PROJECT STEM

AP Computer Science Principles
Course Syllabus and Planning Guide

Curricular Requirements

Curricular Requirements		Page(s)
CR1	The teacher and students have access to college-level computer science resources, in print or electronic format.	3
CR2	The course provides opportunities to develop student understanding of the required content outlined in each of the Big Ideas described in the AP Course and Exam Description.	9
CR3	The course provides opportunities to develop student understanding of the Big Ideas.	12, 15, 18, 28, 28
CR4	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 1: Computational Solution Design.	21
CR5	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 2: Algorithms and Program Development.	15
CR6	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 3: Abstraction in Program Development.	18
CR7	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 4: Code Analysis.	21
CR8	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 5: Computing Innovations.	25
CR9	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 6: Responsible Computing.	28
CR10	The course provides a minimum of three opportunities for students to investigate different computing innovations.	28
CR11	Students are provided at least twelve (12) hours of dedicated class time to complete the AP Create Performance Task.	22

Introduction

AP Computer Science Principles (AP CSP) is a full-year, rigorous course that introduces students to the foundational concepts of computer science and explores the impact computing and technology have on our society. The course covers a broad range of foundational topics including: programming, algorithms, the Internet, big data, digital privacy and security, and the societal impacts of computing.

About the Course

Project STEM developed this course in partnership with the University of Texas at Austin's UTeach Institute. This custom course combines the esteemed UTeach CS Principles curriculum with additional features and tools specific for a technology-driven student-centered curriculum, including: instructional lesson videos and slides, worked practice problems, unit project scaffolding, student activity and task examples and grading rubrics, enhanced online and offline question banks with College Board-style questions, annotated explanations for all assessment questions, and a practice mini performance task. Additionally, UTeach's lesson plans have been substituted for lesson and unit guides, since they have been revised to focus less on teacher-driven directives for students ("say this," "do this," etc.) and more on teaching tips and strategies.

All schools using Project STEM's AP CSP course should use this syllabus.

Course Overview

Prerequisites
The College Board suggests students successfully complete a first year high school Algebra course prior to enrolling in AP CSP. An Algebra course will provide a strong foundation in problem solving, basic linear functions, composition of functions, and the Cartesian (x,y) coordinate system. These skills and topics are essential for student facility in this course. For further preparation, we recommend students complete our Computer Science Python Fundamentals course prior to taking this course. That course introduces students to the fundamentals of computing, providing a foundation on which this course can build.

The College Board adheres to an open enrollment policy for this course, meaning any student that is willing and academically prepared can participate in the course.

Pedagogical Approach
Project STEM's AP CS Principles course follows the blended learning model. It takes a student-centered approach powered by technology to help realize the goal of high achievement for all students. The course promotes student engagement, independent thought and interactive collaboration with peers. Student-centric lessons, activities and assessments are paired with augmentative teacher-centric lesson, activity and task guides and reporting to empower teachers to empower students. Additionally, teacher and student forums with moderation and input from Project STEM staff and team of teaching assistants provide dynamic community and support.

Programming Requirements
The coding languages Scratch and Python are both used in this course. Scratch is a free block-based programming environment that is accessible enough for beginners, yet can support the development of advanced algorithms used in more complex games and applications. Python is a text-based language with easy to read and write syntax - perfect for beginning programmers.

Course Goals

Project STEM's AP CSP course fully addresses the College Board's AP Computer Science Principles Curriculum Framework. The framework defines two through-course curricular requirements: six "computational thinking practices" and five "big ideas." Additionally, the framework describes in detail what students should be able to do, know, and retain by the end of the course with three types of expressions: Enduring Understandings, Learning Objectives, and Essential Knowledge Statements. A basic overview of each of these items is provided below, and we encourage instructors to read more about them in the AP Computer Science Principles Curriculum Framework.

Six Computational Thinking Practices

The six Computational Thinking Practices contain skills that students should develop to not just learn about content, but to change their way of thinking.

Computational Thinking Practices					
P1	P2	P3	P4	P5	P6
Computational Solution Design	Algorithms and Program Development	Abstraction in Program Development	Code Analysis	Computing Innovations	Responsible Computing

Five Big Ideas

The course material focuses on Five Big Ideas. These ideas encompass concepts that are foundational to computer science.

Big Ideas				
Big Idea 1 (CRD)	Big Idea 2 (DAT)	Big Idea 3 (AAP)	Big Idea 4 (CSN)	Big Idea 5 (IOC)
Creative Development	Data	Algorithms and Programming	Computing Systems and Networks	Impact of Computing

Enduring Understandings

Enduring Understandings (EUs) describe the concepts students should understand after going through this course. The goal is for Learning Objectives and Essential Knowledge Statements to build Enduring Understandings.

Learning Objectives

Learning Objectives (LOs) articulate what students should be able to do by the end of the course. Each learning objective corresponds to one of the Five Big Ideas and one part of a computational thinking practice. Both the multiple choice exam and through-course performance task test students' mastery of these learning objectives.

Essential Knowledge Statements

Essential Knowledge Statements (EKs) provide facts or concepts students should know to prove their understanding of the learning objectives.

The AP Exam

The AP Exam will test students on their understanding of the five big ideas through a multiple-choice exam and one through-course performance task. Together, these components will be used to calculate the AP score (on a 1-5 scale).

Multiple Choice Exam
The 70-question multiple choice exam will test students' understanding of computational logic, which they will learn over the course of the year. This section is programming language agnostic, meaning students don't have to know a formal coding language to complete this part of the exam. The multiple-choice exam will be in May (exact date can be found on the College Board website) and accounts for 70% of a student's total AP score.

Performance Task
The performance task in this course is called the Create Task. This task functions as a project that students must complete independently and submit online prior to taking the multiple-choice portion of the exam. The Create Task is worth 30% of a student's overall AP score. In this task, students will create their own program. Students will submit a video of their program running and a written response describing how their program works. The students must be given a minimum of 12 hours in class to work on it.

Students are required to submit their performance task via the College Board's online Digital Portfolio. You can find the due date for the Create Task here.

Course Materials

Project STEM's AP Computer Science Principles has an introductory unit, six instructional units, a mini performance task module, and a final AP review unit. Each is strategically designed to prepare students for the AP CSP Exam. The course units consist of daily lessons, instructional videos, lesson slides, lesson activities, code-along exercises, projects, vocabulary reviews, AP test preparation, quizzes, and tests.

Student Lessons
The student lessons are typically composed of the following components:

- Introduction: a high-level overview of the lesson.
- Objectives: a list of what students will learn and do during the lesson.
- Instructional Video(s): one or more explanatory or demo videos taught by an expert computer science teacher, some of which include code-along activities. Most videos are accompanied by corresponding downloadable slides for review or note taking.
- In-Lesson Activities: these activities take place during the lesson, prior to the graded exercises, such as a class discussion, interactive, or code-along activities.
- Summary: a text version of the key concepts in the lesson.
- Vocabulary: a list of terms and definitions for the lesson.
- Lesson Exercises: one or more graded exercises that ask students to apply or extend the concepts in the lesson. Lesson exercises include coding activities, discussions, research, strategic games, computational practices (non-coding), and more. The variety of formats and tasks prepare students for the diverse questions and tasks on the AP CSP Exam.
- AP-Style Practices: a practice activity where students complete a set of questions in the format and style of the AP exam.

Other Assignments
In addition to the daily lessons and exercises, the AP CSP curriculum also offers other types of assignments for students.

- Vocabulary Practices: Each unit has a vocabulary practice that helps students to reinforce the unit's keywords. These practices are game-based and allow for several different types of practice formats like matching and flashcards. The vocabulary practices are not a graded assignment.
- Big Picture Exercises: classroom investigations or discussions that examine the cultural and societal impact of emerging technologies.
- Unit Projects: Five of the six units include an extended project that challenges the students to apply various concepts from the current and past units in a new or more complex way. These projects demand a high level of critical thinking and problem solving.
- Unit Review Activities: Each unit has an autograded review activity where students complete a set of ~20 multiple choice questions to prepare for the unit test.
- A Mini Performance Task: The course includes a practice performance task that helps prepare students for the official AP CSP Create Task. This is a multi-day project that mirrors the types of tasks and activities that students must complete to meet the College Board requirements for the official task.

Assessments

The AP Computer Science Principles course offers two types of assessments: quizzes and tests.

- Quizzes: Each unit has two short quizzes that act as a checkpoint for understanding. These quizzes range from 4-7 multiple choice questions with shuffled answers.
- Tests: Each unit has a summative test at its conclusion. The tests are always 20 multiple choice questions with shuffled answers.

Grading

For Project STEM’s AP CSP course, there are several types of formative and summative assessments, all intended to prepare students for the end-of-year 70-question multiple choice exam (70% of the overall AP score) as well as the through-course Create performance task (30% of the overall AP score).

The default course grading scheme maps to this 60/40 breakdown:

- 10% Unit Quizzes
- 50% Unit Exams
- 10% Lesson Exercises and Activities
- 10% Mini Performance Task
- 20% Unit Projects

Teacher Sidebar

The Teacher Sidebar, which is located within a teacher’s version of the student course, contains several types of resources that will help you in facilitating the course, including:

- Lesson guides that detail the lesson objectives, lesson components, as well as indicators of key points to emphasize and common misconceptions
- Supplemental resources, such as worksheets or unit project rubrics
- Alternative assessments, such as paper-based and alternate versions of quizzes and tests for each unit
- Answer keys, including annotated solutions to quizzes and tests

Course Sequencing

The year-long curriculum directly addresses the College Board's AP Computer Science Principles curriculum framework. It has been carefully designed to teach students the core skills for 1) creating and using computational tools 2) applying logical reasoning and creative problem solving and 3) recognizing real-world applications for digital technology. As described above, it is comprised of an introductory unit, six instructional units, one mini module for performance task preparation, and a final unit focused on the course's AP exam. The curriculum also provides one window of time for students to complete the required Create Task.

The sequencing and a high-level description of all components is outlined below:

Content Overview	
Unit 0: Course Introduction Enter the world of computer science by learning about the field itself and the goals of this AP-level course.	**Big Ideas:** CRD, AAP, IOC
Unit 1: Computational Thinking Study the iterative development process, and start applying it to build your own programs in *Scratch.*	**Big Ideas:** CRD, AAP
Unit 2: Programming Examine computational logic structures and problem solving capabilities for programs in text-based algorithms, AP-style Pseudocode, and *Scratch*.	**Big Ideas:** CRD, AAP, CSN, IOC
Unit 3: Data Representation Explore the different means of representing information digitally.	**Big Ideas:** CRD, DAT, AAP, IOC
Mini Create Task Module Learn about the Create Performance Task component of the AP exam, and practice the skills required for it.	**Big Ideas:** CRD, AAP
Unit 4: Digital Media Processing Use Python to programmatically manipulate digital images and audio.	**Big Ideas:** CRD, DAT, AAP, IOC
Create Performance Task Students demonstrate their learning by creating a portfolio of their work for submission to the College Board.	**Big Ideas:** AAP
Unit 5: Big Data Discover new knowledge through the use of large data sets.	**Big Ideas:** CRD, AAP, IOC
Unit 6: Innovative Technologies Assess the current state of technology and investigate its role in our everyday lives.	**Big Ideas:** DAT, AAP, CSN, IOC
Unit 7: The AP Exam Students review and prepare for all components of the AP Exam.	**Big Ideas:** CRD, DAT, AAP, CSN, IOC

Unit 0: Course Introduction

Unit 0 is an introduction to the AP Computer Science Principles course. The unit exposes students to the foundational topics of computer science and computing. Additionally, it introduces the major topics and components on the AP exam, so students will become familiar with the big ideas and computational thinking practices around which the course is focused. Before finishing the unit, students will engage with their preconceived notions about computer science and challenge these ideas.

Unit 0 Schedule

Topic	Lesson
Course Context	Welcome to AP Computer Science Principles
	Computer Science Fundamentals
	Course Structure
Course Resources	Student Forum
	Forum Guidelines
	Honor Code
Self-Evaluation	Entry Questionnaire

Unit 0 Topics

Course Context

- Students will examine and discuss the motivations behind a number of high-profile individuals in the field of programming.
- Students will discuss the benefits of programming as a tool and a profession.
- Students will discuss the impact computing has on society, business, and the economy.
- Students will examine the ideas of computational thinking and computational artifacts.

Course Resources

- Students will become familiar with the resources on the Project STEM platform.

Self-Evaluation

- Students will consider their relationship with computer science and programming.

Unit 1: Computational Thinking

This unit lays the foundation for computational logic. Students first explore the iterative development process, seeing how an idea translates to a real, functioning program. Then, they take a closer look at this process by examining algorithms, languages, program execution, and the through-course concept of abstraction. For the second half of the unit, students get started coding in Scratch. Using this visual, block-based programming language, they learn basic programming concepts and constructs, including user input and variables. In creating programs of their own, they have the opportunity to apply the iterative development process. Over the course of the unit, students learn how to build computational artifacts and solve computational problems - two skills essential to the rest of the course.

There are no major projects in this unit, but there are several post-lesson opportunities for students to apply the iterative development process and basic programming concepts.

Unit 1 Schedule

Topic	**Lesson**
Program Development	The Iterative Development Process
	Algorithms
	Languages
	Idea to Execution
Big Picture	Collaboration
Visual Programming	Getting Started in Scratch
	Programming with Blocks
Program State	User Input and Storage
	Defining Variables
	Applying Variables

Unit 1 Topics

Program Development

- Students will examine strategies for approaching large-scale problems.
- Students will explore the non-linear approach to solving problems with the iterative development process.
- Students will identify a number of common features of algorithms, including sequencing, selection, and repetition.
- Students will design and evaluate text-based algorithms.
- Students will examine the need for clarity and precision in communicating an algorithmic solution to a problem.
- Students will examine the shortcomings and ambiguities of natural languages.
- Students will identify the elements of clear communication, including well-specified grammar, vocabulary, and syntax.
- Students will analyze the need for artificial programming languages.
- Students will compare high-level languages with low-level languages.
- Students will examine the process in which a program is written in a high-level language, compiled into a low-level language, loaded into memory, and then executed by a processor.

Big Picture

- Students will examine the benefits of working collaboratively.

Visual Programming

- Students will utilize a graphical editor to read, construct, and execute dynamic programs.
- Students will examine, modify, and execute programs developed by others.
- Students will examine how well-specified behavior of objects can be constructed through sequential actions and operations.
- Students will examine a number of common programming errors.
- Students will explore a number of common debugging strategies.
- Students will develop solutions for correcting common programming errors.

Program State

- Students will write programs that incorporate dynamic, user-driven, keyboard controls and input.
- Students will examine how the dynamic state of an object or program can be stored and changed using variables.
- Students will analyze the role of clear, descriptive names for objects, behaviors, variables, and other identifiers in maintaining the readability of code.
- Students will analyze and evaluate the correctness of their programs.

Unit 1 Highlighted Activities

Throughout the entire unit, students get a chance to complete many activities called “Try it Out!” during the lessons. In these activities students are tasked with exploring new bits and pieces of code and discovering how they can be applied in order to create successful algorithms. As part of this process, they are investigating, reflecting, designing, prototyping and testing their programs, all of which are part of the creative development process. **[CRD]**

Unit 2: Programming

This unit focuses on the three main control structures utilized within algorithms and programs: sequencing, selection, and iteration. Students first examine these structures conceptually, and then learn how to formally construct and evaluate them in Scratch and AP-style Pseudocode. In doing so, they hone their programming abilities and become familiar with the importance of precise commands and well-structured logic. Building on this knowledge, students explore how abstraction can be applied to algorithmic solutions using procedures, and examine 1) how algorithmic solutions should be efficient and help programs scale and 2) what happens when a problem is not able to be solved with an algorithm. At the end of the unit, students get a glimpse of how design documentation for hardware components employs computational logic and abstraction just like programming.

There is one major project in this unit: the Password Generator Project.

Unit 2 Schedule

Topic	Lesson
Control Structures	Defining Sequencing
	Applying Sequencing
Coding Skills	Pseudocode
Control Structures	Defining Selection
	Applying Selection
	Defining Iteration
	Applying Iteration
Procedural Abstraction	Procedures
Decidability and Efficiency	Solvability & Performance
Big Picture	Moore’s Law
Hardware Abstraction	Logic Gates & Hardware
Unit Project	Password Generator Project

Unit 2 Topics

Control Structures
- Students will examine a number of common features of algorithms, including sequencing, selection, and repetition.
- Students will examine how well-specified behavior of objects can be constructed through sequential actions and operations.
- Students will examine the uses of selection statements in programming.
- Students will analyze the differences between simple selection and complex, nested selection statements.
- Students will examine the use of the Boolean operators "AND," "OR," and "NOT" in constructing complex conditional statements.
- Students will examine the uses of iteration statements in programming.
- Students will consider how to make a sequence of events more efficient with iteration statements.
- Students will combine sequencing, selection, and repetition structures alongside programming constructs like user input and variables to create computational artifacts.

Coding Skills
- Students will examine how pseudocode can outline algorithmic processes.
- Students will read, execute, and construct algorithms in AP-style pseudocode.

Procedural Abstraction
- Students will compare the methods and relative efficiencies of different algorithms.

Decidability and Efficiency
- Students will examine the factors that affect the decidability of a problem.
- Students will identify which problems can and cannot always be solved by an algorithm.
- Students will examine methods of comparing equivalent algorithms for relative efficiency.
- Students will evaluate the relative efficiency of equivalent algorithms.
- Students will identify factors that allow solutions to scale efficiently.

Big Picture
- Students will examine the implications of Moore's Law on the research and development of new and existing technologies.

Hardware Abstraction
- Students will explore the logical processes implemented in hardware design documentation.

Unit 2 Highlighted Activities

Throughout the unit students are working on activities where they create algorithms in Scratch to accomplish a specific purpose. They are learning how to implement selection and iteration structures in combination with sequencing to create a solution to a program. This builds to students completing a mini-project (choosing from Drawing a Picture, an Electronic Keyboard, or a Countdown timer) where they demonstrate their ability to use the Scratch programming language to implement algorithms in a program. **[P2]**

Unit Project **[AAP]**
The Password Generator Project occurs after all lesson components, and is a collaborative, in-class activity. In this Scratch programming project, students will explore data security considerations and develop a program for generating unique, secure passwords. Students will:

- Design an algorithm for generating a custom, reproducible password that is uniquely different for each website (e.g., using the domain name as a seed, etc.).
- Write pseudocode to describe each step of the algorithm used to generate a password.
- Exchange algorithms with peers and share feedback with each other on the clarity of the pseudocode and the strengths and weaknesses of the algorithms.
- Construct trace tables documenting the result of each step of the algorithm in generating passwords for different domains.
- Design code in Scratch to implement the password-generating algorithm.

Unit 3: Data Representation

In this unit, students explore the different ways digital information can be represented, stored, and manipulated on a computer. They look at the various levels of abstraction that are used in the digital representation of discrete data and information. Initially, students will focus on the lowest levels of digital representation and storage by examining different base representations of numbers (including decimal and binary) and their application to ASCII and Unicode character encoding. Next, they will examine the distinctions between analog and digital forms of representation. Finally, students will learn about lists, a common abstract data type that can be utilized in programs. They will explore the characteristics of lists and how they can be used to search and sort data.

There is one major project in this unit: the Unintend'o Project.

Unit 3 Schedule

Topic	Lesson
Binary Encoding of Information	Binary
	Base Conversions
	ASCII vs. Unicode
Coding Skills	Programming Binary
Digital Approximations	Digitization
	Analog vs. Digital Data
Big Picture	Reselling Digital Music
Lists	Making a List
	Processing a List
	Sorting a List
	Lists in Pseudocode
Unit Project	Unintend'o Project

Unit 3 Topics

Binary Encoding of Information

- Students will examine how numerical values are represented using different bases, including decimal and binary.
- Students will explore methods of converting values from decimal to binary and binary to decimal.
- Students will examine the exponential relationship between the number of digits and their range of representable values.
- Students will examine how alphanumeric characters and symbols may be represented using ASCII and Unicode character mappings.
- Students will analyze the differences in state space between ASCII and Unicode standards.
- Students will explore how the interpretation of binary data is dependent upon its intended format and use, including base-64, bitmaps (*.BMP), plaintext (*.TXT), audio (*.MP3), etc.

Coding Skills

- Students will construct a Scratch program that simulates candles on a birthday cake being lit so as to show the user's age in binary.

Digital Approximations

- Students will examine the implications of variable-width encodings (e.g., Morse code) versus fixed-width encodings (e.g., Baudot code).
- Students will explore ways in which natural phenomena may be represented digitally.
- Students will analyze the extent to which digital approximations accurately reflect the reality that they represent.
- Students will analyze the differences between discrete (digital) and continuous (analog) representations of natural phenomena.
- Students will examine the social implications of the ease with which perfect digital copies can be made.

Big Picture

- Students will examine and discuss the legality of reselling "used" digital music.

Lists

- Students will examine the use of lists as ordered data structures that may contain multiple values.
- Students will investigate the use of index values to represent the position of an item in a list.
- Students will analyze the implications of accessing an index position beyond the bounds of a list.
- Students will investigate common operations for processing elements of a list, including searching for an element, removing an element, swapping the positions of two elements, or sorting an entire list into ascending or descending order.
- Students will examine the implications of case-sensitivity on ordered lists of strings.
- Students will consider how lists can appear in pseudocode.

Unit 3 Highlighted Activities

Throughout the unit students get a chance to learn about data abstraction as they create and implement algorithms to make, process and sort lists. They also learn about binary, ASCII, hexadecimal and how bits play a role in digitization of different types of digital and analog data.

Unit Project **[DAT]**
The Unintend'o Project is a collaborative, culminating activity positioned at the end of the unit. In this Scratch programming project, students will write a program that directs the input of a video game controller. It exposes how bits and binary can work to turn on and off functionalities within programs. Students will:

- Map each of six controls (UP, DOWN, LEFT, RIGHT, A, and B) to individual bits.
- Map each binary pattern of button presses to different game actions (e.g., walk forward, walk backward, turn left, turn right, jump, duck, whirl, leap, crawl, etc.).
- Use a list to track the history of button presses.
- Write detailed specifications and justifications for each button-to-action mapping of your design.
- Collaborate with peers throughout the design and development process to determine end-user requests for features and to share feedback on design and implementation strategies.
- Write documentation detailing the use of the program and its features using appropriate terminology.
- Develop a Scratch program that acts as a device driver for a video game controller interface.

Mini Create Task Module

This mini-performance task module is a multi-day activity that gives students a chance to deepen their understanding of the AP CSP Create Task. They begin by exploring the requirements of the task itself. Then, they move on to evaluate sample student submissions against the official College Board rubric. After this, students work on a mock create task, learning how to fulfill the project requirements themselves. Students will:

- begin designing their own program using the iterative development process.
- record a video of the program running.
- provide a written explanation of the purpose, process, algorithms, and abstractions in their design.
- practice submitting their work in the format College Board requires.
- review a peer's work and provide feedback, based on the official Create Task rubric.

After completing this activity, students will be prepared for the Create Task later in the course. **[P3]**

Unit 4: Digital Media Processing

In Unit 4, students will use Python to programmatically manipulate digital images and audio. The unit starts by guiding students through the transition of programming in Python, which is a high-level, procedural, text-based language. In Python, students will explore the characteristics of the RGB color model and its use in encoding digital images. They will also investigate the methods of representing and modifying digital audio, including Auto-Tune and audio compression. The unit concludes with a summary of the compression methods related to digital media processing.

There is one major project in this unit: the Image Filter Project.

Unit 4 Schedule

Topic	Lesson
Introduction to Python	Scratch vs. Python
	Python Basics
Control Structures	Selection Structures
	Iteration Structures
Abstractions	Data Abstraction
	Procedural Abstraction
Image Manipulation	RGB Color
	Image Manipulation
	Encoding Schemes
	Digital Manipulation
Big Picture	Ethics of Digital Manipulation
Big Picture	Intellectual Property
Audio Manipulation	Audio Manipulation
	Audio Processing
	Audio Compression
Unit Project	Image Filter Project

Unit 4 Topics

Introduction to Python

- Students will explore the capabilities of a text-based programming language (Python).
- Students will compare and contrast the programming capabilities of a visual programming language (Scratch) with those of a text-based programming language (Python).
- Students will understand the importance of using proper punctuation and syntax when coding in a text-based programming language.

Control Structures

- Students will write code using common programming constructs like conditional if() for selection and while() loops for iteration.
- Students will use boolean, relational and conditional expressions.

Abstraction

- Students will write code using data abstraction (lists).
- Students will create and use procedural abstractions in order to make their programs more readable and versatile.

Image Manipulation

- Students will examine the structure of raster images as compositions of individual pixels.
- Students will explore various methods of representing color, including RGB, CMYK, and HSV.
- Students will explore the various colors that can be produced by the combination of different ratios of red, green, and blue light.
- Students will perform base conversions for decimal, binary, and hexadecimal number systems.
- Students will modify the color channels of pixels in an image to produce a variety of effects.
- Students will design algorithms for modifying the pixels in an image in prescribed ways to create custom image filters.
- Students will explore the difference between lossy and lossless encoding schemes of several common image file formats.

Big Picture

- Students will explore the positive and negative consequences of digitally altering images.
- Students will discuss the ethics of digitally manipulating images, especially in the context of journalism.
- Students will discuss the issues related to intellectual property.
- Students will explore the limitations and rights associated with a number of common licenses, including Creative Commons.

Audio Manipulation

- Students will analyze the differences between analog and digital sound.
- Students will explore the roles that sampling rate and bit depth play in determining the quality of digitized sound.
- Students will explore methods of programmatically generating digital audio.
- Students will explore methods of programmatically altering and modifying digital audio by adjusting volume, pitch, and sampling rate.
- Students will explore the methods and effects of compression algorithms in reducing the amount of data needed to represent an audio sample.

Unit 4 Highlighted Activities

Throughout this unit students complete activities where they are creating programs that implement algorithms. Specifically, there is a program that is similar to “Mastermind” part way through the unit. In this program students are developing an in-depth program that utilizes all of the coding skills they have gained up until this point - selection, iteration, lists, functions, input, output, randomness, and more. Throughout the process of creating their program, they are testing it using different inputs from the user and seeing if they are getting the expected results. As part of their programming process, students are to document their code to explain what each code segment will do in their program. **[P4]**

Unit Project **[P1]**
The Image Filter Project is an in-class, collaborative activity that occurs at the end of the unit. In this Python programming project, students will use their text-based programming skills to develop a program that manipulates digital images similarly to a filter in a photo app. Students will:

- design and implement a program for filtering digital images.
- develop code to systematically transform an image by mathematically manipulating its bits, pixel by pixel.
- write documentation detailing the use of the program and its features using appropriate terminology.
- explain the design and implementation choices by demonstrating and sharing the finished programs with peers.

Create Performance Task

This section serves to fulfill the Performance Task requirements of the AP Computer Science Principles exam. The Create Performance Task will account for 30% of the student's AP exam score. As such, the work produced in this unit should reflect the sole work of the student and performed in-class with minimal involvement from the classroom teacher. During this performance task, students will demonstrate their ability to work collaboratively and individually to design and develop a functional program for solving a problem and/or self-expression.

Create Performance Task Schedule

Topic	**Tasks**
Create – Applications from Ideas 12 hours of class time required	Identify Project Ideas
	Develop, Implement, and Test Program
	Create Video of Program
	Write Responses on Program
	Submit “Create” Task Program, Video, and Written Responses

Create Performance Task Topics

Creative Development

- ○ Students will individually and/or collaboratively design, implement, and test a program designed to solve a problem of interest to them.
- ○ Students will document the functionality of their program and reflect on its development process.

Create – Applications from Ideas Performance Task

- ○ This project will encompass 12 hours of in-class, independent and/or collaborative work.
- ○ Each student will design, implement, and test a program that solves a problem of personal interest to the student.
- ○ Each student will describe and reflect on their role in the development of the program.
- ○ Students will make a one-minute video demonstrating the use and functionality of the program.
- ○ Students may work collaboratively on their project, but each student will be solely responsible for developing at least one significant part of their program.
- ○ The product of this project, including the program, video, and written responses, will serve as part of the student's formal submission to the College Board for the AP Computer Science Principles exam.

Unit 5: Big Data

One of the most powerful applications of computational thinking relates to the creation and analysis of large datasets. In this unit, students will explore the complete set of processes and techniques that are involved in collecting large volumes of raw data and extracting new and useful information. Students will look at a variety of ways that data scientists use techniques such as statistical analysis, data mining, clustering, classification, automated summarization, modeling and simulation to construct and visualize new knowledge. And finally, using these techniques themselves, students will perform their own analysis on a sample data set to discover new insights, which they will share with the class through a formal presentation.

The final activity described above is the one major project in this unit: the TEDxKinda Project.

Unit 5 Schedule

Topic	Lesson
Data Science	Introduction to Big Data
	Usability and Usefulness of Data
Data Aggregation	Collection
	Extraction
	Data Storage and Persistence
Big Picture	Wisdom of the Crowd
	Data Breaches
Data Analysis	Statistical Analysis
	Data Mining
Models and Simulations	Models and Simulations
Unit Project	TEDxKinda Project
Supplemental Data Analysis	Clustering
	Anomaly Detection
	Regression
	Classification
	Automatic Summarization

Unit 5 Topics

Data Science

- ○ Students will relate the impact of computing to ubiquitous and large-scale data processing.
- ○ Students will explore the ways that patterns within large data sets can be used in a predictive manner.
- ○ Students will discuss the risks and benefits of drawing conclusions from patterns found in large data sets.
- ○ Students will combine visuals, content knowledge, and interaction to create a dynamic infographic that clearly communicates discrete information about a data set.
- ○ Students will identify the characteristics that differentiate usable data from unusable data.
- ○ Students will identify the characteristics that differentiate useful data from useless data.

Data Aggregation

- ○ Students will explore the purposes of various processing tasks, including collection, knowledge extraction, and data storage.
- ○ Students will identify multiple techniques for data collection, both on and off of the Internet.
- ○ Students will analyze the characteristics of structured and unstructured data.
- ○ Students will extract structured information from unstructured data.
- ○ Students will examine methods of extracting information from online sources, including structured and unstructured search engines, screen scrapers, and spiders.
- ○ Students will explore the basic features and functionality of modern relational databases.
- ○ Students will debate the implications of large-scale data storage and data persistence on privacy and utility, including the costs associated with each.

Big Picture

- ○ Students will apply the technique of crowdsourcing to a novel data collection problem.
- ○ Students will examine the security risks and responsibilities assumed by companies that collect and store sensitive personal data.
- ○ Students will examine the causes and impact of data breaches involving sensitive personal data.

Data Analysis (including Supplemental)

- ○ Students will analyze the tradeoff of utility and confidence in descriptive, predictive, and prescriptive data analysis.
- ○ Students will investigate traditional statistical hypothesis testing and exploratory data analysis.
- ○ Students will investigate the use of data mining in the discovery of patterns in large data sets.
- ○ Students will examine the use of cluster analysis, anomaly detection, regression analysis, and data classification in the processing of large data sets.
- ○ Students will use automatic summarization tools to create computer-generated summaries of a large data set.

Models and Simulations

- ○ Students will use models and simulations to represent phenomena.
- ○ Students will explore how models may use different abstractions or levels of abstraction depending on the objects or phenomena being posed.
- ○ Students will utilize models and simulations to formulate, refine, and test hypotheses.
- ○ Students will examine how simulations mimic real world events without the cost or danger of building and testing the phenomena in the real world.

Unit 5 Highlighted Activities

Throughout the unit, students focus on big data and explore how it is collected, extracted, stored and processed. Each step along the way students complete activities where they discover different computational tools that can help them analyze and interpret their data - tools like Google trends and correlate, Microsoft Excel, Wordle, HeatMapTool, Google Docs. Through these tools, students mine for data and draw real conclusions.

Unit Project **[P5]**
The TEDxKinda Project is a collaborative activity in this unit. In this data analysis project, students will work together to select and analyze a large data set, then develop a TED-style presentation to present the implications of that data. Students will:

- o collaborate in groups to analyze public data sets and extract insightful information and new knowledge using a number of big data analysis techniques and tools.
- o evaluate and justify the appropriateness of the chosen data set(s).
- o construct informative and aesthetically pleasing data visualizations.
- o write a script and prepare speaker notes for a formal presentation of the findings.
- o cite all online and print sources used in the research and presentation preparation.
- o deliver a TED-style presentation discussing the data analysis and findings using appropriate terminology.

Unit 6: Innovative Technologies

This unit aims to broaden students' awareness of the computing tools they use and rely on every day and to encourage them to start thinking about the decisions and processes that go into the creation of these technologies. Students will begin by exploring many of the key roles that technology plays in their lives, including social networking, online communication, search, commerce, and news, examining the ways these ever-evolving technologies have impacted individuals and societies in recent years. With so many of these technologies relying on the Internet to connect users and data across varied and remote locations, the students will then "take a peek under the hood" to examine the systems and protocols that make up the global infrastructure of the Internet. Students also take a look at the past, present and future of technology and imagine the role that new innovations might play in shaping their future.

There is one major project in this unit: the Exploring Computing Innovations Project.

Unit 6 Schedule

Topic	Lesson
Big Picture	Defining a Computing Innovation
Implications of Computing	Global Impact
	Impact of Internet Access
	Cloud Computing
Big Picture	The Digital Divide
The Internet	Internet in Action
	Communication Protocols
	Internet Protocols
Cryptography	Encryption
Big Picture	Net Neutrality
Cybersecurity	Cybersecurity
Interconnectedness in Computing	World Wide Web
	Distributed Computing
	Internet of Things
	Ethics of Autonomous Technology
Unit 6 Project	Exploring Computing Innovations

Unit 6 Topics

Big Picture

- Students will examine computing innovations and consider their impact on the economy, society, culture and environment.
- Students will investigate the socioeconomic causes and effects related to the digital divide.
- Students will discuss the benefits and risks of open versus closed platforms.

Implications of Computing

- Students will explore the ways that innovations in digital technology can impact the lives of individuals and communities.
- Students will analyze the role that digital technology plays in their everyday lives.
- Students will analyze the role that digital technology plays in their social communications and interactions.
- Students will explore the impact that instant access to global search, news, and information has had on individuals and communities.
- Students will analyze the benefits and risks of cloud computing.

The Internet

- Students will examine the overall design and architecture of the Internet.
- Students will explore the role of servers, routers, gateways, and clients.
- Students will examine the domain name system and its role in network routing.
- Students will examine a number of standard network protocols, including IP, TCP, UDP, SMTP, HTTP, and FTP.
- Students will investigate the series of components and events that are involved in the transmission of an email or SMS text over the network.
- Students will investigate the series of components and events that are involved in the transmission of an HTML request from a Web browser.

Cryptography

- Students will identify the needs and applications of cryptography in our digital world.
- Students will encode and decode messages using common cryptographic techniques.
- Students will examine the mathematical foundation of cryptography.
- Students will analyze the differences between symmetric (single-key) encryption and asymmetric (public key) encryption.
- Students will examine the features of open and closed platforms and consider the role cryptography plays in systems security.

Cybersecurity

- Students will examine a number of common threats to cybersecurity, including distributed denial of service attacks (DDoS), phishing, viruses, and social engineering.
- Students will identify the needs for robust cybersecurity.
- Students will analyze the software, hardware, and human components of cybersecurity.
- Students will analyze the function and effectiveness of common cybersecurity solutions, including antivirus software and firewalls.

Interconnectedness in Computing

- Students will investigate the origins and applications of the World Wide Web.
- Students will analyze the impact of hyperlinked documents on how individuals find, acquire, and learn new information.
- Students will analyze the legal, social, and commercial impact that the World Wide Web has had on society.
- Students will examine the roles and applications of distributed computing.
- Students will investigate and extrapolate from recent advances in computing to make predictions about the capabilities of future technologies.
- Students will analyze how future technologies might impact individuals and societies.
- Students will examine the legal and ethical implications of autonomous technology.

Unit 6 Highlighted Activities

In this unit students are asked to do a few different activities in regards to fault tolerance of the internet, protocols used by the World Wide Web and routing, distributed, parallel and sequential computing. They also complete activities where they define and identify computing innovations. In these activities, students are asked a series of questions to prompt in depth analysis of the topics at hand:

- In one specific activity, students are asked to look at a network of nodes and determine the path in which packets might be routed based on specific protocols. Students then create their own protocol to implement on the network. **[CSN]**
- In another activity, students research different types of malware and discuss ways in which they can be prevented, so as to be aware of how to be safe and secure when using computing devices. **[P6]**
- In another activity, students review an image and are asked if they see any computing innovations. After going through the possible computing innovations in the image and why they were considered as such (talking about data they use and / or programs being fundamental to their function) then students are asked to brainstorm a list of their own computing innovations. Not only do they list why these innovations are computing innovations, but they also explain what the world would be like without it - how it impacts the society, economy or culture. **[CI 2, Prompt A]**
- In another activity, students review several computing innovations like search tools, wikis, e-commerce, etc. and are asked to reflect on how the innovation inputs, transforms and outputs data. They further explore and explain a beneficial or harmful effect that this innovation may have on economy, society or culture. **[CI 3, Prompts A and B]**

Unit Project **[IOC] [CI 1, Prompts A, B, and C]**
This multi-day activity gives students a chance to deepen their understanding of computing innovations. Students will research computing innovations and explore multiple aspects of them. Students will create a computational artifact to display information they have learned as well as provide written responses to prompts dealing with ideas like:

- the function and purpose of a computing innovation
- how the computing innovation was developed and created
- the beneficial and harmful effects the computing innovation may have had on society, economy or culture
- how the innovation uses, consumes or transforms data
- how the innovation may have been used beyond the intended purpose

Unit 7: The AP Exam

This final unit provides preparation resources for all components of the AP exam. Students review the design of the assessment and work through a practice multiple choice sequence taken from the College Board’s AP Computer Science Principles Course & Exam Description document. Students also have access to resources that support the completion and submission of the Performance Task requirements. As students practice and prepare for the exam with this module and other past curricular content, they should consider each of the five big ideas and six computational thinking practices central to this course.