CS162 Programming Languages

Join CS 162 discord today!

https://discord.gg/NH65rNBW

Who Am I?

Junrui /june-ray/

CS PhD student (4th year)

❤️ programming languages (PL)

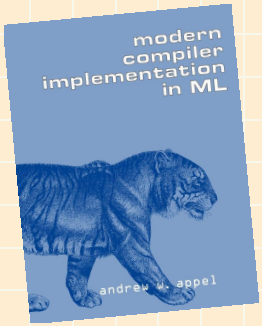
💚 music, anime, video games

Who is your TA?

Jiaming Shan

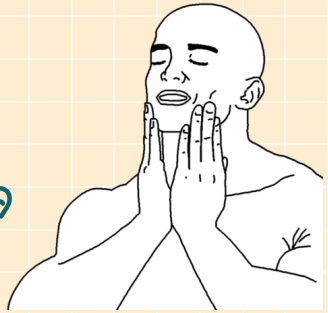
- 1st year PhD student in CS
- Researches AI for programming languages and formal verification
- Likes digital card games and rhythm games

My journey into PL



Took compilers

Learned OCaml + functional programming



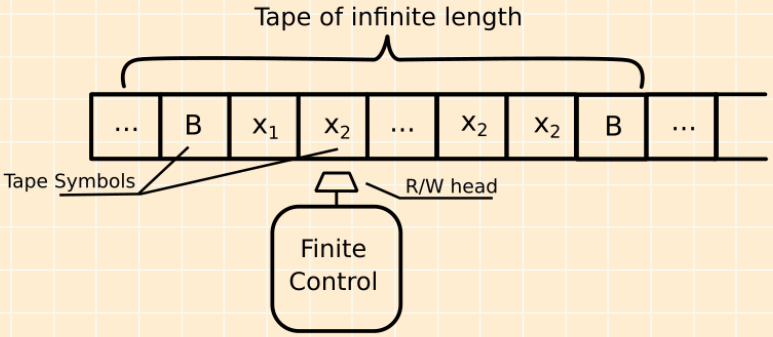
Give me moooore!

PL research

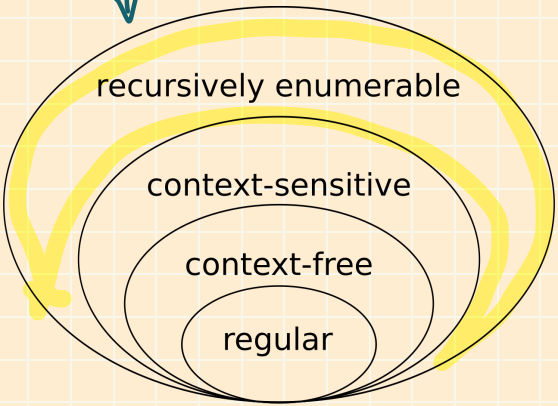
Jane Street

Why PL is exciting!

↳ Languages shape our thoughts.



C++, Python, assembly



Why does nobody (except complexity theorists) program in Turing machines?

Why PL is exciting!

↳ Languages shape our thoughts.

We reach for the most convenient abstraction provided by a language.

C: loops, arrays, pointers
C++: pointers, classes, templates
Assembly: bit-op, stack, jumping
Python: list, dictionary

What if the language doesn't provide the "right" abstraction?

"Wait, you can do that?"

```
void qsort(int a[], int lo, int hi)
{
  int h, l, p, t;
  if (lo < hi) {
    l = lo;
    h = hi;
    p = a[hi];

    do {
      while ((l < h) && (a[l] ⩽ p))
          l = l+1;
      while ((h > l) && (a[h] ⩾ p))
          h = h-1;
      if (l < h) {
          t = a[l];
          a[l] = a[h];
          a[h] = t;
      }
    } while (l < h);

    a[hi] = a[l];
    a[l] = p;

    qsort( a, lo, l-1 );
    qsort( a, l+1, hi );
  }
}
```

QuickSort in 3 LOC??

"Wait, you can do that?"

```
quicksort []     = []
quicksort (p:xs) =
  (quicksort lesser) ++ [p] ++ (quicksort greater) where
      lesser  = filter (< p) xs
      greater = filter (≥ p) xs
```

```
void qsort(int a[], int lo, int hi)
{
  int h, l, p, t;
  if (lo < hi) {
    l = lo;
    h = hi;
    p = a[hi];

    do {
      while ((l < h) && (a[l] ⩽ p))
          l = l+1;
      while ((h > l) && (a[h] ⩾ p))
          h = h-1;
      if (l < h) {
          t = a[l];
          a[l] = a[h];
          a[h] = t;
      }
    } while (l < h);

    a[hi] = a[l];
    a[l] = p;

    qsort( a, lo, l-1 );
    qsort( a, l+1, hi );
  }
}
```

QuickSort in 3 LOC??

"Wait, you can do that?"

Concise like Python, safe like Java, and fast like C++??

Memory safety w/o garbage collection??

QuickSort in 3 LOC??

"Wait, you can do that?"

λ-calculus

Concise like Python, safe like Java, and fast like C++??

Variables = Turing-complete??

Exceptions on steroids = time travel??

1930s

2011

Memory safety w/o garbage collection??

Data & computation = 2 sides of same coin??

QuickSort in 3 LOC??

Coding = proving theorems??

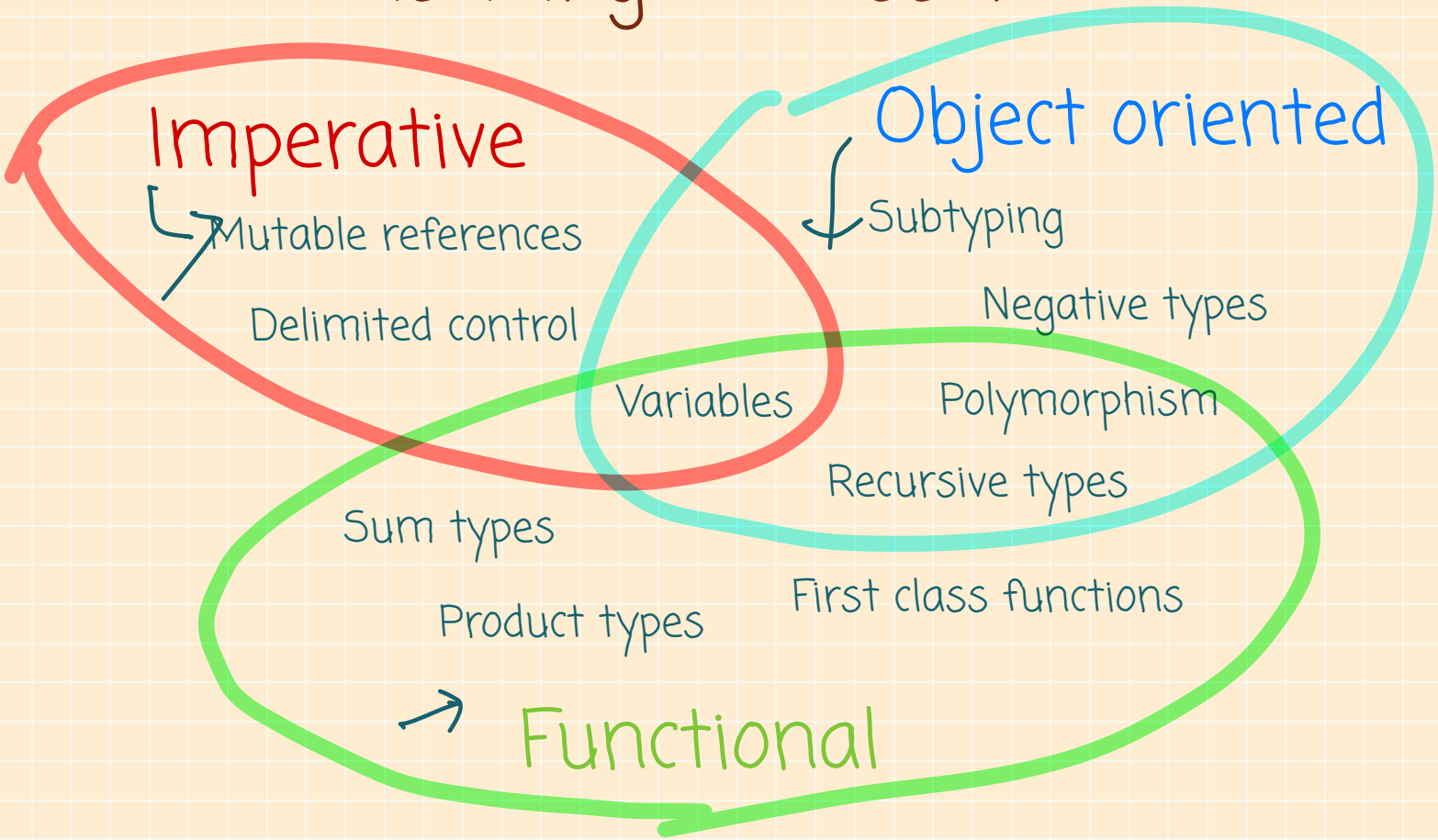
Distilling the Essence

Imperative

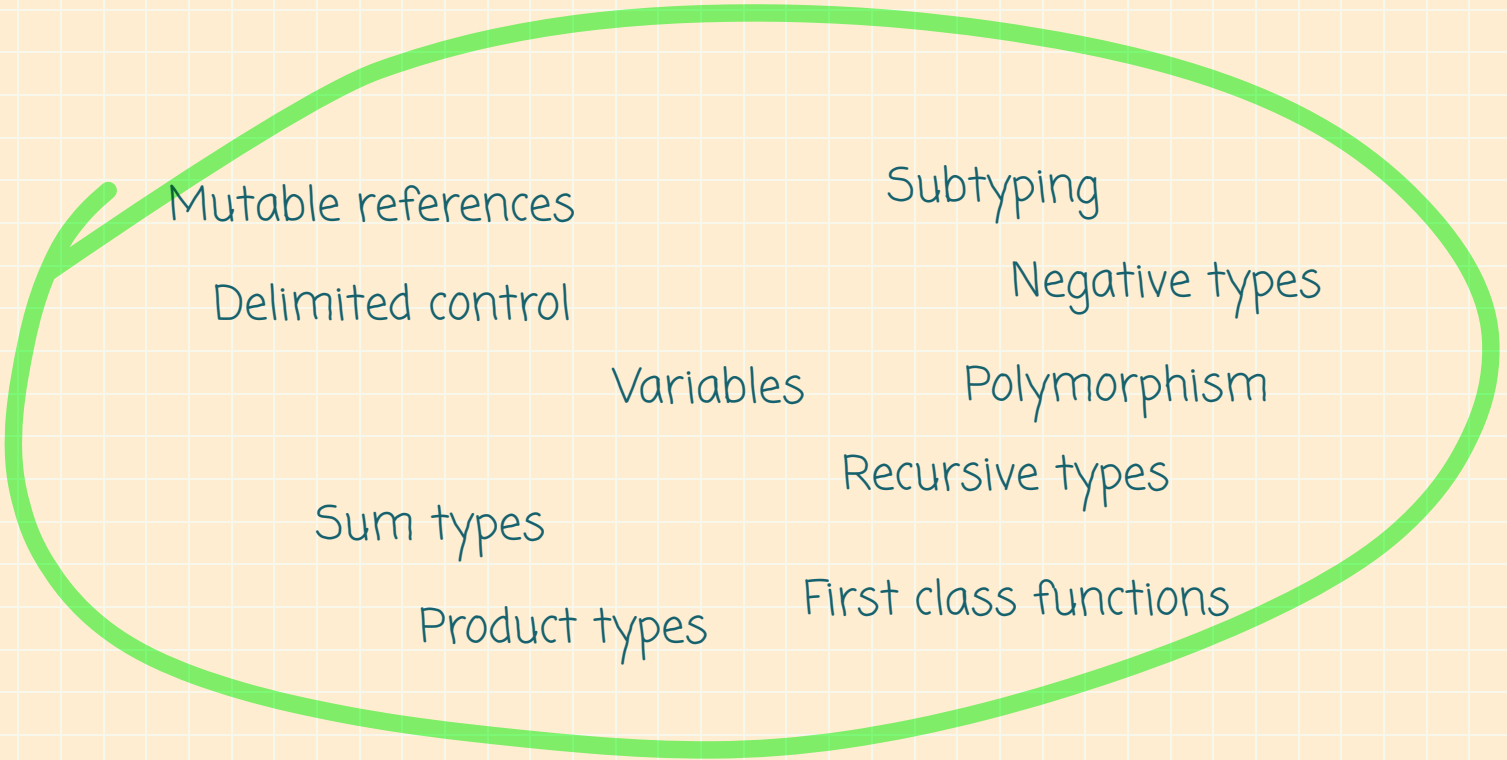
Object oriented

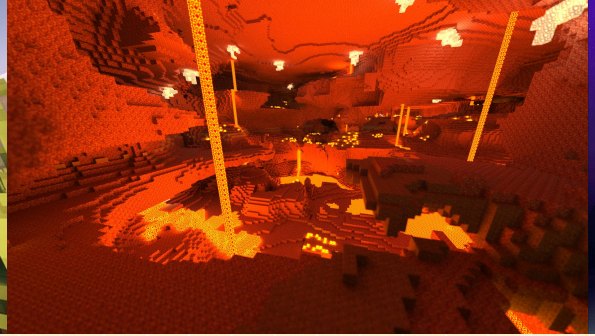
Functional

Distilling the Essence



Distilling the Essence




Programming language = Open-world game

Programmer = Player

Language designer = Game designer

"Wait, I can do that?"

"What about LLMs?"

Shopify CEO says no new hires without proof AI can't do the job

Bloomberg via Getty Images

/ 'Before asking for more Headcount and resources, teams must demonstrate why they cannot get what they want done using AI.'

by **Jay Peters**
Apr 7, 2025, 3:06 PM PDT

79 Comments (79 New)

Jay Peters is a news editor covering technology, gaming, and more. He joined The Verge in 2019 after nearly two years at Techmeme.

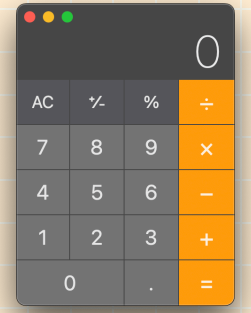
HIGHER EDUCATION

Everyone Is Cheating Their Way Through College ChatGPT has unraveled the entire academic project.

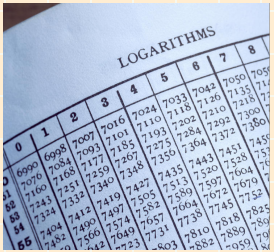
By James D. Walsh, Intelligencer staff writer

"What's the point of studying PL ↩ in the age of AI?"

↳ AI makes PL more relevant, not less!



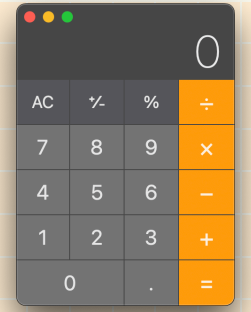
makes



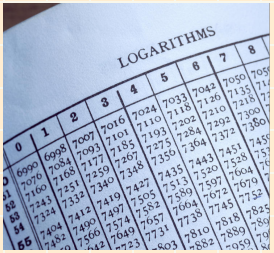
obsolete.

↓

ChatGPT



makes



obsolete.

Imagine a calculator where

- Nobody has a good idea of why it works.
- Randomly outputs completely wrong answers.
- And nobody can predict when that happens.

Would you still trust your calculator?

"Hey, just a heads up, the aviation software in this aircraft was vibe-coded by me 🙂."

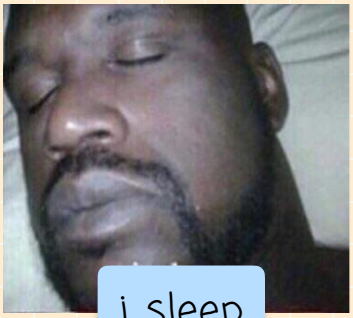
We'll prove this next week!

Theorem: Writing correct code is undecidable.

Corollary: You'll never lose your job to AI,
IF you know how to design correct software.

"If only YOU know how to write correct code..."

Full Title:	Programming Languages
Description:	Concepts of programming languages: scopes, parameter passing, storage management; control flow, exception handling; encapsulation and modularization mechanisms; reusability through genericity and inheritance; type systems; programming paradigms (imperative, object-oriented, functional, and others). Emerging programming languages and their development infrastructures.

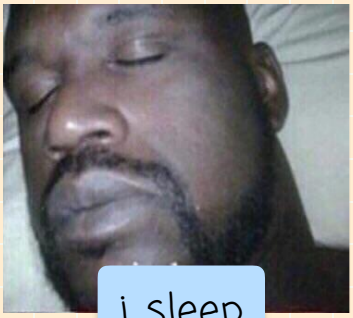


PL designers are obsessed with correctness. Why?

- Q: If a library has a bug, who is affected?
- Q: If a language has a bug, who is affected?

"If only YOU know how to write correct code..."

Full Title:	Programming Languages
Description:	Concepts of programming languages: scopes, parameter passing, storage management; control flow, exception handling; encapsulation and modularization mechanisms; reusability through genericity and inheritance; type systems; programming paradigms (imperative, object-oriented, functional, and others). Emerging programming languages and their development infrastructures.



- Write less buggy code — Typed functional programming
- Reason about code behaviors — Abstract syntax, Operational semantics, Type abstraction
- Prove your system is unhackable — Type soundness, Programs as proofs

The infinite money glitch you've never heard of 🤑

Code needs to be updated

You're hired to fix their shit

AI writes shit code

Company gets sued

Bugs lead to disasters

The infinite money glitch you've never heard of 🤑

AI writes shit code

Code needs to be updated

You're hired to fix their shit

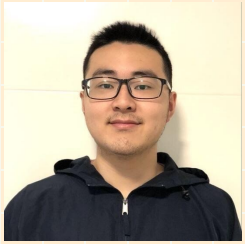
NOT A FICTION !

Bugs lead to disasters

Company gets sued

The infinite money glitch you've never heard of 🤑

- Field called "formal methods", a subfield of PL
- Existed way before LLMs became a thing
- Lots of people made money
- I also made money (thank you, shitty code)
- You can, too - if you pay attention in this class...



- Chaofan (UCSB CS, class of 2022)
- Took CS162, I was a TA
- Went on to found a company called Fuzzland, acquired for an undisclosed amount of $$$

Why you should take this class!

PL is fun!

👾 Languages are like open-world games
🎨 We'll build a simple yet super powerful language together
🎮 You get to be both play & design the game!

PL-thinking matters!

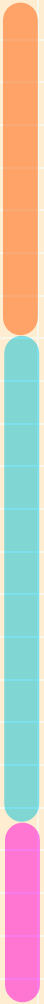
🔥 Learn powerful tools to reason about correctness.
💪 Become irreplaceable in the "AI future"
💰 Get your bags (legal disclaimer: not guaranteed)

A bird's eye view

Foundatioal tools

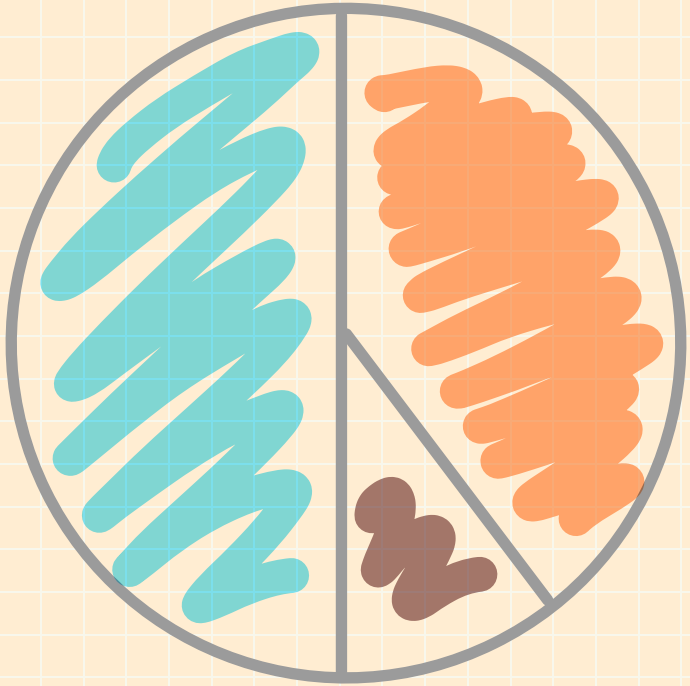
Incrementally adding features

Advanced topics



Date	Topic
Week 1	**How to design a programming language?**
06/24	Why study programming languages? + Python review
06/25	Syntax
06/26	Inference rules
Week 2	**What makes a programming language?**
07/01	Semantics
07/02	Names
07/03	Types
Week 3	**How to abstract *data*?**
07/08	Finite and recursive types
07/09	Pattern-matching
07/10	*Quiz 1 (tentative)*
Week 4	**How to abstract *computation*?**
07/15	Lambda calculus
07/16	Polymorphism, type inference
07/17	Defunctionalization, continuation-passing
Week 5	**How to change the world?**
07/22	Mutable states
07/23	Effect handlers
07/24	*Quiz 2 (tentative)*
Week 6	**What is the future of programming like?**
07/29	Advanced topic, TBD
07/30	Advanced topic, TBD
07/31	Advanced topic, TBD
08/02	**(End of summer session A)**

Assessment



50%

5 HWs

(coding + written problems)

40%

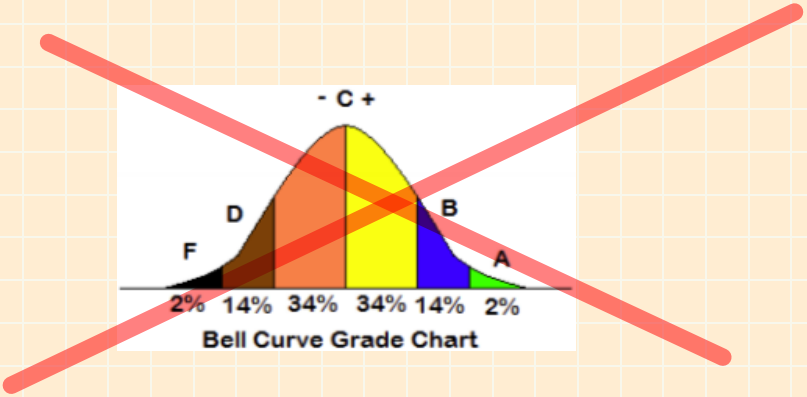
2 quizzes

10%

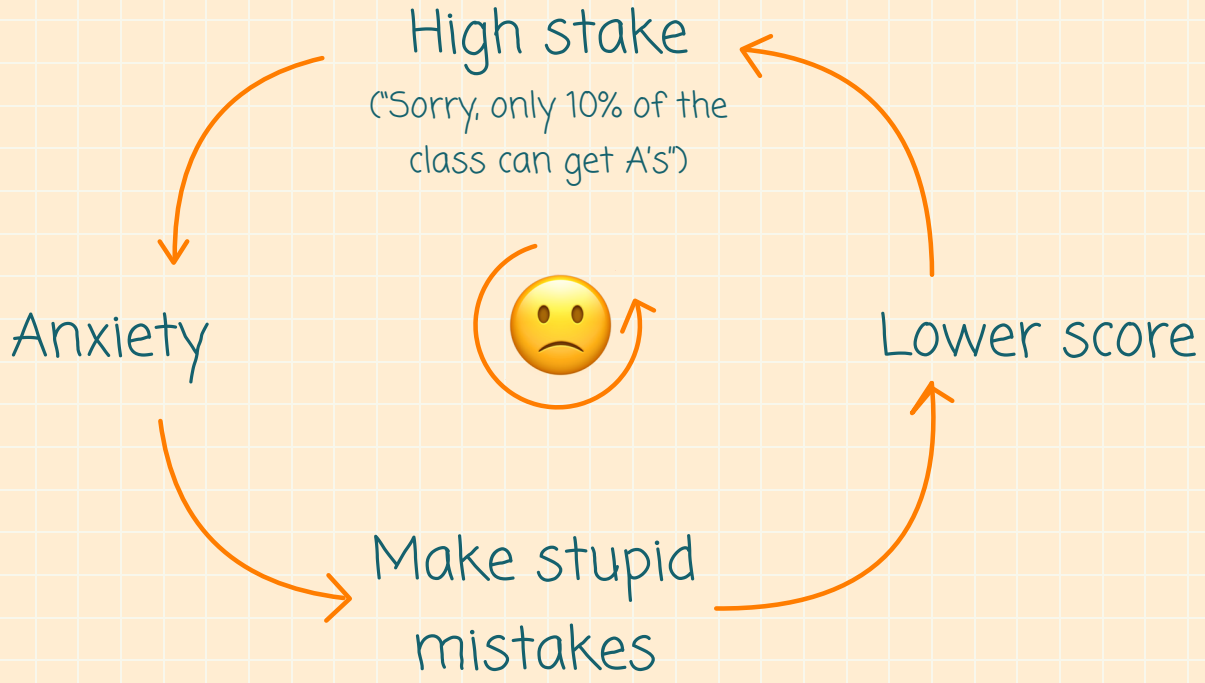
2 reflections

Quizzes (40%)

- Held in class. Tentative dates: week 3 & 5. No makeups.
- Closed book (no notes, devices, or cheat sheet)
- Goal: let me & yourself know how you're doing, so that we can quickly address issues together
- Non-goal: Forcing a bell curve of grades...



Quiz


High stake
("Sorry, only 10% of the class can get A's")
Lower score
Make stupid mistakes
Anxiety

Teaching

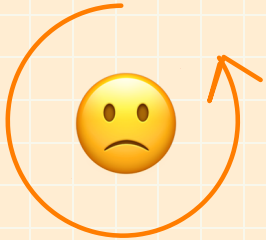
Instructor yaps, students stare at laptops/phones

Instructor freaks out, doesn't know if students are bored or lost

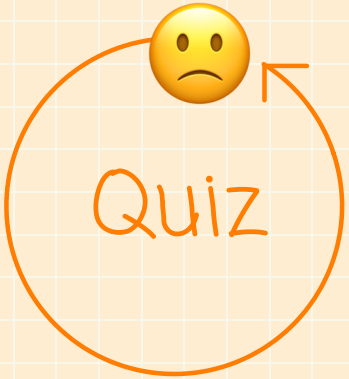
Instructor goes way too fast & makes impossible exams

Students fail, and complain in eval (rightly so)

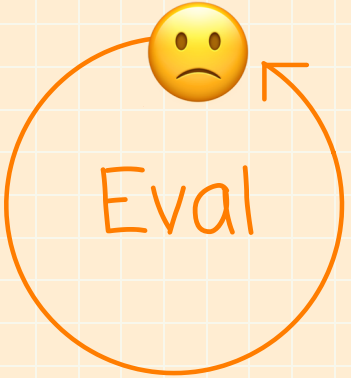
Instructor: "why didn't anybody tell me this earlier?" and gives up



I see some similarities...



- Both are a form of feedback
- Feedback should allow us to improve
- Neither of them is perfect



"Reviewing after a quiz makes no difference!"

Let's make it make a difference.

"Too late to discover the issues only after the quarter ends!"

Let's make it never too late.

Feedback for your learning

Correct your quizzes at the end of the course to get points back.

Example:

- You scored 60/100 in a quiz.
- After the quiz, you review the concepts and redo the problems at home.
- You schedule an appt with me to go over your solution, get back 40 points.

Each correction point consumes a token ⭐ that you possess.

Feedback for my teaching

You earn tokens 💖 of my appreciation by giving me feedback directly or indirectly throughout the course:

- Attending lectures
- Asking questions and participating in lectures
- Coming to office hours
- Filling out surveys
- More...

Finer points

- 🧪 This system is experimental. Expect small tweaks.
- 👊 Tokens are tied to you only, and are non-transferable.
- 😋 Any unconsumed tokens will automatically enter into an end-of-session lottery where you can earn yummy snacks 🍡.
- 🤷‍♀️ Feel free to completely opt out of this system. If you opt out, you won't be able to turn in corrections, just like your average CS class.

How to earn tokens

(💖💖) Fill out a questionnaire/survey.

Starting from now:

(💖) Attend a lecture

(💖💖) 1st time you ask/answer a question in each lecture

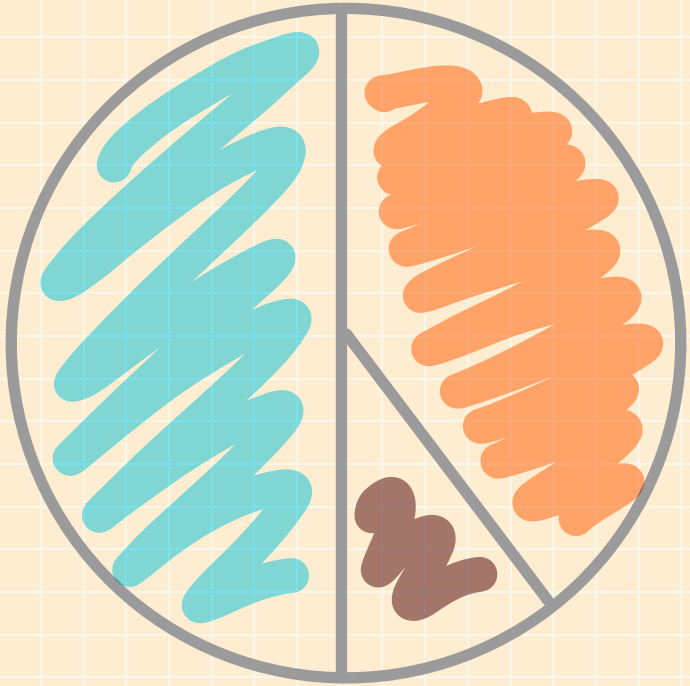
(💖x2) Attend my office hour (every week)

(💖❓) Hidden events. Will be explained when triggered.

More to be added!

Deal?

Assessment



50%

5 HWs

(coding + written problems)

40%

2 quizzes

10%

2 reflections

Assignments - written part 

Goals:

- Show your understanding, so I can give you immediate feedback
- Also give you practice with quiz problems

Self-grade each problem on the following scale

- 2 points = fully solved, work & solution shown
- 1 point = partially solved, work shown
- 0 point = no attempt

I & Jiaming (TA) will cross-check your self grades and give you feedback.

If you're unsure if solution is correct, I'll determine the grade for you

(lean lenient, so be honest!)

Assignments - coding part 👩‍💻

👩‍💻 Coding:

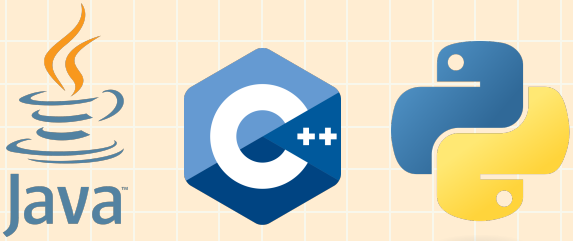
- Incrementally implement a real language in Python
- Goal: check your understanding!
- Short! Each HW is 100-200 LOC, no boilerplate code
- Autograded, unlimited # of tries
- 5 "late days" in total, you decide how to spend HW (2 max) on each HW)
- Done individually (unless explicitly specified otherwise)

Assignments - coding part 👨‍💻

Why Python?

- In PL, we first experiment with designs on paper
- Then translate design -> executable code
- Python syntax ≈ pseudo-code + executable
- Python has garbage collection = no segfault (thank goodness)

Object language = what we design in CS 162



Meta language = how the design gets implemented

Assignments - coding part 👨‍💻

We'll only use a tiny subset of Python:

- Variables, functions, basic arithmetic, if, while
- Lists (similar to C++'s `std::vector`)
- Dictionaries (similar to C++'s `std::map`)
- `@dataclass`
- pattern matching

Come to Friday's discussion section for a Python tutorial by TA Jiaming !!

Collaboration policy

🗣️ Please talk to each other about anything!

✅ DO:

- Discuss ideas for solving coding problems

👊 What you submit must be your own work

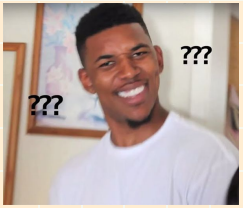
🚫 DONT:

- Copy someone else's work
- Ask LLM / AI to do work for you

General AI / LLM Policy

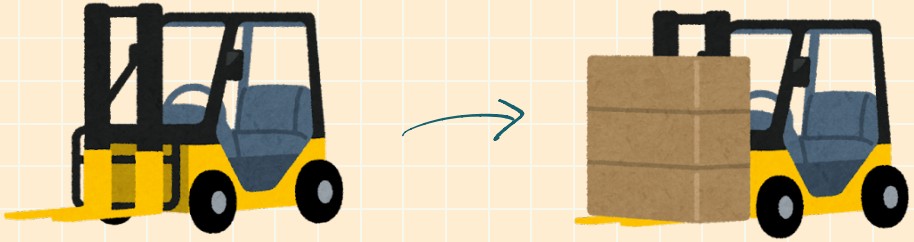
"Although AI doesn't replace senior devs, it'll replace junior devs."

"Where do you think senior devs come from ???"



Learning is like gaining (mental) muscles
You gain muscles via weight training

If AI takes that weight off of you,
no learning occurs !!!



AI / LLM Policy: Don't!

If

1) You're in a time crunch...
   - Use your late days!
   - If it's a personal issue, let's talk!
2) You don't think what you're learning matters...
   - That means I failed at motivating the materials.
   - Please let me know (office hours, surveys, ...)
3) You're stuck & want an easy way out...
   - Learning is hard! But think about what you'll miss out on:
   - Joy 💖, future money 💰, changing how you think 💭

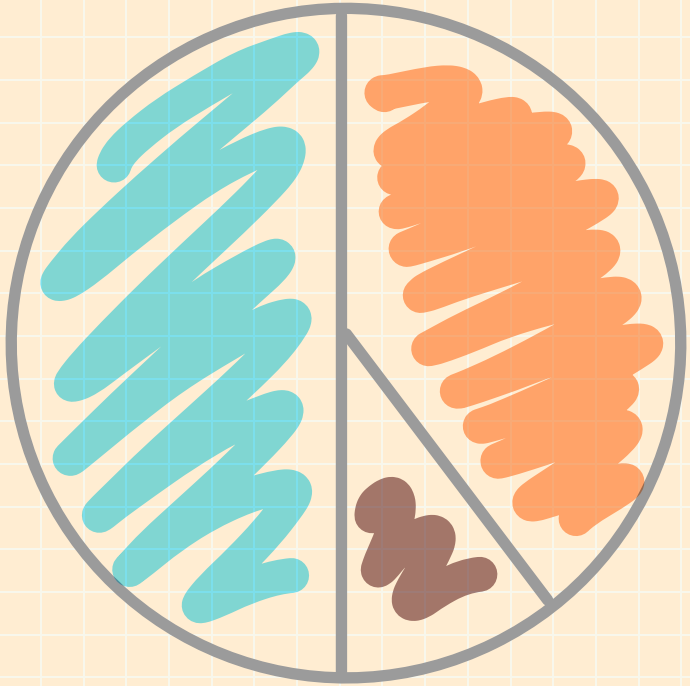
AI / LLM Policy: Don't!

If

1) You're in a time crunch...
   - Use your late days!
   - If it's a personal issue, let's talk!
2) You don't think what you're learning matters...
   - That means I failed at motivating the materials.
   - Please let me know (office hours, surveys, ...)
3) You're stuck & want an easy way out...
   - Learning is hard! But think about what you'll miss out on:
   - Joy 💖, future money 💰, changing how you think 💭

Thank you ❤️ for respecting my work!

Assessment



50%

5 HWs

(coding + written problems)

40%

2 quizzes

10%

2 reflections

Reflections

A lot of PL topics are super important:

- Psychology: how do humans learn PLs?
- Economics: how do PLs get adopted?
- History: why we are where we are today?
- SE & HCI: how do we build better PL tools for everyone?
- Social issues...

But we can have 6 weeks

Reflections (10%)

Each week, I'll share 1-2 YouTube videos on a selected topic. You'll...

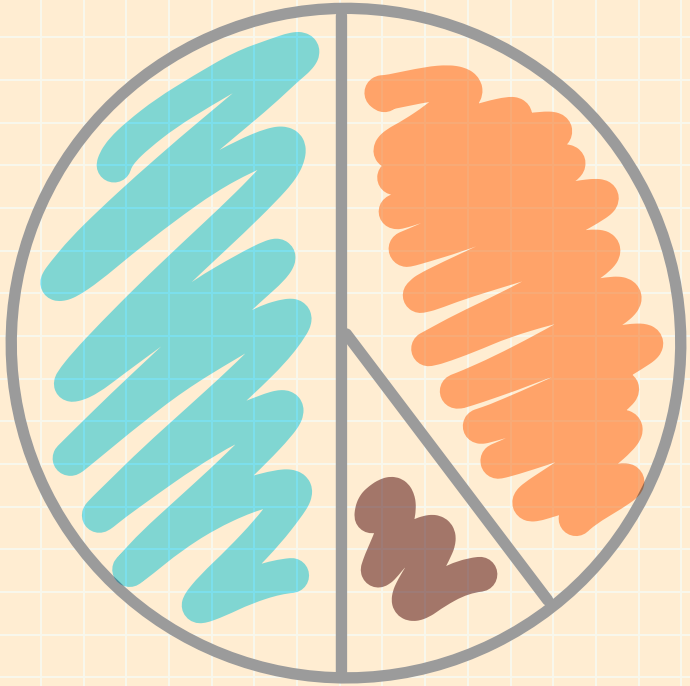
- 📺 watch the video
- ✏️ write a short reflection (1-2 paragraphs)
- 🤗 share your reflection with class
- ❤️ comment on some else's reflection (please be nice!)

You only need to pick 2 weeks to do reflection.

Pick a week if the topic genuinely intrigues you.

- 👌 No need to be a great writer.
- ⛔ Don't use LLM! Just skip the week if you don't feel like doing it.

Assessment



50%

5 HWs

(coding + written problems)

40%

2 quizzes

10%

2 reflections

Heads-up about workload

We only have 6 weeks, compressed from 10 weeks

- UCSB suggested workload for summer: 20 hrs / week
- Even excluding lectures & sections, that's 2 hrs / day of studying

The list of topics is very ambitious.

- By week 6, we'll have caught up with cutting edge PL research
- That's almost 100 years of knowledge
- Previously, CS 162 only covered about 1/2 of them.

Date	Topic
Week 1	**How to design a programming language?**
06/24	Why study programming languages? + Python review
06/25	Syntax
06/26	Inference rules
Week 2	**What makes a programming language?**
07/01	Semantics
07/02	Names
07/03	Types
Week 3	**How to abstract *data*?**
07/08	Finite and recursive types
07/09	Pattern-matching
07/10	*Quiz 1 (tentative)*
Week 4	**How to abstract *computation*?**
07/15	Lambda calculus
07/16	Polymorphism, type inference
07/17	Defunctionalization, continuation-passing
Week 5	**How to change the world?**
07/22	Mutable states
07/23	Effect handlers
07/24	*Quiz 2 (tentative)*
Week 6	**What is the future of programming like?**
07/29	Advanced topic, TBD
07/30	Advanced topic, TBD
07/31	Advanced topic, TBD
08/02	**(End of summer session A)**

Date	Topic
Week 1	**How to design a programming language?**
06/24	Why study programming languages? + Python review
06/25	Syntax
06/26	Inference rules
Week 2	**What makes a programming language?**
07/01	Semantics
07/02	Names
07/03	Types
Week 3	**How to abstract *data*?**
07/08	Finite and recursive types
07/09	Pattern-matching
07/10	*Quiz 1 (tentative)*
Week 4	**How to abstract *computation*?**
07/15	Lambda calculus
07/16	Polymorphism, type inference
07/17	Defunctionalization, continuation-passing
Week 5	**How to change the world?**
07/22	Mutable states
07/23	Effect handlers
07/24	*Quiz 2 (tentative)*
Week 6	**What is the future of programming like?**
07/29	Advanced topic, TBD
07/30	Advanced topic, TBD
07/31	Advanced topic, TBD
08/02	**(End of summer session A)**

How can we possibly be so efficient?

1. Language change

- Previously, programming assignments have to done in OCaml, which takes 2 weeks to learn properly.
- This time, we'll use Python 🤗

2. We don't have to be efficient at all

- Goal is to teach you the "PL way of thinking", not rushing through concepts.
- If I rush, you'll forget what you learned
- If I rush, you'll cram more for quizzes
- You're in charge of slowing me down (because I may get too excited)

Join CS 162 discord today!

https://discord.gg/NH65rNBW

Ask me questions

Let's take a break

Ice breaker

Share with us:

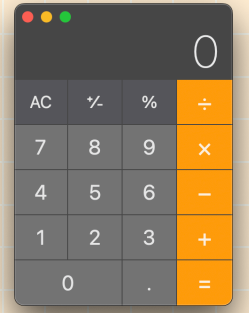
- your preferred name
- pronoun (if you want)
- Major + year
- Where are you from
- A programming language you want to learn next, and why?

CS 162
Program Syntax

- A syntax of a language describes what programs can look like
- In CS 138, you learned about context-free grammars (CFG)
- A CFG determines whether a string belong to the language.

Example: A CFG for the language of calculators:

```
expr ::= NUM
       | - expr
       | expr + expr
       | expr - expr
       | expr * expr
```



expr is called a non-term

NUM, +, -, * are called term

`expr ::= NUM` is called a production / operation

In 162, we call this CFG the concrete syntax of the language

Concrete syntax

determines what programmers can write

In CS 162, we'll also be the language designer / implementer

- assume programs are already written in valid syntax
- want to perform operations on the programs

We need represent programs as data structures!

Programs as data structures

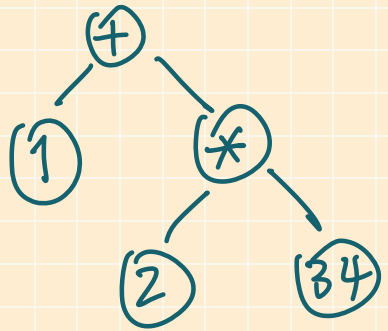
Data Structure	Pros	Cons
Strings	Easy to write	Hard to manipulate

`1+2*34`

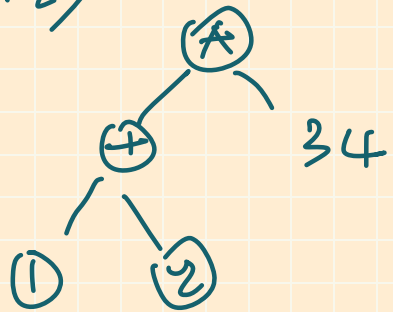
1 + 2 * 3 4

trees

1+(2*34) ←



(1+2)*34



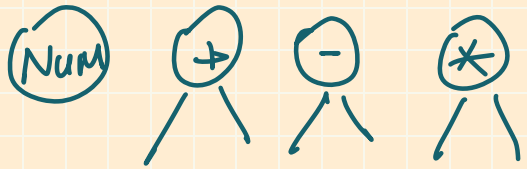
Abstract syntax trees (ASTs)

- represent programs as trees
- we'll also use CFG to describe the tree structure
- This version of CFG is called the abstract syntax

Example: re-interpret the following CFG as abstract syntax:

```
expr ::= NUM
       | expr + expr
       | expr - expr
       | expr * expr
```

[`| - expr`] desugars into "`0 - expr`"



- Every production is a node type
- RHS non-terminals are children nodes
- Potential ambiguities are described verbally
- Kept as small as possible (through desugaring)

Consider the language of Boolean propositional formulas,
like "P -> false /\ (Q \/ true)"

Exercise 1: Design a CFG to describe the ASTs of this language.

Exercise 2: draw the AST corresponding to "P -> false /\ (Q \/ true)"

prop ::= BOOL | VAR ← variables
| prop → prop
| prop ∧ prop
| prop ∨ prop
| (prop) ← not needed in abstract syntax since trees are already unambiguous