



Introduction to Python Wi24 Section B

Welcome!

Instructor



Dr. Brian Hempel
Direct Manipulation
Programming
Interfaces

TAs

Unay Shah - 2nd Year ECE Masters
Sihan Yang - 2nd Year CogSci PhD
Hanqing "Helen" Zhao - 2nd Year CSE Masters

IAs

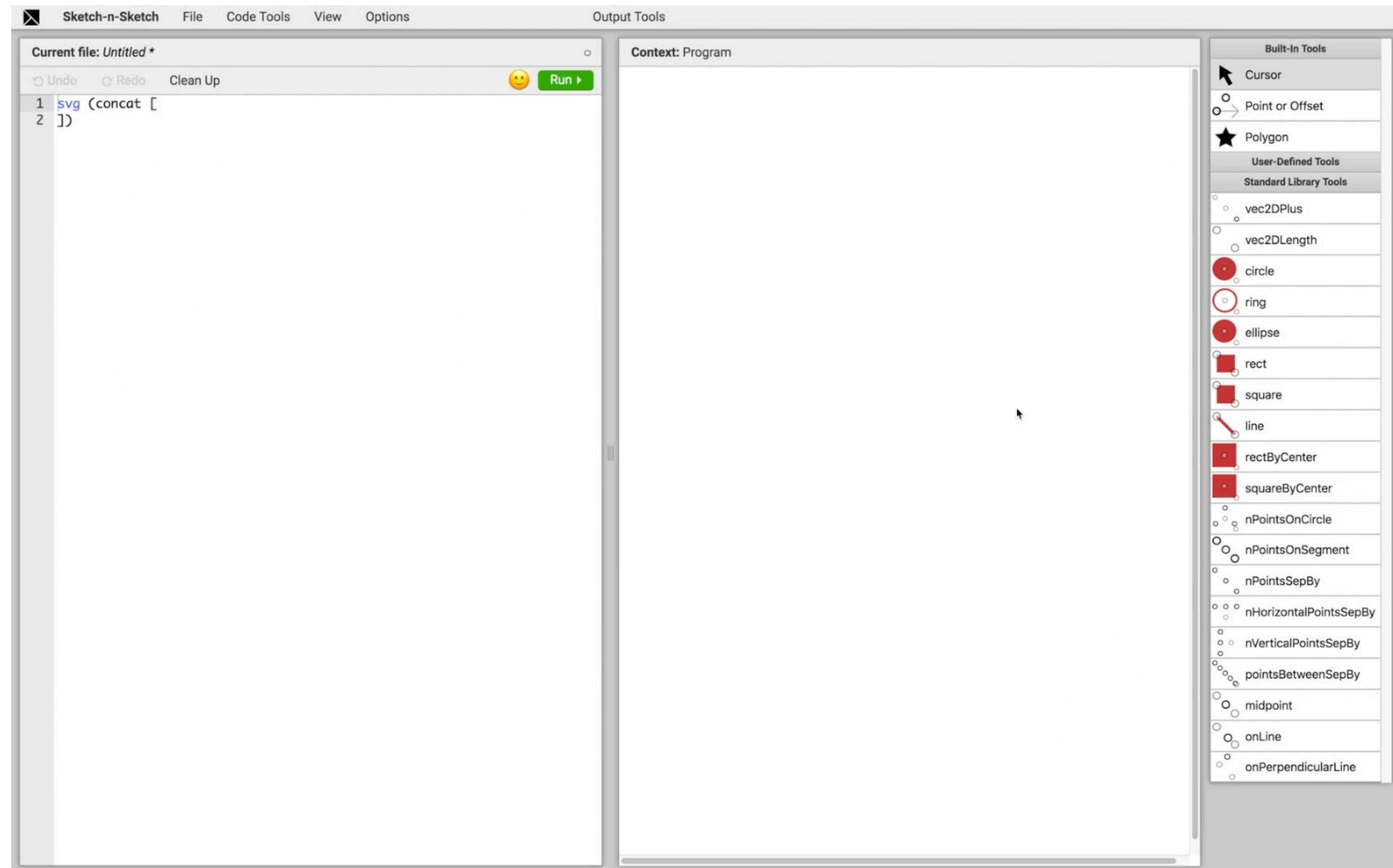
TBA

My Research: Direct Manipulation Programming

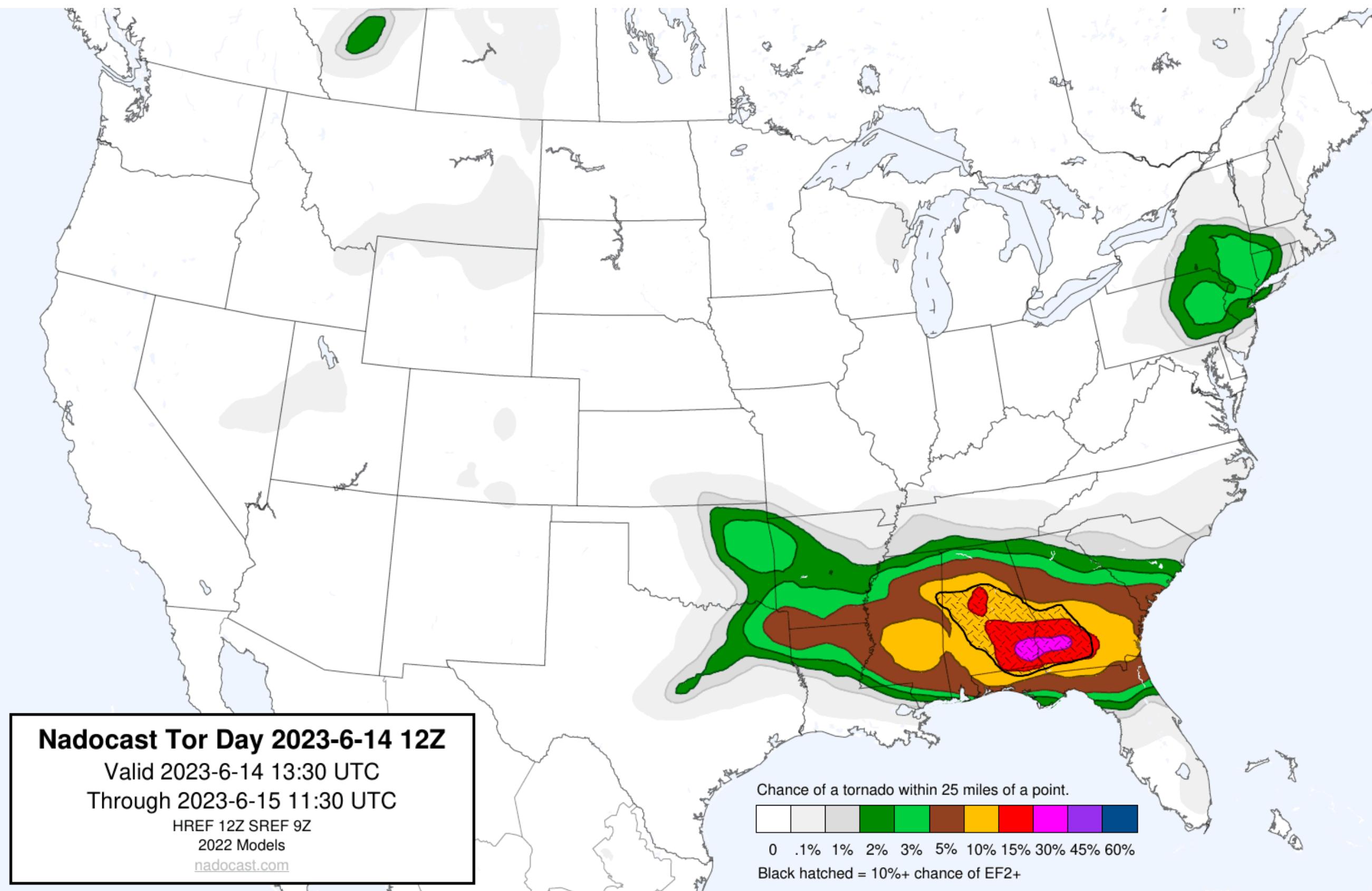
The screenshot shows the Sketch-n-Sketch IDE interface. On the left is the code editor with the title "Current file: Untitled *". The code is written in a functional programming language and generates a fractal snowflake. A red arrow points from the code editor towards the center canvas. On the right is the "Context: Program" panel, which displays the generated fractal snowflake. At the bottom right of the canvas is a black mouse cursor. To the right of the canvas is the "Output Tools" panel, which contains a "Built-In Tools" tree view. The tree includes categories like Cursor, Point or Offset, Polygon, User-Defined Tools (with entries for equiTriPt, oneThirdPt), makeKochPts, Standard Library Tools (vec2DPlus, vec2DLength, circle, ring, ellipse, rect, square, line, rectByCenter, squareByCenter, nPointsOnCircle, nPointsOnSegment, nPointsSepBy, nHorizontalPointsSepBy, nVerticalPointsSepBy, pointsBetweenSepBy), and midpoint.

```
1 equiTriPt [x3, y3] [x2, y2] =  
2   [(x2 + x3 + sqrt 3! * (y2 - y3)) / 2!, (y2 + y3 - sqrt 3! * (x2 - x3)) / 2!]  
3  
4 oneThirdPt [x3, y3] [x, y] =  
5   [x / 1.5! + x3 / 3!, y / 1.5! + y3 / 3!]  
6  
7 point = [39, 314]  
8  
9 point2 = [490, 301]  
10  
11 makeKochPts depth point point2 =  
12   let oneThirdPt2 = oneThirdPt point point2 in  
13   let oneThirdPt3 = oneThirdPt point2 point in  
14   let equiTriPt2 = equiTriPt oneThirdPt3 oneThirdPt2 in  
15   if depth < 2 then  
16     [point, oneThirdPt3, equiTriPt2, oneThirdPt2]  
17   else  
18     let makeKochPts2 = makeKochPts (depth - 1) point oneThirdPt3 in  
19     let makeKochPts3 = makeKochPts (depth - 1) oneThirdPt3 equiTriPt2 in  
20     let makeKochPts4 = makeKochPts (depth - 1) equiTriPt2 oneThirdPt2 in  
21     let makeKochPts5 = makeKochPts (depth - 1) oneThirdPt2 point2 in  
22       concat [makeKochPts2, makeKochPts3, makeKochPts4, makeKochPts5]  
23  
24 depth = 3{1-5}  
25  
26 topPts = makeKochPts depth point point2  
27  
28 botCorner = equiTriPt point2 point  
29  
30 rightPts = makeKochPts depth point2 botCorner  
31  
32 leftPts = makeKochPts depth botCorner point  
33  
34 snowflakePts = concat [topPts, rightPts, leftPts]  
35  
36 polygon1 =  
37   let pts = snowflakePts in  
38   let [color, strokeColor, strokeWidth] = [124, 360, 2] in  
39     polygon color strokeColor strokeWidth pts  
40  
41 svg (concat [  
42   [polygon1]  
43 ])
```

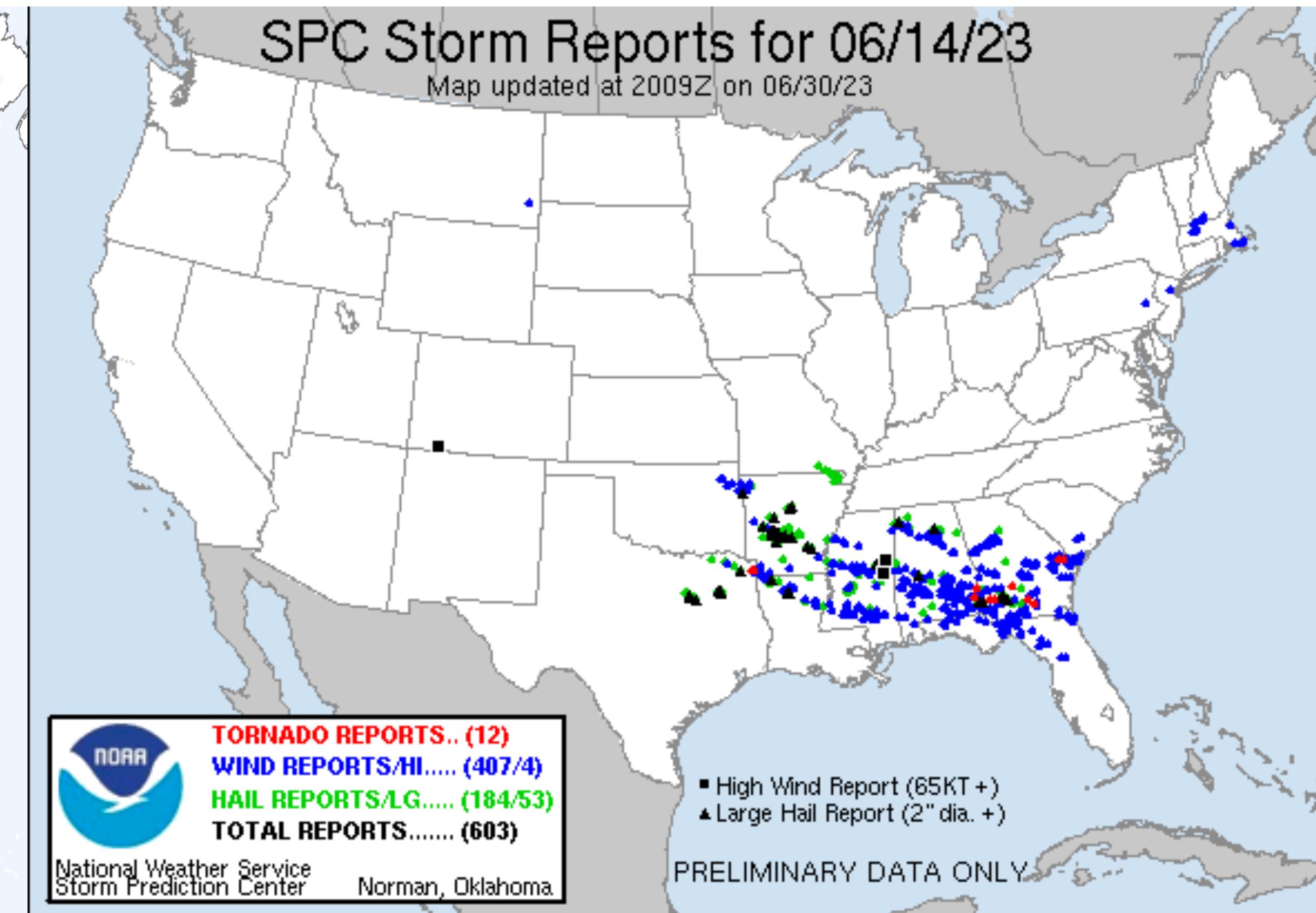
My Research: Direct Manipulation Programming



My Hobby Research: Severe Weather Prediction



Tornado Prediction
June 14, 2023



Red Dots = Actual Tornadoes



Introduction to Python

Section B

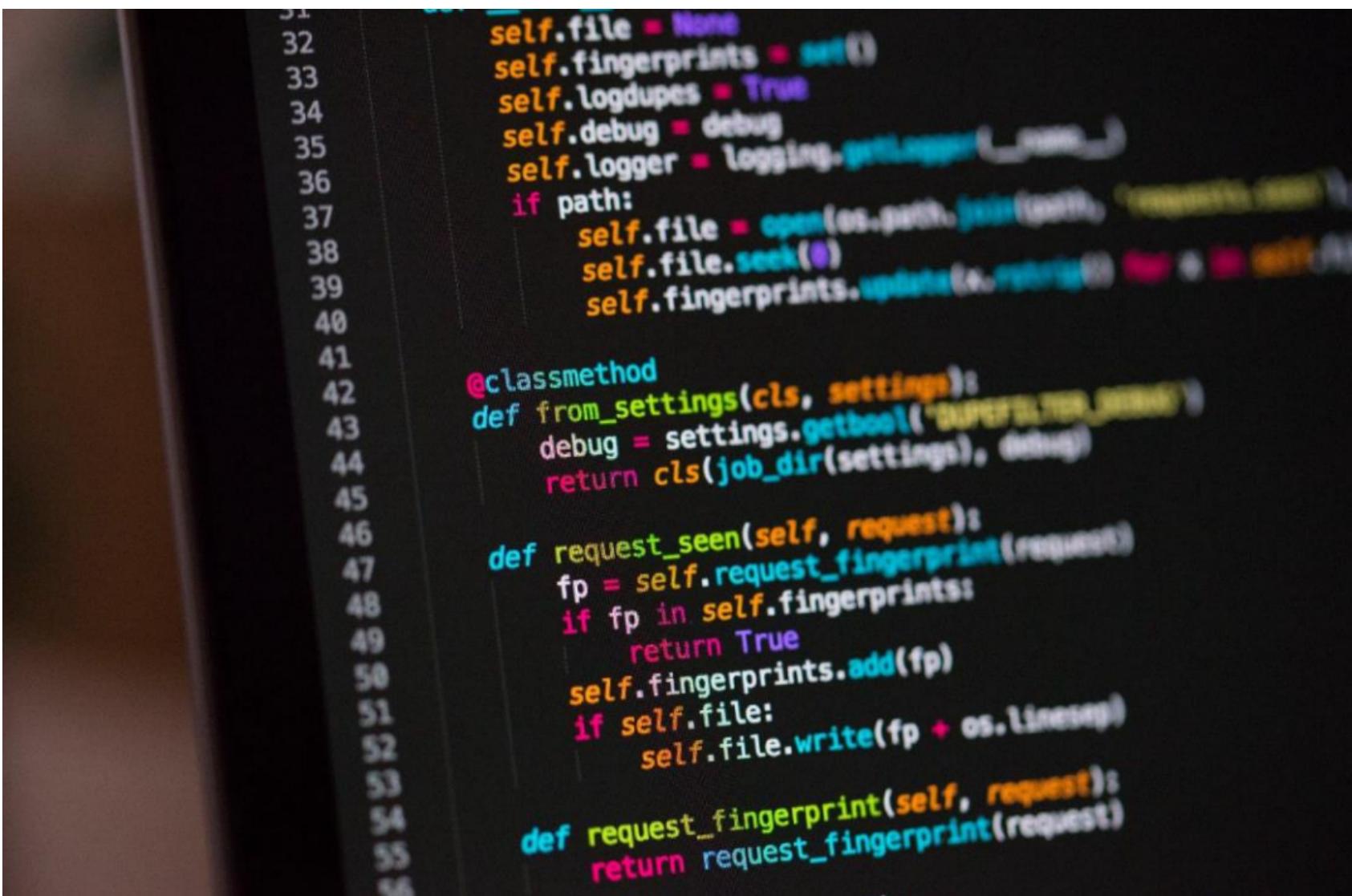
Waitlist

I do not handle it!

- Email cogsadvising@ucsd.edu
- Typically ~3-5 students from each section are enrolled by our staff.
- Enrollment cannot exceed the number of seats in the classroom.
- The waitlist clears at the end of week 2.

Success

You, too, will learn Python.

A photograph of a laptop screen showing a terminal window with Python code. The code is a class definition with methods for handling fingerprints and requests. The background of the slide is a light beige color.

```
31     self.file = None
32     self.fingerprints = set()
33     self.logdups = True
34     self.debug = debug
35     self.logger = logging.getLogger(__name__)
36     if path:
37         self.file = open(os.path.join(path, 'fingerprint.log'),
38                         'w')
39         self.file.seek(0)
40         self.fingerprints.update(os.linesep)
41
42     @classmethod
43     def from_settings(cls, settings):
44         debug = settings.getbool('superuser_debug')
45         return cls(job_dir(settings), debug)
46
47     def request_seen(self, request):
48         fp = self.request_fingerprint(request)
49         if fp in self.fingerprints:
50             return True
51         self.fingerprints.add(fp)
52         if self.file:
53             self.file.write(fp + os.linesep)
54
55     def request_fingerprint(self, request):
56         return request_fingerprint(request)
```

The only thing that is slightly predictive of success in an intro programming course is...

Success

You, too, will learn Python.

A dark computer monitor displays a terminal window with Python code. The code is a class definition with methods for handling fingerprints and requests. The monitor is positioned on the left side of the slide.

```
31     self.file = None
32     self.fingerprints = set()
33     self.logdups = True
34     self.debug = debug
35     self.logger = logging.getLogger(__name__)
36     if path:
37         self.file = open(os.path.join(path, 'requests.txt'),
38                         'w')
39         self.file.seek(0)
40         self.fingerprints.update(self.file.read().splitlines())
41
42     @classmethod
43     def from_settings(cls, settings):
44         debug = settings.getbool('superuser_debug')
45         return cls(job_dir(settings), debug)
46
47     def request_seen(self, request):
48         fp = self.request_fingerprint(request)
49         if fp in self.fingerprints:
50             return True
51         self.fingerprints.add(fp)
52         if self.file:
53             self.file.write(fp + os.linesep)
54
55     def request_fingerprint(self, request):
56         return request_fingerprint(request)
```

The only thing that is slightly predictive of success in an intro programming course is...
how successful the student thinks they will be.

Success

You, too, will learn Python.



```
31     self.file = None
32     self.fingerprints = set()
33     self.logdups = True
34     self.debug = debug
35     self.logger = logging.getLogger(__name__)
36     if path:
37         self.file = open(os.path.join(path, 'requests.txt'),
38                         'w+')
39         self.file.seek(0)
40         self.fingerprints.update(self.file.read().splitlines())
41
42     @classmethod
43     def from_settings(cls, settings):
44         debug = settings.getboolean('DEBUG', False)
45         return cls(job_dir(settings), debug)
46
47     def request_seen(self, request):
48         fp = self.request_fingerprint(request)
49         if fp in self.fingerprints:
50             return True
51         self.fingerprints.add(fp)
52         if self.file:
53             self.file.write(fp + os.linesep)
54
55     def request_fingerprint(self, request):
56         return request_fingerprint(request)
```

The only thing that is slightly predictive of success in an intro programming course is...
how successful the student thinks they will be.

Things that do NOT predict success:

- gender
- age
- personality
- **math ability**

Programming is Concepts You Already Know... ...just expressed precisely for the computer.

Naming stuff

if you can slap a Post-It note on something, you can program

“Yes” and “No”

Numbers

addition, subtraction, occasionally multiplication

Text

Lists

If something Then do something Otherwise do something else

Repeating stuff

Doing the same thing, but with a slight variation

It takes years to become an expert.
**But, by the end of this course, you will be
able to program at an introductory level.**

That is my goal.



We are learning Python! It is:

- **Widely used:** de facto language for scientific computing.
- **Simple.**
- **Not bad.**

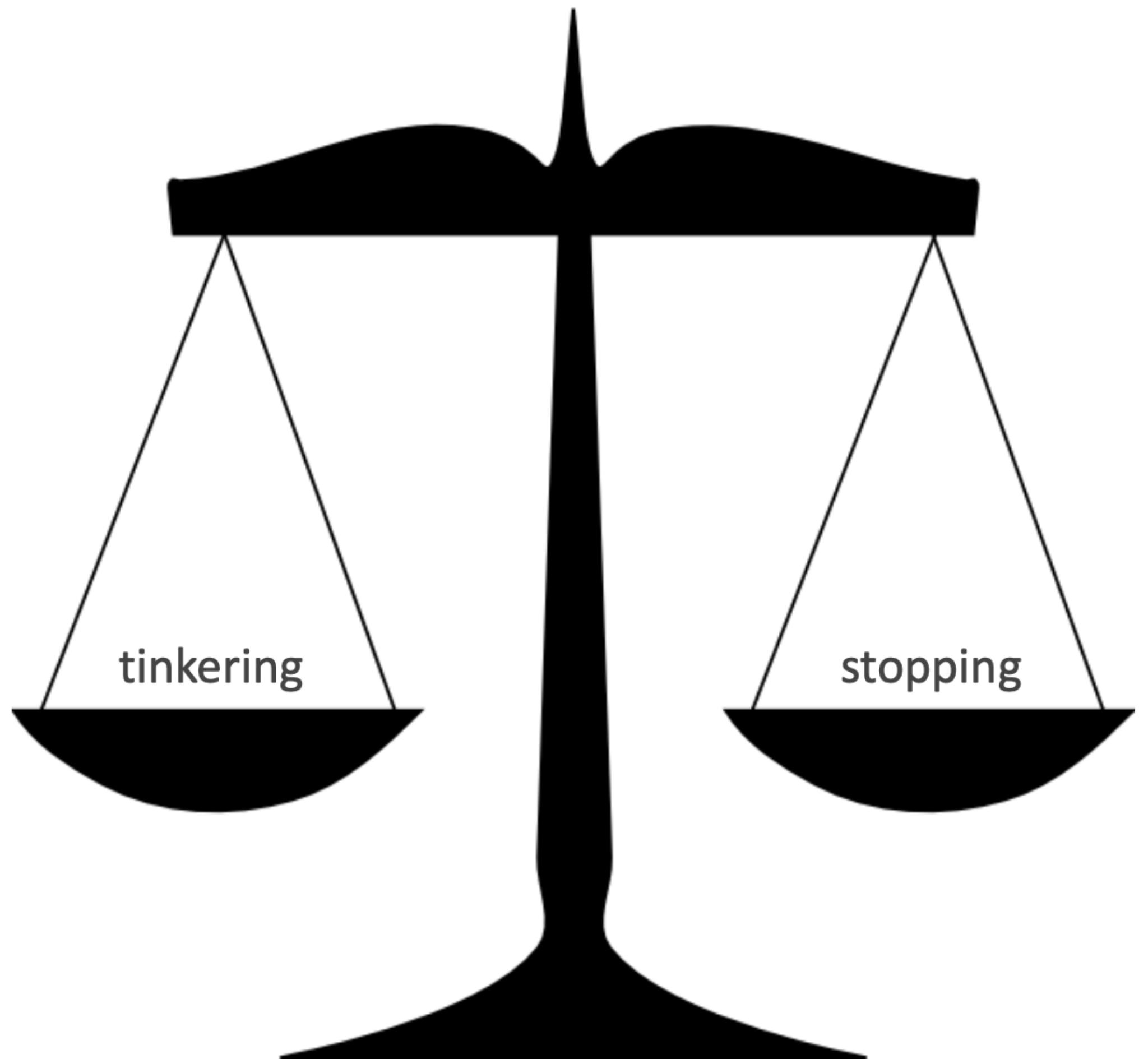
“It’s not the best language for anything, but it’s the second best for everything.” -Brad Voytek

Programming is *not* book learning.

The only way is practice, practice, practice.

To avoid the common pitfalls of intro programming courses, we're going to take the following approach:

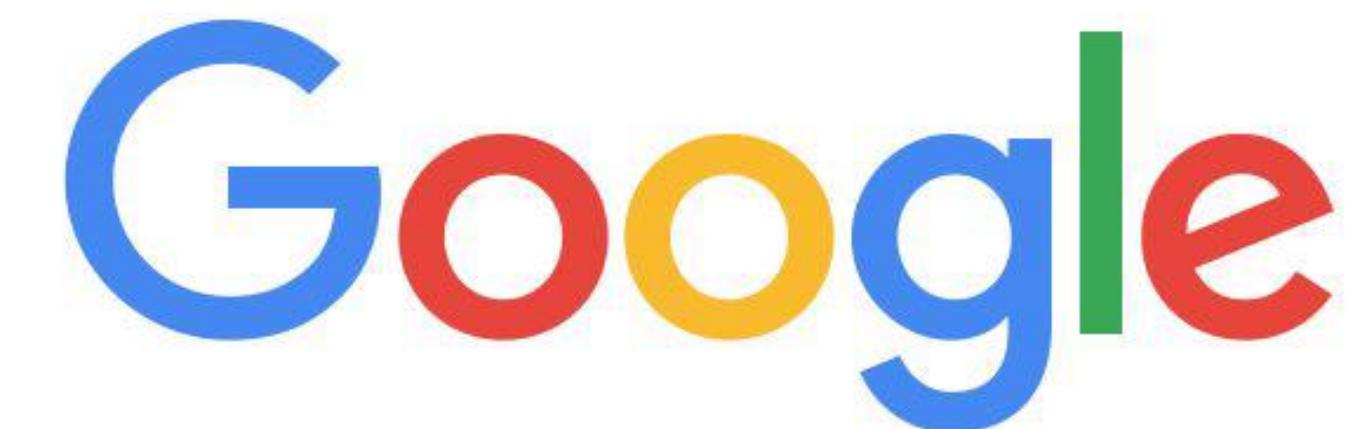
1. First 2/3 of course: **basic concepts**
2. **In-class practice** (no stakes)
 - In-notebook questions for comprehension
3. **Coding Labs** (low stakes)
 - Notebooks provided
 - Staff/classmates there to help
 - Checked for effort, not correctness
4. **Assignments** (mid stakes)
 - After every lecture
 - Completed individually (can work together)
 - Programmatically graded
5. **Exams** (high stakes)
 - Two parts: conceptual (in-class) + technical (take-home)
 - Completed totally individually



Be a mover: Make forward progress. Strike a balance between just stopping and tinkering forever.

- Try for 30min.
- Take a walk.
- Try again.
- Seek help!

Including “in python” in your Google search can be magic



A screenshot of a Google search interface. The search bar at the top contains the text "objects in python". Below the search bar is a list of suggested search terms, each preceded by a small microphone icon indicating they can be spoken. At the bottom of the interface are two buttons: "Google Search" and "I'm Feeling Lucky". A small link at the very bottom right reads "Report inappropriate predictions".

- objects in python
- objects in python 3
- objects in python 2
- objects in python **tutorial**
- objects in python **code**
- objects in python **lists**
- objects in python **django**
- objects in python **inheritance**
- objects in python **return**
- objects in python **for loop**

Google Search I'm Feeling Lucky

Report inappropriate predictions

StackOverflow probably has the answer to your question

The screenshot shows the Stack Overflow 'Tags' page for the 'python' tag. The page has a navigation bar at the top with links for Home, PUBLIC, and Stack Overflow. On the left, there's a sidebar with links for Tags (which is selected), Users, and Jobs. Below that is a 'Teams' section with a 'Q&A for work' button and a 'Learn More' button. The main content area has a search bar with 'python' and a list of tags. The 'python' tag is the most popular, with 113,7913 questions. Other tags listed include python-3.x, python-2.7, python-requests, wxpython, ipython, python-imaging-library, python-3.6, python-3.5, python-import, python-3.4, and python-sphinx.

Home

PUBLIC

Stack Overflow

Tags

Users

Jobs

Teams Q&A for work

Learn More

Search...

Popular Name New

Tags

A tag is a keyword or label that categorizes your question with other, similar questions. Using the right tags makes it easier for others to find and answer your question.

Tag	Questions	Description	Last Activity
python	113,7913	a multi-paradigm, dynamically typed, multipurpose programming language, designed to be quick (to learn, to use, and to run)	1085 asked today, 6241 this week
python-3.x	151128	For questions about Python programming that are specific to version 3+ of the language. Use the more generic [python] tag	273 asked today, 1641 this week
python-2.7	89413	the last major version in the 2.x series. Do not use this tag simply to convey the version of Python you're using, unless the question	40 asked today, 219 this week
python-requests	9229	a full-featured Python HTTP library with an easy-to-use, logical API.	8 asked today, 81 this week
wxpython	6191	a Python wrapper for the cross-platform C++ GUI API wxWidgets.	20 asked this week, 52 this month
ipython	6036	a feature-rich interactive shell for Python, and provides a kernel for frontends such as IPython Notebook and Jupyter Notebook.	15 asked this week, 65 this month
python-imaging-library	4495	The Python Imaging Library (PIL) provides the Python language with a de-facto standard foundation for image work. PIL's	23 asked this week, 113 this month
python-3.6	3882	Version of the Python programming language released in December 2016. For issues specific to Python 3.6. Use more	10 asked today, 43 this week
python-3.5	3260	The version of the Python programming language released on September 13, 2015. For issues that are specific to Python 3.5.	9 asked this week, 34 this month
python-import	3150	For questions about importing modules in Python	18 asked this week, 58 this month
python-3.4	2594	The version of the Python programming language released on March 16, 2014. For issues that are specific to Python 3.4. Use	6 asked this month, 126 this year
python-sphinx	2365	a tool that makes it easy to create intelligent and beautiful documentation. Sphinx is especially suitable for Python	9 asked this week, 34 this month

ChatGPT



Examples

"Explain quantum computing in simple terms" →

"Got any creative ideas for a 10 year old's birthday?" →

"How do I make an HTTP request in Javascript?" →



Capabilities

Remembers what user said earlier in the conversation

Allows user to provide follow-up corrections

Trained to decline inappropriate requests



Limitations

May occasionally generate incorrect information

May occasionally produce harmful instructions or biased content

Limited knowledge of world and events after 2021

ChatGPT

A conversational LLM that will produce prose and code. It has **pros** (you can get your questions answered!) and **cons** (it can be confidently wrong).

Send a message...



Good ChatGPT prompt: “What is an example of _____ in Python?”

What is an example of variables in Python?



In Python, variables are used to store and manipulate data. Here's an example of variables in Python:

python

Copy code

```
# Integer variable  
age = 30
```

```
# String variable  
name = "John Doe"
```

```
# Floating-point variable  
salary = 50000.50
```

Don't ask ChatGPT what you wouldn't ask a classmate.
You wouldn't ask a classmate to just do a question you are stuck on.

Just use “What is an example of...” queries to ChatGPT.

There are also
COGS18-specific
avenues when
looking for help

Questions in **CodingLabs**,
coming to **office hours**, talking
to your **classmates**, or reaching
out for help on **Piazza** are all
options for you. You're
encouraged to help one
another on Piazza!

[LIVE Q&A](#) [Drafts](#) [other lectures](#) [labs](#) [assignments](#) [exams](#) [project](#)Unread Updated Unresolved Following i

Ban User Console · Note History: No history yet

disable history

[New Post](#)

1 view

[Show Actions](#)

Actions ▾

YESTERDAY

Instr Welcome to Piazza!

Hello COGS 18 Students! Piazza is an incredible resource for technical classes. It gives you a place to post questions and answers.

06:40 PM

**Welcome to Piazza!**

Piazza is a Q&A platform designed to get you great answers from classmates and instructors fast. We've put together thi

06:31 PM



Welcome to Piazza!

Piazza is a Q&A platform designed to get you great answers from classmates and instructors fast. We've put together this list of tips you might find handy as you get started:

1. Ask questions!

The best way to get answers is to ask questions! Ask questions on Piazza rather than emailing your teaching staff so everyone can benefit from the response (and so you can get answers from classmates who are up as late as you are).

2. Edit questions and answers wiki-style.

Think of Piazza as a Q&A wiki for your class. Every question has just a single **students' answer** that students can edit collectively (and a single **instructors' answer** for instructors).

3. Add a followup to comment or ask further questions.

To comment on or ask further questions about a post, start a **followup discussion**. Mark it resolved when the issue has been addressed, and add any relevant information back into the Q&A above.

4. Go anonymous.

Shy? No problem. You can always opt to post or edit anonymously.

5. Tag your posts.

It's far more convenient to find all posts about your Homework 3 or Midterm 1 when the posts are tagged. Type a "#" before a key word to tag. Click a blue tag in a post or the question feed to filter for all posts that share that tag.

6. Format code and equations.

Adding a code snippet? Click the **pre** or **tt** button in the question editor to add pre-formatted or inline teletype text.

Mathematical equation? Click the **Fx** button to access the LaTeX editor to build a nicely formatted equation.

7. View and download class details and resources.

Click the **Course Page** button in your top bar to access the class syllabus, staff contact information, office hours details, and course resources—all in one place!

Contact the Piazza Team anytime with questions or comments at team@piazza.com. We love feedback!

Average Response Time:

Special Mentions:

N/A

There are no special mentions at this time.

Online Now | This Week:

1 | 1

In technical
classes, Piazza is
a particularly
helpful resource

There are **rules**:

1. No duplicates.
2. Include Assignment & Question in Summary line.
3. Posts must include your question, what you've tried so far, and resources used.
4. Public posts are best.
5. Helping one another is encouraged.
6. No assignment code in public posts.
7. We're not robots.

A message for **first-gen students, transfer students, and those who don't have older siblings/friends** who have attended UCSD/university

If you are struggling, come to office hours. Ask questions on Piazza. Reach out to me to ask for better approaches. Your classmates ARE doing this. And, you're not alone.

If you need a bit longer on something b/c you fell sick, a family thing came up, work called you in for an extra shift, etc., ask for an extension. Your classmates ARE doing this.