The Code.org CS Principles Curriculum Overview

Code.org has set out to write a comprehensive and rigorous Computer Science Principles curriculum designed specifically for the high school setting - one that is flexible and comfortable for both the “new-to-CS-Principles” educator and the most experienced teacher. The course is being developed to address the learning objectives of CS Principles Framework in its most recent form - published in February, 2014. The curriculum focuses on preparing students in the many skills they will need to succeed in the various “Performance Tasks” in the future AP exam, as defined by the College Board.

Throughout the development of this curriculum, writers have worked backward from the College Board’s curriculum framework, specifically the identified “learning objectives” and “essential knowledge” statements, grouping them into like-categories. As a result of this process, the curriculum follows a smooth sequence for teachers and students which spirals throughout the units to progressively build students’ understanding and skills.

Generally speaking, the lessons and activities employ inquiry- and exploration-based learning strategies, designed to empower students to explore deeply, creatively, and independently. Lessons are designed to allow teachers to access and build on their students’ experiences, whatever they are, to motivate deep and meaningful learning. To borrow an expression from the “Understanding by Design” model of curriculum development, our goal is not to cover the CSP content, but for students to uncover it as they move through the lessons. We believe that this approach to learning and teaching, along with the inherently interesting content explored in the CS Principles course will lead to a classroom experience that promotes student learning and appreciation for the field of computer science.

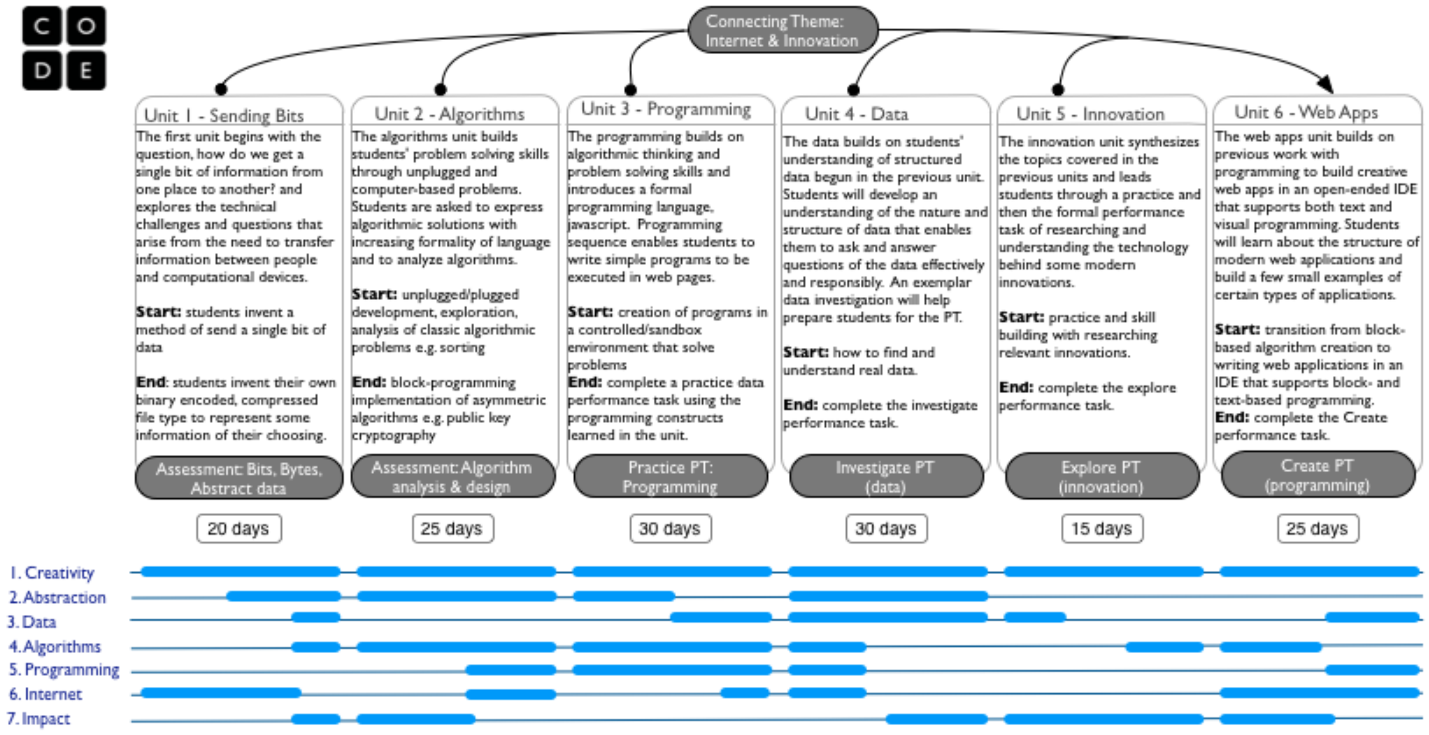
The course is designed for 150 “class meetings” of approximately 50 minutes. Keeping in mind that the most recent version of the CSP Framework has over 300 “essential knowledge” statements, it is likely that projects or activities started in class will have to be finished independently. However, the writing team has worked to make sure the pacing is flexible enough that teachers can adapt it to the needs of their students and their schools. For example, we have built in time for students to work on the Performance Tasks - student work with which the teacher is not allowed to help.

The following summary of the course and units of study is under development. Please see the timeline for content completion, professional development, and piloting phases to see what can be expected throughout the next year, leading up to the 2015-16 academic year. All teacher lesson plans, notes, instructional videos, assessments, student activities, handouts, and software under development by Code.org will come together as a complete collection of curriculum resources for teachers and schools. Our overall goal is to make the curriculum materials and daily activities accessible, free, and easy to use, so teachers can focus on their students' learning needs.

Course Overview

One thing that makes (AP) CS Principles truly different from AP CS-A, is the embedded inclusion of the Internet and how its development has affected both society and computation. The "Internet & Innovation" provide a narrative arc for the course - something that connects everything students learn. The course starts with learning about what is involved in sending a single bit of information from one place to another, and ends with students developing a small web application of their own design.

The diagram below shows unit start and end points, connection to the overarching narrative and, roughly speaking, where the big ideas of the course get the most explicit coverage.



Unit 1 - Sending Bits

The first unit begins with the question, how do we get a single bit of information from one place to another? and explores the technical challenges and questions that arise from the need to transfer information between people and computational devices.

Start: students invent a method of send a single bit of data

End: students invent their own binary encoded, compressed file type to represent some information of their choosing.

Unit 2 - Algorithms

The algorithms unit builds students' problem solving skills through unplugged and computer-based problems. Students are asked to express algorithmic solutions with increasing formality of language and to analyze algorithms.

Start: unplugged/plugged development, exploration, analysis of classic algorithmic problems e.g. sorting

End: block-programming implementation of asymmetric algorithms e.g. public key cryptography

Unit 3 - Programming

The programming builds on algorithmic thinking and problem solving skills and introduces a formal programming language, javascript. Programming sequence enables students to write simple programs to be executed in web pages.

Start: creation of programs in a controlled/sandbox environment that solve problems
End: complete a practice data performance task using the programming constructs learned in the unit.

Unit 4 - Data

The data builds on students' understanding of structured data begun in the previous unit. Students will develop an understanding of the nature and structure of data that enables them to ask and answer questions of the data effectively and responsibly. An exemplar data investigation will help prepare students for the PT.

Start: how to find and understand real data.

End: complete the investigate performance task.

Unit 5 - Innovation

The innovation unit synthesizes the topics covered in the previous units and leads students through a practice and then the formal performance task of researching and understanding the technology behind some modern innovations.

Start: practice and skill building with researching relevant innovations.

End: complete the explore performance task.

Unit 6 - Web Apps

The web apps unit builds on previous work with programming to build creative web apps in an open-ended IDE that supports both text and visual programming. Students will learn about the structure of modern web applications and build a few small examples of certain types of applications.

Start: transition from block-based algorithm creation to writing web applications in an IDE that supports block- and text-based programming.
End: complete the Create performance task.

Figure 1: Curriculum Arc [click to enlarge]

NOTE: As of this writing, the writing of the first three units of study are well underway. Pieces of these units will be selectively piloted by a few instructors before public release. The last three units - 4, 5 and 6 - have been conceived, but work on a draft will not be complete until Summer 2014. Please consult the timeline below to see the stages of development for the course, the pilot program for testing the course, and the associated teacher professional development plan.

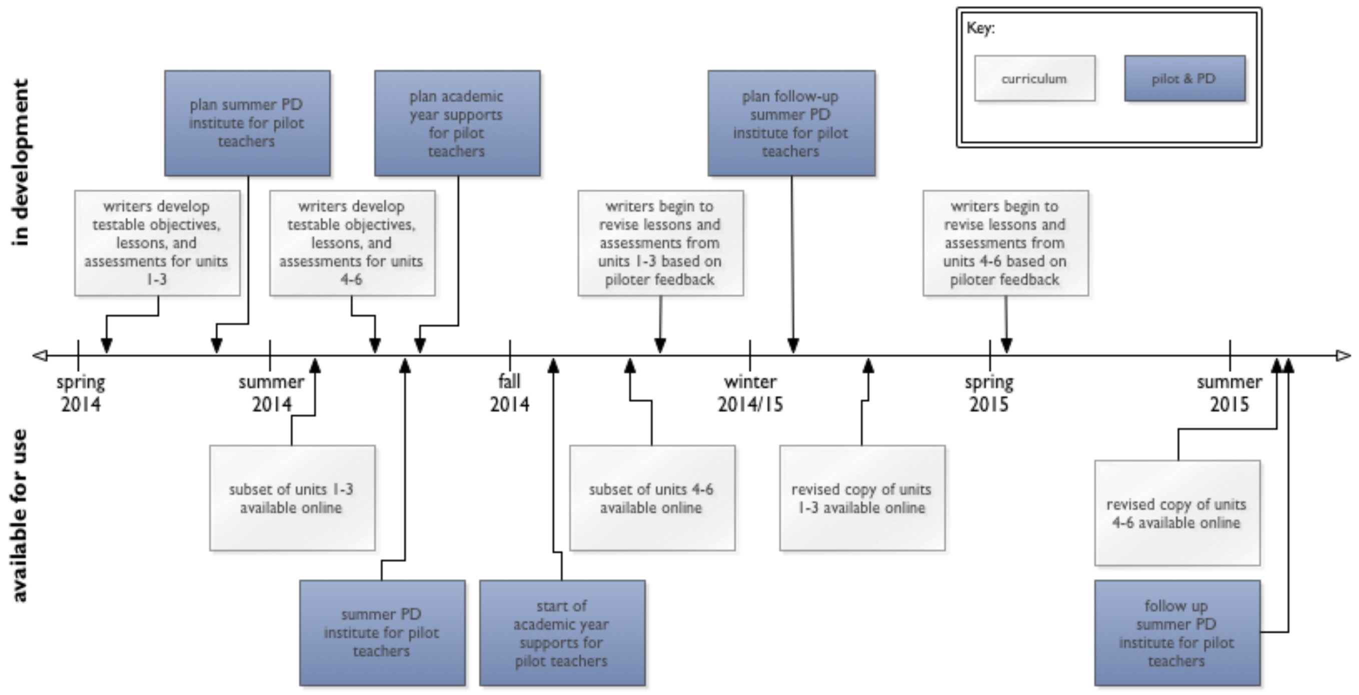

Key:
curriculum
pilot & PD
in development
available for use
plan summer PD institute for pilot teachers
plan academic year supports for pilot teachers
plan follow-up summer PD institute for pilot teachers
writers develop testable objectives, lessons, and assessments for units 1-3
writers develop testable objectives, lessons, and assessments for units 4-6
writers begin to revise lessons and assessments from units 1-3 based on piloter feedback
writers begin to revise lessons and assessments from units 4-6 based on piloter feedback
spring 2014
summer 2014
fall 2014
winter 2014/15
spring 2015
summer 2015
subset of units 1-3 available online
subset of units 4-6 available online
revised copy of units 1-3 available online
revised copy of units 4-6 available online
summer PD institute for pilot teachers
start of academic year supports for pilot teachers
follow up summer PD institute for pilot teachers

Figure 2: Development timeline [click to enlarge]

Curriculum Summary

Each of the planned units of study for the Code.org CSP course are described below. For each unit we describe the big ideas behind the unit and its connection to overarching course themes, as well as to other units of study within the course. The student activities are being designed with testable lesson objectives in mind. Along with each unit description, there is list of some of the testable objectives from which we are deriving lesson plans and activities ("Selected Student Learnings and Understandings"). These lesson objectives are not to be confused with the CSP Framework "Learning Objectives" (CSP LOs) but were created specifically for this course and map to the CSP LOs, although not always one-to-one. A comprehensive mapping of our lessons to the CSP LOs and Essential Knowledge Statements will be made available.

Unit 1: Sending Bits

The course begins with a consideration of what is involved in sending a single bit of information from one place to another. In the "sending bits lab" students work with a partner to devise and build their own bit-sending "machines." Complexity increases as students adapt their machines to handle multi-bit messages, and increasingly complex information. As the unit progresses, students learn more about the binary representation of information, and are asked to think more abstractly about how to encode complex information in bits, and develop algorithms for encoding and decoding streams of bits. Students are asked to build (and then compress) binary files that represent numbers, text, images, and sounds. The unit concludes with students inventing their own file type to encode some information they have chosen...such as a poem or selection of text.

Selected Student Learnings & Understandings

Student can...

- Collaborate with a partner to create a physical device to send a bit of information to a classmate.
- Hypothesize about and connect the concept of sending bits in digital networks.
- Describe the large-scale characteristics of the Internet such as hierarchy, redundancy, and fault-tolerance.
- Name different representations of bit. Define bit, byte, kilobyte, megabyte.
- Describe how multiple levels of abstraction are used in the encoding of information with bits (e.g. numbers, text, sound, images).
- Describe different file formats and any compression within them.
- View the hex representation of a file on a computer.
- Consider the problems of abstraction required in managing large numbers of bits.
- Explain the need for compression, and demonstrate understanding using a purpose-built computer-based tool (called a "widget").
- Describe an invented device that sends periodic data via the Internet, design a data encoding method for this device, write out a sample data file generated by this device, and consider how to compress large files of this type.

Unit 2: Algorithms

The Algorithms unit flows from the ideas in unit 1 related to the processes and protocols for interpreting binary information and the need for algorithms. Students will explore algorithms and the need for a structured approach to problem solving, iteratively increasing the formality of both the language students use to describe algorithms and they way they analyze them. The unit begins with a study of the classic problem of sorting information. In an “unplugged” card-sorting activity students are asked to develop an algorithm and express it as formally as they can. Then students examine their and other students’ sorting algorithms using an online tool that helps them understand the considerations required to answer the question “what is the best way to sort?” Finally students will express their algorithm in a more formal block-based language to see if works as they expect. This general pattern will be repeated for several other classic problems, including searching, problems on graphs and networks, basic encryption, and computationally “hard” problems. The study of problems in NP leads students to develop their own asymmetric encryption schemes and develop an understanding of how computationally hard problems bear on Internet security and privacy.

Selected Student Learnings/Understandings

Student can....

- Identify a real-world problem that requires an algorithmic solution.
- Identify "worst-case" scenarios and behaviors for a sorting algorithm.
- Design and implement a sorting algorithm using sorting widget.
- Connect similarities of algorithmic techniques between sorting, encryption and compression.
- Invent an encryption algorithm (using only text and or numbers) for sending a message to a friend.
- Design/draw/create a graph to represent some information, data, or system.
- Connect algorithms on graphs with algorithms used in Internet contexts, such as routing issues, DNS, etc.
- Describe concept of "computability."
- Argue for the "hardness" of a problem to compute.
- Connect NP-hard problems with security/encryption.
- Describe the situations and transactions that are needed for a public key exchange.

Unit 3: Programming

The programming unit takes the formality of algorithm expression one step further as students write programs in javascript. The levels of abstraction increase as students learn how a programming language can be used to control the computer. The IDE used in the unit will empower the learner to move between block and text representations of their programs. The topics of the programming unit, input/output, calculations with numbers, branching statements and boolean logic, iteration, procedural abstraction and processing data in simple linear structures are considered to be among the more “classic” computer programming topics. However, the context in which students are using these programming constructs will be in terms of the web and web application development. For example, the user input will come from a text box on a web page or on an image that the user uploads to the website. The output will be text shown in the browser or drawn on an HTML5 canvas. In the final project, students will use a timer to animate images stored in an array to make an animation.

Selected Student Learnings/Understandings

Student can...

- Implement an algorithm in a programming language that involves the use of iteration and boolean logic.
- Recognize the need to automate a process in the face of a large amount of data.
- Identify simple algorithms that, when combined, solve a complex problem.
- Recognize and program a solution that requires the use and manipulation of data in a list
- Reason about and justify the need for well-written code (not just functional code) and understand why well-written code is important for robustness and usability.
- Use appropriate tools and strategies for debugging and avoiding errors.

Under development

The following units have been conceptualized to fit the narrative arc and sequence of the course. The learning objectives and activities are currently under development and are planned to be available for piloting during the summer of 2014. Please consult the timeline above for more detail about development, piloting, and release dates.

Unit 4: Data

The data unit builds and extends the students' understandings of structured and abstract data introduced in the programming unit. The data unit builds students' skills for the "Investigate—Bits to Information to Knowledge" Performance Task. Most of the unit focuses on gaining an understanding of the nature and structure of complex data sets, how to leverage that understanding to ask interesting, knowledge-seeking questions of the data, and acquiring the skills to answer those questions with one or more data manipulation tools. Optimally, the the data sets we use as exemplars will tie thematically to the Internet in a more integral way that merely recognizing that the data we get "comes from the Internet." We will also use data sets that foreshadow innovative thinking about ways in which data can be used, in preparation for the following unit on innovation.

Unit 5: Innovation

In the Innovation unit, students will explore many of the innovations that have occurred due to the emergence and applications of big data. Activities in this unit allow students to be much more independent as they conduct research on a topic of innovation. At this point in the course, students should be well prepared to understand technological innovation, but may need some practice in communicating their knowledge in order to be prepared for the Innovations Performance Task. The instructional activity in this unit includes practicing for the Performance Task in which students select and research an innovation, locate reliable information, develop an understanding of the technical/algorithmic components, and create an artifact to demonstrate their understanding. Through this practice, students will gain the skill necessary to complete the Performance Task on their own. Time is included in the plans for students to work independently on their research and complete the Performance Task.

Unit 6: Web Applications

The course concludes with students developing and building web applications of their own design. Our definition of "web application" is a program that lives, and can be interacted with, on the web. The programs will vary from being entirely client-side scripted, or may involve information/data retrieval and storage using remote sources. In this unit, students will use an IDE that makes building web applications easy and intuitive; the coding will flow naturally from the tools used throughout the course. Students will examine several different types of web applications including: simple web utilities with dynamic content, game-like applications, and applications that require management of users' data. The skills required for the "Create" Performance Task include students working both collaboratively and independently in the development of a program. The hope is that the student research in the previous unit on innovation will inspire ideas for web applications to solve a real-world problems. The unit concludes with significant time for students to work on their Performance Task submissions.