Open doors in any career with computer science!

Students create apps for mobile devices, automate tasks in a variety of languages, find patterns in data, and interpret simulations. Students collaborate to create and present solutions that can improve people's lives.

How will computing and connectivity transform your world?

Computer Science Principles (CSP) is a PLTW course to implement the College Board's new AP CS Principles framework. Students work in teams to develop computational thinking and solve problems. The course does not aim to teach mastery of a single programming language but aims instead to develop computational thinking, to generate excitement about the field of computing, and to introduce computational tools that foster creativity. The course also aims to build students' awareness of the tremendous demand for computer specialists and for professionals in all fields who have computational skills. Each unit focuses on one or more computationally intensive career paths. The course also aims to engage students to consider issues raised by the present and future societal impact of computing.

Students practice problem solving with structured activities and progress to open-ended projects and problems that require them to develop planning, documentation, communication, and other professional skills. Problems aim for ground-level entry with no ceiling so that all students can successfully engage the problems. Students with greater motivation, ability, or background knowledge will be challenged to work further.

The following is a summary of the units of study that are included in the. The course is designed to cover all learning objectives in the College Board's AP CS Principles framework and to prepare students to do well on the AP assessment. In specific CSP projects and problems, students create artifacts and associated writing as practice for the AP CS Principles Performance Tasks that can be submitted to the College Board. Alignment with AP CS Principles Learning Objectives and Essential Knowledge statements, CSTA Level 3B Objectives, and alignment with Next Generation Science Standards and Common Core State Standards for Mathematics and English Language Arts is indicated within course materials at the activity-, project-, and problem level. Activities, projects, and problems are be provided to the teacher in the form of student-ready handouts, teacher notes, and supplementary materials including code, instructional videos, and online practice questions as appropriate.

The course is planned for a rigorous pace, and it is likely to contain more material than a skilled teacher new to the course will be able to complete in the first iteration. Building enthusiasm for rigorous computer science among students is a primary goal of the course. Teachers are encouraged to emphasize content that will be fresh and exciting to students, and the course is structured to facilitate local adaptation to a particular group of students' prior knowledge and experience.

CSP Unit Summary

Unit 1	Algorithms, Graphics, and Graphical User Interfaces (77 days, 48%)
Unit 2	The Internet (30 days, 18%)
Unit 3	Raining Reigning Data (30 days, 18%)
Unit 4	Intelligent Behavior (28 days, 17%)

Total 162 days

Unit 1: Algorithms, Graphics, and Graphical User Interfaces (68 days)

The goal of Unit 1 is to excite students about programming and to build their algorithmic thinking and ability to use abstraction. Student creativity is emphasized as they work with ScratchTM, App Inventor, and Python® programming languages to tell graphical stories, publish games and AndroidTM applications, and explore various development environments and programming techniques. Students create original code and read and modify code provided from other sources. An Agile software development process is emphasized and personal, professional, and collaborative skills take center stage. Students debate policy questions about the ownership and control of digital data and examine the implications for creative industries and consumers. In this unit students begin their exploration of career paths tied to computing.

Algorithms, Graphics, and Graphical User Interface Lesson Summary

Lesson 1.1	Algorithms and Agile Development (15 days)
Lesson 1.2	Mobile App Design (15 days)
Lesson 1.3	Algorithms in Python (20 days)
Lesson 1.4	Images and Object-Oriented Libraries (17 days)
Lesson 1.5	GUIs in Python (1 day + 9 optional days)

Lesson 1.1 Algorithms and Agile Development (15 days)
The goal of this lesson is to introduce students to programming at a level appropriate to novice programmers. With an introduction to pair programming and the Agile software development process, students create original programs in Scratch that incorporate audio and visual elements while tackling algorithmic problems. The lesson opens with an introduction to how computing is affecting our lives. Students explore tools for collaboration over the Internet and select from these tools in order to manage the projects that they create. The foundations for later algorithmic thinking are built by focusing on the most common roles that variables fulfill, with an introduction to the conventions of object-oriented programming.

Activity 1.1.1	Principles (2 days)
Activity 1.1.2	LightbotTM: Input, Output, State (1 day)
Activity 1.1.3	Branching and Iteration (2 days)
Activity 1.1.4	Objects and Methods (1 day)

Activity 1.1.5	Variable Roles I (1 day)
Activity 1.1.6	Variable Roles II (2 days)
Problem 1.1.7	Scratch Game or Story (6 days)

Lesson 1.2 Mobile App Design (15 days)
The goal of this lesson is for students to build their skills by analyzing existing code, particularly with an emphasis on the roles of variables. Students create an Android app of their own design. The lesson begins with an introduction to binary representations of numbers, letters, colors, images, etc. using a CS unplugged activity in which students create a physical representation of data storage. Students work with and make minor modifications to two App Inventor programs, building their ability to analyze a complex program and incorporate event handlers into programs in meaningful ways. Students conclude by designing and creating their own Android app using pair programming and practicing the Agile software design process.

Activity 1.2.1	Bits and Bytes (1 day)
Activity 1.2.2	Introducing App Inventor (2 days)
Activity 1.2.3	Creating Mobile Apps (2 days)
Activity 1.2.4	Analyzing a Program (2 days)
Project 1.2.5	Modifying a Program (2 days)
Problem 1.2.6	Designing an App (6 days)

Lesson 1.3 Algorithms in Python (20 days)
The goal of this lesson is for students to understand all information as bits and to transfer their understanding of algorithms to a new language, Python, which is powerful enough to raise all the opportunities and issues targeted in the course. Students are introduced to functional, imperative, and declarative programming paradigms with Python, again learning to use variables in the most common roles. Before learning about variable types and the fundamental algorithmic structures in Python, students simulate program execution in a model assembly language. After building strength with basic Python algorithms, students create algorithms to compete in a round-robin tournament of the Prisoner's Dilemma, using the collaborative programming platform GitHub in the process.

Activity 1.3.1	Programs are Data (1 day)
Activity 1.3.2	Python Variables and Functions (2 days)
Activity 1.3.3	Branching and Output (2 days)
Activity 1.3.4	Nested Branching and Input (2 days)
Activity 1.3.5	Strings (2 days)
Activity 1.3.6	Tuples and Lists (3 days)
Activity 1.3.7	For Loops (3 days)
Activity 1.3.8	While Loops (2 days)
Activity 1.3.9	Tools for Collaboration (1 day)
Project 1.3.10	Game Theory (2 days)

Lesson 1.4 Images and Object-Oriented Libraries (17 days)
The goal of this lesson is for students to become independent learners of a programming language, able to refer to documentation to use object-oriented libraries commonly available. The lesson begins with an unplugged activity to teach object-oriented concepts. Students build additional strength with Python algorithms, manipulating image files by modifying pixel data and using code libraries to work at higher levels of abstraction. As part of that work, they learn to use a variety of documentation including application programming interfaces (APIs). Students read,

discuss, and debate intellectual property issues associated with digital data. In the culminating problem of the lesson, they collaborate to create an image processing function that highlights the power of automation.

Activity 1.4.1	Procedural Abstraction (1 day)
Activity 1.4.2	Objects and Methods (2 days)
Activity 1.4.3	Images and Arrays (2 days)
Activity 1.4.4	Python Imaging Library API (2 days)
Project 1.4.5	Image Algorithms (3 days)
Activity 1.4.6	Digital Property and Forensics (2 days)
Problem 1.4.7	Image Artist (4 days)
Lesson 1.4 PT	Performance Task (1 day)

Lesson 1.5 GUIs in Python (10 days, optional)
The goal of this lesson is for students to conceive of any class of objects as an abstraction. Students will create a graphical user interface (GUI) with considerations of audience and accessibility. The lesson begins with an unplugged activity that generalizes the user interface topic of this lesson to the field of human-computer interaction. The remainder of the lesson is optional and used to differentiate the curriculum across different schools, depending on whether students entered the course having already learned some programming in earlier grades using Scratch, App Inventor, or other environments such that Lessons 1.1 and 1.2 are expedited or omitted.

In the lesson, students practice using an application programming interface (API) to learn methods that affect an object's state. Students work with two APIs: the Tkinter Canvas for drawing and animation and the Tkinter toolbox of GUI widgets. Students are provided code for a simple GUI that implements a model-view-controller (MVC) pattern. Students will modify the elements of that pattern to suit their own needs. The lesson concludes with a problem in which students create a model-view-controller GUI using Scratch or Python. Strategies for documentation are reinforced, and Agile development is emphasized in the concluding problem.

Activity 1.5.1	Human-Computer Interaction (1 day)
Activity 1.5.2	The API for the Tkinter Canvas (2 days)
Activity 1.5.3	The MVC Pattern with Tkinter (2 days)
Problem 1.5.4	Design a Python GUI (5 days)

Unit 2: The Internet (30 days)

The goal of Unit 2 is for students to have a more concrete understanding of the Internet as a set of computers exchanging bits and the implications of these exchanges. Students use PHP and SQL to structure and access a database hosted on a remote server, learn how HTML and CSS direct the client computer to render a page, and experiment with JavaScript™ programming language to provide dynamic content. The focus of the unit is on the protocols that allow the Internet to function securely as it delivers social media and eCommerce content. Students work briefly in each of several web languages to understand how the languages work together to deliver this content. The history and workings of the Internet are explored, and issues of security, privacy, and democracy are considered. Practical cybersecurity hygiene is included. Career paths in cybersecurity, web development, and information technology are highlighted.

The Internet Lesson Summary

Lesson 2.1	The Internet and the Web (9 days)
Lesson 2.2	Shopping and Social on the Web (13 days)
Lesson 2.3	Security and Cryptography (8 days)

Lesson 2.1 The Internet and the Web (9 days)
In this lesson the goal is to build student understanding of the Internet as a set of computers exchanging bits in the form of packets. Students will learn to identify the components of their digital footprint and compare the designs, strengths, and weaknesses of their favorite web pages. In this context students use an unplugged activity to understand (in broad brushstrokes) the content and flow of data when browsing the web. They compare results from different search engines and learn to refine their search techniques. They review how to assess the trustworthiness of web-based media and consider the data flow that permits targeted advertisements. Students employ appropriate tools to explore the hierarchical nature of DNS and IP. Students identify ways that a web developer's decisions affect the user and ways that the user's decisions impact society. The tree structure of web documents is introduced alongside HTML and CSS. Students exchange keys and messages, use Python functions to encrypt and decrypt, and explain how paired key encryption and certification authorities provide security and authentication.

Activity 2.1.1	The Rise of the Internet (2 days)
Activity 2.1.2	Your Favorite Web Page (1 day)
Activity 2.1.3	Protocols and Bandwidth (2 days)
Project 2.1.4	HTML and CSS (3 days)
Activity 2.1.5	Secure Protocols (1 day)

Lesson 2.2 Shopping and Social on the Web (13 days)
The goal for this lesson is for students to understand the role of client-side code, server-side code, and databases in delivering interactive web content. The hook is a problem in which CS students collaborate with art students to publish content on the web. Students are provided with JavaScript and PHP code and can access an SQL database from a shell command line as well as through PHP. Students compare languages encountered so far to generalize the concepts of sequencing instructions, selection of instructions by conditionals, iteration, and the common roles of variables. Students explore and compare career paths within computing. Students also begin selecting topics for the CS Principles Performance Tasks as described in the final Problem 4.2.5 of the course.

Activity 2.2.1	HTML5 and JavaScript (3 days)
Activity 2.2.2	Introducing PHP (3 days)
Activity 2.2.3	Databases and SQL (2 days)
Problem 2.2.4	Dynamic Data-Driven Design (4 days)
Activity 2.2.5	Career Fields of CS and IT (1 day)

Lesson 2.3 Security and Cryptography (8 days)
The goal of this lesson is for students to personally invest in maintaining online security and to improve their personal cybersecurity hygiene. Students focus on cybersecurity from the perspectives of the user, the software developer, the business, the nation, and the citizen. In the team competition at the end of the lesson, students

explore parallel strands in encryption and security. Encryption is used as a route to explore the efficiency of algorithms and how the time for an algorithm to execute can be dependent on its input.

Activity 2.3.1	The Vulnerable User (2 days)
Activity 2.3.2	Security by Encryption (1 day)
Activity 2.3.3	Security and Liberty (2 days)
Project 2.3.4	The Heist (3 days)

Unit 3: Raining Reigning Data (30 days)

The goal of Unit 3 is for students to see the availability of large-scale data collection and analysis in every area they can imagine. Students examine very large data sets tied to themselves as well as to areas of work and society. They learn a variety of data visualization techniques and work to recognize opportunities to apply algorithmic thinking and automation when considering questions that have answers embedded in data. The complexity of the data sets, visualizations, and analysis increases in the second lesson of the unit, challenging students to generalize concepts developed in the first lesson.

Lesson 3.1	Visualizing Data (14 days)
Lesson 3.2	Discovering Knowledge from Data (16 days)
Lesson 3.1	Visualizing Data (14 days)

The goal of this lesson is for students to be able to create visualizations to analyze large sets of data and to meaningfully interpret the patterns they uncover. They draw conclusions relevant to themselves from data, including local weather, the economics of their community, and naming trends with their name. At the beginning of the lesson, students weigh societal concerns around the collection and persistence of Big Data. The students learn how to use Python to make useful graphic representations of data, developing from familiar visualizations to more modern visual analyses like scaled-dot or colorized scatter plots of multidimensional data sets. Students are introduced to basic Excel® spreadsheet programming and cell manipulation. A Monte Carlo simulation is used to help students appreciate the meaning of evidence for association between two variables.

Activity 3.1.1	Time Series and Trends (3 days)
Activity 3.1.2	Privacy Issues with Data (2 days)
Activity 3.1.3	Data Innovations and Parallel Algorithms (2 days)
Activity 3.1.4	Pie Charts and Bar Graphs (3 days)
Activity 3.1.5	Histograms and Distributions (4 days)

Lesson 3.2 Discovering Knowledge from Data (16 days)
As in the previous lesson, the goal of this lesson is for students to be able to create a range of visualizations to analyze large complex sets of data and to meaningfully interpret the patterns they uncover. Students use statistics to deepen the meaning of knowledge gained by visualization. The hooks are again conclusions they can draw from data relevant to themselves, including various geographic perspectives on their life and facial recognition of their own features. The lesson uses Excel as well as Python to manipulate and visualize data. Students examine multidimensional data sets using scatter plot arrays and view geographic and social data using heat maps and

data while simulating mutations in a population and then investigate genomic data of many species using public scientific database. Finally, student teams choose a question and answer it using large sets of data.

Activity 3.2.1	Inferential Statistics (2 days)
Activity 3.2.2	Image Data (1 day)
Activity 3.2.3	Linked Data (1 day)
Activity 3.2.4	Geographic Data (2 days)
Activity 3.2.5	Simulation Data (1 day)
Activity 3.2.6	Genomic Data (3 days)
Problem 3.2.7	Investigating with Data (6 days)

Unit 4: Intelligent Behavior (28 days)

In Unit 4 the emergence of intelligent behavior is explored from two distinct approaches: from human crowd sourcing of data and from separate algorithmic agents working in parallel. The goal is to galvanize the connections among computing concepts and between computing and society. The first lesson explores the hardware layer of computing, working from discrete components to integrated circuits. The exponential advancement of electronics, low on the ladder of abstraction, is connected to advancements at the highest levels on the ladder of abstraction, where artificial intelligence and simulation and modeling are impacting all fields. In the concluding lesson, students identify problems and questions that can be addressed with computer simulation, incorporating agent-based modeling. Students are challenged to explore the assumptions and parameters built into several simulations and to attach meaning to the results. Having explored a few applications of intelligent behavior emerging from algorithmic components, students reflect on the current and future state of artificial intelligence.

Lesson 4.1	Moore's Law and Modeling (12 days)
Lesson 4.2	Intelligent Agents (16 days)

Lesson 4.1 Moore's Law and Modeling (12 days)
In this lesson students construct an understanding of how the explosion of technology over the last two decades has impacted every realm of study and employment. Students begin by researching the impact of computer modeling and simulation which have been made possible by the rapid increase in computational power due to the continued applicability of Moore's Law. They then manipulate discrete electronic components to create logic gates and then create comparable results using integrated circuits. Students comprehend what it means to double the number of transistors that can fit in a given area. Students explore simulation in NetLogo directly by manipulating a model of predation and a model of the spread of viruses in humans. The lesson concludes with an examination of the code of ethics for simulationists and reflection on the necessity of adhering to such a code.

Activity 4.1.1	Computing Impacts All Fields (2 days)
Activity 4.1.2	Basic Control Circuits (3 days)
Activity 4.1.3	Introducing Simulations (3 days)
Activity 4.1.4	Varying Parameters (2 days)
Activity 4.1.5	Assumptions, Abstractions, and Ethics (2 days)

Lesson 4.2 Intelligent Agents (12 days + 20 hours through-course)
In this lesson students experiment with materials designed to illuminate the rise of intelligent and complex behavior from simple rules and seemingly unintelligent agents. Students begin by studying a model of Langton's ant, a simple Turing machine with some surprising emergent behavior. The students manipulate models of neurons and neural networks. Students design and conduct their own experiments on a model of their own choosing using Monte Carlo methods. Students explore the generation and observation of fractals and study a diffusion-limited aggregation model for producing fractal behavior. In the final project of the course, students choose a tool or tools that they have learned about in the course and apply their knowledge to create a novel product of their own design and to investigate the beneficial and harmful impacts of a computing innovation. In the final project, students present the artifacts they produced for the College Board AP Create and Explore Performance Tasks. The tasks are typically begun during Unit 2 or 3 so that the work can be completed in accordance with College Board AP submission timeline. Students present their Create product to their class along with reflections about how the artifacts tie together what they've learned about computer science.

Activity 4.2.1	Emergent Behavior (2 days)
Activity 4.2.2	Neural Networks (3 days)
Project 4.2.3	Modifying a Simulation's Assumptions (5 days)
Activity 4.2.4	Beauty in Chaos and Fractals (2 days)
Project 4.2.5	CS Principles (8 hours Explore + 12 hours Create)

AP CSP Endorsed
PLTW 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 PLTW 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.