Syllabus & Algorithms

15-110 – Monday 08/31

Learning Objectives

- Understand the **expectations**, **resources**, and **policies** associated with 15-110
- Define the essential components of computer science, **algorithms** and **abstraction**
- **Follow steps** provided by an algorithm to perform specific tasks

Course Introduction

Purpose of 15-110

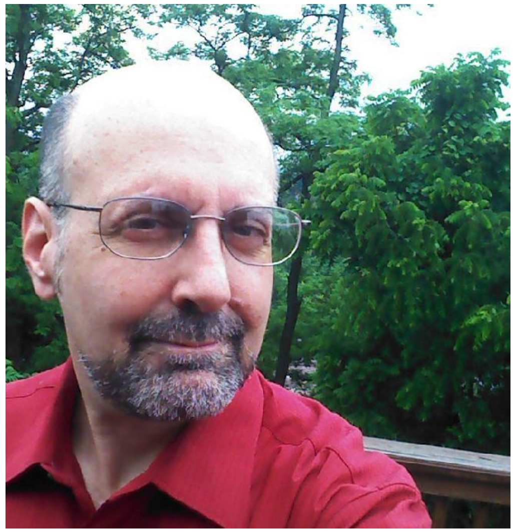
The goal of this course is to introduce you to the field of **computer science**. This includes both programming and more general algorithmic concepts.

The course is broken into five units:

- Programming Skills and Computer Organization
- Data Structures and Efficiency
- Scaling Up Computing
- CS as a Tool
- CS in the World

Staff

Kelly Rivers



Dave Touretzky

28 Teaching Assistants!

Course Interaction

Lecture: remote, via Zoom

- use your CMU Zoom account
- video on, audio off
- questions via chat & raised hand

Recitation: in-person **or** remote

- **in person:** wear a mask!!! Physically distance & monitor health
- **remote:** see lecture

How to Learn in 15-110

1. Attend class. If you can't attend live, watch the posted recording promptly.
2. Complete the **quiz** associated with the lecture.
3. Complete the **check-in/homework assignment** associated with the lecture content.
4. Demonstrate your knowledge in the **test** associated with the lecture.
5. Demonstrate all collected knowledge in the **final exam**.

Important Resources

Course website: www.cs.cmu.edu/~110/

Also:

- Piazza – announcements, questions
- OH Queue – one-on-one OH
- Gradescope – take quizzes, submit assignments, read feedback
- Canvas – take tests, see grades

Collaboration Policy

We encourage you to collaborate on the assignments! We'll hold **collaboration hours** where you can meet other students in the class to find collaborators.

When you collaborate, **all students** should contribute intellectually to the work, and **each student** must write up their solutions independently.

The following actions count as cheating, not collaboration, and lead to penalties: copying, providing answers to others, searching for answers online, collaborating during tests/the final exam.

Frequently Asked Questions - Placement

- **I have no prior programming experience. Can I succeed in this class?**
  - Most of your classmates have no prior experience as well. You can definitely succeed, and you're not alone!

- **Should I take 15-110 or 15-112?**
  - 15-110 gives a broad overview of programming and computer science. 15-112 focuses more deeply on programming and problem solving. 15-112 also moves through the core elements of programming (data, conditionals, loops) very quickly.
  - Feel free to talk to the professors if you want advice on your individual situation.

Frequently Asked Questions – Deadlines

- **What if I need to turn something in late?**
  - Each assignment has a regular deadline and a **revision deadline**. Submissions received between the regular deadline and the revision deadline are graded for a max of 90 points.
  - Submissions made before the regular deadline may also be resubmitted for a max of 90 points. Submit early and get feedback so you can fix your mistakes!

- **What if I still can't finish something by the revision deadline?**
  - The revision deadline is placed right before the test associated with the content. Therefore, it is a hard deadline. Submit whatever you have done, even if it is incomplete.
  - Students in exceptional situations (COVID/medical/family/personal emergencies) may reach out to the instructors to arrange extensions.

Take care of yourself!

Taking care of yourself is incredibly important, especially in tumultuous times. Your personal wellbeing is more important than academics.

Make sure you regularly eat healthy food, get enough sleep, exercise, socialize, and take some time to relax. You will be happier, and you will do better academically as a direct result.

We want everyone to feel welcomed and capable of learning in 15-110. If you feel that the course is negatively impacting your wellbeing, or you do not feel included, **reach out and let us know**.

Take-Home Task

Before Wednesday, do the following:

- Read the course syllabus.
  - It includes many details we did not cover here. Seriously, read it!
- Install the **Python programming language** and the **Pyzo IDE** onto your computer
  - If you don't have a computer and you're on campus, the cluster machines already have this software installed. If you do not have a computer and you're remote, contact the professors immediately for advice.
  - Instructions can be found here.

Algorithms

What is Computer Science?

Video: Computer Science is Changing Everything

https://www.youtube.com/watch?v=QvyTEx1wyOY

Computers are impacting many different aspects of the world - medicine, engineering, shopping, safety, politics, friendships, and so much more.

The core of computer science boils down to two main ideas: **algorithms** and **abstraction**.

Algorithms and Abstraction

Algorithms are procedures that specify how to do a needed task or solve a problem.

Algorithms are used to standardize processes and communicate them between different people.

Algorithms are like recipes, tax codes, and sewing patterns.

Abstraction is a technique used to make complex systems manageable by reducing the amount of detail used to represent or interact with the system.

This can be done by identifying the most important features of a system and generalizing away unessential features.

For example, consider that an image on your computer may be a JPEG, GIF, or PNG file, yet all can be edited by the same application. The application must implement support for each new filetype, but you see a single edit option for all of them. The details are abstracted away!

Activity – Make a PB & J Sandwich

You do: work with your breakout group to write a list of instructions (an algorithm!) on how to make a peanut butter and jelly sandwich.

We'll test your instructions in a few minutes...

Making a Peanut Butter and Jelly Sandwich

1. Open bag of bread
2. Reach hand in and take out 2 slices of bread
3. Place each slice on a plate
4. Open jar of peanut butter
5. Pick up knife and stick sharp side of knife into open jar
6. Use knife to scoop out peanut butter
7. Wipe and spread peanut butter on one slice of bread
8. Repeat 5, 6, 7 until slice of bread is covered in peanut butter. Close jar
9. Open jar of jelly
10. Pick up knife and stick sharp side of knife into open jar
11. Use knife to scoop out jelly
12. Wipe and spread jelly on other (non-PB) slice of bread
13. Repeat 10, 11, 12 until the slice of bread is covered in jelly. Close jar.
14. Put the peanut butter side of one slice of bread on the jelly side of the other.
15. Pick up and eat sandwich.

Designing Good Algorithms

Designing good algorithms is a large part of computer science. When we represent algorithms as **program code**, we are communicating with a computer to tell it how to do a specific task.

What makes an algorithm 'good'?

- It should specify what is needed at the beginning (**input**)
- It should specify what is produced at the end (**output**)
- It should produce the right output for all given inputs (**correctness**)
- It shouldn't take too long to finish (**efficiency**)
- Others should be able to understand and modify it as needed (**clarity**)

Algorithm Example – Calculating Age

Let's say we want to write a basic algorithm that calculates someone's age based on their birth date.

What's the **input**?

Their birth date, and the current day.

What's the **output**?

A number (their age).

Algorithm Example – Calculating Age

1. Subtract the birth date year from the current year.

Is there an example that won't work with just Step #1?
What if someone was born on December 31st?

1. Subtract the birth date year from the current year.
2. If the month/day of the birth date comes after today's date, subtract 1 from the result.

We only take the action in Step #2 **if** the assumption is true. This is a common algorithmic technique.

Algorithm Example – New Year's Countdown

Now let's say we want to write an algorithm that simulates a New Year's Eve countdown (from 10 to 1).

What is the **input**?

There is no input!

What is the **output**?

The numbers from 10 to 1, backwards.

Algorithm Example – New Year's Countdown

We could write out ten output lines, but that would be tedious.

Alternatively, we could just tell the reader to **repeat** steps!

1. Set a value **counter** to be 10
2. If **counter** is greater than 0, do the following. Otherwise, go to Step 3.
    a) Output **counter**
    b) Subtract 1 from **counter**
    c) Repeat Step 2 with new values
3. Output the text "Happy New Year!"

Algorithm Example – Following Steps

Now let's try walking through the steps of the New Year's algorithm, to see if it produces the correct result.

1. Set a value **counter** to be 10
2. If **counter** is greater than 0, do the following. Otherwise, go to Step 3.
    a) Output **counter**
    b) Subtract 1 from **counter**
    c) Repeat Step 2 with new values
3. Output the text "Happy New Year!"

Post-Lecture Feedback

We encourage all students to fill out the **post-lecture feedback form** after each lecture, to let us know what you're still confused about.

We'll review common confusion points at the beginning of the following lectures.

Link: https://bit.ly/110-feedback

Learning Objectives

- Understand the **expectations**, **resources**, and **policies** associated with 15-110
- Define the essential components of computer science, **algorithms** and **abstraction**
- **Follow steps** provided by an algorithm to perform specific tasks