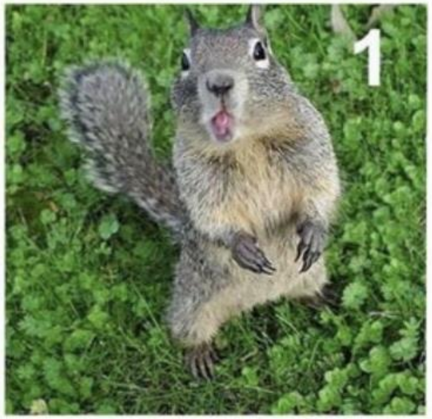
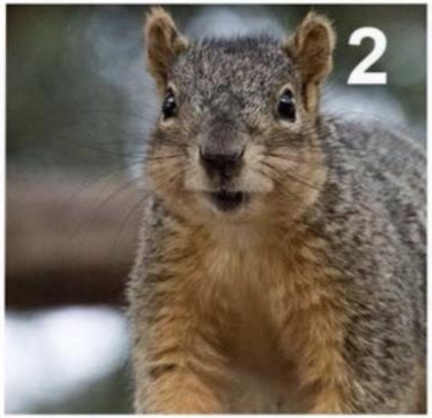
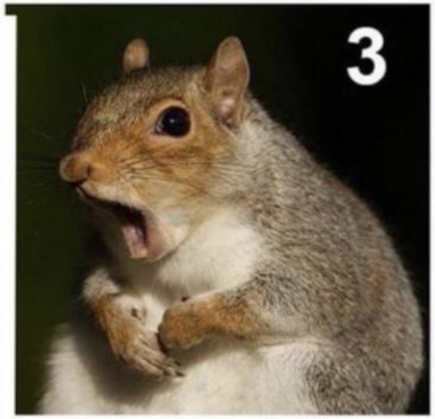
# Computer Programming for Lawyers

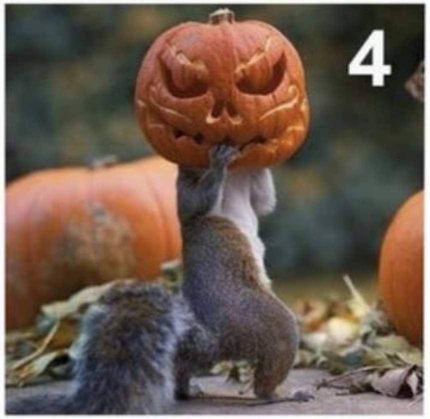
**Prof. Rachel Orey**

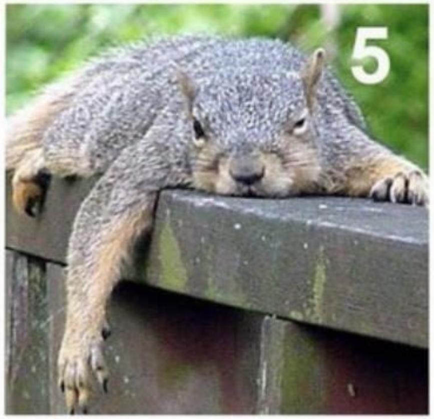
Spring 2025

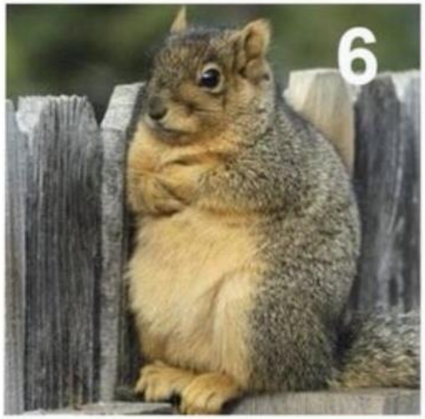

1

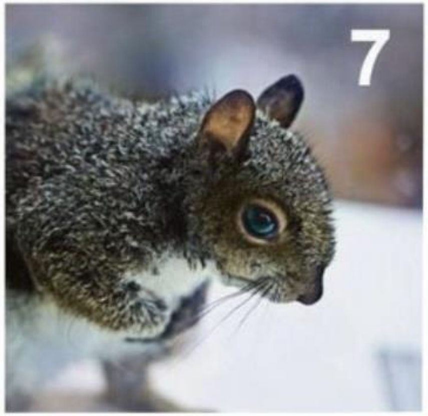

2

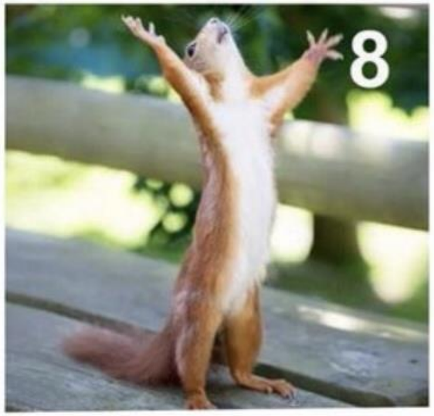

3

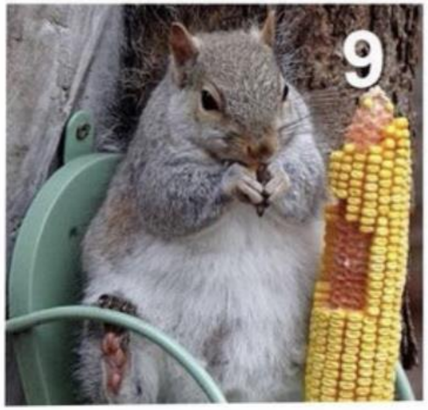

4

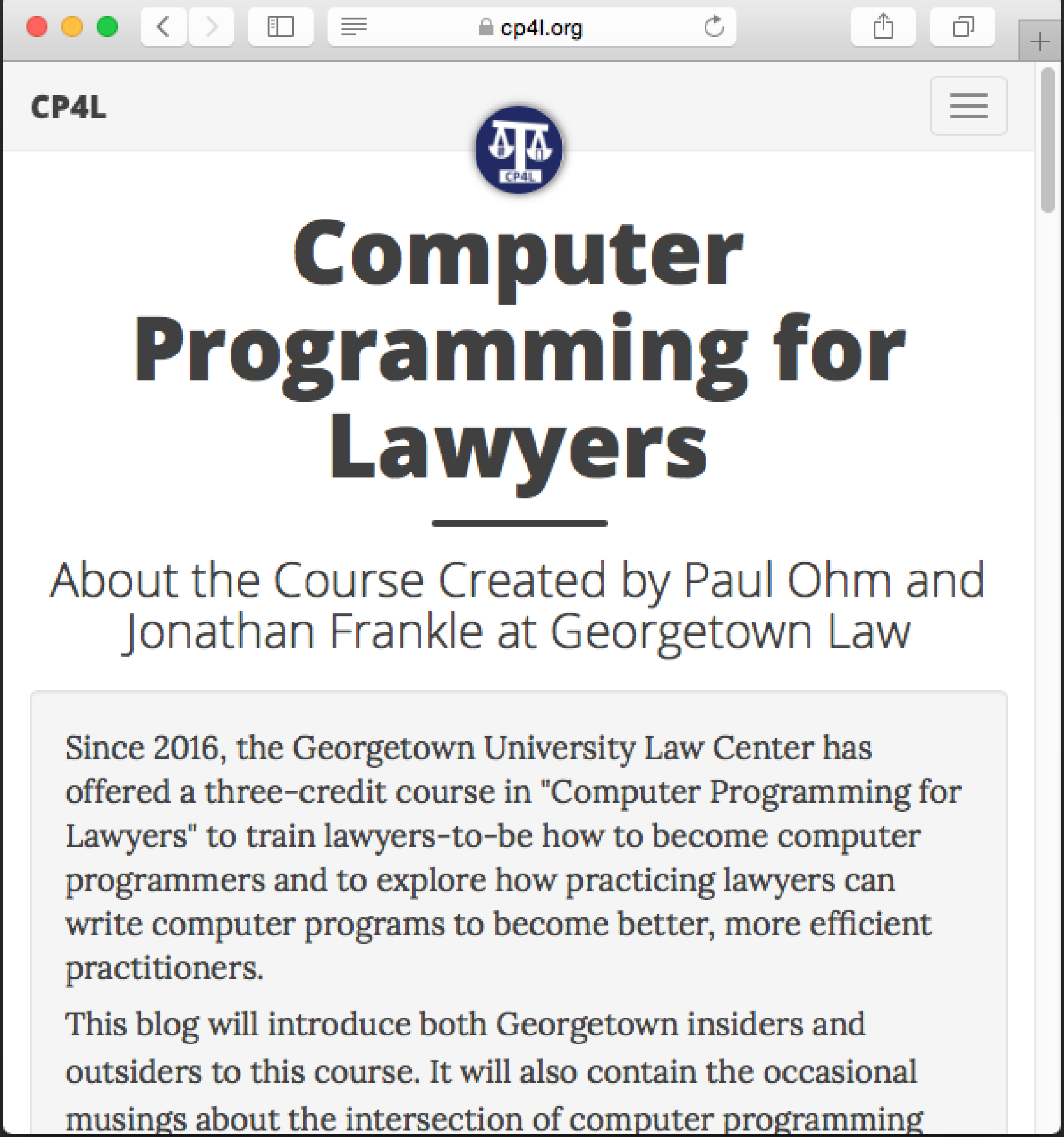

5


6


7


8


9



# Computer Programming for Lawyers

## About the Course Created by Paul Ohm and Jonathan Frankle at Georgetown Law

Since 2016, the Georgetown University Law Center has offered a three-credit course in "Computer Programming for Lawyers" to train lawyers-to-be how to become computer programmers and to explore how practicing lawyers can write computer programs to become better, more efficient practitioners.

This blog will introduce both Georgetown insiders and outsiders to this course. It will also contain the occasional musings about the intersection of computer programming

# Course philosophy

- Lawyers process information, but use inefficient tools for the job
- Coding is an emerging legal skill, like writing and research
    - Automate the ‘Boring Stuff’
    - Work at Scale
    - Have Fun!

# Goals and side-effects

- Make you more efficient in ordinary legal practice
  - Text manipulation, search, summary
  - Document manipulation (pdf, docx, xlsx)
  - File manipulation
  - Application Programming Interfaces (APIs)
  - Scraping
  - Knowing how to work with AI to do… anything?!

# What this class is not

- A law class.
- A class on cybersecurity, the internet, or any other specific technology.
- A math-heavy class.

**This is a <u>skills</u> class, plain and simple.**

# Beginners Only!

- Ineligible if you have:
    - Taken a programming class in college or graduate school.
    - Mastered any programming language.
- Must consult with instructors if you are unsure if you are eligible.

# Waitlisters

- You must go to one lab on Wednesday to get off the list.
- If you don’t submit PS1 and PS2 before add/drop deadline, you will be removed from the class.

# TAs for next year!

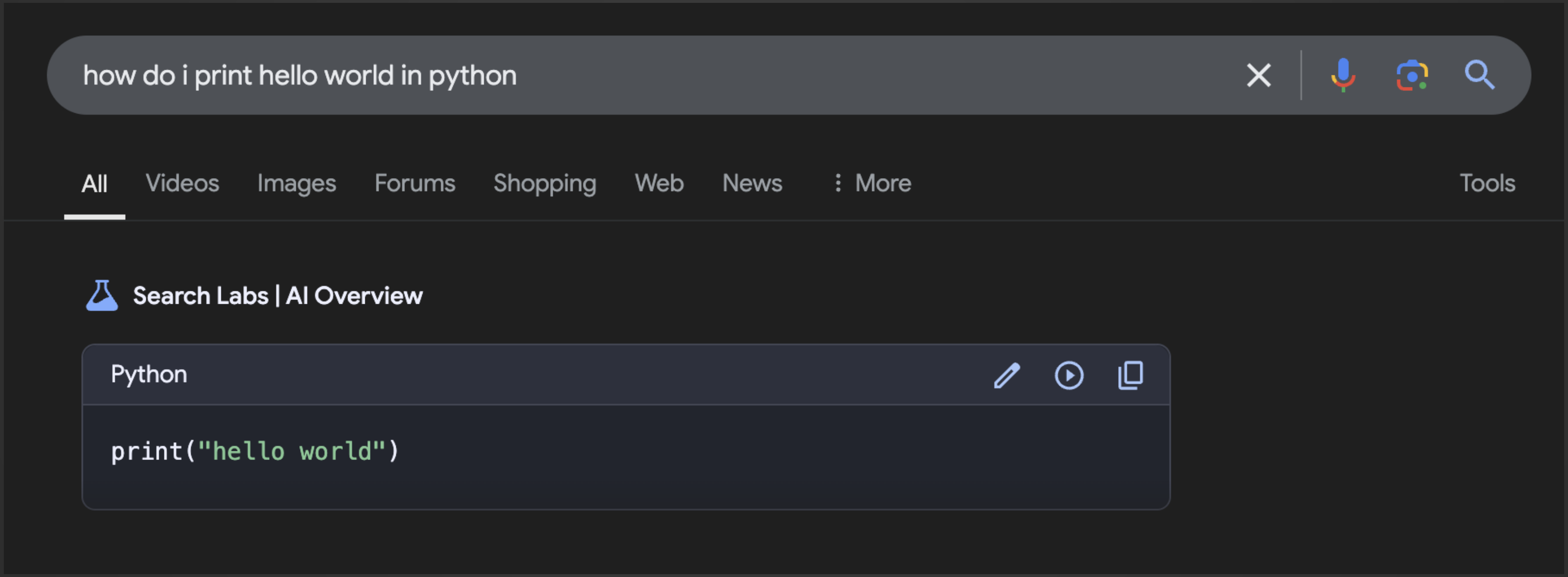
- If you’re not graduating this semester, we’re on the lookout for TAs for next spring
- Intermediate “course“ with me and Prof. Ohm + grading + labs

# Course Logistics

# Policy on use of AI

| | Problem sets 1–5 | Problem sets 6-8 |
|---|---|---|
| **Level of detail in instructions** | High | Low |
| **How you can use AI** | Only to ask general questions. No copying and pasting to or from an AI. Using AI is not necessary for these problem sets. | You will be using AI to achieve high-level goals, using skills that you will not have been taught. |
| **Prior knowledge required to be successful** | Nothing! | A deep understanding of the concepts covered in problem sets 1–5 |

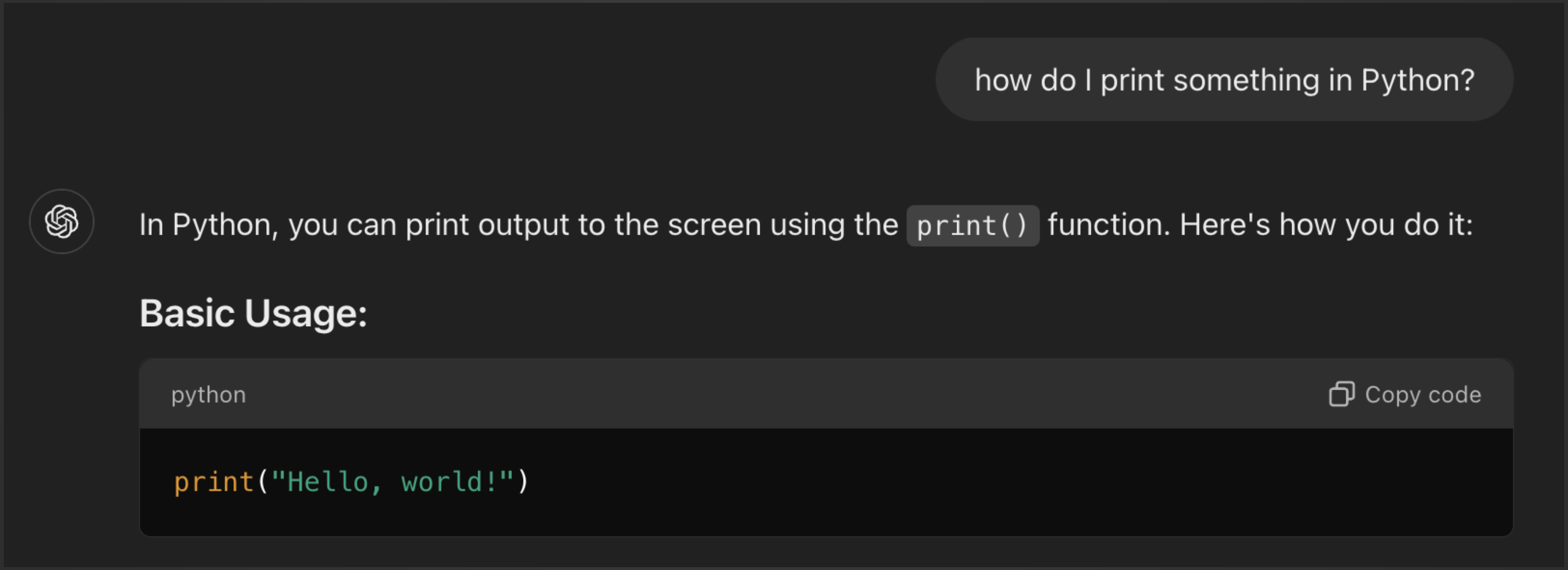
# AI-augmented search okay to use



how do i print hello world in python
All
Videos
Images
Forums
Shopping
Web
News
More
Tools
Search Labs | AI Overview
Python

```
print("hello world")
```

# Asking AI general questions okay, too



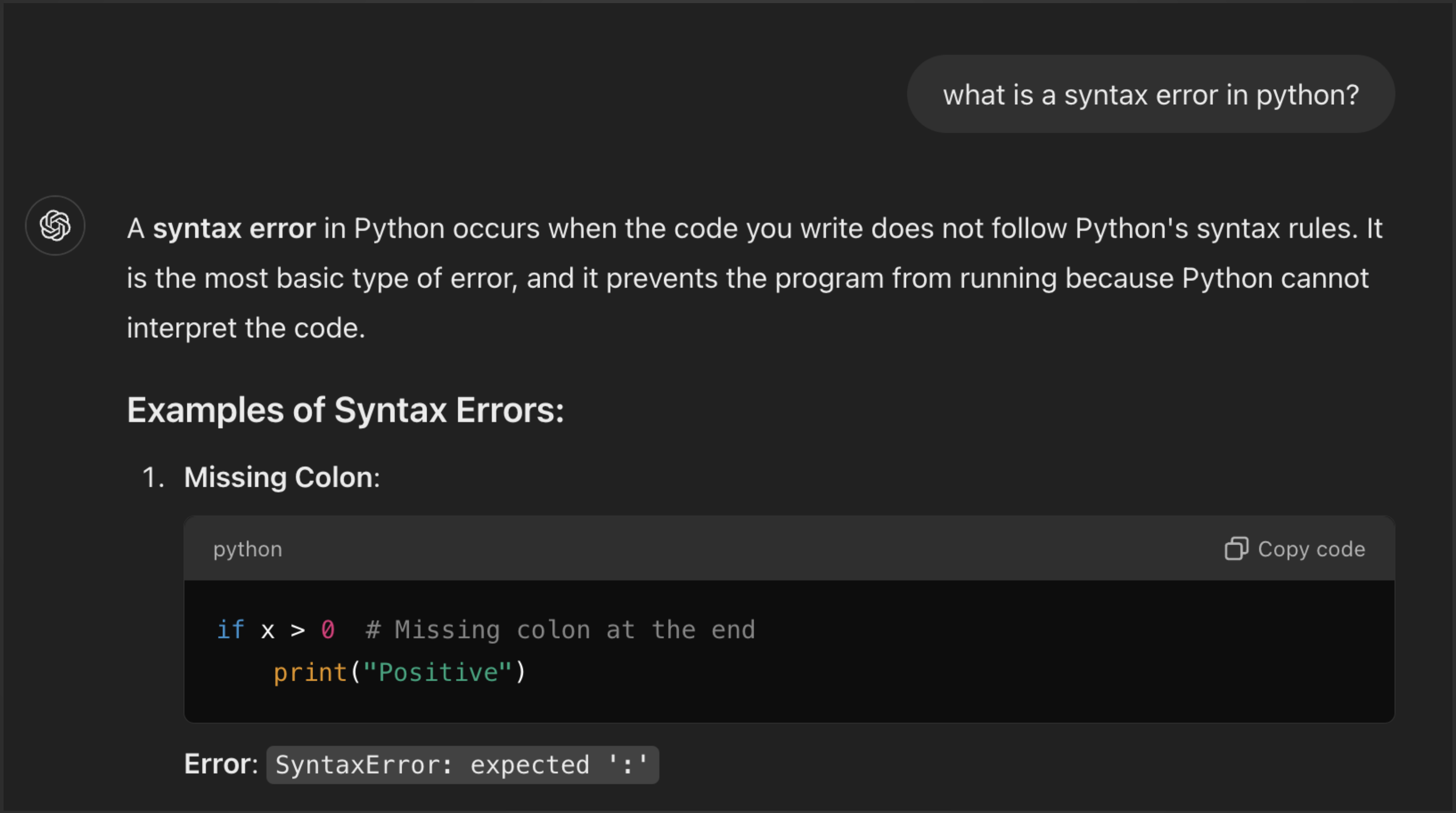
how do I print something in Python?

In Python, you can print output to the screen using the `print()` function. Here's how you do it:

**Basic Usage:**

```python
print("Hello, world!")
```

# Asking AI general questions okay, too



what is a syntax error in python?

A **syntax error** in Python occurs when the code you write does not follow Python's syntax rules. It is the most basic type of error, and it prevents the program from running because Python cannot interpret the code.

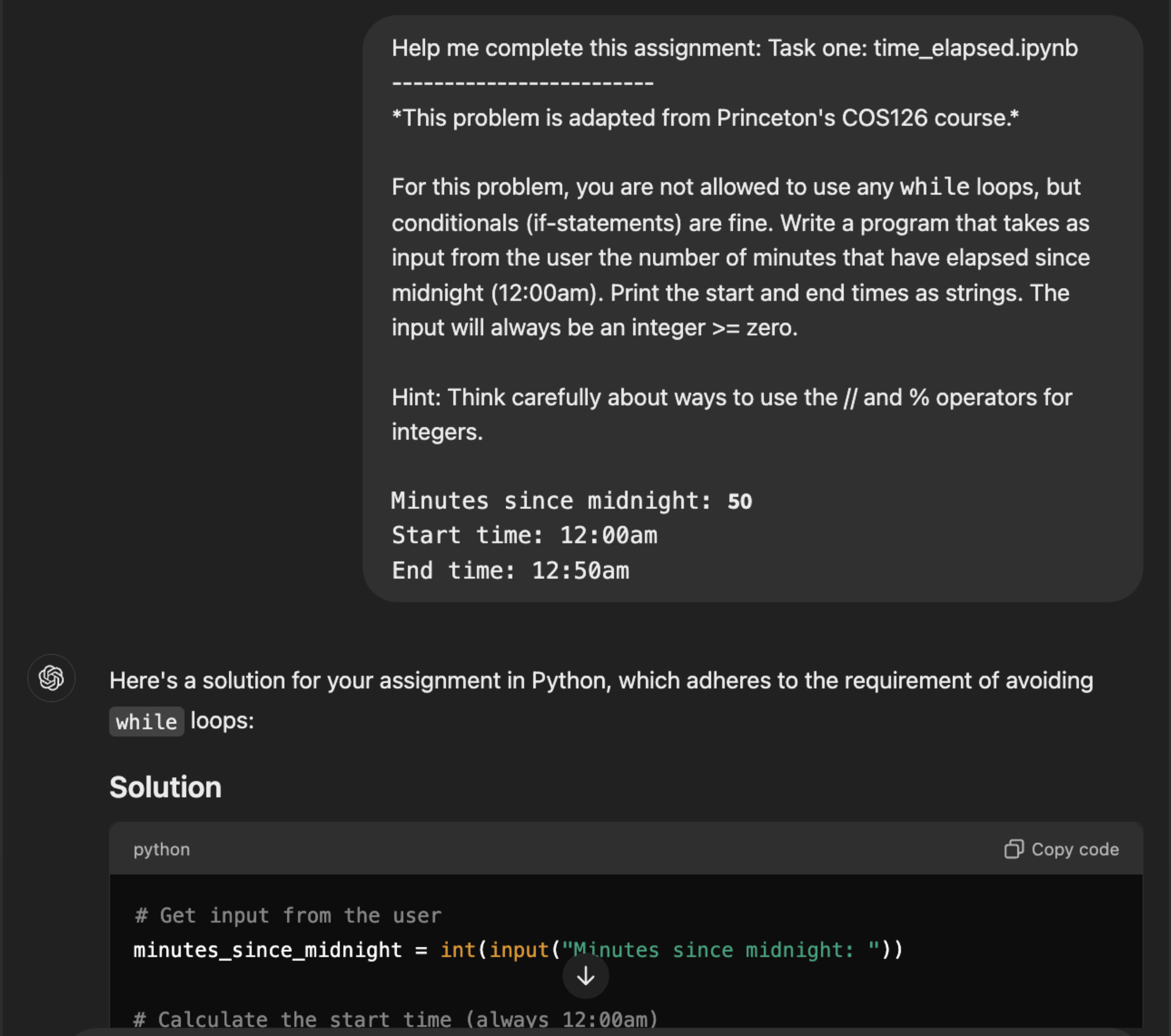
### Examples of Syntax Errors:

1. **Missing Colon**:

```python
if x > 0  # Missing colon at the end
    print("Positive")
```

**Error**: `SyntaxError: expected ':'`

**Sharing specific or paraphrased instructions from the problem set is NOT okay**



Help me complete this assignment: Task one: time_elapsed.ipynb
------------------------
*This problem is adapted from Princeton's COS126 course.*

For this problem, you are not allowed to use any `while` loops, but conditionals (if-statements) are fine. Write a program that takes as input from the user the number of minutes that have elapsed since midnight (12:00am). Print the start and end times as strings. The input will always be an integer >= zero.

Hint: Think carefully about ways to use the // and % operators for integers.

```
Minutes since midnight: 50
Start time: 12:00am
End time: 12:50am
```

Here's a solution for your assignment in Python, which adheres to the requirement of avoiding `while` loops:

### Solution

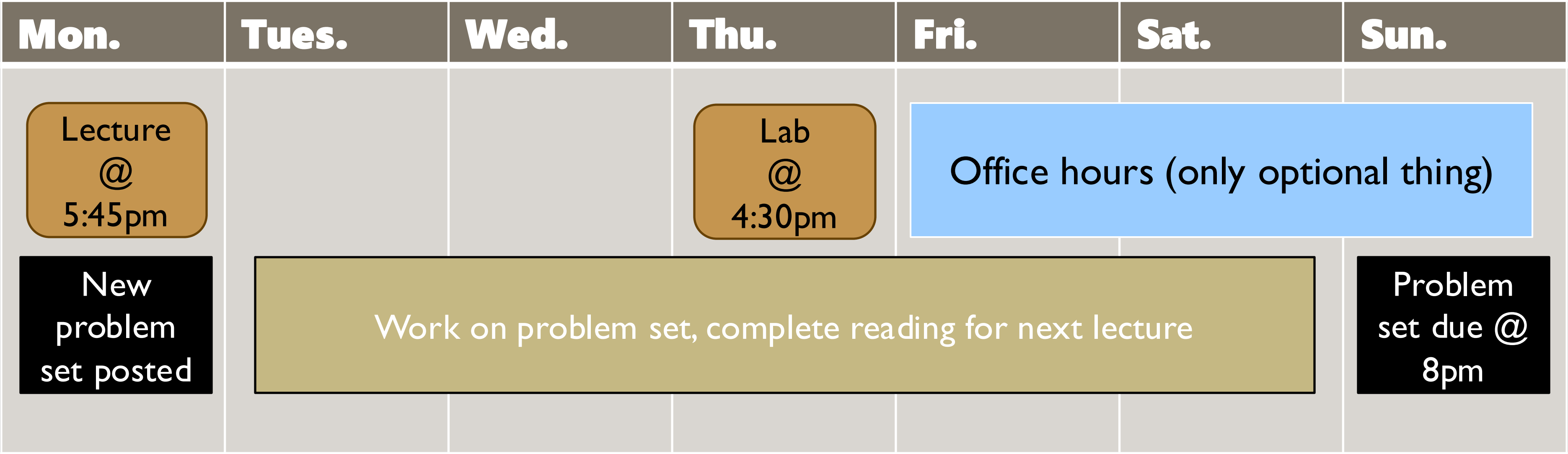
```python
# Get input from the user
minutes_since_midnight = int(input("Minutes since midnight: "))

# Calculate the start time (always 12:00am)
```

# Course Structure

- 8 problem sets. You’ve already done one: PS1
  - Next four: “core” Python.
  - Last three: co-programming with AI.

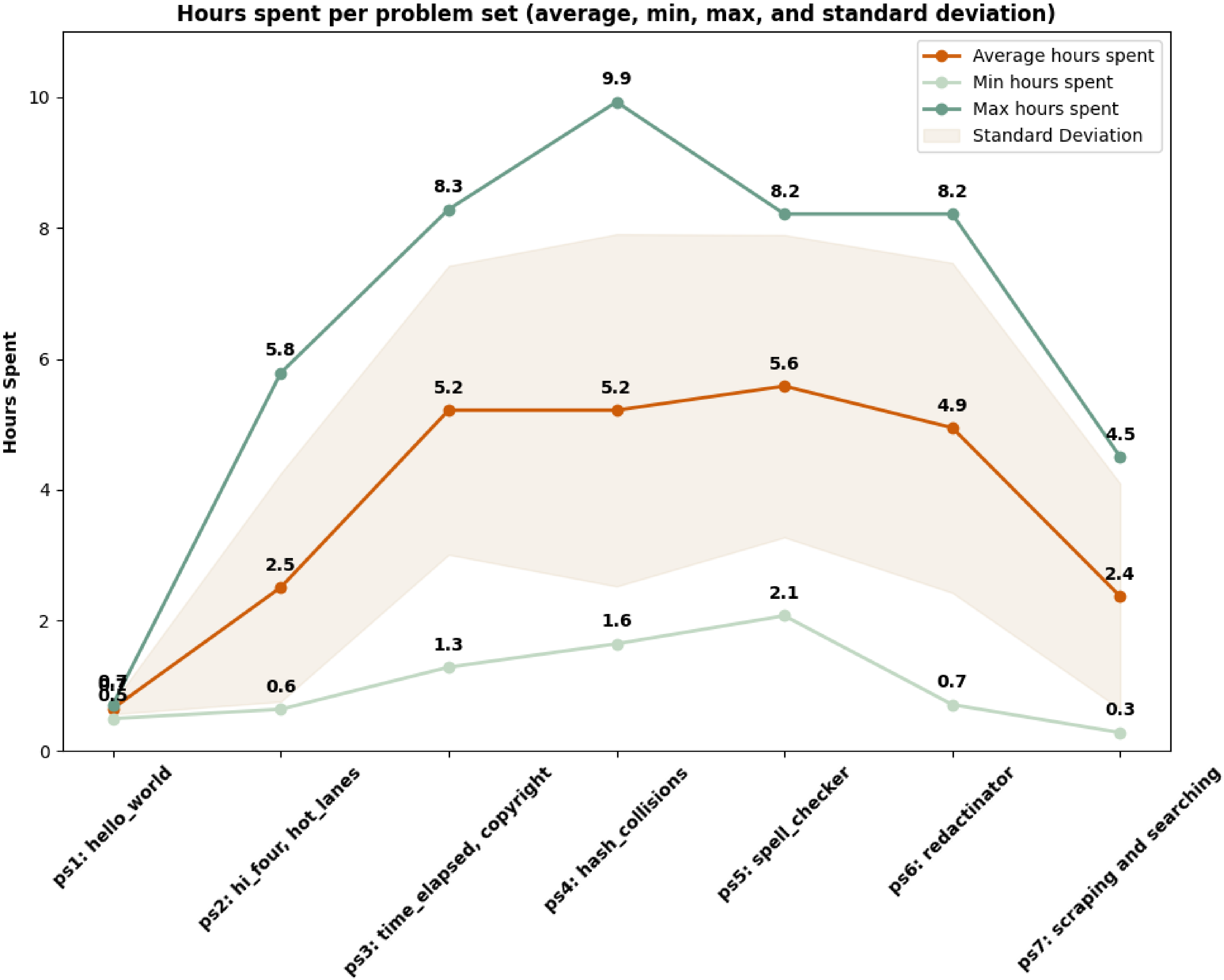
# Weekly Schedule



| Mon. | Tues. | Wed. | Thu. | Fri. | Sat. | Sun. |
|---|---|---|---|---|---|---|
| Lecture @ 5:45pm | | | Lab @ 4:30pm | Office hours (only optional thing) | | |
| New problem set posted | Work on problem set, complete reading for next lecture | | | | | Problem set due @ 8pm |

# Course Readings

- Recommended, generally not required
- Skim to get comfortable with ideas that will be presented during lecture
- Lean on cheat sheets


Hours spent per problem set (average, min, max, and standard deviation)
Average hours spent
Min hours spent
Max hours spent
Standard Deviation
Hours Spent
0
2
4
6
8
10
0.7
0.5
5.8
2.5
0.6
8.3
5.2
1.3
9.9
5.2
1.6
8.2
5.6
2.1
8.2
4.9
0.7
4.5
2.4
0.3
ps1: hello_world
ps2: hi_four, hot_lanes
ps3: time_elapsed, copyright
ps4: hash_collisions
ps5: spell_checker
ps6: redactinator
ps7: scraping and searching

# Why are problem sets difficult?

- The coding-debugging cycle is very different from writing. It’s more like puzzle-solving.
- You can’t predict which problem sets will be most challenging for you.
- **But we have lots of support!**
    - TAs
    - Office Hours
    - Ed
    - Lab
    - Google/AI/Stack Overflow/Python docs

# Grading Scheme

- **2/2**: Code logic is sound and executes assignment instructions properly, even if it fails autograding tests due to negligible typos.
- **1/2**: There are some syntax or logic errors but code runs without error, or code functions as intended but there are significant deviations from the course style.
- **0/2**: Code fails to execute, or a majority of tests fail due to multiple logic errors.

# Collaboration Policy

- Unless told otherwise, you must work on your own for the problem sets. Except in lab/office hours, you may ask each other only general questions.
- See syllabus for more.

# Ed

- Along with office hours, the best resource for getting help
- Discouraged:
    - Emailed questions
    - Ed posts that say, “Why is this code not working?”

# Laptop Policy

- I encourage you not to have your laptop open while we’re coding together during lecture, but keep it handy for self-guided exercises.
- No need to copy down all the code. It will be available on the course GitHub page (link is on syllabus).
- Trust yourself if you think you need your laptop open.

# Pass/fail

- Your TA and I will score each problem set as 0, 1, or 2, but only I can officially grade you.
- To pass, you must demonstrate:
  - Strong attempt at (and non-zero score on) all problem sets.
  - Attendance at and participation in all classes and labs. Excused absences okay.
- Do not obsess over your percentage.
- No one has failed the class. But dozens of students have had to work **very, very hard** not to fail.

# Programming Tools We'll Use This Semester

# Terms you'll hear a lot

| Tool | Purpose | Key Features |
|---|---|---|
| **Git** | Version control system | Tracks code changes<br>Branching and merging<br>Commit history |
| **GitHub** | Online Git repository hosting | Repository hosting<br>Pull requests<br>Issue tracking<br>Code review |

# Terms you'll hear a lot

| Tool | Purpose | Key Features |
|---|---|---|
| ~~Git~~ | ~~Version control system~~ | ~~Tracks code changes~~<br>~~Branching and merging~~<br>~~Commit history~~ |
| **GitHub** | Online Git repository hosting | Repository hosting<br>Pull requests<br>Issue tracking<br>Code review |

# Terms you'll hear a lot

| Tool | Purpose | Key Features |
|---|---|---|
| **GitHub** | Online Git repository hosting | Repository hosting<br>Pull requests<br>Issue tracking<br>Code review |

# Terms you'll hear a lot

| Tool | Purpose | Key Features |
|---|---|---|
| **GitHub** | Online Git repository hosting | Repository hosting<br>Pull requests<br>Issue tracking<br>Code review |
| **GitHub Classroom** | Education-focused GitHub extension | Assignment distribution<br>Automatic repository creation<br>Integrated feedback |
| **GitHub Codespaces** | Cloud-based development environments | Pre-configured w customization options<br>Auto-save for 30 days<br>Accessible from anywhere |

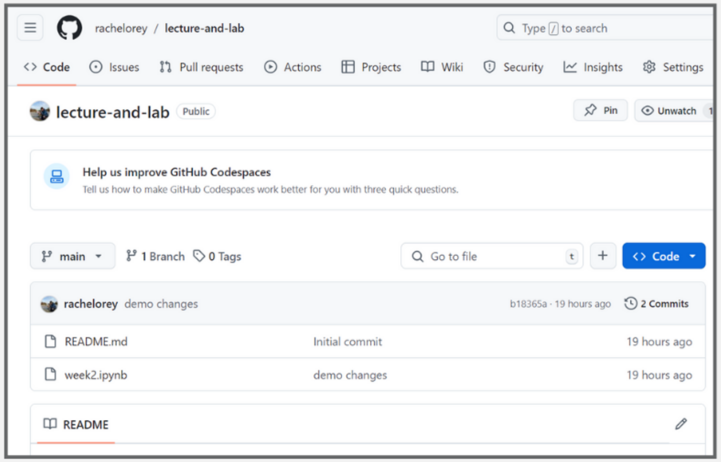
# Terms you'll hear a lot

| Tool | Purpose | Key Features |
|---|---|---|
| **Python** | General-purpose programming language | Simple syntax<br>Extensive libraries<br>Cross-platform compatibility |
| **Jupyter** | Interactive coding environment | Code and Markdown cells<br>Inline visualizations<br>Support multiple languages, incl Python |

# Style Guide

- Adhere to the style guide posted on the course GitHub page
- Some highlights:
    - Program headers
    - Heavily “comment” (explain) your code
    - Use descriptive variable names

## **Remote** Repository on GitHub



Help us improve GitHub Codespaces

Tell us how to make GitHub Codespaces work better for you with three quick questions.

rachelorey demo changes b18365a · 19 hours ago

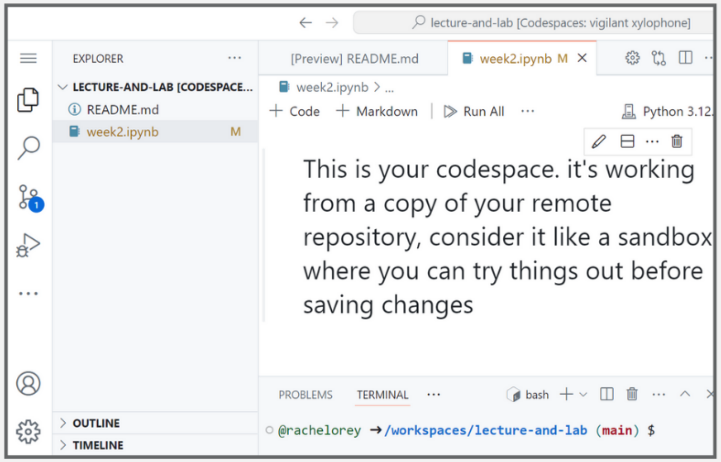
| | | |
|---|---|---|
| README.md | Initial commit | 19 hours ago |
| week2.ipynb | demo changes | 19 hours ago |

- Consider your files on GitHub to be the official version. Those are the ones that your colleagues and collaborators we'll see, and the ones that we'll grade in assignments.
- These files are separate from the work you do on Codespaces unless you add, commit, and push your changes.
- These files are not deleted when you delete your Codespace.

### Push changes from local to remote

### Pull changes from remote to local

## **Local** Clone (Copy) on Codespaces

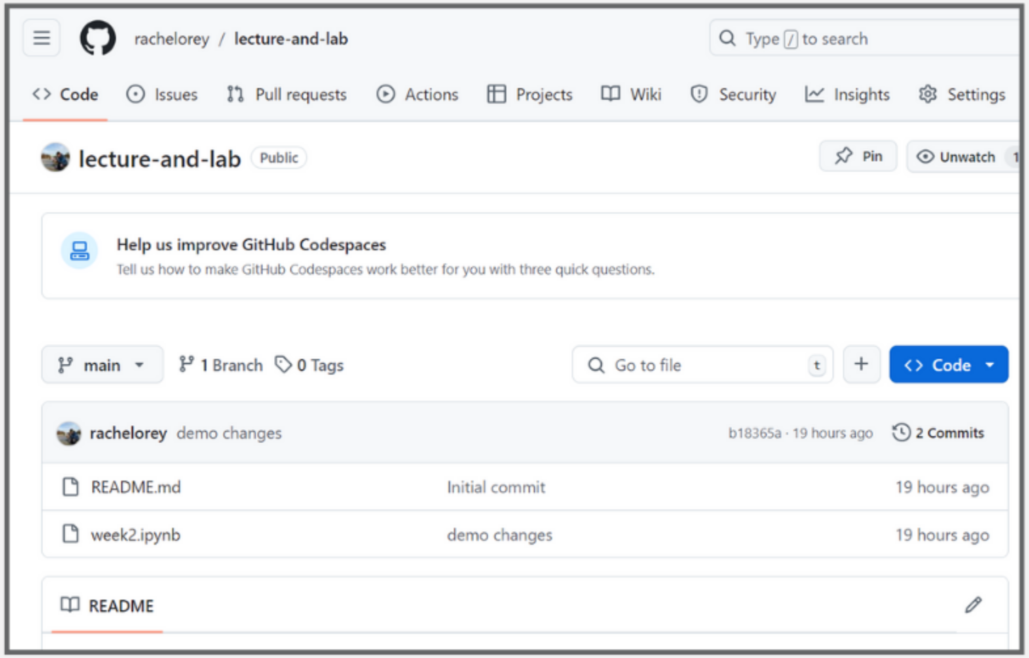


This is your codespace. it's working from a copy of your remote repository, consider it like a sandbox where you can try things out before saving changes

```
@rachelorey →/workspaces/lecture-and-lab (main) $
```

- Codespaces effectively makes a copy of your remote repository that you can code from safely without worrying about messing up the original files.
- To make sure your changes are captured, you have to “add”, “commit”, and “push” your changes back to the remote repository when you're ready.

# Remote Repository on GitHub



**Help us improve GitHub Codespaces**

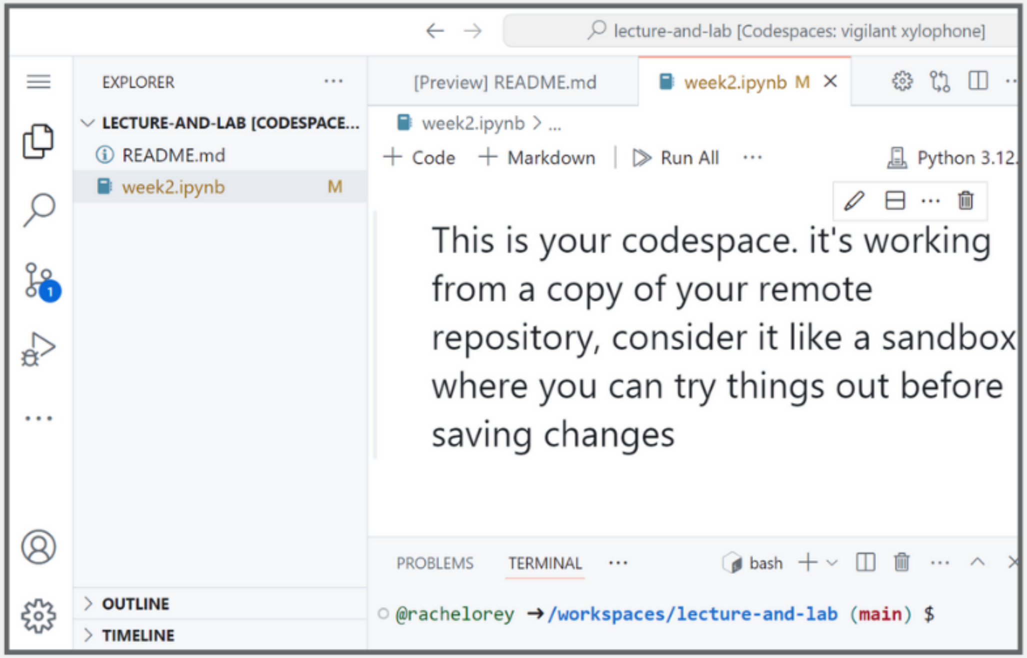
Tell us how to make GitHub Codespaces work better for you with three quick questions.

rachelorey demo changes b18365a · 19 hours ago 2 Commits

| | | |
|---|---|---|
| README.md | Initial commit | 19 hours ago |
| week2.ipynb | demo changes | 19 hours ago |

- Consider your files on GitHub to be the official version. Those are the ones that your colleagues and collaborators we'll see, and the ones that we'll grade in assignments.
- These files are separate from the work you do on Codespaces unless you add, commit, and push your changes.
- These files are not deleted when you delete your Codespace.

# Local Clone (Copy) on Codespaces

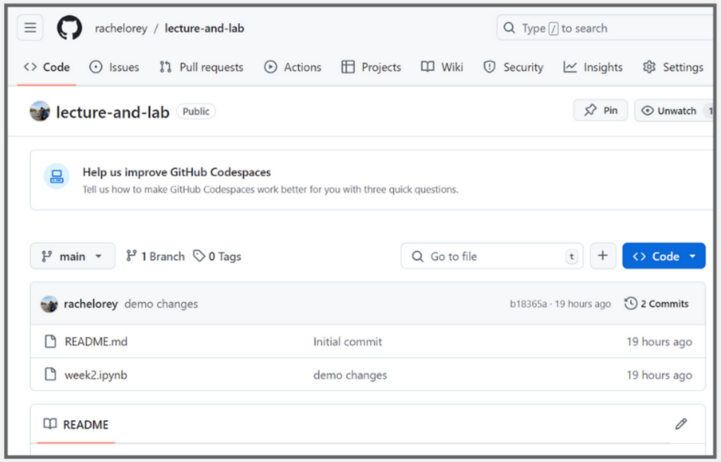


This is your codespace. it's working from a copy of your remote repository, consider it like a sandbox where you can try things out before saving changes

```
@rachelorey ➜ /workspaces/lecture-and-lab (main) $
```

- Codespaces effectively makes a copy of your remote repository that you can code from safely without worrying about messing up the original files.
- To make sure your changes are captured, you have to “add”, “commit”, and “push” your changes back to the remote repository when you’re ready.

## **Remote** Repository on GitHub



**Help us improve GitHub Codespaces**

Tell us how to make GitHub Codespaces work better for you with three quick questions.

rachelorey demo changes

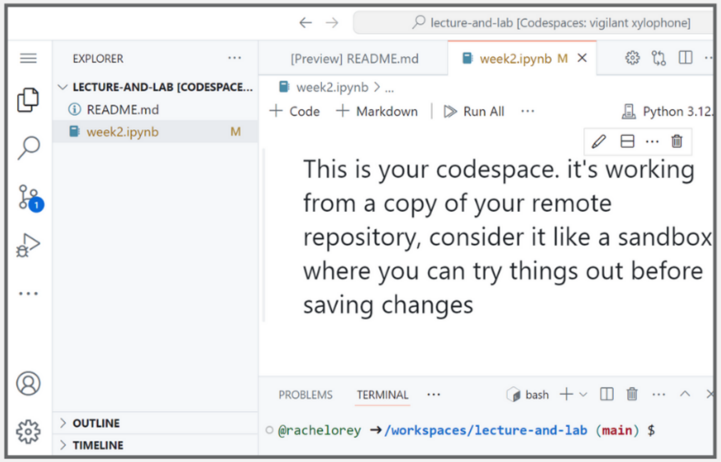
| | | |
|---|---|---|
| README.md | Initial commit | 19 hours ago |
| week2.ipynb | demo changes | 19 hours ago |

- Consider your files on GitHub to be the official version. Those are the ones that your colleagues and collaborators we'll see, and the ones that we'll grade in assignments.
- These files are separate from the work you do on Codespaces unless you add, commit, and push your changes.
- These files are not deleted when you delete your Codespace.

Push changes from local to remote

Pull changes from remote to local

## **Local** Clone (Copy) on Codespaces



This is your codespace. it's working from a copy of your remote repository, consider it like a sandbox where you can try things out before saving changes

```
@rachelorey ➜ /workspaces/lecture-and-lab (main) $
```

- Codespaces effectively makes a copy of your remote repository that you can code from safely without worrying about messing up the original files.
- To make sure your changes are captured, you have to “add”, “commit”, and “push” your changes back to the remote repository when you're ready.