

Welcome to CS106B: Programming Abstractions!

Where in the world are you right now?
(put your answers the chat)

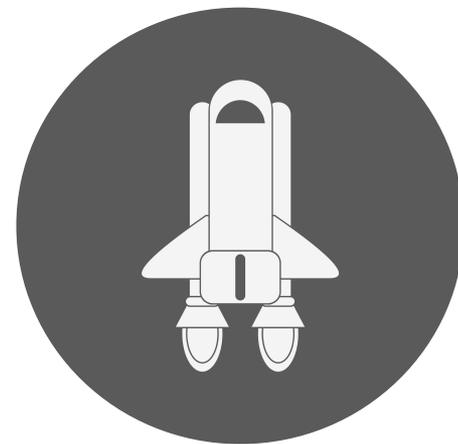
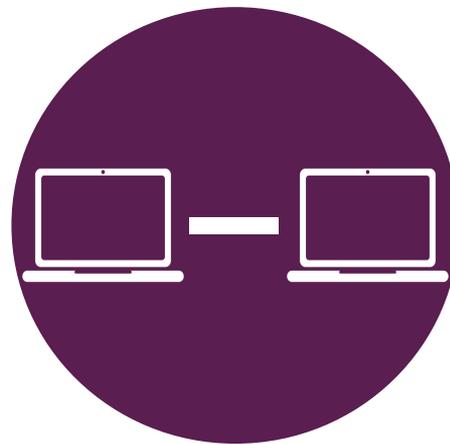


Who are we?

Kylie Jue



Nick Bowman



Today's questions

Why take CS106B?

What is an abstraction?

What is CS106B?

Why C++?

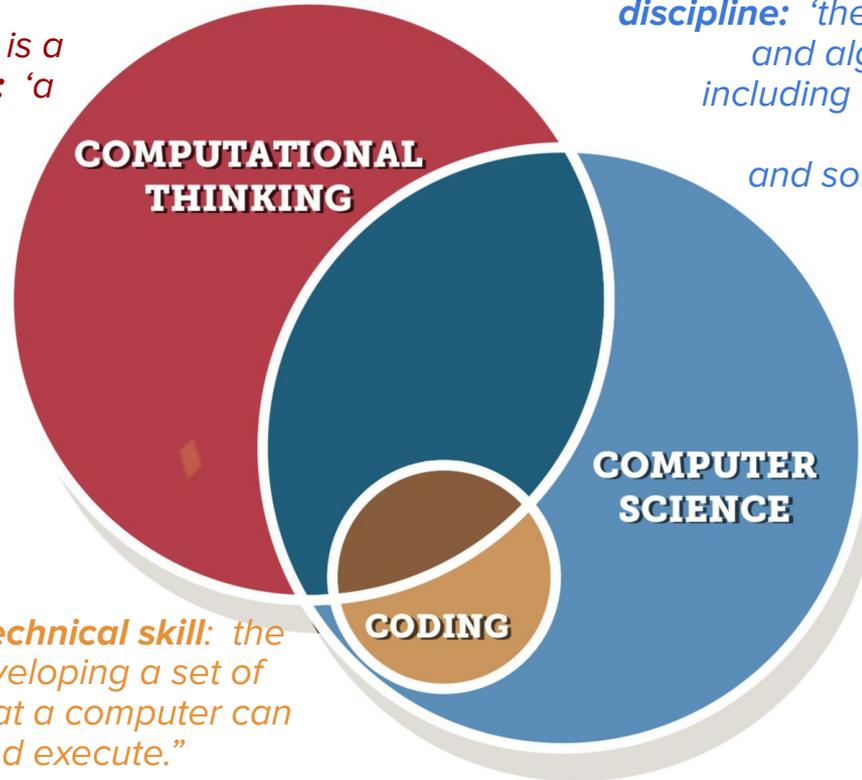
What's next?

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'"

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

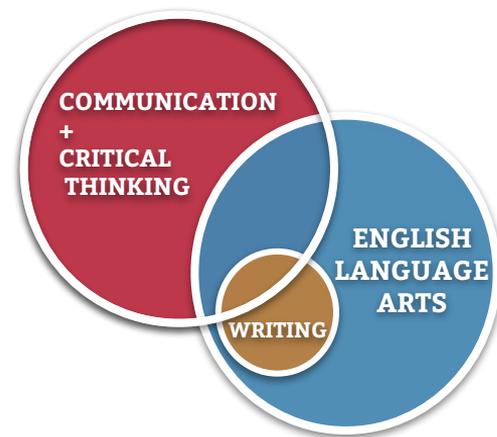
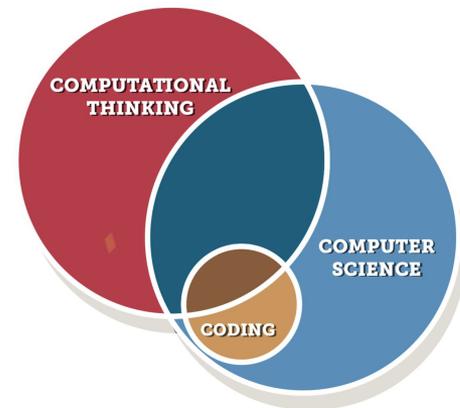


"Computer science is an academic discipline: 'the study of computers and algorithmic processes, including their principles, their hardware and software designs, their applications, and their impact on society'"

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

[\(Wells 1981\)](#)

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?

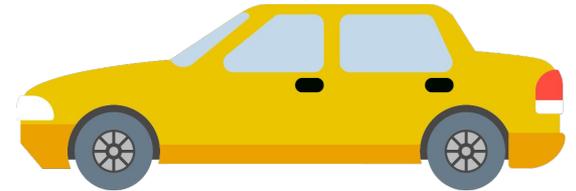
Breakout rooms!

Definition

abstraction

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

Examples of abstraction



What is an abstraction?

- Another example: Programming languages are abstractions through which we communicate with computers.
- **Key idea:** 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.
- People are important part of defining abstractions and defining the boundary between usage and implementation (i.e. What should that simpler interface look like?)
- CS106B focuses on the design and/or use of abstractions in computer science.

Moving across the “abstraction boundary”

Your journey into learning abstractions will be like learning to cook.

Moving across the “abstraction boundary”

Your journey into learning abstractions will be like learning to cook.

You start off by using other people’s recipes – tools that others have created to make it easy to prepare food and ensure you have sustenance.

Moving across the “abstraction boundary”

Your journey into learning abstractions will be like learning to cook.

You start off by using other people’s recipes – tools that others have created to make it easy to prepare food and ensure you have sustenance.

Some of these recipes (tools) are better than others, and you learn how to evaluate them and use them in ways that work best for you as you gain more practice.

Moving across the “abstraction boundary”

Your journey into learning abstractions will be like learning to cook.

You start off by using other people’s recipes – tools that others have created to make it easy to prepare food and ensure you have sustenance.

Some of these recipes (tools) are better than others, and you learn how to evaluate them and use them in ways that work best for you as you gain more practice.

The abstraction boundary is the cookbook, with its recipes and cooking techniques.

Moving across the “abstraction boundary”

Your journey into learning abstractions will be like learning to cook.

You start off by using other people’s recipes – tools that others have created to make it easy to prepare food and ensure you have sustenance.

Some of these recipes (tools) are better than others, and you learn how to evaluate them and use them in ways that work best for you as you gain more practice.

The abstraction boundary is the cookbook, with its recipes and cooking techniques.

You begin to learn more about the science of cooking – understanding how different flavors and ingredients work together, what certain cooking techniques do to various foods, and maybe even how to write some of your own recipes.

abstraction boundary
(what the abstraction looks like)

the user/client side
(how the abstraction is used)

the implementation side
(how the abstraction works)

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 works)

classes
object-oriented programming

abstract data structures
(vectors, maps, etc.)

arrays
dynamic memory
management
linked data structures

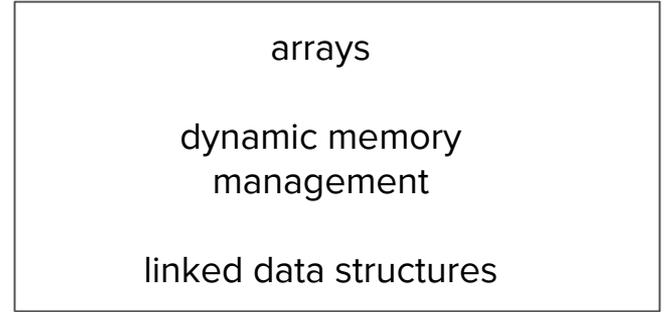
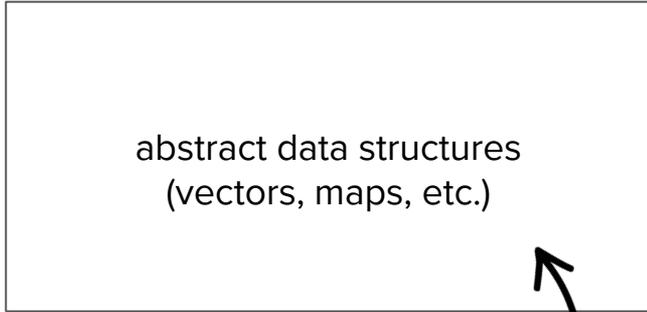
testing

algorithmic analysis

recursive problem-solving

classes

object-oriented programming



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

testing

algorithmic analysis

recursive problem-solving

classes

object-oriented programming



How to write abstractions for others to use

abstract data structures
(vectors, maps, etc.)

arrays
dynamic memory management
linked data structures

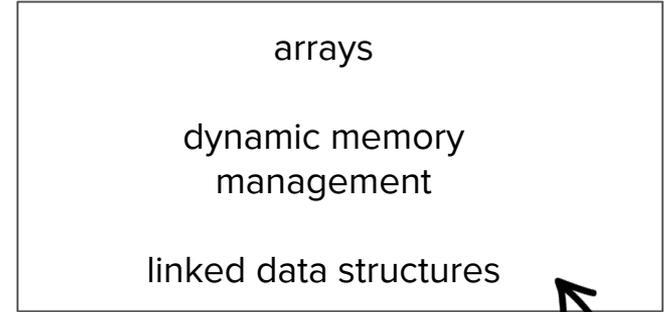
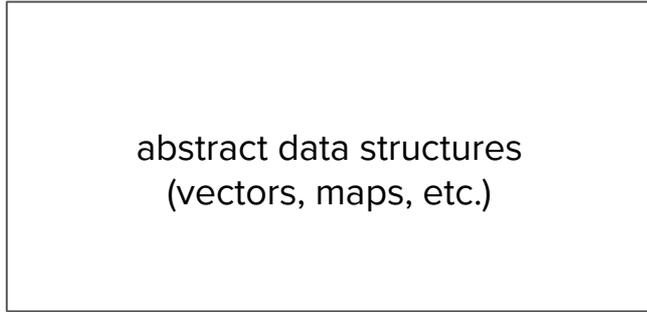
testing

algorithmic analysis

recursive problem-solving

classes

object-oriented programming



*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

C++ basics

User/client

vectors + grids

stacks + queues

sets + maps

Object-Oriented
Programming

Implementation

arrays

dynamic memory
management

linked data structures

real-world
algorithms

Life after CS106B!

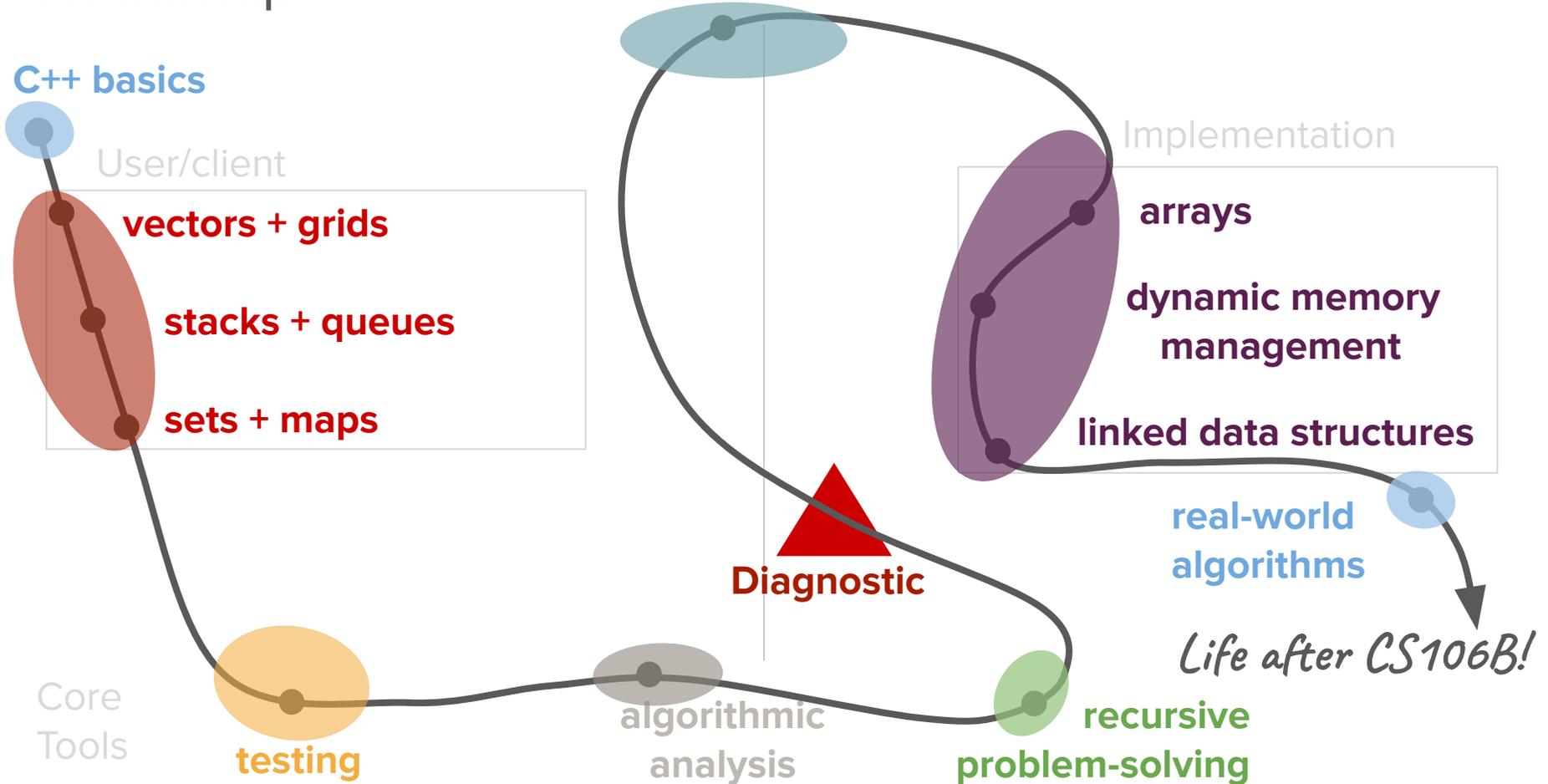
Core
Tools

testing

algorithmic
analysis

recursive
problem-solving

Diagnostic



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?

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 breakout rooms, section, and 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 breakout rooms, section, and 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 breakout rooms, section, and Ed.
2. Active engagement
 - Put your best foot forward in all parts of your learning process: lectures, assignments, etc.
3. Celebration of struggle

Zoom norms

- Avoid video fatigue – it's okay to turn off your video during lecture.
- But if you can turn on video during breakout rooms and sections, please try to do so for engagement!
- You will be muted by default. If you have questions during lecture, type them into the chat or use the “Raise hand” function if you would like to speak.
- Use the chat only for answering questions and asking questions.

(Your section leader will have separate norms for discussion sections.)

Course logistics

Is CS106B the right course for me?

- Where are you in your CS literacy journey?
- **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 Programming Abstractions

Summer 2020, Lectures: MTWTh 11:30am-12:20pm (Pacific Daylight Time, GMT-7)

TEACHING TEAM



Lecturer: Nick Bowman
@nbowman@stanford.edu



Lecturer: Kylie Jue
@kyliej@stanford.edu



Head TA: Trip Master
@tmaster@stanford.edu

ANNOUNCEMENTS

Welcome to CS106B!

3 days ago by Nick and Kylie

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 some of the action items of the email announcement here. Please make sure to work through this list of to-do items before the first day of class:

- Black Lives Matter in CS106B. The course teaching staff is committed to cultivating an equitable and socially-conscious learning environment this summer. Please take the time to read our full [statement of solidarity](#) and familiarize yourself with the variety of resources presented there.
- Read the [course syllabus](#).
- Join the [CS106B Ed Discussion Forum](#). This is where we'll post weekly announcements and important updates for the course. Additionally, you will be able to use this forum to ask questions and have discussions with course staff and your fellow students.
- Read the guide linked [in this form](#) and fill out the associated questions so we can find out what you're interested in reviewing about core programming concepts in C++.
- Get started on [Assignment 0!](#) Due on Wednesday, June 24, this assignment will help us get to know you and will introduce you to some of the tools we'll be using in the course.
- Discussion sections are an important part of CS106B, and section sign-ups will open on Sunday. You'll be able to [rank section time preferences on the CS198 website](#) from Sunday at 5pm until Tuesday at 5pm. Section assignments will be made and announced by Wednesday morning, so keep an eye out for an email from the CS198 coordinators then.
- Our first class will be on Monday, June 22 from 11:30am-12:20pm. Go to [Canvas](#) to find the Zoom info.
- Keep an eye on the course website (where you are at right now), our central hub for lectures, assignments, and other resources.

Computer science is a tool that allows us to make constructive and powerful contributions to the world, and we hope to help unlock those opportunities for you through this course. Please don't hesitate to reach out to us if

QUICK LINKS

- Statement of Solidarity with
- Course Communication
- Office Hours
- LalR Signup
- Paperless
- Zoom Details
- Ed Discussion Forum
- QT Creator
- Common Build/Run Issue
- Python-to-C++ Guide
- Blank Qt Project
- C++ Standard Library Doc
- Stanford Library Document
- CS106B Style Guide
- CS106B Testing Guide
- Submission Checklist

New Thread

Search

COURSES

CS106B	1449
CS 298	1
Code in Place	7182
Code in Place Small Group 1	8
41 more	

CATEGORIES

- General
- Lectures
- Sections
- Assignments
- Diagnostic
- Final Project
- Announcements

Pinned

Welcome to the CS106B Ed Discussion Forum!

Announcements Nick Bowman **INSTRUCTOR** 3d 2 12

Last Week

Code input for non-English speakers

General Anonymous 4h 1

Qt Setup Being Really Slow

Assignments - Assignment 0 Shirley Li 10h 6

Unable to Access Google Form

Assignments - Assignment 0 Michael Hu 16h 2

Mystery number rejected by Google form

Assignments - Assignment 0 Richee Ling 1d 2

sample-project Build Error

General Anonymous 1d 3

Debugger not working

Assignments - Assignment 0 Runze Yu 1d 7

CS106B without CS106A

General Arul Saxena 1d 2

CLion as a C++ IDE

General Dima Timofeev 1d 1

Welcome to the CS106B Ed Discussion Forum

Nick Bowman **INSTRUCTOR**
3 days ago in Announcements



12

Hi everyone! Welcome to Ed Discussion, which is the platform that we will be using for the core foundations of our online learning experience this quarter. The Ed Discussion Forum offers opportunities for students to ask questions about course content, discuss with course staff and other students, and participate in collaborative learning experiences during lecture and section. We're really excited to be able to use this platform this quarter, and we hope that you find Ed to be empowering and helpful.

Getting Started

Here is the [Quick Start Guide](#) to using Ed Discussion. We strongly recommend that you start exploring the website for yourself and getting familiar with all the different features that are offered.

Community Norms and Expectations

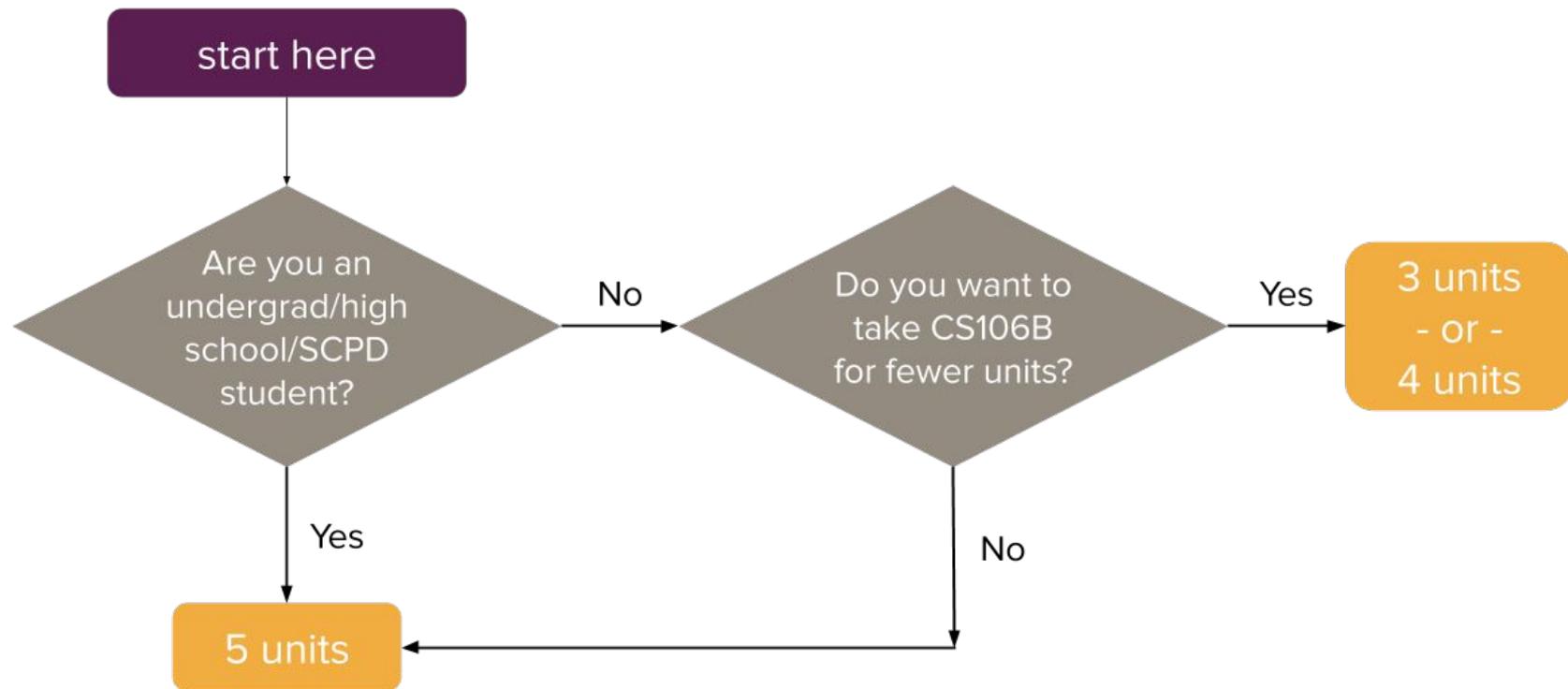
In order to cultivate the online experience for all students here, we have a set of guidelines that we want to establish as community expectations for using Ed Discussion:

1. **Always be respectful and kind to other students and members of the community.** We will not tolerate insensitive posts or comments on the platform.
2. **Stay up to date with announcements and other content posted on the platform.** We will be making all important announcements after the first day of class using this platform. We recommend checking Ed on a daily basis to look for newly posted content and announcements.
3. **Read through prior posts on Ed before asking a question.** This "Search" feature to look through previously answered questions to see if you have already been an answer to the question you have. As you start asking questions, Ed will also start suggesting other posts for you to look at depending on your question.

cs106b.stanford.edu

<https://us.edstem.org/>

How many units?



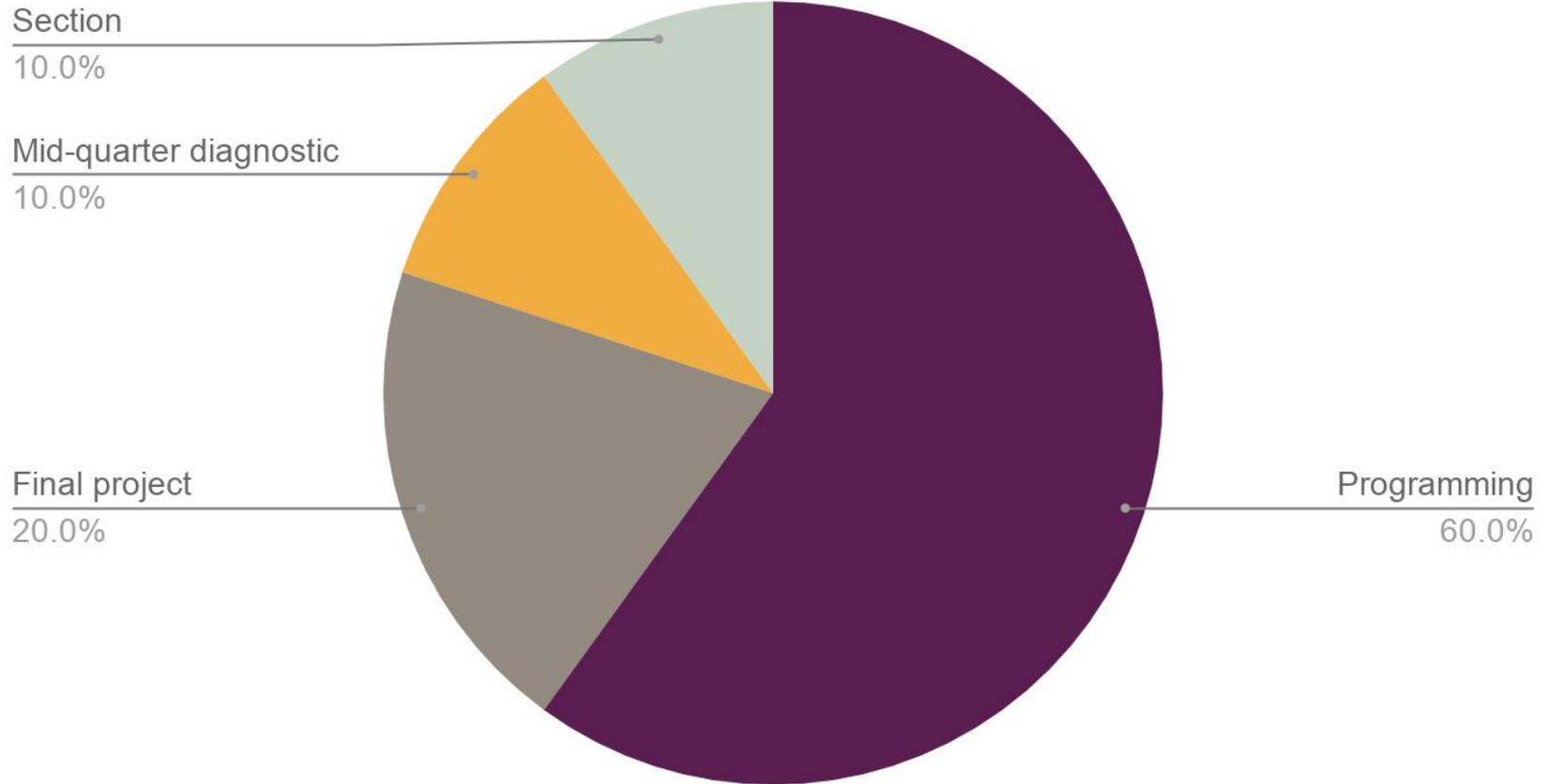
Why should I come to
lecture?

Lecture pedagogy

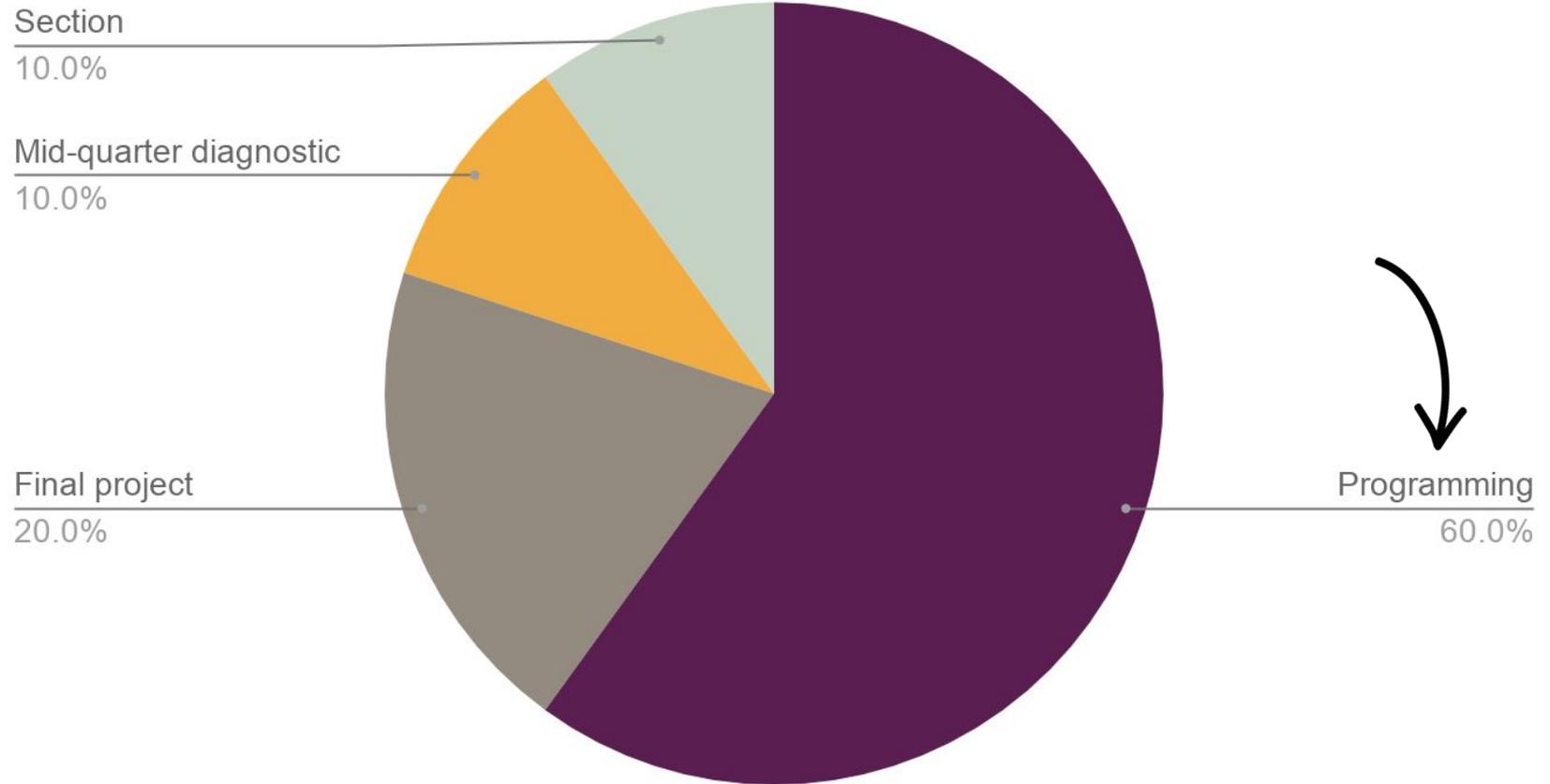
- Not just us talking at you: active learning exercises
- Quick lecture-to-usage turnaround for concepts covered in class
- We'll stick around to answer questions afterward!

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
 - 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.
 - 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
- 24-hour grace period for each 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

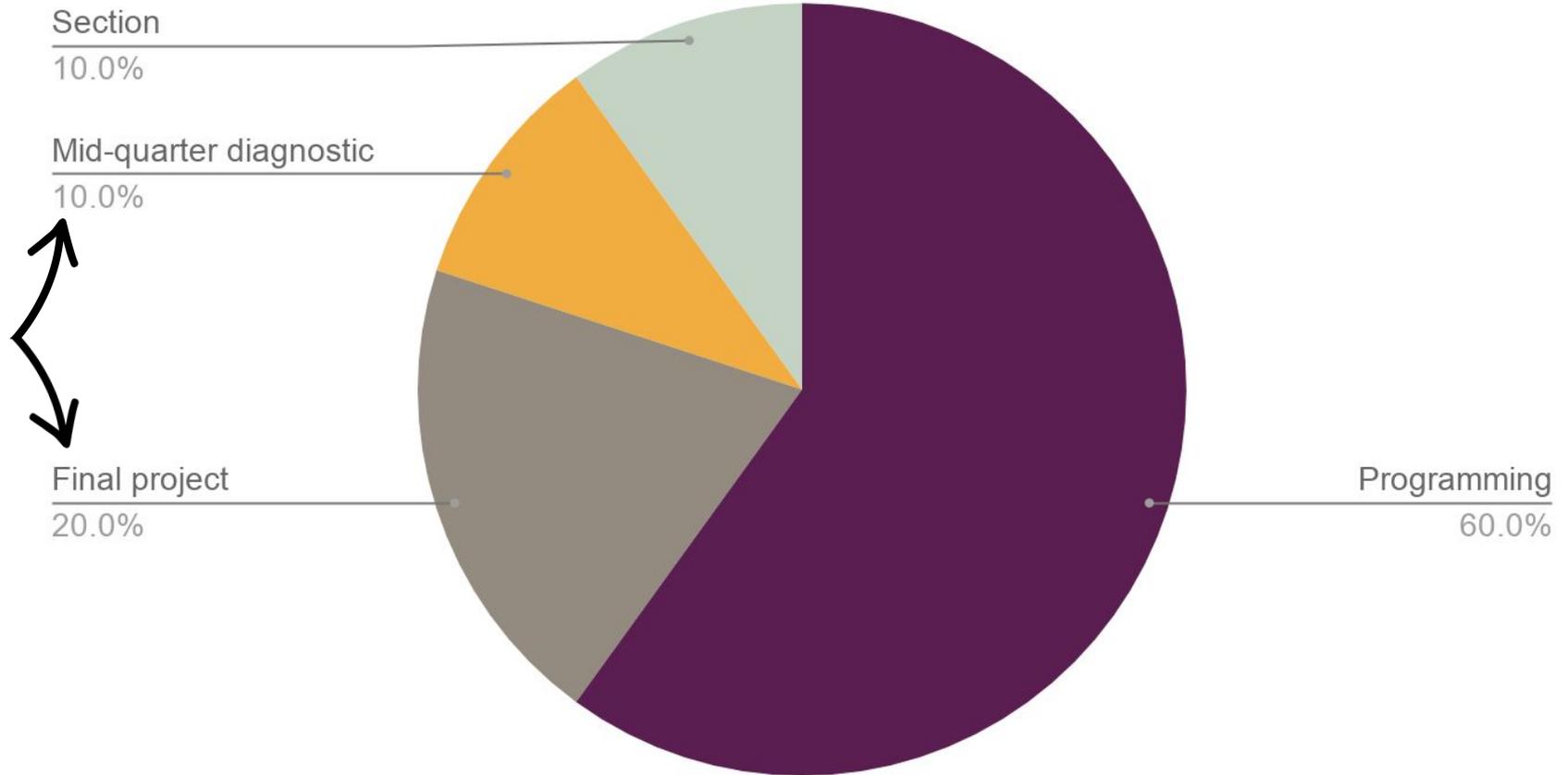
- There will be 6 total
- Graded on functionality and style using buckets
- You can submit revisions if you receive below a check
- 24-hour grace period for each assignment

Programming assignments

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

All deadlines are at **11:59pm in your local time zone**
(including for revisions).

What we will ask you to do



Assessments

- Mid-quarter diagnostic
- Final project

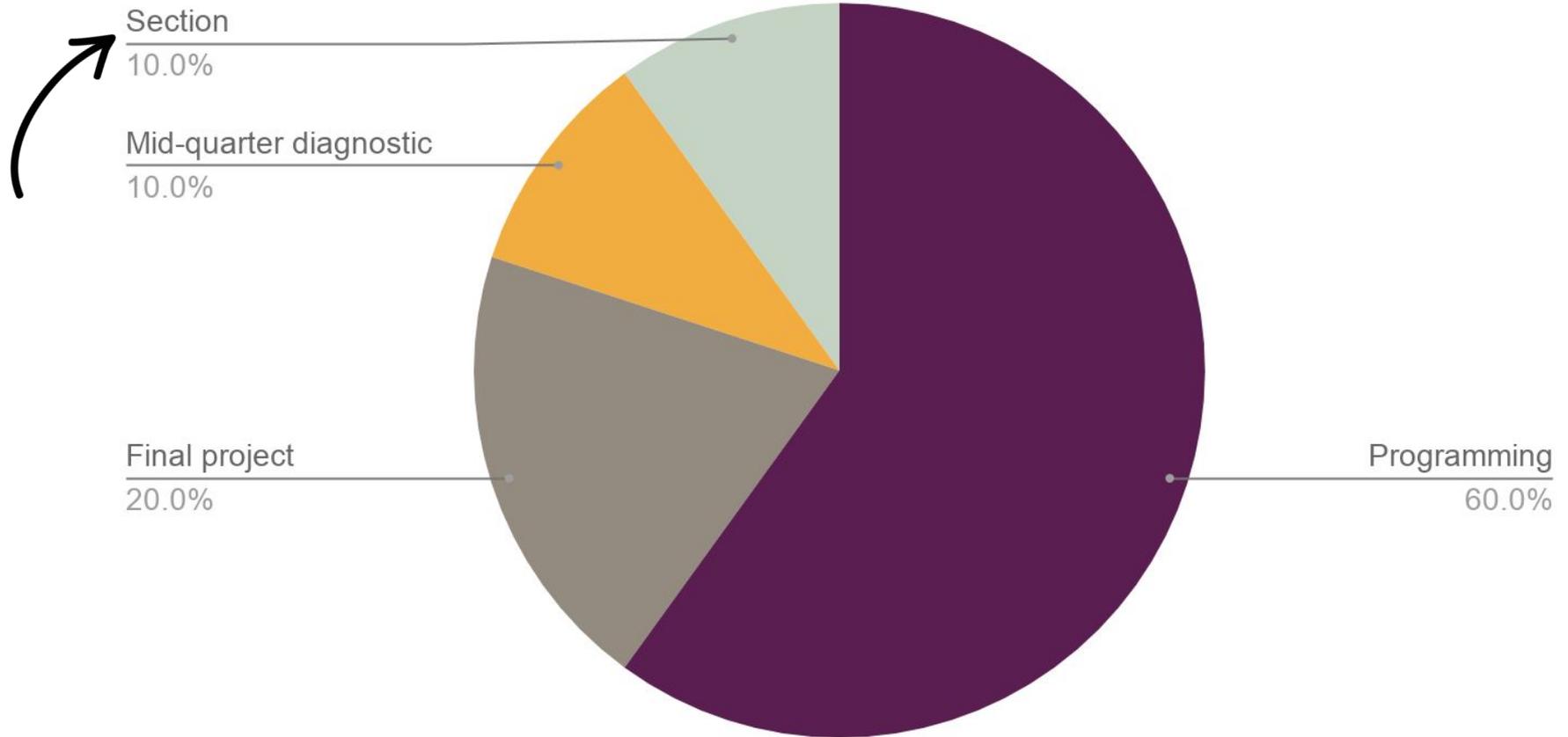
Assessments

- Mid-quarter diagnostic
 - Opportunity to **evaluate your understanding of the core, fundamental topics** from the first 3 weeks of the course
 - Designed to take 1.5 hours; completely open notes
 - Available to complete over a 72-hour time span from July 17-19
 - We'll provide software for you to take the exam on your computer
 - once you open the file, you'll have 3 hours to complete it
 - Post-exam feedback and self-reflection
- Final project

Assessments

- Mid-quarter diagnostic
- Final project
 - Choose a topic area that you're interested in and that you would like to improve in
 - **Write your own section/exam problem + solution**
 - Present the problem to your section leader at the end of the quarter
 - More guidelines will be released on July 20 after the diagnostic

What we will ask you to do



Section

- Sign up by **Tuesday at 5pm PDT** at cs198.stanford.edu
 - Sections with remaining spots will open for signups after Wednesday at 9am PDT

Section

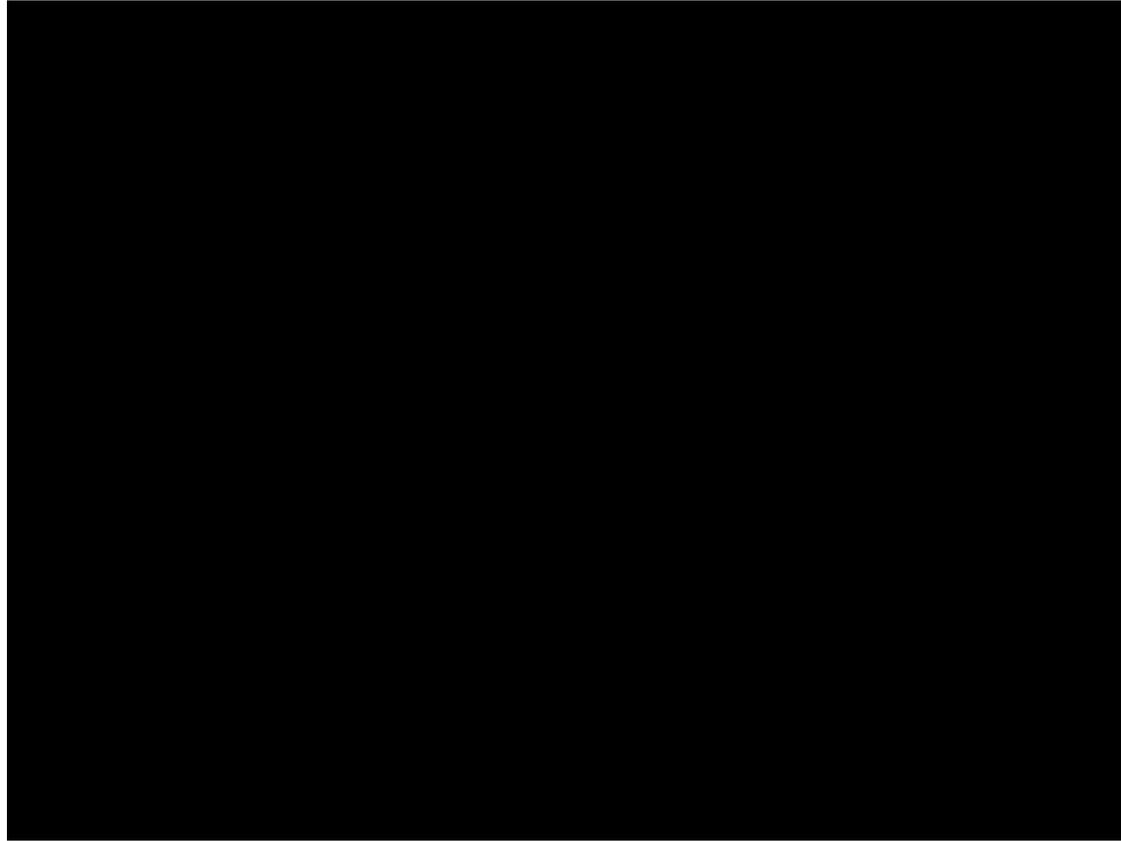
- Sign up by Tuesday at 5pm PDT at cs198.stanford.edu
- Sections start this Wednesday!

Section

- Sign up by Tuesday at 5pm PDT at cs198.stanford.edu
- Sections start this Wednesday!

How do I get help?

Trip Master
(Head TA)





Section Leaders



Staff who can work with minors (under 18)

- Kylie and Nick
- Lauren Saue-Fletcher
- Eric Bear
- Jonathan Kula
- Garrick Fernandez
- Kara Eng
- Ricardo Iglesias
- Sidhika Balachandar
- Nicholas Negrete
- Jesse Doan
- Jillian Tang



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)
 - Open Sunday through Wednesday, 5pm-9pm
 - Check for minors vs. non-minors LaIR hours (alternate by day)
 - Morning LaIR once a week for each group
 - Starts this Tuesday evening
- Your section leader
- Trip's office hours (no minors)
- Kylie's + Nick's office hours
- Ed

Resources for getting help

- LaIR (general office hours)
- Your section leader
- Trip's office hours (no minors)
- Kylie's + Nick'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/Nick/Trip office hours
- Ed

Resources for getting help

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

Conceptual question?

Resources for getting help

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

Conceptual question?

Resources for getting help

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

Debugging help + code questions?

Resources for getting help

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

*Administrative
questions?*

Resources for getting help

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

*General CS + life
questions?*

Resources for getting help

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

When in doubt, check the [Course Communication guidelines!](#)

Resources for getting help

- LaIR
- Your section leader
- Kylie/Nick/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.

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 submissions.
- We run code similarity software on all of your programs.
- 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++ 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

```
#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++

```
import sys

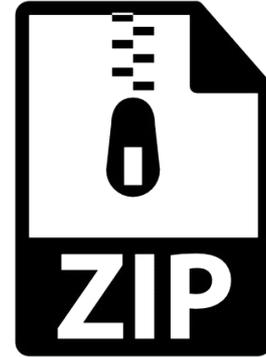
# This function does not need to be called "main"
def main():
    print('Hello, world!')

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



Announcements

- Complete the [C++ survey](#).
- Fill out your section time preferences by Tuesday at 5pm PDT.
 - Make sure to check what time you've been assigned on Wednesday morning.
- Finish [Assignment 0](#) by Wednesday.
 - If you're running into issues with Qt Creator, come to the Qt Installation Help Session tomorrow (Tuesday) at 7pm PDT.
- Assignment 1 will be released tomorrow!

Roadmap

C++ basics

User/client

vectors + grids

stacks + queues

sets + maps

Core
Tools

testing

algorithmic
analysis

recursive
problem-solving

Object-Oriented
Programming

Implementation

arrays

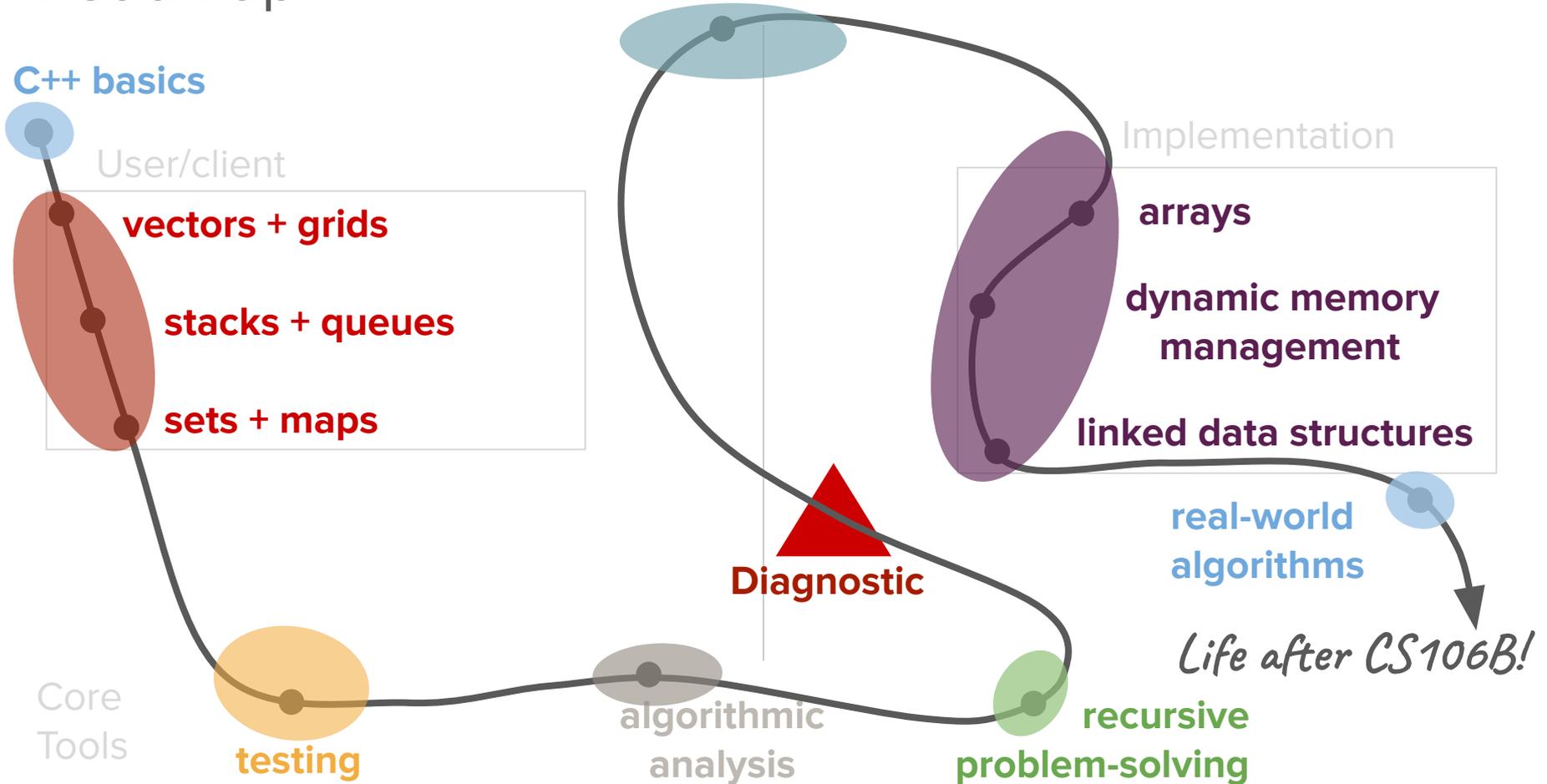
dynamic memory
management

linked data structures

real-world
algorithms

Life after CS106B!

Diagnostic



Roadmap

C++ basics

User/client

vectors + grids

stacks + queues

sets + maps

Core
Tools

testing

algorithmic
analysis

recursive
problem-solving

Object-Oriented
Programming

Implementation

arrays

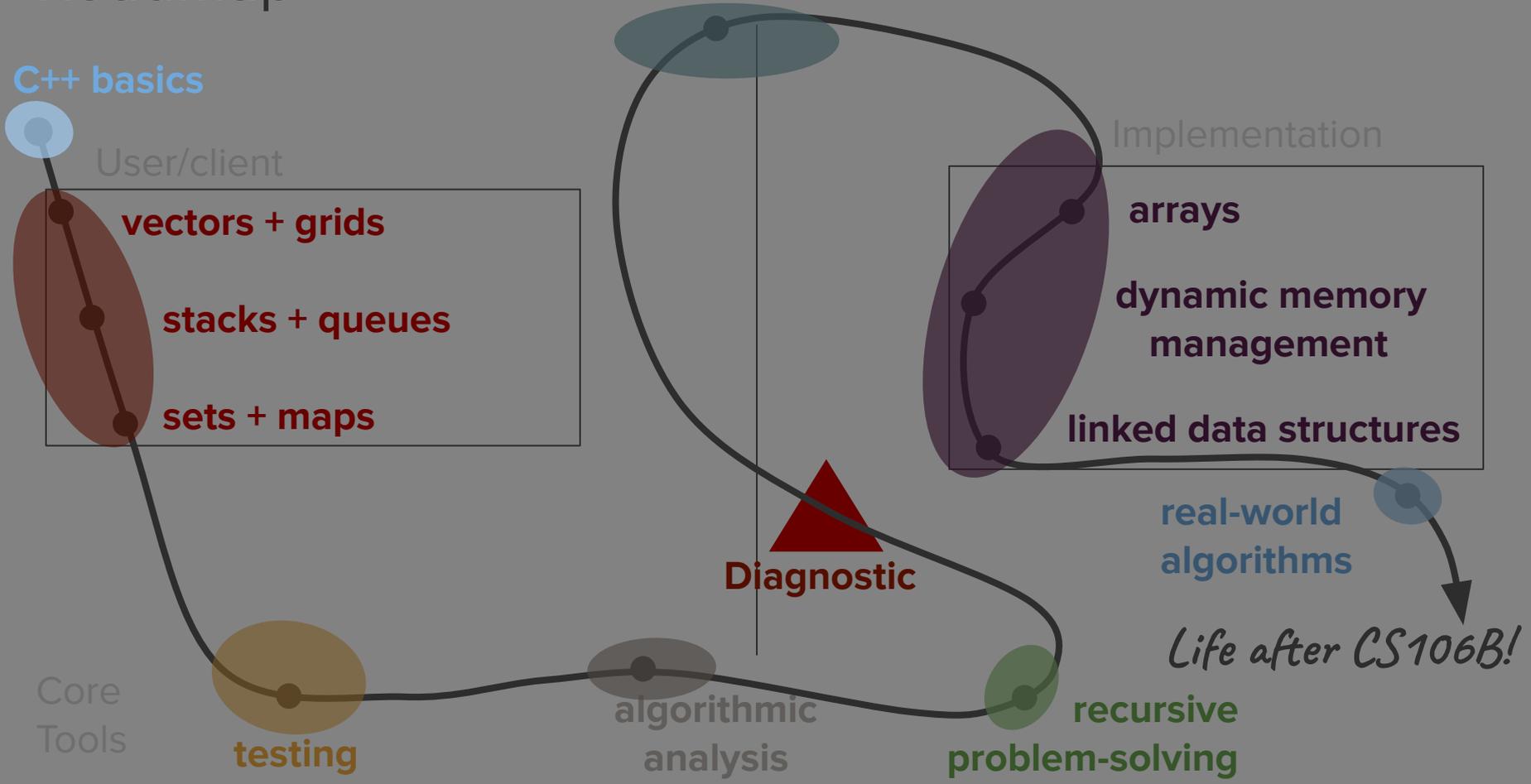
dynamic memory
management

linked data structures

real-world
algorithms

Life after CS106B!

D diagnostic



Roadmap

Object-Oriented Programming

C++ basics

User/client

vectors + grids

stacks + queues

sets + maps

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

arrays

dynamic memory management

linked data structures

real-world algorithms

Life after CS106B!

D diagnostic

Core Tools

testing

algorithmic analysis

recursive problem-solving