Engineering Summer Academy At Penn—Computer Science (2015)

About

Computer science, the study of computation, influences every aspect of our lives today in the information age. As such, it is an absolutely massive field spanning hardware, software, theory, and design, and it is growing every day. What is computer science? What are the core elements of computer science? And what are the essential lessons about computing that every one should know?

The computer science course of the Engineering Summer Academy at Penn (ESAP, formerly SAAST) is a three week crash course in the fundamentals of computer science. In this course, we will learn about the core skill shared by all computer scientists, *computation thinking* and how to apply it towards *computer programming*, the primary way that we harness the power of computation. Along the way, we will survey the vast field of computer science and learn what it is like to pursue computer science as a profession. In addition to this, we will also explore how computation affects our lives and what what we need to understand about computing in order to be well-informed citizens of the information age.

Learning Goals

Even though ESAP lasts for only three weeks, the breadth of the course is comparable to Penn's own introductory programming course, CIS 110. In particular, you will learn how to *program in the small*, *i.e.*, write small programs to solve targeted tasks. In terms of programming, after this course, you will be able to:

- Use *functions* to decompose problems into manageable chunks and express solutions to problems in a clear, readable manner.
- Use *recursion* to capture the special case where a problem decomposes to a smaller version of itself.
- Predict the behavior of computer programs with a clear *mental model of computation*.
- Perform actual computation using mathematical *expressions*.
- Model simple problems using *primitive data types*.
- Build up more complicated models of problems using *algebraic data types*.

However, while programming is the primary activity of the course, it is not its sole learning goal. In addition to programming, you will also be able to:

- Articulate *what* computer science is and its role in the information age.
- Understand what a computer scientist does and what a typical computer scientist's career path looks like.
- Assess whether computer science is a field of study that you would like to pursue in college.

Finally, as a college preparatory course, we also explicitly teach several ``meta-skills'' in this course—skills that will help you be a better college student. After this course, you will be set on the road to:

- Work effectively with a group to promote self-learning.
- Learn a new skill alone more effectively by understanding the value of targeted, focused practice.

Technology

No prior programming experience is required for this course. To enforce this point, we will be using a programming language you may have not heard of: F#. F# is a *functional programming language* that will allow us to focus on the fundamentals of computational thinking and program design. However, even if you have prior programming experience, the course staff will still have plenty for you to do and learn!

No equipment is necessary as our lab is stocked with a number of Linux machines for you to work on. However, if you have brought a laptop then we will help you set it up so that you can develop programs on it.

Format

To take advantage of the enormous amounts of hours that you will spend in the lab, we have structured the course in a collaborative workshop style. Rather than listen to lectures, you will be constantly exploring, experimenting with, and solving problems to help hone your computational thinking skills. For example, when we are together as a whole class, we will frequently all work together and collaborate on a single problem. Class participation is greatly encouraged!

For the homeworks, you will work with a partner in a style called *pair programming*. As alluded to in the learning goals above, we stress that working in a group, when done correctly, helps everyone in the group learn, irrespective of differences in their skill

level. We will devote a significant amount of time in class to help you learn how to work with a partner in a constructive manner.

Schedule

The typical structure of each day of class is:

- 9:00 AM-11:30 AM: Session (A)
- 11:30 PM-1:00 PM: Lunch
- 1:00 PM-3:00 PM: Session (B)
- 3:00 PM-3:30 PM: Break
- 3:30 PM-5:00 PM: Session (C)

Sessions (A) and (B) will typically present new content, and session (C) will be reserved to work on the assigned homework for the day.

(Note: the class schedule is tentative! The day-to-day content will vary based off of class need and interest.)

Week 1: Decomposition (Project #1: Computer Art)

- **7/6**: Day 1—Introduction to Computer Science (Welcome Assembly: 9-10:30 @ Cohen G17)
  - A: What is Computer Science?
  - B/C: Installing F#, Navigating the Command-line, and Compiling Your First Program
  - Labs: None
- **7/7**: Day 2—Functions, ASCII Art (ISSS Orientation for Int'l Students: 11-12)
  - A: Function Application
  - B/C: Function Definition and Problem Decomposition
  - Labs: SharpBots (submit via Canvas), The Song of Love
- **7/8**: Day 3—Expressions
  - A: Primitives, Values, and the Substitutive Model of Computation
  - B: Function Arguments, Return Values, and Conditionals
  - C: Recursion
  - Labs: Function Practice
- **7/9**: Day 4—Recursion
  - A/B/C: Recursion Practice!
  - Lab: Rocket Ship
- **7/10**: Day 5—Computer Graphics
  - A: Using Objects and Forms
  - B: Recursive Graphics
  - C: (Work Time)
  - Labs: Purple Boxes (picture only), Ehrenstein Circles, Sierpiski Triangles, Computer Artists

Week 2: Data (Project #2: Mystery Hunt)

- **7/13**: Day 6—Aggregate Data and Higher-order Functions
  - A: Lists and List Processing
  - B/C: Higher-Order Functions
  - Labs: (Assert.fs), List Practice,
- **7/14**: Day 7—Imperative Computation and Program Design
  - A/B: Algebraic Datatypes and File IO Basics
  - C: Work time
  - Labs: Computing Human Evolution
- **7/15**: Day 8—Project #2 Work Time (Master Lecture—Mark Yim: 11:30-12:30 @ Cohen G17)
  - A: Processing Data
  - B: Maps (1:30 PM start)
  - C: Work time
  - Labs: Text Stats
- **7/16**: Day 9—Field Trip to NY
  - 9:30 AM @ 33rd and Walnut St.
- **7/17**: Day 10—Project #2 Work Time
  - Labs: Speed Reader, …

Week 3: Applications (Project #3: Microgames)

- **7/19**: Day 11—Project #2 Work Time
- **7/20**: Day 12—Microgame Introduction
  - MicrogamesTester: Windows, OSX
  - StavVsPepe-v1.zip

    - StavVsPepe-v2.fs
    - StavVsPepe-v3.fs
    - Project Submission—Read the instructions!
- **7/21**: Day 13—Topics in Computer Science #2, Project Work Time!
- **7/22**: Day 14—Topics in Computer Science #3, Project Work Time!
- **7/23**: Day 15—The End of the Line

Evaluation

Evaluation is last because it is the least important part of the course! The bottom line is this: if you do all the work that we set forth (and there's a lot of it), you will get a good grade in the course. So relax, and work to learn and have fun rather than chase the grade!

For each lab and project, we will assign three points:

1. *Style*: is your code readable and solve the problem in an elegant manner?
2. *Correctness*: does your code accomplish its intended purpose
3. *Coorperation* did you work together with your partner to solve the problems in the lab?

You should first check with a TA for the first point, *style*, before submitting the lab. The final two points will be assessed by the TAs after you submit, and you will be notified if you need to change your submission.

To get an A in this class, you need to get 100% in this course, *i.e.*, three points on everything. There are no deadlines in this class (other than keeping up with the daily work), and we will inform you of all your mistakes, so as long as you put in the effort to do your best work, you will get an A!