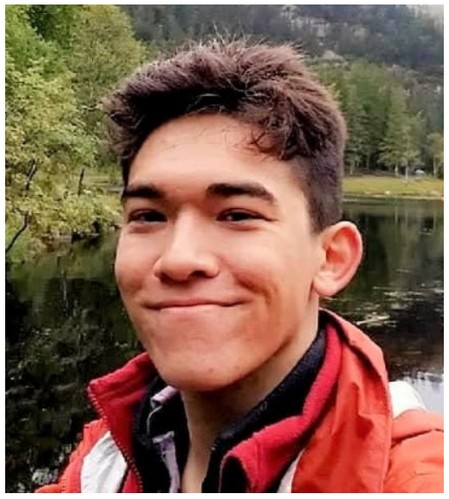
CS 242

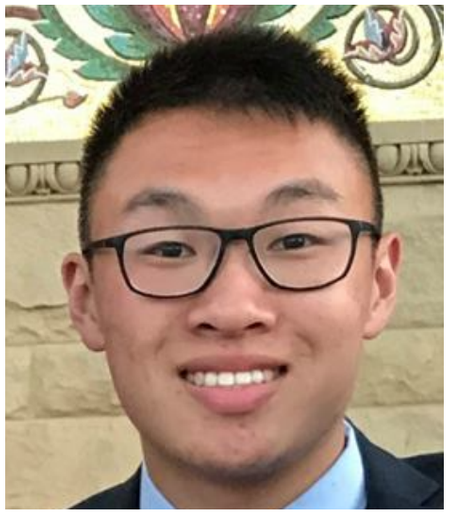
Programming Languages

Course staff



John

Will



Jintian

Varun

Today's goals

- What is a programming language?
- How do we learn about them?
- Why is the study of PLs important?

What is a programming language?

"A vocabulary and set of grammatical rules for instructing a computer to perform specific tasks."

- *Fundamental of Programming Languages* (Ellis Horowitz)

"A programming language is a notation for writing programs, which are specifications of a computation or algorithm."

- Wikipedia

"Programming languages are the medium of expression in the art of computer programming."

- *Concepts in Programming Languages* (John Mitchell)

"A good programming languge is a conceptual universe for thinking about programming".

- Alan Perlis

When in doubt: majority vote!

My proposed definitions

- **Programming model**
  A precise, composable specification of things

- **Programming paradigm**
  Common properties of models

- **Programming language**
  Syntax for expressing a programming model

- **Program**
  Instance of: a model (abstractly) or a language (concretely)

Definitions matter because they shape understanding and direction

Definition	→	Value	# Discoveries
PL is math	→	certainty	
PL is interface	→	efficiency	
PL is design	→	utility	
PL is notation	→	sharing	
PL is media	→	expression	
PL is power	→	control	
PL is language	→	exchange	
PL is communication	→	understanding	
PL is glue	→	connection	
PL is legalese	→	promise	
PL is infrastructure	→	fellowship	
PL is path	→	equity	

Andy Ko, SPLASH 2016

This course covers *general-purpose* PLs

- "Turing-complete," but that's not a useful standard
- Described through abstractions over *data* and *control*
  - Abstraction: means of hiding complexity via interfaces
  - Data: information about things and their relationships
  - Control: producing new data and interacting with outside world
- Abstractions chosen based on:
  - Mapping to underlying resources
  - Ease of understanding for humans
  - Distance to "actual" description of the problem

Prolog demo

Complete (unsafe) control, little data

```
_main:
   pushq %rbp
   movq  %rsp, %rbp
   subq  $16, %rsp
   leaq  L_.str(%rip), %rdi
   movb  $0, %al
   callq _printf
   xorl  %ecx, %ecx
   movl  %eax, -4(%rbp)
   movl  %ecx, %eax
   addq  $16, %rsp
   popq  %rbp
   retq
```

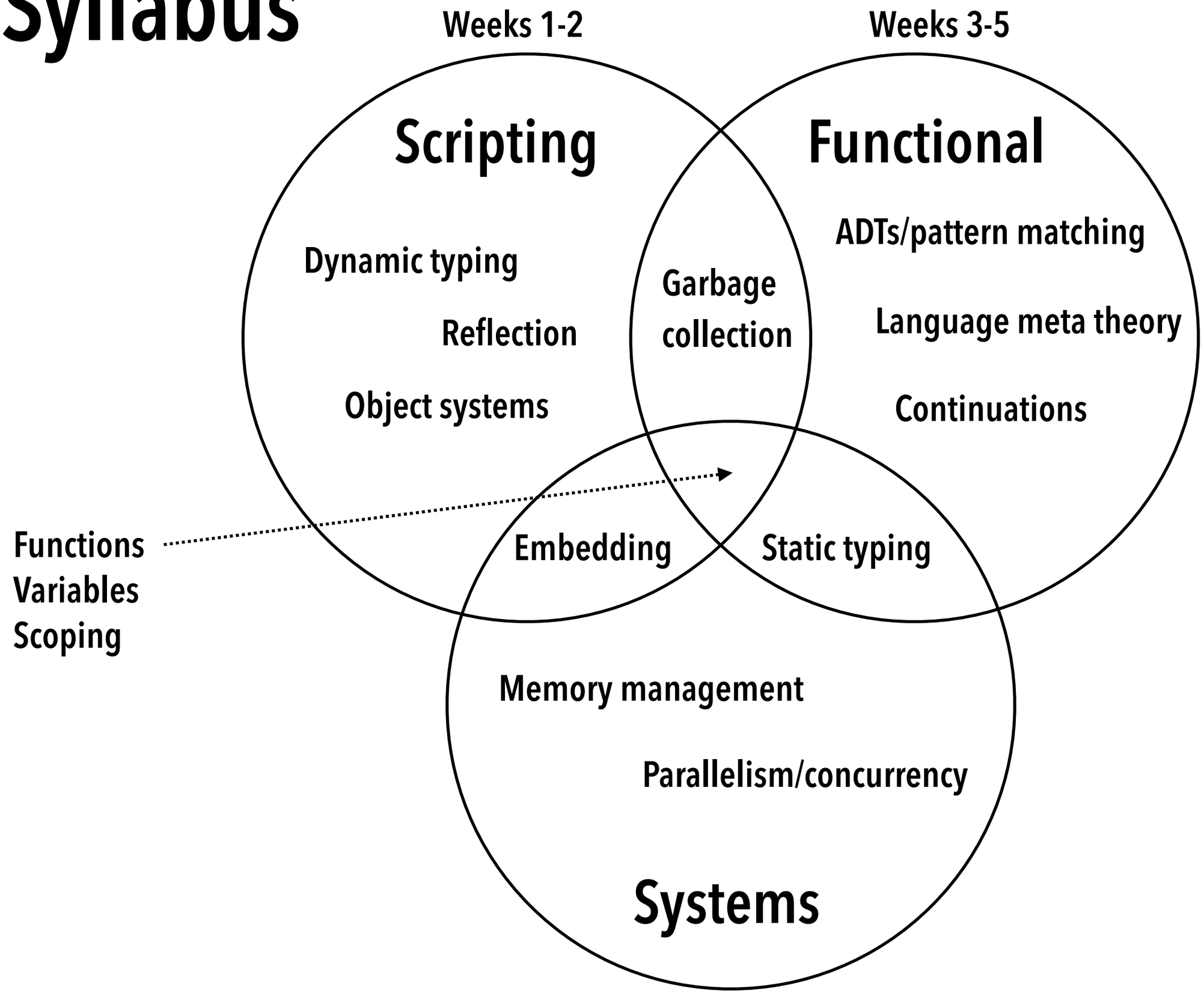
A language is also its…

- Compiler
- Package manager
- Debugger
- Libraries

Course goals

- Understand the concepts underlying modern PLs
  - Distinguish syntax from semantics, language from model
  - View the world in diffs: "it's just X but with Y"
- Explore the tradeoffs in common design decisions
  - Scripting languages are expressive, but hard to debug and maintain
  - Functional languages are safe, but hard to program
  - Systems languages are fast, but don't map to the problem domain
- Learn by doing: both use and implement language features
  - Assignments are mostly coding

Syllabus


Weeks 1-2
Weeks 3-5
Scripting
Functional
Dynamic typing
Reflection
Object systems
Garbage collection
ADTs/pattern matching
Language meta theory
Continuations
Functions
Variables
Scoping
Embedding
Static typing
Memory management
Parallelism/concurrency
Systems
Weeks 6-7

Course structure

- Programming assignment every week (70%)
  - Released Wednesday evening after class, due 4:20pm next Wednesday
  - Learning three new languages, so start early to iron out logistics
- No midterm
  - Assignments are a little bit harder to compensate
- Final project, not an exam (30%)
  - More details later in the semester
  - Final exam slot will be used for presentations

Expected prerequisites

- **Required: moderate programming experience**
  - Know well at least one general-purpose language (C, Python, Java, …)
  - Also assume CS 107 level of systems knowledge

- **Required: basic logic**
  - First order logic (boolean algebra, quantifiers)
  - You've written proofs before, know what induction is

- **Recommended: command line experience**
  - Makes your life easier dealing with different programming environments

Tech stack

- Course website: cs242.stanford.edu
  - Lecture slides, assignment handouts
- Announcements/assignment help: Piazza
- Grades: Gradescope
- Assignment submission: FarmShare2 cluster

For next lecture

- Follow the Lua installation guide
- Come to lecture with your laptop and editor at the ready

Why study PLs?

- Everything old is new again
  - Declarative programming
  - Type inference
  - Algebraic data types
  - Closures/lexical scoping

- Entering an era of domain-specific languages (DSLs)
  - Big data: Spark, TensorFlow, Halide, …
  - Interfaces: HTML, LaTeX, jQuery, React, D3, …