ESAP Computer Science (16su): Home

ESAP Computer Science

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) 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.

People

- Instructor: Peter-Michael Osera, osera@cs.grinnell.edu
- Course Staff: Claire Frankel, AJ Hallac, Allen He, Emmett Neyman, Eddie Smith, and Alice Zhou

Schedule

The class meets each day in Moore 207. 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. Session (C) will alternate between content presentations and time to work on the assigned labs 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

- 7/4—No class (4th of July)
- 7/5—Welcome Assembly (Cohen Hall G17), 9:00–10:30 AM
  - Groups (7-05)
  - A: Introduction to Computer Science
  - B: Setting Up Your Environment
  - C: Python Bots
- 7/6—ISSS Orientation (Towne, Raisler Lounge)
  - Groups (7-06)
  - A: Functional Decomposition
  - B: Primitives, Values, and the Substitutive Model of Computation
  - C: *(Work time)*
- 7/7
  - Groups (7-07)
  - A: Function Arguments and Return Values
  - B: Using Objects by Drawing
  - C: (Work time)
- 7/8
  - Groups
  - A: Recursion
  - B: *Project 1: Computer Art*
  - C: *(Work time)*

Week 2: Data

- 7/11
  - Groups
  - A: Manipulating Strings
  - B: Sequential Structures: Lists
  - C: List Transformations

- 7/12—Master Lecture (Stiteler Hall B6), 11:00–12:00 PM
  - Groups
  - A: File Input
  - B: File Output
  - C: *(Work Time)*
- 7/13
  - Groups
  - A: Mapping Structures: Dictionaries
  - B and C: *Project 2: It's a Mystery…*
- 7/14—Field Trip (tentative)
- 7/15
  - A–C: *(Project Work Time)*

Week 3: Applications

- 7/18
  - Groups
  - A: Case Study: Video Game Architecture
  - B: Introduction to PyGame
  - C: *Project 3: Game Jam*
- 7/19—Admissions Workshop (Cohen Hall G17)
  - A: Case Study: The Internet, Security, and Privacy
  - B–C: *(Project Work Time)*
- 7/20
  - A: Case Study: Ethics in Computing
  - B–C: *(Project Work Time)*
- 7/21
  - A: Case Study: The Mathematics of Computation
  - B–C: *(Project Work Time)*
- 7/22—Final Day of Class

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 primitive computation using mathematical expressions.
- Model simple problems using primitive data types.
- Build up more complicated models of problems using basic aggregate data types (lists and records).

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.

Group Work

All of our work in this course will be conducted in groups, usually pairs randomly chosen for each day. While you will likely work slower with a partner than alone, you will (1) have a person to bounce ideas off of when you get stuck and (2) *write higher quality code that is more correct* than if you had written it alone. You are required to complete all labs and projects with your assigned group, *no exceptions*. If you

are having difficulties working with your group, please see a staff member to discuss the matter further.

Balancing Self-Learning and Getting Help

When learning how to program, you will inevitably get stuck, whether it is a nasty bug or you aren't sure how to put together the concepts discussed in course together. An important skill to learn in this course is *perseverance*. In this context, perseverance is the ability to use what you currently know to systematically diagnose your problem and resolve the issue on your own. It may feel like you are wasting time going through this process, but in reality, these moments are the greatest learning opportunities in the course.

Of course, when you are starting out on your journey in computer science, you may not have the knowledge to persevere through a bug, and so it is natural to seek help. However, we want to ensure that every such question you have is a learning opportunity, so we require that if you do need help that you *only consult your group or a course staff member*. In particular, please do not discuss particulars of a lab or project solution with other groups as others may simply give you the solution to your problem rather than help you discover the solution for yourself.

Evaluation and Grades

This course features many labs and projects designed to help you become proficient at programming. In order to facilitate the course's pace, we'll employ a lightweight method for evaluating your work. After you have completed a lab or a project, you should hail down a member of the course staff to review your work. *All members of your group must be present for the evaluation, no exceptions!* The evaluation process should take approximately 5 minutes:

- First, you will demonstrate your work, *e.g.*, by running your program on the test suite for the lab.
- Next, you will briefly walk through your work with the staff member, explaining its design.

Afterwards, the staff member will evaluate the correctness and design of the work and grade it on the following three point scale.

1. *Not yet functional*: fails major tests or otherwise has very obvious bugs.
2. *Functional* but needs revisions: generally correct with minor bugs and/or stylistic problems with

program design.
3. *Complete*: correct both in terms of correctness and style.

In the first two cases, you should revise your work and re-present it to the same staff member to be reviewed again. In the third case, you are done—congratulations! Note that there are no strict deadlines in this course, but with the sheer amount of material we cover, you ought to finish the *labs for any given day by the morning of the next class day*. Projects will frequently take you through the weekend to complete and should be checked that weekend or at the beginning of the class the following week.

Your grade in this course is determined by the following scale:

- **A**: all labs and projects are marked *complete*.
- **B**: at most three labs/projects are marked *functional*; the remainder are *complete*.
- **C**: more than three labs/projects are marked *functional*; the remainder are *complete*.
- **D**: some labs are marked *not yet functional*; the remainder are *functional* or *complete*.
- **F**: some labs are not evaluated.

There are no intermediate grades, *e.g.*, minus or plus grades. The criteria is purposefully binary because I expect everyone to complete every lab and project.
