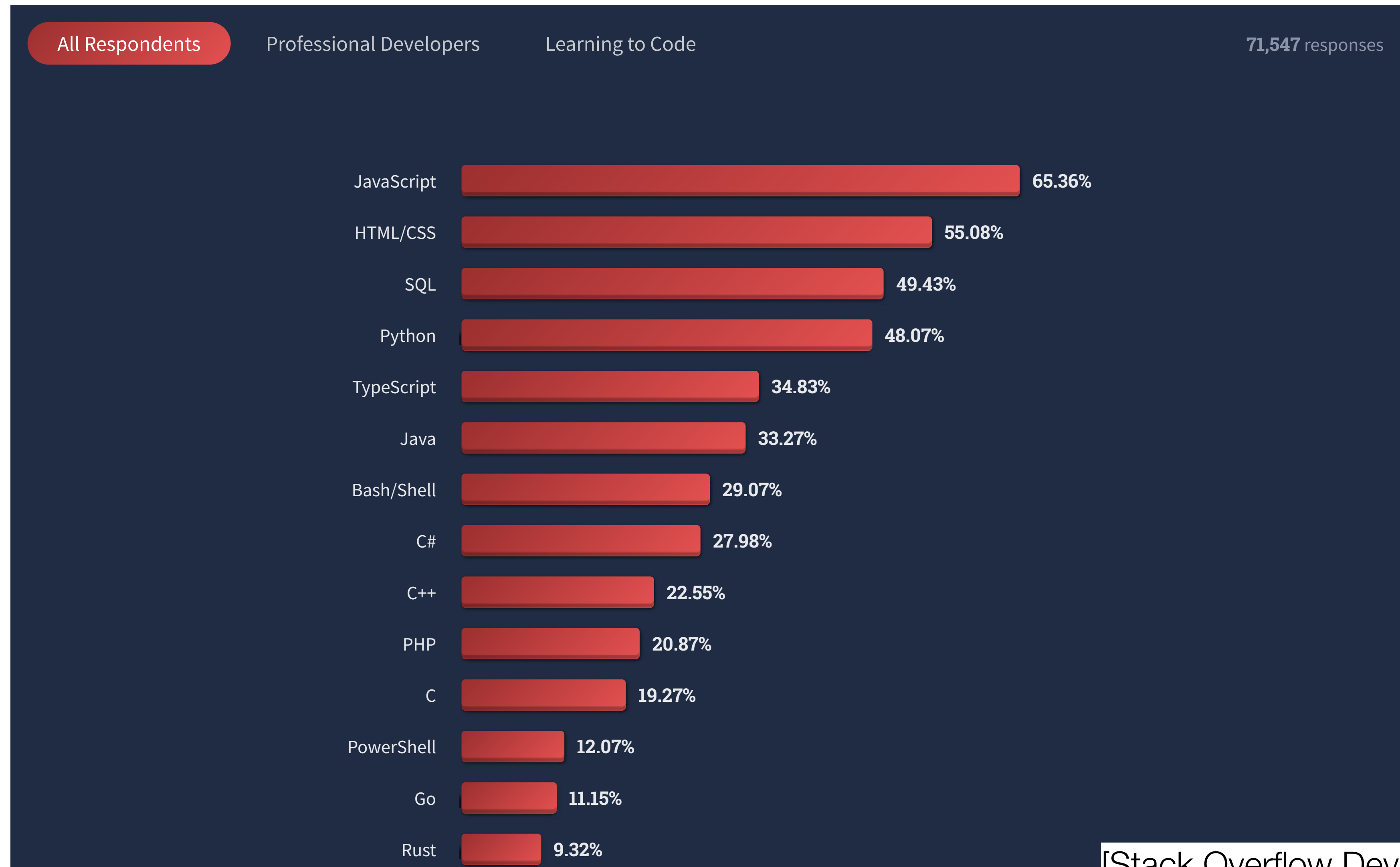
Even Wider Gap in Google Tutorial Searches

PYPL Popularity of Programming Language



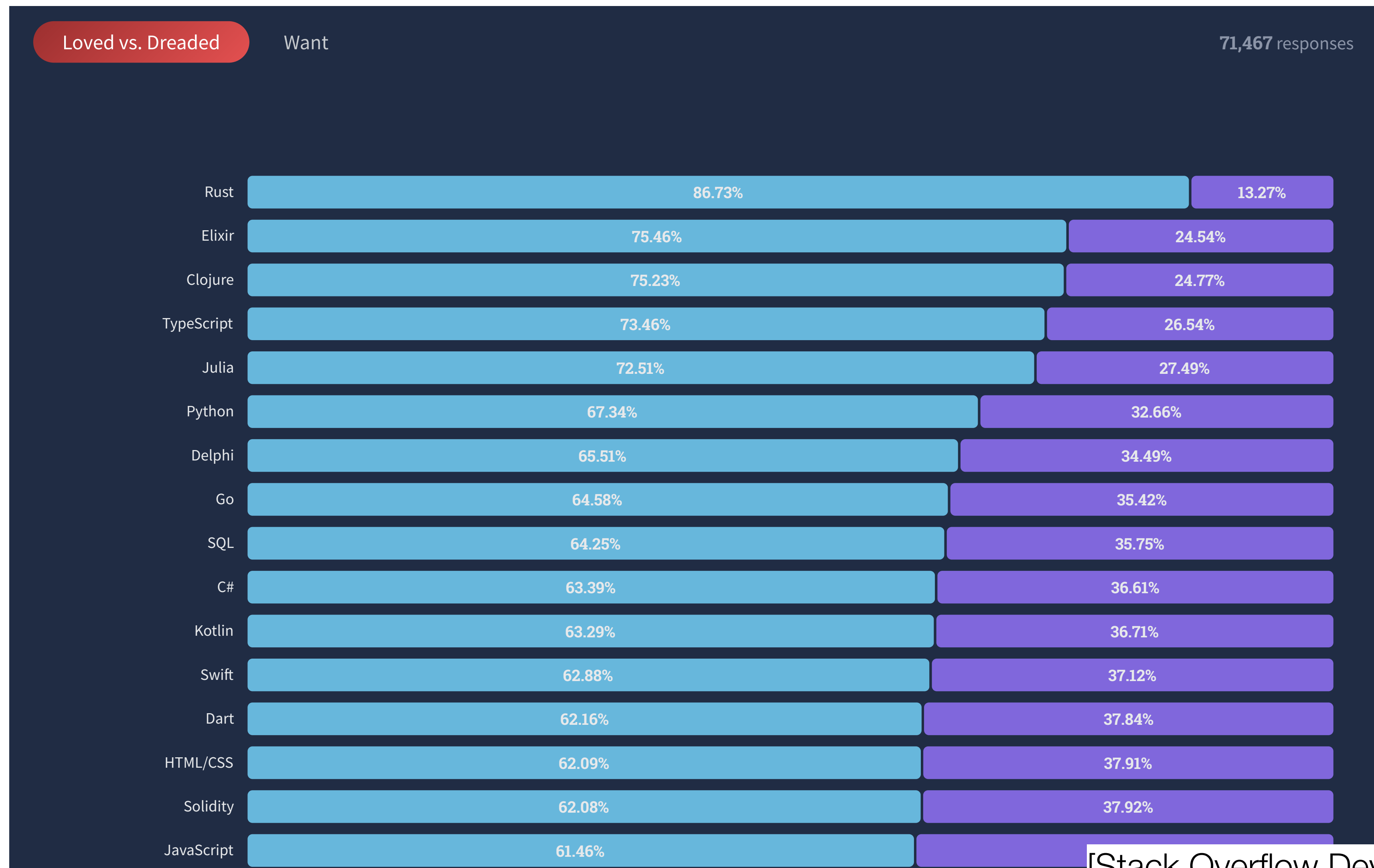
[Popularity of Programming Language]

StackOverflow Language Usage



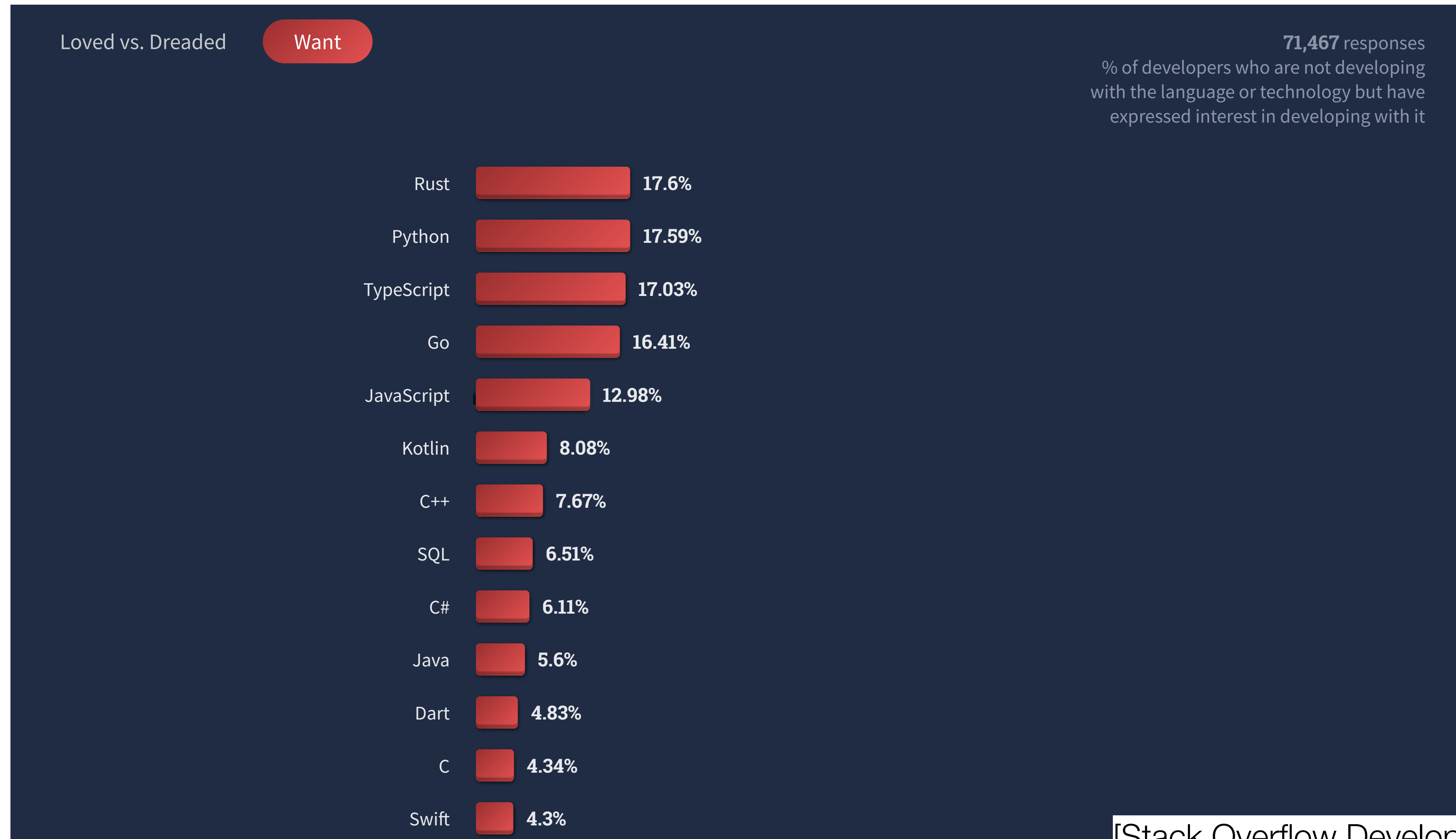
[Stack Overflow Developer Survey, 2022]

StackOverflow Language Preferences



[Stack Overflow Developer Survey, 2022]

StackOverflow Language Interest



[Stack Overflow Developer Survey, 2022]

Modes of Computation

- Python is **interpreted**: you can run one line at a time 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

The screenshot displays the JupyterLab environment. On the left, a sidebar contains a 'Files' panel with a list of notebooks and files, including 'Data.ipynb', 'Fasta.ipynb', 'Julia.ipynb', 'Lorenz.ipynb' (selected), 'R.ipynb', 'iris.csv', 'lightning.json', and 'lorenz.py'. Below this are 'Running', 'Commands', 'Cell Tools', and 'Tabs' panels. The main area is divided into three panes. The top pane shows the 'Lorenz.ipynb' notebook with a text cell containing the Lorenz system equations and a code cell with the following Python code:

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

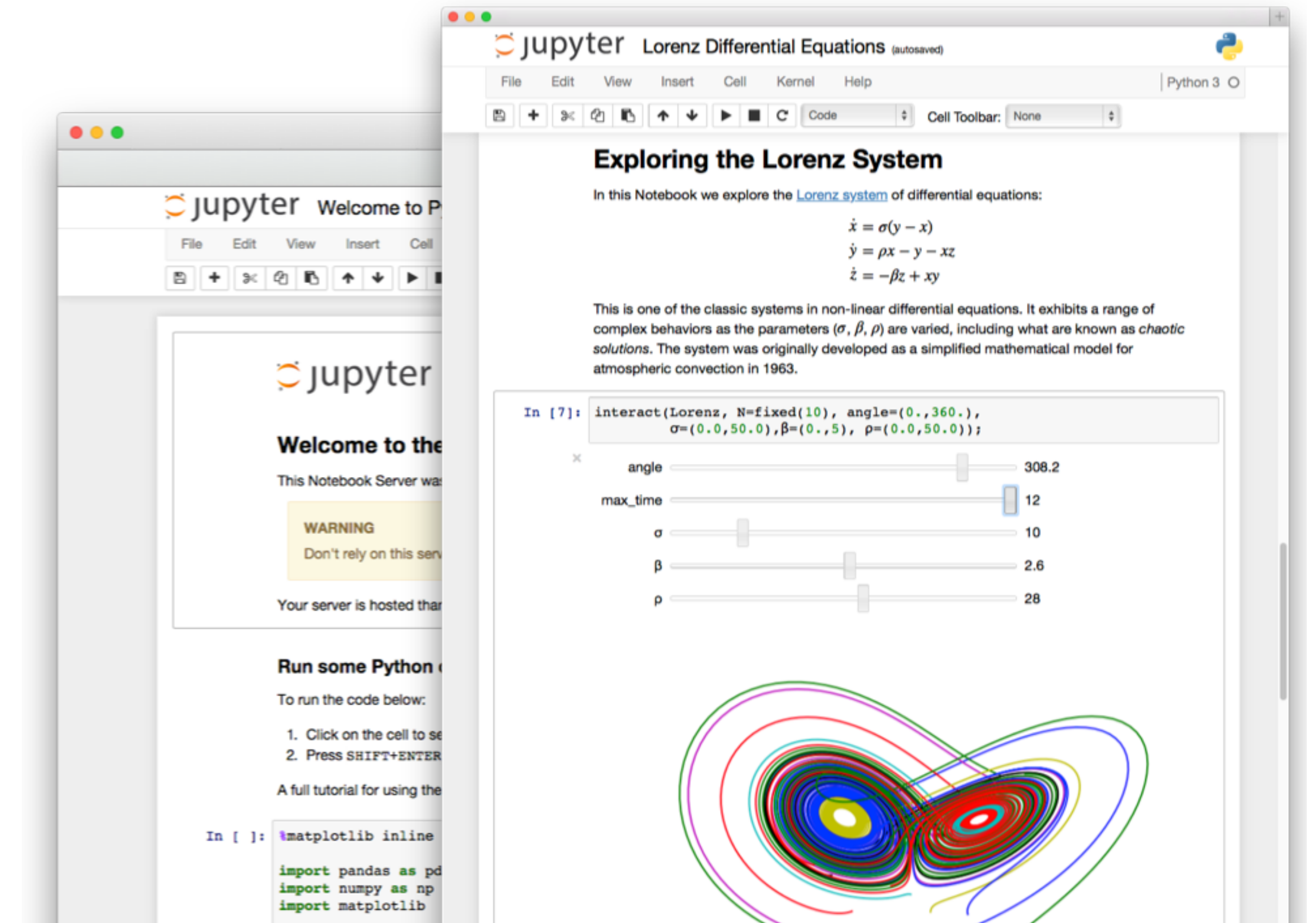
The bottom-left pane, labeled 'Output View', displays a 3D plot of the Lorenz attractor, which is a complex, swirling trajectory in a 3D space. Above the plot are three sliders for parameters: 'sigma' (set to 10.00), 'beta' (set to 2.67), and 'rho' (set to 28.00). The bottom-right pane shows the 'lorenz.py' file with the following Python code:

```
9 def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=8./3, rho=28.0):
10     """Plot a solution to the Lorenz differential equations."""
11     fig = plt.figure()
12     ax = fig.add_axes([0, 0, 1, 1], projection='3d')
13     ax.axis('off')
14
15     # prepare the axes limits
16     ax.set_xlim((-25, 25))
17     ax.set_ylim((-35, 35))
18     ax.set_zlim((5, 55))
19
20     def lorenz_deriv(x_y_z, t0, sigma=sigma, beta=beta, rho=rho):
21         """Compute the time-derivative of a Lorenz system."""
22         x, y, z = x_y_z
23         return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]
24
25     # Choose random starting points, uniformly distributed from -15 to 15
26     np.random.seed(1)
27     x0 = -15 + 30 * np.random.random((N, 3))
28
```

[JupyterLab Documentation]

Jupyter Notebooks

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



[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 3.10
- 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=j6VSAAsKAj98>
- Our in-class exploration:
 - Don't focus on the syntax
 - Focus on how interactive Python makes this exploration work well