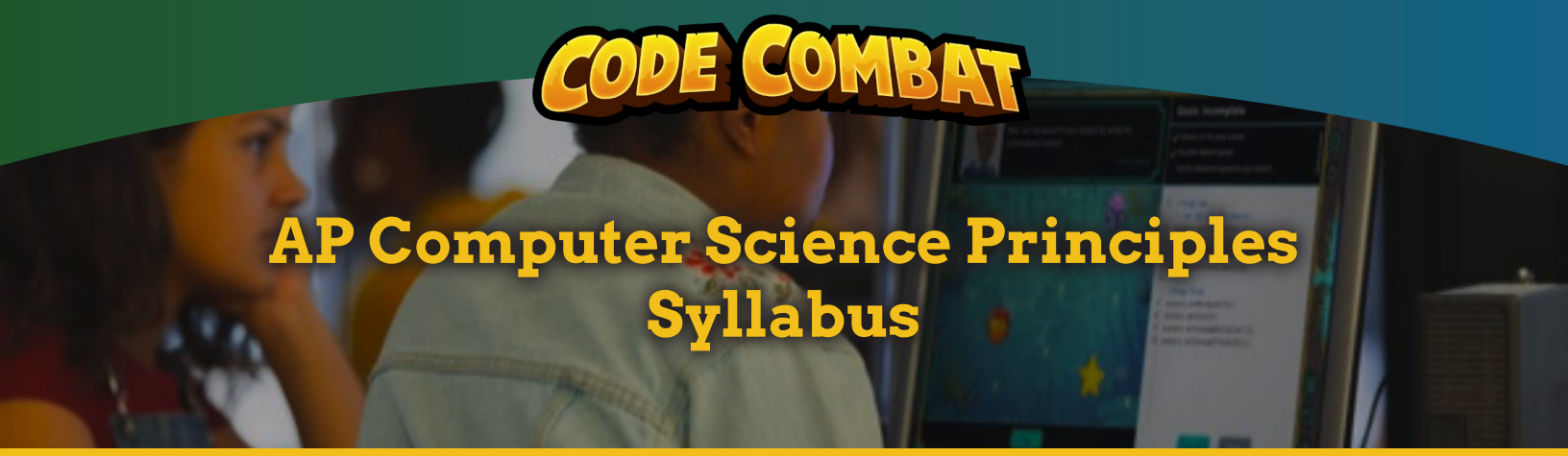

# AP Computer Science Principles Syllabus

CodeCombat's core mission is to change the way that computer science is taught. When we started CodeCombat, we knew that everything out there for learning to code was failing to serve most people, and we knew we could make it better – we could make the solution we'd wished we'd had when we were kids.

**Programming is magic**, and we wanted to give **all learners** the feeling of wizardly power at their fingertips by using **real typed code** on our platform. As it turns out, that was the key to get students to learn much faster as well; by not relying on training wheels like block-based code (which are especially underwhelming for students older than elementary school age), we empowered our students to learn programming syntax and code structure and enabled them to realize their creative potential much sooner.

## Computer Science for everyone

It just so happened that as CodeCombat's product matured, computer science turned out to be the number one job skill, so we expanded the scope of our vision to include not only programming skills, but also the broader aspects of computer science that we knew would set our students up for long-term success in the job market.

Making computer science accessible to every student on Earth isn't just about giving them programming tools -- it's also about helping students and teachers understand what computer science is, and who computer scientists are. It's about shattering common stereotypes by creating a curriculum that is friendly to beginner students AND beginner teachers, focuses on the amazing applications of computing and not just the technical jargon, and provides integrated, actionable feedback so that learners can quickly grasp new concepts within the context in which they're first learned. We've made sure that in-game, students can choose characters that can reflect their gender and race identities, knowing the importance of representation and its effect on students' perceptions.

The AP Computer Science Principles course is CodeCombat's natural accomplice in achieving the goals of reaching every potential computer science learner. The course's focus on driving diversity, increasing access and preparing students for the job markets of today and tomorrow aligns closely with our platform's core mission. We believe that our approach to the AP CS Principles Framework can inspire a brand new generation of students to see themselves as "computer science people," continue onto Computer Science and adjacent majors, and bring a fresh perspective to technology industries and beyond.

# AP Computer Science Principles Course Outline

The following descriptions are a high level overview of the content discussed in the CodeCombat AP CSP course. For more details regarding a day-to-day implementation, please review our comprehensive pacing guide.

| | | |
|---|---|---|
| **Unit 1** | **Computer Science 1** | Students begin the course focusing on creativity, problem solving, and the basic syntax of Python. |
| **Unit 2** | **Computer Science 2** | Students focus on designing and implementing algorithms using the building blocks of Python. |
| **Unit 3** | **Computer Science 3** | Students explore the concept of abstraction by developing their own abstractions inside program code. |
| **Unit 4** | **Computer Science 4** | Students apply their understanding of problem solving, algorithms, and abstractions to design and implement digital games. |
| **Unit 5** | **Creative Development** | Students use all that they have learned about Python to complete and submit their Create Performance Task. |
| **Unit 6** | **Data** | Students learn how computers consume, transform, store, and produce new information in order to solve problems. |
| **Unit 7** | **Computer Systems and Networks** | Students experience the power of sharing information via computer networks by learning about the Internet. |
| **Unit 8** | **Impact of Computing** | Students examine how computing has revolutionized our lives and society. |
| **Unit 9** | **Exam Prep** | Students prepare to take the multiple choice section of the AP CSP exam. |

# CodeCombat & AP Computer Science Principles Conceptual Framework

The College Board has outlined five **Big Ideas** and six **Computational Thinking Practices** all students must learn in order to be successful on the AP CSP exam. These Big Ideas and Computational Thinking Practices are described below:

## Big Ideas

**Creative Development (CRD):**
When developing computing innovations, developers can use a formal, iterative design process or experimentation. While using either approach, developers will encounter phases of investigating and reflecting, designing, prototyping, and testing. Additionally, collaboration is an important tool to use at any phase of development because considering multiple perspectives allows for improvement of innovations.

**Data (DAT):**
Data are central to computing innovations because they communicate initial conditions to programs and represent new knowledge. Computers consume data, transform data, and produce new data, allowing users to create new information or knowledge to solve problems through the interpretation of these data. Computers store data digitally, which means that the data must be manipulated in order to be presented in a useful way to the user.

**Algorithms and Programming (AAP):**
Programmers integrate algorithms and abstraction to create programs for creative purposes and to solve problems. Using multiple program statements in a specified order, making decisions, and repeating the same process multiple times are the building blocks of programs. Incorporating elements of abstraction, by breaking problems down into interacting pieces, each with their own purpose, makes writing complex programs easier. Programmers need to think algorithmically and use abstraction to define and interpret processes that are used in a program.

**Computing Systems and Networks (CSN):**
Computer systems and networks are used to transfer data. One of the largest and most commonly used networks is the Internet. Through a series of protocols, the Internet can be used to send and receive information and ideas throughout the world. Transferring and processing information can be slow when done on a single computer but leveraging multiple computers to do the work at the same time can significantly shorten the time it takes to complete tasks or solve problems.

**Impact of Computing (IOC):**
Computers and computing have revolutionized our lives. To use computing safely and responsibly, we need to be aware of privacy, security, and ethical issues. As programmers, we need to understand the potential impacts of our programs and be responsible for the consequences. As computer users, we need to understand any potential beneficial or harmful effects and how to protect ourselves and our privacy when using a computer.

You can more deeply explore how our units align with each Big Idea's learning objectives and enduring understandings in our standards alignment document.

## Computational Thinking Practices

**Practice 1 - Computational Solution Design:** Design and evaluate computational solutions for a purpose.

**Practice 2 - Algorithms and Program Development:** Develop and implement algorithms.

**Practice 3 - Abstraction in Program Development:** Develop programs that incorporate abstractions.

**Practice 4 - Code Analysis:** Evaluate and test algorithms and programs.

**Practice 5 - Computing Innovations:** Investigate computing innovations.

**Practice 6 - Responsible Computing:** Contribute to an inclusive, safe, collaborative, and ethical computing culture.

You can learn more about how these **Big Ideas** and **Computational Thinking Practices** are broken down by reviewing the **official AP CSP conceptual framework** found here:
https://apcentral.collegeboard.org/media/pdf/ap-computer-science-principles-conceptual-framework-2020-21.pdf

Below, you will find detailed descriptions of each unit that illustrate how each **Big Idea** and **Computational Thinking Practice** is addressed within the course.

**Unit 1 - Computer Science 1:**
Students begin the course focusing on creativity, problem solving, and programming, which sets the tone for CodeCombat's holistic approach to computer science. In this unit, students will get hands-on experience with basic coding concepts, from sequencing to iteration **(AAP)**. Every lesson in this unit includes an activity that asks students to practice identifying and correcting errors found in pre-written programs **(Practice 4)**. Finally, this unit culminates with a capstone software development project, where students apply their understanding of the Engineering Design Process in order to create static, digital images **(CRD, Practice 1, Practice 6)**. Here, students reason with the ethical and legal concerns surrounding collaborative software development processes **(IOC, Practice 6)**. The capstone project ends with a structured reflection activity that asks students to reason about the purpose of their capstone programs **(CRD)**.

**Unit 2 - Computer Science 2:**
In this unit, students will focus the bulk of their efforts towards understanding how to design, implement, evaluate, and test algorithms using programming concepts like conditionals, functions, and events **(AAP, Practice 2, Practice 4)**. Every lesson in this unit includes a compare and contrast activity that prompts students to determine if two code segments produce the same result **(Practice 4)**. This unit culminates with another capstone project, where students apply their understanding of the Engineering Design Process in order to develop a text-based adventure game for a friend **(CRD, Practice 1, Practice 2, Practice 3, Practice 4, Practice 6)**. In doing so, students will continue to reason about the ethical and legal concerns surrounding collaborative software development processes **(IOC, Practice 6)**. Again, this capstone project ends with a structured reflection activity that asks students to describe how a code segment in their program functions **(Practice 4)**.

**Unit 3 - Computer Science 3:**
This unit covers more advanced programming concepts like functions with return statements, complex conditional statements, and arithmetic expressions **(AAP, Practice 2, Practice 3, Practice 4)**. Students continue to engage in the error detection activities and output prediction activities introduced in Units 1 and 2 in order to deepen their reading comprehension skills **(Practice 4)**. More importantly, this unit exposes students to the concept of procedural abstraction, and begins to scaffold how students will build their own abstractions within increasingly more complex programs **(Practice 3)**. This unit culminates with another capstone project, where students apply their understanding of the Engineering Design Process in order to develop a simulation **(CRD, Practice 1, Practice 2, Practice 3, Practice 4, Practice 6)**. In doing so, students will continue to reason about the ethical and legal concerns surrounding collaborative software development processes **(IOC)**. This capstone project ends with a structured reflection activity that builds upon previous capstone reflections by asking students to describe how the functionality of a specific code segment in their program ultimately contributes to the overall purpose of their project **(CRD, Practice 4)**.

**Unit 4 - Computer Science 4:**
This unit exposes students to lists and the statements students need to access and modify the data stored within lists **(AAP, DAT)**. Students continue to engage in error detection activities and output prediction activities in order to deepen their reading comprehension skills of programs that utilize increasingly complex algorithms **(Practice 4)**. The capstone project of Unit 4 asks students to design and implement a digital game experience of their choosing, incorporating skills – like debugging and effective collaboration – acquired across Units 1 - 4 **(AAP, CRD, IOC, Practice 1, Practice 2, Practice 3, Practice 4, Practice 6)**. The reflection activity of this capstone project integrates the reflection prompts seen in previous capstone projects **(CRD, Practice 4)**, culminating with a prompt that asks students to reason about how their chosen abstractions help to manage the complexity of their games **(Practice 3)**.

**Unit 5 - Creative Development:**
In this unit, students are provided the scaffolds they need to successfully complete the Create Performance Task **(CRD)**. Students will apply their understanding of the Engineering Design Process, collaboration, and debugging in order to develop a program that meets all of the requirements outlined by the College Board's Create Performance Task Scoring Guidelines. We have allotted 12 hours for students to work on their Create Task projects, but encourage instructors to consider providing additional work time if need be. In the process of building their programs, students will undoubtedly continue to practice effective collaboration skills **(CRD)**, and continue to reason about the ethical and legal concerns of building software collaboratively **(IOC)**.

**Unit 6 - Data:**
This unit begins by demonstrating how computers represent complex information, like sound and images, as binary **(DAT)**. The unit continues to build on a students' appreciation of data by having students practice converting data into information through common analytical practices **(DAT)**. The capstone project for this unit consists of a data processing project in which students perform data analysis using Google Sheets functions in order to answer a question about their local school community **(DAT)**. Here, students examine work together to identify a problem and to propose actionable next steps that the broader audience could adopt **(Practice 6)**. Students also describe the societal benefits and harms of collecting large amounts of data **(IOC)**.

**Unit 7 - Computer Systems and Networks:**
In this unit, students learn about the structure and protocols of the Internet **(CSN)**. In doing so, students more deeply explore the societal implications of being able to transmit, share, and collect information via a computer network **(IOC)**. Students combine their understanding of the Internet and its societal impact to create instructional videos that review important network vocabulary and highlight the pros and cons of different computing systems.

**Unit 8 - Impact of Computing:**
The final instructive unit of our curriculum connects each of the important concepts that students have been exploring throughout the year to a larger understanding — that computing is powerful and that students themselves can effect change in the world via computing **(IOC)**. Throughout the unit, students investigate well known computing innovations. They begin by exploring the benefits and harms that self-driving cars have had on society, economy, and culture **(CI1A, Practice 5)**. Students then explore how data is consumed, produced, or transformed by TikTok **(CI2B, Practice 5)** and extend their understanding of privacy concerns by examining Facebook and the Cambridge Analytica scandal **(CI3C, Practice 5)**. Finally, this unit culminates in a project that asks students to design a prototype of their own computing innovation and reflect on how this innovation may impact society **(Practice 5)**. Exploring all of these computing innovations will indirectly help students prepare to answer the critical reasoning questions found on the exam that address topics once covered by the Explore Performance Task.

**Unit 9 - Exam Preparation:**
This is time set aside prior to the exam to give students an opportunity to review the content tested on the multiple choice section of the AP CSP exam and to understand how the test will be administered.

# AP Computer Science Principles

## Materials Provided

This curriculum includes all of the resources a teacher may need to successfully teach an AP CSP course, including: a comprehensive pacing guide, lesson plans, and slide decks for every day of instruction, formative and summative assessments, and the programming environment, CodeCombat. These resources have been designed to help teachers of all experience levels facilitate and differentiate learning across a diverse classroom. All resources for the CodeCombat AP CSP course can be accessed once a license has been purchased. Information on license pricing and structure can be obtained by speaking to CodeCombat's school specialists (email: apcsp@codecombat.com). We recommend that licenses are obtained by early August of the coming school year so that students can begin as early as the first day of the Fall semester.

## Technical Requirements

CodeCombat's AP Computer Science Principles course can be used to learn programming in Python. CodeCombat runs best on computers with at least 4GB of RAM, on a modern browser such as Chrome, Safari, Firefox, or Edge. Chromebooks with 2GB of RAM may have minor graphics issues in courses beyond Computer Science 3, though there should be minimal issues with the recommended content for AP Computer Science Principles as outlined. A minimum of 200 Kbps bandwidth per student is required, although 1+ Mbps is recommended.

**Questions?**

**We invite you to reach out to CodeCombat's school specialists at:**
**apcsp@codecombat.com**