Welcome to CS106B: Programming Abstractions!

What's your hometown?

Respond at PollEv.com/jennyhan903

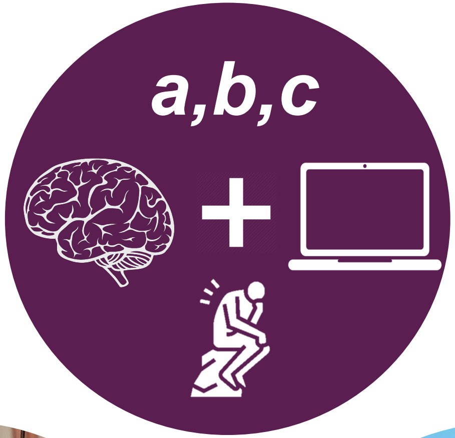
Our CS106B Hometowns

Powered by Poll Everywhere

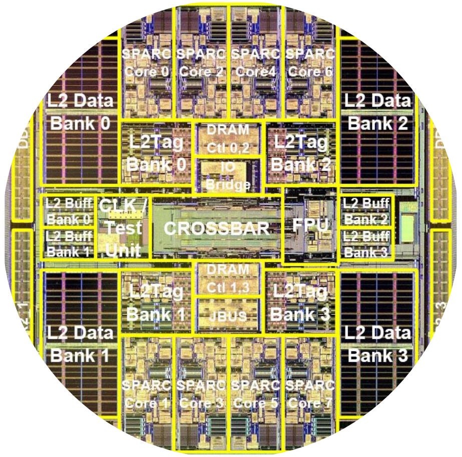
Who are we?

Kylie Jue

Jenny Han



Trip Master


SPARC Core 0
SPARC Core 2
SPARC Core4
SPARC Core 6
L2 Data Bank 0
L2 Data Bank 2
L2Tag Bank 0
DRAM Ctl 0,2
IO Bridge
L2Tag Bank 2
L2 Buff Bank 0
L2 Buff Bank 1
CLK / Test Unit
CROSSBAR
FPU
L2 Buff Bank 2
L2 Buff Bank 3
L2Tag Bank 1
DRAM Ctl 1,3
JBUS
L2Tag Bank 3
L2 Data Bank 1
L2 Data Bank 3
SPARC Core 1
SPARC Core 3
SPARC Core 5
SPARC Core 7

Today's questions

Why take CS106B?

What is an abstraction?

What is CS106B?

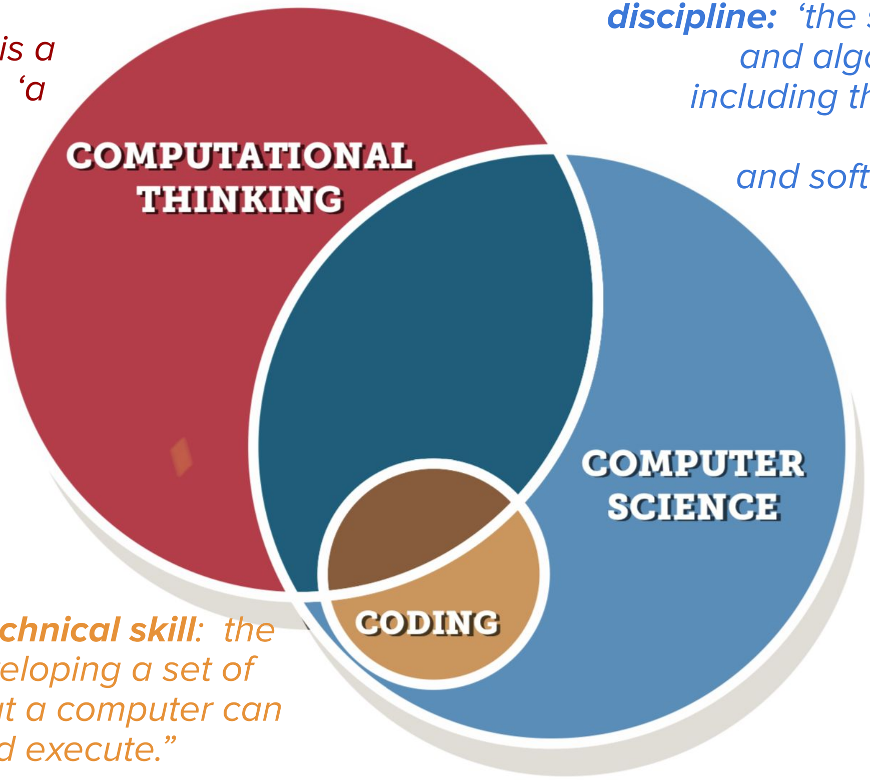
Why C++?

What's next?

Why take CS106B?

Defining key terms

"***Computational thinking*** *is a* ***problem solving process:*** *'a way of solving problems, designing systems, and understanding human behavior that draws on concepts fundamental to computer science... a fundamental skill for everyone, not just computer scientists'"*



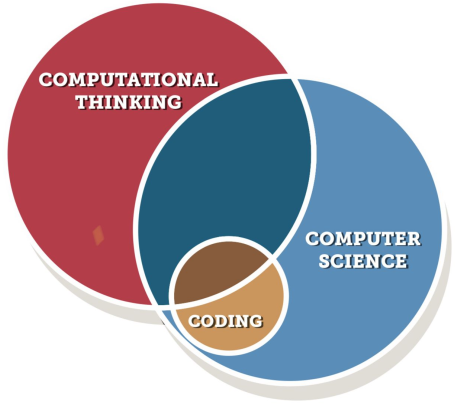
"***Computer science*** *is an* ***academic discipline:*** *'the study of computers and algorithmic processes, including their principles, their hardware and software designs, their applications, and their impact on society'"*

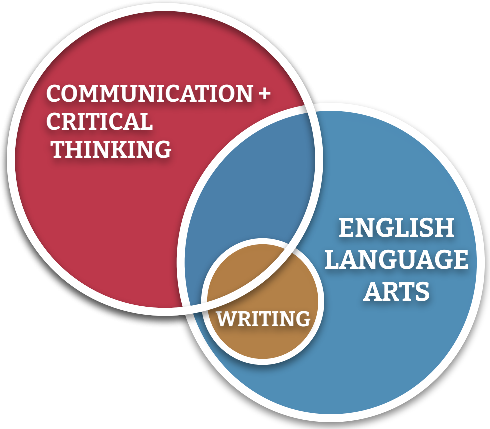
"***Coding*** *is a* ***technical skill****: the practice of developing a set of instructions that a computer can understand and execute."*

(Digital Promise 2017)
(Wing, 2006)

Defining key terms

- **Coding** as a technical skill
- **Computer science** as an academic discipline
- **Computational thinking** as a problem-solving process





CS education is more than just "learning how to code"!

Phases of language development

1. Discovery that language is a pattern of sounds that takes on meaning and purpose

2. Participation in everyday social aspects of language that enable an understanding of encoded cultural values and assumptions

3. Ability to self-reflect on the use of language and to see language as a “tool for thinking” and communicating thoughts, even when not actively speaking or interacting with others

the acquisition of literacy

(Wells 1981)

What CS106B *is not*

- A course to teach you how to program from scratch
- A course that will teach you the specifics of the C++ language

What CS106B *is*

- A logical follow-up course to an introductory computer science class
- A course that will give you practice with computational thinking skills through basic C++ coding
- A survey of data structures and algorithms to prepare you for future exploration in computing and to build your understanding of technology

What is an abstraction?

What is an abstraction?

Talk to a neighbor! What comes to mind when you think of the word abstraction?

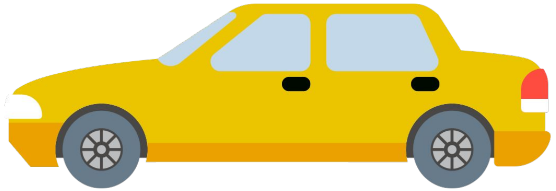
Definition

abstraction

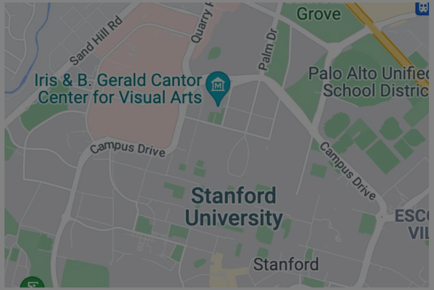
Design that hides the details of how something works while still allowing the user to access complex functionality

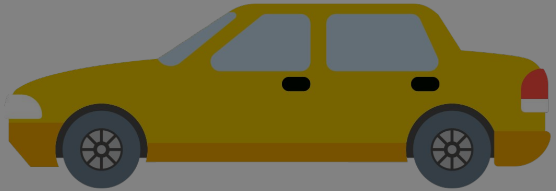
(ie. design that makes complex systems simple to use)

Examples of abstraction

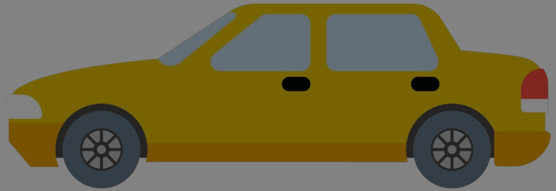


Examples of abstraction

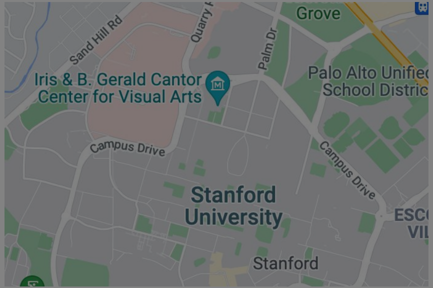


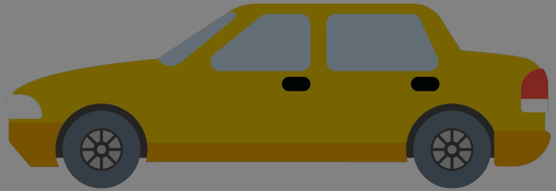


Examples of abstraction



Examples of abstraction



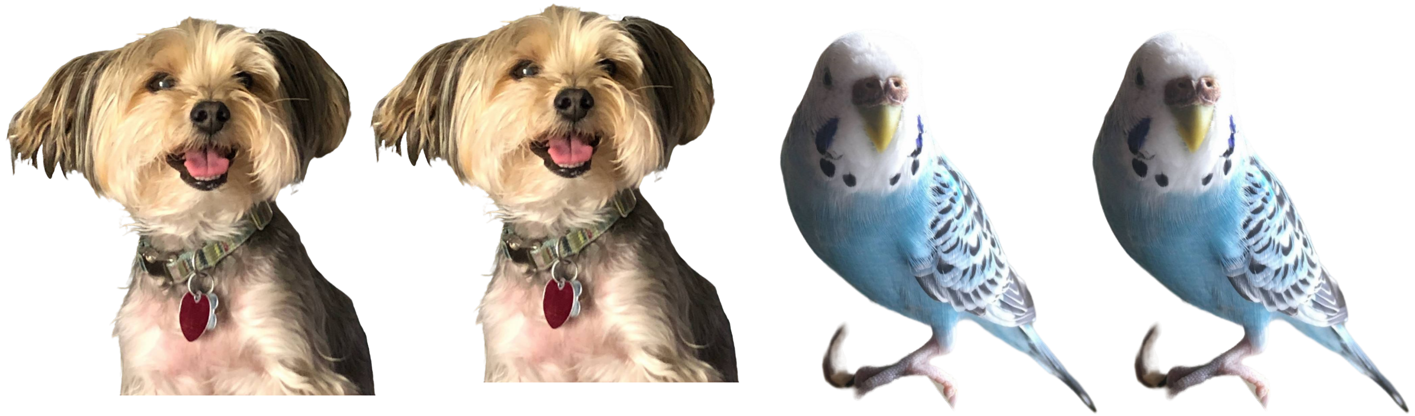


Key idea

Abstractions are tools to help us solve complex problems!

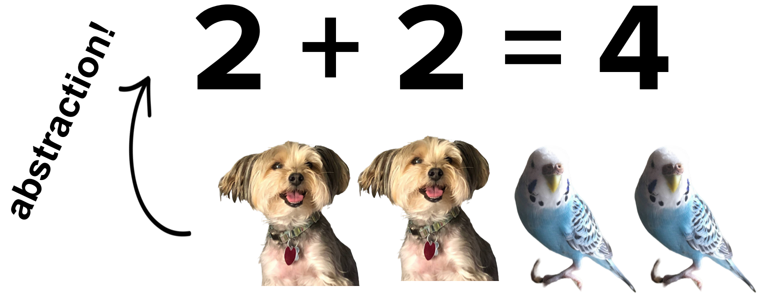
Through a simpler interface, users are able to take full advantage of a complex system without needing to know how it works or how it was made.

Complex problem: count the number of animals



Abstractions are tools (for your brain!)

- Numbers are abstract representations.
- Addition is an algorithm that helps us count things.



If we didn’t have numbers as abstractions….

I’d have to show you 100 objects every time I wanted to express the idea of “100”

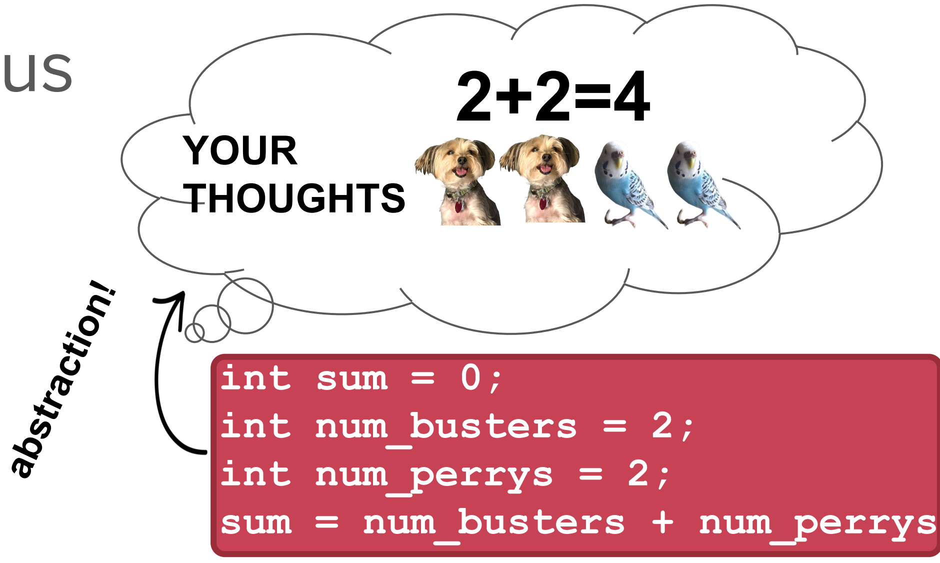
...is everything an abstraction?
yes
numbers
the universe
knowledge
language
emotions
love

Key idea

Abstractions are tools to help us solve complex problems!

We built computers to help us solve complex problems.

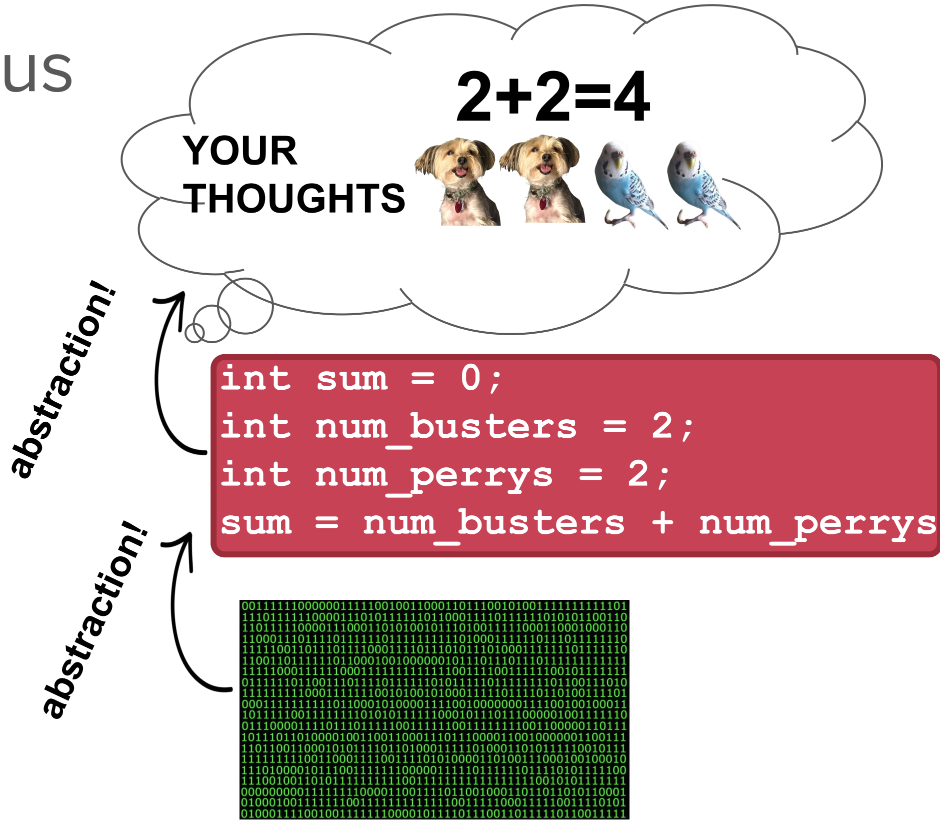
- We use programming languages as an abstraction to help us communicate our thoughts to computers.



```
int sum = 0;
int num_busters = 2;
int num_perrys = 2;
sum = num_busters + num_perrys
```

We built computers to help us solve complex problems.

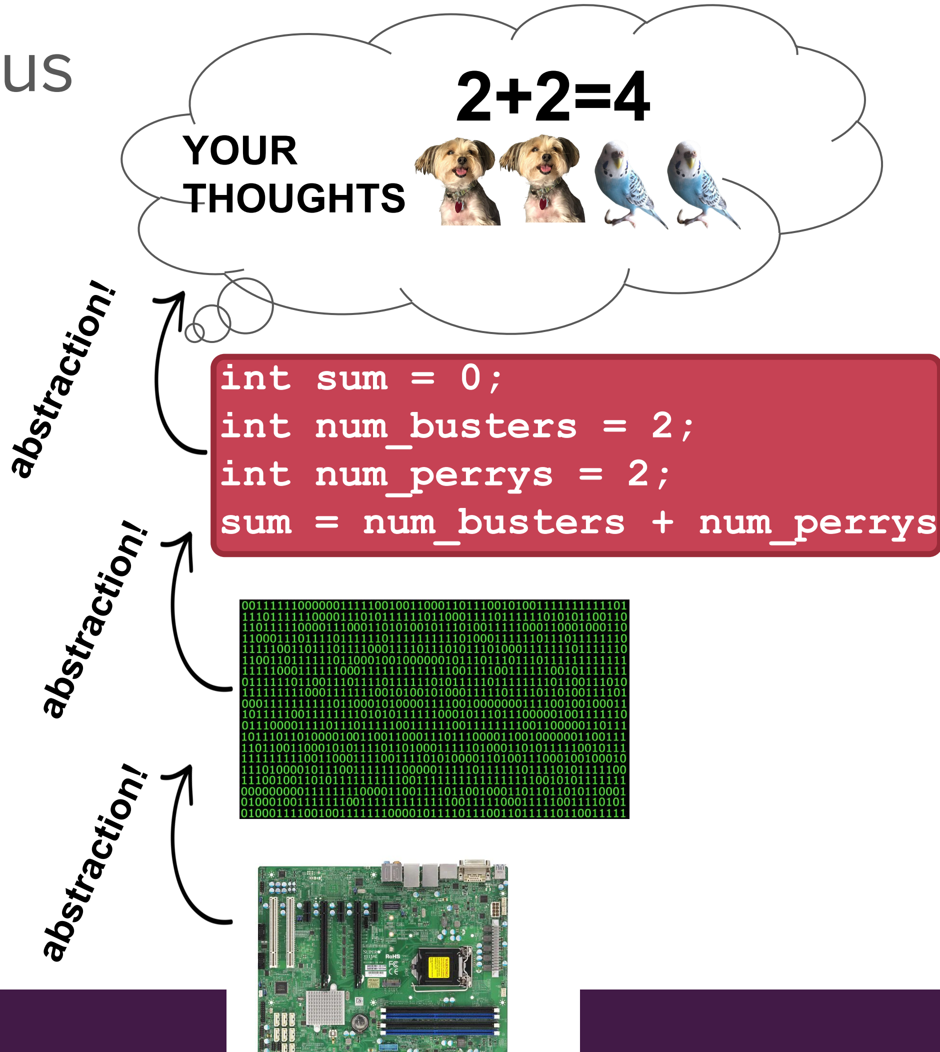
- We use programming languages as an abstraction to help us communicate our thoughts to computers.
- Programming languages are an abstraction for digital bits - 0s and 1s that help computers represent everything



```
int sum = 0;
int num_busters = 2;
int num_perrys = 2;
sum = num_busters + num_perrys
```

We built computers to help us solve complex problems.

- We use programming languages as an abstraction to help us communicate our thoughts to computers.
- Programming languages are an abstraction for digital bits - 0s and 1s that help computers represent everything
- 0s and 1s are abstractions for tiny physical switches in our computer.

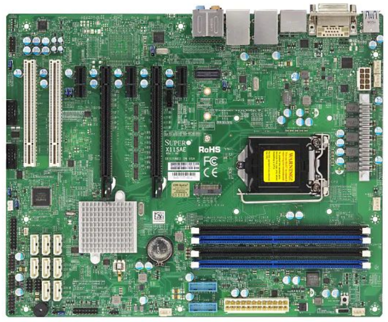


```
int sum = 0;
int num_busters = 2;
int num_perrys = 2;
sum = num_busters + num_perrys
```

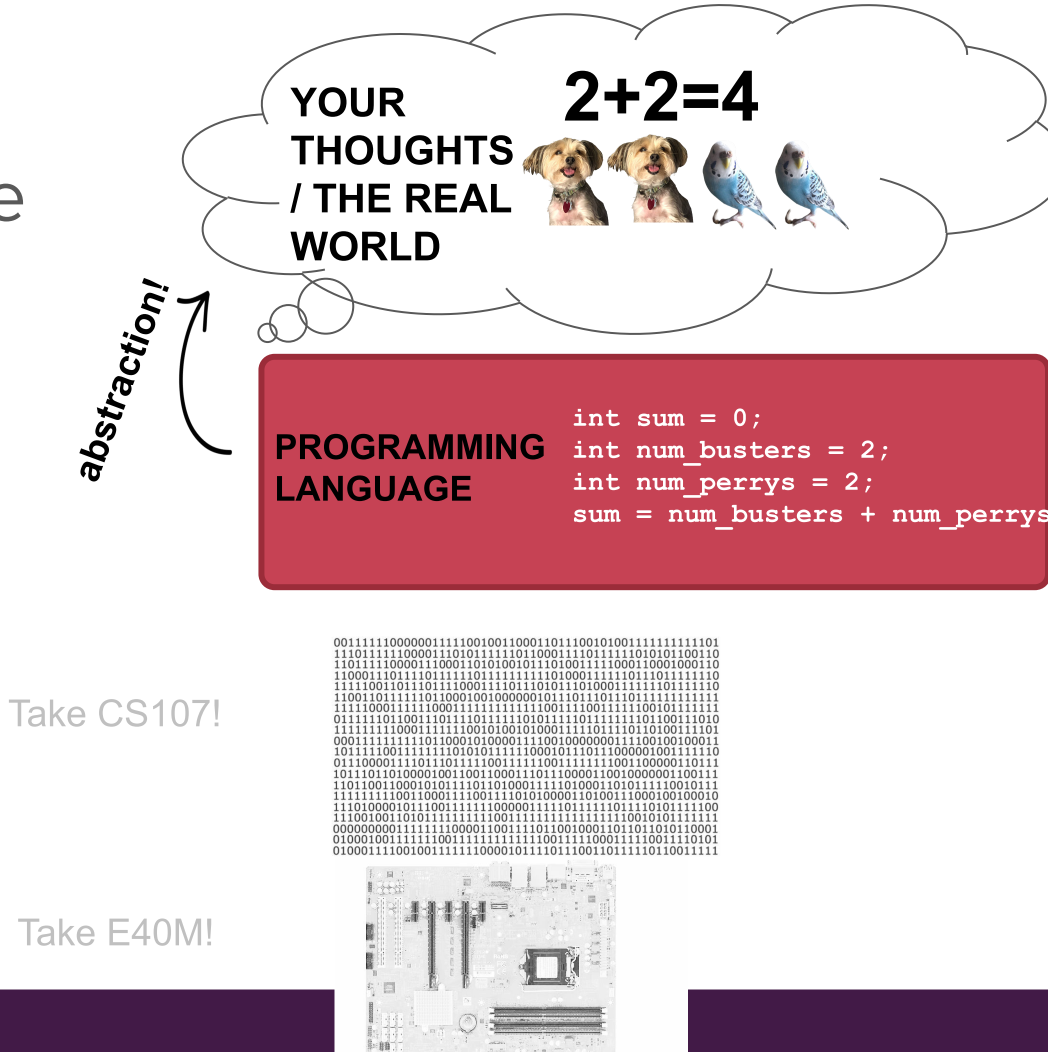
If abstractions didn't exist…

We'd have to physically reprogram our hardware every time we wanted to solve a problem like 2 + 2

"low-level"



Luckily, in CS106B, we’re only focused on the highest level of abstraction ➜



```
int sum = 0;
int num_busters = 2;
int num_perrys = 2;
sum = num_busters + num_perrys
```

```
00111111000000111110010011000110111001010011111111111101
11101111110000111010111111011000111101111110101011001 10
11011111000011100011010100101110100111110001100010001 10
```

Just to recap

- Programming languages are abstractions through which we communicate with computers.

- **Key idea**: Abstractions are simple tools that let users to control a complex system without needing to know the low-level details (how it works or how it was made).

- People are important part of designing abstractions (i.e. What should that simpler interface look like?)

- CS106B focuses on the design and/or use of abstractions in computer science.

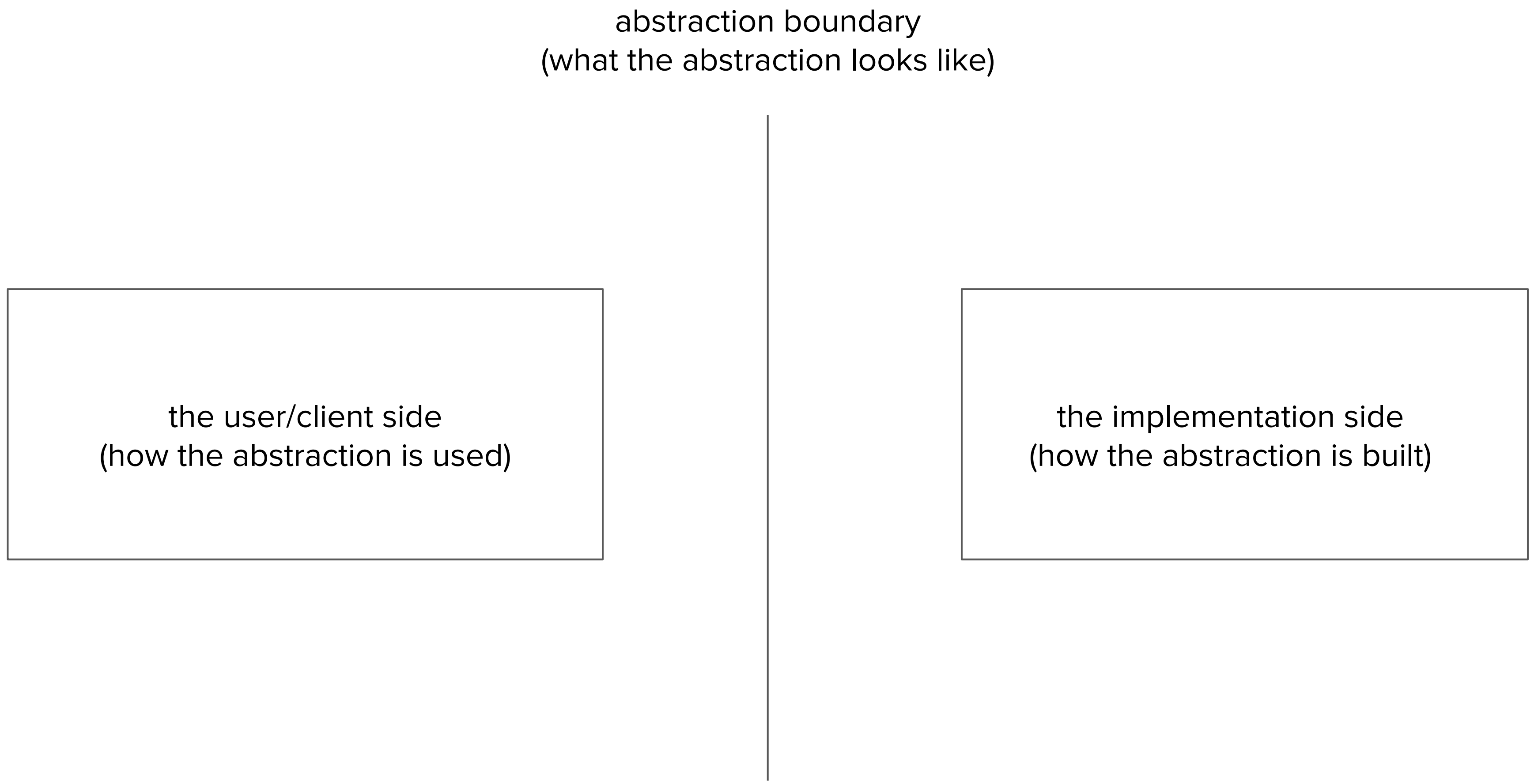
Attendance ticket:

https://tinyurl.com/june20cs106b

Please don’t send this link to students who are not here. It’s on your honor!

What is CS106B?

(the nuts and bolts)


abstraction boundary
(what the abstraction looks like)
the user/client side
(how the abstraction is used)
the implementation side
(how the abstraction is built)

classes

object-oriented programming

abstract data structures
(vectors, maps, etc.)

How to use abstractions created by others (Stanford C++ libraries)

arrays

dynamic memory
management

linked data structures

testing

algorithmic analysis

recursive problem-solving

classes

object-oriented programming

How to design abstractions for others to use

abstract data structures
(vectors, maps, etc.)

arrays

dynamic memory management

linked data structures

testing

algorithmic analysis

recursive problem-solving

classes

object-oriented programming

abstract data structures
(vectors, maps, etc.)

arrays

dynamic memory
management

linked data structures

How lower-level abstractions are used
to implement higher-level abstractions

testing

algorithmic analysis

recursive problem-solving

classes

object-oriented programming

abstract data structures
(vectors, maps, etc.)

arrays

dynamic memory
management

linked data structures

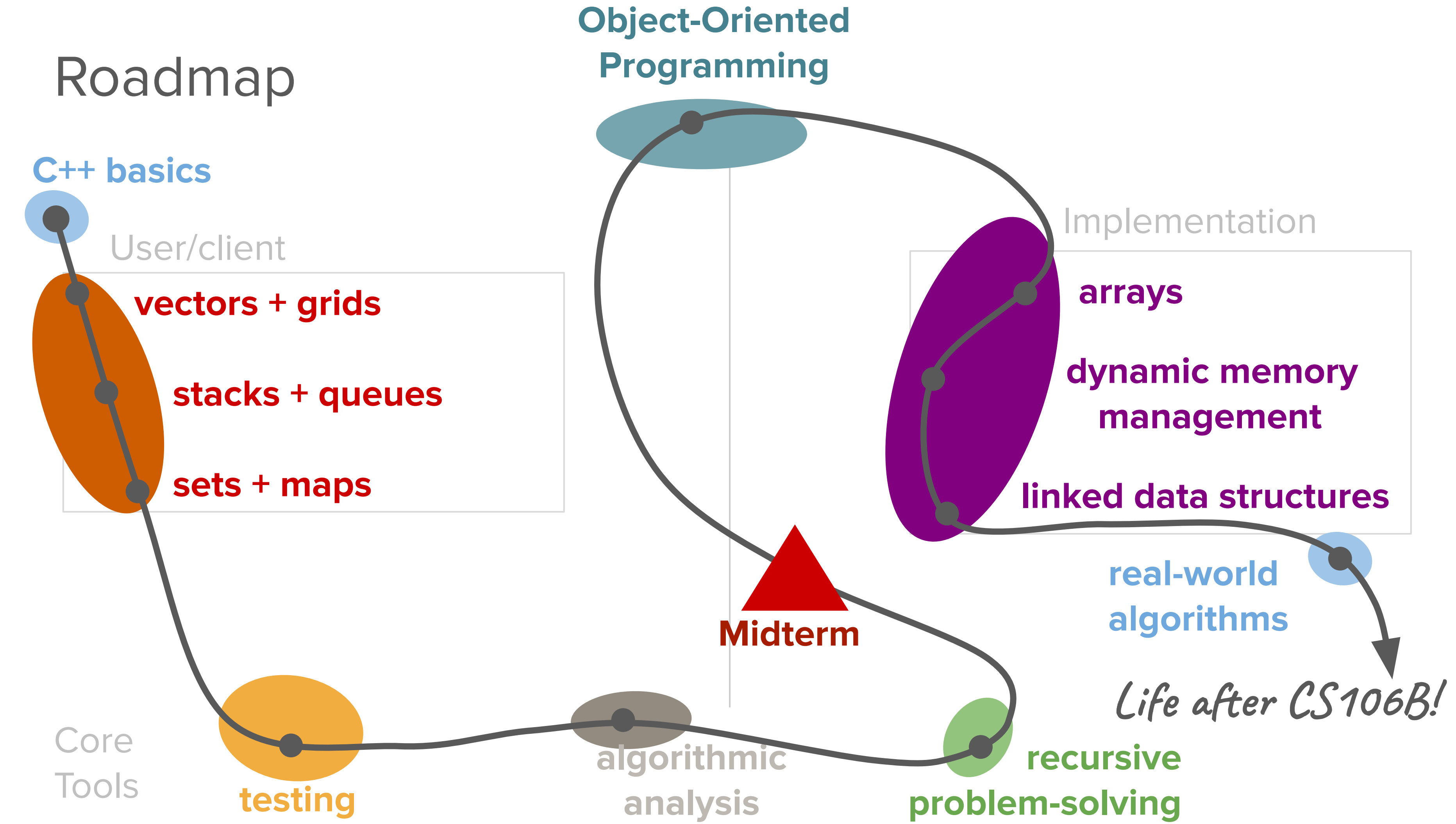
Core Tools

testing

algorithmic analysis

recursive problem-solving

Roadmap


Object-Oriented Programming
C++ basics
User/client
vectors + grids
stacks + queues
sets + maps
Implementation
arrays
dynamic memory management
linked data structures
real-world algorithms
Midterm
Life after CS106B!
Core Tools
testing
algorithmic analysis
recursive problem-solving

Learning goals

Learning goals

- I am excited to use programming to solve real-world problems I encounter outside class.

Learning goals

- I am excited to use programming to solve real-world problems I encounter outside class.
- I recognize and understand common abstractions in computer science.

Learning goals

- I am excited to use programming to solve real-world problems I encounter outside class.
- I recognize and understand common abstractions in computer science.
- I can identify programmatic concepts present in everyday technologies because I understand how computers process and organize information.

Learning goals

- I am excited to use programming to solve real-world problems I encounter outside class.
- I recognize and understand common abstractions in computer science.
- I can identify programmatic concepts present in everyday technologies because I understand how computers process and organize information.
- I can break down complex problems into smaller subproblems by applying my algorithmic reasoning and recursive problem-solving skills.

Learning goals

- I am excited to use programming to solve real-world problems I encounter outside class.
- I recognize and understand common abstractions in computer science.
- I can identify programmatic concepts present in everyday technologies because I understand how computers process and organize information.
- I can break down complex problems into smaller subproblems by applying my algorithmic reasoning and recursive problem-solving skills.
- I can evaluate design tradeoffs when creating data structures and algorithms or utilizing them to implement technological solutions.

Learning goals

- I am excited to use programming to solve real-world problems I encounter outside class.
- I recognize and understand common abstractions in computer science.
- I can identify programmatic concepts present in everyday technologies because I understand how computers process and organize information.
- I can break down complex problems into smaller subproblems by applying my algorithmic reasoning and recursive problem-solving skills.
- I can evaluate design tradeoffs when creating data structures and algorithms or utilizing them to implement technological solutions.

Overarching questions

Overarching questions

1. What is possible with technology and code? What isn’t possible?

Overarching questions

1. What is possible with technology and code? What isn’t possible?

2. How can I use programming to solve problems that I otherwise would not be able to?

Overarching questions

1. What is possible with technology and code? What isn’t possible?
2. How can I use programming to solve problems that I otherwise would not be able to?
3. What makes for a “good” algorithm or data structure? Why?

Overarching questions

1. What is possible with technology and code? What isn't possible?

2. How can I use programming to solve problems that I otherwise would not be able to?

3. What makes for a "good" algorithm or data structure? Why?

4. Which problems should I solve with algorithms and data structures? What does a responsible programmer do when using data about real people?

Overarching questions

1. What is possible with technology and code? What isn’t possible?

2. How can I use programming to solve problems that I otherwise would not be able to?

3. What makes for a “good” algorithm or data structure? Why?

4. Which problems should I solve with algorithms and data structures? What does a responsible programmer do when using data about real people?

Course norms

Course culture + norms

- Please put your mental health and wellbeing first this quarter.
- We’re here to learn - including your instructors!

Course culture + norms

- Please put your mental health and wellbeing first this quarter.
- We’re here to learn - including your instructors!

What makes for good learning?

Course culture + norms

- Please put your mental health and wellbeing first this quarter.

- We’re here to learn - including your instructors!

What makes for good learning?

1. Safe environment
   - Be kind and respectful to one another in lecture, in section, and on Ed.

Course culture + norms

- Please put your mental health and wellbeing first this quarter.

- We’re here to learn - including your instructors!

What makes for good learning?

1. Safe environment
   - Be kind and respectful to one another in lecture, in section, and on Ed.
2. Active engagement
   - Put your best foot forward in all parts of your learning process: lectures, assignments, etc.

Course culture + norms

- Please put your mental health and wellbeing first this quarter.

- We're here to learn - including your instructors!

What makes for good learning?

1. Safe environment
   - Be kind and respectful to one another in lecture, in section, and on Ed.
2. Active engagement
   - Put your best foot forward in all parts of your learning process: lectures, assignments, etc.
3. Celebration of struggle

We can center questions around learning.

Thinking about your own learning (metacognition) is important!

We can center questions around learning.

Thinking about your own learning (metacognition) is important!

Sometimes asking a question immediately and waiting for an answer can distract from the learning experience (and the question will often get answered in a slide or two).

We can center questions around learning.

Thinking about your own learning (metacognition) is important!

Sometimes asking a question immediately and waiting for an answer can distract from the learning experience (and the question will often get answered in a slide or two).

There are two (vastly oversimplified) types of questions:

1. Questions that will enable you to understand the rest of the topic/lecture.
2. Questions will expand your depth of knowledge but that your immediate understanding does not depend upon.

We can center questions around learning.

Thinking about your own learning (metacognition) is important!

Sometimes asking a question immediately and waiting for an answer can distract from the learning experience (and the question will often get answered in a slide or two).

There are two (vastly oversimplified) types of questions:

1. Questions that will enable you to understand the rest of the topic/lecture.

Strategy: Ask immediately by raising your hand. If you found something confusing, someone else probably did, too. And remember, celebrate struggle!

We can center questions around learning.

Thinking about your own learning (metacognition) is important!

Sometimes asking a question immediately and waiting for an answer can distract from the learning experience (and the question will often get answered in a slide or two).

There are two (vastly oversimplified) types of questions:

2. Questions will expand your depth of knowledge but that your immediate understanding does not depend upon.

Strategy: Write down your question and ask when we transition to a new topic. We'll also often stop for questions then. Or write code to test your question!

We can center questions around inclusivity.

There is also a third type of question:

Some students ask questions that are not really questions so much as opportunities to demonstrate knowledge of jargon or facts that are beyond the scope of the topic at hand. This can have a discouraging effect on other students. If you find yourself wanting to make such a question or comment in lecture, I encourage you to consider office hours as a better venue for exploring that topic with me.

- Cynthia Lee, Stanford Senior Lecturer in CS

We can center questions around inclusivity.

One of the most difficult things about teaching CS is catering to an audience of diverse backgrounds and prior programming experience.

We can center questions around inclusivity.

One of the most difficult things about teaching CS is catering to an audience of diverse backgrounds and prior programming experience.

Curiosity is wonderful, and we’re happy to talk about advanced CS topics with you during office hours.

We can center questions around inclusivity.

One of the most difficult things about teaching CS is catering to an audience of diverse backgrounds and prior programming experience.

Curiosity is wonderful, and we're happy to talk about advanced CS topics with you during office hours.

But we also don't want to send the message that you need to know about these things when entering CS106B.

- In particular, we don't expect students in this class to have prior C++ knowledge or knowledge of the topics that we explicitly introduce from scratch. So please keep this mind when you're asking questions!

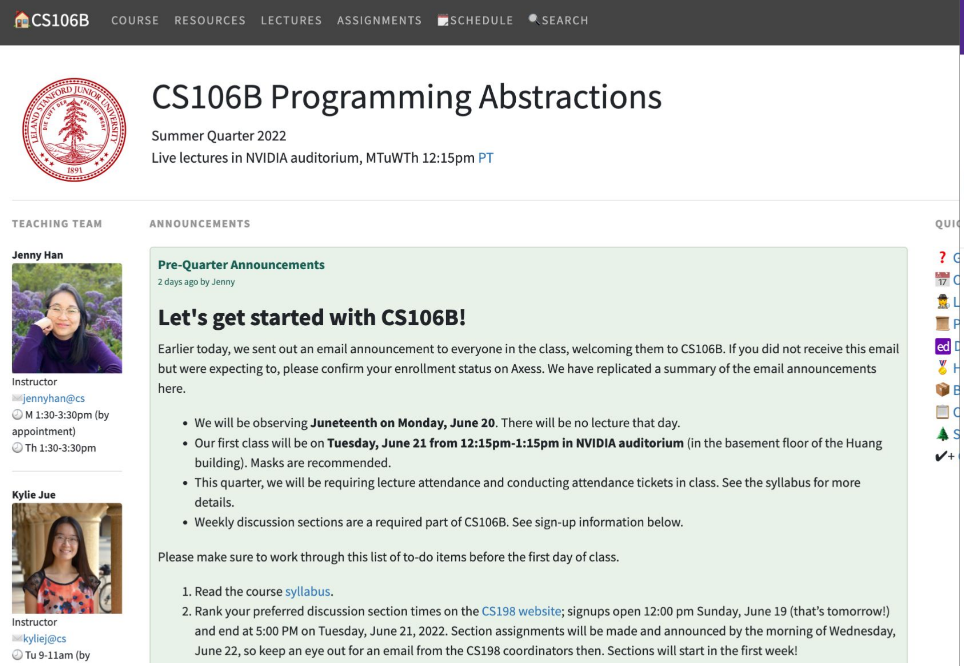
Course logistics

Is CS106B the right course for me?

- **Take the CS106B C++ survey**. This will give you a sense of the core topics we expect you to be familiar with from prior programming experience.

- Read the course placement guide on the class website.

- You cannot enroll in both CS106A and CS106B simultaneously, but you are welcome to shop both to figure out which is a better fit.



CS106B COURSE RESOURCES LECTURES ASSIGNMENTS SCHEDULE SEARCH

CS106B Programming Abstractions

Summer Quarter 2022

Live lectures in NVIDIA auditorium, MTuWTh 12:15pm PT

TEACHING TEAM

Jenny Han

Instructor

jennyhan@cs

M 1:30-3:30pm (by appointment)

Th 1:30-3:30pm

Kylie Jue

Instructor

kyliej@cs

Tu 9-11am (by

ANNOUNCEMENTS

Pre-Quarter Announcements

2 days ago by Jenny

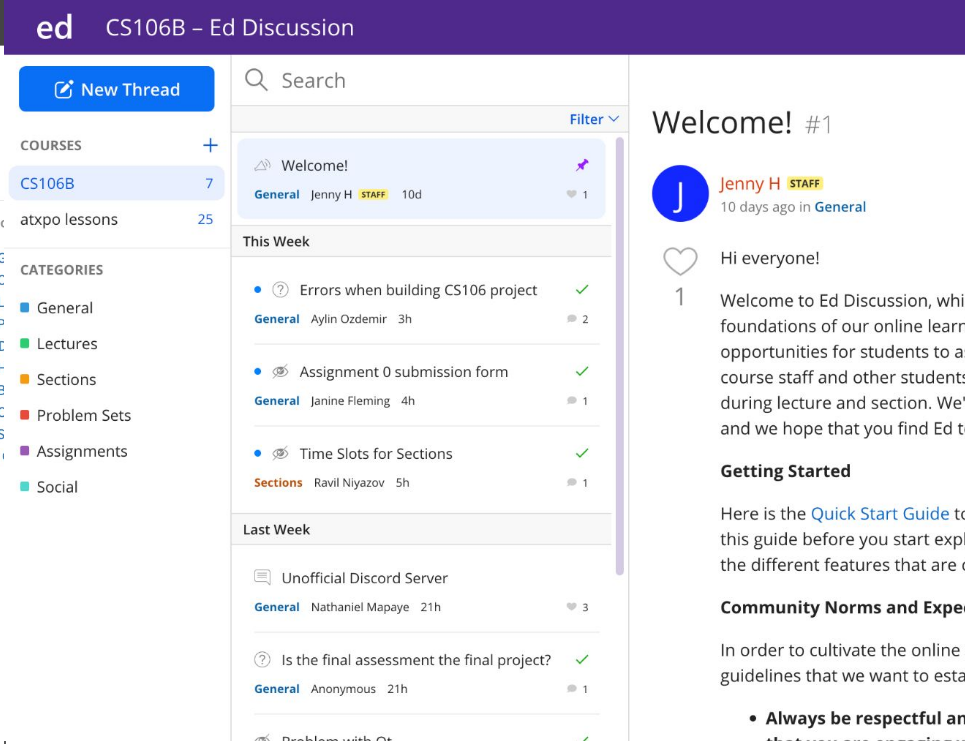
Let's get started with CS106B!

Earlier today, we sent out an email announcement to everyone in the class, welcoming them to CS106B. If you did not receive this email but were expecting to, please confirm your enrollment status on Axess. We have replicated a summary of the email announcements here.

- We will be observing **Juneteenth on Monday, June 20**. There will be no lecture that day.
- Our first class will be on **Tuesday, June 21 from 12:15pm-1:15pm in NVIDIA auditorium** (in the basement floor of the Huang building). Masks are recommended.
- This quarter, we will be requiring lecture attendance and conducting attendance tickets in class. See the syllabus for more details.
- Weekly discussion sections are a required part of CS106B. See sign-up information below.

Please make sure to work through this list of to-do items before the first day of class.

1. Read the course syllabus.
2. Rank your preferred discussion section times on the CS198 website; signups open 12:00 pm Sunday, June 19 (that's tomorrow!) and end at 5:00 PM on Tuesday, June 21, 2022. Section assignments will be made and announced by the morning of Wednesday, June 22, so keep an eye out for an email from the CS198 coordinators then. Sections will start in the first week!



ed CS106B – Ed Discussion

New Thread

COURSES

CS106B	7
atxpo lessons	25

CATEGORIES

- General
- Lectures
- Sections
- Problem Sets
- Assignments
- Social

Search

Filter

Welcome!
General Jenny H STAFF 10d 1

This Week

Errors when building CS106 project
General Aylin Ozdemir 3h 2

Assignment 0 submission form
General Janine Fleming 4h 1

Time Slots for Sections
Sections Ravil Niyazov 5h 1

Last Week

Unofficial Discord Server
General Nathaniel Mapaye 21h 3

Is the final assessment the final project?
General Anonymous 21h 1

Welcome! #1

Jenny H STAFF

10 days ago in General

1

Hi everyone!

Welcome to Ed Discussion, whi
foundations of our online learn
opportunities for students to a
course staff and other students
during lecture and section. We'
and we hope that you find Ed t

Getting Started

Here is the Quick Start Guide t
this guide before you start exp
the different features that are

Community Norms and Expe

In order to cultivate the online
guidelines that we want to esta

- **Always be respectful an**

cs106b.stanford.edu

https://us.edstem.org/

How many units?

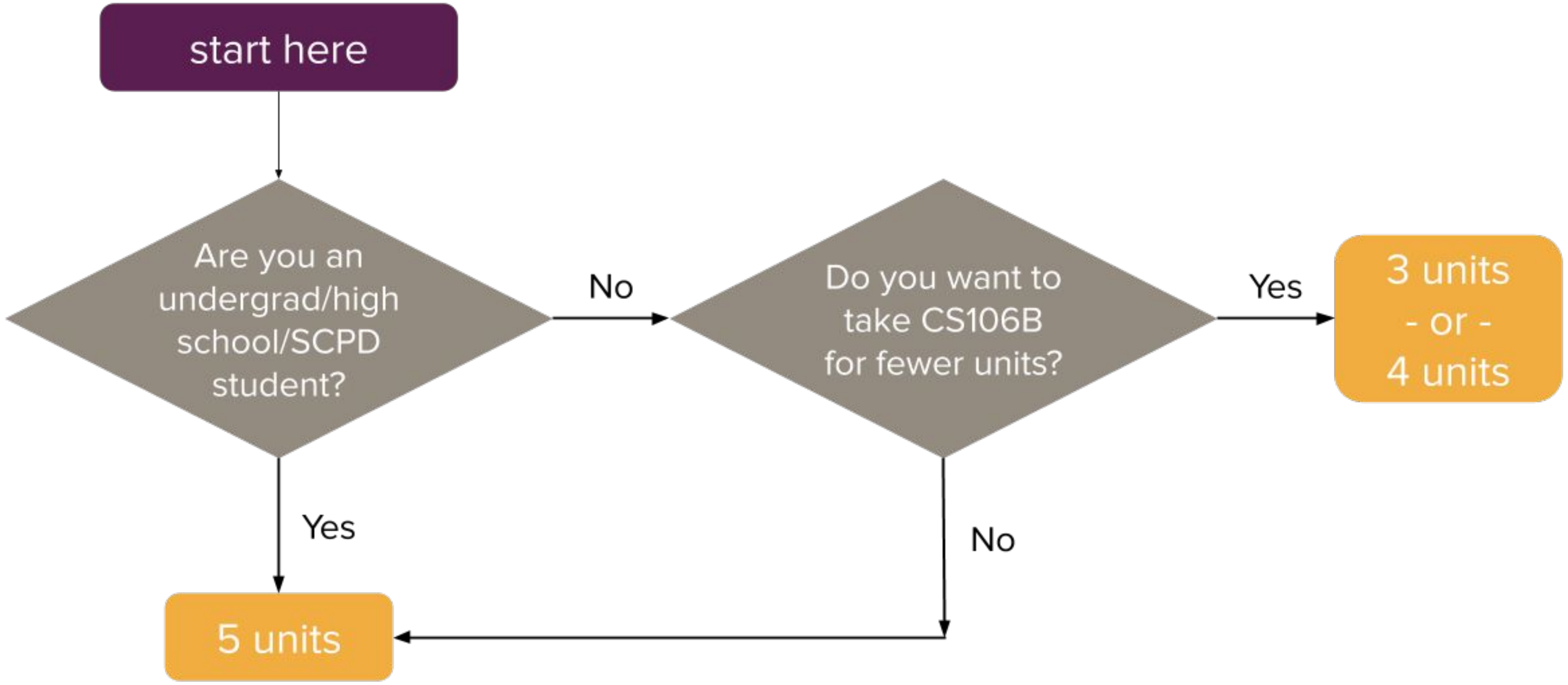
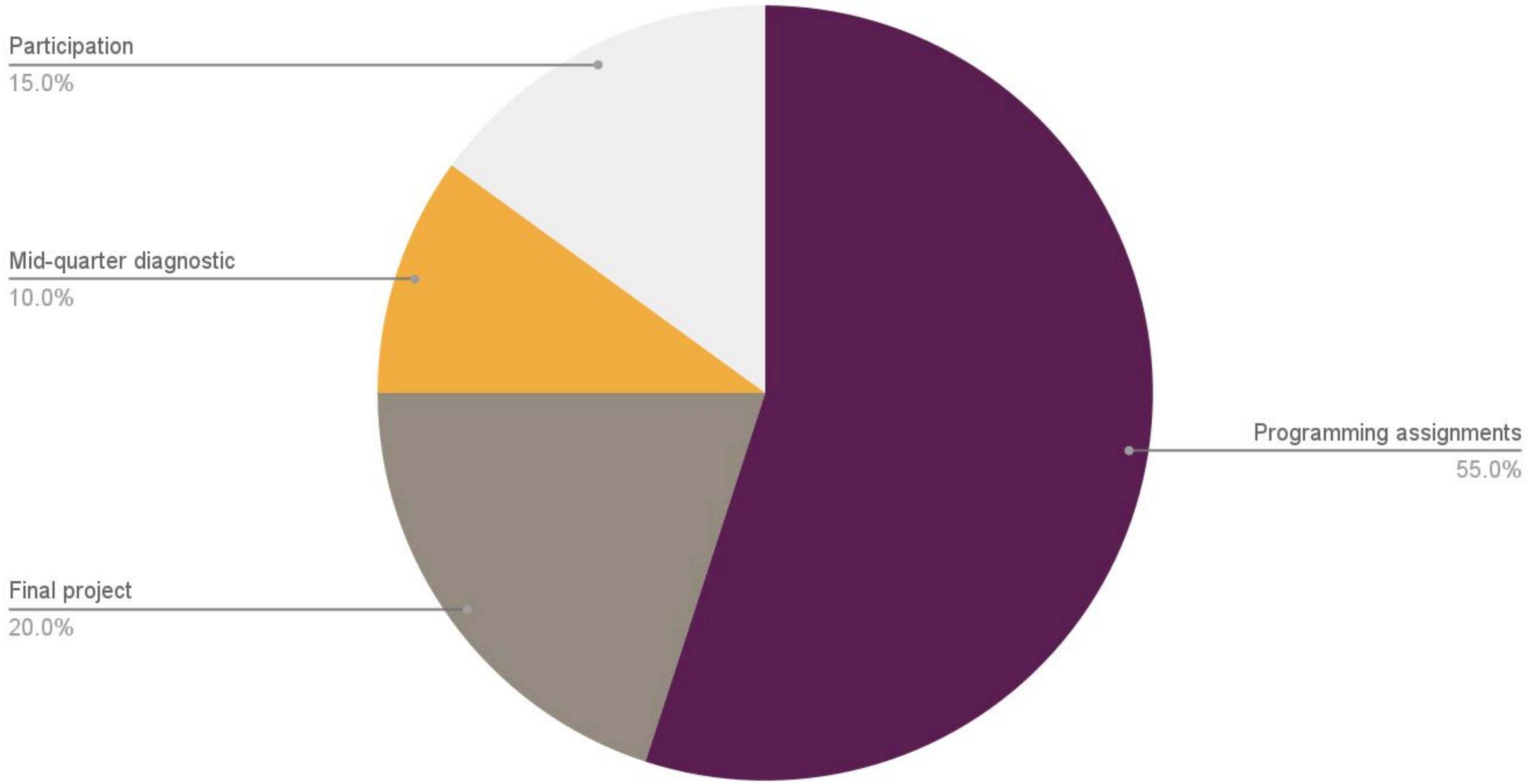

start here
Are you an undergrad/high school/SCPD student?
No
Do you want to take CS106B for fewer units?
Yes
3 units - or - 4 units
Yes
5 units
No

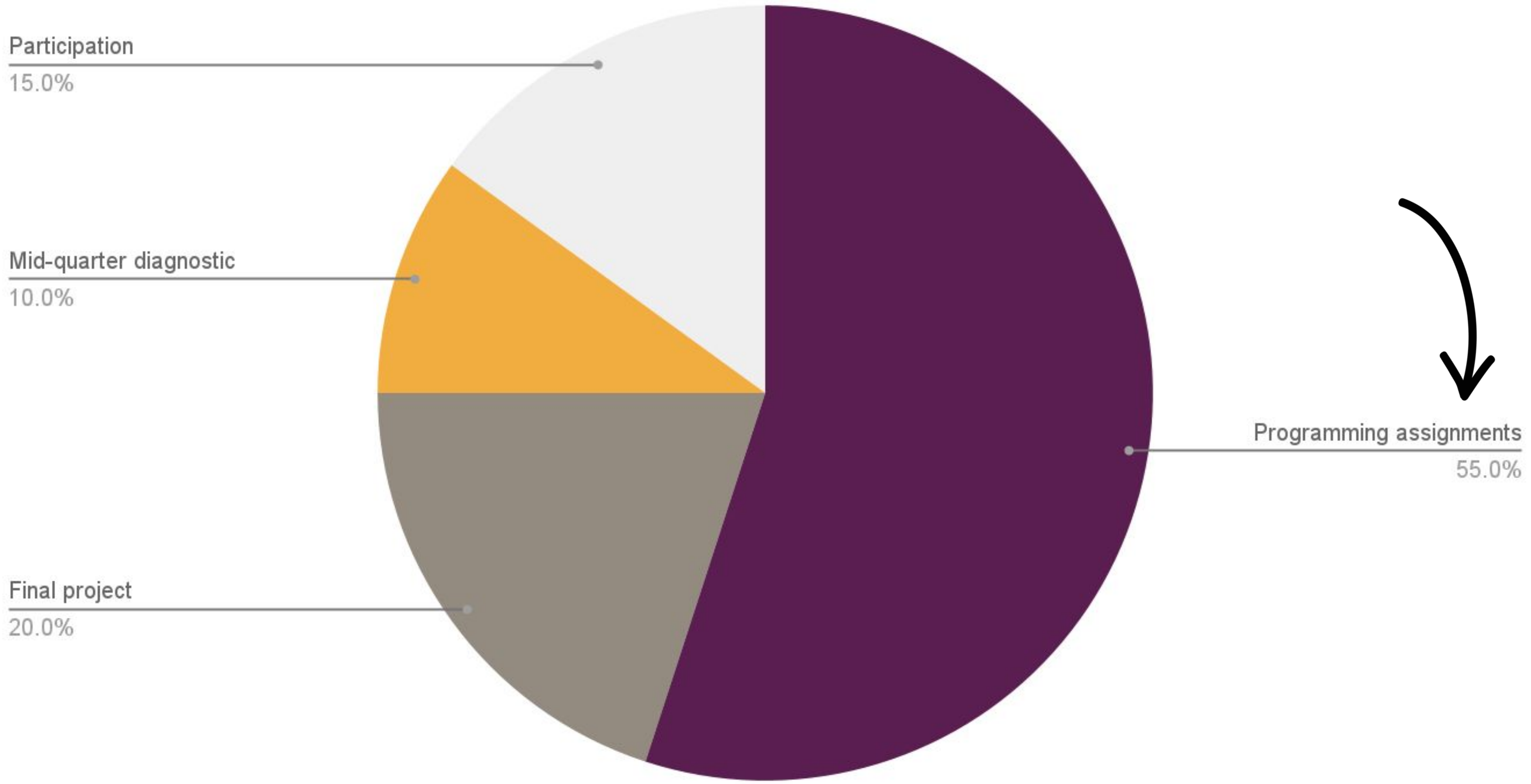
Diagram courtesy of Chris Piech

How will I be assessed?

What we will ask you to do



What we will ask you to do



Programming assignments

- There will be 6 total
  - A1: C++ Legs
  - A2: Using abstractions (abstract data structures)
  - A3: Recursion
  - A4: Defining the abstraction boundary itself
  - A5: Implementation-side of the abstraction boundary
  - A6: Real-world algorithms

Programming assignments

- There will be 6 total
- Graded on **functionality** and **style** using buckets

✓	Meets requirements, possibly with a few small problems

Programming assignments

- There will be 6 total
- Graded on **functionality** and **style** using buckets

✓+	Satisfies all requirements for the assignment
✓	Meets requirements, possibly with a few small problems
✓-	Has problems serious enough to fall short of requirements

Programming assignments

- There will be 6 total
- Graded on **functionality** and **style** using buckets

++	Absolutely fantastic submission (extremely rare)
+	"Perfect" or exceeds our standard expectations
✓+	Satisfies all requirements for the assignment
✓	Meets requirements, possibly with a few small problems
✓-	Has problems serious enough to fall short of requirements
-	Extremely serious problems, but shows some effort
--	Shows little effort and does not represent passing work

Programming assignments

- There will be 6 total
- Graded on **functionality** and **style** using **buckets** ← Why?

++	Absolutely fantastic submission (extremely rare)
+	"Perfect" or exceeds our standard expectations
✓+	Satisfies all requirements for the assignment
✓	Meets requirements, possibly with a few small problems
✓-	Has problems serious enough to fall short of requirements
-	Extremely serious problems, but shows some effort
--	Shows little effort and does not represent passing work

Programming assignments

- There will be 6 total
- Graded on functionality and style using buckets
- You can submit revisions if you receive below a check in functionality
  - Must be turned in up to three days after the next assignment is due.
  - We want to give you opportunities to demonstrate learning!
  - The revisions must include the updated code, tests to catch previous errors, and must not introduce new errors.
  - Functionality grade capped at a check.

Programming assignments

- There will be 6 total
- Graded on functionality and style using buckets
- You can submit revisions if you receive below a check in functionality
- 24-hour grace period for each assignment (specified per-assignment)
  - Most people will submit by the deadline. (“on-time” bonus)
  - The grace period is a free 24-hour extension that you can use if you have a particularly difficult week.

Programming assignments

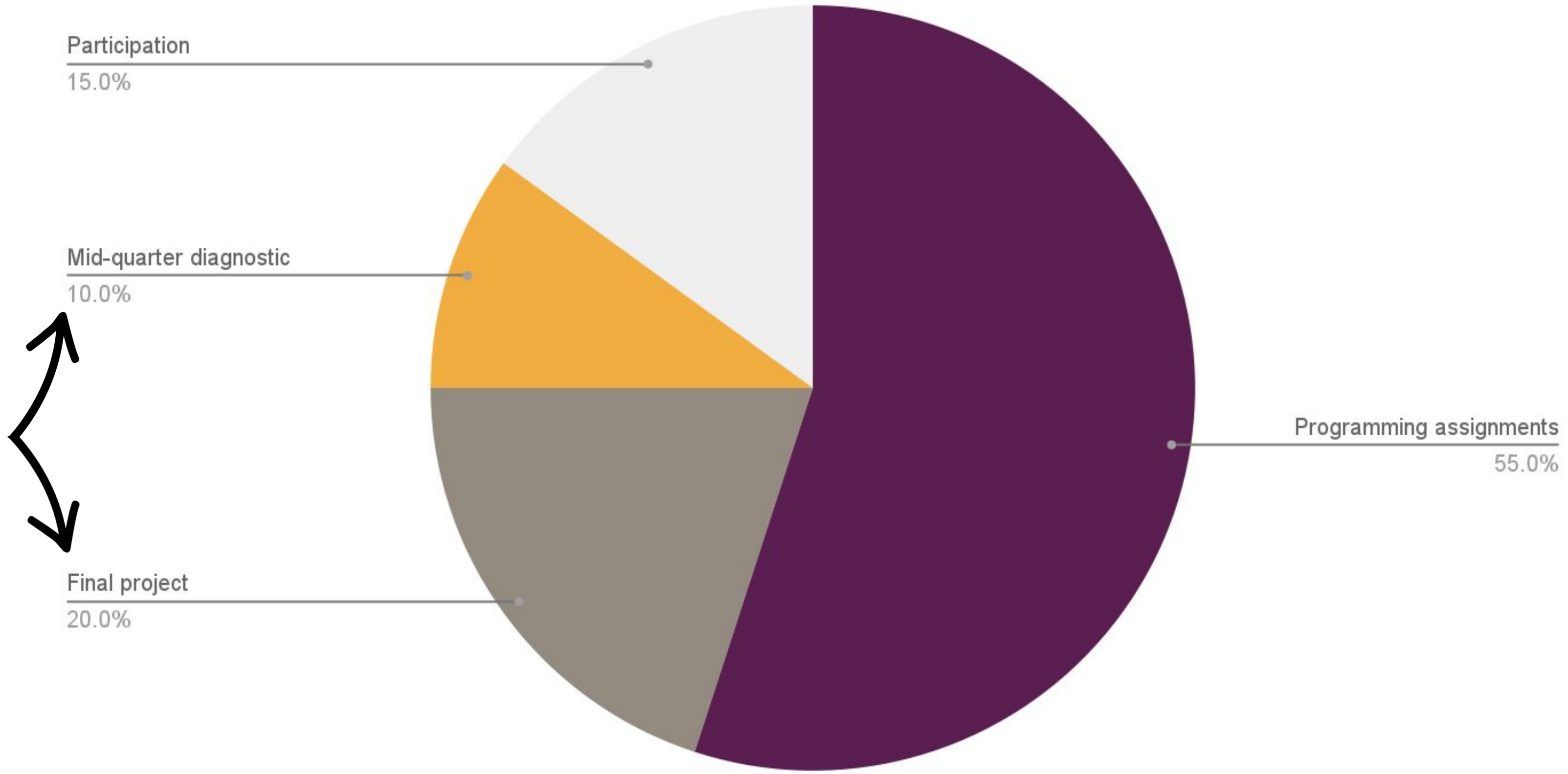
- There will be 6 total
- Graded on functionality and style using buckets
- You can submit revisions if you receive below a check in functionality
- 24-hour grace period for each assignment

Programming assignments

- There will be 6 total
- Graded on functionality and style using buckets
- You can submit revisions if you receive below a check in functionality
- 24-hour grace period for each assignment

All deadlines are at **11:59pm PDT**
(including for revisions).

What we will ask you to do



Assessments

- Mid-quarter exam
- Final project

Assessments

- Mid-quarter exam
  - Opportunity to **evaluate your understanding of the core, fundamental topics** from the first 4 weeks of the course
  - Will be in lecture on Monday, July 11 in person (SCPD students will get more logistical information later)
  - We'll provide software for you to take the diagnostic on your computer.

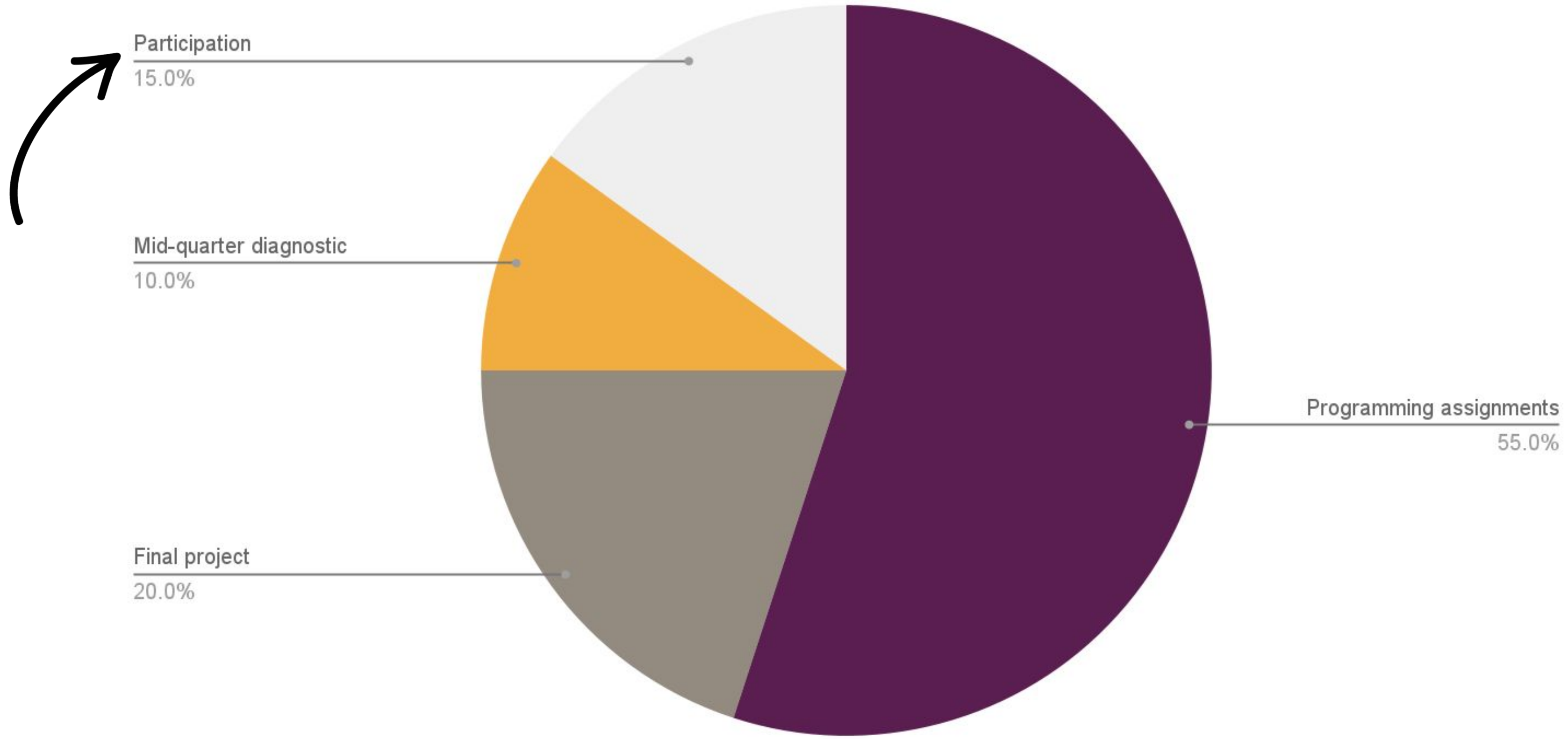
- Final project

Assessments

- Mid-quarter exam

- Final project
  - Choose a topic area that you're interested in and that you would like to improve in
  - **Write your own section/midterm problem + solution**
  - Present the problem to your section leader at the end of the quarter
  - More guidelines will be released after the midterm is over

What we will ask you to do



Why is lecture required, and how will that work?

- Not just us talking at you: active learning exercises
- Ask questions during class; we’ll also stick around to answer questions afterward!
- Quick lecture-to-usage turnaround for concepts covered in class
- At a random time during lecture, we’ll have an attendance ticket. You must turn in the attendance ticket to get credit for attending lecture.

Section attendance

- Sign up for section by **Tuesday (today) at 5pm** at cs198.stanford.edu
  - Sign-ups are already open and close tonight at 5pm PDT!
  - Sections with remaining spots will open for signups shortly after assignments have been made.

- Sections start Wednesday (tomorrow!)

How do I get help?

Section Leaders

(and some not pictured!)

What the course staff do

- Clarify conceptual material
- Help you develop good debugging practices
- Answer any administrative questions
- Chat about CS and life in general!

What the course staff do

- Clarify conceptual material
- Help you develop good debugging practices
- Answer any administrative questions
- Chat about CS and life in general!

We're always happy to help you apply CS and the concepts you've learned in class to real-world applications/areas you're interested in.

What the course staff **don’t** do

- Write your code for you

- Solve your bugs on assignments

What the course staff **don’t** do

- Write your code for you
- Solve your bugs on assignments

This is how you learn as a student!

Resources for getting help

- LaIR (general office hours)
- Your section leader
- Kylie/Jenny/Trip office hours
- Ed

Resources for getting help

- LaIR (general office hours)
    - Open Sunday through Thursday in Durand 353 (remote access is available for SCPD students)
        - Sunday/Wednesday/Thursday: 7pm-11pm
        - Monday/Tuesday: 5pm-9pm
    - Starts Wednesday, June 22
- Your section leader
- Kylie’s + Jenny’s + Trip’s office hours
- Ed

Resources for getting help

- LaIR (general office hours)
- Your section leader
- Kylie’s + Jenny’s + Trip’s office hours
  - Group office hours
  - Individual office hours - please only sign up for one 15-min slot!
- Ed

Resources for getting help

- LaIR
- Your section leader
- Kylie/Jenny/Trip office hours
- Ed

Resources for getting help

- **LaIR**
- **Your section leader**
- **Kylie/Jenny/Trip office hours**
- **Ed**

Conceptual question?

Resources for getting help

- **(C)LaIR**
- **Your section leader**
- **Kylie/Jenny/Trip office hours**
- **Ed**

Conceptual question?

Resources for getting help

- **LaIR**
- **Your section leader**
- **Kylie/Jenny/Trip office hours**
- Ed

Debugging help + code questions?

Resources for getting help

- LaIR
- Your section leader
- **Kylie/Jenny/Trip office hours**
- **Ed**

Administrative questions?

Resources for getting help

- LaIR
- **Your section leader**
- **Kylie/Jenny/Trip office hours**
- Ed

General CS + life questions?

Resources for getting help

- LaIR
- Your section leader
- Kylie/Jenny/Trip office hours
- Ed

When in doubt, check the Course Communication guidelines!

The Summer Academic Resource Center (SARC) also offers tutoring and academic support separate from our course.

Extra Practice sessions

- 1 extra hour of content review, practice problems, and homework support outside your required section.
- If you feel that more review in a small-group setting would help you succeed in CS106A/B, these sessions are for you.
- If you're looking for additional challenges or extensions to the course content, these sessions may not be for you.
- Capped at 10 people - you commit for the entire quarter.

Fill out this interest form by Thursday, June 23:
https://tinyurl.com/extrapracticecs106

Honor Code

Stanford's Honor Code

- All students in the course must abide by the **Stanford Honor Code**.
- Make sure to read over the **Honor Code handout** on the CS106B website for CS-specific expectations.
- Acknowledge any help you get outside course staff directly in your work.
- We run code similarity software on all of your programs and check final projects against online resources.
- Anyone caught violating the Honor Code will automatically fail the course.

Why C++?

How is C++ different from other languages?

- C++ is a compiled language (vs. interpreted)
  - This means that before running a C++ program, you must first compile it to machine code.

How is C++ different from other languages?

- C++ is a compiled language (vs. interpreted)

- C++ gives us access to lower-level computing resources (e.g. more direct control over computer memory)
  - This makes it a great tool for better understanding abstractions!

How is C++ different from other languages?

- C++ is a compiled language (vs. interpreted)

- C++ is gives us access to lower-level computing resources (e.g. more direct control over computer memory)

- If you’re coming from a language like Python, the syntax will take some getting used to.
    - Like learning the grammar and rules of a new language, typos are expected. But don’t let this get in the way of working toward literacy!

Demo program!

The structure of a program

```cpp
#include <iostream>
#include "console.h"
using namespace std;

// The C++ compiler will look for a function
// called “main”
int main() {
   cout << "Hello, world!" << endl;
   return 0;  // must return an int to indicate
              // successful program completion
}
```

C++

```python
import sys


# This function does not need to be called “main”
def main():
   print('Hello, world!')


if __name__ == '__main__':
    # Any function that gets placed here will get
    # called when you run the program with
    # `python3 helloworld.py`
    main()
```

Python

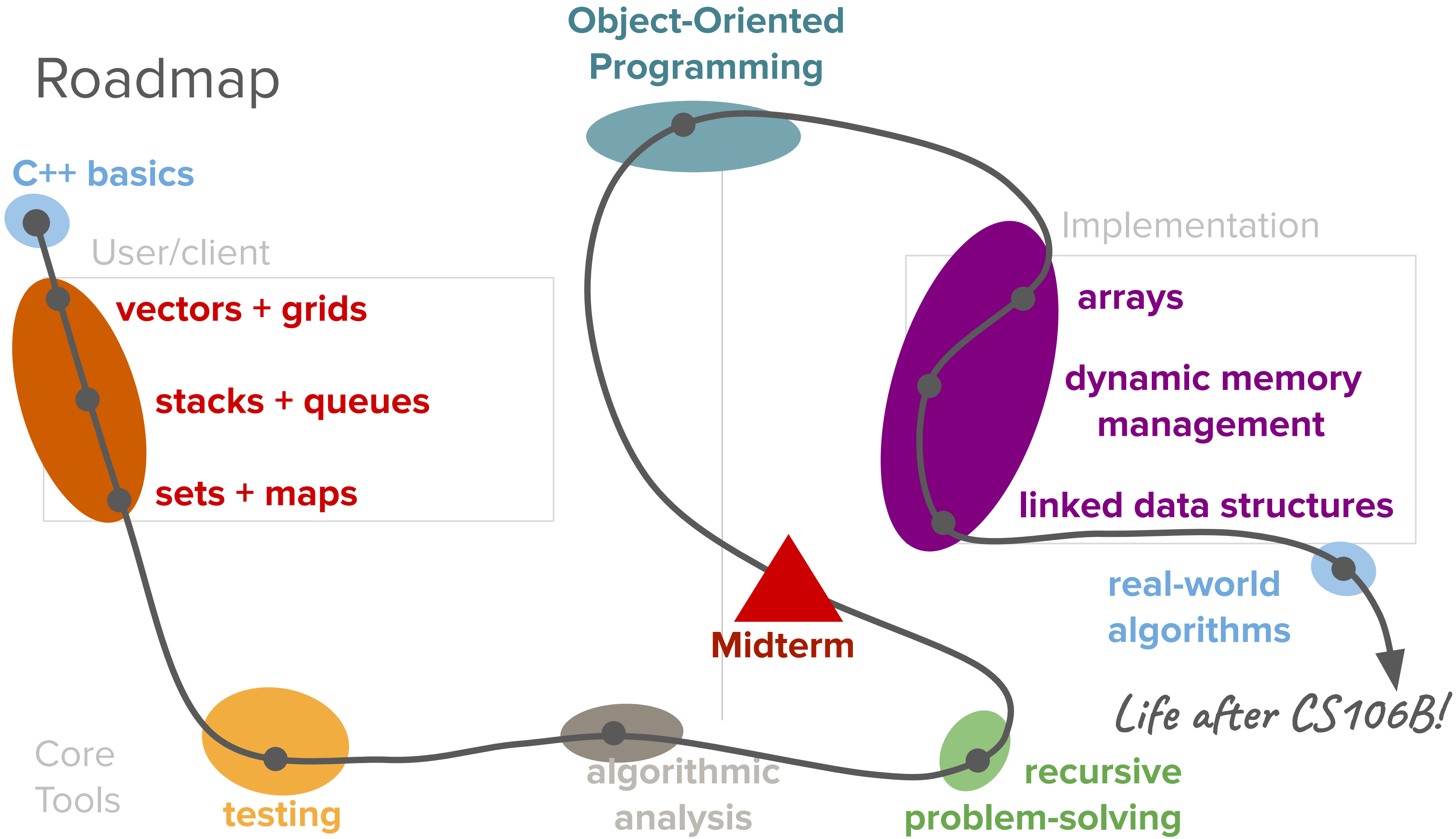
What’s next?

Applications of abstractions

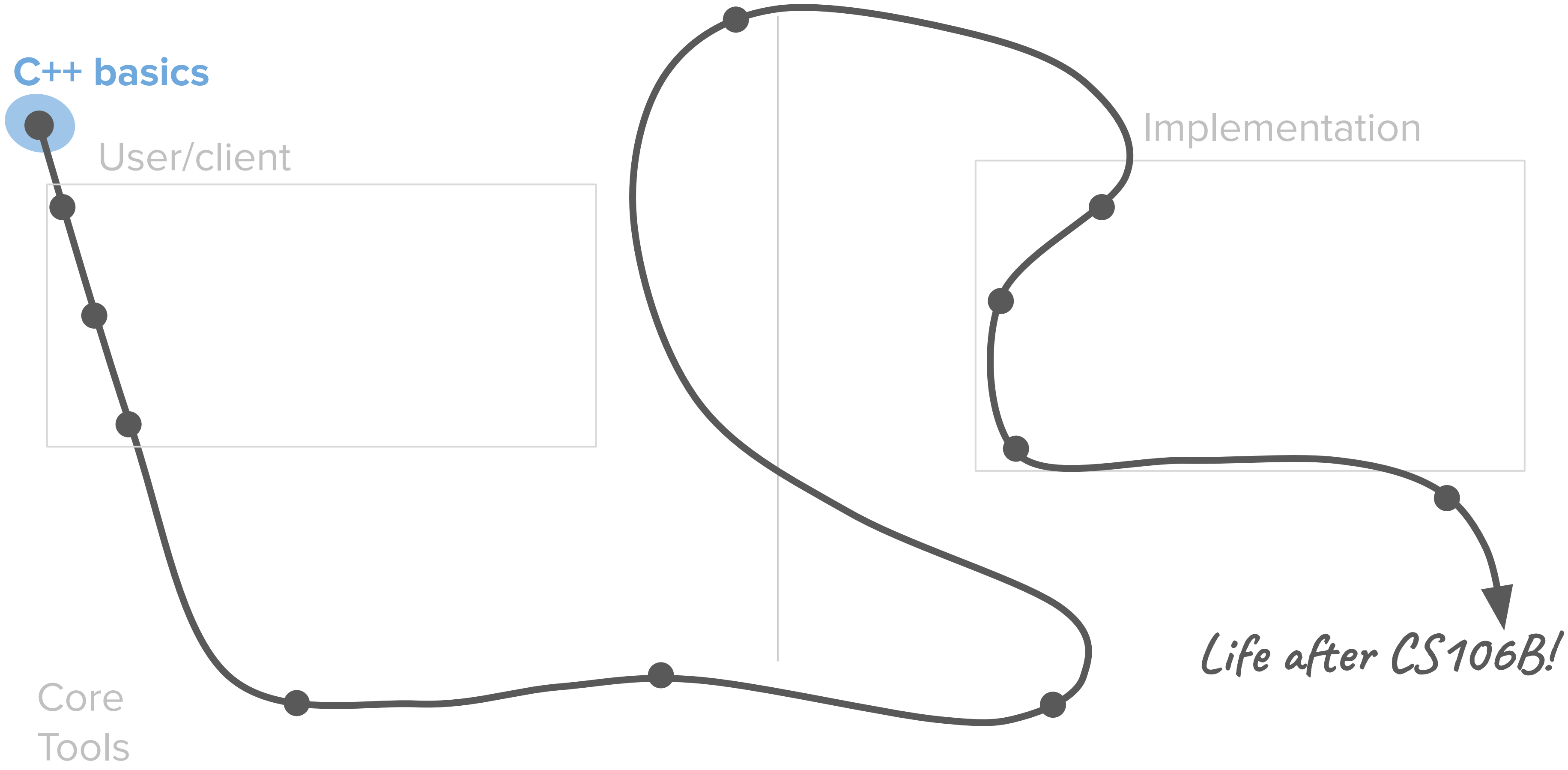
Reminders

- Complete the C++ survey ASAP.
- Fill out your section time preferences by **today at 5pm PDT**.
  - Make sure to check what time you've been assigned tomorrow morning.
- If you're interested in the extra help session, fill out this form by Thursday.
- Finish Assignment 0 by Friday.
  - If you're running into issues with Qt Creator, come to the Qt Installation Help Session on Wednesday (tomorrow) from **1:15-3:45pm PDT in Huang 019**.

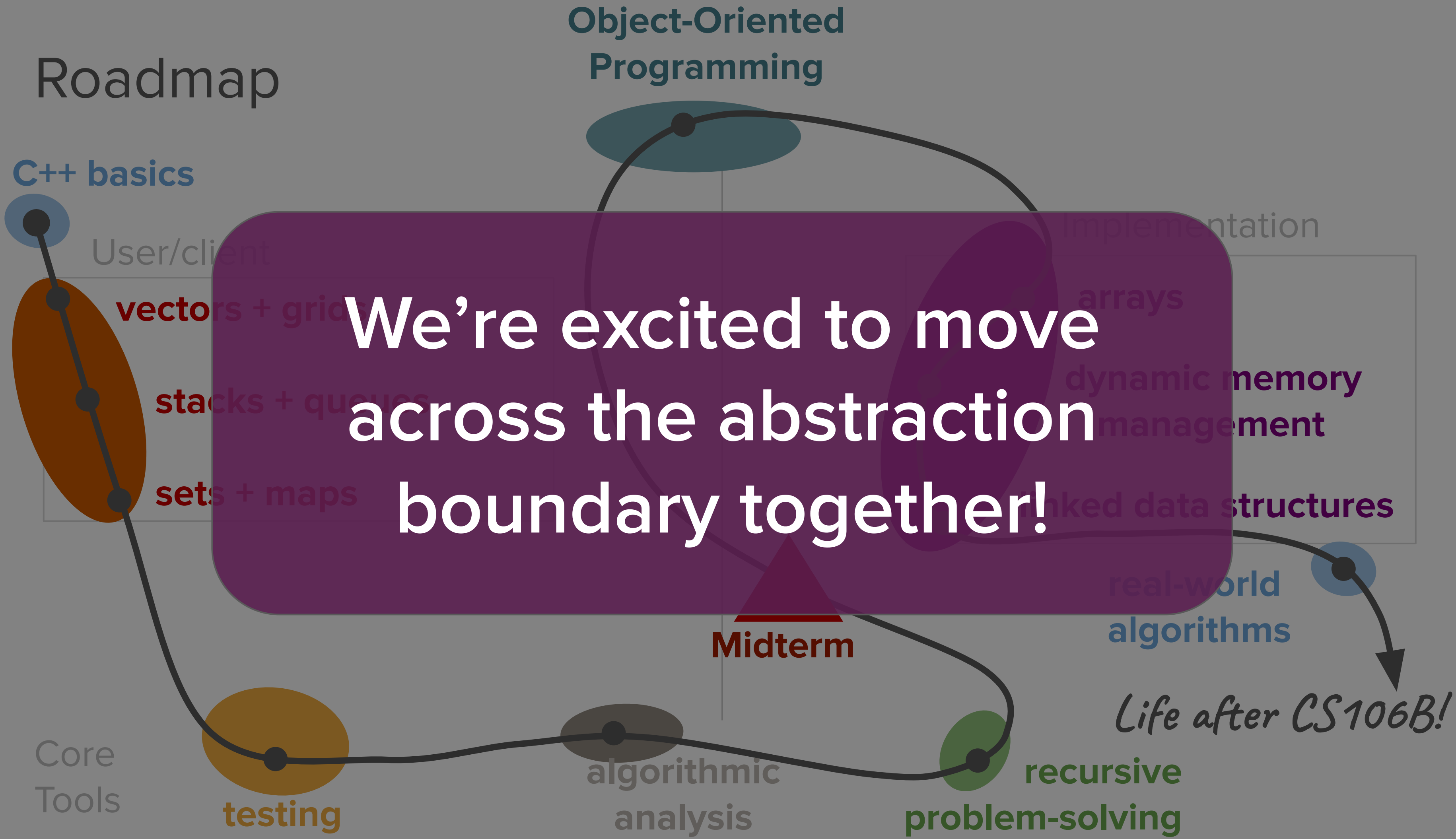
Roadmap


Object-Oriented Programming
C++ basics
User/client
vectors + grids
stacks + queues
sets + maps
Implementation
arrays
dynamic memory management
linked data structures
real-world algorithms
Midterm
Life after CS106B!
Core Tools
testing
algorithmic analysis
recursive problem-solving

Tomorrow...


C++ basics
User/client
Implementation
Core
Tools
Life after CS106B!

Roadmap



We're excited to move across the abstraction boundary together!