CSE130: Programming Languages

Winter 2017

Mon&Wed 6:30-7:50 PM

Deian Stefan

Who am I?

- Assistant Professor in CSE
  - First time teaching undergrad class at UCSD
  - Prior to UCSD: PhD at Stanford
- Research: building secure systems
  - Security + PL + Systems
- Industry: startup building secure runtime for Node.js
  - Lots of PL ideas appear in daily work

What is CSE 130 about?

What this course is **not** about?

- Learning how to write...
  - JavaScript in January
  - Haskell in February
  - C++ in March
  - etc.
- Learning C++, JavaScript, etc. to spec

What this course is about

- Concepts in programming languages
  - Fundamentals and core features and building blocks
  - Different programming paradigms and their use
- Design and implementation of languages
  - Goals and trade-offs (with historical context)
  - The cost of a language feature

What this course is about

- Concepts in programming languages
  - Fundamentals and core features and building blocks
  - Different programming paradigms and their use
- Design and implementation of languages
  - Goals and trade-offs (with historical context)
  - The cost of a language feature

Why?

- Concepts in programming languages
  - Language shapes your thinking! Language features dictate how we express ideas and computation
  - E.g., think of error handling in C vs. Java
- Design and implementation of languages
  - Nothing is free: understand what you're giving up and what you're gaining when choosing a language
  - E.g., exception handling, garbage collection, etc.

Why?

- Concepts in programming languages
  - Language shapes your thinking! Language features dictate how we express ideas and computation
  - E.g.,

This program prints “Hello World!”:

```
++++++++[>++++[>++>+++>+++>+<<<<-]>+>+>->>+[<]<-]>>.>---.
+++++++..+++.>>.<-.<.+++.------.--------.>>+.>++.
```

https://en.wikipedia.org/wiki/Brainfuck

- Design and implementation of languages
  - Nothing is free: understand what you’re giving up and what you’re gaining when choosing a language
  - E.g., exception handling, garbage collection, etc.

Why?

- Concepts in programming languages
  - Language shapes your thinking! Language features dictate how we express ideas and computation
  - E.g.,

This program prints “Hello World!”:

```
++++++++[>++++[>++>+++>+++>+<<<<-]>+>+>->>+[<]<-]>>.>---.
+++++++..+++.>>.<-.<.+++.------.--------.>>+.>++.
```

https://en.wikipedia.org/wiki/Brainfuck

- Design and implementation of languages
  - Nothing is free: understand what you’re giving up and what you’re gaining when choosing a language
  - E.g., exception handling, garbage collection, etc.

Why else?

- You can learn any of those languages… once you have a grasp of the fundamentals and understand features
- You'll usually want to use the right lang for the job… this ultimately comes down to what features you need
- You will be able to think about programs differently… since you will understand what's going on underneath
- You will be in better shape to design and implement new languages… great features ➠ great language!

Why else?

- You can learn any of those languages… once you have a grasp of the fundamentals and understand features
- You'll usually want to use the right lang for the job… this ultimately comes down to what features you need
- You will be able to think about programs differently… since you will understand what's going on underneath
- You will be in better shape to design and implement new languages… great features ➠ great language!

Why else?

- You can learn any of those languages… once you have a grasp of the fundamentals and understand features
- You'll usually want to use the right lang for the job… this ultimately comes down to what features you need
- You will be able to think about programs differently… since you will understand what's going on underneath
- You will be in better shape to design and implement new languages… great features ➠ great language!

Why else?

- You can learn any of those languages… once you have a grasp of the fundamentals and understand features
- You'll usually want to use the right lang for the job… this ultimately comes down to what features you need
- You will be able to think about programs differently… since you will understand what's going on underneath
- You will be in better shape to design and implement new languages… great features ➠ great language!

I’ll be working on languages?

- Lots of systems have their own languages or have a language runtime system at their core:
  - Editors (Lisp for Emacs, JavaScript for Atom)
  - DBs (SQL, MongoDB’s JavaScript, …)
  - PDF viewers (JavaScript!?)
- PL is hot! Likely to work on something new in industry

Flow, React @ Facebook Rust, Emscripten @ Mozilla, TypeScript @ Microsoft Swift @ Apple CUDA @ NVIDIA

I’ll be working on languages?

- Lots of systems have their own languages or have a language runtime system at their core:
  - ➤ Editors (Lisp for Emacs, JavaScript for Atom)
  - ➤ DBs (SQL, MongoDB’s JavaScript, …)
  - ➤ PDF viewers (JavaScript!?)
- PL is hot! Likely to work on something new in industry

Flow, React @ Facebook Rust, Emscripten @ Mozilla,
TypeScript @ Microsoft Swift @ Apple CUDA @ NVIDIA

If nothing else...

You can put Haskell on your resume!

Syllabus: The great ideas [Ramsey]

Expressive power (say more with less)

First-class functions	Pattern matching
Type inference	Exception handling
Monads	Continuations

Reliability and reuse

Type polymorphism	Type classes
Modules	Objects & inheritance

Cross-cutting concerns

Memory management	Concurrency

Syllabus: The great ideas [Ramsey]

Expressive power (say more with less)

First-class functions	Pattern matching
Type inference	Exception handling
Monads	Continuations

Reliability and reuse

Type polymorphism	Type classes

Feb 22

Modules	Objects & inheritance

Cross-cutting concerns

Memory management	Concurrency

Syllabus: The great ideas [Ramsey]

Expressive power (say more with less)

First-class functions	Pattern matching
Type inference	Exception handling
Monads	Continuations

Reliability and reuse

Type polymorphism	Type classes

Feb 22

Modules	Objects & inheritance

Cross-cutting concerns

Memory management	Concurrency

Mar 22

Logistics & course mechanics

Contact information

- Course website: http://cse130.programming.systems
  - ➤ Goto place for links and resources
- Piazza: https://piazza.com/ucsd/winter2017/cse130
  - ➤ Use this for general discussions and questions
- Staff email: ucsd-cse130-winter17@googlegroups.com
  - ➤ Use this if you need to get in touch with us directly

Contact information

- Course website: http://cse130.programming.systems
  - Goto place for links and resources
- Piazza: https://piazza.com/ucsd/winter2017/cse130
  - Use this for general discussions and questions
- Staff email: ucsd-cse130-winter17@googlegroups.com
  - Use this if you need to get in touch with us directly

Contact information

- Course website: http://cse130.programming.systems
  - ➤ Goto place for links and resources
- Piazza: https://piazza.com/ucsd/winter2017/cse130
  - ➤ Use this for general discussions and questions
- Staff email: ucsd-cse130-winter17@googlegroups.com
  - ➤ Use this if you need to get in touch with us directly

Logistics: Lectures & Section [5%]

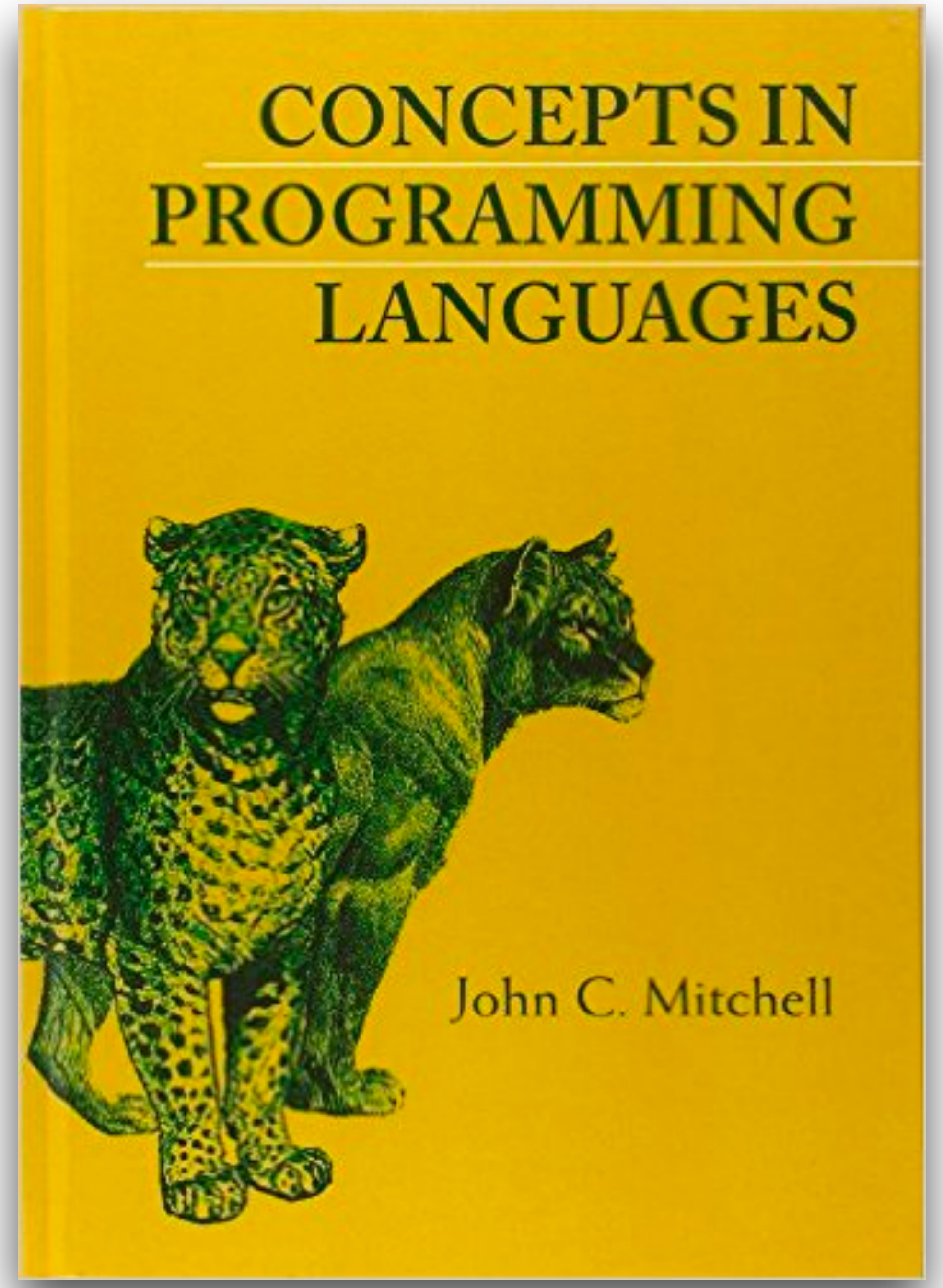
- Lectures: Mondays and Wednesdays
  - We will assign reading before every class
  - Come prepared, bring clickers: we will ask questions during lecture
- Section: Fridays
  - Come to section with questions!
  - Goal: go over course material and problems similar to those assigned for homework

Logistics: Lectures & Section [5%]

- Lectures: Mondays and Wednesdays
  - ➤ We will assign reading before every class
  - ➤ Come prepared, bring clickers: we will ask questions during lecture
- Section: Fridays
  - ➤ Come to section with questions!
  - ➤ Goal: go over course material and problems similar to those assigned for homework

Assigned reading from:

- Course textbook
  - Concepts in Programming Languages by John Mitchell
  - Renting: cheaper option
  - We'll be distributing new Chapters
- Papers & online resources
  - Usually optional, but useful!



Logistics: Homework [30%]

- Homework: weekly
  - Due: 1 week from the release date
  - Submit solution in **groups of 3** (but try to do it on your own first!) using online tool
- Programming assignments: roughly one every 2 weeks
  - Due: 2 weeks from the release date
  - Submit solution **by yourself** using online tool

Logistics: Homework [30%]

- Homework: weekly
  - Due: 1 week from the release date
  - Submit solution in **groups of 3** (but try to do it on your own first!) using online tool
- Programming assignments: roughly one every 2 weeks
  - Due: 2 weeks from the release date
  - Submit solution **by yourself** using online tool

Late policy: 7 late days

- No questions asked
- Can be used for homework or programming assignment
- Used in whole: late by 5mins = used up 1 day
- Can’t use more than 1 day for an assignment
- Make sure everybody in your group has late days if you’re going to hand something in late!

Exams [65%]

- Midterm exam: Feb 22, in class [30%]
  - Date may change depending on progress (unlikely)
  - Can screw up; we'll compute your score as:

    ```
    midterm > 0 ?  max(final, midterm) : 0
    ```
- Final exam: March 22, location and time TBA [35%]

Summary: grading breakdown

- Participation: 5%
  - ➤ In class, with clickers + answering questions online
- Homeworks: 30%
  - ➤ All worth same amount, take each seriously
- Exams: 65%
  - ➤ Must show up to both exams to pass class

Collaboration policy

- Talk with each other, talk on Piazza, use resources
  - Collaboration is a good thing! Just credit the person or resource in you submission
- That said: I expect you to turn in your own work
  - Don’t discuss particularities of a solution with others
  - Don’t ask for a solution on StackOverflow and the like
  - See academic integrity statement

Collaboration policy

- Talk with each other, talk on Piazza, use resources
  - ➤ Collaboration is a good thing! Just credit the person or resource in you submission
- That said: I expect you to turn in your own work
  - ➤ Don’t discuss particularities of a solution with others
  - ➤ Don’t ask for a solution on StackOverflow and the like
  - ➤ See academic integrity statement

Academic integrity, conduct, etc.

- Goal: welcoming class where all can learn and feel included, safe, healthy
  - I don’t want to run the class like a police state, but these two rules will be enforced: these matter even once you graduate!
  - Eat, sleep, take care of your health
  - Talk to me if you’re concerned

Feedback wanted!

- First time teaching this class at UCSD
  - How’s the pace?
  - Are there particular topics you want to spend more time on?
  - How difficult/interesting are the homeworks?
  - What can I do to make your learning experience better?
- We’ll ask for formal feedback, but feel free to send it before we do

Questions?