Express your creativity through code. Analyze computing innovations and the impacts they have on our lives. Use abstraction and algorithmic thinking to solve problems and create value for others. Develop, analyze, implement, and test programs developed for a purpose. Learn to uncover patterns in data, protect data, and explore how the internet connects the world in which we live.

Whether seeking a career in the growing field of computer science or learning how computer science is transforming all careers, students in Computer Science Principles learn the fundamentals of coding, data processing, data security, and automating tasks while learning to contribute to an inclusive, safe, and ethical computing culture.

PLTW Computer Science Principles NEW (2020-2021)

PLTW's Computer Science Principles is a full-year course recommended for students in grades 10–12. It is as an introductory college-level course for non-computer science majors. The course aligns to CSTA Level 3B Objectives, ISTE Standards, and the K–12 CS Framework. Additionally, PLTW courses are designed to prepare students to thrive in college, careers, and beyond. As a result, many students choose to take AP exams to demonstrate the knowledge and skills they've gained to colleges and universities.

Exploring Careers in Computer Science

In Computer Science Principles, students develop the in-demand computer science skills critical to thrive in any of today's and tomorrow's careers. The course promotes computational thinking and coding fundamentals and introduces computational tools that foster creativity. It aims to build students' awareness of the tremendous demand for computer scientists and those who have computational thinking skills, and engages students to consider issues raised by the impact of computing on society. Each unit also focuses on one or more computer science-specific career paths.

The Big Ideas of Computer Science Principles

PLTW Computer Science Principles provides students opportunities to develop understanding of each of the five Big Ideas described in the CSP AP Course and Exam Description. While the CSP Big Ideas are spiraled throughout the course, each is highlighted in a specific project or problem.
The "Big Ideas" in Computer Science Principles Include:

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

The Computational Thinking Practices of Computer Science Principles

PLTW Computer Science Principles provides students opportunities to apply computational thinking practices described in the CSP AP Course and Exam Description. While the CSP Computational Thinking Practices are spiraled throughout the course, each is highlighted in a specific project or problem.
The "Computational Thinking Practices"" in Computer Science Principles Include:

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

Exploring Computing Innovations in Computer Science Principles

PLTW Computer Science Principles provides students opportunities to investigate different computing innovations throughout the course. As part of those explorations, students will:

- *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.*

PLTW Explore Tasks

- *Activity 2.2.1 The Internet and the Web: Explore Task 1 (A,B,C)*
- *Activity 2.2.5 Analyzing Data and Computing: Explore Task 2 (A,B,C)*
- *Problem 4.3.1 Impacts of Computing: Explore Task 3 (A,B,C)*

Unit 1: Creative Computing for All (44 days) (27%)

IIn Unit 1, students are introduced to Creative Development (CRD), Algorithms and Programming (AAP), and the Computational Thinking Practices of successful computer science professionals. Students unlock the power of creativity as they apply coding fundamentals to create digital images, animations, interactive stories, and games. They engage in fun, authentic experiences that reflect the diverse and globally relevant transportable skills of computer science.

Lesson 1.1 Algorithms (23 Days)

Lesson 1.1 introduces students to text-based programming at a level appropriate for novice programmers. Students create original programs using turtle graphics while learning how variables, inputs, and outputs come together in an algorithm to make things happen. The foundations for later algorithmic thinking are established by focusing on the most common roles that variables fulfill and using standard code libraries to customize their programs.

Activity 1.1.1	Algorithmic Thinking	(2 days)
Activity 1.1.2	Planning a Picture	(2 days)
Activity 1.1.3	Fun with Flowers	(2 days)
Activity 1.1.4	Spinning with Spirographs	(3 days)
Activity 1.1.5	Buggy Image	(4 days)
Activity 1.1.6	Traversing Turtles	(3 days)
Activity 1.1.7	Turtles in Traffic	(3 days)
Project 1.1.8	Algorithms and Art	(4 days)
	(CPT2) Algorithms and Program Development (AAP) Algorithm and Programming	

Lesson 1.2 Abstraction **(17 Days)**

In Lesson 1.2, students use a development process and abstractions, such as procedures, functions, lists, and datatypes, to collaborate and create a game.

Activity 1.2.1	Catch-A-Turtle	(3 days)
Activity 1.2.2	Catch-A-Turtle Leaderboard	(4 days)
Activity 1.2.3	Apple Avalanche	(3 days)
Activity 1.2.4	Turtle Escape	(3 days)
Project 1.2.5	Shall We Play a Game? (CTP3) Abstraction in Program Development (AAP) Algorithms and Programming	(4 days)

Lesson 1.3 Artistic Expression Through Code **(4 days)**

In Lesson 1.3, students apply all the coding fundamentals and computational thinking practices they have learned to create a program of their choosing.

Problem 1.3.1	Artistic Expression through Code (CTP1) Computational Solution Design (CRD) Creative Development	(4 days)

Unit 2: Every Bit of the Internet (41 days) (25%)

In Unit 2, In students are explore **Data (DAT), Computing Systems and Networks (DAT), and Impact(s) of Computing (IOC)**. Students assume the role of a network analyst as they write programs that help manage or observe data from the internet. They explore a variety of internet protocols and formats for data while examining the ways in which their data can either be protected or exposed.

Lesson 2.1 Data Diligence **(16 days)**

Lesson 2.1 introduces students to personal cybersecurity by exploring password strength, encryption, and what it takes to protect data. Students focus on cybersecurity from the perspectives of the user, the software developer, the business, the nation, and the citizen.

Activity 2.1.1	Alert: Phishing Warning!	(2 days)
Activity 2.1.2	Encryption: Keep it Confidential	(3 days)
Activity 2.1.3	Password Strength – Strong!	(2 days)
Activity 2.1.4	Design the User Experience	(3 days)
Activity 2.1.5	Securing Sloppy Code (CPT4) Code Analysis	(2 days)
Project 2.1.6	A pHishy Fish Tank (CTP6) Responsible Computing (CSN) Computing Systems and Networks	(4 days)

Lesson 2.2 How the Internet Works **(20 days)**

In Lesson 2.2, students come to understand the internet as a set of computers exchanging bits in the form of packets. Students employ appropriate tools to explore the internet's hierarchical infrastructure and create their own custom user interfaces to examine the internet and understand how it works.

Activity 2.2.1	The Internet and the Web: Explore Task 1 (A,B,C) (CTP5) Computing Innovations	(3 days)
Activity 2.2.2	A Little Bit of Data	(2 days)
Activity 2.2.3	Demystifying Data Transmission (Cloud9)	(3 days)
Activity 2.2.4	Parallel and Distributed Computing	(2 days)
Activity 2.2.5	Analyzing Data and Computing : Explore Task 2 (A,B,C) (CTP5) Computing Innovations (CSN) Computing Systems and Networks	(2 days)
Activity 2.2.6	A GUI Situation	(3 days)
Project 2.2.7	Creating a Command Line GUI	(5 days)

Lesson 2.3 Creating a Custom Encoder **(5 days)**

In Lesson 2.3, students exchange keys and messages and use Python® functions to encode and decode data. The encoders that students create may store data in any number of ways, from notes in a song to alpha values in an image or the movements of objects in a virtual environment on their screen.

Problem 2.3.1	Creating a Custom Encoder (CTP6) Responsible Computing (DAT) Computing Systems and Networks	(5 days)

Unit 3: Little Data to Big Data (39 days) (24%)

In Unit 3, In students dig deeper into **Data (DAT), Algorithms and Programming (AAP), Impact of Computing(s) (IOC)**. Students uncover patterns and gain meaning from large data sets. Students begin with small data sets and progress to larger ones as they examine how computing impacts today's society and helps to inform our decisions.

Lesson 3.1 Little Data **(20 days)**

In Lesson 3.1, students create a range of visualizations of small sets of data and find meaning in the patterns they uncover. Students learn that information is a collection of facts and patterns that they can extract from data. They explore how our world can be translated into digital representations to be collected, stored, and analyzed. Students use grade-level-appropriate statistics to deepen the meaning of knowledge gained through visualization.

Activity 3.1.1	Data Visualization: What's the "Point"?	(2 days)
Activity 3.1.2	Speculations in Sound	(4 days)
Activity 3.1.3	The Color of Light	(4 days)
Activity 3.1.4	CO2 Rising	(3 days)
Activity 3.1.5	Gaming with Force	(4 days)
Project 3.1.6	Rover Phone Home (CTP4) Code Analysis (DAT) Data	(4 days)

Lesson 3.2 Trendy Data **(14 days)**

As in the previous lesson, the goal of this lesson is for students to create a range of visualizations to analyze and interpret the patterns they uncover, this time using larger, complex sets of data. From the data, they draw conclusions relevant to themselves, including local weather, the economics of their community, and trends across the world. Students explore the wide range of data sets available today and begin to understand how claims can be made by examining correlation and causation.

Activity 3.2.1	Trends in Temperature	(3 days)
Activity 3.2.2	Shocking Data Trends	(4 days)
Activity 3.2.3	Pirates Are the Problem	(3 days)
Project 3.2.4	Making Meaning from Data **(CTP6) Responsible Computing** **(DAT) Data**	(4 days)

Problem 3.3 Making Predictions from Data **(5 days)**

Finally, students work in teams to choose a question or problem, making and supporting an argument using large sets of data.

Problem 3.3.1	Making Predictions from Data (CTP6) Responsible Computing (DAT) Data	(5 days)

College Board: Create Performance Task (12 days) (8%)

Students are provided with at least twelve (12) hours of class time to complete the AP Create Performance Task.

Throughout the course, students have the flexibility to write programs that reflect their interests (e.g., their desire to solve a problem, program a game, or produce digital art appealing to a specific audience). This allows students to engage in the study of computer science from a creative perspective.

During these 12 hours, students apply all they have learned to select an interest, develop a program, document the program, and submit to the College Board for scoring if they are seeking advanced placement standing. No new content is introduced during this time.

Students provide evidence of their knowledge regarding important programming concepts, such as developing algorithms and using abstractions. Students are required to submit an individual program but are able to collaborate on the development of their program. This performance task focuses on students developing computer programs and describing significant aspects of the program that allow it to run as intended.

Unit 4: Solving Complex Problems (24 days) (16%)

In Unit 4, Students examine on the current and future state of artificial intelligence and the ways in which artificial intelligence and simulation and modeling are impacting all fields highlighting **Data (DAT), Algorithms and Programming (AAP), and the Impact(s) of Computing (IOC)** Students identify problems and questions that can be addressed with computer simulations. Students are challenged to explore the assumptions and parameters built into several simulations and to attach meaning to the results.

Lesson 4.1 Simulating the Real World (12 days)

In Lesson 4.1, students explain how computers can be used to represent real-world phenomena or outcomes. They compare the use of simulations with real-world contexts. They begin by exploring modeling and simulations to study systems that are complex, dangerous, expensive, big, or even too small to easily observe otherwise.

Activity 4.1.1	Simulations in Science	(3 days)
Activity 4.1.2	Simulations to Predict Growth Rates	(2 days)
Activity 4.1.3	Simulations to Predict Behavior	(2 days)
Project 4.1.4	Understanding Complex Systems (CTP1) Computational Solution Design (IOC) Impact of Computing	(5 days)

Lesson 4.2 Future Innovations (9 days)

In Lesson 4.2, students explore computing innovations, such as machine learning, artificial intelligence, and cloud computing, by exploring the vast amount of tools and resources available through an AWS educate account. They also examine factors that contribute to the digital divide.

Activity 4.2.1	Machine Learning and AI	(4 days)
Project 4.2.2	Computing Exploration	(2 day)
Problem 4.2.3	Mobile Classroom (CTP2) Algorithms and Program Development (IOC) Impact of Computing	(3 days)

Lesson 4.3 Impacts of Computing Innovations (3 days)

Students select a computing innovation and create a digital artifact that describes the computing innovation's impact. They explore the legal, ethical, and unintended consequences of its use.

Problem 4.3.1	Impacts of Computing: Explore Task 3 (CTP6) Responsible Computing (IOC)Impact of Computing	(3 days)