rev. Feb. 2015

CS Principles: **Decoded**

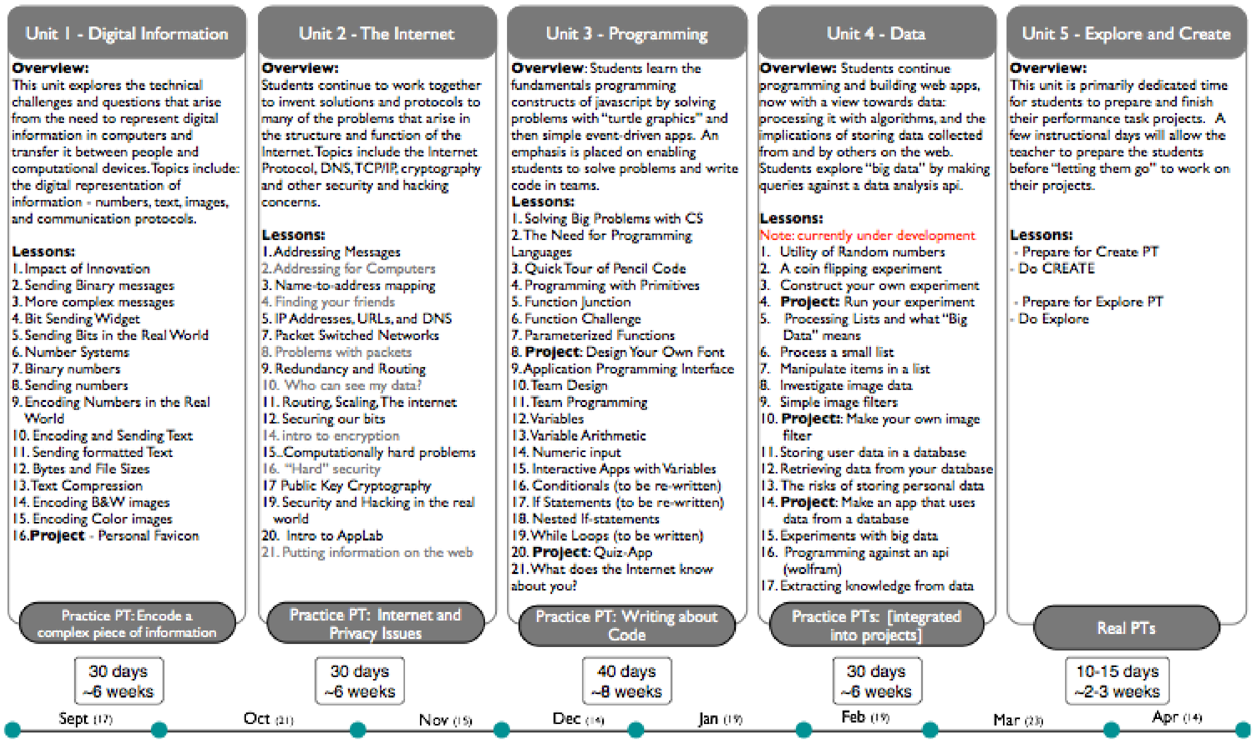
a curriculum from Code.org

Description

Computer Science Principles (CSP) is a course designed around the AP Computer Science Principles Framework. It becomes an official AP® course in the 2016-17 school year.

Code.org's CS Principles course is designed to be a full-year, rigorous, but entry-level course for high school students. The curriculum is also written to support teachers new to the discipline with inquiry-based activities, videos, assessment support, and computing tools that have built-in tutorials and student pacing guides.

Below is a snapshot of the course. The course contains 4 core units of study, roughly equal in length, each containing about 20 lessons. Most lessons take one or two days to complete in class, assuming a typical 50-minute period. [expand image]



Unit 1 - Digital Information

Overview:
This unit explores the technical challenges and questions that arise from the need to represent digital information in computers and transfer it between people and computational devices. Topics include: the digital representation of information - numbers, text, images, and communication protocols.

Lessons:
1. Impact of Innovation
2. Sending Binary messages
3. More complex messages
4. Bit Sending Widget
5. Sending Bits in the Real World
6. Number Systems
7. Binary numbers
8. Sending numbers
9. Encoding Numbers in the Real World
10. Encoding and Sending Text
11. Sending formatted Text
12. Bytes and File Sizes
13. Text Compression
14. Encoding B&W images
15. Encoding Color images
16. **Project** - Personal Favicon

Practice PT: Encode a complex piece of information

30 days
~6 weeks

Unit 2 - The Internet

Overview:
Students continue to work together to invent solutions and protocols to many of the problems that arise in the structure and function of the Internet. Topics include the Internet Protocol, DNS, TCP/IP, cryptography and other security and hacking concerns.

Lessons:
1. Addressing Messages
2. Addressing for Computers
3. Name-to-address mapping
4. Finding your friends
5. IP Addresses, URLs, and DNS
7. Packet Switched Networks
8. Problems with packets
9. Redundancy and Routing
10. Who can see my data?
11. Routing, Scaling, The internet
12. Securing our bits
14. intro to encryption
15..Computationally hard problems
16. "Hard" security
17 Public Key Cryptography
19. Security and Hacking in the real world
20. Intro to AppLab
21. Putting information on the web

Practice PT: Internet and Privacy Issues

30 days
~6 weeks

Unit 3 - Programming

Overview: Students learn the fundamentals programming constructs of javascript by solving problems with "turtle graphics" and then simple event-driven apps. An emphasis is placed on enabling students to solve problems and write code in teams.

Lessons:
1. Solving Big Problems with CS
2. The Need for Programming Languages
3. Quick Tour of Pencil Code
4. Programming with Primitives
5. Function Junction
6. Function Challenge
7. Parameterized Functions
8. **Project**: Design Your Own Font
9. Application Programming Interface
10. Team Design
11. Team Programming
12. Variables
13. Variable Arithmetic
14. Numeric input
15. Interactive Apps with Variables
16. Conditionals (to be re-written)
17. If Statements (to be re-written)
18. Nested If-statements
19. While Loops (to be written)
20. **Project**: Quiz-App
21. What does the Internet know about you?

Practice PT: Writing about Code

40 days
~8 weeks

Unit 4 - Data

Overview: Students continue programming and building web apps, now with a view towards data: processing it with algorithms, and the implications of storing data collected from and by others on the web. Students explore "big data" by making queries against a data analysis api.

Lessons:
Note: currently under development
1. Utility of Random numbers
2. A coin flipping experiment
3. Construct your own experiment
4. **Project:** Run your experiment
5. Processing Lists and what "Big Data" means
6. Process a small list
7. Manipulate items in a list
8. Investigate image data
9. Simple image filters
10. **Project:** Make your own image filter
11. Storing user data in a database
12. Retrieving data from your database
13. The risks of storing personal data
14. **Project**: Make an app that uses data from a database
15. Experiments with big data
16. Programming against an api (wolfram)
17. Extracting knowledge from data

Practice PTs: [integrated into projects]

30 days
~6 weeks

Unit 5 - Explore and Create

Overview:
This unit is primarily dedicated time for students to prepare and finish their performance task projects. A few instructional days will allow the teacher to prepare the students before "letting them go" to work on their projects.

Lessons:
- Prepare for Create PT
- Do CREATE

- Prepare for Explore PT
- Do Explore

Real PTs

10-15 days
~2-3 weeks

Sept (17) Oct (21) Nov (15) Dec (14) Jan (19) Feb (19) Mar (23) Apr (14)

Each unit concludes with a practice performance task (PT) that mimics the assessment style of the tasks students are required to complete on their own for submission to the College Board as part of the official assessment for the course.

Unit 5 is devoted almost entirely to time for students to work on and finish their PT projects for submission to the College Board. A timeline showing a typical Sept-May school year is shown at the bottom. The AP Exam and PT submission deadline will typically be in the first week of May.

Who should take this course

There are no formal prerequisites for this course, though the College Board recommends that students have taken at least Algebra 1. The course also contains a significant amount of expository writing (as well as writing computer code, of course). For students wishing to complete the requirements of the AP Exam and Performance Tasks, we recommend they be in 10th grade or above.

Who should teach this course

The curriculum is designed so that a teacher who is new to teaching this material has adequate support and preparation - especially for those who go through our professional development program. A teacher who is motivated to teach a course like this and who has even limited technical or formal computer science experience should be able to teach the course. We do strongly recommend that the teacher have a decent level of comfort using computers (using the web, email, downloading and saving files, basic troubleshooting, etc.) and at least some experience with computer programming obtained through self-instruction, an online course, or a course taken in college.

Technical Requirements

The course requires a 1:1 computer lab or setup such that each student in the class has ad hoc access to an internet-connected computer every day in class. Each computer must have a modern web browser installed. All of the course tools and resources (lesson plans, teacher dashboard, videos, student tools, programming environment, etc.) are online and accessed through a web browser. While the course features many “unplugged” activities away from the computer, and group work both away and at the the computer, ad hoc access to a computer is essential for every student. It is not required that students have access to computers at home, though obviously that greatly enables learning to continue outside the walls of the classroom.

Resources

The Code.org CSP curriculum includes almost all resources teachers need to teach the course including:

- Instructional guides for every lesson
- Formative and summative assessments, exemplars,and rubrics, and teacher dashboard
- Student videos - including tutorials, instructional and inspirational videos
- Teacher videos - including lesson supports, and pedagogical tips and hints
- Widgets and simulators for exploring individual computing concepts
- *AppLab* - Code.org’s javascript programming environment for making apps

Curriculum Overview

The Internet and innovation provides a narrative arc for the course, a thread connecting all of the units. 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 small applications of their own design that live on the web.

Unit Structure

While the layout of units appears to be modular, the units of study are intended to be taught in the order suggested, and each not only builds students skills and knowledge through the course, but lessons in units frequently refer back to work or lessons from previous units.

Each unit typically ends with a practice Performance Task (PT) in which students do work similar to the College Board PTs. These practice PTs are smaller in scope than the real PTs and are intended to focus on particular elements or skills required to complete the PTs at the end of the course.

Lesson Structure and Philosophy

Lessons are designed to be student-centered and to engage students with inquiry-based and concept-discovery activities. The course does not require the new-to-computing teacher to lecture or present on computer science topics if they do not want to. Direct instruction is built into our tools and videos. The teacher plays a large role making choices and ensuring that the activities, inquiry, and reflection are engaging and appropropriate for their students, as well as assessing student learning.

Most lessons follow a basic routine:

- A warm-up activity to activate prior knowledge and present a thought-provoking problem
- An activity that varies but is typically one of:
    - Unplugged concept invention, and problem solving scenarios
    - Creating digital artifacts (including programming)
    - Research / reflection / presentation
- Activities are also frequently done in pairs or small groups
- A wrap-up activity or reflection

Computer Science Principles

rev. Feb. 2015

Coverage of the CS Principles Framework

The CS Principles framework is not intended to be taught in any particular order. The “Learning Objectives” (LOs) are typically associated with 10 or more “Essential Knowledge” (EK) statements, and are themselves grouped under 7 Big Ideas. Each LO and its associated EKs typically overlap, intersect, or are closely related to other LOs and Eks. For example, the LOs about *programming* also refer to *abstraction*.

Below is a table showing which units contain “coverage” of a learning objective. We define “coverage” to mean that some number of EKs associated with an LO are addressed in the course either explicitly or implicitly through practice or performance. (For example, the LOs under *Creativity* are about student action or behavior more than facts about computing that need to be learned.)

Units				CS Principles Framework
				Learning Objectives
1	2	3	4	**Creativity**
●	●	●	●	1.1.1 Apply a creative development process when creating computational artifacts. [P2]
●	●	●	●	1.2.1 Create a computational artifact for creative expression. [P2]
●	●	●	●	1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. [P2]
●		●		1.2.3 Create a new computational artifact by combining or modifying existing artifacts. [P2]
●	●	●	●	1.2.4 Collaborate in the creation of computational artifacts. [P6]
●	●	●	●	1.2.5 Analyze the correctness, usability, functionality, and suitability of computational artifacts. [P4]
●	●	●	●	1.3.1 Use computing tools and techniques for creative expression. [P2]
				Abstraction
●	●			2.1.1 Describe the variety of abstractions used to represent data. [P3]
●	●			2.1.2 Explain how binary sequences are used to represent digital data. [P5]
●		●		2.2.1 Develop an abstraction when writing a program or creating other computational artifacts. [P2]
		●		2.2.2 Use multiple levels of abstraction to write programs. [P3]
		●	●	2.2.3 Identify multiple levels of abstractions that are used when writing programs. [P3]
	●		●	2.3.1 Use models and simulations to represent phenomena. [P3]
			●	2.3.2 Use models and simulations to formulate, refine, and test hypotheses. [P3]
				Data
●	●	●	●	3.1.1 Use computers to process information, find patterns, and test hypotheses about digitally processed information to gain insight and knowledge. [P4]
	●		●	3.1.2 Collaborate when processing information to gain insight and knowledge. [P6]
		●	●	3.1.3 Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notations, and precise language. [P5]
		●	●	3.2.1 Extract information from data to discover and explain connections, patterns, or trends. [P1]
	●	●	●	3.3.1 Analyze how data representation, storage, security, and transmission of data involve computational manipulation of information. [P4]
				Algorithms
		●	●	4.1.1 Develop an algorithm for implementation in a program. [P2]
	●	●	●	4.1.2 Express an algorithm in a language. [P5]
	●		●	4.2.1 Explain the difference between algorithms that run in a reasonable time and those that do not run in a reasonable time. [P1]
	●			4.2.2 Explain the difference between solvable and unsolvable problems in computer science. [P1]
	●			4.2.3 Explain the existence of undecidable problems in computer science. [P1]
	●		●	4.2.4 Evaluate algorithms analytically and empirically for efficiency, correctness, and clarity. [P4]
				Programming
		●	●	5.1.1 Develop a program for creative expression, to satisfy personal curiosity, or to create new knowledge. [P2]
		●	●	5.1.2 Develop a correct program to solve problems. [P2]
		●	●	5.1.3 Collaborate to develop a program. [P6]
	●	●	●	5.2.1 Explain how programs implement algorithms. [P3]
		●	●	5.3.1 Use abstraction to manage complexity in programs. [P3]
	●	●	●	5.4.1 Evaluate the correctness of a program. [P4]
		●	●	5.5.1 Employ appropriate mathematical and logical concepts in programming. [P1]
				Internet
	●			6.1.1 Explain the abstractions in the Internet and how the Internet functions. [P3]
	●			6.2.1 Explain characteristics of the Internet and the systems built on it. [P5]
	●			6.2.2 Explain how the characteristics of the Internet influence the systems built on it. [P4]
	●			6.3.1 Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it. [P1]
				Global Impacts
●	●			7.1.1 Explain how computing innovations affect communication, interaction, and cognition. [P4]
		●		7.1.2 Explain how people participate in a problem solving process that scales. [P4]
●	●	●	●	7.2.1 Explain how computing has impacted innovations in other fields. [P1]
	●		●	7.3.1 Analyze the beneficial and harmful effects of computing. [P4]
●	●	●	●	7.4.1 Explain the connections between computing and economic, social, and cultural contexts. [P1]

Unit Overviews

What follows are more in-depth descriptions of each unit of study which explain the topics covered and what students will be doing.

Unit 1: The Digital Representation of Information

This unit sets the foundations for thinking about the digital (binary) representation of information and how that affects the world we live in. This unit explores the technical challenges and questions that arise from the need to represent digital information in computers and transfer it between people and computational devices. Topics include: the digital representation of information - numbers, text, images, and communication protocols.

The unit 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.

The unit concludes with a practice Performance Task in which students invent their own file type/protocol for encoding a complex type of information that has some personal significance to them. A draft of the lessons and prototypes of tools for Unit 1 are available here: http://code.org/educate/csp/unit1

Lessons	**CSP Framework Learning Objectives Addressed** (addressed explicitly or implicitly through practice)
1. Impact of Innovation 2. Sending Binary messages 3. More complex messages 4. Bit Sending Widget 5. Sending Bits in the Real World 6. Number Systems 7. Binary numbers 8. Sending numbers 9. Encoding Numbers in the Real World 10. Encoding and Sending Text 11. Sending formatted Text 12. Bytes and File Sizes 13. Text Compression 14. Encoding B&W images 15. Encoding Color images 16.**Project** - Personal Favicon **17. Practice Performance Task: Encode a Complex Thing**	**Creativity** 1.1.1 Apply a creative development process when creating computational artifacts. 1.2.1 Create a computational artifact for creative expression. [P2] 1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. [P2] 1.2.3 Create a new computational artifact by combining or modifying existing artifacts. [P2] 1.2.4 Collaborate in the creation of computational artifacts. [P6] 1.2.5 Analyze the correctness, usability, functionality, and suitability of computational artifacts. [P4] 1.3.1 Use computing tools and techniques for creative expression. [P2] **Abstraction** 2.1.1 Describe the variety of abstractions used to represent data. [P3] 2.1.2 Explain how binary sequences are used to represent digital data. [P5] 2.2.1 Develop an abstraction when writing a program or creating other computational artifacts. [P2] **Data** 3.1.1 Use computers to process information, find patterns, and test hypotheses about digitally processed information to gain insight and knowledge. [P4] **Global Impacts** 7.1.1 Explain how computing innovations affect communication, interaction, and cognition. [P4] 7.2.1 Explain how computing has impacted innovations in other fields. [P1] 7.4.1 Explain the connections between computing and economic, social, and cultural

	contexts. [P1]

Unit 2: The Internet

In this unit students extend their understanding of the internet and how it functions by building off the concepts they learned when sending bits in Unit 1. The unit largely explores the structure and design of the internet and the implications of those design decisions including the reliability of network communication, the security of data, and personal privacy. Topics include the Internet Protocol (IP), DNS, TCP/IP, cryptography and other security and hacking concerns.

The unit starts with students being presented with a more robust Internet Simulator that students will use to solve some of the classic problems of network communication such as addressing devices, redundancy, latency and throughput. Students work together to invent solutions and protocols to many of the problems that arise. The second half of the unit asks students to consider how information might be encrypted to ensure privacy and some of the tradeoffs involved. Finally, some problems in encryption are used as a way to talk about computationally hard problems.

Lessons	**CSP Framework Learning Objectives Addressed** (addressed explicitly or implicitly through practice)
1. Addressing Messages 2. Addressing for Computers 3. Name-to-address mapping 4. Finding your friends 5. IP Addresses, URLs, and DNS 7. Packet Switched Networks 8. Problems with packets 9. Redundancy and Routing 10. Who can see my data? 11. Routing, Scaling, The internet 12. Securing our bits 14. Intro to encryption 15. Computationally hard problems 16. “Hard” security 17 Public Key Cryptography 19. Security and Hacking in the real world 20. Intro to AppLab 21. Putting information on the	**Creativity** 1.2.1 Create a computational artifact for creative expression. [P2] 1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. [P2] 1.2.4 Collaborate in the creation of computational artifacts. [P6] 1.2.5 Analyze the correctness, usability, functionality, and suitability of computational artifacts. [P4] 1.3.1 Use computing tools and techniques for creative expression. [P2] **Abstraction** 2.1.1 Describe the variety of abstractions used to represent data. [P3] 2.1.2 Explain how binary sequences are used to represent digital data. [P5] 2.3.1 Use models and simulations to represent phenomena. [P3] **Data** 3.1.1 Use computers to process information, find patterns, and test hypotheses about digitally processed information to gain insight and knowledge. [P4] 3.1.2 Collaborate when processing information to gain insight and knowledge. [P6] 3.3.1 Analyze how data representation, storage, security, and transmission of data involve computational manipulation of information. [P4] **Algorithms** 4.1.2 Express an algorithm in a language. [P5] 4.2.1 Explain the difference between algorithms that run in a reasonable time and those that do not run in a reasonable time. [P1] 4.2.2 Explain the difference between solvable and unsolvable problems in computer science. [P1] 4.2.3 Explain the existence of undecidable problems in computer science. [P1] 4.2.4 Evaluate algorithms analytically and empirically for efficiency, correctness, and clarity. [P4] **Programming** 5.2.1 Explain how programs implement algorithms. [P3]

web **Practice Performance Task:** The Internet and your privacy	5.4.1 Evaluate the correctness of a program. [P4] 6.1.1 Explain the abstractions in the Internet and how the Internet functions. [P3] **Internet** 6.2.1 Explain characteristics of the Internet and the systems built on it. [P5] 6.2.2 Explain how the characteristics of the Internet influence the systems built on it. [P4] 6.3.1 Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it. [P1] **Global Impacts** 7.1.1 Explain how computing innovations affect communication, interaction, and cognition. [P4] 7.2.1 Explain how computing has impacted innovations in other fields. [P1] 7.3.1 Analyze the beneficial and harmful effects of computing. [P4] 7.4.1 Explain the connections between computing and economic, social, and cultural contexts. [P1]

Unit 3: Programming

This unit introduces students to programming in the javascript language and creating small applications (apps) that live on the web. This introduction places a heavy emphasis on understanding the principles of computer programming and revealing those things that are universal to all computing and any programming language. Students will program in a new environment created by Code.org called *App Lab* that has many features, chief among them the ability to write javascript programs with click-and-drag blocks or just typing text - allowing the user to switch back and forth at will. This should greatly ease the transition to typing text-based programming languages.

The unit begins with students solving problems with classic turtle-style programming, focusing on the power of procedural abstraction and personal expression with code. After learning some basics of programming with the turtle, we gradually blend in elements more commonly seen in apps, like buttons and text inputs, images and so on, teaching programming from an event-driven perspective. The unit concludes with students creating a small app of their own to share with friends and family.

Lessons	**CSP Framework Learning Objectives Addressed** (addressed explicitly or implicitly through practice)
1. Solving Big Problems with CS 2. The Need for Programming Languages 3. A Quick Tour of AppLab 4. Programming with Primitives 5. Function Junction 6. Function Challenge 7. Parameterized Functions 8. Project: Design Your Own Font 9. Application Programming	**Creativity** 1.1.1 Apply a creative development process when creating computational artifacts. [P2] 1.2.1 Create a computational artifact for creative expression. [P2] 1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. [P2] 1.2.4 Collaborate in the creation of computational artifacts. [P6] 1.2.5 Analyze the correctness, usability, functionality, and suitability of computational artifacts. [P4] 1.3.1 Use computing tools and techniques for creative expression. [P2] **Abstraction** 2.2.1 Develop an abstraction when writing a program or creating other computational artifacts. [P2] 2.2.2 Use multiple levels of abstraction to write programs. [P3] 2.2.3 Identify multiple levels of abstractions that are used when writing programs. [P3] **Data** 3.1.1 Use computers to process information, find patterns, and test hypotheses about digitally processed information to gain insight and knowledge. [P4]

Interface 10. Team Design 11. Team Programming 12. Variables 13. Variable Arithmetic 14. Numeric input 15. Interactive Apps with Variables 16. Conditionals (to be re-written) 17. If Statements (to be re-written) 18. Nested If-statements 19. Iteration and Loops 20. Project: Quiz-App 21. What does the Internet know about you? **Practice PT:** Writing about code	3.1.3 Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notations, and precise language. [P5] 3.2.1 Extract information from data to discover and explain connections, patterns, or trends. [P1] 3.3.1 Analyze how data representation, storage, security, and transmission of data involve computational manipulation of information. [P4] **Algorithms** 4.1.1 Develop an algorithm for implementation in a program. [P2] 4.1.2 Express an algorithm in a language. [P5] **Programming** 5.1.1 Develop a program for creative expression, to satisfy personal curiosity, or to create new knowledge. [P2] 5.1.2 Develop a correct program to solve problems. [P2] 5.1.3 Collaborate to develop a program. [P6] 5.2.1 Explain how programs implement algorithms. [P3] 5.3.1 Use abstraction to manage complexity in programs. [P3] 5.4.1 Evaluate the correctness of a program. [P4] 5.5.1 Employ appropriate mathematical and logical concepts in programming. [P1] **Global Impacts** 7.1.2 Explain how people participate in a problem solving process that scales. [P4] 7.2.1 Explain how computing has impacted innovations in other fields. [P1] 7.4.1 Explain the connections between computing and economic, social, and cultural contexts. [P1]

Unit 4: Data

In this unit students continue programming and building apps, but now with a heavier focus on data. Being able to extract knowledge from data is an important aspect of CS Principles and in this unit students will do that in a number of ways. Students will write programs that generate data to model or simulate a scenario they wish to investigate. Students will process large lists of data imported from other sources and also pull data from live data APIs. Students will also more fully use App Lab's cloud data storage capabilities to create databases to use with their own apps.

The unit begins with students designing and running monte carlo-type experiments to investigate the answer to data-driven questions that can be simulated on the computer with many trials. Students then write programs that process large lists of data to perform simple searches or aggregations. The unit concludes with some big data investigations that encourages students to query a remote API that can return data and artifacts they can use in their apps.

Lessons	**CSP Framework Learning Objectives Addressed** (addressed explicitly or implicitly through practice)
1. Utility of Random numbers	**Creativity** 1.1.1 Apply a creative development process when creating computational artifacts. [P2] 1.2.1 Create a computational artifact for creative expression. [P2]

2. A coin flipping experiment 3. Construct your own experiment 4. **Project:** Run your experiment 5. Processing Lists and what “Big Data” means 6. Process a small list 7. Manipulate items in a list 8. Investigate image data 9. Simple image filters 10. **Practice PT**: Make your own image filter 11. Storing user data in a database 12. Retrieving data from your database 13. The risks of storing personal data 14. **Practice PT**: Make an app that uses data from a database 15. Experiments with big data 16. Programming against an api (wolfram) 17. Extracting knowledge from data	1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. [P2] 1.2.4 Collaborate in the creation of computational artifacts. [P6] 1.2.5 Analyze the correctness, usability, functionality, and suitability of computational artifacts. [P4] 1.3.1 Use computing tools and techniques for creative expression. [P2] **Abstraction** 2.2.3 Identify multiple levels of abstractions that are used when writing programs. [P3] 2.3.1 Use models and simulations to represent phenomena. [P3] 2.3.2 Use models and simulations to formulate, refine, and test hypotheses. [P3] **Data** 3.1.1 Use computers to process information, find patterns, and test hypotheses about digitally processed information to gain insight and knowledge. [P4] 3.1.2 Collaborate when processing information to gain insight and knowledge. [P6] 3.1.3 Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notations, and precise language. [P5] 3.2.1 Extract information from data to discover and explain connections, patterns, or trends. [P1] 3.3.1 Analyze how data representation, storage, security, and transmission of data involve computational manipulation of information. [P4] **Algorithms** 4.1.1 Develop an algorithm for implementation in a program. [P2] 4.1.2 Express an algorithm in a language. [P5] 4.2.1 Explain the difference between algorithms that run in a reasonable time and those that do not run in a reasonable time. [P1] 4.2.4 Evaluate algorithms analytically and empirically for efficiency, correctness, and clarity. [P4] **Programming** 5.1.1 Develop a program for creative expression, to satisfy personal curiosity, or to create new knowledge. [P2] 5.1.2 Develop a correct program to solve problems. [P2] 5.1.3 Collaborate to develop a program. [P6] 5.2.1 Explain how programs implement algorithms. [P3] 5.3.1 Use abstraction to manage complexity in programs. [P3] 5.4.1 Evaluate the correctness of a program. [P4] 5.5.1 Employ appropriate mathematical and logical concepts in programming. [P1] **Internet** 6.3.1 Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it. [P1] **Global Impacts** 7.2.1 Explain how computing has impacted innovations in other fields. [P1] 7.3.1 Analyze the beneficial and harmful effects of computing. [P4] 7.4.1 Explain the connections between computing and economic, social, and cultural contexts. [P1]

Unit 5 - Performance Tasks

This unit is primarily set aside to ensure that students have enough time in class to work on and complete their performance tasks for submission to the college board. There are a few guided activities for teachers to run that will help students get organized and ensure they have reasonable project plans that can be achieved in the time allotted.

Lessons: TBD