# Code.org Computer Science Principles

Syllabus and Overview

rev. May 2017

## 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. The course is designed for typical school settings with teachers in classrooms. All teacher and student materials are provided for free online.

## 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, the AP CSP assessment, and the AP framework for professional development. 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.


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

## Curriculum Overview and Goals

Computing affects almost all aspects of modern life and *all* students deserve access to a computing education that prepares them to pursue the wide array of intellectual and career opportunities that computing has made possible. Here is a brief summary of each of the units in the Code.org CSP curriculum.

| | |
|---|---|
| **Unit 1:** The Internet | Learn how the multi-layered systems of the Internet function as you collaboratively solve problems and puzzles about encoding and transmitting data, both 'unplugged' and using Code.org's Internet Simulator. |
| **Unit 2:** Digital Information | Use a variety of digital tools to look at, generate, clean, and manipulate data to explore the relationship between information and data. Create and use visualizations to identify patterns and trends. |
| **Unit 3:** Algorithms and programming | Learn the JavaScript language with turtle programming in Code.org's App Lab. Learn general principles of algorithms and program design that are applicable to any programming language. |
| **Unit 4:** Big Data and Privacy | Research current events around the complex questions related to public policy, law, ethics, and societal impact. Learn the basics of how and why modern encryption works. |
| **Unit 5:** Building Apps | Continue learning how to program in the JavaScript language. Use Code.org's App Lab environment to create a series of applications that live on the web. Each app highlights a core concept of programming. |

This course is not a tour of current events and technologies. Rather, it 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.

rev. May 2017

## 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 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. 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.

## 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.

rev. May 2017

## 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.).

Teachers of this course should be especially interested in creating and nurturing equitable, engaging classrooms. The work of providing an equitable classroom doesn't start or stop with curriculum -- the classroom environment and teaching practices must also be structured such that all learners can access and engage with the material at a level that doesn't advantage some 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.

# Code.org CS Principles Course Snapshot

The chart on the following page is intended to show the big picture of the entire course.

- Each unit is broken into "chapters" - groups of lessons that address a related set of topics. Each chapter ends with some kind of assessment.
- Each chapter shows every lesson title along with suggested pacing guidance
- Note the *Performance Tasks* cluster of lessons at the end. They are not a formal unit of the course because work on the tasks can take place at various points throughout the course. However, you must plan to allocate time toward completing them.

# Code.org Computer Science Principles
## Syllabus and Overview

rev. May 2017

## CS Principles Course Snapshot

### Unit 1 - The Internet

**Ch. 1: Digital Information**

| | |
|---|---|
| wk 1 | Personal Innovations<br>Sending Binary Messages<br>Sending Messages with the Simulator |
| 2 | Number Systems<br>Binary Numbers<br>Sending Numbers |
| 3 | Encoding and Sending Formatted Text<br>Unit 1 Chapter 1 Assessment |

**Ch. 2: Inventing The Internet**

| | |
|---|---|
| | The Internet is for Everyone |
| 4 | The Need for Addressing<br>Routers and Redundancy<br>Packets and Making a Reliable Internet |
| 5 | The Need for DNS<br>HTTP and Abstraction<br>Practice PT - The Internet and Society<br>Unit 1 Chapter 2 Assessment |

### Unit 2 - Digital Information

**Ch. 1: Encoding and Compressing Info**

| | |
|---|---|
| wk 1 | Bytes and File Sizes<br>Text Compression<br>Encoding B&W Images |
| 2 | Encoding Color Images<br>Lossy Compression and File Formats |
| 3 | Encode an Experience<br>Unit 2 Chapter 1 Assessment |

**Ch. 2 - Manipulating and Visualizing Data**

| | |
|---|---|
| 4 | Intro to Data<br>Finding Trends with Visualizations<br>Check Your Assumptions<br>Good and Bad Data Visualizations |
| 5 | Making Data Visualizations<br>Discover a Data Story |
| 6 | Cleaning Data<br>Creating Summary Tables<br>Practice PT - Tell a Data Story<br>Unit 2 Chapter 2 Assessment |

### Unit 3 - Intro to Programming

**Ch. 1 - Programming Languages & Algorithms**

| | |
|---|---|
| wk 1 | The Need For Programming Languages<br>The Need for Algorithms<br>Creativity in Algorithms |
| 2 | Using Simple Commands<br>Creating Functions<br>Functions and Top-Down Design |

### Unit 3 -Intro to Programming (con'd)

| | |
|---|---|
| wk 3 | APIs and Function Parameters<br>Creating functions with Parameters<br>Looping and Random Numbers |
| 4 | Practice PT - Design a Digital Scene<br>Unit 3 Chapter 1 Assessment |

### Unit 4 -Big Data and Privacy

**Ch. 1: The World of Big Data and Encryption**

| | |
|---|---|
| wk 1 | What is Big Data?<br>Rapid Research - Data Innovations<br>Identifying People with Data |
| 2 | The Cost of Free<br>Simple Encryption |
| 3 | Encryption with Keys and Passwords<br>Public Key Crypto<br>Rapid Research - Cybercrime |
| 4 | Practice PT - Big Data and Security Dilemmas<br>Unit 4 Chapter 1 Assessment |

### Unit 5 -Building Apps

**Ch. 1: Event-Driven Programming**

| | |
|---|---|
| wk 1 | Buttons and Events<br>Multi-screen Apps<br>Building an App: Multi-Screen App |
| 2 | Controlling Memory with Variables<br>Building an App: Clicker Game<br>Unit 5 Assessment 1<br>User Input and Strings |
| 3 | "If" Statements Unplugged<br>Boolean Expressions and "If" Statements |
| 4 | "if-else-if" and Conditional Logic<br>Building an App: Color Sleuth<br>Unit 5 Assessment 2 |

**Ch. 2: Programming with Data Structures**

| | |
|---|---|
| 5 | While Loops<br>Loops and Simulations<br>Introduction to Arrays |
| 6 | Building an App: Image Scroller<br>Unit 5 Assessment 3<br>Processing Arrays<br>Functions with Return Values |
| 7 | Building an App: Canvas Painter<br>Unit 5 Assessment 4<br>Practice PT: Create<br>Unit 5 Assessment 5 - AP Pseudocode Practice |

### Performance Tasks

| | |
|---|---|
| 1 hr | Tech Setup (Can be completed at any time) |
| 10 hrs | Explore prep (Can be completed after Unit 4)<br>Explore PT (8 class hours) |
| 14 hrs | Create Prep(Can be completed after Unit 5 Chapter 1)<br>Create PT (12 class hours) |

While the layout of units appears to be modular, the units of study are 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.

## Technical and Material 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 modern web browser. For more details on the technical requirements, please visit: code.org/educate/it

While the course features many "unplugged" activities designed to be completed away from the computer, daily access to a computer is essential for every student. It is not required that students have access to internet-connected computers at home to teach this course. But because almost all of the materials are online, it is certainly an advantage. PDFs of handouts, worksheets and readings are available on the course website.

### Computational Tools, Resources and Materials Provided

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** 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

## Required Materials / Supplies:

One potentially significant cost to consider is printing. Many lessons have handouts that are designed to guide students through activities. While it is not required that all of these handouts be printed, many were designed with print in mind and we highly recommend their use.

Beyond printing, a some lessons call for typical classroom supplies and manipulatives such as:

- Student Journal
- Poster paper
- Markers
- Post-it notes
- Graph paper, etc.

There is a complete materials list in the curriculum front matter. Besides printing costs, all other materials are highly suggested, and are low cost (cups, string, playing cards, etc.).

## 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)

# 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.

rev. May 2017

## 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 in and of themselves.** The *Big Ideas* 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.

# Unit Overviews

On the pages that follow are more in-depth prose descriptions of each unit of study which explain the topics covered, what students will be doing and how the lessons *build toward the Enduring Understandings* in the framework.

We also show how each lesson maps to the CSP Framework for the *Enduring Understandings* as well as the specific *Learning Objective* and *Essential Knowledge* statements. For example you might see a table like:

| | |
|---|---|
| **Unit 1: The Internet** | ← **Unit Title** |
| Ch 1: Representing Information | ← Chapter Title |

• • •

| | | |
|---|---|---|
| **2. Sending Binary Messages** | | ← **Lesson Name** |
| 2.1 | 2.1.1[P3] (ABCE) | ← EU LO [P] (EKs) |
| 2.1 | 2.1.2[P5] (DEF) | \| \| \| ↳ *EK = Essential Knowledge statements* |
| 3.3 | 3.3.1[P4] (AB) | \| \| ↳ *[P] = Computational Thinking Practice* |
| | | \| ↳ *LO = Learning Objective* |
| | | ↳ *EU = Enduring Understanding* |

*See the CSP Framework document for listings of all the Enduring Understandings.*

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.

rev. May 2017

# 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: 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

rev. May 2017

CSP Framework mappings

## Unit 1: The Internet

### Ch 1: Representing and Transmitting Information

**1. Personal Innovations**

| | |
|---|---|
| 7.1 | 7.1.1[P4] (ABCDEFGHIJKLMNO) |
| 7.2 | 7.2.1[P1] (ABCG) |
| 7.3 | 7.3.1[P4] (ABCDEGHIJKLMNO) |
| 7.4 | 7.4.1[P1] (ABCD) |

**2. Sending Binary Messages**

| | |
|---|---|
| 2.1 | 2.1.1[P3] (ABCE) |
| 2.1 | 2.1.2[P5] (DEF) |
| 3.3 | 3.3.1[P4] (AB) |

**3. Sending Binary Messages with the Internet Simulator**

| | |
|---|---|
| 2.1 | 2.1.1[P3] (ABCE) |
| 2.1 | 2.1.2[P5] (DEF) |
| 2.3 | 2.3.1[P3] (ABCD) |
| 2.3 | 2.3.2[P3] (A) |
| 3.1 | 3.1.3[P5] (A) |
| 3.3 | 3.3.1[P4] (AB) |
| 6.1 | 6.1.1[P3] (AC) |
| 6.2 | 6.2.2[P4] (DJK) |

**4. Number Systems**

| | |
|---|---|
| 2.1 | 2.1.1[P3] (ABCDE) |
| 2.3 | 2.3.1[P3] (AB) |
| 2.3 | 2.3.2[P3] (ABC) |

**5. Binary Numbers**

| | |
|---|---|
| 2.1 | 2.1.1[P3] (ABCDEG) |
| 2.1 | 2.1.2[P5] (ABCDEF) |
| 2.3 | 2.3.1[P3] (ABCD) |
| 2.3 | 2.3.2[P3] (ABCDE) |
| 3.1 | 3.1.3[P5] (AB) |

**6. Sending Numbers**

| | |
|---|---|
| 2.1 | 2.1.1[P3] (ABCDEG) |
| 2.1 | 2.1.2[P5] (ABCDEF) |
| 2.3 | 2.3.1[P3] (ABCD) |
| 2.3 | 2.3.2[P3] (ABCDE) |
| 3.1 | 3.1.3[P5] (ABDE) |
| 6.2 | 6.2.2[P4] (DGH) |

**7. Encoding and Sending Formatted Text**

| | |
|---|---|
| 2.1 | 2.1.1[P3] (ABCDE) |
| 2.1 | 2.1.2[P5] (BDEF) |
| 2.2 | 2.2.1[P2] (AB) |
| 3.1 | 3.1.3[P5] (AE) |
| 3.3 | 3.3.1[P4] (AB) |
| 6.1 | 6.1.1[P3] (ABCD) |
| 6.2 | 6.2.2[P4] (DFGH) |

### Ch. 2: Inventing the Internet

**8. The Internet Is for Everyone**

| | |
|---|---|
| 6.1 | 6.1.1[P3] (BCE) |
| 6.2 | 6.2.2[P4] (E) |
| 7.3 | 7.3.1[P4] (ADEGL) |
| 7.4 | 7.4.1[P1] (CDE) |

**9. The Need for Addressing**

| | |
|---|---|
| 6.1 | 6.1.1[P3] (CDFH) |
| 6.2 | 6.2.1[P5] (C) |
| 6.2 | 6.2.2[P4] (D) |
| 6.3 | 6.3.1[P1] (A) |

**10. Routers and Redundancy**

| | |
|---|---|
| 3.3 | 3.3.1[P4] (AF) |
| 6.1 | 6.1.1[P3] (BCE) |
| 6.2 | 6.2.1[P5] (AD) |
| 6.2 | 6.2.2[P4] (B) |

**11. Packets and Making a Reliable Internet**

| | |
|---|---|
| 6.2 | 6.2.1[P5] (AD) |
| 6.2 | 6.2.2[P4] (ABG) |

**12. The Need for DNS**

| | |
|---|---|
| 6.1 | 6.1.1[P3] (G) |
| 6.2 | 6.2.1[P5] (B) |
| 6.2 | 6.2.2[P4] (CD) |

**13. HTTP and Abstraction on the Internet**

| | |
|---|---|
| 6.1 | 6.1.1[P3] (I) |
| 6.2 | 6.2.2[P4] (H) |

**14. Practice PT - The Internet and Society**

| | |
|---|---|
| 6.3 | 6.3.1[P1] (AB) |
| 7.1 | 7.1.1[P4] (ABCDHIJKMO) |
| 7.4 | 7.4.1[P1] (ABDE) |

rev. May 2017

# Unit 2: Digital Information

This unit further explores the ways that digital information is encoded, represented and manipulated. In this unit 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" - 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: Practice PT Highlights

**Practice PT: Encode an Experience**
Students invent a binary encoding (file format) for a real life 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. 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

**Practice PT: Tell a Data Story**
This small project culminates a series of lessons in which students must use digital tools to collaboratively investigate a data set to discover possible connections and trends. Students must produce a visual explanation of their findings in the data and write a short piece about what the data shows.

**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

rev. May 2017

CSP Framework mappings

## Unit 2: Digital Information

### Ch. 1: Encoding and Compressing Complex Information

**1. Bytes and File Sizes**

| | |
|---|---|
| 2.1 | 2.1.1[P3] (BC) |
| 2.1 | 2.1.2[P5] (BCEF) |
| 3.3 | 3.3.1[P4] (G) |

**2. Text Compression**

| | | | |
|---|---|---|---|
| 2.1 | 2.1.1[P3] (ABC) | 3.3 | 3.3.1[P4] (A) |
| 2.2 | 2.2.1[P2] (B) | 4.2 | 4.2.1[P1] (ABCD) |
| 3.1 | 3.1.1[P4] (ADE) | 4.2 | 4.2.2[P1] (BC) |
| 3.1 | 3.1.2[P6] (ABCD) | 4.2 | 4.2.3[P1] (ABC) |
| 3.1 | 3.1.3[P5] (AE) | 4.2 | 4.2.4[P4] (ACD) |

**3. Encoding B&W Images**

| | | | |
|---|---|---|---|
| 1.1 | 1.1.1[P2] (AB) | 3.1 | 3.1.1[P4] (ADE) |
| 1.2 | 1.2.1[P2] (A) | 3.1 | 3.1.2[P6] (ABCD) |
| 1.3 | 1.3.1[P2] (C) | 3.1 | 3.1.3[P5] (AE) |
| 2.1 | 2.1.1[P3] (ABC) | 3.2 | 3.2.1[P1] (GHI) |
| 2.1 | 2.1.2[P5] (AB) | 3.3 | 3.3.1[P4] (A) |
| 2.3 | 2.3.1[P3] (ABCD) | | |

**4. Encoding Color Images**

| | | | |
|---|---|---|---|
| 1.1 | 1.1.1[P2] (AB) | 2.3 | 2.3.1[P3] (ABCD) |
| 1.2 | 1.2.1[P2] (A) | 3.1 | 3.1.1[P4] (ADE) |
| 1.3 | 1.3.1[P2] (C) | 3.1 | 3.1.2[P6] (ABCD) |
| 2.1 | 2.1.1[P3] (ABCDF) | 3.1 | 3.1.3[P5] (AE) |
| 2.1 | 2.1.2[P5] (DEF) | 3.2 | 3.2.1[P1] (GHI) |
| 2.2 | 2.2.1[P2] (AB) | | |

**5. Lossy Compression and File Formats**

| | |
|---|---|
| 3.3 | 3.3.1[P4] (ACDEG) |

**6. Practice PT - Encode an Experience**

| | |
|---|---|
| 2.1 | 2.1.1[P3] (ABCDE) |
| 2.1 | 2.1.2[P5] (ABDF) |
| 2.2 | 2.2.1[P2] (AB) |

### Ch. 2: Manipulating and Visualizing Data

**7. Introduction to Data**

| | |
|---|---|
| 2.2 | 2.2.3[P3] (E) |
| 3.1 | 3.1.1[P4] (BCDE) |
| 3.1 | 3.1.3[P5] (D) |
| 3.2 | 3.2.1[P1] (ABC) |
| 7.3 | 7.3.1[P4] (HJ) |

**8. Finding Trends with Visualizations**

| | |
|---|---|
| 3.1 | 3.1.1[P4] (ABE) |
| 3.1 | 3.1.2[P6] (ABCDEF) |
| 3.1 | 3.1.3[P5] (ABCE) |
| 3.2 | 3.2.1[P1] (ABCDE) |

**9. Check Your Assumptions**

| | |
|---|---|
| 3.1 | 3.1.1[P4] (E) |
| 3.1 | 3.1.2[P6] (ABCDF) |
| 3.2 | 3.2.1[P1] (ABC) |
| 7.4 | 7.4.1[P1] (ABCD) |

**10. Good and Bad Data Visualizations**

| | |
|---|---|
| 1.2 | 1.2.5[P4] (ABCD) |
| 3.1 | 3.1.1[P4] (DE) |
| 3.1 | 3.1.2[P6] (ABCDF) |
| 3.1 | 3.1.3[P5] (ABCDE) |

**11. Making Data Visualizations**

| | |
|---|---|
| 1.2 | 1.2.5[P4] (ABCD) |
| 3.1 | 3.1.1[P4] (DE) |
| 3.1 | 3.1.2[P6] (ABCDF) |
| 3.1 | 3.1.3[P5] (ABCDE) |

**12. Discover a Data Story**

| | | | |
|---|---|---|---|
| 1.1 | 1.1.1[P2] (AB) | 1.3 | 1.3.1[P2] (E) |
| 1.2 | 1.2.1[P2] (ABC) | 3.1 | 3.1.1[P4] (DE) |
| 1.2 | 1.2.2[P2] (A) | 3.1 | 3.1.2[P6] (ABCDF) |
| 1.2 | 1.2.4[P6] (A) | 3.1 | 3.1.3[P5] (ABCD) |
| 1.2 | 1.2.5[P4] (D) | | |

**13. Cleaning Data**

| | | | |
|---|---|---|---|
| 1.2 | 1.2.1[P2] (ABCDE) | 3.1 | 3.1.2[P6] (ABCDEF) |
| 1.2 | 1.2.4[P6] (AB) | 3.2 | 3.2.1[P1] (ABCDEF) |
| 3.1 | 3.1.1[P4] (AB) | 3.2 | 3.2.2[P3] (BCG) |
| | | 7.1 | 7.1.2[P4] (CD) |

**14. Creating Summary Tables**

| | | | |
|---|---|---|---|
| 1.1 | 1.1.1[P2] (AB) | 3.1 | 3.1.2[P6] (DEF) |
| 1.2 | 1.2.1[P2] (ABE) | 3.1 | 3.1.3[P5] (ABCD) |
| 1.2 | 1.2.4[P6] (AB) | 3.2 | 3.2.1[P1] (CF) |
| 3.1 | 3.1.1[P4] (ABCDE) | | |

**15. Practice PT - Tell a Data Story**

| | |
|---|---|
| 1.2 | 1.2.1[P2] (ABCE) |
| 1.2 | 1.2.2[P2] (AB) |
| 1.2 | 1.2.5[P4] (ABCD) |
| 3.1 | 3.1.3[P5] (ABCD) |
| 7.3 | 7.3.1[P4] (J) |

# 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.

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: Practice PT Highlight

**Practice PT: Design a Digital Scene**
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. Students reflect on their experience in a way that is similar to the *Create* performance task. A heavy programming 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

rev. May 2017

CSP Framework mappings

## Unit 3: Algorithms and Programming

### Ch. 1: Programming Languages and Algorithms

**1. The Need for Programming Languages**
- 4.1 4.1.2[P5] (ABCFI)
- 5.2 5.2.1[P3] (E)

**2. The Need for Algorithms**
- 4.1 4.1.1[P2] (ABCDHI)
- 4.1 4.1.2[P5] (ABG)
- 4.2 4.2.4[P4] (F)
- 5.4 5.4.1[P4] (FI)

**3. Creativity in Algorithms**
- 2.2 2.2.3[P3] (C)
- 4.1 4.1.1[P2] (ABCDEFGHI)
- 5.2 5.2.1[P3] (ABDEJ)

**4. Using Simple Commands**
- 5.1 5.1.2[P2] (AI)
- 5.1 5.1.3[P6] (BCDF)
- 5.2 5.2.1[P3] (AB)
- 5.4 5.4.1[P4] (ADEI)

**5. Creating Functions**
- 2.2 2.2.1[P2] (AB)
- 2.2 2.2.2[P3] (AB)
- 5.3 5.3.1[P3] (ABCL)
- 5.4 5.4.1[P4] (ABCD)

**6. Functions and Top-Down Design**
- 2.2 2.2.1[P2] (A)
- 2.2 2.2.2[P3] (AB)
- 2.2 2.2.3[P3] (AD)
- 5.1 5.1.2[P2] (ABC)
- 5.1 5.1.3[P6] (ABCDEF)
- 5.3 5.3.1[P3] (ABCD)

**7. APIs and Using Functions with Parameters**
- 2.2 2.2.2[P3] (AB)
- 2.2 2.2.3[P3] (AB)
- 5.1 5.1.2[P2] (DEF)
- 5.3 5.3.1[P3] (DEFGMNO)

**8. Creating Functions with Parameters**
- 2.2 2.2.1[P2] (C)
- 2.2 2.2.2[P3] (AB)
- 5.3 5.3.1[P3] (ACDFGL)
- 5.4 5.4.1[P4] (CDEFGHIJK)

**9. Looping and Random Numbers**
- 4.1 4.1.1[P2] (D)
- 5.1 5.1.2[P2] (BC)
- 5.3 5.3.1[P3] (ACDFGL)
- 5.4 5.4.1[P4] (CDEFGHIJK)

**10. Practice PT - Design a Digital Scene**
- 2.2 2.2.1[P2] (C)
- 2.2 2.2.2[P3] (AB)
- 2.2 2.2.3[P3] (AB)
- 4.1 4.1.1[P2] (D)
- 5.1 5.1.2[P2] (BC)
- 5.1 5.1.3[P6] (ABCDEF)
- 5.3 5.3.1[P3] (ACDFGL)
- 5.4 5.4.1[P4] (CDEFGHIJK)

# Unit 4: Big Data and Privacy

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 Practice PT Highlights

**Practice PT: Big Data and Cybersecurity Dilemmas**
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

CSP Framework mappings

## Unit 4: Big Data and Privacy

### Ch. 1: The World of Big Data and Encryption

**1. What is Big Data?**

| | |
|---|---|
| 3.2 | 3.2.2[P3] (ABCDEFGH) |
| 7.2 | 7.2.1[P1] (ABCDEFG) |
| 7.5 | 7.5.2[P5] (AB) |

**2. Rapid Research - Data Innovations**

| | |
|---|---|
| 1.2 | 1.2.3[P2] (C) |
| 1.2 | 1.2.5[P4] (AD) |
| 3.2 | 3.2.2[P3] (ABCDEFGH) |
| 7.1 | 7.1.1[P4] (DEFGHIJKLMNO) |
| 7.4 | 7.4.1[P1] (ABCDE) |
| 7.5 | 7.5.2[P5] (AB) |

**3. Identifying People With Data**

| | |
|---|---|
| 3.2 | 3.2.2[P3] (D) |
| 3.3 | 3.3.1[P4] (BF) |
| 7.3 | 7.3.1[P4] (GJKL) |

**4. The Cost of Free**

| | |
|---|---|
| 3.3 | 3.3.1[P4] (ABF) |
| 7.3 | 7.3.1[P4] (AGHJKLMN) |

**5. Simple Encryption**

| | |
|---|---|
| 1.2 | 1.2.2[P2] (A) |
| 3.3 | 3.3.1[P4] (BF) |
| 6.3 | 6.3.1[P1] (CHIK) |
| 7.3 | 7.3.1[P4] (G) |

**6. Encryption with Keys and Passwords**

| | |
|---|---|
| 2.3 | 2.3.2[P3] (A) |
| 3.1 | 3.1.1[P4] (A) |
| 4.2 | 4.2.1[P1] (ABCD) |
| 6.3 | 6.3.1[P1] (CHIJK) |

**7. Public Key Cryptography**

| | |
|---|---|
| 4.2 | 4.2.1[P1] (ABCD) |
| 4.2 | 4.2.2[P1] (A) |
| 4.2 | 4.2.3[P1] (A) |
| 4.2 | 4.2.4[P4] (ABC) |
| 6.3 | 6.3.1[P1] (HIL) |

**8. Rapid Research - Cybercrime**

| | |
|---|---|
| 6.2 | 6.2.2[P4] (H) |
| 6.3 | 6.3.1[P1] (CDEFGH) |
| 7.3 | 7.3.1[P4] (G) |

**9. Practice PT - Big Data and Cybersecurity Dilemmas**

| | |
|---|---|
| 1.1 | 1.1.1[P2] (AB) |
| 1.2 | 1.2.1[P2] (ABCE) |
| 1.2 | 1.2.2[P2] (A) |
| 1.2 | 1.2.5[P4] (B) |
| 6.3 | 6.3.1[P1] (ABCDEFGHIJKLM) |
| 7.3 | 7.3.1[P4] (ADGHL) |
| 7.4 | 7.4.1[P1] (ABE) |

rev. May 2017

# 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)**. 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 Practice PT: *Create Your Own 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: Practice PT Highlight

**Practice PT: Create Your Own App**
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.

**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

rev. May 2017

CSP Framework mappings

## Unit 5: Building Apps

### Ch. 1: Event-Driven Programming

**1. Introduction to Event-Driven Programming**

| | | | |
|---|---|---|---|
| 1.1 | 1.1.1[P2] (B) | 5.1 | 5.1.2[P2] (J) |
| 1.2 | 1.2.1[P2] (ABE) | 5.4 | 5.4.1[P4] (ACDEFHKM) |
| 5.1 | 5.1.1[P2] (AB) | | |

**2. Multi-Screen Apps**

| | | | |
|---|---|---|---|
| 1.1 | 1.1.1[P2] (B) | 5.1 | 5.1.1[P2] (BC) |
| 1.2 | 1.2.1[P2] (A) | 5.1 | 5.1.2[P2] (J) |
| 1.2 | 1.2.3[P2] (A) | 5.1 | 5.1.3[P6] (D) |
| 1.2 | 1.2.4[P6] (ADE) | 5.4 | 5.4.1[P4] (CEFM) |

**3. Building an App: Multi-Screen App**

| | | | |
|---|---|---|---|
| 1.1 | 1.1.1[P2] (AB) | 5.1 | 5.1.2[P2] (ABCDEFJ) |
| 1.2 | 1.2.1[P2] (ABCE) | 5.1 | 5.1.3[P6] (BCD) |
| 5.1 | 5.1.1[P2] (AB) | 5.4 | 5.4.1[P4] (EFGL) |

**4. Controlling Memory with Variables**

| | | | |
|---|---|---|---|
| 5.2 | 5.2.1[P3] (CF) | | |

**5. User Input and Strings**

| | | | |
|---|---|---|---|
| 5.1 | 5.1.1[P2] (B) | | |
| 5.3 | 5.3.1[P3] (L) | | |

**6. 'if-statements unplugged**

| | | | |
|---|---|---|---|
| 4.1 | 4.1.1[P2] (AC) | | |
| 4.1 | 4.1.2[P5] (ABG) | | |
| 5.2 | 5.2.1[P3] (ABD) | | |

**7. Boolean Expressions and 'if' Statements**

| | | | |
|---|---|---|---|
| 4.1 | 4.1.1[P2] (ABC) | | |
| 5.5 | 5.5.1[P1] (EG) | | |

**8. 'If-else-if' and Conditional Logic**

| | | | |
|---|---|---|---|
| 4.1 | 4.1.1[P2] (C) | | |
| 5.1 | 5.1.2[P2] (J) | | |
| 5.5 | 5.5.1[P1] (EG) | | |

**9. Building an App: Color Sleuth**

| | | | |
|---|---|---|---|
| 1.1 | 1.1.1[P2] (B) | 4.1 | 4.1.2[P5] (G) |
| 1.2 | 1.2.4[P6] (ACDF) | 5.1 | 5.1.2[P2] (A) |
| 3.1 | 3.1.2[P6] (B) | 5.1 | 5.1.3[P6] (ABCDEF) |
| 4.1 | 4.1.1[P2] (AC) | 5.5 | 5.5.1[P1] (DE) |

### Ch. 2: Programming with Data Structures

**10. While Loops**

| | | | |
|---|---|---|---|
| 3.1 | 3.1.1[P4] (A) | 5.2 | 5.2.1[P3] (ABCDIJK) |
| 4.1 | 4.1.1[P2] (ABCDH) | 5.4 | 5.4.1[P4] (BEFGHKLM) |
| 4.1 | 4.1.2[P5] (ABCDEFG) | 5.5 | 5.5.1[P1] (DEFG) |

**11. Loops and Simulations**

| | | | |
|---|---|---|---|
| 2.3 | 2.3.1[P3] (ACD) | 4.1 | 4.1.2[P5] (AB) |
| 2.3 | 2.3.2[P3] (ABDEFGH) | 5.1 | 5.1.1[P2] (A) |

**12. Introduction to Arrays**

| | | | |
|---|---|---|---|
| 1.1 | 1.1.1[P2] (B) | 5.3 | 5.3.1[P3] (ABCKL) |
| 5.1 | 5.1.2[P2] (A) | 5.5 | 5.5.1[P1] (HIJ) |

**13. Building an App: Image Scroller**

| | | | |
|---|---|---|---|
| 5.1 | 5.1.1[P2] (A) | 5.4 | 5.4.1[P4] (BCGM) |
| 5.2 | 5.2.1[P3] (EFIJK) | 5.5 | 5.5.1[P1] (HIJ) |
| 5.3 | 5.3.1[P3] (ABCDGKL) | | |

**14. Processing Arrays**

| | | | |
|---|---|---|---|
| 1.2 | 1.2.3[P2] (A) | 5.1 | 5.1.2[P2] (AB) |
| 4.1 | 4.1.1[P2] (ABCD) | 5.3 | 5.3.1[P3] (KL) |
| 4.2 | 4.2.4[P4] (DEFH) | 5.5 | 5.5.1[P1] (EJ) |

**15. Functions with Return Values**

| | | | |
|---|---|---|---|
| 1.1 | 1.1.1[P2] (B) | 4.1 | 4.1.1[P2] (ABCDEF) |
| 1.2 | 1.2.3[P2] (AC) | 5.1 | 5.1.2[P2] (ABCJ) |
| 2.2 | 2.2.1[P2] (AC) | 5.3 | 5.3.1[P3] (ABCDEFGKL) |
| 2.2 | 2.2.2[P3] (AB) | 5.5 | 5.5.1[P1] (EJ) |

**16. Building an App: Canvas Painter**

| | | | |
|---|---|---|---|
| 1.2 | 1.2.1[P2] (ABCD) | 5.1 | 5.1.1[P2] (ABCDE) |
| 1.3 | 1.3.1[P2] (CDE) | 5.1 | 5.1.2[P2] (ABCJ) |
| 2.2 | 2.2.1[P2] (ABC) | 5.3 | 5.3.1[P3] (ABCDEFGJKL) |
| 2.2 | 2.2.2[P3] (AB) | 5.4 | 5.4.1[P4] (ABCDEFGHLMN) |
| 4.1 | 4.1.1[P2] (ABCD) | 5.5 | 5.5.1[P1] (DEFGHIJ) |
| 4.1 | 4.1.2[P5] (ABCGI) | | |

**17. Practice PT - Create Your Own App**

| | | | |
|---|---|---|---|
| 1.1 | 1.1.1[P2] (AB) | 4.1 | 4.1.1[P2] (ABCDEFGHI) |
| 1.2 | 1.2.1[P2] (ABCDE) | 4.1 | 4.1.2[P5] (ABCDEFGHI) |
| 1.2 | 1.2.2[P2] (AB) | 5.1 | 5.1.1[P2] (ABCDE) |
| 1.2 | 1.2.3[P2] (ABC) | 5.1 | 5.1.2[P2] (ABCDEFGHIJ) |
| 1.2 | 1.2.4[P6] (ABCDEF) | 5.1 | 5.1.3[P6] (ABCDEF) |
| 2.2 | 2.2.1[P2] (ABC) | 5.4 | 5.4.1[P4] (CEFGHJLMN) |
| 2.2 | 2.2.2[P3] (AB) | 5.5 | 5.5.1[P1] (ABCDEFGHIJ) |

# Performance Tasks

In Units 1-5 students learn the content and practice the skills they need 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 especially related to **finding, evaluating and citing sources (7.5).**

The curriculum provides a few lessons to help instructors and students prepare and make a plan for completing the AP Performance Tasks. 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.

Please note that instructors are required to provide the appropriate amount of class time for students to complete each Performance Task - **8 hours for *Explore*, 12 hours for *Create***. The hours are the minimum amount of class time required, and do not need to be contiguous. In the official submission to the College Board, teachers will attest that all student work is original and that the required amount of time was provided.

## Performance Tasks

### AP Tech Setup

**1.Digital Tools and the AP Digital Portfolio**

| | | | |
|---|---|---|---|
| 1.1 | 1.1.1[P2] (B) | 5.1 | 5.1.2[P2] (J) |
| 1.2 | 1.2.1[P2] (ABE) | 5.4 | 5.4.1[P4] (ACDEFHKM) |
| 5.1 | 5.1.1[P2] (AB) | | |

### Explore Task

**1.Explore PT Prep - Reviewing the Task**

| | | | |
|---|---|---|---|
| 1.1 | 1.1.1[P2](AB) | 7.5 | 7.5.1[P1](ABC) |
| 1.2 | 1.2.1[P2](ABCDE) | 7.5 | 7.5.2[P5](AB) |
| 1.2 | 1.2.3[P2](ABC) | | |

**2. Explore PT Prep - Making a Plan**

| | | | |
|---|---|---|---|
| 1.1 | 1.1.1[P2](AB) | 7.5 | 7.5.1[P1](C) |
| 1.2 | 1.2.1[P2](ABCDE) | | |

**3. Explore PT - Complete the Task (8 hours)**

| | | | |
|---|---|---|---|
| 1.2 | 1.2.1[P2] (ABCDE) | 3.3 | 3.3.1[P4] (ABCDEFGHI) |
| 1.2 | 1.2.2[P2] (AB) | 7.1 | 7.1.1[P4] (A-O) |
| 1.2 | 1.2.3[P2] (ABC) | 7.2 | 7.2.1[P1] (A-G) |
| 1.2 | 1.2.5[P4] (ABCD) | 7.3 | 7.3.1[P4] (A-Q) |
| 3.1 | 3.1.3[P5] (ABCDE) | 7.4 | 7.4.1[P1] (ABCDE) |
| 3.2 | 3.2.1[P1] (A-I) | 7.5 | 7.5.1[P1] (ABC) |

### Create Task

**1.Create PT Prep - Reviewing the Task**

| | | | |
|---|---|---|---|
| 1.1 | 1.1.1[P2] (B) | 5.1 | 5.1.2[P2] (A) |
| 1.2 | 1.2.4[P2] (ABCDEF) | 5.1 | 5.1.3[P6] (ABCDEF) |

**2. Create PT Prep - Making a Plan**

| | | | |
|---|---|---|---|
| 1.2 | 1.2.1[P2] (ABE) | 7.5 | 7.5.1[P1] (ABC) |
| 1.2 | 1.2.4[P6](CDEF) | 7.5 | 7.5.2[P5] (AB) |

**3. Create PT - Complete the Task (12 hours)**

| | | | |
|---|---|---|---|
| 1.2 | 1.2.1[P2] (ABCDE) | 5.1 | 5.1.1[P2] (ABCDEF) |
| 1.2 | 1.2.2[P2] (AB) | 5.1 | 5.1.2[P2] (A-J) |
| 1.2 | 1.2.3[P2] (ABC) | 5.1 | 5.1.3[P6] (ABCDEF) |
| 1.2 | 1.2.4[P6] (ABCDEF) | 5.2 | 5.2.1[P3] (A-K) |
| 2.2 | 2.2.1[P2] (ABC) | 5.3 | 5.3.1[P3] (A-O) |
| 2.2 | 2.2.2[P3] (AB) | 5.4 | 5.4.1[P4] (A-N) |
| 4.1 | 4.1.1[P2] (A-I) | 5.5 | 5.5.1[P1] (A-J) |
| 4.1 | 4.1.2[P5] (A-I) | | |