SAAST Computer Science

Overview

July 2013

This course is an introduction to computer science by way of learning the the fundamentals of computer programming in Python. The heart of computer science lies in algorithmic thinking or how to "think like a computer". While algorithmic thinking and computer programming are not the same thing, they are closely related as the famous computer scientist Donald Knuth wrote:

It has often been said that a person does not really understand something until after teaching it to someone else. Actually a person does not really understand something until after teaching it to a computer, i.e., expressing it as an algorithm.

Thus, this course develops your skills in the three key areas of algorithmic thinking — precision, decomposition, and abstraction — using computer programs as a way of making these concepts concrete. So in addition to learning how to think like a computer, you will also gain practical experience with computer programming.

SAAST Computer Science is a college-caliber introductory course in computer science crammed into three weeks. No prior programming experience is assumed, but we will be moving fast, so your commitment and hard-work is required!

Lectures and Labs

The typical class day runs from 9 AM to 5 PM as follows:

9:00-11:30 AM - Session #1
11:30-1:00 PM - Lunch
1:00-3:00 PM - Session #2
3:00-3:30 PM - Break
3:30-5:00 PM - Lab/Work time

All of the sessions will be held in our Linux computer lab, Moore 207. They will consist of lecture, group work, as well as time to work on lab assignments and projects. All of these sessions are required! If you need to miss a session, please let the instructor know as soon as possible so accommodations can be made.

Important The class schedule is fluid and exact times will vary day-to-day based off of actual session lengths, class progress, and SAAST-wide activities. Changes to this schedule will be announced in-class as well as posted to the class calendar.

Lab Work, Projects, and Grades

As a three-week intensive summer course, we have the special opportunity to work closely with you on the material. Our goal is for you to be able to complete all of the lab assignments and projects. Throughout this process you will be learning programming the only way we reliably know how to teach it: practice, practice, practice!

This is a tall order given that we are covering a semester's worth of material in three weeks. To help you along this process, you will work in groups of 2 on the lab work and projects. This practice is called Pair programming where two developers work on the same computer. One person writes the code while the other observers and checks for mistakes. This is helpful for us in two ways:

It is a well-known phenomenon that typing at a keyboard makes you 100% less smart. Your partner is there to help you from making small, hard-to-catch mistakes that will give you headaches later.

Programming is a strenuous mental exercise and as such, it is useful to have another person available to discuss potential solutions and ideas.

Pairs will be randomly assigned each day. You are expected to work in your pairs for the lab work that is assigned for that day.

Lab Work

Lab work consists of short assignments designed to give you practice with the programming and algorithmic thinking concepts presented in lecture as well as longer assignments that explore computer science topics as well as give you additional practice. Generally, the shorter lab assignments will be due at the end of a session and the longer lab assignments will be due the following day.

Projects

There are three week-long projects for the course, open-ended assignments that let you put together the concepts discussed in class in a larger context. You are expected to work on each project throughout the week. Each project will generally be due at the end of the week.

Grades

Because we emphasize that you will complete all of the lab assignments and projects in this course, our grading policy is straightforward:

Completing a lab assignment or project means submitting a program that gets full points for that assignment or project. This involves writing a program that produces the correct answer given a variety of inputs as well as a program that is stylistic correct, i.e., formatted correct as well as of .

If you complete all of the lab assignments and projects, you get an A, otherwise you get an F.

To alleviate this, there are no hard deadlines in this course. While there are a number of soft deadlines in the course (i.e., you should aim to complete the lab assignments the day they are assigned and the week's project by the end of the week), all work is finally due on the last day of class.

Don't use this as an excuse to slack off, though! There will be enough work constantly due that you will need to stick to our soft deadlines to stay afloat! It is your responsibility to keep up with the material and engage with your lab partners and instructional staff if you are falling behind.

Collaboration Policy

Most of the learning in computer programming comes not the final program but the journey it took to craft that program, bumps and all. Because of this, we would like you to limit your collaboration and questions about the lab work and projects to your assigned partner and the course staff. This way, we can ensure that you are getting the most out of the practice problems throughout the course.

Course Content

What follows is a rough outline of the topics that this course covers. Please be advised that this is not set in stone as we will explore a variety of additional topics (in particular, advanced topics) as time and the interest of the class dictates.

Programming in Python

Source files and interpreting computer programs.

The uses of Python in real-world computing.

Precision and Decomposition

Functions: definition and invocation.

Primitive datatypes, variables, and expressions.

Function mechanics: parameters, returns, and our mental model of computation.

Control flow: conditional, repetitive, and indefinite execution.

Recursive decomposition.

Abstraction

Handling user input/output.

Processing large sets of data with files.

Aggregate data structures: lists and dictionaries.

Introduction to object-oriented programming: classes and objects.

To motivate these core concepts, we will also explore at a high-level the wide variety of sub-fields that encompass computer science in our lab assignments. These fields include computer graphics, networks, algorithms and theory, cryptography and security, systems, and others.

Questions and Comments

We're here to answer your questions, help you learn, and evolve the course so that current and future students can learn why we love computer science so much! Feel free to talk to the instructor or your RTA if you have any questions, concerns, or comments about the class via email or in-person during class.