15-110: COURSE OVERVIEW

Introductions

- Instructor: Norman Bier
- TA Team
  - Rhea Kudtarkar
  - Elizabeth La
  - Jiachen Ning
  - Brent Hong
  - Jonan Seely
  - Freda Ding

Introductions

- Who are you?
- What are you studying?
- What do you hope to get from this course?

Students From Different Disciplines

- Basic Sciences
- Engineering
- Psychology
- Business
- Modern Languages
- Architecture
- Others ...

Why Are You Here?

- **Curiosity:** find out about computing technology and its many effects on society.
- **Professional development:** computing skills can make you more successful at work.
- **Academic requirement:** a computing course is required for your major. Why?
- **Intellectual growth:** computing changes how we think about problems. You can learn to think like a computer scientist.

Computation

- Computer science is the study of what can be computed and how to compute it:
  - Computation: Performance of a sequence of simple, well-defined steps that lead to the solution of a problem
  - A computer: Performs steps and remembers the results of those steps

What Kind of a Discipline is Computer Science?

- Science: focuses on abstract, artificial things in a virtual world.
- Engineering: Building complex things by using techniques to manage complexity
- Liberal arts: Strong connections to traditional liberal arts of grammar, rhetoric, logic, arithmetic, geometry, music

High-level Goals of the Course

- When you think like a computer scientist you will be able to
  - Identify problems that are amenable to computation and express computations to find a solution
  - Understand the power and limitations of computational tools and techniques
  - Ask new questions that were not thought of or dared to ask because of scale, easily addressed computationally

Skills to Be Gained

- Systematic problem solving, applying abstractions as needed
- Reading, writing, and debugging small to medium-sized programs using the language Python
- Familiarity with computational concepts underlying pervasive technologies
- Familiarity with computational vocabulary

In their capacity as a tool, computers will be but a ripple on the surface of our culture. In their capacity as intellectual challenge, they are without precedent in the history of mankind.

Edsger Dijkstra,
1972 Turing Award Lecture

Course Information

Course Organization

- Instructor:
  - Norman Bier
  - Email is welcome. Please use [15-110] in the subject line
- Lectures: MTWThF
  - 9:00-10:20
- 2 Lab/Recitation sections
- 6 Teaching Assistants (TAs) to help you!

Lab Information

Lab (Section U - undergraduates)	MTWRF 4:30-5:20PM*	GHC 5208	TA:
Lab (Section E1 - APEA students)	MTWRF 4:30-5:20PM*	GHC 5210	TA:
Lab (Section E2 - APEA students)	MTWRF 4:30-5:20PM*	GHC 5207	TA:

Resources

Via Canvas(cmu.edu/canvas):

- Course web site
- Open Learning Initiative course
- Piazza: course message board
- Autolab
- Gradescope

Office Hours

- By instructor after class & by appointment
- By Teaching Assistants after class and in evenings

- Schedule is near-final; minor adjustments can still occur. See course web page for schedules.

Textbooks

- There is no designated textbook.
- See the course web page for recommended books.

Assignments

- OLI materials
- Labs: do in recitation hand in using Autolab.
- Written problem sets: hand in using Gradescope
- Programming assignments:
  - Pre-published
  - Due by end of day (11:59 PM)
  - Handed in using Autolab

Late Policy

- Assignments must be handed in on time.
  - Late assignments receive a grade of 0.
- We will drop 1 written assignment and 1 programming assignment without penalty (except where noted)
- We will drop 2 labs without any penalty.

Exams

- You must take all the exams, at the time they are given.
- No makeups except for extreme circumstances (major illness, death in immediate family, or a university-sanctioned event with documentation and prior permission)
  - 2 Lab Exams
  - 2 Written Exams
  - Final Exam

Grading

OLI Materials: 10%

Homework Assignments: 25%

Lab Participation: 5%

2 Lab Exams: 10% (5% each)

2 Written Exams: 30% (15% each)

Final Exam:20%

Expected Effort

- We assume that you have no prior knowledge in computing.
- Remarkably fast paced, aggressive schedule
- Expect to put in 12-15 hours a week outside of class

How aggressive?

- Summer Session: 2.5X times as fast
- What to do if questions or concerns?
- Can you catch up?

Academic Integrity Policy

- University Policy on Cheating and Plagiarism
- Complete OLI Course module, including quiz.
- Academic Integrity Form
  - On the SYLLABUS page of the course web page.
  - Print it out.
  - Read it.
  - Sign it.
  - **Bring it to class on Thursday, July 5th**

Getting Started with Computational Thinking

Computation

- A computer does 2 things
  - Performs instructions
  - Remembers their results

- Historically computation speed was limited by the human brain and the ability to record results by the human hand. But modern computers relieved us from those constraints.

The very basics

- Letters -> words -> sentences -> …
- Computers:
  - On – Off
  - Yes – No
  - True – False
- 0 - 1

How many options?

0000	0100	1000	1100
0001	0101	1001	1101
0010	0110	1010	1110
0011	0111	1011	1111

Mechanical Procedure

- Computers execute **mechanical procedures** -- procedures that can be followed without any thought.
- We need to give them unambiguous instructions such that when followed step by step the execution will finish and yield a result.
- An **algorithm** is a mechanical procedure that is guaranteed to eventually finish.

Procedure Example:

- …Combine the flour, sugar, yeast and salt in a mixing bowl.
- Start the mixer.
- Add water and 2 tablespoons of oil.
- Beat until the dough forms into a ball.
- If the dough is too sticky, add additional flour and beat.
- If the dough is too dry, adda ddtional water and beat.
- Otherwise, stop and kneed.

Describing what to do

If I was a robot, how would you describe/direct me on how to exit from the class…?

Example

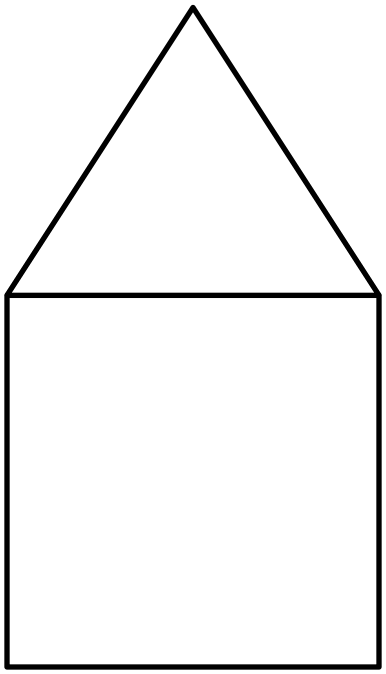
- I can execute the instructions that I understand. I have memory to remember things. Suppose I know how to do arithmetic. Give me a sequence of instructions to count the number of students present in the room.
  - Multiple ways to do this.
  - How can you express the instructions unambiguously?
  - How can you compare alternative procedures in terms of efficiency -- time it takes to get the end result.
- What if multiple people can do the counting in parallel?

One Algorithm

1. Take a card
2. Write the number 1 and stand up
3. Pair with a student who has a card in hand
4. Sum the numbers on your cards
5. Shorter student: cross out number on card, write new sum.
6. Taller student sits, the other goes back to step 3.

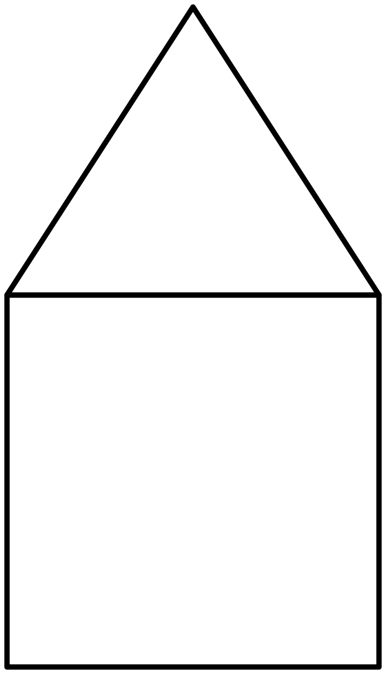
Primitives and Abstraction

- Suppose I know how to draw lines along a given direction and turn a given number of degrees as I draw. Can you give me instructions to draw this house?



Abstraction

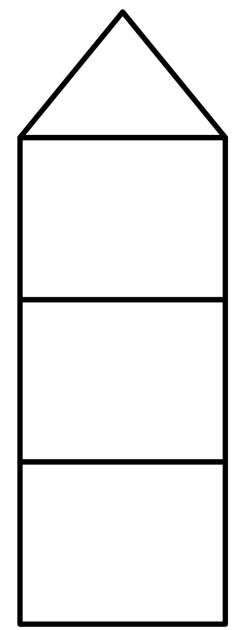
- Suppose I already know how to draw a triangle and a square. Can you give me instructions to draw this house?

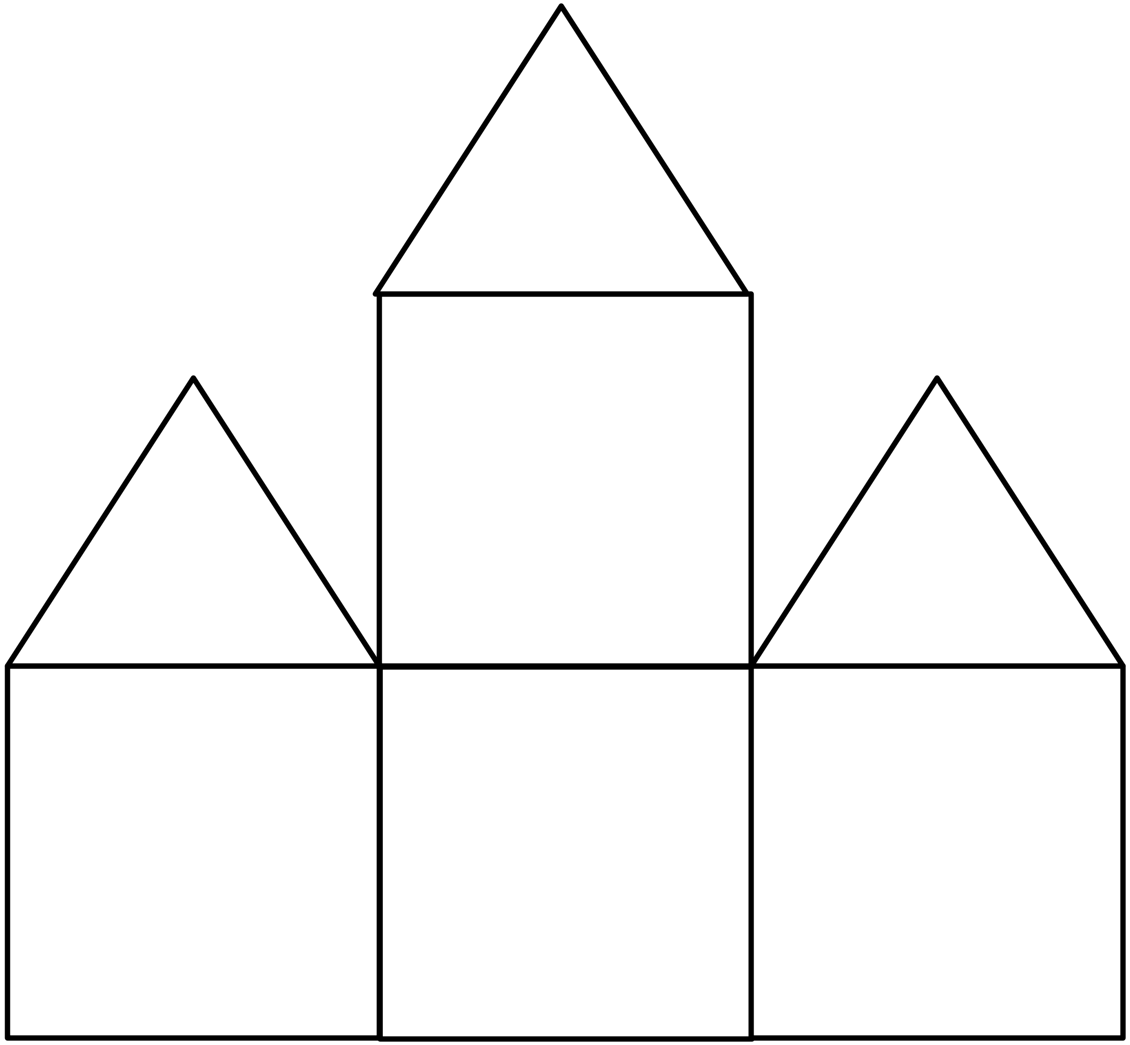


Abstraction: now we can think in terms of triangles and squares instead of lines

Reusability

How to draw this?





We can even think in terms of houses. What is the advantage of using abstraction like this?

Managing complexity, reusability …

General Purpose Computing

- We can design machines for specific computing tasks (averages, sums)
- Many earlier machines were fixed program computers
- Modern computers store sequences of instructions and execute them

Programs to Describe Mechanical Procedures

- What if we have millions of steps to specify for a computer?

We typically use higher-level programming languages to describe computations

Online Communication

- Email addresses are on the website (use *[15-110]* in the subject), **but** we prefer **Piazza. Why?**

- Do not copy your source code in your posting.
- Help one another, but do not provide direct solutions/assignments.

Today's Lab and First PA

- Getting set up and using the lab
- Programming with Blockly (code.org)

First Assignments

- Find and bookmark the course web page: www.cmu.edu/canvas
- Review the syllabus and schedule on that page.
- Complete the OLI module on Academic Integrity. Read, sign and return the Academic Integrity Form to your TA in the lab on Thursday.

Reading

- Blown to Bits Chapter 1 “Digital Explosion”.
- Begin reading ‘‘What is Code’’

Next Lecture

- A brief history of computing
- Programming with Python
- Wednesday: No class!