CodeHS

AP Computer Science Principles in Python Course Syllabus
One Year for High School, 125 Hours

Introduction

AP Computer Science Principles introduces students to the foundational concepts of computer science and explores the impact computing and technology have on our society.

With a unique focus on creative problem solving and real-world applications, the CodeHS AP Computer Science Principles course gives students the opportunity to explore several important topics of computing using their own ideas and creativity, use the power of computing to create artifacts of personal value, and develop an interest in computer science that will foster further endeavors in the field.

Course Overview

Technology Requirements: To complete all activities and exercises in this course, students must have access to the 3rd party sites and tools listed here: AP Computer Science Principles in Python Course Links

Prerequisites: There are no official prerequisites for the CodeHS AP Computer Science Principles course. This course is meant to be a first-time introduction to computer science and does not require students to come in with any computer programming experience. However, we recommend that students take our Introduction to Computer Science prior to our AP courses (more info at codehs.com/library). Students who have completed our Intro to CS course will be able to apply knowledge of concepts covered in the Intro course to the more advanced setting of the AP courses. We also recommend that students complete a first-year high school algebra course prior to taking this course. Students should be comfortable with functions and function notation such as $f(x) = x + 2$ as well as using a Cartesian (x, y) coordinate system to represent points in a plane.

Overarching Goals:

- Increase and diversify participation in computer science
- Students, regardless of prior experience in computing, will develop confidence using computer science as a tool to express themselves and solve problems, and this confidence will prepare them for success in future endeavors in the field of computer science
- Students will understand the core principles of computing, a field which has and continues to change the world
- Students will be able to develop computational artifacts to solve problems, communicate ideas, and express their own creativity
- Students will be able to collaborate with others to solve problems and develop computational artifacts

- Students will be able to explain the impact computing has on society, economy, and culture
- Students will be able to analyze existing artifacts, identify and correct errors, and explain how the artifact functions
- Students will be able to explain how data, information, or knowledge is represented for computational use
- Students will be able to explain how abstractions are used in computation and modeling
- Students will learn to be informed and responsible users of technology

Learning Environment: The course utilizes a blended classroom approach. The content is a mix of web-based and physical activities. Students will write and run code in the browser, create websites and digital artifacts, and engage in in-person collaborative exercises with classmates. Teachers utilize tools and resources provided by CodeHS to leverage time in the classroom and give focused 1-on-1 attention to students. Each unit of the course is broken down into lessons. Lessons consist of video tutorials, short quizzes, example programs to explore, written programming exercises, free response exercises, collaborative creation projects, and research projects.

Programming Environment: Students write and run programs in the browser using the CodeHS editor. Students will be able to write text-based Python programs, and students will use a graphics library to create Python graphical programs. Students gain programming experience early on in the course that will enable them to explore the rest of the course topics through computational thinking practices.

Course Resources: Access to a computer and high-speed internet is required. There is also an online textbook available for many modules and topics which can be accessed through the lesson plans or at https://codehs.gitbooks.io/introcs/content/

Quizzes: At the end of most units, students take a summative multiple choice unit quiz in the style of the AP Exam that assesses their knowledge of the concepts covered in the unit. The course also provides an AP Test Practice unit with a cumulative AP Practice Multiple Choice Test.

Course Objectives

This course is based directly on the College Board AP Computer Science Principles Framework. We recommend reading the curriculum framework here for context. The main course objectives are summarized below in the six computational thinking practices and five big ideas for the course.

Computational Thinking Practices:
The six computational thinking practices represent important aspects of the work that computer scientists engage in, and are denoted here by P1 through P6:

- **Practice 1: Computational Solution Design**
    - *Design and evaluate computational solutions for a purpose.*
- **Practice P2: Algorithms and Program Development**
    - *Develop and implement algorithms.*

- **Practice P3: Abstraction in Program Development**
    - *Develop programs that incorporate abstractions.*
- **Practice P4: Code Analysis**
    - *Evaluate and test algorithms and programs.*
- **Practice P5: Computing Innovations**
    - *Investigate computing innovations.*
- **Practice P6: Responsible Computing**
    - *Contribute to an inclusive, safe, collaborative, and ethical computing culture.*

Big Ideas:
The five big ideas of the course encompass foundational ideas in the field of computer science, and are denoted here by B1 through B5:

- **Big Idea 1: 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.

- **Big Idea 2: Data (DAT)**
  Data is central to computing innovations because it communicates initial conditions to programs and represents 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 this 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.

- **Big Idea 3: 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.

- **Big Idea 4: 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.

- **Big Idea 5: 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 how our programs will be used and be responsible

for the consequences. As computer users, we need to understand how to protect ourselves and our privacy when using a computer.

The AP Create Performance Task:
The through-course assessment is a performance task designed to gather evidence of student proficiency in the learning objectives. The AP Create Performance Tasks (PT) is an in-class assessment, administered by the teacher, that allows students to exemplify their learning through an authentic, “real-world” creation. In the Create Performance Task, students will design and implement a program to solve a problem, enable innovation, explore personal interest, or express creativity. Their development process should include exploration, investigation, reflection, design, implementation, and testing your program.

Students will gain the experience necessary to complete the Create Performance Task in class. Each unit comes with practice PTs in which students will research topics in computing and create their own digital artifacts. Sufficient time is set aside in the course for students to prepare for and complete the Create Performance Task.

The AP Exam:
The AP Computer Science Principles end-of-course exam has consistent question types and weighting every year, so you and your students know what to expect on exam day.

Section I: End-of-Course Multiple-Choice Exam
70 multiple-choice questions | 120 minutes | 70% of score | 4 answer options

- 57 single-select multiple-choice
- 5 single-select with reading passage about a computing innovation
- 8 multiple-select multiple-choice: select 2 answers

Section II: Create Performance Task: Written Responses
30% of score

- Create Performance Task program code, video, and student-authored Personalized Project Reference | 9 hours in-class
- 4 written response prompts | 60 minutes end-of-course exam

The second section of the AP Computer Science Principles Exam consists of a through-course Create Performance Task where students will develop a computer program of their choice and an end-of-course written response section where students demonstrate their understanding of their personal Create Performance Task by answering four prompts. Students will be provided 9 hours of in-class time to complete their program, video, and develop a Personalized Project Reference.

Course Breakdown

Unit 1: Introduction to Programming with Karel the Dog (3 weeks, 15 hours)

This course begins with a strong focus on programming in order to allow students to create computational artifacts early on in the course. Students will be able to use their knowledge of programming to explore future topics in the course.

We use Karel, a dog that only knows how to move, turn left, and place tennis balls in his world, to show students what it means to program, and allow students to focus on computational problem-solving. Students will learn about the need for programming languages, the uses of programs, how to write programs to solve computational problems, how to design algorithms, how to analyze and compare potential solutions to programming problems, and learn the value and challenges involved in collaborating with others to solve programming problems. Students will use the grid coloring functionality of Karel to create a digital painting and embed this program in their portfolio website.

Subsection	EKs	Lessons / Topics
Abstraction Lessons: *Abstraction*	AAP-3.B.1 AAP-3.B.2 AAP-3.B.3 AAP-3.B.4 AAP-3.B.6 AAP-3.B.7 CRD-2.G.1 DAT-1.A.2 DAT-1.A.5	Procedural Abstraction Modularity Program Reuse Digital Data (Bits) Reducing Complexity
Programming Style Lessons: *Intro to Programming* *Super Karel* *Ultra Karel* *Top-Down Design* *Commenting Your Code*	CRD-2.G.1 CRD-2.G.2 AAP-2.M.1 AAP-2.M.3 CRD-2.B.1 CRD-2.B.2 CRD-2.B.5 AAP-3.D.1 AAP-3.D.2 AAP-3.D.3 AAP-3.D.4 AAP-3.D.5	Program Documentation Using Existing Code and Libraries APIs Commenting Code
Control Structures Lessons: *If/Else Statements* *For Loops* *While Loops in Karel*	AAP-2.G.1 AAP-2.J.1 AAP-2.K.1	If/Else Statements (Selection) For Loops and While Loops (Iteration)
Debugging Strategies Lessons: *Functions in Karel* *Debugging Strategies*	CRD-2.I.1 CRD-2.I.2 CRD-2.I.3 CRD-2.I.5	Logic Errors Syntax Errors Run-Time Error Testing
Designing Algorithms Lessons: *Karel Algorithms*	AAP-2.A.4 AAP-2.B.1 AAP-2.B.2 AAP-2.B.6 AAP-2.M.2 AAP-4.A.2 AAP-4.A.4 AAP-4.A.5	Sequencing, Selection, Iteration Clarity and Readability Using Existing Algorithms Optimization and Efficiency

	AAP-2.B.7 AAP-4.A.6	

Example Activities and Big Idea/Computational Thinking Practice
The Two Towers: In this program, students have Karel build two towers of tennis balls. Each tower should be 3 tennis balls high. In the end, Karel should end up on top of the second tower, facing East. Students need to write at least 3 functions in order to solve this problem. This activity requires students to design and create functions for repeated processes within their program. Students need to consider top-down design and decomposition through the following questions:

- How can you break this problem down into smaller problems?
- What is a subtask that Karel needs to do more than once in this problem?

[Big Idea AAP][Computational Thinking Practice 1]

Unit 2: Practice PT: Pair-Programming Paint (3 days, 3 hours)
Students will use the grid coloring functionality of Karel to create a digital image. They will then embed this Karel program into their personal website portfolio.

Subsection	EKs	Lessons / Topics
Collaboration and Communication	CRD-1.A.3 CRD-2.F.7 CRD-1.A.4 CRD-2.G.1 CRD-1.B.2 CRD-2.G.3 CRD-1.C.1 CRD-2.G.4 CRD-2.F.5 CRD-2.G.5 CRD-2.F.6 CRD-2.H.1 CRD-2.H.2	Collaboration Diverse Perspectives Bias Avoidance Pair-Programming Design and Planning Program Documentation Acknowledgement of Reused Code

Example Activity and Big Idea/Computational Thinking Practice
Create Your Own UltraKarel Image: Following the milestones and the pseudocode plan that students have laid out, students use pair-programming to write the code for their final project. They then test their code along the way to make sure they have solved each milestone. This activity allows students to develop something completely unique with their programming skills and implement a successful algorithm of their own design.

Students then reflect upon and answer the following questions:

1. Identify the programming language and purpose of your program.

2. Describe the incremental and iterative development process of your program. How did you divide the program into smaller tasks and make a plan to complete them all?

3. Describe the difficulties and/or opportunities you encountered and how they were resolved or incorporated.

4. Identify an algorithm that is fundamental for your program to achieve its intended purpose and includes two or more additional algorithms.

5. Describe how each algorithm within your selected algorithm functions independently, as well as in combination with others, to form a new algorithm that helps to achieve the intended purpose of the program.

6. Identify an abstraction you developed, and explain how your abstraction helped manage the complexity of your program.
[Big Idea CRD][Computational Thinking Practice 2]

Unit 3: Programming with Python (2 weeks, 10 hours)
This unit introduces students to the basics of Python, including variables, user input, control structures, functions with parameters and return values, and basic graphics, how to send messages to objects.

Subsection	EKs	Lessons / Topics
Programming Languages Lessons: *What is Code?* *Uses of Programs*	AAP-2.A.2 AAP-2.A.3 CRD-1.A.1 CRD-1.A.2 CRD-2.B.1	What is Programming? Pseudocode Programming Languages Computing Innovations
Variables Lessons: *Variables*	AAP-1.A.1 AAP-1.A.2 AAP-1.A.3 AAP-1.A.4 AAP-1.B.1 AAP-1.B.2 AAP-1.B.3 DAT-1.A.1	Variable Names Assignment Operators Data Types Variables as Abstractions
Arithmetic Expressions Lessons: *Basic Math in Python*	CRD-2.B.4 CRD-2.I.5 CRD-2.J.1 CRD-2.J.2 CRD-2.J.3 AAP-2.A.1 AAP-2.A.2 AAP-2.A.3 AAP-2.A.4 AAP-2.B.3 AAP-2.B.4 AAP-2.B.5 AAP-2.C.1 AAP-2.C.2 AAP-2.C.3 AAP-2.C.4 AAP-2.D.1 AAP-2.D.2	Program Behavior Testing using Inputs Arithmetic Expressions Order of Operations Modulus String Concatenation
User Input Lessons: *User Input* *Mouse Events: Mouse Clicked* *Key Events*	AAP-1.C.4 AAP-3.A.6 AAP-3.A.9 CRD-2.C.2 CRD-2.C.3 CRD-2.C.5 CRD-2.C.6 CRD-2.D.2	Strings User Input Program Output Events Mouse and Key Events
Example Activity and Big Idea/Computational Thinking Practice		

Computing Innovations (as part of *Uses of Programs* lesson)*:* In this activity, students perform an online search for examples of computing innovations that have had an impact on society, economy, or culture. The computing innovations must consume, produce, and/or transform data. A computing innovation can be a physical object like a self-driving car, non-physical software like a picture editing software, or a non-physical concept like e-commerce.

Students

- practice searching and evaluating sources relevant to computing innovations
- write the definition of *computing innovation* in their own words
- list 5 items that ARE computing innovations and 5 items that are NOT computing innovations. For each one, explain the reason why it is or is not a computing innovation
- 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. **[Computing Innovation 1, Prompt B][Big Idea IOC][Computational Thinking Practice 5]**

Unit 4: Python Control Structures (2 weeks, 10 hours)
In this unit, students learn how to use booleans and logical operators with control structures to make more advanced programs in Python.

Subsection	EKs	Lessons / Topics
Comparison Operators Lessons: *Booleans* *Comparison Operators*	AAP-2.E.1 AAP-2.E.2 AAP-2.F.1 AAP-2.F.2 AAP-2.F.3 AAP-2.F.4 AAP-2.F.5	Booleans Relational Operators Operands
Selection Lessons: *If Statements* *Random Numbers*	AAP-2.G.1 AAP-2.H.1 AAP-2.H.2 AAP-2.H.3 AAP-2.I.1 AAP-2.I.2 AAP-2.L.3 AAP-2.L.4 AAP-3.E.2	Selection Conditional Statements Nested Conditionals Equivalent Boolean Statements Random Numbers
Iteration Lessons: *While Loops*	AAP-2.K.2 AAP-2.K.3 AAP-2.K.4 AAP-2.K.5 AAP-2.L.1 AAP-2.L.2 AAP-2.L.5	Iteration Loops Different but Equivalent Algorithms

Example Activity and Big Idea/Computational Thinking Practice
Better Password Prompt: Students write a program that uses a while loop to prompt a user for a password. They keep prompting the user for the password, and if they get it correct, they then break out of the loop. If they don't get it correct, they should give the user an error message. This activity requires that students use multiple program statements in a specific order to solve a problem.
[Big Idea AAP][Computational Thinking Practice 2]

Unit 5: Functions and Parameters (2 weeks, 10 hours)
In this unit, students learn to write reusable code with functions and parameters.

Subsection	EKs	Lessons / Topics
Functions and Parameters Lessons: *Functions and Parameters 1* *Functions and Parameters 2* *Functions and Return Values 1* *Functions and Return Values 2*	CRD-2.C.6 AAP-3.A.3 CRD-2.D.2 AAP-3.A.4 CRD-2.B.3 AAP-3.B.5 CRD-2.C.4 AAP-3.C.1 AAP-3.A.1 AAP-3.C.2 AAP-3.A.2 AAP-2.M.2	User and Application Input Program Output Procedures Parameters Return Values Using Existing Algorithms
Example Activity and Big Idea/Computational Thinking Practice *Pool Table:* Students write a program with a function that draws a pool ball. This function should take as parameters, the color, the number that should go on the pool ball, and the location of the center of the pool ball. Students need to consider the function abstractly as a means for taking specific data via the parameters and creating a unique graphical output based on those inputs. **[Big Idea DAT][Computational Thinking Practice 3]**		

Unit 6: Practice PT: Tell a Story (3 days, 3 hours)
In this project, students will write a Python program that tells a graphical story

Example Activity and Big Idea/Computational Thinking Practice *Tell a Story!* In this activity, students write a Python program that tells a graphical story in at least 4 scenes. Following the milestones and the pseudocode plan that students have laid out prior to this exercise, students write the code for their final project. They iterate and test their code along the way to make sure they have solved each milestone. **[Big Idea CRD][Computational Thinking Practice 4]**

Unit 7: Basic Data Structures (2 weeks, 10 hours)
In this unit, students learn to write reusable code with functions and parameters.

Subsection	EKs	Lessons / Topics
Basic Data Structures Lessons: *Lists*	DAT-1.A.1 AAP-1.A.1 AAP-1.C.1 AAP-1.C.2 AAP-1.C.3 AAP-1.D.6 AAP-1.D.7	Data Values Lists and Elements Indices List Procedures

	AAP-1.D.8 AAP-2.N.2 AAP-2.N.1	
Data Abstractions Lessons: *Lists* *For Loops and Lists*	AAP-1.D.1 AAP-1.D.5 DAT-2.E.4 AAP-1.D.2 AAP-1.D.3 AAP-1.D.4 DAT-2.E.2 DAT-2.D.4 DAT-2.E.5	Data Abstraction Translating and Transforming Data Filtering and Cleaning Patterns
Traversing a List Lessons: *Lists* *For Loops and Lists*	DAT-2.D.6 AAP-2.O.1 AAP-2.O.2 AAP-3.C.1 AAP-3.C.2 AAP-3.A.6 AAP-2.O.3 AAP-3.A.5 AAP-3.A.7 AAP-3.A.8 AAP-3.E.1	Extract and Modify Information Traversing a List Iteration Statements
Algorithm Efficiency Lessons: *For Loops and Lists* *List Methods*	AAP-2.O.4 DAT-2.D.3 AAP-2.O.5 AAP-2.P.1 AAP-2.P.2 AAP-2.P.3 AAP-4.A.1 AAP-4.A.3 AAP-4.A.7 AAP-4.A.8 AAP-4.A.9	Using Existing Algorithms Search Tools Linear Search Binary Search Algorithm Efficiency Heuristics
Simulation Lessons: *Simulation*	AAP-3.F.1 AAP-3.F.2 AAP-3.F.3 AAP-3.F.4 AAP-3.F.5 AAP-3.F.6 AAP-3.F.7 AAP-3.F.8	Simulations as Abstractions Bias in Simulations Random Number Generators
Example Activity and Big Idea/Computational Thinking Practice *Librarian, Part 2:* Students write a program to ask the user for an author's full name. Students will then use list procedures to split the full name into individual names and then slice the list to add the last name to a new list. Once the student has collected all of the last names, they will sort them and then print the results. This program development requires students to use user		

input data that can contain a variable number of names. The students must then use various list techniques to manipulate the data.
[Big Idea DAT][Computational Thinking Practice 2]

Unit 8: Digital Information (3 weeks, 15 hours)
In this unit, students will learn about the various ways we represent information digitally. Topics covered include number systems, encoding data, programmatically creating pixel images, comparing data encodings, compressing and encrypting data. Students will work in pairs to develop their own data encryption algorithms and attempt to crack the encryptions of their peers. Their text encryption tool will be embedded in their portfolio websites.

Subsection	EKs	Lessons / Topics
Number Systems Lessons: *Intro to Digital Information* *Number Systems*	CRD-2.C.1 CRD-2.D.1 CRD-2.J.2 CRD-2.J.3 CRD-2.I.4 DAT-1.A.2 DAT-1.A.3 DAT-1.A.4 DAT-1.A.5 DAT-1.A.6 DAT-1.A.7 DAT-1.B.1 DAT-1.B.2 DAT-1.B.3 DAT-1.C.1 DAT-1.C.2 DAT-1.C.3 DAT-1.C.4 DAT-1.C.5	Computing Devices Abstraction Program Input and Output Bits and Bytes Overflow Errors Range of Value Limits Binary and Decimal Systems
Data Compression Lessons: *Data Compression* *Lossy Compression*	DAT-1.A.8 DAT-1.A.9 DAT-1.A.10 DAT-1.D.1 DAT-1.D.2 DAT-1.D.3 DAT-1.D.4 DAT-1.D.5 DAT-1.D.6 DAT-1.D.7 DAT-1.D.8	Lossless Data Lossy Data Digital and Analog Data
Cryptography Lessons: *Cryptography*	AAP-4.B.1 AAP-4.B.2 AAP-4.B.3 IOC-2.B.8 IOC-2.B.5	Decidable Problems Computer Viruses Encryption

Example Activity and Big Idea/Computational Thinking Practice
Guess the Passcode: Students first imagine they forgot their 4-digit passcode for their phone, and need to guess the correct passcode. They develop a program to guess passcodes for them to speed up the process. Once the correct passcode has been guessed, the program should print out how many guesses it took to reach the correct one. This activity encourages students to consider security issues which can be expanded to how we create a safer computing culture.

Students discuss the following questions with a partner:

1. How many possible passcodes will you need to guess before you've guessed every possible passcode?
2. Why is this dangerous for the security of your phone?
3. Imagine a hacker had access to your phone and had written a program to guess every possible passcode until they had broken in. What defenses could we build into the phone to keep this guess and check strategy from working? (What happens when you guess incorrectly over and over again?)
4. Can you think of any guessing strategies that might be faster than starting at 0000 and iterating all the way up to 9999

[Big Idea IOC][Computational Thinking Practice 6]

Unit 9: Practice PT: The Shopping List (3 days, 3 hours)

For this creative endeavor, students will develop a Python-based application designed to maintain a digital shopping list. This task requires students to integrate their understanding of complex data structures, algorithmic iteration, and modular procedural abstraction.

Example Activity and Big Idea/Computational Thinking Practice

The Shopping List- Students develop a program that manages a shopping list using lists and other data structures. They apply functions and iteration to add, remove, and display items, demonstrating their understanding of data abstraction and algorithm design.

[Big Idea AAP][Computational Thinking Practice 2]

Unit 10: The Internet (2 weeks, 10 hours)

This unit explores the structure and design of the internet, and how this design affects the reliability of network communication, the security of data, and personal privacy. Students will learn about the protocols and algorithms used on the internet and the importance of cybersecurity. Students will choose an innovation that was enabled by the Internet and explore the positive and negative impacts of their innovation on society, economy, and culture. Students will develop a computational artifact that illustrates, represents, or explains the innovation's purpose, its function, or its effect, and embed this artifact in their personal portfolio website.

Subsection	EKs	Lessons / Topics
Internet Hardware and Addresses Lessons: *Welcome to the Internet* *Internet Hardware* *Internet Addresses*	CSN-1.A.1 CSN-1.A.2 CSN-1.A.3 CSN-1.A.4 CSN-1.A.7 CSN-1.A.8 CSN-1.B.3 CSN-1.B.4	Protocols Computing Devices Computer Networks Bandwidth
Routing	CSN-1.A.5 CSN-1.A.6 CSN-1.B.5 CSN-1.E.2 CSN-1.E.3 CSN-1.E.4	Routing Scalability Fault-Tolerance

Lessons: *Routing*	CSN-1.B.6 CSN-1.E.5 CSN-1.B.7 CSN-1.E.6 CSN-1.E.1 CSN-1.E.7	Redundancy
Packets and Protocols Lessons: *Packets and Protocols*	CSN-1.B.1 CSN-1.D.1 CSN-1.B.2 CSN-1.D.2 CSN-1.C.1 CSN-1.D.3 CSN-1.C.2 DAT-2.B.1 CSN-1.C.3 DAT-2.B.3 CSN-1.C.4 DAT-2.B.5	Datastreams Packets IP, TCP, UDP HTTP Metadata
Computing Systems Lessons: *Sequential, Parallel & Distributed*	DAT-2.C.7 CSN-2.A.6 DAT-2.C.8 CSN-2.A.7 CSN-2.A.1 CSN-2.B.1 CSN-2.A.2 CSN-2.B.2 CSN-2.A.3 CSN-2.B.3 CSN-2.A.4 CSN-2.B.4 CSN-2.A.5 CSN-2.B.5	Parallel Systems Scalability of Systems Sequential Computing Parallel Computing Distributed Computing Efficiency of Solutions Speedup
Impact of the Internet Lessons: *The Impact of the Internet* *Creative Credit and Copyright*	IOC-1.A.1 IOC-1.E.2 IOC-1.A.3 IOC-1.E.3 IOC-1.A.4 IOC-1.E.4 IOC-1.A.5 IOC-1.E.5 IOC-1.B.1 IOC-1.E.6 IOC-1.B.2 IOC-1.F.1 IOC-1.B.3 IOC-1.F.2 IOC-1.B.4 IOC-1.F.3 IOC-1.B.5 IOC-1.F.4 IOC-1.B.6 IOC-1.F.5 IOC-1.C.1 IOC-1.F.6 IOC-1.C.2 IOC-1.F.7 IOC-1.C.3 IOC-1.F.9 IOC-1.C.4 IOC-1.F.10 IOC-1.C.5 IOC-1.F.11 IOC-1.E.1	Computing Innovations Unintended Effects Impact on Society Rapid Sharing Digital Divide Citizen Science Crowdsourcing Creative Credit and Copyright
Cybersecurity Lessons: *Cybersecurity*	IOC-1.F.8 IOC-2.B.5 IOC-2.A.1 IOC-2.B.6 IOC-2.A.7 IOC-2.B.7 IOC-2.A.8 IOC-2.B.9 IOC-2.A.9 IOC-2.B.10 IOC-2.A.11 IOC-2.B.11 IOC-2.A.12 IOC-2.C.1 IOC-2.A.13 IOC-2.C.2 IOC-2.A.15 IOC-2.C.3 IOC-2.B.1 IOC-2.C.4 IOC-2.B.2 IOC-2.C.5 IOC-2.B.3 IOC-2.C.6 IOC-2.B.4 IOC-2.C.7	Legal and Ethical Concerns Personally Identifiable Info (PII) Digital Footprint Authentication Certificate Authorities (CAs) Computer Viruses Malware Phishing Keylogging Rogue Access Points Encryption
Example Activity and Big Idea/Computational Thinking Practice		

Reflection: Unintended Effects - Students consider the WWW, targeted advertising and machine learning and data mining as examples of computing innovations. They also learn that responsible programmers try to consider the unintended ways their computing innovations can be used and the potential beneficial and harmful effects of these new uses although it may not be possible for a programmer to consider all the ways a computing innovation can be used.

They then consider *Pokemon Go* (from the previous video) or research another innovation that had unintended effects. Students answer in their reflections:

1. What were the intended effects and what were the unintended effects?
2. Explain beneficial and harmful effects of at least one other computing innovation on society, economy, or culture.

[Computing Innovation 2, Prompt A][Big Idea IOC][Computational Thinking Practice 5]

Packets and Protocols: The Story of the Internet - In their own words, students tell the story of downloading an image from a website on the internet. They tell the story step by step of how their computer finds the relevant server, requests information from the server, and receives it. Students are required to include distinctions between the internet and the World Wide Web, such as:

- The World Wide Web is a system of linked pages, programs, and files.
- HTTP is a protocol used by the World Wide Web.
- The World Wide Web uses the Internet.

[Big Idea CSN][Computational Thinking Practice 5]

Unit 11: Project: The Effects of the Internet (3 days, 3 hours)

In this project, students will choose an innovation that was enabled by the Internet and explore the positive and negative impacts of their innovation on society, economy, and culture. Students will develop a computational artifact that illustrates, represents, or explains the innovation's purpose, its function, or its effect, and embed this artifact in their personal portfolio website.

Example Activity and Big Idea/Computational Thinking Practice

The Effects of the Internet: Students provide evidence of the extensive knowledge they have developed about a chosen Internet-based innovation and its impact(s). Students include citations, as applicable, within their written responses.

Within their computational artifact, students explain at least one beneficial effect and at least one harmful effect the Internet-based innovation has had, or has the potential to have, on society, economy, or culture. They also identify data privacy, security, or storage concerns for the computing innovation.

[Computing Innovation 3, Prompt C][Big Idea IOC][Computational Thinking Practice 5]

Unit 12: Data (1 week, 5 hours)

In this unit, students will explore using computational tools to store massive amounts of data, manipulate and visualize data, find patterns in data, and draw conclusions from data. Students will

consider how the modern wealth of data collection has impacted society in positive and negative ways. Students will work in teams to investigate a question of personal interest and use public data to present a data-driven insight to their peers. They will develop visualizations to communicate their findings, and embed their visualizations in their portfolio websites.

Subsection	EKs	Lessons / Topics
Visualizing and Interpreting Data Lessons: *Getting Started with Data* *Visualizing and Interpreting Data*	DAT-2.A.1 DAT-2.D.5 DAT-2.A.2 DAT-2.D.6 DAT-2.C.1 DAT-2.E.1 DAT-2.D.1 DAT-2.E.2 DAT-2.D.2 DAT-2.E.3 DAT-2.D.3 DAT-2.E.5 DAT-2.D.4	Filtering and Cleaning Data Patterns and Trends Search Tools Tables, Diagrams and Displays Interactive Visualizations Combining Data Sources
Collecting Data and Data Limitations Lessons: *Data Collection and Limitations*	DAT-2.A.3 DAT-2.C.2 DAT-2.A.4 DAT-2.C.3 DAT-2.B.1 DAT-2.C.4 DAT-2.B.2 DAT-2.C.5 DAT-2.B.3 DAT-2.C.6 DAT-2.B.4 DAT-2.D.6 DAT-2.B.5 CRD-2.F.3	Metadata Correlation Using a Variety of Sources Incomplete or Invalid Data Bias Surveys, Testing, Interviews
Example Activity and Big Idea/Computational Thinking Practice *Importance of Metadata:* Students consider how metadata can increase the effective use of data or data sets by providing additional information. They consider the importance of metadata and reflect on why metadata is important for a data set, how metadata help in finding specific data, and what metadata should reveal about the data. **[Big Idea DAT][Computational Thinking Practice 5]**		

Unit 13: Project: Present a Data-Driven Insight (3 days, 3 hours)
In this project, students will work with a partner to answer a question of personal interest using a publicly available data set. Students will need to produce data visualizations and explain how these visualizations led to their conclusions. They will develop a computational artifact that illustrates, represents, or explains their findings, communicate their findings to their classmates, and embed their artifact in their personal portfolio website.

Example Activity and Big Idea/Computational Thinking Practice
Present a Data-driven Insight: Students consider how the amount of collected data impacts our lives in ways that require considerable study and reflection for us to fully understand them. Students explore a question that can be answered by analyzing a dataset. They form a question and use visualization techniques to analyze the data to answer the question.
[Big Idea DAT][Computational Thinking Practice 6]

Unit 14: Strings (2 weeks, 10 hours)
Students use more sophisticated strategies for manipulating text in their programs, including indexing, slicing, concatenating, and formatting strings.

Example Activity and Big Idea/Computational Thinking Practice
Strings Quiz: Students demonstrate their understanding of string manipulation techniques including indexing, slicing, and built-in string methods through a summative assessment.
[Big Idea AAP][Computational Thinking Practice 4]

Unit 15: Practice PT: Personal Data Tracker (3 days, 3 hours)
In this project, students plan and build a personal data tracker program that stores and analyzes data of personal interest, practicing the skills needed for the Create Performance Task.

Example Activity and Big Idea/Computational Thinking Practice
Practice Personal Data Tracker: Students plan their data tracker, build its core features, and practice Create PT deliverables including designing a program that uses a list, writing code that processes data, and producing a video walkthrough.
[Big Idea AAP][Computational Thinking Practices 1-4]

Unit 16 & 17: Project: The Impacts of Computing and Create Performance Task (3 weeks, 15 hours)
This time is set aside for students to prepare for the MCQs that were part of the Explore Task and create their AP Create Performance Task. Students will be given the chance to review course content and practice the skills necessary to complete the Create Performance Task. The Create PT will be administered over 9 hours of class time.

Subsection	EKs	Lessons / Topics
AP CSP Explore MCQ Practice	IOC-2.A.2 IOC-2.A.10 IOC-2.A.3 IOC-2.A.14 IOC-2.A.4 IOC-1.F.11 IOC-2.A.5 CRD-1.A.1 IOC-2.A.6 CRD-1.A.2	Artifact Creation Computing Innovations Data Input and Output Data Privacy and Security
Prepare for Create PT	ALL	Review Course Content Incremental Development Documentation Debugging Collaborative Development
Create PT		9 hours of class time to conduct Create PT
Example Activity and Big Idea/Computational Thinking Practice		

Create Performance Task: Students develop a program of their choice. Their development process includes iteratively designing, implementing, and testing their program. Students are strongly encouraged to work with another student in their class.
[Big Idea AAP][Computational Thinking Practices 1-4]

Unit 18: Review for the AP Exam (1 week, 5 hours)
This unit gives students a review of the topics covered in the course and provides practice solving AP Exam-style multiple-choice questions.

Subsection	Lessons / Topics
Prepare for Practice Exam	Review course content What to expect on the exam
Practice AP Exam	Cumulative Final AP Review Multiple Choice Test

Unit 19: Creative Development (Remainder of the school year, 2-4 weeks, 10-20 hours)
In this unit, students will brainstorm their own final project, discuss their ideas with their peers, scope their project to fit within the time constraints of the class, plan out milestones for incremental development, and create their own final product from scratch. This project allows students to think creatively about the applications of the concepts covered in the course, and create something of personal value.

Subsection	EKs	Lessons / Topics
Design Thinking Lessons: *Intro to Design Thinking*	CRD-1.A.4 CRD-1.A.5 CRD-1.A.6 CRD-2.A.1 CRD-2.A.2 CRD-2.E.1 CRD-2.E.2 CRD-2.E.4 CRD-2.F.1 CRD-2.F.2 CRD-2.F.5 CRD-2.F.6 CRD-2.F.7 IOC-1.A.2	Computing Innovations Development Process Program Specifications Design Phase Communication Collaboration
Brainstorm, Prototype & Test Lessons: *Prototype* *Test*	CRD-2.E.2 CRD-2.F.7 CRD-1.A.5 CRD-1.A.6 CRD-1.A.4 CRD-2.E.3 CRD-2.F.4 CRD-2.F.3 IOC-1.D.1 IOC-1.D.2 IOC-1.D.3 IOC-1.F.11	Development Process User Testing User Research Diverse Perspectives Iterative Development Human Biases Legal and Ethical Concerns
Project Prep and	CRD-1.B.1	Online Collaboration Tools

Development Lessons: *Project Prep and Development*		
Example Activity and Big Idea/Computational Thinking Practice *User Interface Scavenger Hunt:* Students search for 2 websites or apps, one with a good UI and one with a not-so-good UI. They learn to discriminate features of solid UI design in terms of accessibility and more before moving onto prototyping their creative project for the unit. **[Big Idea CRD][Computational Thinking Practices 6]**		

AP Computer Science Principles Supplemental Materials

Supplementary Units	**Prerequisite/Recommended Unit(s)**	**# of activities**
Extra Karel Practice	Intro to Programming	12
Extra Karel Puzzles	Intro to Programming	11
Karel Challenges	Intro to Programming	7
Web Development	After Pretest	79
Classes and Objects	After Basic Data Structures	12
Additional Topics	After Basic Data Structures	10
Project: Who Said It	After Basic Data Structures	8
Project: Mastermind	After Basic Data Structures	7
Project: Steganography	After Digital Information	
Project: Create an Image Filter!	After Digital Information	
Practice PT: Testing 1, 2, 3 ...	After Functions and Parameters	
Section I: End-of-Course Multiple-Choice Exam Review	After AP Exam Review	
Section II: Create Performance Task: Written Responses Review	After Create Performance Task	
Reader Question Bank Quizzes	After AP Exam Review	