Code.org Computer Science Principles

Syllabus and Overview

rev. April 2016

AP® Computer Science Principles

Code.org's Computer Science Principles (CSP) curriculum is a **full-year, rigorous, entry-level course** that introduces high school students to the foundations of modern computing. The course covers a broad range of foundational topics such as programming, algorithms, the Internet, big data, digital privacy and security, and the societal impacts of computing.

Course Snapshot

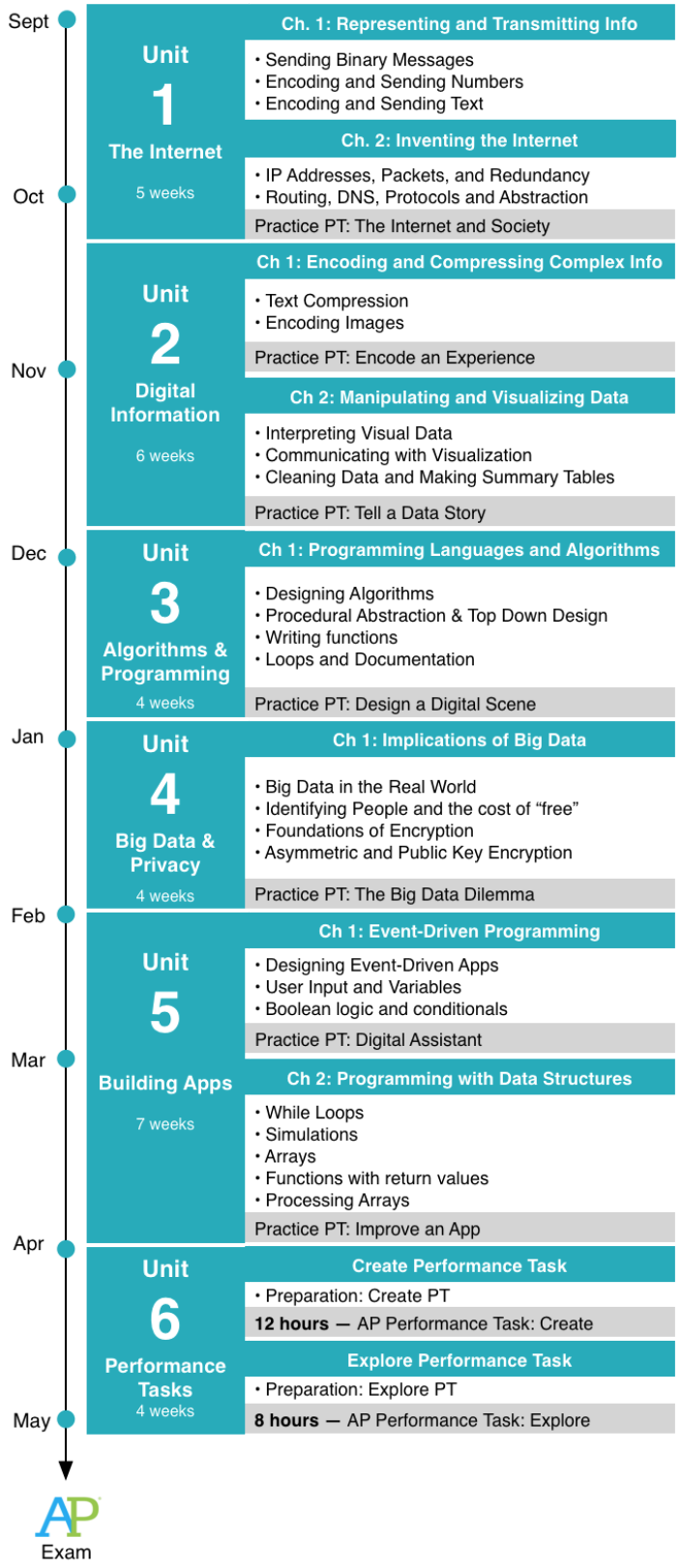
To the right is a snapshot of the course. The course contains **five core units of study,** with a sixth unit devoted almost exclusively to students working on their *AP Performance Task* (PT) projects. Each unit has one or two "chapters" of related lessons that usually conclude with some kind of project or summative assessment. A timeline showing a typical school year is shown to give a rough estimate of pacing. **Note: the performance task submission deadline is the end of April, and the written AP Exam is May 5, 2017.**

AP Endorsed

Code.org is recognized by the College Board as an endorsed provider of curriculum and professional development for AP® Computer Science Principles (AP CSP). This endorsement affirms that all components of Code.org CSP's offerings are aligned to the AP Curriculum Framework standards and the AP CSP assessment. Using an endorsed provider affords schools access to resources including an AP CSP syllabus pre-approved by the College Board's AP Course Audit, and officially recognized professional development that prepares teachers to teach AP CSP.

Code.org Endorsed
Syllabus ID #1648108v1

AP is a trademark registered and owned by the College Board.



Code.org Computer Science Principles
Syllabus and Overview

rev. April 2016

Curriculum Overview and Goals

Computing affects almost all aspects of modern life and *all* students deserve a computing education that prepares them to pursue the wide array of intellectual and career opportunities that computing has made possible.

This course is not a tour of current events and technologies. Rather, this course seeks to provide students with a "future proof" foundation in computing principles so that they are adequately prepared with both the knowledge and skills to live and meaningfully participate in our increasingly digital society, economy, and culture.

The Internet and Innovation provide 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 considering the implications of a computing innovation of their own design. Along the way students learn:

- How the Internet works and its impacts on society.
- How to program and rapidly prototype small JavaScript applications both to solve problems and to satisfy personal curiosity.
- How to collect, analyze and visualize data to gain insight and knowledge.
- How to evaluate the beneficial and harmful effects to people and society brought on by computing innovations.

Addressing Diversity, Equity, and Broadening Participation in the Curriculum

A central goal of Code.org's CSP curriculum is for it to be accessible to all students, especially those in groups typically underrepresented in computing. To this end, we have worked to provide examples and activities that are relevant and topical enough for students to connect back to their own interests and lives. Wherever possible, and especially in the videos that accompany the curriculum, we seek to **highlight a diverse and impressive array of role models** in terms of gender, race, and profession from which students can draw inspiration and "see themselves" participating in computing.

The curriculum assumes no prior knowledge of computing and is written to support *both* students and teachers who are new to the discipline. Activities are designed and structured in such a way that students with diverse learning needs have space to find their voice and to express their thoughts and opinions. The activities, videos, and computing tools in the curriculum strive to have a broad appeal and to be accessible to a student body diverse in background, gender, race, prior knowledge of computing, and personal interests.

Broadening student participation in computer science is a national goal, and effectively an issue of social justice. Fancy tools and motivational marketing messages only get you so far. We believe that the real key to attracting students to computer science and then sustaining that growth has as much to do with the teacher in the classroom as it does with anything else. The real "access" students need to computing is an **opportunity to *legitimately and meaningfully participate* in every lesson** regardless of the student's background or prior experience in computing coming into the course. For example, the course begins with material that is

challenging but typically unfamiliar even to students who have some prior experience or knowledge of computer science. Students should not feel intimidated that others in the class are starting with a leg up on the material.

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 requires 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 due the expectations of student responsibility and maturity for an AP course.

The curriculum itself does not assume any prior knowledge of computing concepts before entering the course. It is intended to be suitable as a **first course in computing** though students with a variety of backgrounds and prior experiences will also find the course engaging and with plenty of challenges. While it is increasingly likely that students entering this AP course in high school will have had *some* prior experience in computer science (particularly with programming), that experience is equally likely to be highly varied both in quantity and quality. It is for this reason that the course *does not* start with programming, but instead with material that is much more likely to put all students on a level playing field for the first few weeks of class. Read more about this below in the description of Unit 1.

Teaching the course

The work of providing an equitable classroom doesn't stop with curriculum -- the classroom environment and teaching practice must also be structured such that all learners can access and engage with the material at a level that doesn't advantage a few at the expense of others.

Equitable teaching practices are inextricably linked and woven into the design and structure of our lessons, and in some cases the reason for their existence.

The curriculum provides a number of resources for the teacher, such as assessment support, computing tools that are designed for learning specific concepts, and the programming environment, App Lab. These resources have been specifically curated *for each step of each lesson*, which allows the teacher to act in the role of facilitator and coach when addressing unfamiliar material, rather than having to worry about presenting or lecturing.

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 Code.org's professional development program. A teacher who is motivated to teach a course like this, but who has limited technical or formal computer science experience should be able to be successful. At a minimum, we strongly recommend that the teacher have a reasonable 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 other formal computer science training or coursework.

Unit Structure: Units, Chapters, Lessons

While the layout of units appears to be modular, the units of study are loosely scaffolded, **and sequenced to build students' skills and knowledge toward the Enduring Understandings of the CSP Course Framework.** The lessons for each unit assume that students have the knowledge and skills obtained in the previous units. There are also many thematic connections that can be made between and among lessons and units.

Each **unit** attempts to "tell a story" about a particular topic in computing from a more primitive beginning to a more complex end. The lessons in each unit are grouped into one or two **chapters** of a few weeks worth of lessons whose content is related or connected in some way. The course snapshot on the first page shows the chapters for each unit. Each **lesson** is intended to be a complete thought that takes the student from some motivational question or premise to an activity that builds skills and knowledge toward some learning objective(s).

Each unit contains at least one summative assessment, project, or Practice PT that asks students to complete tasks similar to the official PTs. Sometimes these come mid-unit, and sometimes they come closer to the end.

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 teacher to lecture or present on computer science topics if they do not want to. Direct instruction, where necessary, is built into our tools and videos.

Another goal of each lesson is to provide more resources, supports, and activities than a teacher could (or should) use in one lesson. **The teacher plays a large role making choices and ensuring that the activities, inquiry, and reflection are engaging and appropriate for their students, as well as assessing student learning**.

Most lessons have the following structure:

- **A warm-up activity** to activate prior knowledge and/or present a thought-provoking problem
- **An activity** that varies but is typically one of 5 possible types:
    - Unplugged concept invention and problem solving scenarios
    - Using a computational widget to discover concepts more deeply
    - Using external computational tools (such as spreadsheets, or presentation tools)
    - Programming in App Lab
    - Research and communication (Research / Writing / Reflection / Presentation)
- **A wrap-up** activity or reflection to pull together the core concepts of the lesson

Technical Requirements

The course requires and assumes a 1:1 computer lab or setup such that each student in the class has 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 accessible through a web browser.

While the course features many “unplugged” activities away from the computer, daily access to a computer is essential for every student. It is not required that students have access to computers at home, but because almost all of the materials are online, students with access to computers outside of class and at home will have access to much of their coursework outside of class.

Computational Tools, Resources and Materials

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

Lesson Plans

- Instructional guides for every lesson
- Activity Guides and handouts for students
- Formative and summative assessments
- Exemplars, rubrics, and teacher dashboard

Videos
- Student videos - including tutorials, instructional and inspirational videos
- Teacher videos - including lesson supports and pedagogical tips and tricks

Computational Tools
- Widgets and simulators for exploring individual computing concepts
- **Internet Simulator** - Code.org's tool for investigating the various "layers" of the internet
- **App Lab** - Code.org's JavaScript programming environment for making apps

A few lessons call for typical classroom supplies and manipulatives such as poster paper and markers, as well as additional materials like dixie cups, string, playing cards, a handful of Lego blocks, etc. In most cases there are alternatives to these materials if necessary.

Suggested Text:

Blown to Bits http://www.bitsbook.com/

This course does not require or follow a textbook. *Blown to Bits* is a book that can be accessed online **free of cost**. Many of its chapters are excellent supplemental reading for our course, especially for material in Units 1, 2 and 4. We refer to chapters as supplemental reading in lesson plans as appropriate.

AP® Assessment

The AP Assessment consists of a 74-question multiple choice exam and two "through-course" assessments called the *AP Performance Tasks* (PTs). The tasks can be found in the official **AP CS Principles Exam and Course Description**.
- Create Performance Task (p. 108)
- Explore Performance Task (p. 111)

Assessments in the Curriculum

The course provides a number of assessment types and opportunities. For students, the goal of the assessments is to prepare them for the AP exam and performance tasks. For teachers, the goal is to use assessments to help guide instruction, give feedback to students, and make choices about what to emphasize in lessons.

Summative Assessments:

The curriculum contains two types of summative assessments that teachers may elect to use. They are intended to mimic the AP assessments though in more bite-sized chunks.

1. Fixed Response Assessments

After a group of concepts has been adequately covered - typically this means every 5-8 lessons (roughly every few weeks) - a fixed response assessment with items such as multiple choice, matching, choose two, short answer, etc. appears in the curriculum.

2. Practice Performance Task Assessments

Each unit contains at least one project designed in the spirit of the Advanced Placement Performance Tasks (PTs). These **Practice PTs** are smaller in scope, contextualized to the unit

of study and are intended to help prepare students to engage in the official administration of the AP PTs at the end of the course.

3. Project Rubrics
The curriculum contains rubrics for assessing certain kinds of student work:

- Written and project work
- Practice PTs
- Programming projects
- Student presentations

Formative Assessments:

The curriculum provides teachers many opportunities for formative assessment (such as checks for understanding). These include, but are not limited to:

Assessments in Code Studio
All lesson materials can be accessed by students on a single platform called Code Studio. In addition to housing lesson descriptions, instructional materials, and programming exercises in App Lab, Code Studio includes features that assist the teacher in completing formative assessment including:

- Multiple choice or matching questions related to questions on the chapter summative assessment.
- Free-response text fields where students may input their answer.
- Access to student work within the App Lab programming environment and other digital tools and widgets used in the curriculum.
- The ability for students to submit final versions of App Lab projects

Worksheets and Activity Guides

- Many lessons contain worksheets or activity guides that ask students to write, answer questions, and respond to prompts (Answer keys provided) that could be used as formative assessment

It is up to the classroom teacher:

- to determine the appropriateness of the assessments for their classrooms
- to decide how to use, or not to use, the assessments for grading purposes. The curriculum and Code Studio does not provide teachers with a gradebook, and we do not provide recommendations for how to assign grades based on performance on an assessment.

Coverage of the AP CS Principles Framework and Computational Thinking Practices

The CS Principles Framework outlines seven "Big Ideas" of computing, and six "Computational Thinking Practices". Activities in the course should ensure that students are engaging in the Computational Thinking Practices while investigating the Big Ideas.

Seven Big Ideas

The [course is] organized around seven big ideas, which encompass ideas foundational to studying computer science.

Big Idea 1: Creativity
Big Idea 2: Abstraction
Big Idea 3: Data
Big Idea 4: Algorithms
Big Idea 5: Programming
Big Idea 6: The Internet
Big Idea 7: Global Impacts

Six Computational Thinking Practices

Computational thinking practices capture important aspects of the work that computer scientists engage in.

P1: Connecting Computing
P2: Creating Computational Artifacts
P3: Abstracting
P4: Analyzing Problems and Artifacts
P5: Communicating
P6: Collaborating

These *Big Ideas* and *Practices* are not intended to be taught in any particular order, nor are they units of study. The Big Ideas all overlap, intersect, and reference each other. The practices represent higher order thinking skills, behaviors, and habits of mind that need to be constantly visited, repeatedly honed, and refined over time.

For example, a learning objective listed under the Big Idea *Abstraction* also references the Practice of *Programming*.

> *LO 2.2.1 Develop an abstraction when writing a program or creating other computational artifacts. [P2]*

Even though this particular learning objective highlights practice *P2: Creating Computational Artifacts,* it clearly will also engage the practice of *Abstracting.* Therefore, this single learning objective represents an intersection of two *Big Ideas*: Abstraction and Programming, while also engaging at least two *Computational Thinking Practices*.

This curriculum takes the view that the 7 Big Ideas actually represent a body of knowledge in which topics of study (The Internet, Programming and Data) intersect with more general principles of computing (Creativity, Abstraction, Algorithms and Global Impacts). It is much more usefully viewed in two dimensions. The chart below shows the intersections of the big ideas and examples of topics addressed in the curriculum.

rev. April 2016

	Internet	Data	Programming
Creativity	Invent a communication protocol	Visualizing Data Create a visualization	Make a digital scene. Program an app.
Abstraction	Internet Protocols	Encoding images in binary	Writing procedures and functions
Algorithms	Routing, Encryption	Data Compression, Searching and data mining	String manipulation Array processing
Global Impact	Security, Privacy, Hacking	Implications of collection and storage of big data	Software can solve some but not all problems
	Units 1, 4	**Units 2, 4**	**Units 3, 5**

Units 1, 2 and 3 survey the the Big Ideas *Internet*, *Data* and *Programming* as major topics of study. For coverage we look at their intersections with the other four big ideas: *Creativity*, *Abstraction*, *Algorithms*, *Global Impact*. **Unit 4** goes deeper into big data, especially related to the societal implications of big data. And Unit **Unit 5** goes deeper into programming concepts.

The six **computational thinking practices** are addressed continuously throughout the curriculum in a number of ways. They are woven into the curriculum, engineered into activities and projects, as well as in teaching tips for lessons. The acts of **abstracting** [P3] and **creating** and **analyzing computational artifacts** [P2 and P4] are part and parcel of many of the lessons, activities, and projects themselves. The teacher plays a large role in ensuring that students are **connecting computing** [P1], **collaborating** [P6] effectively, and **communicating** [P5] both in writing and speaking. You can find explicit reference to the computational practices used in lessons in the unit overviews below.

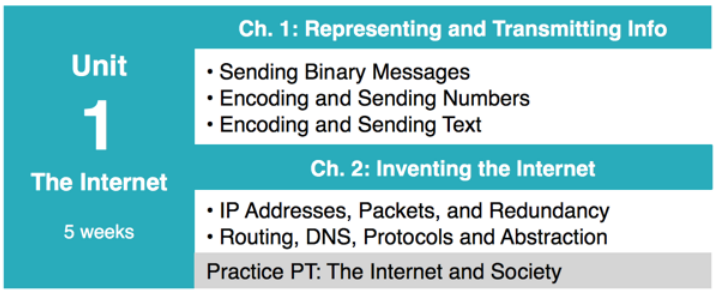
Unit Overviews

What follows are more in-depth descriptions of each unit of study which explain the topics covered and what students will be doing. Each unit also highlights a particular lesson, project or assignment of interest, explaining what students do and showing which learning objectives and computational thinking practices that particular assignment addresses. In the unit descriptions we also reference the *Enduring Understandings* from the CSP Framework with bolded text and a parenthetical number. For example you might see: **people write programs to execute algorithms (5.1).** See the CSP Framework document for listings of all the Enduring Understandings.

rev. April 2016

Unit 1: The Internet

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 - especially, numbers, text, and communication protocols. The first unit of this course purposefully addresses material that is fundamental to computing but with which many students, even those with computers at home or who have some prior experience with programming, are unfamiliar. This levels the playing field for participation and engagement right from the beginning of the course.



Chapter 1 of 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 Binary Messages* lesson students work with a partner to invent and build their own bit-sending "device." Complexity increases as students adapt their machines to handle multi-bit messages and increasingly complex information. Students use an Internet Simulator that allows them to develop and test binary encodings and communication protocols of their own invention.These should be an illustrative set of activities that helps build toward the enduring understandings that: **A variety of abstractions built upon binary sequences can be used to represent all digital data (2.1)** and that **characteristics of the Internet influence the systems built on it (6.2).**

Chapter 2 of the unit is called "Inventing the Internet" because through the unit students continue to solve problems similar ones that had to be solved to build the real Internet. Students design their own versions of protocols, each one layered on the previous one, in a process that mimics the layered sets of protocols on the real Internet and builds toward the enduring understandings that **The Internet is a network of autonomous systems (6.1)** and that **How the Internet was designed (layers of protocols) affects the systems built on top of it (6.2)** as well as its ability to grow and adapt.

For the practice Performance Task at the end of the unit students research a modern societal issue related to the Internet such as "Net Neutrality" or internet censorship, which layers in enduring understandings about the fact that **computing has a global affect -- both beneficial and harmful -- on people and society (7.3)**.

Unit 1 Lessons

Topics	EU	LO [P] (Ek)	Lessons
Chapter 1: Representing and Transmitting Information			
Getting Started	7.1 7.2 7.3 7.4	7.1.1 [P4] (A-O) 7.2.1 [P1] (A-C,G) 7.3.1 [P4] (A-O) 7.4.1 [P1] (A-D)	Personal Innovations
Sending Binary Messages	2.1 2.3 3.3 6.1 6.2	2.1.1 [P3] (A-C,E) 2.1.2 [P5] (D-F) 2.3.1 [P3] (A-D) 2.3.2 [P3] (A) 3.3.1 [P4] (A-B) 6.1.1 [P3] (A-D) 6.2.1 [P5] (A,D) 6.2.2 [P4] (A-K)	Build a Bit Sending Device Sending Binary Messages with the Internet Simulator *Sending Bits in the Real World (Optional)*
Encoding and Sending Numbers	2.1 2.3 3.1 3.3 6.2	2.1.1 [P3] (A-G) 2.1.2 [P5] (A-F) 2.3.1 [P3] (A-D) 2.3.2 [P3] (A-E) 3.1.1 [P4] (A,B,D,E) 3.3.1 [P4] (A,B) 6.2.2 [P4] (D,G,H)	Number Systems - Circles, Triangles, Squares Binary Numbers Sending Numbers *Encoding Numbers in the Real World (Optional)*
Encoding and Sending Text	2.1 3.1 3.3 6.1 6.2	2.1.1 [P3] (A-E) 2.1.2 [P5] (B-F) 3.1.1 [P4] (A,D,E) 3.1.2 [P6] (A-D) 3.1.3 [P5] (A,E) 3.3.1 [P4] (A,B,G) 6.1.1 [P3] (A-D) 6.2.2 [P4] (D,F-H)	Encoding and Sending Formatted Text
Chapter 2: Inventing the Internet			
Internet Addresses, Packets, and Redundancy	2.1 3.3 6.1 6.2 6.3 7.3 7.4	2.1.1 [P3] (A-C,E) 2.1.2 [P5] (D-F) 3.3.1 [P4] (A-F) 6.1.1 [P3] (B-E) 6.2.1 [P5] (D) 6.2.2 [P4] (B,D,G) 6.3.1 [P1] (A) 7.3.1 [P4] (A,D,E,G,L) 7.4.1 [P1] (C-E)	The Internet is for Everyone Internet Addressing Routers and Redundancy Packets and Making a Reliable Internet
Algorithms of the Internet: Routing *(Optional)*	*4.1* *4.2*	*4.1.1 [P2] (B,H,I)* *4.1.2 [P5] (A-C,F,I)* *4.2.1 [P1] (A,B)* *4.2.4 [P4] (A-D,G)*	*Minimum Spanning Tree (Optional)* *Shortest Path Problem (Optional)* *How Routers Learn (Optional)*
Protocols and Abstraction	6.1 6.2 6.3	6.1.1 [P3] (A-I) 6.2.1 [P5] (B,C) 6.2.2 [P4] (C-E,H) 6.3.1 [P1] (B)	The Need for DNS HTTP and Abstraction on the Internet
Practice PT		**6.3.1 [P1], 7.1.1 [P4], 7.3.1 [P4], 7.4.1 [P1] 7.5.2 [P5]**	**Practice PT - The Internet and Society**

Unit 1: Practice PT Highlight

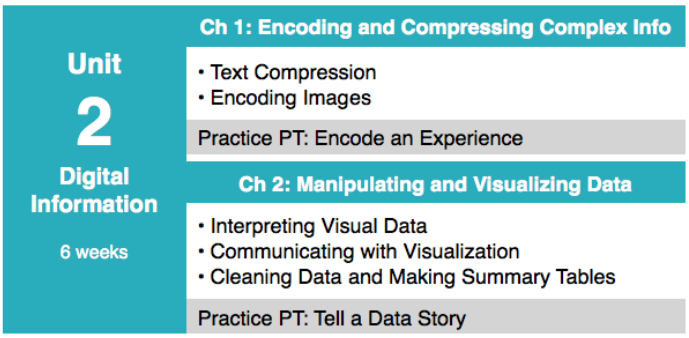
Practice PT: The Internet and Society
Students will research and prepare a flash talk about a social issue related to the Internet. Students pick one of: Net Neutrality, Internet Censorship, or Computer/Network Surveillance. This lesson is good practice for certain elements of the Explore Performance Task, which students will complete later in the school year. Students will do a bit of research about impacts of the Internet, explain some technical details related to ideas in computer science, and connect these ideas to global and social impacts. Students will practice synthesizing information, and presenting their learning in a flash talk.

Learning Objectives Addressed:
Internet: 6.1.1[P3], 6.2.2[P4], 6.3.1 [P1]
Global Impacts: 7.1.1 [P4], 7.3.1 [P4], 7.4.1 [P1], 7.5.2 [P5]

Computational Thinking Practices Emphasized:
P1: Connecting Computing
P5: Communicating

Unit 2: Digital Information

This unit further explores the ways that digital information is encoded, represented and manipulated. In this unit students will look at and generate data, clean it, manipulate it, and create and use visualizations to identify patterns and trends. Students will use a variety of tools including Code.org widgets and external data manipulation and visualization tools (such as Excel or Google Sheets).



The unit begins by continuing to look at the possibilities and limitations of encoding information in binary and building on the enduring understanding that **there are trade offs when representing information as digital data (3.3)**. Students create an image that they encode with binary by hand, and also look at a variety of data compression techniques for text and images. The Practice PT: Encode an Experience has students devise their own completely new data encoding scheme for something complex like a human experience which has students deeply consider how a **variety of abstractions built upon binary sequences can be used to represent all digital data (2.1).**

In the second chapter students develop skills interpreting visual data and using spreadsheet and visualization tools to create their own digital artifacts. Through an ongoing project - the "class data tracker" - in which students themselves are the subject of the data, students learn how to collect and clean data, and to use a few common tools for computing aggregations and creating visualizations. These activities build toward the enduring understandings that **people use computer programs to process information to gain insight and knowledge (3.1)** and that **computing facilitates exploration and the discovery of connections in information (3.2)**.

As students explore ways in which **computing enhances communication, interaction, and cognition (7.1)**, they also examine how human error during data analysis can lead to inaccurate and potentially damaging conclusions. This leads to a deeper understanding that there are **trade offs**, and potentially **beneficial and harmful effects when representing information as digital data (3.3, 7.3).**

Unit 2 Lessons

Topics	EU	LO [P] (Ek)	Lessons
Chapter 1: Encoding and Compressing Complex Information			
Compression and Encoding Images	1.1 1.2 1.3 2.1 2.2 2.3 3.1 3.2 3.3	1.1.1 [P2] (A,B) 1.2.1 [P2] (A) 1.3.1 [P2] (C) 2.1.1 [P3] (A-C) 2.1.2 [P5] (D-F) 2.2.1 [P2] (A,B) 2.3.1 [P3] (A-D) 3.1.1 [P4] (A,D,E) 3.1.2 [P6] (A-D) 3.1.3 [P5] (A,E) 3.2.1 [P1] (G-I) 3.3.1 [P4] (A-E,G)	Text Compression Encoding B&W Images Encoding Color Images Lossy Compression and File Formats
Practice PT	**2.1** **2.2**	2.1.1 [P3] (A-E) 2.1.2 [P5] (A,B,D,F) 2.2.1 [P2] (A,B)	**Practice PT - Encode an Experience**
Chapter 2: Manipulating and Visualizing Data			
Collecting Data	3.2 5.1 7.1 7.2	3.2.1 [P1] (A,B,C) 5.1.1 [P2] (F) 7.1.1 [P4] (C) 7.2.1 [P1] (A,B,G)	Introduction to Data The Data Tracker Project
Interpreting Visual Data	1.1 1.2 3.1 3.2 7.1 7.4	1.1.1 [P2] (A,B) 1.2.1 [P2] (A,B,E) 1.2.5 [P4] (A-D) 3.1.1 [P4] (A,B,D,E) 3.1.2 [P6] (A-F) 3.1.3 [P5] (A-E) 3.2.1 [P1] (A-E) 7.1.1 [P4] (E-G) 7.4.1 [P1] (A,C,D)	Finding Trends with Visualizations Check Your Assumptions Good and Bad Data Visualizations
Communicating with Visualization	1.1 1.2 3.1 3.2 3.3 7.3	1.1.1 [P2] (A,B) 1.2.1 [P2] (A-C) 1.2.4 [P6] (A,B,F) 3.1.1 [P4] (A-E) 3.1.2 [P6] (A-F) 3.1.3 [P5] (A-C) 3.2.1 [P1] (A-G,I) 3.2.2 [P3] (C,G) 3.3.1 [P4] (F) 7.3.1 [P4] (G)	Making Data Visualizations Discover a Data Story Cleaning Data Creating Summary Tables
Practice PT	1.2 3.1 7.3 7.5	1.2.1 [P2] (A-C,E) 1.2.2 [P2] (A,B) 1.2.5 [P4] (A-D) 3.1.3 [P5] (A-D) 7.3.1 [P4] (J) 7.5.2 [P5] (A,B)	Tell a Data Story

Unit 2: Practice PT highlights

Practice PT: Encode an Experience
Students invent a binary encoding (file format) for a real life experience. Students must figure out a way to encode or represent with data, the elements of some kind of human experience. How might you encode a birthday party? or a soccer game? or the brush strokes of a real painting? Students come up with their own creation and present their work in a format similar to that of a Performance Task. While the project is done individually the lesson helps students through an iterative feedback process with a partner. This assignment emphasizes the writing process, and giving and incorporating feedback from peers.

Learning Objectives Addressed:
Creativity: 1.1.1 [P2], 1.2.4 [P6]
Abstraction: 2.1.1 [P3], 2.1.2 [P5], 2.2.1 [P2]
Data: 3.2.1 [P1], 3.3.1 [P4]

Computational Thinking Practices Emphasized:
P1: Connecting Computing
P3: Abstracting
P5: Communicating
P6: Collaborating

Tell a Data Story - Communicate Data Visually
This small project culminates a series of lessons in which students, provided a set of raw data, must use digital tools to collaboratively investigate the data to discover possible connections and trends. In the end students must produce a visual explanation of their findings in the data and write a short piece about what the data shows. The emphasis is on producing the visual communication. The reflection questions mimic those on the Explore PT.

Learning Objectives Addressed:
Creativity: 1.1.1 [P2], 1.2.1 [P2], 1.2.4 [P6], 1.2.5 [P4]
Data: 3.1.1 [P4], 3.1.2 [P6], 3.1.3 [P5], 3.2.1 [P1]
Global Impacts: 7.1.1 [P4], 7.4.1 [P1]

Computational Practices Emphasized:
P1: Connecting Computing
P2: Creating Computational Artifacts
P5: Communicating
P6: Collaborating

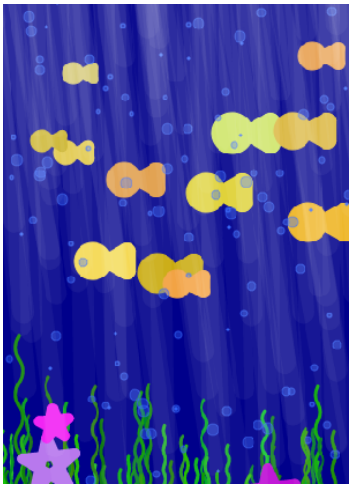
Unit 3: Algorithms and Programming

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 general principles of computer programming and revealing those things that are universally applicable to any programming language.

Unit 3 Algorithms & Programming 4 weeks	Ch 1: Programming Languages and Algorithms
	• Procedural Abstraction • Top Down Design • Writing functions • Loops and Documentation
	Practice PT: Design a Digital Scene

To start the unit we use unplugged activities to introduce algorithms and highlight the need for a programming language to implement them on a computer. These activities will involve the whole class working in small groups to solve problems using simple manipulatives like playing cards or blocks. We want to draw connections here between the rules of Internet protocols developed earlier in the course, in which students acted as the computer processing the information. Many of the structured and systematic thinking that goes into developing communication protocols feels similar to designing algorithms - ultimately you're designing a series of steps to solve a problem that a machine could follow. We want to establish the dual enduring understandings that **algorithms are precise sequences of instructions for processes that can be executed by a computer and are implemented using programming languages (4.1)** and **people write programs to execute algorithms (5.2).**

Students are introduced to the App Lab programming environment by writing programs to control a "turtle", an imaginary character that moves around the screen and can draw. In the lessons students learn features of the JavaScript language by going through a series of short tutorials to familiarize students with the environment, and new concepts. There is a heavy emphasis on writing procedures (functions in JavaScript), and using top-down program design - a process by which a large problem is broken down into smaller and more manageable parts. These lessons highlight the way **multiple levels of abstraction are used to write programs (2.2)**.



Along the way students create more and more sophisticated drawings culminating in the **Practice PT: Design a Digital Scene** in which small groups must collaborate to design and share code to create a small vignette created with turtle art. Through the lessons and PTs we want to build toward some enduring understandings that **creative development can be an essential process for creating computational artifacts (1.1)** and that collaboration and **computing enables people to use creative development processes to create computational artifacts for creative expression or to solve a problem (1.2)**.

Unit 3 Lessons

Topics	EU	LO [P] (Ek)	Lessons
Chapter 1: Programming Languages and Algorithms			
Algorithms	4.1 5.2	4.1.2 [P5] (A-C, F, I) 5.2.1 [P3] (E)	The Need For Programming Languages The Need for Algorithms Creativity in Algorithms
Procedural Abstraction and Top-Down Design	2.2 5.1 5.2 5.3 5.4	2.2.1 [P2] (A, B) 2.2.2 [P3] (A, B) 2.2.3 [P3] (A) 5.1.2 [P2] (A-C, I) 5.1.3 [P6] (A-F) 5.2.1 [P3] (A, B) 5.3.1 [P3] (A-D, L) 5.4.1 [P4] (A-E, I)	Using Simple Commands Creating Functions Functions and Top-Down Design
Documentation and Simple Loops	2.2 4.1 5.1 5.3 5.4	2.2.1 [P2] (C) 2.2.2 [P3] (A, B) 2.2.3 [P3] (A, B) 4.1.1 [P2] (D) 5.1.2 [P2] (B-F) 5.3.1 [P3] (A, C-G, L-O) 5.4.1 [P4] (C-K)	APIs and Function Parameters Creating functions with Parameters Looping and Random Numbers
Practice PT	2.2 4.1 5.1 5.3 5.4	2.2.1 [P2] (C) 2.2.2 [P3] (A, B) 2.2.3 [P3] (A, B) 4.1.1 [P2] (D) 5.1.2 [P2] (B, C) 5.1.3 [P6] (A-F) 5.3.1 [P3] (A, C, D, F, G, L) 5.4.1 [P4] (C-K)	**Design a Digital Scene**

Unit 3 Practice PT Highlights

Practice PT: Digital Scene Design
In this project students work with a small team to create a digital scene with turtle graphics. They plan the scene together, code the parts separately and bring them together to make a whole. An important focus of this project is on how teams of programmers work together, and some insight is given into how real engineering teams do this. Students are asked to reflect on their experience in a way that is similar to the *Create* performance task. In terms of programming, a heavy emphasis is on writing functions (procedures) that can be easily incorporated into others' code.

Learning Objectives Addressed:
Creativity: 1.1.1 [P2], 1.2.1 [P2], 1.2.4 [P6], 1.3.1 [P2]
Abstraction: 2.2.1 [P2], 2.2.2 [P3]
Algorithms: 4.1.1 [P2]
Programming: 5.1.1 [P2], 5.1.3 [P6], 5.3.1 [P3]

Computational Practices Emphasized:
P2: Creating Computational Artifacts
P3: Abstracting
P6: Collaborating

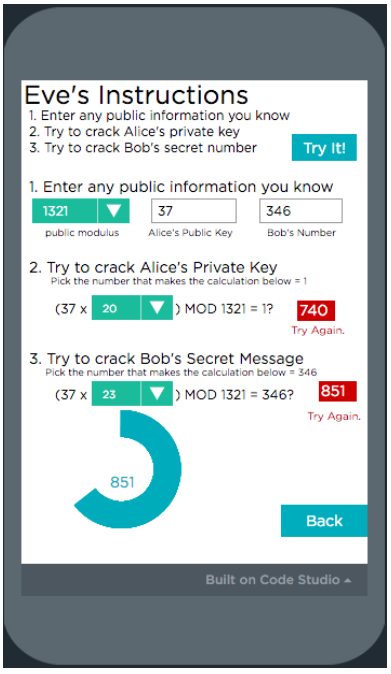
Unit 4: Big Data and Privacy

Unit	Ch 1: Implications of Big Data
4 Big Data & Privacy 4 weeks	• Big Data in the Real World • Identifying People and the cost of "free" • Foundations of Encryption • Asymmetric and Public Key Encryption
	Practice PT: The Big Data Dilemma

The data rich world we live in also introduces many complex questions related to public policy, law, ethics and societal impact. In many ways this unit acts as a unit on current events. It is highly likely that there will be something related to big data, privacy and security going on in the news at any point in time. The major goals of the unit are 1) for students to develop a well-rounded and balanced view about data in the world around them and both the positive and negative effects of it and 2) to understand the basics of how and why modern encryption works.

During the first two weeks of the unit students will research and discuss **innovations enabled by computing in a wide variety of fields (7.2)**. During this time views about the benefits - "Big Data is great!" - and drawbacks - "Big Data is scary!" will swing quickly. We primarily want to build toward the dual enduring understandings that **Computing facilitates exploration and the discovery of connections in information (3.2)** and that **Computing innovations influence and are influenced by the economic, social, and cultural contexts in which they are designed and used (7.4)** while the **beneficial and harmful effects (7.3)** of these things must be weighed and kept in balance.

The activities in the third week around data encryption follow a pattern: introduce an encryption concept through an unplugged activity or thinking prompt, and then "plug it in" by using a Code.org widget to explore the concept further. The purpose of the widgets is to allow students time to play with some of the ideas - often mathematical in nature - underlying different methods of encryption and why they might be susceptible to being "cracked." These explorations lead towards an understanding of computationally hard problems and the fact that **algorithms can solve many but not all computational problems (4.2)**. In particular students should come away with a high level understanding of how asymmetric encryption works and why it makes certain things possible (sending encrypted data without a shared key) and certain things basically impossible (cracking a key). By investigating some of the mathematical foundations of encryption we build toward the enduring understanding that **cybersecurity is an important concern for the Internet and the systems built on it (6.3)** and as always **There are trade offs when representing information as digital data (3.3).**



Unit 4 Lessons

Chapters	EU	LO [P] (Ek)	Lessons / Topics
Chapter 1: Implication of Big Data			
Data in the Real World	1.2 3.1 3.2 3.3 7.1 7.2 7.3 7.5	1.2.5 [P4] (A-D) 3.1.1 [P4] (C-E) 3.1.2 [P6] (F) 3.2.1 [P1] (A-D,G) 3.2.2 [P3] (A-D,G,H) 3.3.1 [P4] (A,B,F) 7.1.1 [P4] (F) 7.2.1 [P1] (A) 7.3.1 [P4] (A,D-M) 7.5.2 [P5] (A,B)	What is Big Data? Rapid Research - Data Innovations Identifying People with Data The Cost of "Free" The Security/Privacy Dilemma
Security and Symmetric Encryption	2.3 3.1 3.3 4.2 6.3 7.3	2.3.2 [P3] (A) 3.1.1 [P4] (A) 3.1.2 [P6] (A,C) 3.3.1 [P4] (B,E,F) 4.2.1 [P1] (A,C,D) 6.3.1 [P1] (C, H-K) 7.3.1 [P4] (G)	The Need for Encryption Cracking the Code
Computationally Hard Problems *(Optional)*	*2.3* *4.2* *6.3*	*2.3.1 [P3] (A,B)* *4.2.1 [P1] (A-D)* *4.2.2 [P1] (A-D)* *4.2.3 [P1] (A,D)* *4.2.4 [P4] (A-C)* *6.3.1 [P1] (H-L)*	*Hard Problems - The Traveling Salesperson Problem (Optional)* *One Way Functions - The WiFi Hotspot Problem (Optional)*
Asymmetric Encryption	2.3 4.2 6.3	2.3.1 [P3] (A,B) 4.2.1 [P1] (A-D) 4.2.2 [P1] (A-D) 4.2.3 [P1] (A,D) 4.2.4 [P4] (A-C) 6.3.1 [P1] (H-L)	Asymmetric and Public Keys
Practice PT	1.1 1.2 6.3 7.2 7.3 7.4 7.5	1.1.1 [P2] (A,B) 1.2.1 [P2] (A-C,E) 1.2.2 [P2] (A) 1.2.5 [P4] (B) 6.3.1 [P1] (A-M) 7.2.1[P1] (A,F,G +) 7.3.1 [P4] (A,D,G,H,L) 7.4.1 [P1] (A,B,E) 7.5.1 [P1] (A,B) 7.5.2 [P5] (A,B)	The Big Data Dilemma

Unit 4 Practice PT Highlights

Practice PT: The Big Data Dilemma
Students will complete a research project on a current issue or event related to Big Data, security and encryption. Students will need to identify appropriate online resources to learn about the issues and find good artifacts to include with their findings. The written components and audio / visual artifact students will identify are similar to those students will see in the AP Performance Tasks.

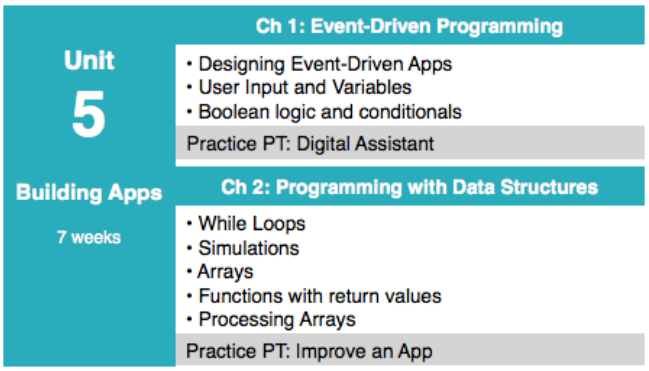
Learning Objectives Addressed:
Data: 3.3.1 [P4]
Internet: 6.1.1 [P3], 6.2.1 [P5], 6.2.2 [P4], 6.3.1 [P1]
Global Impacts: 7.1.1 [P4], 7.3.1 [P4], 7.5.1[P1], 7.5.2 [P5]

Computational Thinking Practices Emphasized:
P1: Connecting Computing
P5: Communicating

rev. April 2016

Unit 5: Building Apps

This unit continues to develop students' ability to program in the JavaScript language, using Code.org's App Lab environment to create a series of small applications (apps) that live on the web, each highlighting a core concept of programming. In this unit students transition to creating event-driven apps. The unit assumes that students have learned the concepts and skills from Unit 3, namely: writing and using functions, using simple repeat loops, being able to read documentation, collaborating, and using the Code Studio environment with App Lab.



The first chapter begins by introducing App Lab's "Design Mode" which allows students to rapidly prototype an app. Again, we want to highlight the enduring understanding that **Computing enables people to use creative development processes to create computational artifacts for creative expression or to solve a problem (1.2)**. As students construct simple apps that respond to user actions and inputs, the lessons progress through some core concepts of programming; Specifically, the lessons cover variables, boolean logic and conditionals, which enforces the understanding that **Programming uses mathematical and logical concepts (5.5).**

The second chapter goes deeper into core programming concepts including looping, arrays, and the **use of models and simulation to develop new insight and knowledge (2.3)**. Students again create a number of small exemplar apps each of which emphasizes a different context in which a program (or app) can be applied to solve a problem. We want to reinforce the idea that **programs are developed, maintained, and used by people for different purposes (5.4)**. Each app also emphasizes a different core concept and skill of programming allowing us to further the connections that **people write programs to execute algorithms (5.2)** and that **programs employ appropriate abstractions (5.3)** (such as list structures) as well as **mathematical and logical concepts (5.5)** to **extend traditional forms of human expression and experience (1.3)**.

The unit also features two practice performance tasks. The first: *Digital Assistant* helps students prepare for certain aspects of the Create Performance task, specifically, creating a video walk through of their program, and explaining an algorithm orally or in writing. The second: *Improve an App* asks students to look back at the apps they've created during the unit and use one as a point of inspiration for creating their own app. This project is designed to highlight the way **programs can be developed for creative expression, to satisfy personal curiosity, to create new knowledge, or to solve problems (5.1)** and is designed to closely mirror the actual Create PT which students will complete in the following unit.

Unit 5 Lessons

Topics	EU	LO [P] (Ek)	Lessons
Chapter 1: Event-Driven Programming			
Event Driven Programming and Apps	1.1 1.2 2.2 5.1 5.2 5.4	1.1.1 [P2] (A, B) 1.2.1 [P2] (A-E) 2.2.1 [P2] (B, C) 5.1.1 [P2] (A-C) 5.1.2 [P2] (J) 5.2.1 [P3] (D, G, H) 5.4.1 [P4] (C, E, F, M)	Introducing Design Mode Building Apps
Variables and Strings	4.1 5.1 5.2 5.3	4.1.1 [P2] (A, C) 5.1.1 [P2] (B) 5.2.1 [P3] (C, F) 5.3.1 [P3] (I)	Controlling Memory with Variables Using Variables in Apps User Input and Strings
Conditionals and Boolean Logic	1.2 1.3 2.2 4.1 5.1 5.3 5.5 7.1	1.2.3 [P2] (A-C) 1.2.4 [P6] (A-D) 1.3.1 [P2] (E) 2.2.3 [P3] (F) 4.1.1 [P2] (A-C, I) 5.1.2 [P2] (A-C) 5.1.3 [P6] (A-F) 5.3.1 [P3] (I) 5.5.1 [P1] (E-G) 7.1.1 [P4] (L-N)	Introduction to Digital Assistants Understanding Program Flow and Logic Conditional Logic
Practice PT	1.3 4.1 5.1 5.5	1.3.1 [P2} (E) 4.1.1 [P2] (A-C, I) 5.1.2 [P2] (A-C) 5.5.1 [P1] (E-G)	**Digital Assistant Project**
Chapter 2: Programming with Data Structures			
Loops and Arrays	2.3 3.1 4.1 5.1 5.2 5.3 5.4 5.5	2.3.1 [P3] (A, C, D) 2.3.2 [P3] (A-F) 3.1.1 [P4] (A) 4.1.1 [P2] (A-D, H) 4.1.2 [P5] (A-G) 5.1.1 [P2] (A, B) 5.1.3 [P6] (A-F) 5.2.1 [P3] (A-F, I-K) 5.3.1 [P3] (A-D, G, K, L) 5.4.1 [P4] (B, C, E-H, K-M) 5.5.1 [P1] (D-J)	While Loops Loops and Simulations Introduction to Arrays Image Scroller with Key Events
Processing Arrays of Data	1.1 1.2 1.3 2.2 4.1 4.2 5.1 5.2 5.3 5.4 5.5	1.1.1 [P2] (B) 1.2.1 [P2] (A-D) 1.2.3 [P2] (A-C) 1.3.1 [P2] (C-E) 2.2.1 [P2] (A-C) 2.2.2 [P3] (A, B) 4.1.1 [P2] (A-I) 4.1.2 [P5] (A-C, G, I) 4.2.4 [P4] (D-F, H) 5.1.1 [P2] (A-E) 5.1.2 [P2] (A-C, J) 5.2.1 [P3] (A-F, I, J) 5.3.1 [P3] (A-G, J-L) 5.4.1 [P4] (A-H, L-N)	Processing Arrays Functions with Return Values Canvas and Arrays in Apps

rev. April 2016

		5.5.1 [P1] (D-J)
Practice PT	1.1 1.2 2.2 4.1 5.1 5.2 5.3 5.4 5.5	1.1.1 [P2] (A, B) 1.2.1 [P2] (A-E) 1.2.2 [P2] (A, B) 1.2.3 [P2] (A-C) 1.2.4 [P6] (A-F) 1.2.5 [P4] (A-D) 2.2.1 [P2] (A-C) 2.2.2 [P3] (A, B) 4.1.1 [P2] (A-I) 4.1.2 [P5] (A-I) 5.1.1 [P2] (A-E) 5.1.2 [P2] (A-J) 5.1.3 [P6] (A-F) 5.2.1 [P3] (A-F, I-K) 5.3.1 [P3] (A-O) 5.4.1 [P4] (C, E-H, J, L-N) 5.5.1 [P1] (A-J)
Improve Your App		

Unit 5 Practice PT Highlights

Practice PT: Improve an App
To conclude their introduction to programming, students will design an app based off of one they have previously worked on in the programming unit. Students will choose the kinds of improvements they wish to make to a past project in order to show their ability to add new abstractions (procedures and functions) and algorithms to an existing program. The project concludes with reflection questions similar to those students will see on the AP Create Performance Task. Students can either complete the project individually or with a partner. Every student will need a collaborative partner with whom they will give and receive feedback.

Learning Objectives Addressed:
Creativity: 1.1.1 [P2], 1.2.1 [P2], 1.2.2 [P2], 1.2.3 [P2], 1.2.4 [P6], 1.3.1 [P2]
Abstraction: 2.2.1 [P2], 2.2.2 [P3]
Algorithms: 4.1.1 [P2], 4.1.2 [P5]
Programming: 5.1.1 [P2], 5.1.2 [P2], 5.1.3 [P6], 5.2.1 [P3], 5.3.1 [P3], 5.4.1 [P4], 5.5.1 [P1]

Computational Practices Emphasized:
P2: Creating Computational Artifacts
P3: Abstracting
P5: Communicating
P6: Collaborating

Unit 6 - Performance Tasks

In Units 1-5 students learned and practiced the skills and content they needed to know in order to succeed on the AP CSP Performance Tasks. Still, a certain level of guidance during the PT development process is not only recommended, but vital. For example, coaching students early on helps them clarify their ideas and/or approaches to the PTs. The only real new instruction in this unit is to shore up understandings for the Explore Performance task related to **finding, evaluating and citing sources (7.5).**

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. In the official submission to the College Board, teachers will attest that all student work is original and that the appropriate amount of class time - 8 hours for *Explore*, 12 hours for *Create* - was provided.

Chapters	EU	LO [P] (Ek)	Lessons / Topics
Create PT Overview	5.1	5.1.1 [P2] (A, B, C) 5.1.2 [P2](A, B, C) 5.1.3 [P6] (B,C)	Planning to do the Create PT Requirements and managing time.
Create PT			**Administration of** ***Create* Performance Task** **12 hours**
Explore PT Overview	7.5	7.5.1 [P1] (A,B) 7.5.2 [P5] (A,B)	Planning to do the Explore PT Research Tips and Tricks Requirements and managing time.
Explore PT			**Administration of** ***Explore* Performance Task** **8 hours**