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. All teacher and student materials are provided for free online and can be accessed at code.org/csp.

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.

AP At-a-Glance

The curriculum is divided into roughly 120 daily lesson plans which comprise 10 units of study. More detailed information about each unit can be found later in this syllabus.

Unit 1 Digital Information	Explore how computers store complex information like numbers, text, images and sound and debate the impacts of digitizing information.
Unit 2 The Internet	Learn about how the Internet works and discuss its impacts on politics, culture, and the economy.
Unit 3 Intro to App Design	Design your first app while learning both fundamental programming concepts and collaborative software development processes.
Unit 4 Variables, Conditionals, and Functions	Expand the types of apps you can create by adding the ability to store information, make decisions, and better organize code.
Unit 5 Lists, Loops, and Traversals	Build apps that use large amounts of information and pull in data from the web to create a wider variety of apps.
Unit 6 Algorithms	Design and analyze algorithms to understand how they work and why some are considered better than others.
Unit 7 Parameters, Return, and Libraries	Learn how to design clean and reusable code that you can share with a single classmate or the entire world.
Unit 8 Create PT Prep	Practice and complete the Create Performance Task (PT).
Unit 9 Data	Explore and visualize datasets from a wide variety of topics as you hunt for patterns and try to learn more about the world around you.
Unit 10 Cybersecurity and Global Impacts	Research and debate current events at the intersection of data, public policy, law, ethics, and societal impact.

Our Vision

Code.org's vision is that every student in every school should have the opportunity to learn computer science (code.org/about). Our curriculum is designed so that an empowered teacher can lead a diverse group of students through experiences that are supportive, equitable, engaging, and lead to valuable learning (code.org/educate/curriculum/values).

Historically this vision has contrasted sharply with reality. Until recently, most schools did not offer computer science at all, and what classes there were notoriously lacked in diversity. Additionally, many students found these classes unengaging, intimidating, or simply disconnected from their lived experiences with technology. Thanks to efforts by many organizations and individuals, this world is beginning to change: many more schools now offer computer science courses; more diverse students take those courses; and more engaging, relevant, and equitable pedagogy has become the established norm. Even so, there is much work still to be done. This course is designed to continue this momentum as the collective CS education community moves towards this vision of an equitable CS education system.

How We Support Our Vision

Many aspects of Code.org's CS Principles curriculum are designed to bring about the eventual change we aim to see more broadly in CS education. Some of the most significant features are listed below.

Free and open: We make our curriculum, videos, and tools free and open for anyone to adopt.

Prioritize New-to-CS Teachers: Historically only a few schools could hire trained computer scientists as teachers, which severely limited which schools could offer a CS course. Reaching all schools has meant developing our CS Principles course with the understanding that most of our teachers are new-to-CS and prioritizing their needs. As such, our course includes some distinctive features.

- Comprehensive lesson plans and resources designed to ensure new-to-CS teachers have everything they need to implement the course
- Clear and consistent pedagogy to help new-to-CS teachers develop best practices as CS teachers
- High-quality videos that help teachers introduce and explain CS concepts
- An associated professional learning program that pays particular attention to the needs of new-to-CS teachers

Equitable Pedagogy: Our curriculum is designed to promote an equitable classroom environment for all students, with particular attention paid to the experiences of historically excluded groups, most notably young women and students from underrepresented minorities in computing. Drawing from extensive feedback from our classrooms, as well as CS education research, our course includes many features designed to support and prioritize these students:

- Pedagogy that develops a collaborative and supportive classroom environment
- Specific attention paid to language demands of our lessons
- Projects and activities that highlight a variety of applications of computing and frequently ask students to incorporate their own backgrounds and interests.
- A sequencing of topics that intentionally delays the introduction of programming (the CS topic with which privileged groups are most likely to have prior experience)
- Curriculum videos that feature a cast of diverse role models in terms of race, gender, and profession who empower our diverse students to "see themselves" as part of the world of computing
- A professional learning program that highlights these features and helps teachers reflect on how best to implement them within their own classroom

Join Us in this Vision

We think our vision is audacious and deeply motivating. If you feel the same, the best way to join us in this vision is to teach this course! We know that for many teachers this represents a significant undertaking, and we have aimed to do our best to help share the load. Based on the feedback of many teachers we know it will be a challenging, but ultimately gratifying experience. Code.org is here to support you, and we look forward to your feedback so that we can continue to make CS Principles an even better experience for our students and teachers.

Provided Materials

The curriculum provides a comprehensive set of resources for the teacher, including detailed minute-by-minute lesson plans for every day of instruction, engaging activities and projects, formative and summative assessments, 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 and help provide a unified experience. Together, these resources typically allow the teacher to act in the role of facilitator and coach when addressing unfamiliar material. In instances when the teacher acts as the primary source of information, generous supports are provided.

All resources below can be accessed free of charge at code.org/csp.

Lesson Plans

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

Videos

- Tutorials, instructional videos, and inspirational videos

Tools

- Widgets - designed 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

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. All of the course tools and resources (lesson plans, teacher dashboard, videos, student tools, programming environment, etc.) are available online. Tablets are not currently supported. 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. The course is developed to be completed within the classroom - no homework or after-hours computer access is assumed.

Required Materials and Supplies

Lessons make use of common classroom materials such as:

- Student journals or notebooks
- Poster paper
- Markers
- Post-it notes
- Plastic baggies

Suggested substitutions can be found in individual lesson plans. There is a complete materials list in the curriculum front matter available at code.org/csp. Optional materials are highly suggested, and low cost (cups, string, playing cards, etc.).

Covering the AP CSP Conceptual Framework

The CS Principles Conceptual Framework developed by the College Board outlines five “Big Ideas” of computing which are further subdivided into Enduring Understanding, Learning Objectives, and Essential Knowledge Statements. The framework further identifies six “Computational Thinking Practices,” containing skills that students should employ and develop. The curriculum is designed such that students investigate each of these big ideas while practicing the computational thinking practices.

Big Ideas

CRD: Creative Development
DAT: Data
AAP: Algorithms and Programming
CSN: Computing Systems and Networks
IOC: Impact of Computing

Conceptual Thinking Practices

CTP1: Computational Solution Design
CTP2: Algorithms and Program Development
CTP3: Abstraction in Program Development
CTP4: Code Analysis
CTP5: Computing Innovations
CTP6: Responsible Computing

Below, you will find detailed descriptions of each unit, that highlight the big ideas and computational thinking practices that are developed in that unit.

Unit 1 - Digital Information

Students explore how computers store complex information like numbers, text, images, and sound, and they debate the impacts of digitizing information (**DAT**). Alternating between lessons away from the computer (“unplugged”), and lessons that use digital tools called “widgets,” this unit encourages an exploratory and collaborative approach to learning about digital information. For example, in one activity students design a device using household items like pipe-cleaners, cups, string, etc. that will allow them to communicate simple messages across a room. As students are challenged to send increasingly complex messages, they must improve their device collaboratively with their partner while confronting some of the challenges underlying the representation of digital information (**CTP1**). To close out the unit, students debate the pros and cons of digitizing information and the impacts of digital information on society and culture at large (**CTP5**) (**IOC**).

Unit 2 - The Internet

Students learn how the Internet works and discuss its impacts on politics, culture, and the economy (**CSN**). Throughout this unit, students use a digital tool called the Internet Simulator that simulates how different parts of the Internet work and forces students to grapple with and solve the problems each aspect of the Internet was designed to solve (**CTP1**). At the conclusion of the unit, students investigate an “Internet Dilemma,” both from the standpoint of its technical background and its impacts on different groups of people (**CTP5**) (**IOC**).

Unit 3 - Intro to App Design

Students design their first app while learning both fundamental programming concepts and collaborative software development processes (**CRD**, **AAP**). Students work with partners to develop this simple app that teaches classmates about a topic of personal interest (**CTP1**, **CTP4**, **CTP6**). Throughout the unit they learn how to use App Lab to design user interfaces and write simple event-driven programs. Along the way, students learn practices like debugging, pair programming, and collecting and responding to feedback, which they will be able to use throughout the course as they build ever more complex projects. The unit concludes with students sharing the apps they develop with their classmates.

Unit 4 - Variables, Conditionals, and Functions

Students expand the types of apps they can create by adding the ability to store information (variables), make decisions (conditionals), and better organize code (functions) (**AAP**). Students are introduced to these concepts through guided hands-on activities that feature approachable manipulatives like sticky notes and plastic bags. They are then provided opportunities to explore working examples of programs that use each concept (**CTP4**) before setting out on a series of increasingly challenging practice activities. The exploration of each of these three concepts concludes with a lesson in which students must write the code for a simple app that uses each concept without starter code. The entire unit concludes with an open-ended project in which students must build an app that helps their classmates make a decision based on user input (**CTP1**, **CTP3**, **CTP4**, **CTP6**).

Unit 5 - Lists, Loops, and Traversals

Students learn to build apps that use and process lists of information (**AAP**). Like the previous unit, students explore the core concepts of lists, loops, and traversals through a variety of lesson types ranging from hands-on unplugged activities, to reading and modifying working code, to collaboratively working through programming challenges (**CTP4**). Late in the unit, students are introduced to tools that allow them to import tables of real-world data to help further power the types of apps they can make. At the conclusion of the unit, students complete a week-long project in which they must design an app around a goal of their choosing that uses one of these data sets (**CTP1**, **CTP4**, **CTP6**).

Unit 6 - Algorithms

Students learn to design and analyze algorithms to understand how they work and why some algorithms are considered more efficient than others (**AAP**). This short unit is entirely unplugged, and features hands-on activities that help students get an intuitive sense of the differences between how quickly different algorithms run and the pros and cons of different algorithms (**CTP2**). Later in the unit, students explore concepts like undecidable problems and parallel and distributed computing (**CSN**).

Unit 7 - Parameters, Return, and Libraries

Students learn how to design clean and reusable code that can be shared with a single classmate or the entire world (**AAP**). In the beginning of the unit, students are introduced to the concepts of parameters and return, which allow for students to design code that encapsulates algorithms (**CTP3**). In the second half of the unit, students learn how to design libraries of functions that can be used in a variety of situations. The unit concludes with students designing a small library of functions that can be used by a classmate (**CTP1**, **CTP2**, **CTP3**, **CTP4**).

Unit 8 - Create PT Prep

Students practice and complete the Create Performance Task (PT). The unit begins with a series of activities that ensure students understand the requirements of the Create PT, which they have practiced throughout the year. Subsequently, students are given at least 12 class hours in which to complete the Create PT.

Unit 9 - Data

Students explore and visualize datasets from a wide variety of topics as they hunt for patterns and try to learn more about the world around them (**DAT**). Students once again work with datasets in App Lab, but are now asked to make use of a data visualizer tool that assists students in finding data patterns. Students learn how different types of visualizations can be used to better understand the patterns contained in data sets and investigate hypotheses. At the conclusion of the unit, students learn about the impacts of data analysis on the world around them, before completing a final project in which they must uncover and present a data investigation they've completed independently (**CTP5**).

Unit 10 - Cybersecurity and Global Impacts

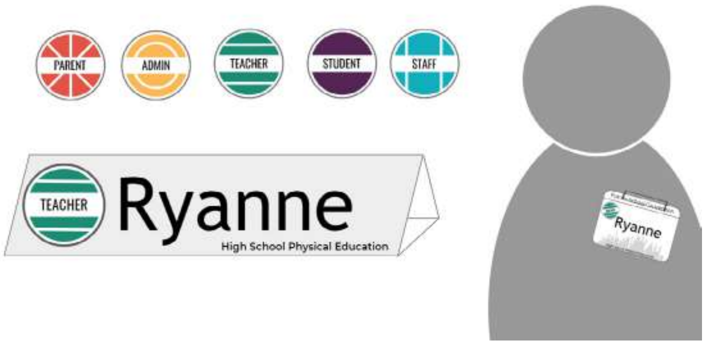
Students research and debate current events at the intersection of data, public policy, law, ethics, and societal impact (**IOC**). This unit is built around a simulated "future school" conference in which students must take on the persona of a stakeholder in a school setting and propose and debate technological innovations that could improve schools. Throughout the unit students learn about the privacy and security risks of many computing innovations, and learn about the ways some of these risks can be mitigated. Students complete their Explore Curricular Requirement as part of this project as they investigate at least three computing innovations, then discuss and debate many other innovations with their classmates (**CTP5**, **CTP6**). At the conclusion of the unit, the class holds a conference in which teams present their overall vision for a school of the future and the computing innovations that would power it.

Explore Curricular Requirement

If teaching this course as an AP course, the College Board requires that students explore Computing Innovations (**CI**) a minimum of three times, and address the following prompts at least once:

A. Explain beneficial and harmful effects of at least one computing innovation on society, economy, or culture.
B. Identify the data used in at least one computing innovation and explain how the data is consumed, produced, or transformed by the given computing innovation.
C. Identify data privacy, security, or storage concerns for at least one computing innovation.

Students complete the Explore Curricular Requirements in Unit 10: Cybersecurity and Global Impacts, during the Innovation Simulation project where they act as delegates proposing innovations to improve a “future school.”



- Students research and discuss three different computing innovations and consider the data that is being collected along with the function and purpose of the innovations (**CI 1, Prompt B**).
- Students choose one computing innovation to develop into a one-page proposal. Included in this proposal are the purpose and benefits of the computing innovation, its function, the data collected, and potential concerns along with how those concerns could be addressed (**CI 2, Prompts A, B, and C**).
- Students work with their groups to create a presentation about their computing innovations with the goal of convincing others that their innovations together are the best to address the concerns of the “future school.” Students discuss their innovations with their group to create a unifying theme for their presentation. Before the presentations are delivered, groups meet together to learn about each other’s innovations and give feedback on presentations. At this point in the project, students will have explored and discussed all the innovations in their own group along with the top innovations from another group (**CI 3, Prompts A, B, and C**).