AP Computer Science Principles
Indian Mountain School 2026-2027

Prerequisites:

- Rising ninth-grade students
- Completion of Algebra I with a minimum final grade of 90 or completion of a more advanced math course

No prior programming or computing knowledge is required!

Course Overview:

AP Computer Science Principles is designed to encourage a diverse group of students to explore computer science. This course introduces students to a broad set of ideas. These "big ideas" are creative development (CRD), data (DAT), algorithms and programming (AAP), computing systems and networks (CSN), and the impact of computing (IOC). In addition, this course emphasizes the use of computational thinking practices for learning experiences and problem solving. These practices include computational solution design, algorithms and program development, abstraction in program development, code analysis, computing innovations, and responsible computing.

Students will build their knowledge and understanding through participation in a wide variety of activities.. Before, during, and after activities, connections are made to the five big ideas at the core of the course. Activities also encourage students to apply the six computational thinking practices to their work.

Within the cumulative nature of learning a programming language, assessments will build on prior knowledge. Student performance expectations will increase with each assessment. Students will also work collaboratively as they investigate and learn. Their collaborative work and efforts will also be assessed within the framework of the course.

The primary programming language used in this course is Python. Python's text-based syntax and logical framework makes it one of the more popular languages for beginning and advanced coders.

Throughout this course, students are guided toward personal discoveries. They will be introduced to computer science topics that are related to current events and their own experiences. Students will focus on topics related to popular culture, historical events, and other areas. Students are encouraged to pursue personal interests related to these topics. Student-initiated explorations are among the most valuable for the student experience in this course. Explorations for AP CSP are designed to spark curiosity, enthusiasm, and enjoyment.

Both students and the instructor provide input in choosing the activities and requirements associated with the course's final assessment and production of an application in Python.

The AP Exam:

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

Multiple Choice Exam
The multiple choice exam will test students' understanding of computational logic. Students will not need to understand any formal coding language to complete this part of the exam. The multiple-choice exam accounts for 70% of a student's total AP score. Students will have 120 minutes to complete 70 multiple choice questions.

Written Response
Students will prepare a written response based on their performance task. They will answer similar questions about concepts within their code as part of the AP Exam. Students will have 60 minutes to complete 4 written response prompts.

Performance Task
The Performance Task is submitted as a project that students must complete and submit online. This is submitted prior to taking the multiple-choice and written response portions of the exam. This task, when combined with the written response, is worth 30% of a student's overall AP score.

> **Create:** Students create and submit an original application. Students will also submit a video of their application, demonstrating its functionality. In addition to this, they will prepare, but not submit, a Personalized Project Reference Sheet describing how their application functions, including required components and theory.
>
> Students will be given a minimum of 12 hours of class time to complete the Create Task, some of which can be done collaboratively with a partner. The reference sheet that students create can be used during the written response portion of the exam.

Text and Materials: CR-1

CS Principles: Big Ideas in Programming. Mark Guzdial, Barbara Ericson, and Jeffrey Elkner, 2022. *Hands-on introductory computer programming using Python 3.*

Blown to Bits. Abelson, H., Ledeen, K., & Lewis, H. 2008, Addison-Wesley.

Coding in Python will be completed and submitted via the PyCharm Education module.

Articles and videos will also be used as they pertain to each topic or discussion.

Units and Scoring:

Each unit follows the structure of the text, CS Principles: Big Ideas in Programming.

The text directly addresses the big ideas and computational thinking practices and skills, as stated in its "Standards" page.

Students will complete each chapter of the text and subsequent review activities. These chapters explain and demonstrate new concepts and ideas related to computing and coding. They also provide self-scoring activities and problem sets which help to demonstrate use cases for these concepts and ideas.

When students have completed the textbook activities, where applicable, students will then engage in a larger, more comprehensive activity that will aid in developing and demonstrating mastery of the topics introduced in that chapter. These activities often include coding, collaborative work, research, creative design, and discussions or presentations. Students will be assessed on their work and production for these activities. These are listed in the next section of the document, titled Course Outline, below each unit. They more directly address the AP curricular requirements that are touched upon in the text.

The design of the text and the supplemental activities present a scaffolded approach to understanding the big ideas, which are the focus of this course, and to preparing students for the tasks and exam that will be ultimately assessed by the College Board.

Students will earn a grade in this course at the end of each term as determined by the instructor. These grades are not related to the AP score, but are based on in-class activities, discussions, homework, and projects. The AP score is determined by the student's Performance Tasks and Multiple Choice Exam as judged by the College Board.

Curricular Requirements:

The following requirements are presented in activities that supplement the text. The text itself addresses computational thinking practices as demonstrated in its "Standards" page.

	Requirements and Computational Thinking Practices	Pages
CR-1	The teacher and students have access to college-level computer science resources, in print or electronic format.	3
CR-2	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.	5-14
CR-3	The course provides opportunities to develop student understanding of the Big Ideas.	5-14
CR-4	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 1: Computational Solution Design.	7,11-12
CR-5	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 2: Algorithms and Program Development.	6-13
CR-6	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 3: Abstraction in Program Development.	9,11,12
CR-7	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 4: Code Analysis.	6,9.11
CR-8	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 5: Computing Innovations.	5,8,13
CR-9	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 6: Responsible Computing.	6,8

Requirements and Computational Thinking Practices		Pages
CR-10	The course provides a minimum of three opportunities for students to investigate different computing innovations.	5,8,13
CR-11	Students are provided at least twelve (12) hours of dedicated class time to complete the AP Create Performance Task.	13

Course Outline: CR-2

Unit 1: *"COMPUTER ABILITIES," CS Principles, Section 1*

Text Guided Topics/Activities:

- What can computers do?
  - Different ways we compute
    - Readings, Discussion
  - Origins of computers
    - Readings, Videos, Activities
  - Limitations of computers
    - Readings, Discussion

Big Ideas:

AAP, IOC, CSN

Unit Activities:

- **2, Program your Friend - AAP**CR-3
  - Students create a list of instructions for their partner to accomplish a specific task. The tasks will be distributed via index cards to each group. It is important that the instructions given to the partner are explicitly followed. Groups will note any challenges they face and a class discussion will follow within the scope of the unit, bringing in ideas about ways we compute, the limitations of computers, and an understanding of what an algorithm is.

- **2, Evolution of the Computer - IOC, CSN** CR-3, CR-8, CR-!0
  - In groups of 2 or 3, students will research the history of the computer using images affixed to a timeline to illustrate changes in computing over time. This research should include a modern computer innovation. Groups will ask to predict a future innovation by adding a "future" element to their timeline.

Groups will briefly present their timelines, explaining what each image represents and why it was or will be important.

Unit 2: *"NAMING," CS Principles, Section 2*

Text Guided Topics/Activities:

- 3 Names for Numbers
  - Readings, Videos, Discussion, Activity
- 4 Names for Strings
  - Readings, Discussion, Activity
- 5 Names for Turtles
  - Readings, Videos, Activity
- 6 Names for Everything
  - Readings, Videos, Discussion

Big Ideas:

CRD, AAP

Unit Activities:

- **3, Splitting the Check - AAP** CR-5
  - Students code in Python a calculator for splitting a check.
  - Before coding, students will watch this short video and attempt to use pseudocode to design their program.
    5 Minutes to Code: Programming Basics "Pseudocode"
  - The app will return the amount each person should spend including the tip. It will take inputs for the number of people and the price of the bill. When finished, students will pair to test each other's code. The class will discuss strategies used for completing the activity and discuss the efficacy of using pseudocode.
- **4, Mad Libs Extension - AAP** CR-5
  - Students extend the *Mad Libs* examples from Chapter 4 their texts. Use the "input" command to create a personal version of a Madlib. The code should output a story that takes user input to fill-in blanks within the story. At least one of the inputs should be a number to help demonstrate that all inputs are strings unless directly assigned in the code. Post-activity students will pair up to test each other's code and produce short fun and funny stories.
- **6, Coding a Work of Art - AAP, CRD** CR-9
  - Students will be writing the code using blocks or Python to create a picture of their choosing. The code must include a function for a regular polygon and at least two functions that create something other than a basic shape.

- The code must include at least three colors, at least one loop, and at least one shape filled with color.
- Post-activity, students will pair and exchange code. Each student will incorporate elements from their partner's code. They must give credit to the other student for their contributions to the code via comments. They should use this guide for citing code: https://integrity.mit.edu/handbook/writing-code

Unit **3:** ***"REPEATING," CS Principles, Section 3***

Text Guided Topics/Activities:

- 7 Computers can Repeat Steps
  - Readings, Discussion
- 8 While and For Loops
  - Readings, Videos, Activity
- 9 Repeating Steps with Strings
  - Readings, Videos, Activity, Discussion
- 10 Repeating Steps with Turtles
  - Readings, Videos, Activity
- 11 Repeating Steps with Images
  - Readings

Big Ideas:

CRD, AAP, IOC, CSN

Unit Activities:

- **8, Fibonacci Sequence to the Nth - AAP** CR-5, CR-7
  - Students write a function which takes a positive number, n, and returns the n-th number in the Fibonacci sequence using a for loop.
  - Students will use pseudocode to draft their plan, then code in Python.
  - Students compare their solutions and test them being sure that they work for positive integers of n. They should see what happens when negative numbers or decimals are input and adjust their code if necessary.
- **9, Caesar Cipher - AAP, IOC, CRD** CR-3, CR-4, CR-5, CR-7
  - Students research the Caesar Cipher. The code should prompt the user with options to encode or decode input. Both options take input from the user as a string and encrypt it with a supplied key or decrypt it with a supplied key. The code will return the encrypted or decrypted string.
  - Flowchart:
    - Students will watch:
      5 Minutes to Code: Programming Basics "Flow Charts"

        - Next, they will attempt to create a flowchart for the Splitting the Check activity from the previous unit and Fibonacci activity they completed in this unit. Once they feel confident creating flow charts, they will create one for the Caesar Cipher.
    - After they individually pre-code, students will code this in Python, working in pairs. They will follow guidelines for collaborative work.
    - Debug:
        - Once the code is complete, students will debug their work by adding "print" statements at each step of their code, removing these when they are assured that each step is functioning correctly.
        - While debugging, students will journal about what errors existed in their code and how they tested and addressed them. They will submit their journal entries.
    - Students then write a short paragraph about a humorous or exciting moment in their life. They encrypt this and share it and the key with another student. This student will attempt to decrypt the message using their own code.
    - Reflect:
        - A discussion will follow focusing on the flow chart. Students will share what advantages or challenges using this strategy presented as it pertained to writing their code and how their preparation helped to guide their solution. They will compare this with writing pseudocode. As part of the discussion, other options will be presented, such as using Venn diagrams or other more visual representations of code or data. They should begin to realize that different processes may help lead to solutions more readily for different coding challenges.
- **Cryptography, *Blown to Bits,* Chapter 5 - IOC, CSN** CR-3, CR-8, CR-9, CR-10
    - Students will read p. 161-193 of *Blown to Bits*, the chapter covering Cryptography. The reading will be spread over multiple nights and will prompt discussion about the history of cryptography, the role of computers in cryptography, concerns about encryption, and innovations in cryptography.
    - After the reading, students will be asked to research and report on recent innovations or future projections of data privacy and encryption.
    - They will present to the class the innovation and explain how or why it occurred and what changes it might bring about, both positive and negative.
- **10, Polygon Art Round Two - AAP, CRD** CR-9
    - Coding in pairs, students will create polygons using user defined values for number of vertices, line length, color, pen width, etc.
    - With this and all subsequent coding activities, students will start by making a flowchart, diagram, or using pseudocode.
    - The code will take user provided input for the different possible values and draw a polygon using this input.
    - A discussion will follow where students reflect on their two experiences coding in pairs. They will express thoughts on these questions: How might the program or experience have differed if worked individually? What type of feedback did

working with pairs provide? What additional feedback might be useful to improve a program?

Unit 4: *"DECISIONS," CS Principles, Section 4*

Text Guided Topics/Activities:

- 12 A Computer Can Make Decisions
  - Readings, Videos, Discussion, Activity
- 13 Using Decisions with Strings
  - Readings, Videos, Activity, Discussion
- 14 Using Decisions with Turtles
  - Readings
- 15 Using Decisions with Images
  - Readings, Activity, Discussion

Big Ideas:

CRD, AAP

Unit Activities:

- **12, The Classic Waiter Game - AAP** CR-3, CR-5
  - At our school, the "guess the number" game is used traditionally to choose a waiter at the lunch table.
  - The computer randomly generates a number from 1 to 100, including 100. The user guesses a number.
  - If the guess is incorrect, the computer prints a prompt to guess again, "higher" if the guess was too low, "lower" if too high. This will loop until the correct number is guessed.
  - The game will keep track of the number of guesses and print this number with a positive statement when the user guesses correctly.
  - Students plan and code the game.
- **15, Image Editor - AAP** CR-5
  - Students will code a simple image editor. It will prompt the user for different effects. Students can create their own parameters for these effects or use the ones from their text, such as "posterize" or "pencil sketch." Within the code, students should use lists for the parameters and functions where possible to simplify their code.
  - After debugging, they will test each other's code. A discussion will follow where students reflect on how different functions of photo editors work on their computers and phones. They will talk about how these devices communicate the values for colors and ways of changing these.

Unit 5: *"DATA," CS Principles, Section 5*

Text Guided Topics/Activities:

- 16 Working with Collections
    - Readings, Videos, Discussion, Activities
- 17 Getting Pieces out of Strings and Lists
    - Readings, Videos, Activity, Discussion
- 18 Working with Data on the Web
    - Readings, Videos, Discussion, Activity
- 19 Computer Abilities Summary
    - Readings

Big Ideas:

CRD, AAP, DAT, IOC

Unit Activities:

- **16, The Classic Waiter Game Extended - AAP, CRD** CR-5, CR-6, CR-7
    - The code from the waiter game will be edited for use with a list of strings. Students can choose any list of their making, examples might be colors, sports, pro sports teams, car models, influencers, etc. The computer will randomly generate a string from the list and the user will attempt to guess the correct string.
    - Students will adjust their flowchart/pseudocode to include functions where they might simplify the code. They will also extend the previous code to run in a loop, prompting the user to continue the game if they wish. It should also recall and print the lowest score (lowest number of guesses) if the user chooses to continue the game.
    - They will test each other's code by playing the game.
    - Finally, students will write a paragraph that explains how the code works and how the functions are procedural abstractions that manage the complexity of the program.
- **16, Sorting Algorithms - AAP, IOC, CRD** CR-3, CR-4, CR-5
    - Students should familiarize themselves with different algorithm based sorting methods: insertion, selection, quick, merge, and bubble. - https://medium.com/learning-python-programming-language/sorting-algorithms-insertion-sort-selection-sort-quick-sort-merge-sort-bubble-sort-4f23bda6f37a
    - As a class, each student will be assigned a number and will be randomly ordered in a line.
    - The students will have to demonstrate each sorting algorithm to order themselves from lowest to highest.
    - Discussions will follow as to what is the most efficient and how does the data or size of the list impact which is the most efficient choice. Students will be asked

why sorting is important as it pertains to data. The idea of space complexity will also be introduced.

- **16, Dice Roll Data - DAT, AAP, IOC** CR-4, CR-5, CR-6
  - In groups of two, students will roll a die 100 times and collect data based on the number shown from each roll.
  - They will create code that uses this data and then outputs the probability in percentage of rolling a 1, 2, 3, 4, 5, or 6.
  - They will also print a statement that explicitly tells the user which number they have the highest probability of rolling and the lowest probability of rolling.
  - Once they have finished and submitted this code, they will edit their code to create a list generated by computer generated rolls of the dice using the random library. They should create a list of results from 10000 rolls and return the same output as before - percentages for each roll and print statements.
  - To complete this activity, students will have to research randrange(), min(), max(), and append().
  - The activity will conclude with a brief discussion of what results they would expect in a perfect world and why these differed. They should understand the difference between abstraction and real world data. They will be asked what they could do to come closer to expected results, what the data might look like if graphed or visually represented using iterations as one axis, and where in the real world such simulations might be useful, i.e. when might a computer be useful in solving real world problems.
- **17, 18.1-18.3, Hangman - DAT, AAP** CR-5, CR-7
  - Students will create a text based hangman game in Python.
  - The game will use this .txt for a word list: https://docs.google.com/document/d/1LGKrzri_G_o2IHaJOzsWtBNHZUJF31Ld6KhOm4qrKhs/edit?usp=sharing
  - The game should randomly choose a word from the .txt.
  - The game should show how many known or unknown letters are in the word.
  - It should accept input as letter or word guesses and print whether the guesses are correct or incorrect.
  - If a guess is correct, a positive message should be displayed.
  - If it is incorrect, a count down or visual representation of the hangman should be displayed.
  - Once the word is guessed and the game is won, print a positive message along with the total number of guessed letters. If the player loses, i.e. they reach zero on the count down or the hangman is depicted, a message should be displayed, along with the correct answer.
  - As a bonus or extension of this, using a separate file, a “low guess” record can be kept and displayed at the start or end of the game.
  - The code must use procedures and parameters in addition to the other tools learned in chapters 16 and 17 as they relate to using files, populating lists, and splitting strings.

    - After completing this, students will discuss the different methods they used in their programming around the idea of efficiency. They will consider, what is the most efficient way to code this and, more fundamentally, how do we determine efficiency? Distributive, parallel, and sequential computing will be presented within the discussion.
- **18, Playing with Data - AAP, DAT, CRD** CR-3, CR-4, CR-5, CR-6
    - Students, in pairs, will choose a dataset from this collection: https://www.kaggle.com/code/rtatman/fun-beginner-friendly-datasets
    - Students will explore the dataset by sorting the data and finding min, max, mean, and median values. They will compare and contrast entries by combining values via ratios and percentages.
    - Students will experiment with different chart types for visualizing that data, using https://discovery.cs.illinois.edu/learn/Exploratory-Data-Analysis/Basic-Data-Visualization-in-Python/ as a resource.
    - Students will answer questions in a shared document that ask them to describe patterns and trends in the data. They will be asked to be critical of the data and the different visualizations, asking why they differ and what stories they tell.
    - They will write about different audiences that might benefit from the data and other ways to manipulate or use the data to draw conclusions or to make inferences. Part of this exposition should include what other data might be useful to add to this set and what information might be missing that could lead to misleading results.
    - Each group will present their findings and conclusions to the class with a short video production.

Unit 6: *"WHAT'S POSSIBLE," CS Principles, Section 6*

Text Guided Topics/Activities:

- 20 The Internet
    - Readings, Videos, Discussion, Activity
- 21 Creativity
    - Readings, Videos, Activity
    - ***Create Task - 12 hour minimum***
- 22 Global Impact
    - Readings, Videos, Discussion, Activity
    - ***Explore Task***
- 23 What's Next
    - Readings, Videos, Activity

Big Ideas:

CRD, IOC, CSN

Unit Activities:

- **20, How Information Moves Through the Internet - IOC, CSN** CR-3, CR-5
    - Read Appendix in *Blown to Bits*, p. 301-316, The Internet as System and Spirit.
    - Working in pairs, students will
        - Design a flowchart in Google Slides which demonstrates how information is passed from a sending machine to a receiving machine through the internet.
        - Visit the Maker Space and create a physical representation of the flowchart. Students can use any parts from take-apart events to represent the various computers and routers, wires to demonstrate connectivity, and labels to explain briefly what is happening at each stage.
    - The project will culminate with a discussion where students extend their learning to consider where, when, and how this process can be disrupted. They will be asked what places or people have limited access to technology and what aspects from their flowchart might be missing in those areas. They will be asked to brainstorm solutions to this issue within their groups and present to class.
- **21, Performance Task: Create** CR-11
    - Students will be given 12 hours of class time to complete, approximately 4 weeks. (3 hours, 20 minutes per week).
- **22, Global Impact of Computing - IOC, CRD, CSN** CR-8, CR-10
    - Each student visits and familiarizes themselves with Google Trends: https://trends.google.com/
    - In groups of three, students will choose and research trends regarding one of the following:
        - different social media platforms
        - different phone brands
        - different popular email interfaces or systems
    - Students will report their observations of the data based on:
        - Worldwide trends
        - Comparison of any two countries (both must be of similar GDP, i.e. first, second, or third world)
    - Students will report their opinions and research based on:
        - Similarities, differences, and patterns they observe in the data
        - Moral implications of that type of technology based on the current trends - what is the “good” and “bad” of this type of technology.
        - Expectations of future trends - what they project will change and why, and how future innovations might impact current moral implications.
- **22, Performance Task: Explore**
    - Students will be provided with 8 hours of class time to complete, approximately 2 ½ weeks. (3 hours, 20 minutes of class per week.)