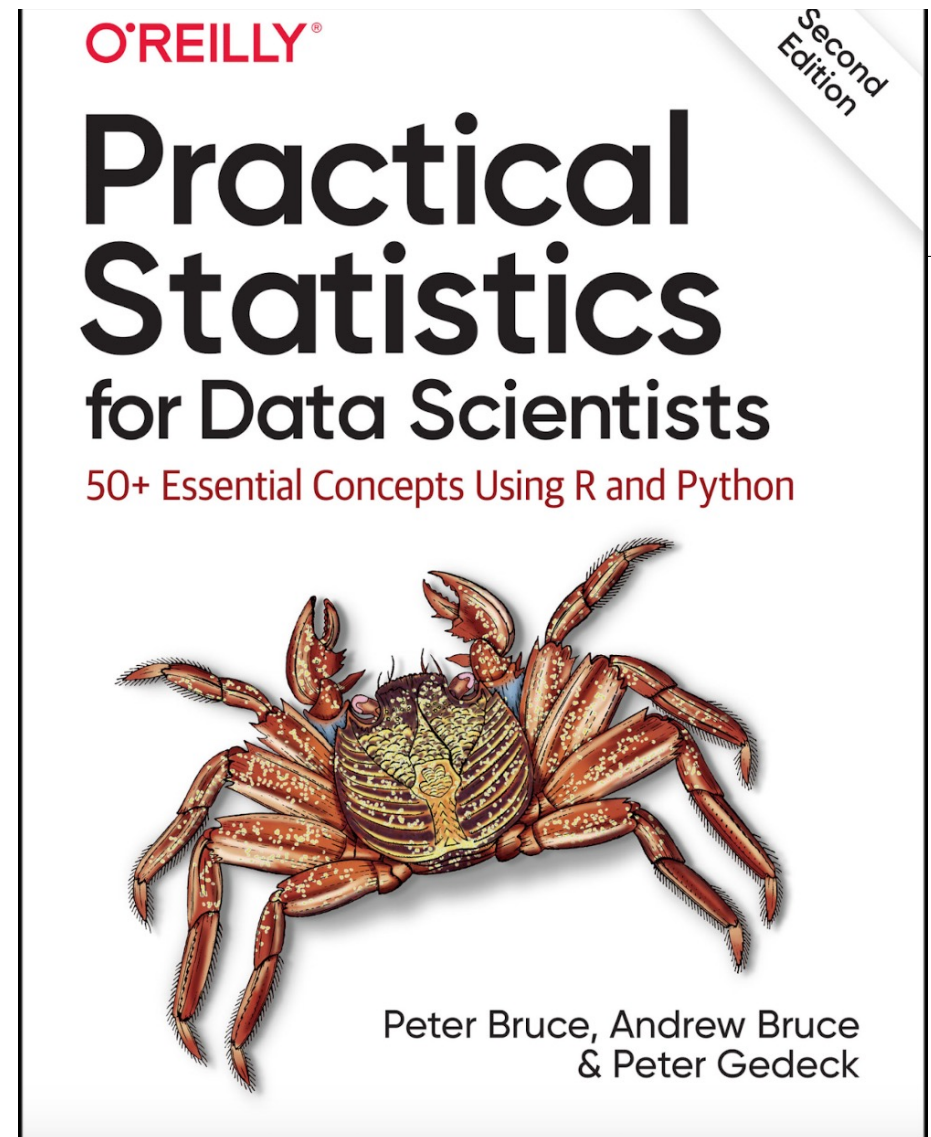
A series of thin, black, intersecting lines of various orientations and lengths, creating a complex, abstract geometric pattern in the upper left quadrant of the slide.

CSCI 692: LECTURE 1 NUMPY AND DATA FRAMES

Professor David Harrison

TODAY

- Remainder of Chapter 1
- Why Python is slow
- Why Python at all?
- Numpy
- Data Frames
- Review:
 - Central Tendency
 - Dispersion





HOMework 1

Posted last Friday.

Due Thursday 11 pm.

Focuses on

- setting up accounts,
- using github and Databricks
- Notebooks.

Submission:

- Submit archived Databricks Notebook to Blackboard.



OFFICE HOURS

Due to scheduling conflict, office hours updated

Tuesday	4:00-5:00 PM
Wednesday	12:30-2:30 PM

.

BLACKBOARD

All lecture slides, homeworks, and solutions will appear on blackboard.

The screenshot shows the Blackboard Ultra user interface. At the top, a browser address bar displays the URL `blackboard.olemiss.edu/ultra/courses/_121946_1/cl/outline`. Below the address bar is a navigation bar with icons for play, stop, left, right, up, down, back, enter, fling, and add_script. The main content area has a header with the course title "Csci 443 Advanced Data Science Section 1 2023-2024 SPRG" and a "Home Page" link. A dark sidebar on the left contains a search icon, a close button (X), and a list of course modules: "Csci 443 Advanced Data Science Section 1 2023-2024 SPRG", "Home Page", and "Announcements". The main content area displays "Home Page" with a dropdown arrow, followed by "Add Course Module". At the bottom, there is a "Mv Announcements" section with a dropdown arrow, a settings gear icon, and a close button (X).

GITHUB

Example files I create during class will be put on github.

The project is at

https://github.com/dosirrah/CSCI443_AdvancedDataScience

You will need to create a Github account independent of your olemiss accounts.

GitHub is free for our purposes.

I highly recommend committing any code you create to GitHub.

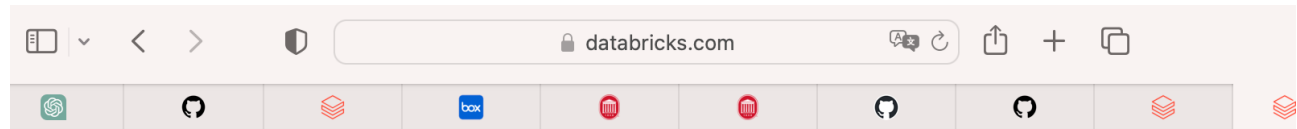
20XX

The screenshot shows the GitHub interface for a repository named 'CSCI443_AdvancedDataScience' by user 'David Harrison'. The repository is public. At the top, there are buttons for 'Pin', 'Unwatch' (with a count of 1), and a fork icon. Below this, there's a 'main' branch selector and icons for repository actions. A file list shows '.gitignore', 'CSCI443 Syllabus.pdf', and 'README.md'. A 'Clone' dropdown menu is open, showing options for 'Local' and 'Codespaces'. Under 'Local', there are tabs for 'HTTPS', 'SSH' (which is selected and underlined), and 'GitHub CLI'. The SSH option shows a text input field with the URL 'git@github.com:dosirrah/CSCI443_AdvancedDataScience' and a copy icon. Below the input field, it says 'Use a password-protected SSH key.'

DATABRICKS

We will use the databricks community edition.

<https://community.cloud.databricks.com/login.html>



Try Databricks free

Test-drive the full Databricks platform free for 14 days on your choice of AWS, Microsoft Azure or Google Cloud. Sign-up with your work email to elevate your trial experience.

Create your Databricks account 1/2

Sign up with your work email to elevate your trial with expert assistance and more.

First name

Last name

Email

COMMIT NOTEBOOKS TO GITHUB

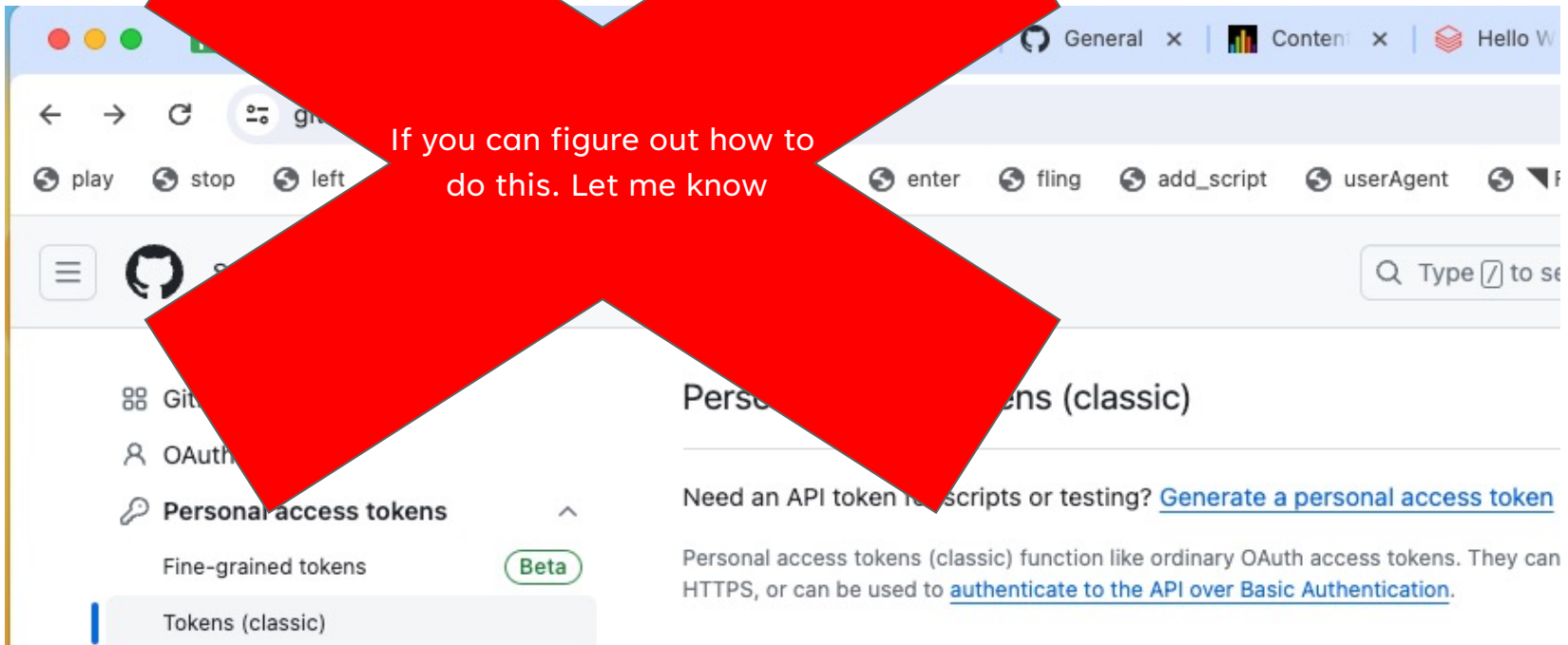
Please connect your notebooks to github.

Checkpoint your work regularly.

Ask chatgpt how

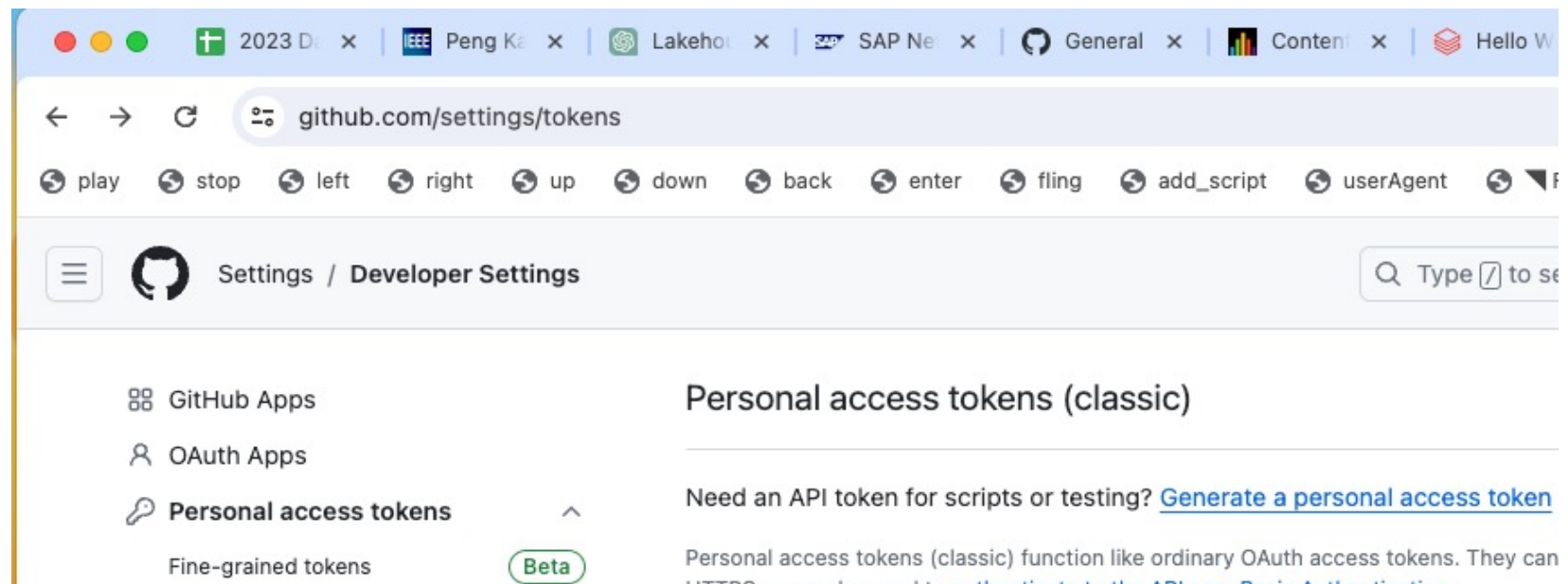
1. On github go to Personal Access Tokens

If you can figure out how to
do this. Let me know



CLARIFICATION GITHUB

You can still commit notebooks to Github, but you have to download it first.
The direct connection between Github and Databricks is not available in the community edition of Databricks.





WHY IS PYTHON SLOW?

Everything in Python is a PyObject.

Everything in Python is allocated from dynamic memory

- i.e., from the memory heap.

Python uses duck-typing:

- makes inlining often impossible

Python command-line is interpreted

- .py files are just-in-time compiled (faster, but still slow)

Python is single threaded

- Global Interpreter Lock (GIL)

Python does not natively make use of vector processing.



WHY USE PYTHON FOR DATA SCIENCE?

Python is easy to program.

Python has beautiful, terse syntax.

Python is well supported by tools.

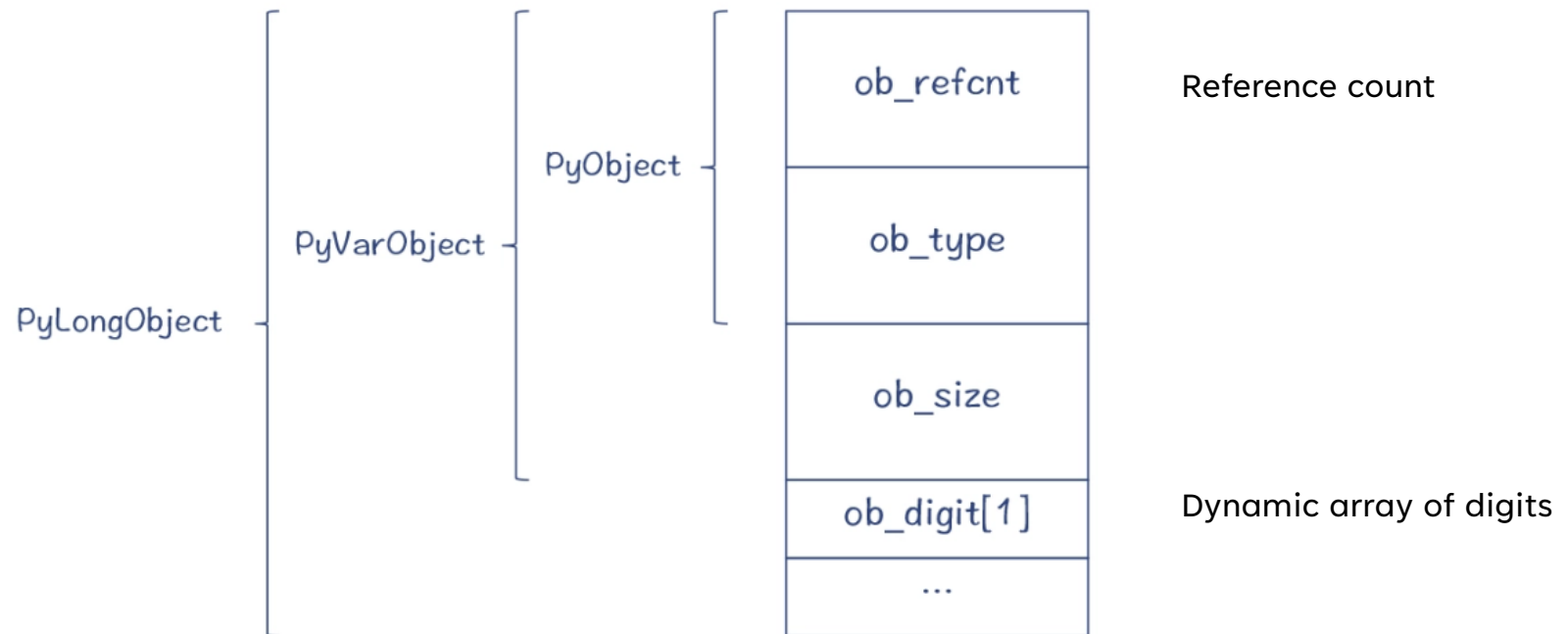
The big stuff is usually not done in Python, so slow doesn't matter (usually):

- Apache Spark
- PyTorch
- TensorFlow

The medium-sized stuff and postprocessing is usually not done in Python either

- NumPy
- DataFrames
- SciPy

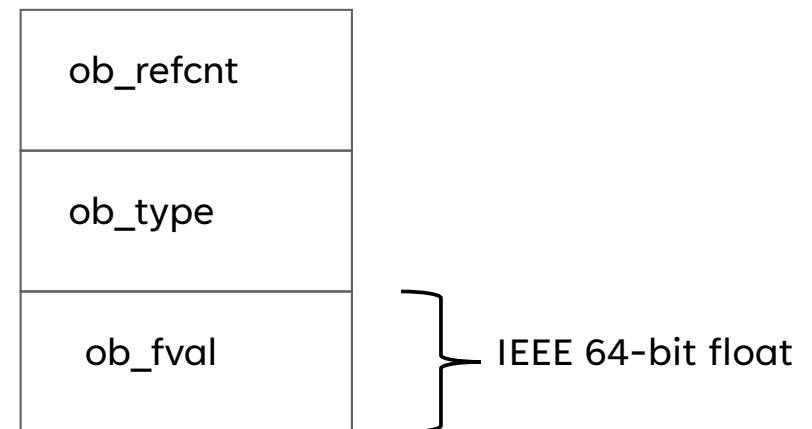
PYTHON INTEGERS ARE OBJECTS



PYTHON FLOATS ARE ALSO OBJECTS

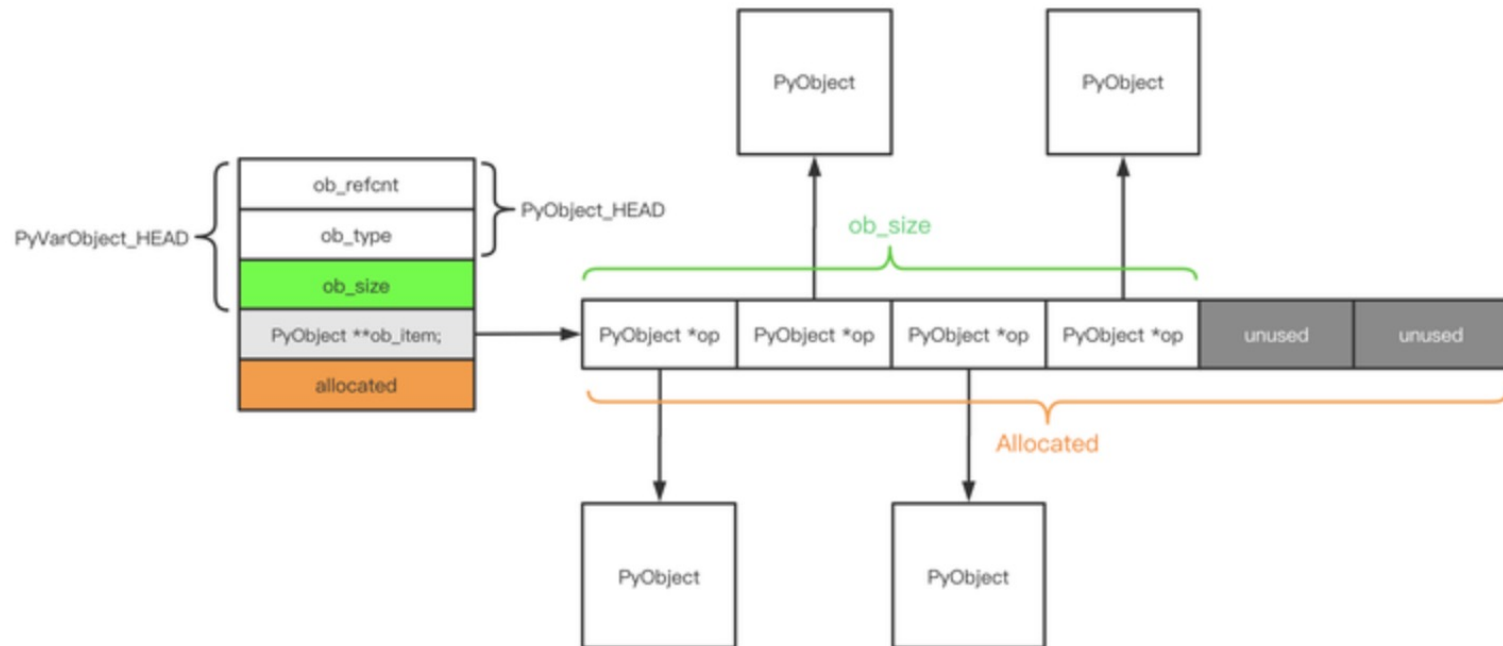
A float in python is an IEEE 64-bit floating point value wrapped in a PyObject.

- Compared to floating point arithmetic:
 - **Integer math is faster for small integers,**
 - **but slower for large integers.**



PYTHON LISTS ARE DYNAMIC ARRAYS

A list is a dynamic array of pointers to PyObjects.





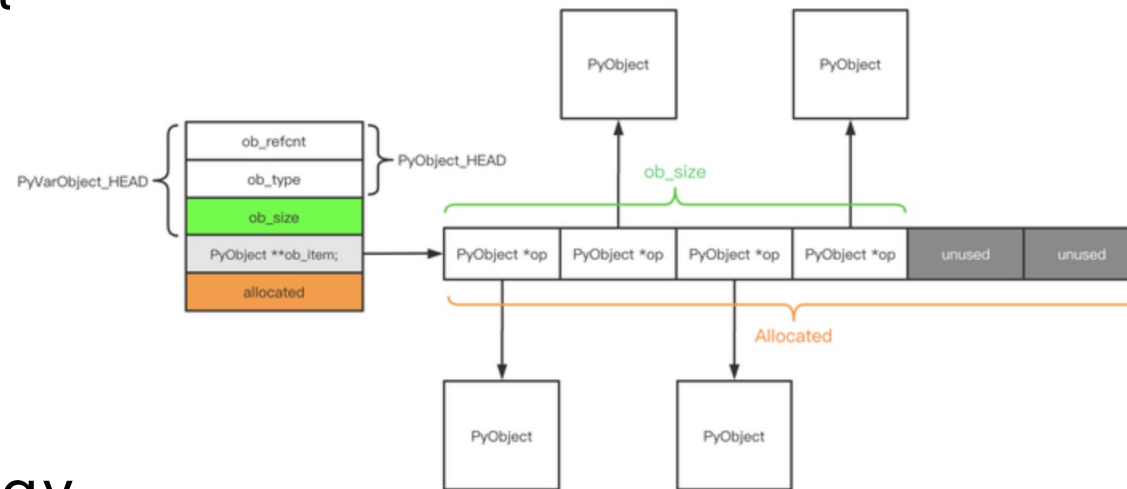
WHY NUMPY?

NumPy provides

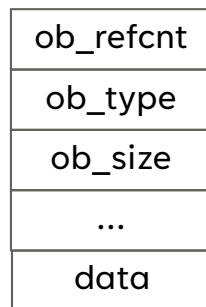
- large, memory-efficient, multi-dimensional arrays
- Fixed size integers and floats without
 - reference count field
 - type field
 - object size field
- Array and matrix operations
 - Utilizes vector operations when supported by the hardware. Thus FAST.

PYTHON LIST VS. NUMPY ARRAY

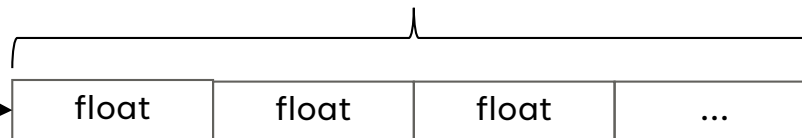
Python list



Numpy array

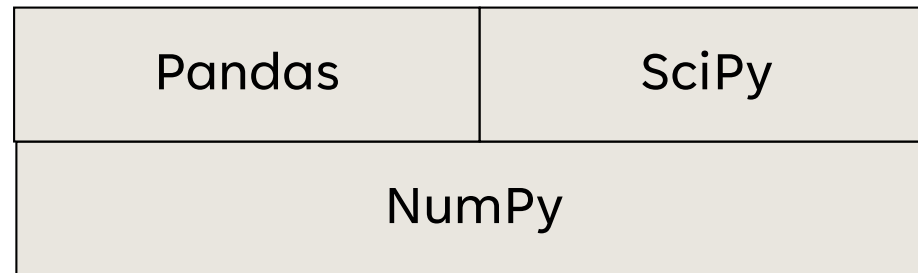


Contiguous memory of fixed length 64 bit floating point numbers. Unlike PyObjects, each float has no reference count, type or size fields





MANY OTHER LIBRARIES BUILD ON NUMPY



EXAMPLE 1: ADD CONSTANT TO ARRAY

Python version

```
array = [1, 2, 3, 4, 5]
new_array = [0] * len(array)
for i in range(len(array)):
    new_array[i] = array[i] + 5
```

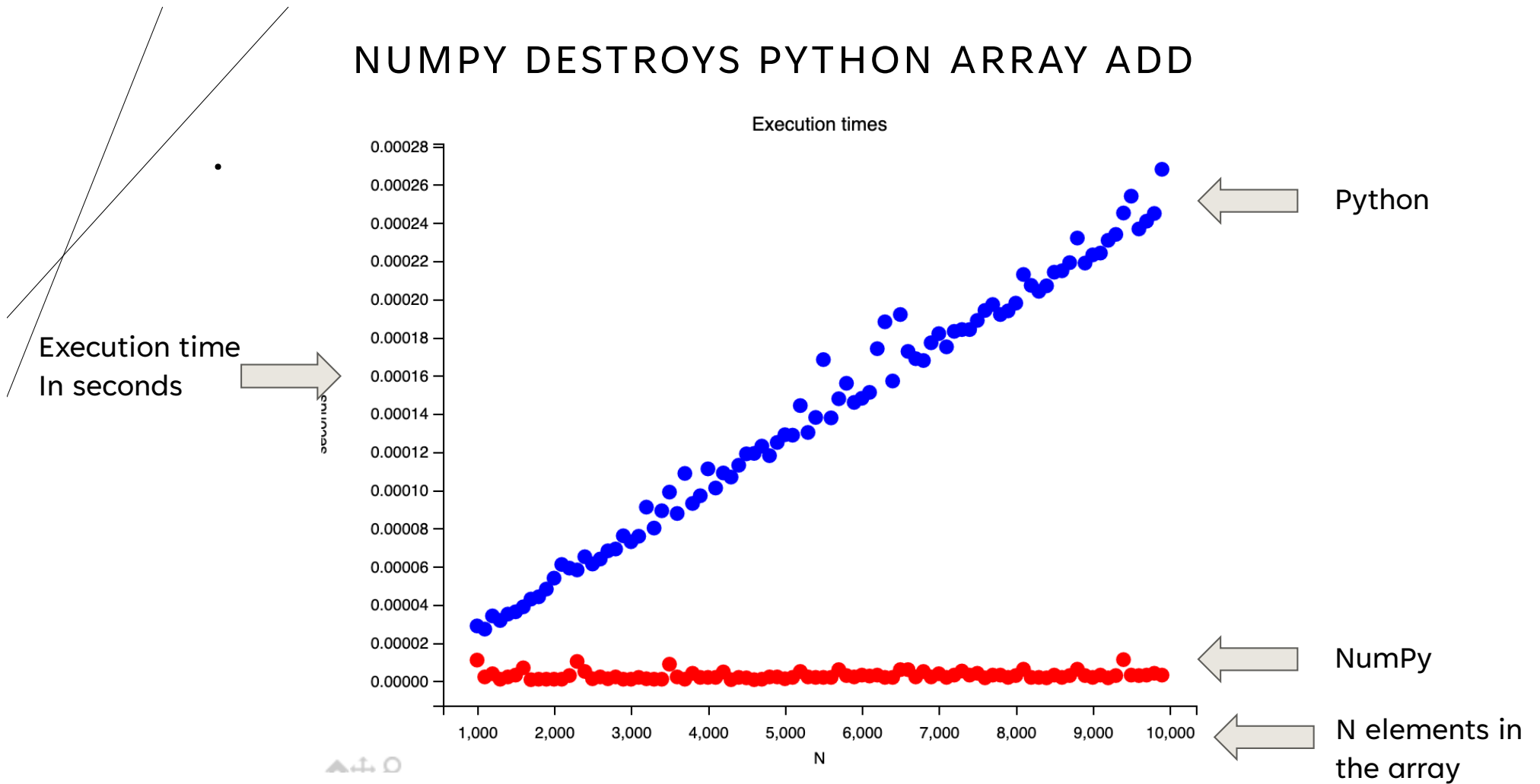
Both Python and NumPy versions can be found in the github class repository at [lecture02/example1/example_1_array_add.py](#)

Python with NumPy version

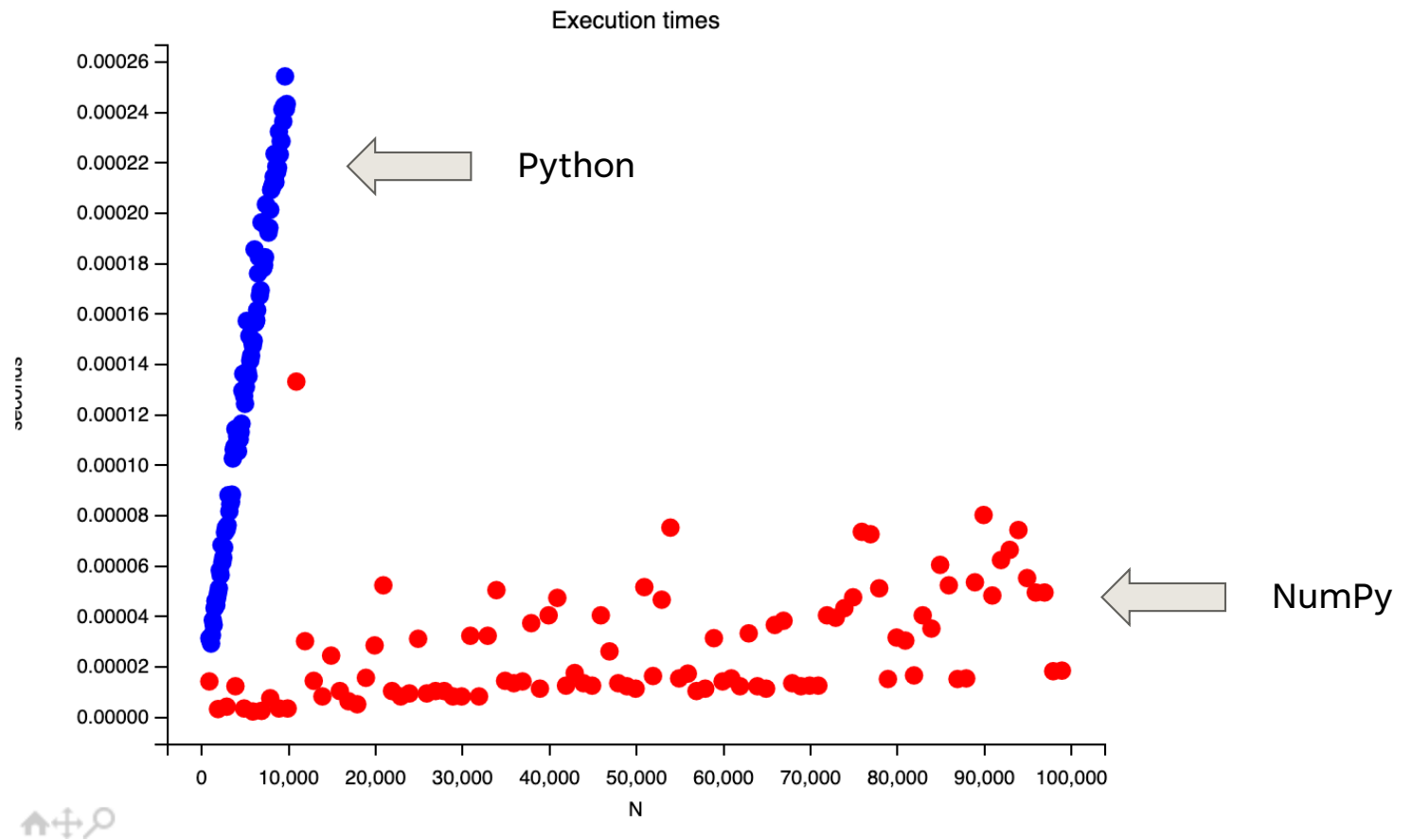
```
# Create a one-dimensional NumPy array
array = np.array([1, 2, 3, 4, 5])

# Add constant to all elements of the array
new_array = array + 5
```

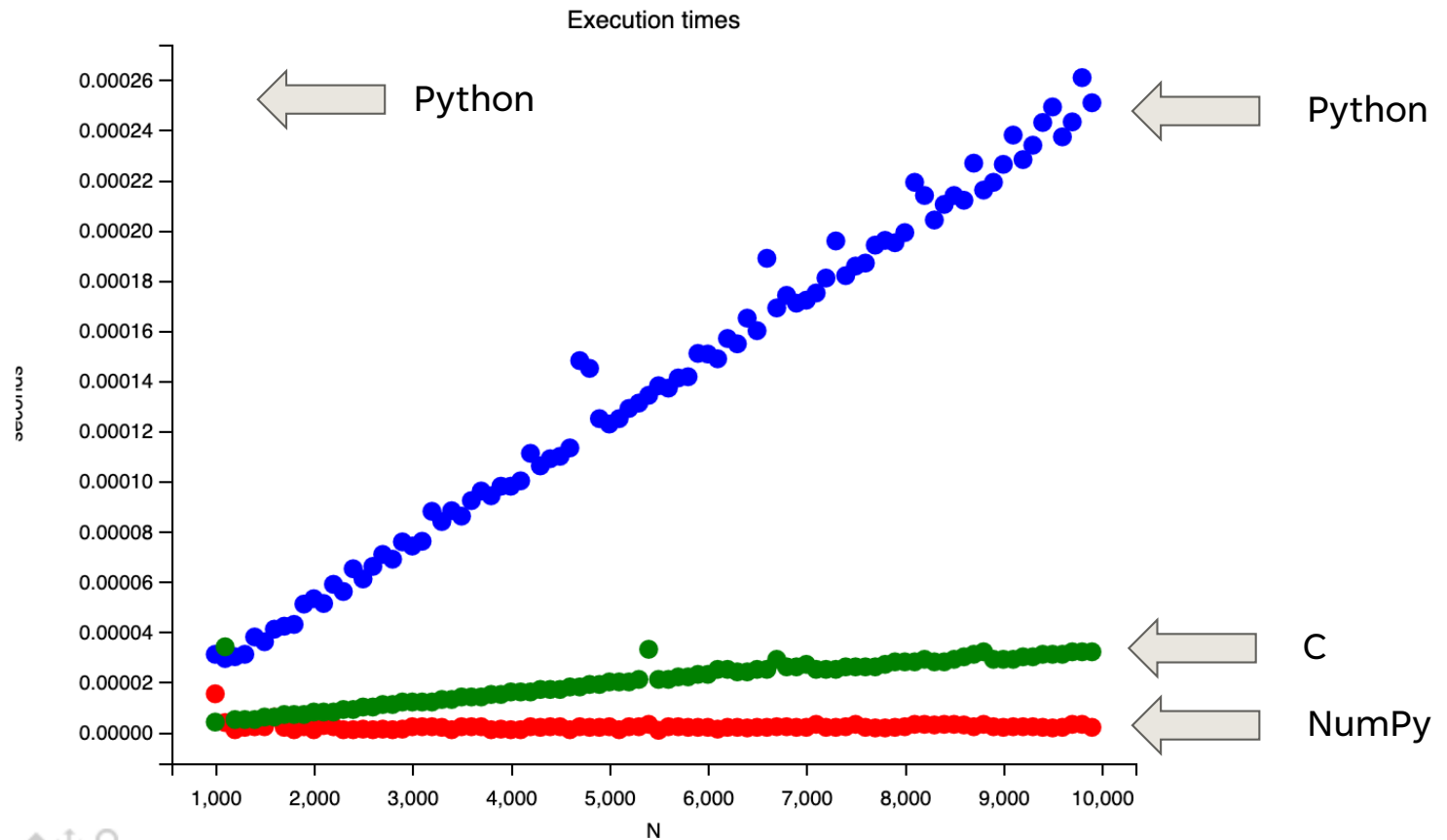
NUMPY DESTROYS PYTHON ARRAY ADD



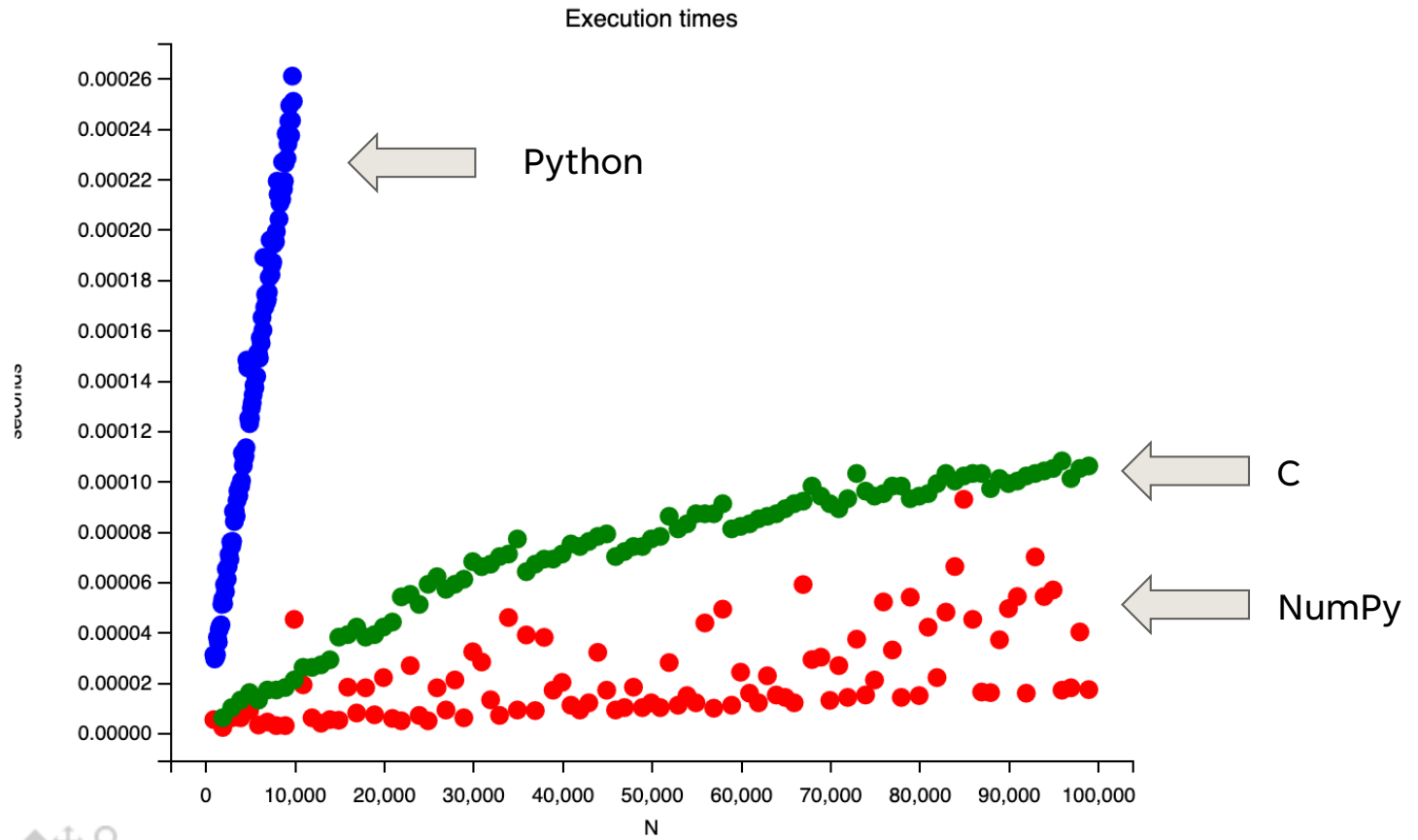
NUMPY DESTROYS PYTHON ARRAY ADD



FOR KICKS COMPARE TO C WITHOUT VECTOR OPERATIONS



FOR KICKS COMPARE TO C WITHOUT VECTOR OPERATIONS



EXAMPLE 2: MATRIX MULTIPLICATION

Python version

```
A = [[random.rand() for _ in range(n)] for _ in range(n)]  
B = [[random.rand() for _ in range(n)] for _ in range(n)]
```

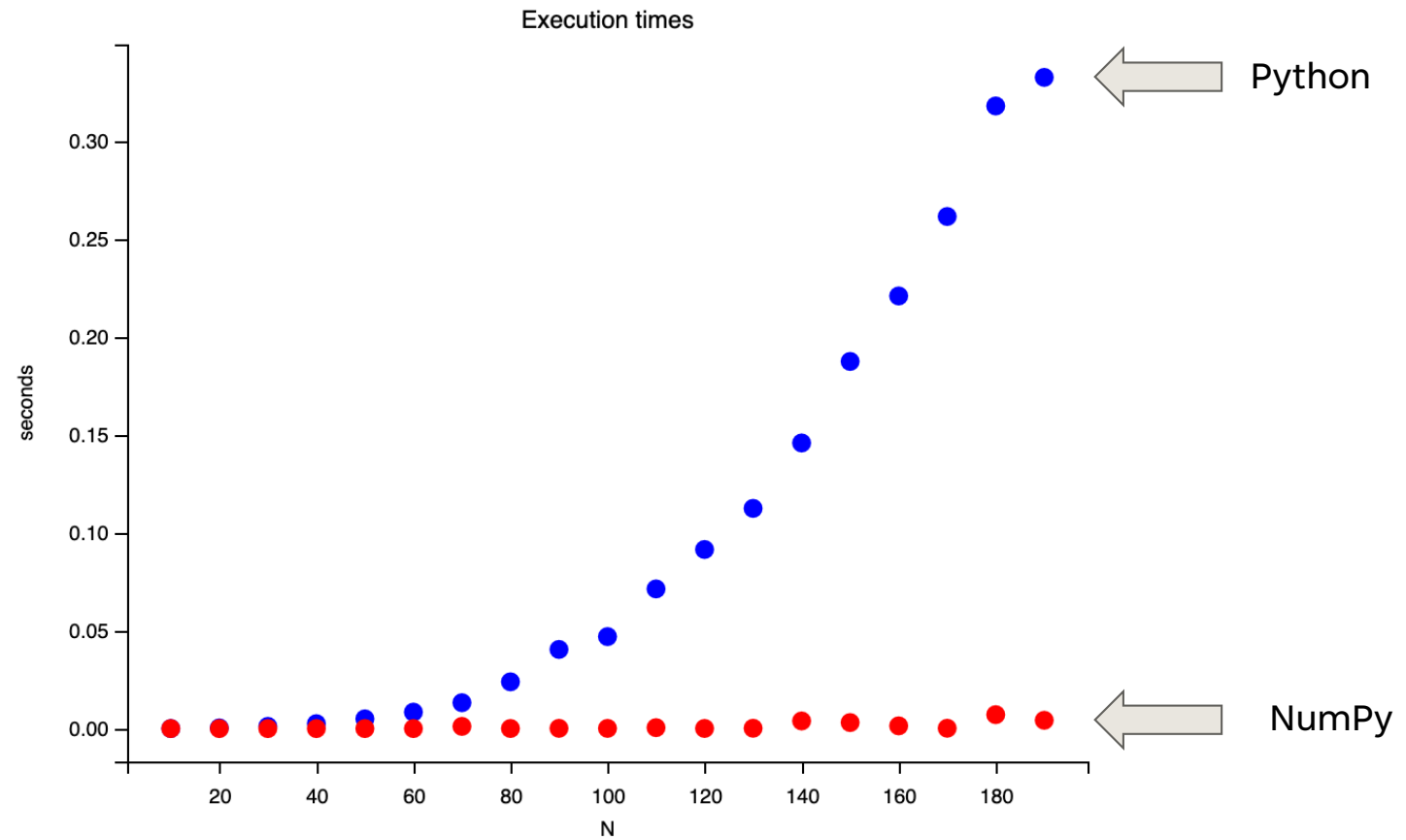
```
for i in range(n):  
    for j in range(n):  
        C = sum(A[i][k] * B[k][j] for k in range(n))
```

Python with NumPy version

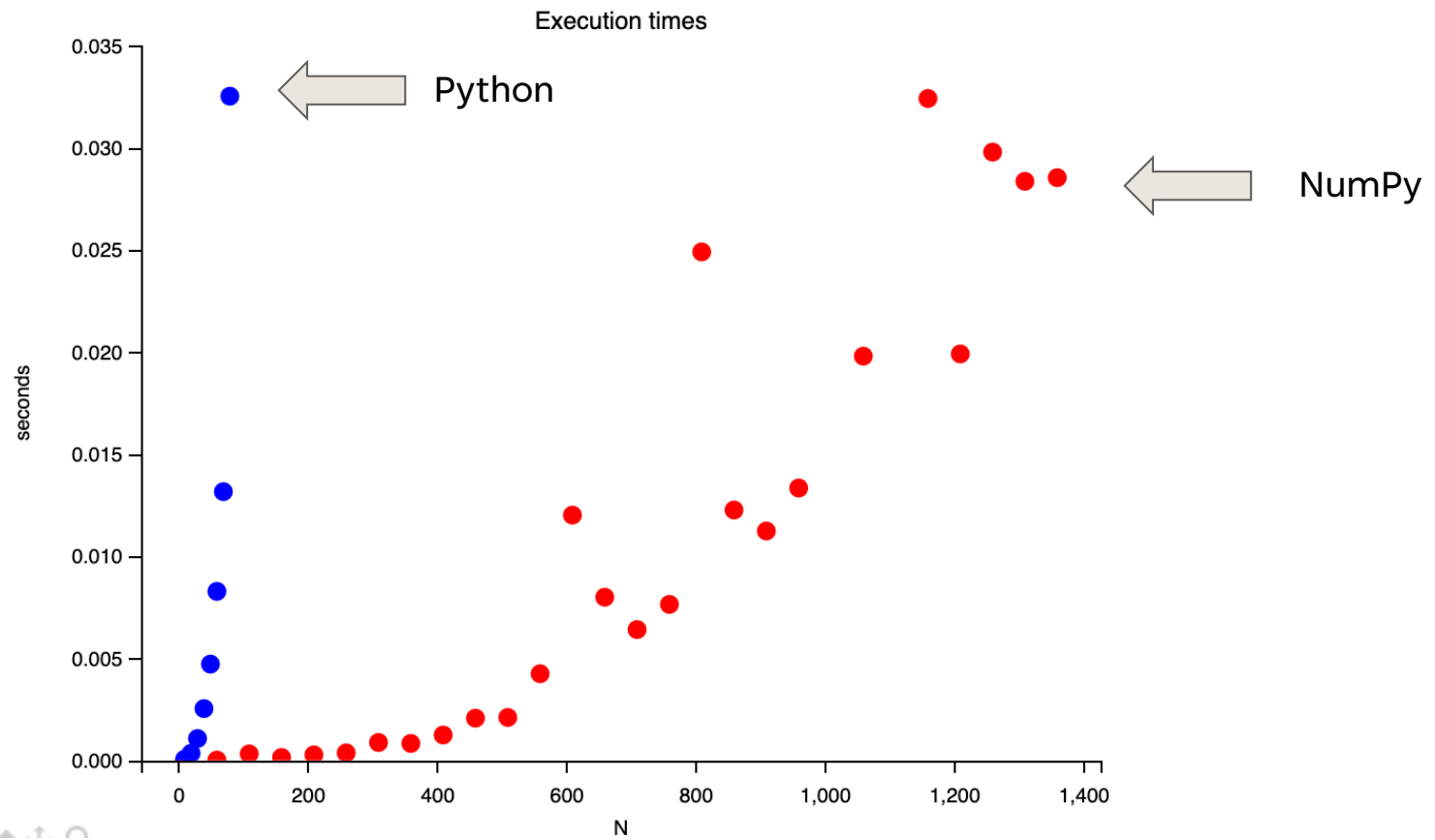
```
A_np = np.random.rand(n, n)  
B_np = np.random.rand(n, n)
```

```
C_np = np.dot(A_np, B_np)
```

NUMPY DESTROYS PYTHON MATRIX MULT



NUMPY DESTROYS PYTHON MATRIX MULT





THANK YOU

David Harrison

Harrison@cs.olemiss.edu