CIS 5520
Advanced Programming

Fall 2024

Welcome!

- Sit at any table
- Introduce yourself to your table
- Pick a team name

Course Staff

Instructor: Stephanie Weirich

sweirich@seas.upenn.edu

OH: *Wednesdays, 2-3pm, Levine 510*

TAs: Jonathan Chan, Gary Chen, Mayank Keoliya

What is Advanced Programming?

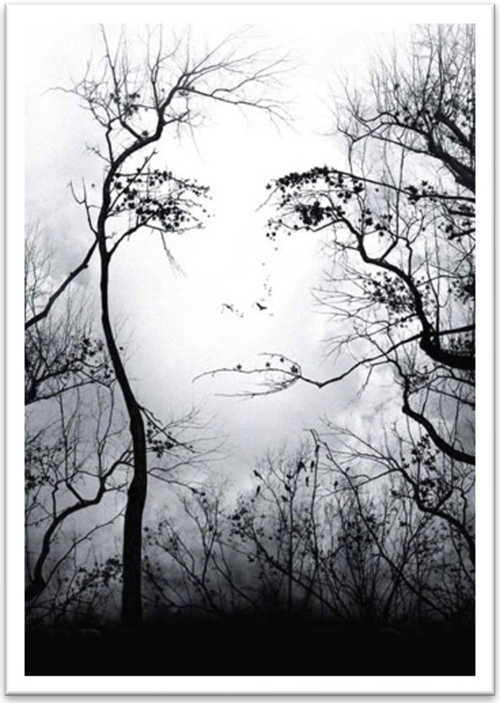
- Good programmers get the job done

- Excellent programmers
  - write code that other people can understand, maintain and modify
  - rewrite/refactor code to make it clear and *simple*
  - use and create *abstractions* to capture fundamental designs
  - can explain *semantics* precisely: what their code does and why

Simplicity through Abstraction

- Readable
- Reusable
- Modifiable
- Predictable
- Checkable

- Advanced type structures: Multiple levels of abstraction available



Simplicity through Purity

- Readable
- Reusable
- Modifiable
- Predictable
- Checkable

- Functional Programming: Use mathematics to explain what code **means**, instead of what it does
- Pure code makes all dependencies explicit, nothing is hidden

CIS 5520

Haskell

Course content

Functional Programming

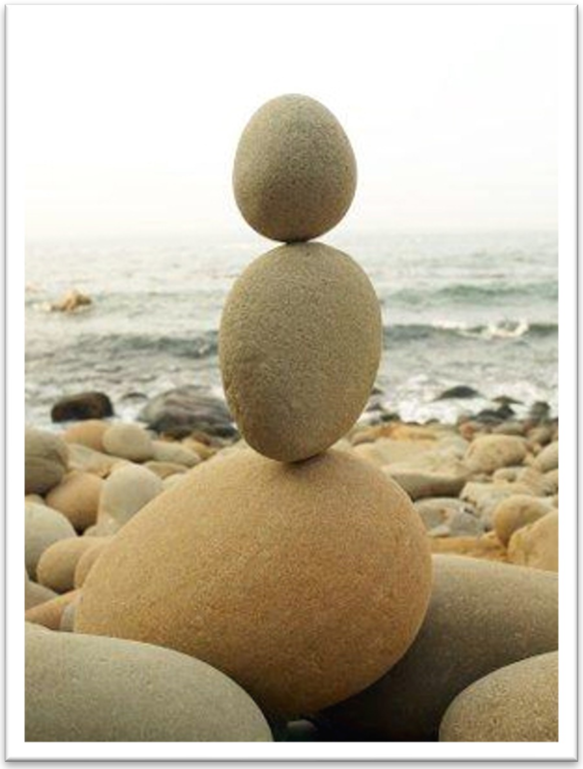
- Black-belt Haskell
- Mathematical approach to programming
- Focus on semantics and types
- Many small-scale case studies

Advanced Techniques

- Modular design and abstraction
- How to make types work for you
- Test and property driven development
- Collaboration (pair programming)

Lots of programming!

- Small in-class exercises
- Bi-weekly homework assignments
- End of semester project



What this course is not

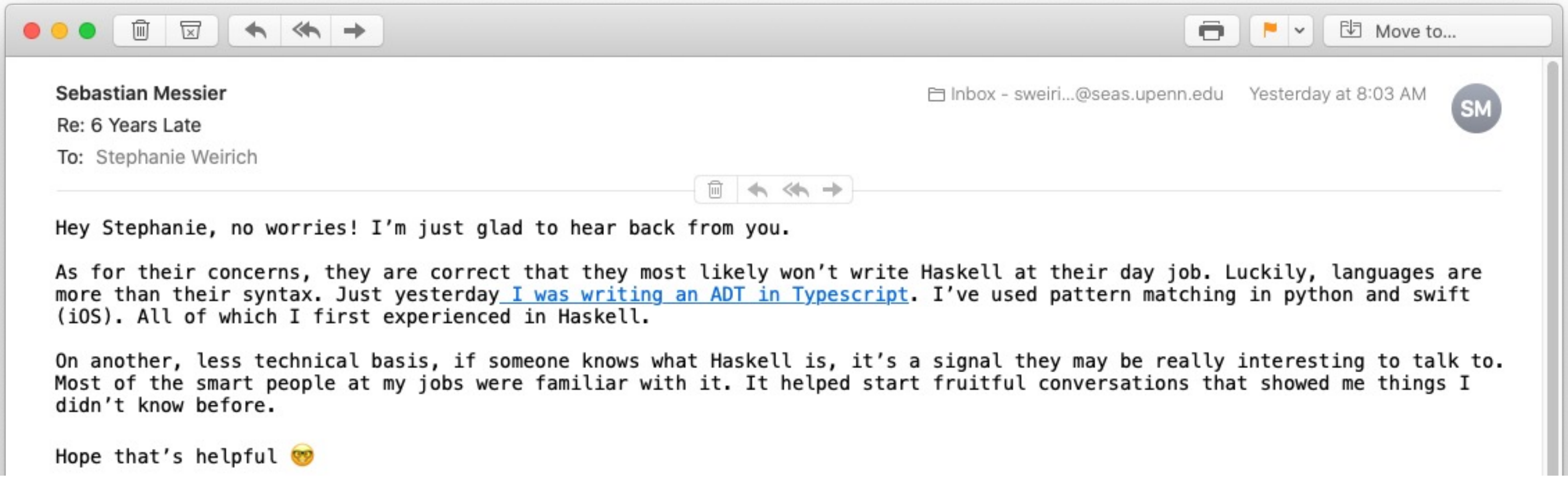
- CIS 3500/5730, Software Engineering
  - Focuses on "Software in the large”
  - How to deal with code you didn’t write
  - Problems that arise in projects that are too large for one person
    - lifecycle models
    - project management
    - design modeling notations (UML)
    - formal specification
- The two courses complement each other

What are you most excited about for CIS 5520?

- Functional Programming (11x)
- Haskell (6x)
- Learning to be a better programmer (5x)
- Learning different programming techniques and new ways to solve problems
- Learning the fundamentals behind compilers and query languages
- Fun with monads. Also really looking forward to the project
- Reassociating with what real-world programming is like.

What concerns do you have?

- I have never learned about functional programming languages before, and I'm worried I can't follow the pace of this class since it seems advanced
- I am a little bit worried about what projects we are going to do as well as the workload
- Maybe about it being too hard to manage
- The theory behind functional programming looks intimidating
- Functional programming lol
- a little worried about random partner for homeworks



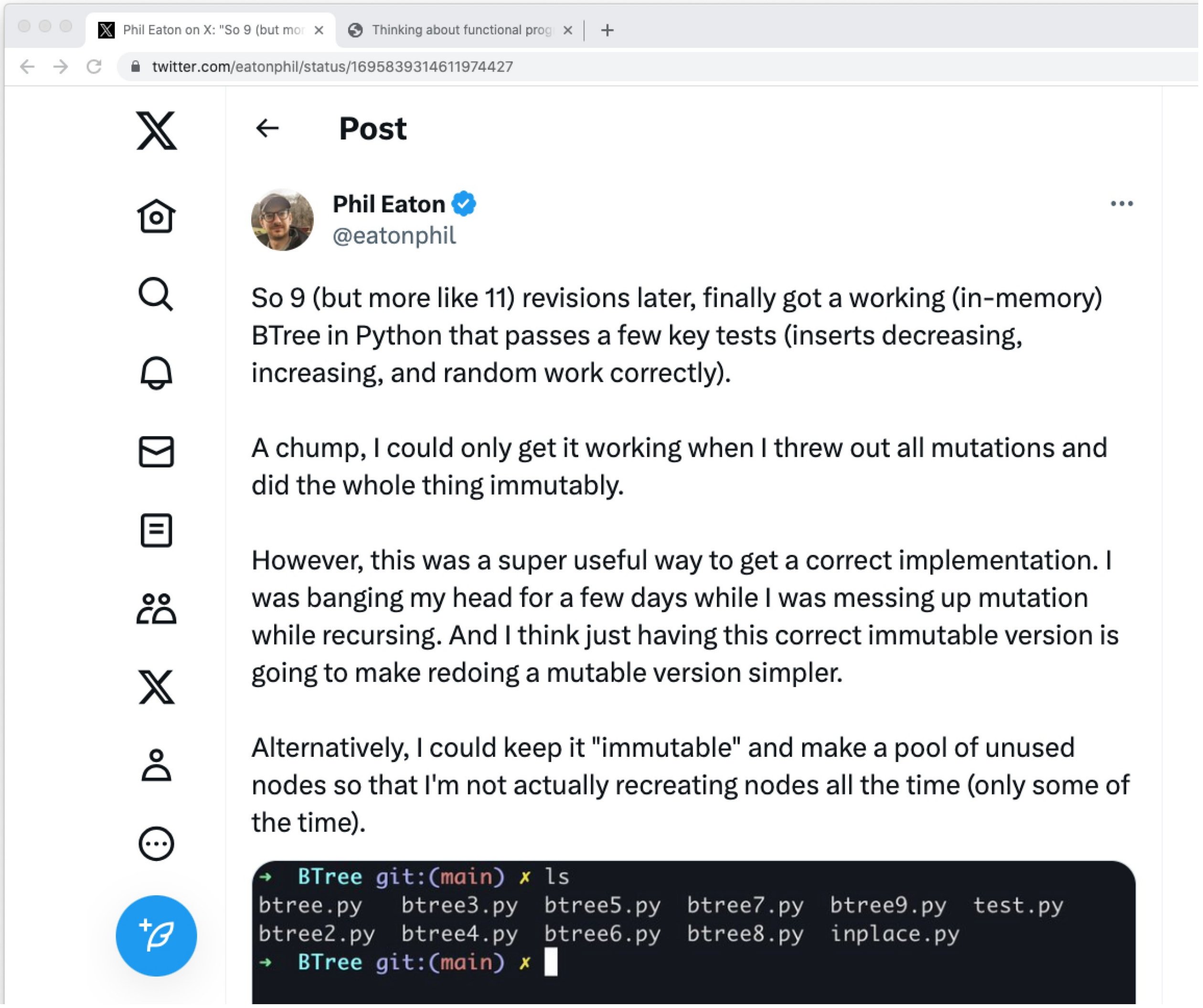
Sebastian Messier
Re: 6 Years Late
To: Stephanie Weirich
Inbox - sweiri...@seas.upenn.edu   Yesterday at 8:03 AM

Hey Stephanie, no worries! I'm just glad to hear back from you.

As for their concerns, they are correct that they most likely won't write Haskell at their day job. Luckily, languages are more than their syntax. Just yesterday I was writing an ADT in Typescript. I've used pattern matching in python and swift (iOS). All of which I first experienced in Haskell.

On another, less technical basis, if someone knows what Haskell is, it's a signal they may be really interesting to talk to. Most of the smart people at my jobs were familiar with it. It helped start fruitful conversations that showed me things I didn't know before.

Hope that's helpful 🤓



Phil Eaton on X: "So 9 (but mor
Thinking about functional prog
twitter.com/eatonphil/status/1695839314611974427

Post

Phil Eaton
@eatonphil

So 9 (but more like 11) revisions later, finally got a working (in-memory) BTree in Python that passes a few key tests (inserts decreasing, increasing, and random work correctly).

A chump, I could only get it working when I threw out all mutations and did the whole thing immutably.

However, this was a super useful way to get a correct implementation. I was banging my head for a few days while I was messing up mutation while recursing. And I think just having this correct immutable version is going to make redoing a mutable version simpler.

Alternatively, I could keep it "immutable" and make a pool of unused nodes so that I'm not actually recreating nodes all the time (only some of the time).

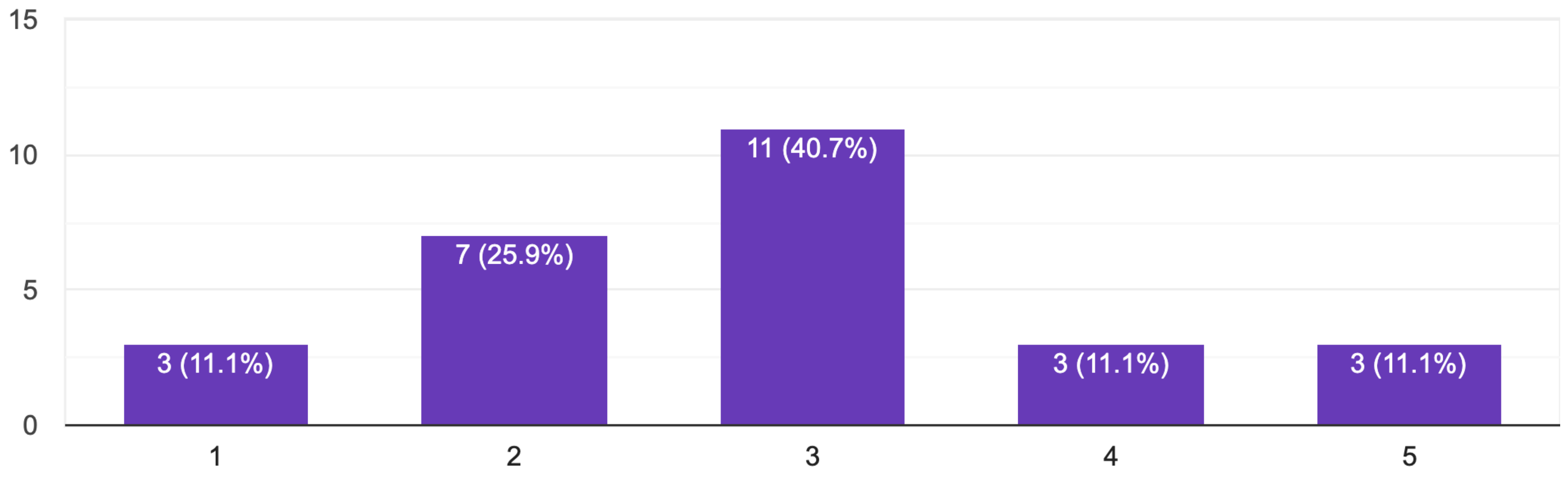
```
➜  BTree git:(main) ✗ ls
btree.py   btree3.py  btree5.py  btree7.py  btree9.py  test.py
btree2.py  btree4.py  btree6.py  btree8.py  inplace.py
➜  BTree git:(main) ✗ 
```

Audience

- People with strong background in programming and mathematics
- No experience with FP expected, but helps
- Undergraduates, Masters, and PhD students together

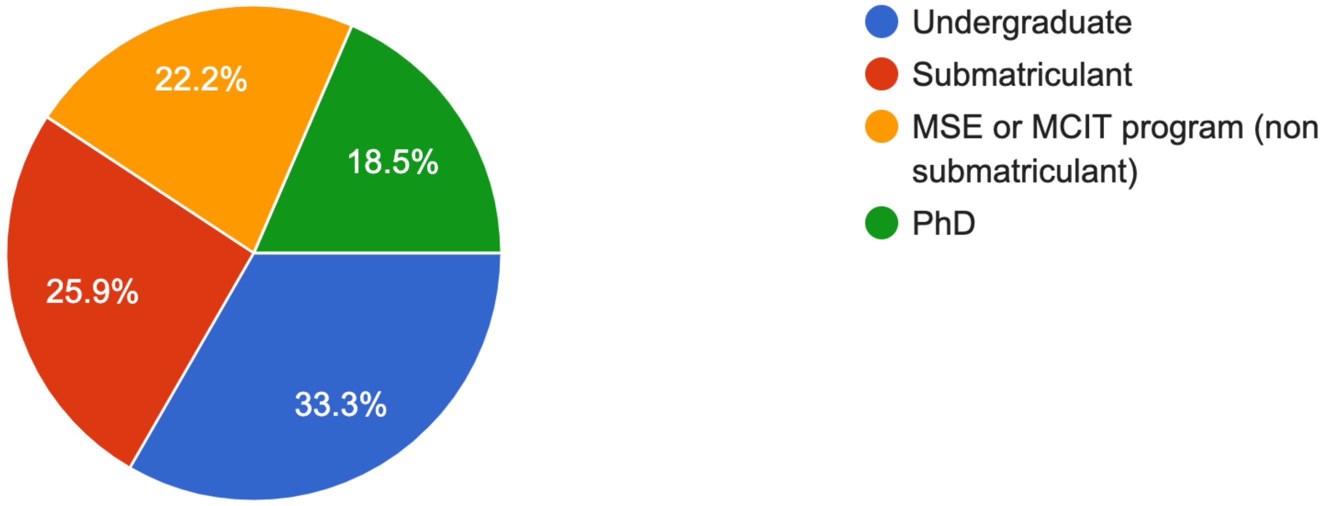
How much experience do you have with functional programming ?

27 responses



What is your status in Fall 2024

27 responses



How will this all work?

General Course Structure

- Every week has a **github repo**!
  - Read module and complete quiz by end of class Monday
  - Interactive lecture Monday (module highlights w/live coding)
  - In-class exercise Wednesday
  - Homework due alternate Thursdays (midnight), covers two topics
- Some weeks are different (Labor day, Fall break, Thanksgiving)
- End-of-semester: final project

Grading Structure

- 15 % Quizzes
  - quizzes (usually due Mondays, can complete before or during class)
  - **first module/quiz available now**
- 15 % Active learning / engagement
  - in class exercises
  - office hours – let's chat!
- 50 % Programming assignments
  - in pairs, most *randomly* assigned
  - graded on correctness, style and (asymptotic) efficiency
  - first assignment available now
- 20 % Final Projects (your choice)

Because of the active learning component, in person participation is essential!

Course Content

- Course content available in two forms
  - Formatted reading: on the public course website (under "Schedule")
  - IDE experimentation (*recommended*): public repo in github
- **Read module "Basics" before next class**
  - Part of the "01-intro" project on github
  - Fill in the "undefined" parts in your IDE
- **Gradescope quiz on material due at the end of next class**
  - Answers will be provided during class, if needed

Active Learning Goals

- Goal for the semester: create a CIS 5520 *community*
  - You should get to know me and the TAs (they're great!)
  - You should get to know each other (you are all great!)
- Forced, random interactions during class time and outside
  - Small and large group discussions
  - In-class exercises with a partner or table
  - Random homework partners
  - TODAY: PL-themed icebreaker game

Homework #1

- Based on "Basics" (available now) and "HigherOrder" modules (tba)
- Clone public repo to complete the assignment
- Work alone or with a partner (your choice), only one person should submit via Gradescope
- Must compile to get any credit, submit early to make sure there are no problems
- **Due Thursday, Sept 12th at midnight**
- Late policy (all homework assignments)
  - 10 point penalty for up to 24 hours late
  - 20 point penalty for up to 48 hours late
  - no credit for assignments submitted after 48 hours
  - *if you have an emergency, please ask for an extension*

Academic Integrity Expectations

- CIS 5520 is a course and not a developer job
  - we will ask you to refrain from using standard libraries or referring to (easily accessible) solutions
- **Homework solutions must be yours**
  - Don't ask ChatGPT to solve your homework
  - Don't search for solutions online
  - Don't ask someone else (other than your partner) to do your homework for you
- Can make limited use of ChatGPT, but *do so with caution*
- *Ask if you are unsure!*

Where to go for more information

- Public site (http://www.seas.upenn.edu/~cis5520)
  - Haskell related material, HW instructions
- Github (https://github.com/upenn-cis5520)
  - Code repos for lecture content, in-class exercises (public)
  - HW repos
- Canvas site (https://canvas.upenn.edu/courses/1741501)
  - **Syllabus**
  - Link to Ed (Announcements and questions)
  - Link to Gradescope (Quizzes, Homework submission)

First three weeks

- Today: Introductions/Game

- Wed, Sep 4: **Basics** module, first quiz

- Mon, Sep 9: **HigherOrder** module, second quiz
- Wed, Sep 11: **Foldr** in-class exercise
- Thurs, Sep 12: HW #1 due

Waitlist and registration

- Current status: should be space for everyone
- I will process waitlist requests until September 9th
- If you are not yet registered, tell me today so I can add you to Canvas
- Let me know if you no longer want to be on the waitlist

Things to do right now

- Read syllabus on Canvas
- Create a github account (if you do not have one)
- Respond to Fall 2024 intro survey (if you haven't already)
- Introduce yourself to the others at your table
- Start reading "Basics" module, install software, clone hw01 repo (after class)
- Office hours:
  Stephanie: Today, 2-3 PM, Levine 510

PL game!

- Each table is a team and *must* have a team name
- Match each code listing with its **algorithm** and **programming language**
- Each algorithm / language is used only once
- No google / web searching / ChatGPT allowed
- Only one guess per sheet!
- We will calculate scores at **1:15 PM**

fin

So, Who Uses FP?

Google™

So, Who Uses FP?

So, who uses FP?

facebook

So, Who Uses FP?

So, Who uses FP?

JANE STREET

BARCLAYS

Standard Chartered

CREDIT SUISSE

Bank of America Merrill Lynch

Goal: Obviously no deficiencies

- Want code that is so simple, it obviously works

$$\text{simple code is} \left\{ \begin{array}{l} \text{Readable} \\ \text{Reusable} \\ \text{Modifiable} \\ \text{Predictable} \\ \text{Checkable} \end{array} \right.$$

- OK… so what makes code simple?