CSCI 275: Programming Abstractions

Lecture 01: Introduction

Spring 2025

Stephen Checkoway, Oberlin College
Slides gratefully borrowed from Molly Q Feldman

Goals for Today's Class

- Why take this class
  - A bit about PL
  - A bit about me

- Course logistics

- An introduction to what we'll be learning

About the course

This is a course about *Programming Languages and Functional Programming*

We will use the language Racket to discuss, analyze and implement various aspects of programming languages.

Opportunities for Help

My office hours (King 231)

- Times TBA! (Sorry, I'll have this figured out next week)

Elinor Frost will run specific Office Hours for 275 this semester

Why learn this material?

1. Learning New Computational Models and Speeding Language Learning
2. Choosing the Right Language
3. Learning Widely-Applicable Design and Implementation Techniques
4. Creating new Domain Specific Languages or Virtual Machines
5. Developing Fundamental Concepts and Problem Solving Skills

Why Undergraduates Should Learn the Principles of Programming Languages. ACM SIGPLAN Education Board. 2010.

Why learn this material?

“The best preparation for quickly learning and effectively using new languages is **understanding the fundamentals underlying all programming languages** and to have some prior experience with a variety of computational models.

Such knowledge will **endure** longer than today’s “hot” languages, which will undoubtedly become obsolete and give way to new languages in the future.

In addition, this knowledge will enable students to quickly look beyond an unfamiliar language’s surface-level details (such as syntax) and **grasp the underlying computational model’s design principles**.”

Why Undergraduates Should Learn the Principles of Programming Languages. ACM SIGPLAN Education Board. 2010.

Who am I?

Stephen Checkoway (he/him)

Feel free to call me any of Steve, Stephen, Professor Checkoway

Research interests: computer security, computation in unexpected places

Fun facts about me:

I have three adorable Oberlin-born cats (Kirk, Tasha, and Reginald)

I’m face blind

What if PL is not your interest?

What if you don’t like this class? What if you’re super into systems/theory/ML/knitting/soccer instead?

Hypothesis: Learning **functional programming** and **programming languages concepts** makes you a better programmer. Period.

The Course

Parts of the course

Fundamentals: Racket and things you can do with it

Application: Implementing Racket and other language ideas

Extensions: Special Topics (foundations, language design, etc.)

Course Technology

- DrRacket (free, version 8.14)
  - https://racket-lang.org/download/ (most Macs, pick Apple Silicon)

- Ed
  - This will be the main contact method for the entire course, including questions for me

  - **Why?** It reduces the number of places you need to check for information

  - You should have been added to Ed already; **email me if not!**

- GitHub

Assessment

Weekly Homework (10 Assignments) (60% of the course grade)

- You can work by yourself or in teams of exactly 2 for most homeworks
- Each assignment has lots of small, independent parts
- First commit by end of each Monday!
- Full homework due by end of each Friday

Final Project - Summary Problems (10%)

Two take home exams (see the course schedule) (20%)

Class participation via iClickers (10%)

What are Summary Problems?

Three common student concerns in classes such as 275 are:

(a) how to use feedback from previous assignments to help them grow

(b) It is hard to see the full picture of the course's goals and learning objectives

(c) There is only one (graded) opportunity to complete an assignment.

Summary Problems are 9 programming problems designed to address these concerns.

Logistics of Summary Problems

- This is an **independent**/solo **endeavor**
- Modeled after the idea of Mastery Learning

- You can turn them in for feedback as many times as you'd like
- They will only be graded as part of a separate Final Project assignment
- Worth 10% of your grade
- **This is the Summative (Final) Assessment in the class**

In Class: Clickers!

- iClickers have been shown to have *significant* positive learning impacts in CS Education
- Similar to most CS courses you've been in
- You need one by next **Monday**

I'll pose a (carefully designed) question to the class…

Some question about a concept we just talked about?

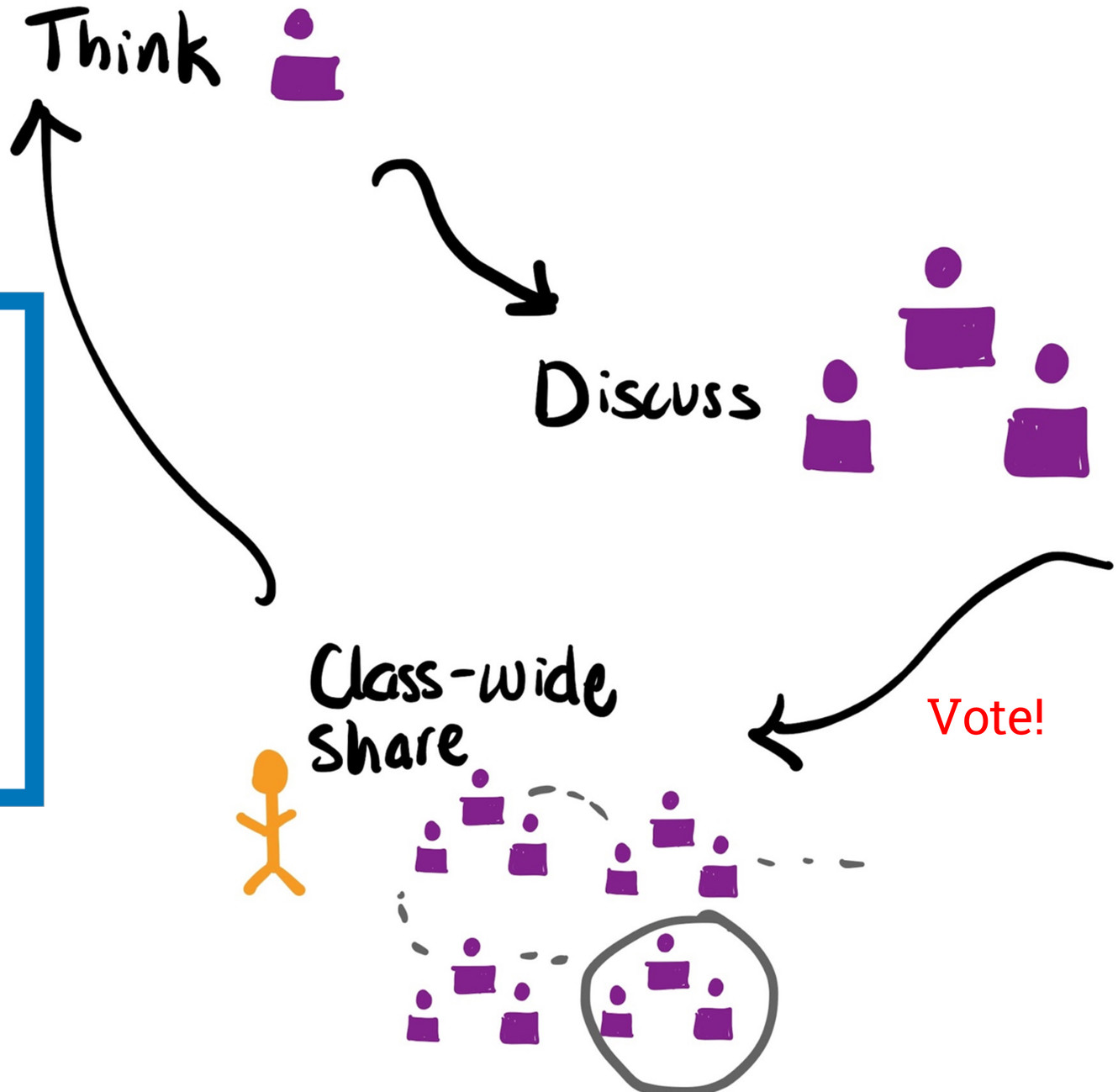
A. Distractor answer 1

B. The right answer

C. Distractor answer 2

D. Distractor answer 3

E. None of the above



Group Discussion Norms

Make sure everyone gets to talk

Have everyone state their answer before discussing which answer is correct

Take turns reporting out to the class as a whole

If you think someone is wrong, ask them to explain their thinking rather than just dismissing it

Everyone has something to contribute, everyone has something to learn

Keep your discussions *on topic* to the course material

Class Norms

Contribute as you feel comfortable

- If you're not comfortable answering, you can pass.
- If you're not usually inclined to speak much in class, push yourself to ask questions more often.

Be aware of the space you take up in class

- Make space for others, use some space for yourself

The main goal of every person in the class should be to engage proactively with the ideas we understand the least. If someone asks a question/makes a comment that seems obvious to you, show them respect.

This is not a competition.

Masking in CSCI 275

In 275 lecture/my office:

- You are welcome to wear a mask at any time, for any reason
- I request that if you're not feeling well or recovering from sickness, please wear a mask

Turn to your neighbor, introduce yourself (name, pronouns, etc.) Then discuss the following question:

What is your favorite punctuation mark?

Turn to your neighbor, introduce yourself (name, pronouns, etc.) Then discuss the following question:

What is your favorite punctuation mark?

After this class, you will have strong feelings about **parentheses** (good or bad)!!!

More Information?

We have a **course syllabus –** it is your responsibility to read it, it’s the main course website (linked from BlackBoard)

The website also has the course schedule – where readings are posted, topics, etc.

Slides will be posted to the course website (before class if I am able, after class if I am not)

Can’t AI just do this for me?

Syllabus: you can’t find out! AI tools are prohibited here.

However, to address your curiosity…

Most Code LLMs are *really bad at Racket*

T	Models	Win Rate	humaneval-python	racket
🟢	Replit-2.7B	6.38	20.12	3.22

https://huggingface.co/spaces/bigcode/bigcode-models-leaderboard

What we are going to learn

A Racket Program

```
(define (eggs x)
  (if (< x 2)
      "not enough for an omelette"
      "enough"))
```

A quick history

For some languages, there are language *families*

- Languages have similar sources, but diverge along the way

Racket is a LISP-style language

- It has more “modern” up-to-date features that make it easier to program in
- Diverged from Scheme around 2010
- The Racket creators started with *education in mind*

https://felleisen.org/matthias/manifesto/Racket__Historically_Speaking.html

Functional Language of the Week: LISP

John McCarthy invented LISP at MIT around 1960 as a language for AI.

LISP grew quickly in both popularity and power. As the language grew more powerful it required more and more of a system's resources. By 1980, 5 simultaneous LISP users would bring a moderately powerful PDP-11 to its knees.

Guy Steele developed Scheme at MIT 1975-1980 as a minimalist alternative to LISP.

Scheme is an elegant, efficient subset of LISP. It has some nice properties that we will look at that allow it to be implemented efficiently.

Why Racket for CS 275?

All LISP-type languages have lists as the main data structure

- Programs are lists
- Data are lists
- Racket programs can reason about other programs. This makes Racket useful for thinking about programming languages in general.

Racket is a different programming paradigm

- Python, Java, C and other languages are imperative languages. Programs in these languages do their work by changing data stored in variables
- Racket programs can be written as functional programs—they compute by evaluating functions and avoid variable assignments.

Why Racket for CS 275?

Racket is very elegant. It is much less verbose than Java, for instance, which means it is easier to see what is happening in a Racket program.

I think its fun.

It lets you learn functional programming without a lot of extra features.

Next Up!

See the Course Schedule for the Readings

Try to install DrRacket and create a GitHub account by next class

Homework 0 is available now

If you've never used Git/GitHub, please **start ASAP**

- Due Friday, Feb. 7 at 23:59

See you Wednesday!