Welcome to CS106B: Programming Abstractions!

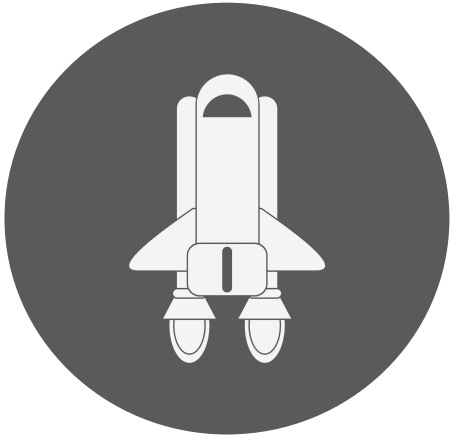
Where in the world are you right now?

(put your answers the chat)

Who are we?

Kylie Jue

Nick Bowman



Katie Creel

Today's questions

Why take CS106B?

What is an abstraction?

What is CS106B?

Why C++?

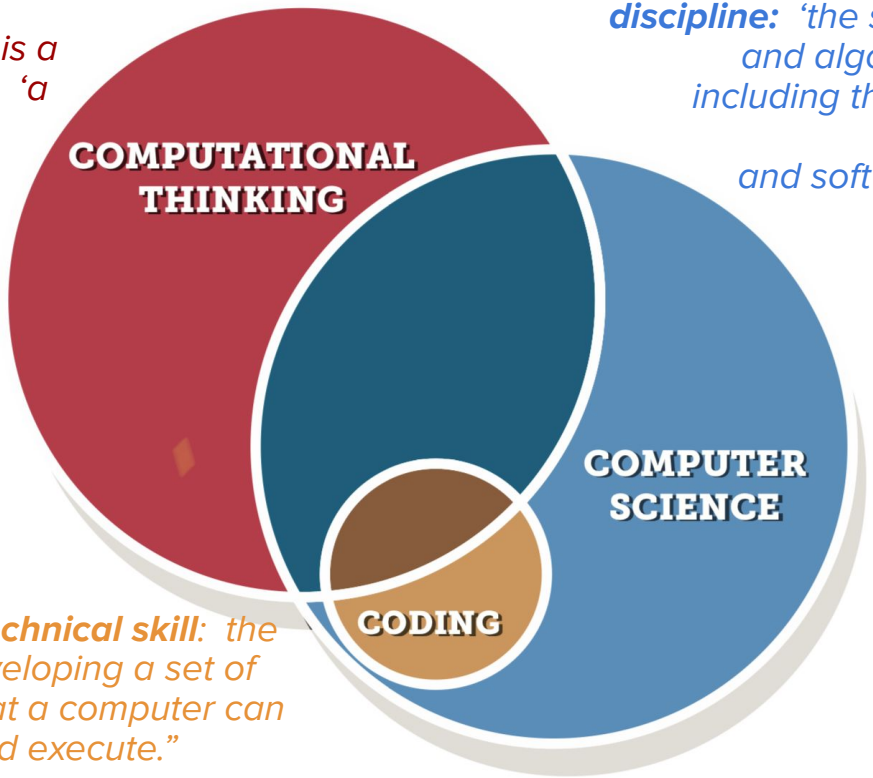
What's next?

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'"*

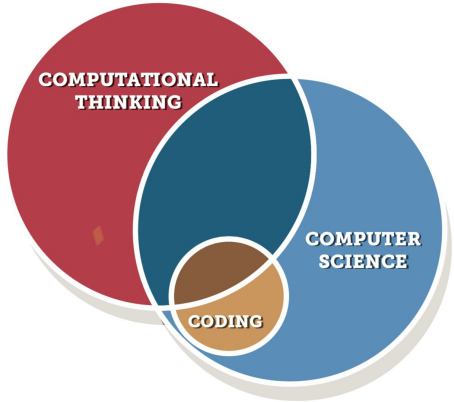


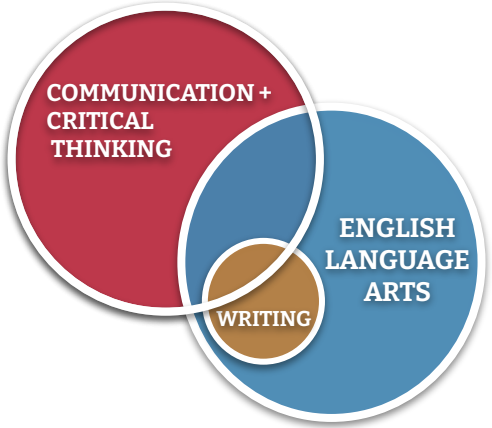
*"**Coding** is a **technical skill**: the practice of developing a set of instructions that a computer can understand and execute."*

(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

(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?

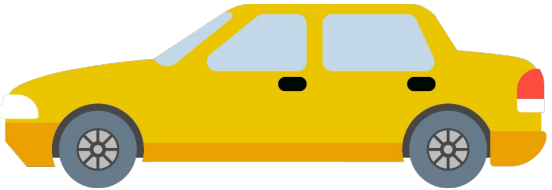
Breakout rooms!

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.

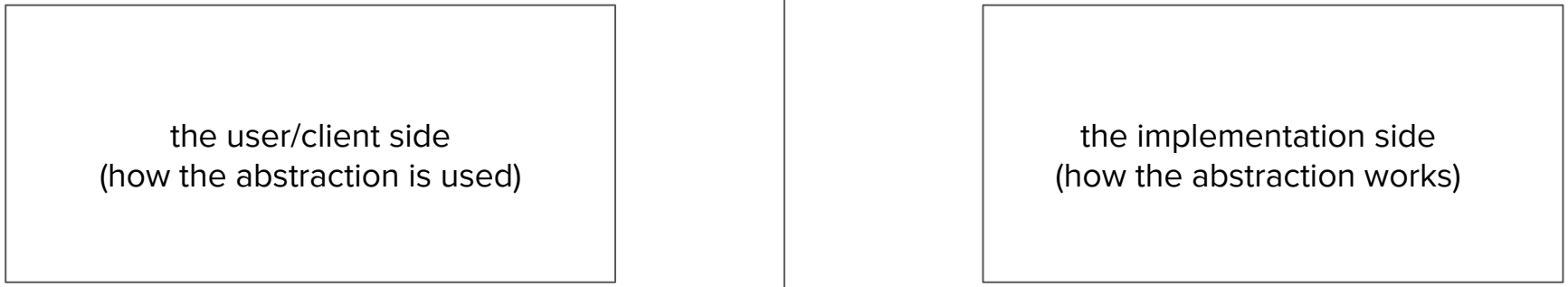
You start off by using other people’s recipes – tools that others have created to make it easy to prepare food and ensure you have sustenance.

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.

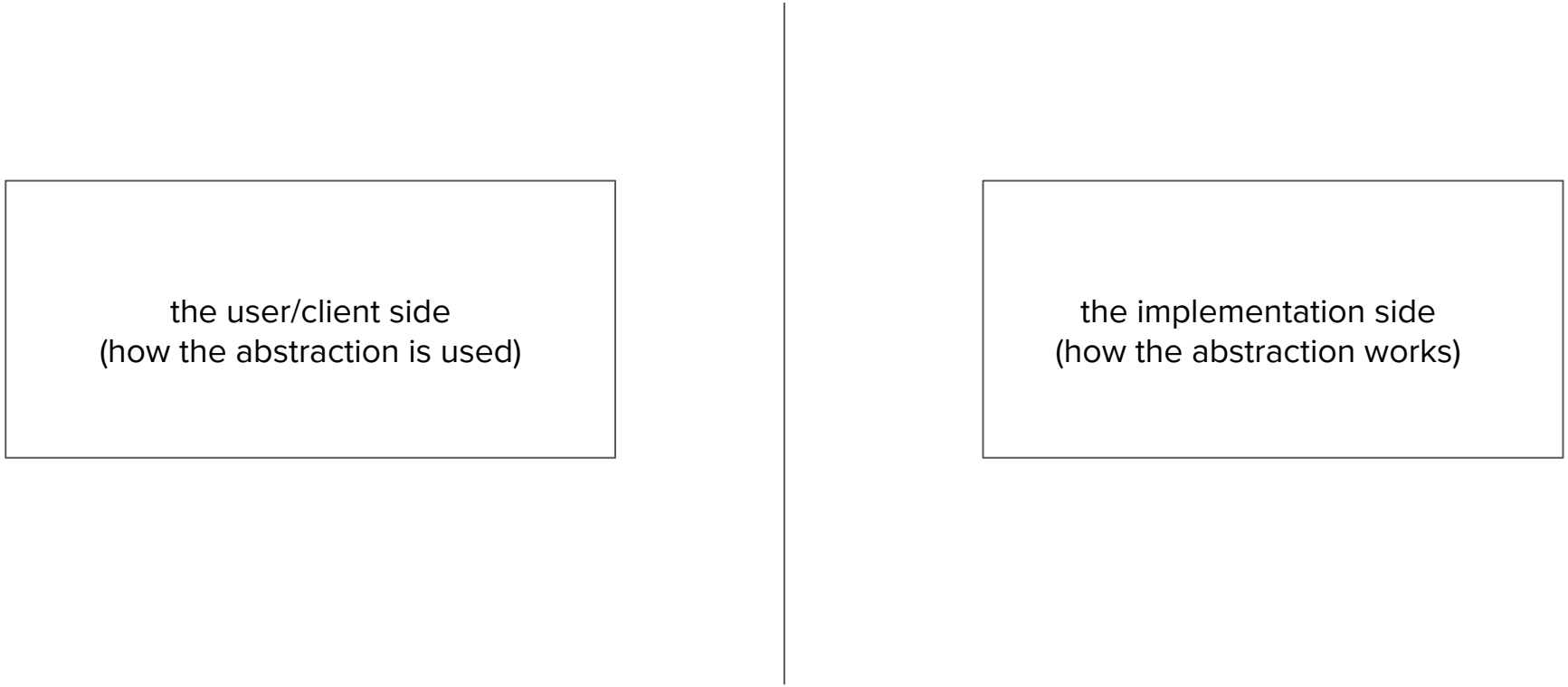
abstraction boundary
(what the abstraction looks like)



What is CS106B?

(the nuts and bolts)

abstraction boundary
(what the abstraction looks like)



classes

object-oriented programming

abstract data structures
(vectors, maps, etc.)

arrays

dynamic memory
management

linked data structures

testing

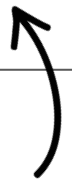
algorithmic analysis

recursive problem-solving

classes

object-oriented programming

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 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

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

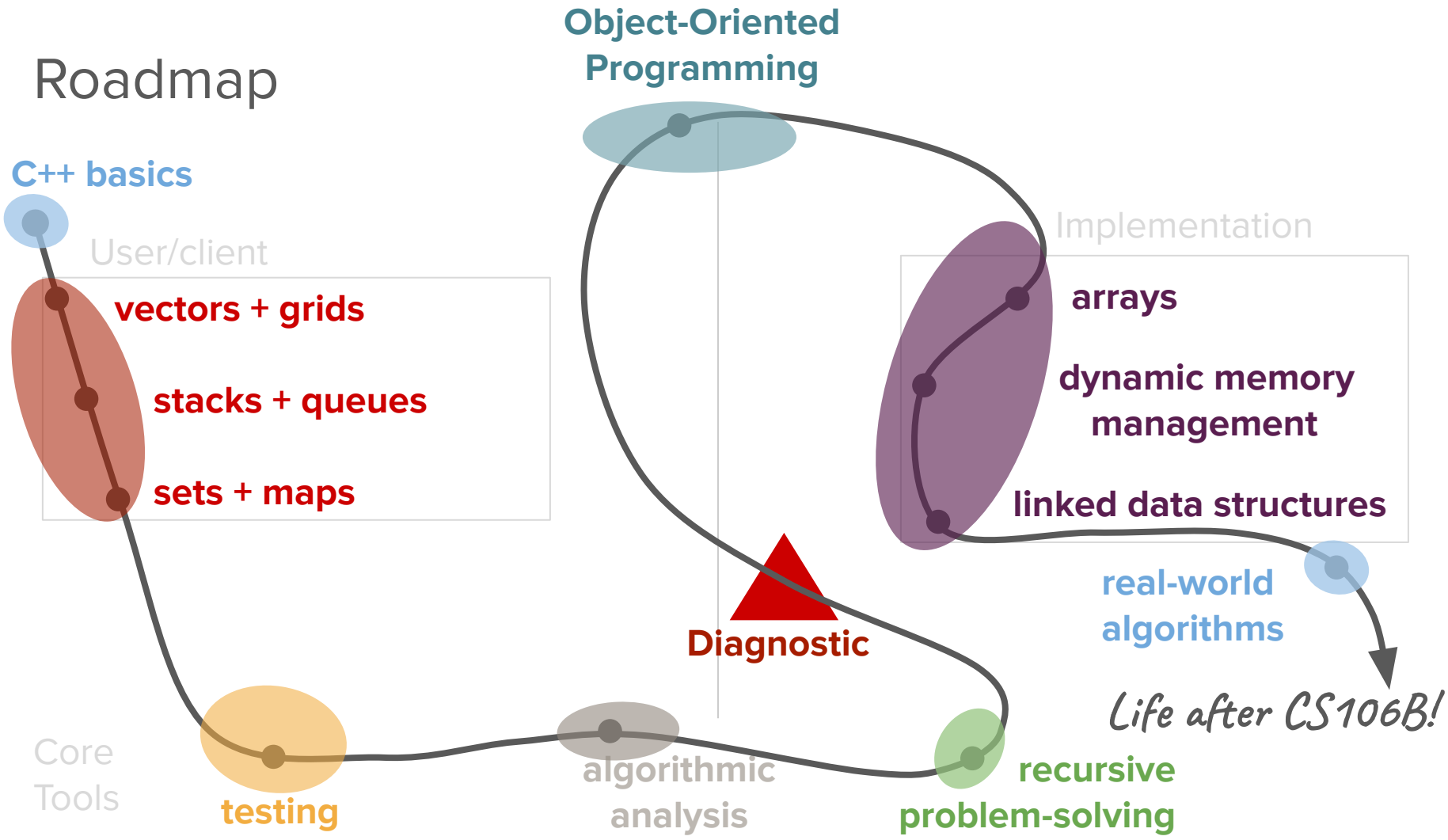
Core Tools

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
Diagnostic
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?

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 asking questions and let course staff answer them.

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

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 (or putting it in the chat if you're more comfortable with that). 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 it's clear we're transitioning to a new topic. We'll also often stop for questions then.

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: If you can answer the question yourself by writing a small piece of code to test your question, we encourage you to do that, too!

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.

Think about how to use questions to maximize your concentration and learning!

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!

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.

If you do have prior experience in C++ or in the topics we'll be covering, that's great! It also benefits your learning to approach these concepts with a beginner's mindset – you might notice and learn things that you didn't before.

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.

It also benefits your learning to approach these concepts with a beginner's mindset – you might notice and learn things that you didn't before.

Consider if lecture or individual office hours is the right venue for your question.

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.
    - **Note**: This is our first time with a (long) waitlist so please don't wait to drop if you know you won't be taking the class!

CS106B COURSE LECTURES ASSIGNMENTS SECTIONS ASSESSMENTS SCHEDULE SEARCH

CS106B Programming Abstractions

Summer Quarter 2021

Live lectures on Zoom MWF 11:30am PT

TEACHING TEAM

Nick Bowman

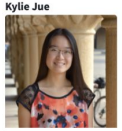
Instructor

nbowman@

M 1:30pm-3:30pm

Th 10:00am-12:00pm

Kylie Jue



Instructor

kyliej@cs

ANNOUNCEMENTS

Summer Quarter 2021

yesterday by Nick

This is the course website for CS106B Summer Quarter 2021. This **website is under construction** in preparation for our start on June 21; please pardon our dust as we work. In the meantime, if you are a prospective student looking to learn more about CS106B, check out the course syllabus and our answers to frequently asked questions from prospective students.

Previous quarter's website

yesterday by Nick

Here is the archived website for spring quarter.

See older announcements.

QUICK

? Ge

Of

La

Pa

Zo

Dis

Ho

Tu

Qt

Co

C+

St

Gu

Su

Co

cs106b.stanford.edu

ed CS106B – Discussion

New Thread

Search

Filter

COURSES

CS106B

CS 106B 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 – Assign 0 Shirley Li 10h 6

Unable to Access Google Form

Assignments – Assign 0 Michael Hu 16h 2

Mystery number rejected by Google form

Assignments – Assign 0 Richie Ling 1d 2

sample-project Build Error

General Anonymous 1d 3

Debugger not working

Assignments – Assign 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 Fo

Nick Bowman INSTRUCTOR

3 days ago in Announcements

UNPIN STAR

12

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

Getting Started

Here is the Quick Start Guide to using Ed Discussion. We strongly recomm this guide before you start exploring the website for yourself and getting all the different features that are offered.

Community Norms and Expectations

In order to cultivate the online experience for all students here, we have guidelines that we want to establish as community expectations for using

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

https://us.edstem.org/

How many units?

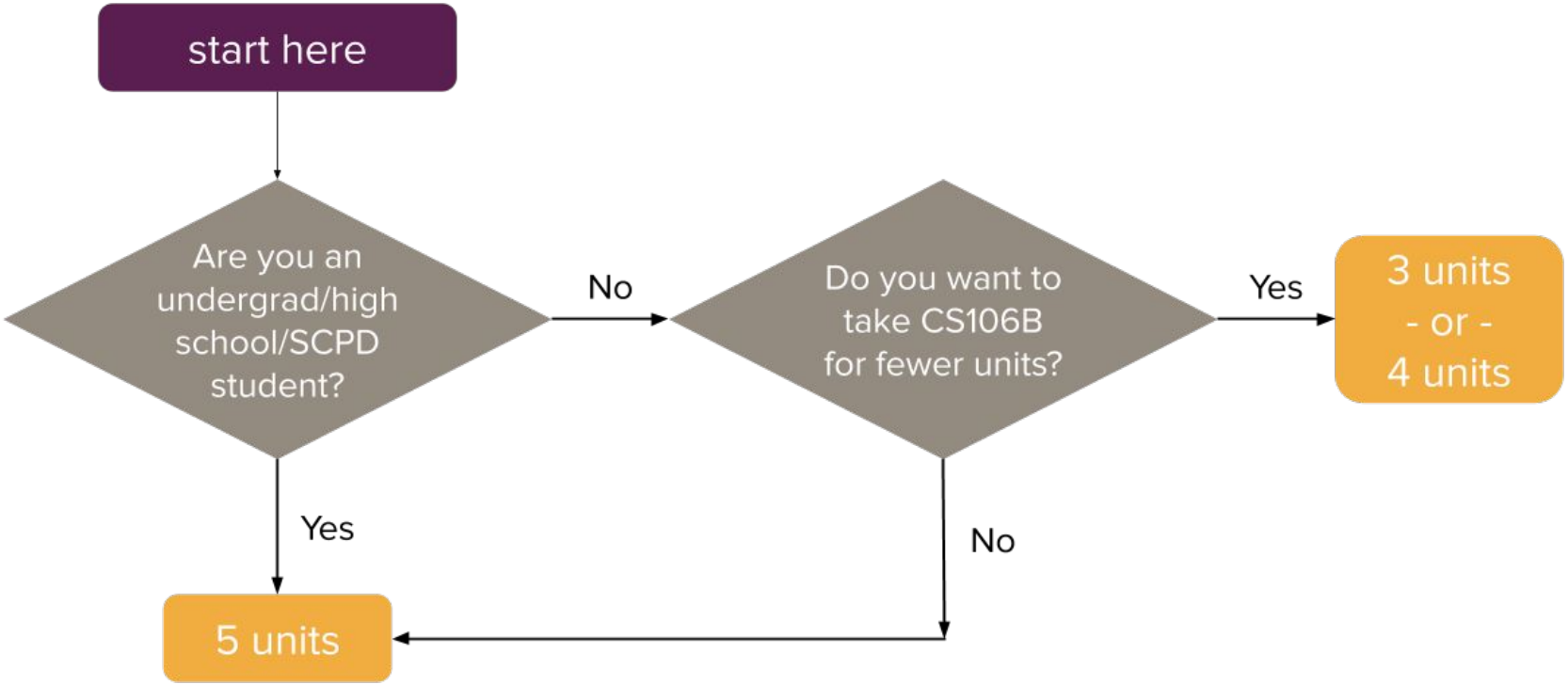


Diagram courtesy of Chris Piech

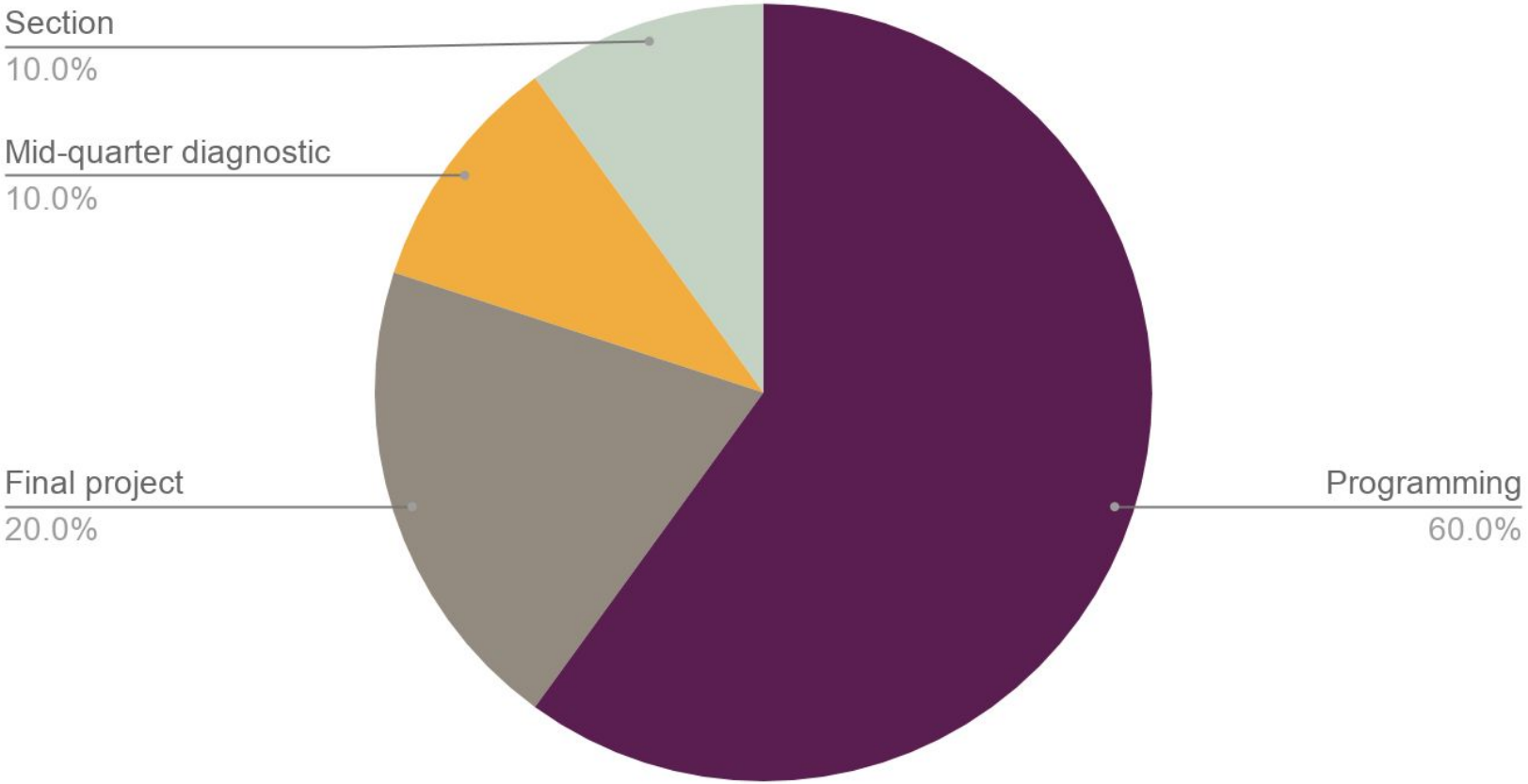
Why should I come to lecture?

Lecture pedagogy

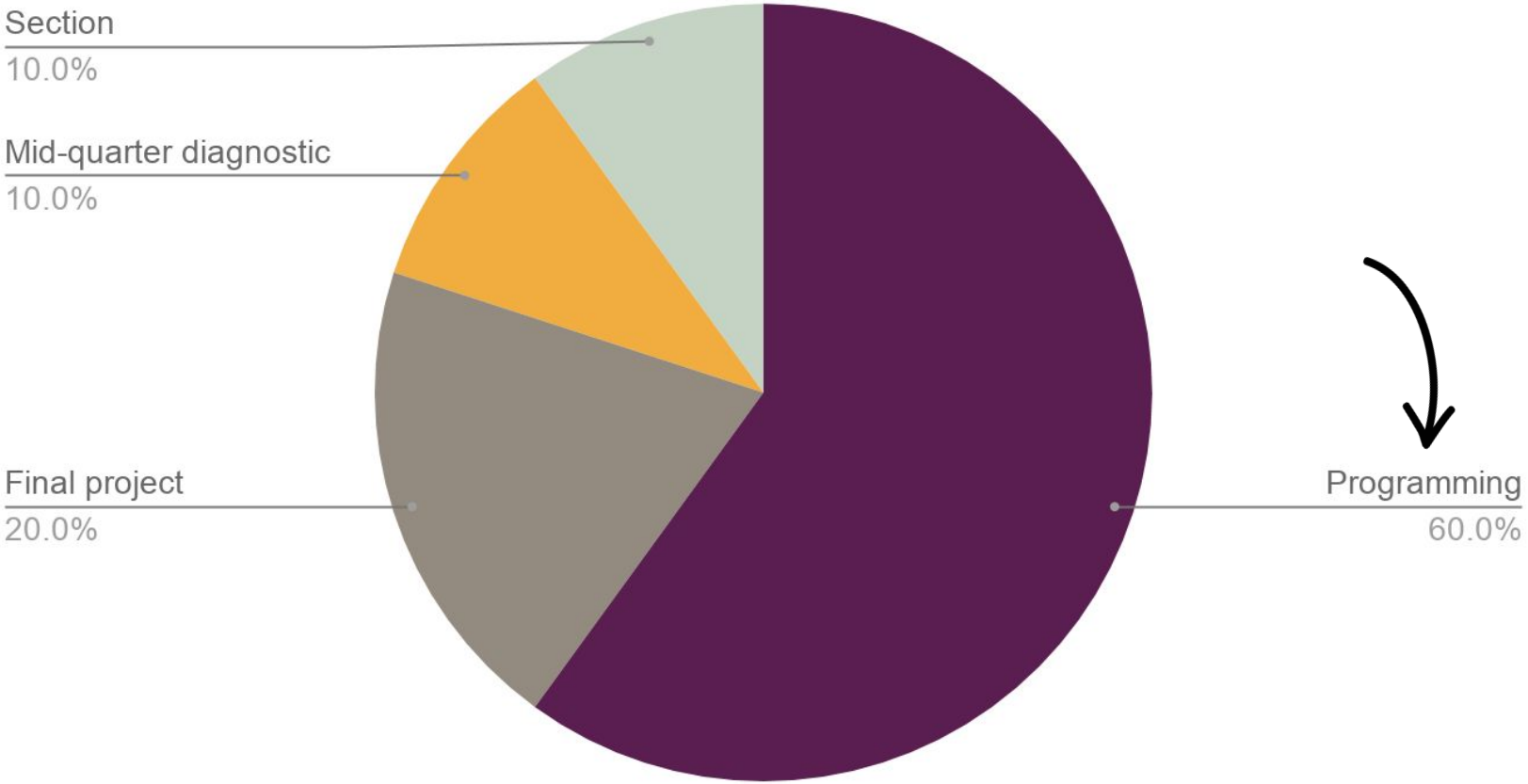
- Not just us talking at you: active learning exercises
- Quick lecture-to-usage turnaround for concepts covered in class
- We’ll stick around to answer questions afterward!
- Please note that this is a 60-minute long class.

How will I be assessed?

What we will ask you to do



What we will ask you to do



Programming assignments

- There will be 7 total
  - A1: C++ Legs
  - A2: Using abstractions (abstract data structures)
  - A3: Recursion
  - A4: Backtracking recursion
  - A5: Defining the abstraction boundary itself
  - A6: Implementation-side of the abstraction boundary
  - A7: Real-world algorithms

Programming assignments

- There will be 7 total
- Graded on **functionality** and **style** using buckets

✓	Meets requirements, possibly with a few small problems

Programming assignments

- There will be 7 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 7 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 7 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 7 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 7 total
- Graded on functionality and style using buckets
- You can submit revisions if you receive below a check
- 24- or 48-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- or 48-hour extension that you can use if you have a particularly difficult week.

Programming assignments

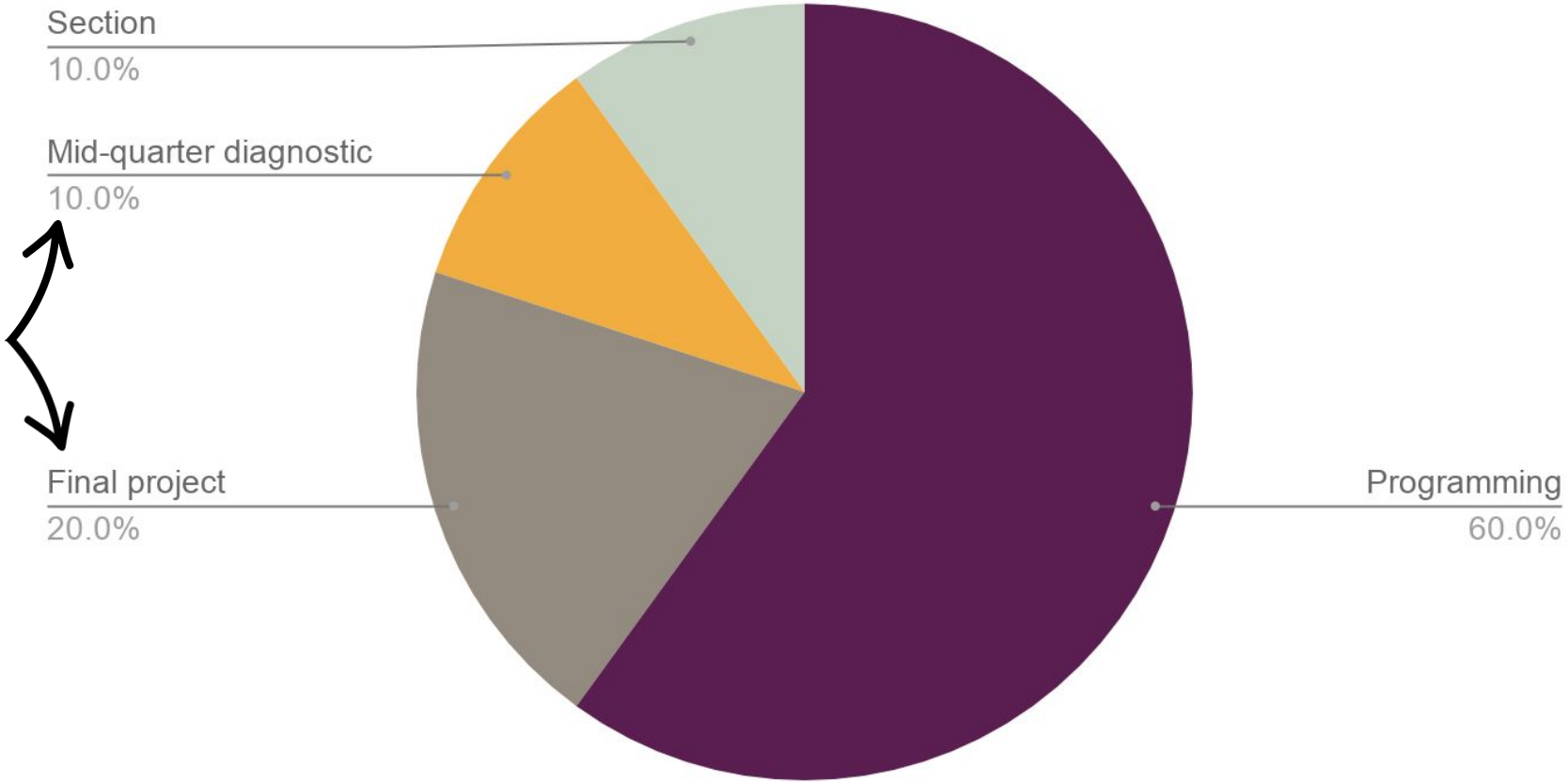
- There will be 7 total
- Graded on functionality and style using buckets
- You can submit revisions if you receive below a check
- 24- or 48-hour grace period for each assignment

Programming assignments

- There will be 7 total
- Graded on functionality and style using buckets
- You can submit revisions if you receive below a check
- 24- or 48-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 diagnostic
- Final project

Assessments

- Mid-quarter diagnostic
  - Opportunity to **evaluate your understanding of the core, fundamental topics** from the first 4 weeks of the course
  - Designed to take 1.5 hours; completely open notes
  - Available to complete over a 47-hour time span from July 21-23 (between Wednesday and Friday lecture)
  - We'll provide software for you to take the diagnostic on your computer – once you open it, you'll have 3 hours to complete it

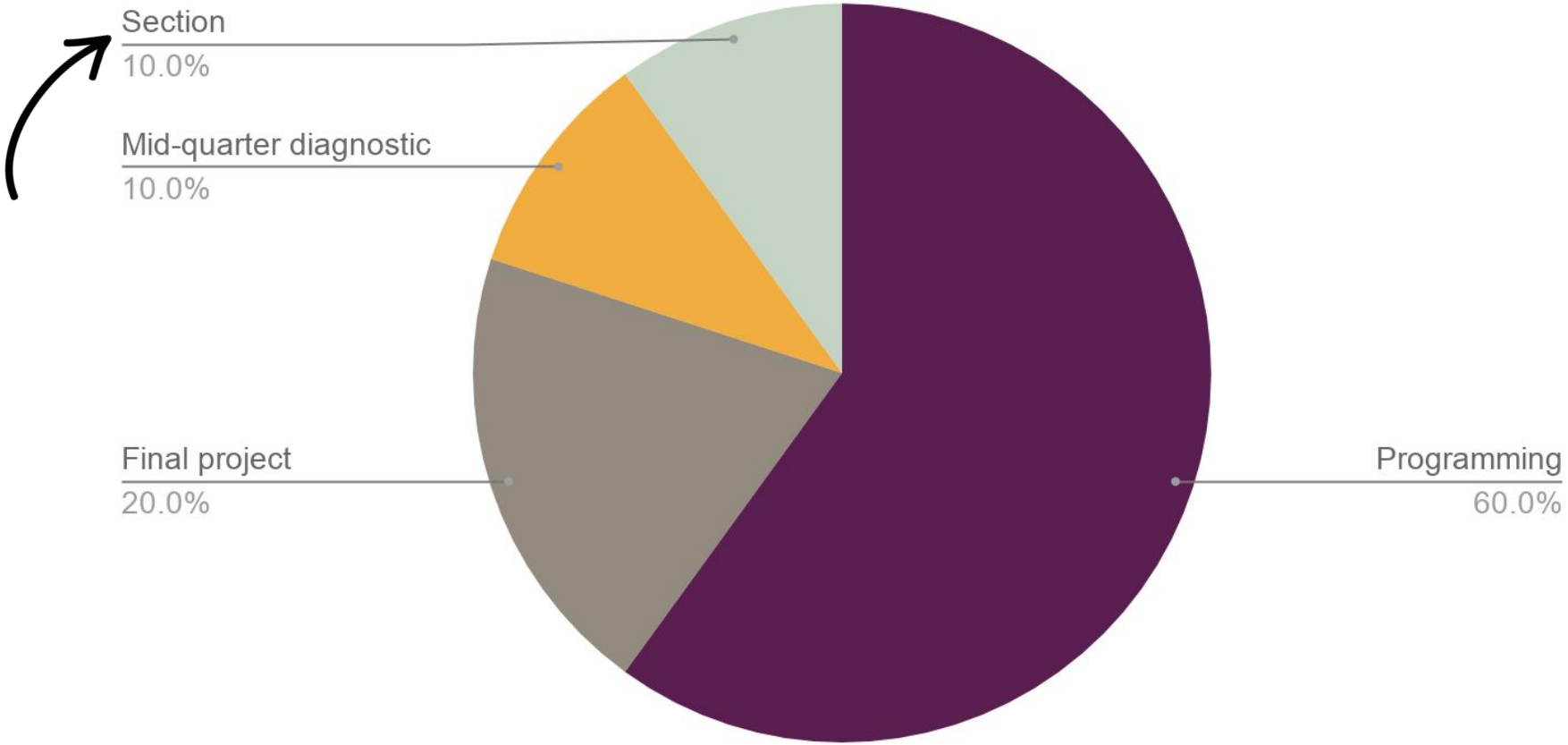
- Final project

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/diagnostic problem + solution**
  - Present the problem to your section leader at the end of the quarter
  - More guidelines will be released on July 26 after the diagnostic

What we will ask you to do



Section

- Sign up by **Sunday at 5pm PDT** at cs198.stanford.edu
  - Sign-ups will open on Thursday, June 24 at 5pm PDT
  - Sections with remaining spots will open for signups after Tuesday, June 29 at 9am PDT

Section

- Sign up by Sunday at 5pm PDT at cs198.stanford.edu

- Sections start next Wednesday!

Section

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

How do I get help?














Section Leaders



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 Monday through Thursday
    - Monday/Wednesday: 7pm-9pm PDT
    - Tuesday/Thursday: 5pm-7pm PDT
  - Starts Monday, June 28
- Your section leader
- Kylie’s + Nick’s office hours
- Ed

Resources for getting help

- LaIR (general office hours)
- Your section leader
- 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 office hours
- Ed

Resources for getting help

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

Conceptual question?

Resources for getting help

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

Conceptual question?

Resources for getting help

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

Debugging help + code questions?

Resources for getting help

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

Administrative questions?

Resources for getting help

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

General CS + life questions?

Resources for getting help

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

When in doubt, check the Course Communication guidelines!

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++ is 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
}
```

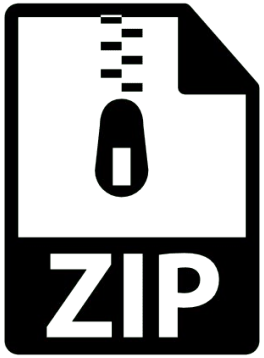
C++

```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



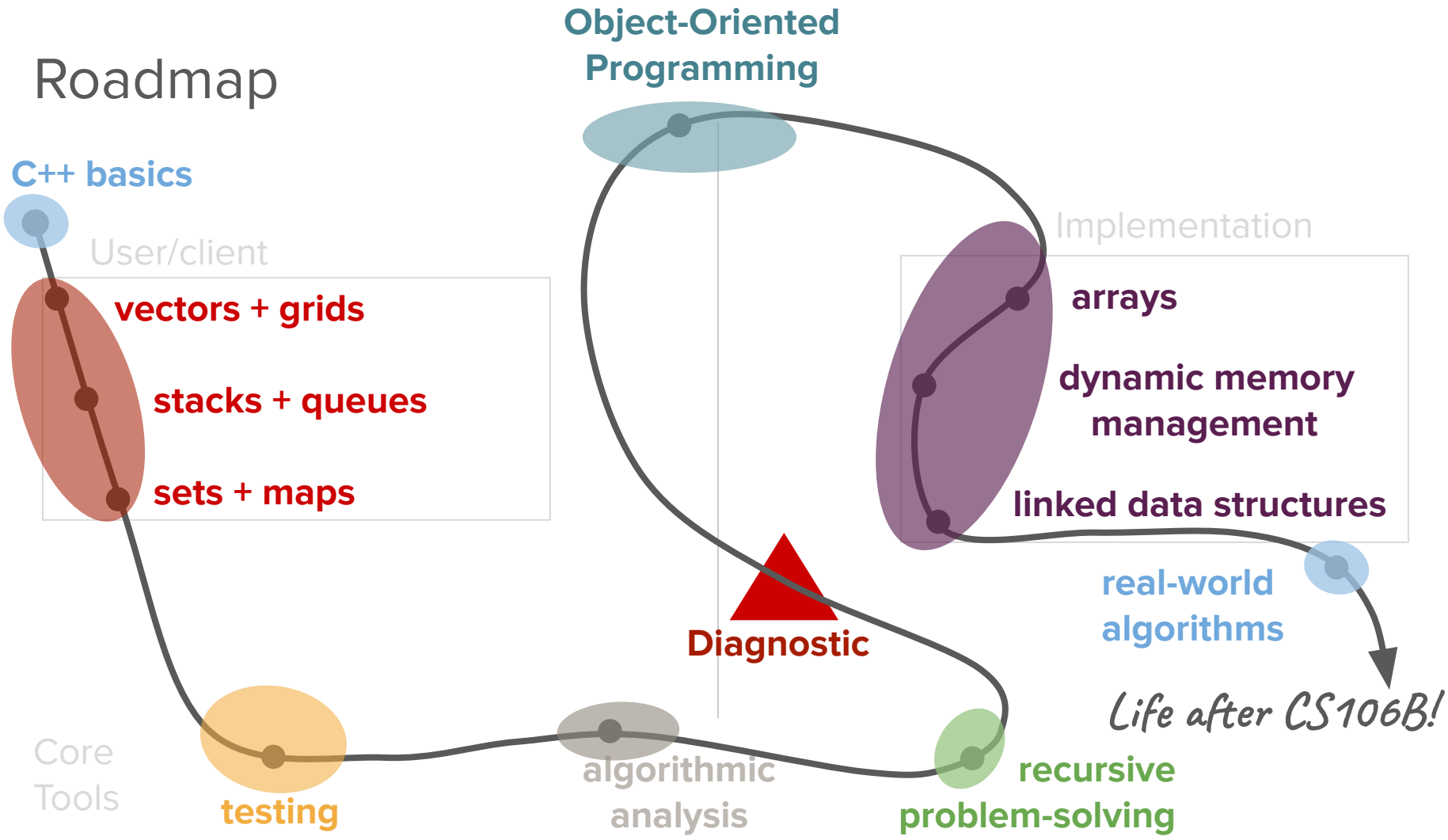
Reminders

- Complete the C++ survey.

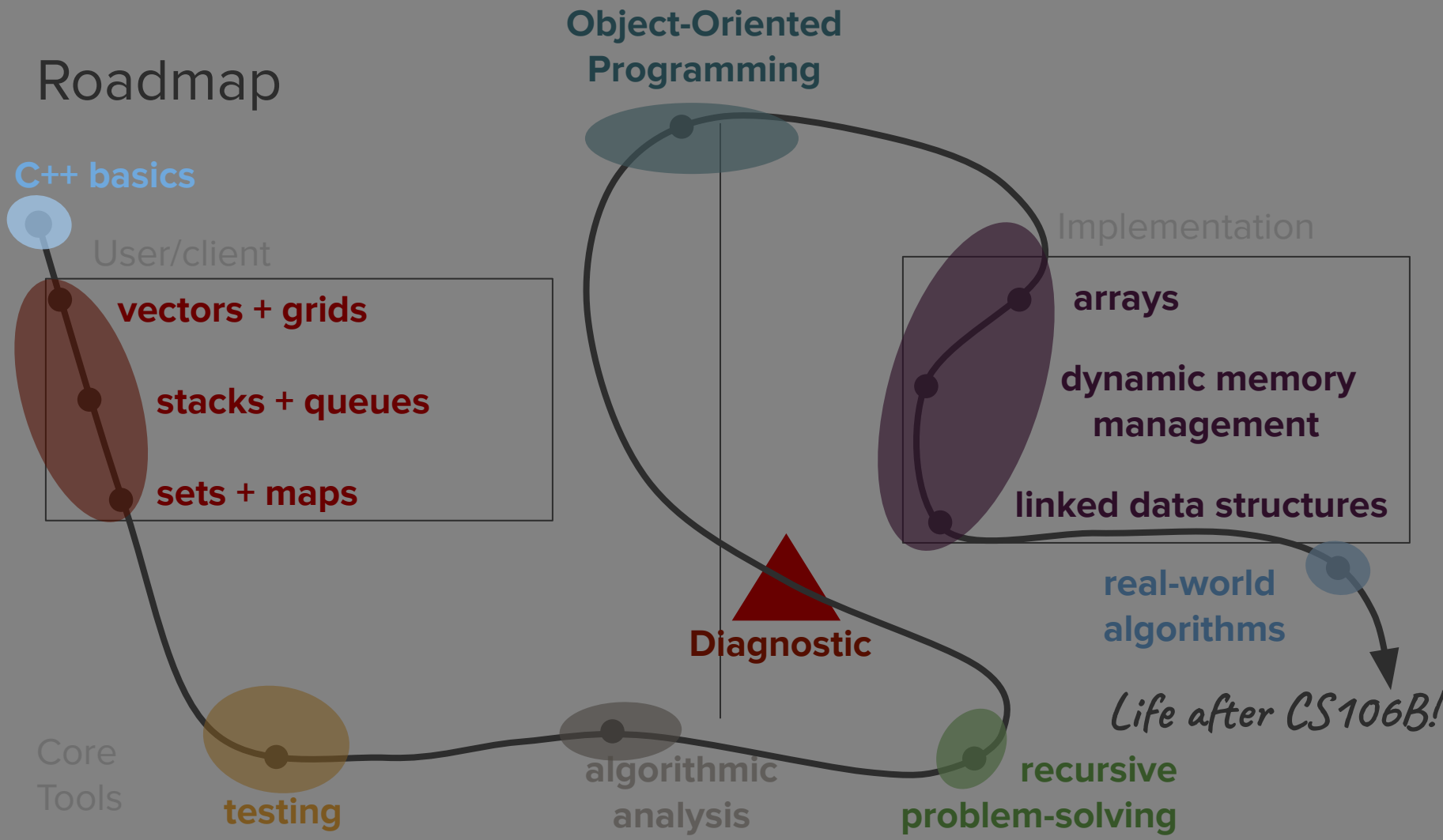
- Fill out your section time preferences by Sunday at 5pm PDT.
  - Make sure to check what time you've been assigned on the morning of Wednesday, June 30.

- Finish Assignment 0 by Friday.
  - If you're running into issues with Qt Creator, come to the Qt Installation Help Session Thursday from 5-7pm PDT.

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
Diagnostic
Life after CS106B!
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
Diagnostic
Life after CS106B!
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
We're excited to move across the abstraction boundary together!
Diagnostic
real-world algorithms
Life after CS106B!
Core Tools
testing
algorithmic analysis
recursive problem-solving