AP® Computer Science Principles

PREVIEW REVISED COURSE FRAMEWORK

For Use Beginning with the 2027–2028 School Year

What AP® Stands For

Thousands of Advanced Placement teachers have contributed to the principles articulated here. These principles are not new; they are, rather, a reminder of how AP already works in classrooms nationwide. The following principles are designed to ensure that teachers' expertise is respected, required course content is understood, and students are academically challenged and free to make up their own minds.

1. AP stands for clarity and transparency. Teachers and students deserve clear expectations. The Advanced Placement Program makes public its course frameworks and sample assessments. Confusion about what is permitted in the classroom disrupts teachers and students as they navigate demanding work.
2. AP is an unflinching encounter with evidence. AP courses enable students to develop as independent thinkers and to draw their own conclusions. Evidence and the scientific method are the starting place for conversations in AP courses.
3. AP opposes censorship. AP is animated by a deep respect for the intellectual freedom of teachers and students alike. If a school bans required topics from their AP courses, the AP Program removes the AP designation from that course and its inclusion in the AP Course Ledger provided to colleges and universities. For example, the concepts of evolution are at the heart of college biology, and a course that neglects such concepts does not pass muster as AP Biology.
4. AP opposes indoctrination. AP students are expected to analyze different perspectives from their own, and no points on the AP Exam are awarded for agreement with any specific viewpoint. AP students are not required to feel certain ways about themselves or the course content. AP courses instead develop students' abilities to assess the credibility of sources, draw conclusions, and make up their own minds.

   As the AP English Literature course description states: "AP students are not expected or asked to subscribe to any one specific set of cultural or political values, but are expected to have the maturity to analyze perspectives different from their own and to question the meaning, purpose, or effect of such content within the literary work as a whole."
5. AP courses foster an open-minded approach to the histories and cultures of different peoples. The study of different nationalities, cultures, religions, races, and ethnicities is essential within a variety of academic disciplines. AP courses ground such studies in primary sources so that students can evaluate experiences and evidence for themselves.
6. Every AP student who engages with evidence is listened to and respected. Students are encouraged to evaluate arguments but not one another. AP classrooms respect diversity in backgrounds, experiences, and viewpoints. The perspectives and contributions of the full range of AP students are sought and considered. Respectful debate of ideas is cultivated and protected; personal attacks have no place in AP.
7. AP is a choice for parents and students. Parents and students freely choose to enroll in AP courses. Course descriptions are available online for parents and students to inform their choice. Parents do not define which college-level topics are suitable within AP courses; AP course and exam materials are crafted by committees of professors and other expert educators in each field. AP courses and exams are then further validated by the American council on Education and studies that confirm the use of AP scores for college credits by thousands of colleges and universities nationwide.

The AP Program encourages educators to review these principles with parents and students so they know what to expect in an AP course. Advanced Placement is always a choice, and it should be an informed one. AP teachers should be given the confidence and clarity that once parents have enrolled their child in an AP course, they have agreed to a classroom experience that embodies these principles.

© 2026 The Advanced Placement Program, AP, AP Central, and the acorn logo are registered trademarks of the College Board. All other products and services may be trademarks of their respective owners. Visit College Board on the web: **collegeboard.org**.

Contents

iv **Acknowledgements**

1 **About AP**

5 **About the Computer Science Principles Course**

COURSE FRAMEWORK

8 **Course Framework Components**

9 **Computational Thinking Practices**

11 **UNIT 1:** Creating Purposeful Apps and Programs

27 **UNIT 2:** Building Solutions Using Selection and Iteration

41 **UNIT 3:** Developing with Abstraction Using Lists and Procedures

51 **UNIT 4:** Making Data-Driven Decisions Using the Internet and AI

69 **UNIT 5:** Exploring Computing Technologies

Acknowledgements

College Board would like to acknowledge the following committee members, contributors, and reviewers for their assistance with and commitment to the development of this course. All individuals and their affiliations were current at the time of contribution.

Christine Alvarado, University of California San Diego, San Diego, CA

Jeremy Brown, Rochester Institute of Technology, Rochester, NY

Joy Beckenbach, Providence Hall High School, Herriman, UT

Elizabeth Begley, Cherry Hill High School West, Cherry Hill, NJ

Daniel Bonarigo, Riverside Brookfield High School, Chicago, IL

Miriam Brady, Microsoft, Redmond, WA

Jacqueline Corricelli, West Hartford Public Schools, West Hartford, CT

Rebecca Cunningham, Arkansas Tech University, Russellville, AR

Toby Dragon, Ithaca College, Ithaca, NY

Mark Emry, University of Texas, Austin, TX

Jamie Fagan, Webster Thomas High School, Webster, NY

Joelle Henry, Virtual Learning Academy, Exeter, NH

Scott Johnson, Rochester Institute of Technology, Rochester, NY

Douglas Kiang, Punahou High School, Honolulu, HI

Carol Kim, IBM, Armonk, NY

Cheryl Swanier, University of Massachusetts Amherst, Amherst, MA

Melodie Ting, The Browning School, New York, NY

Thomas Thompson, Calvert High School, Prince Frederick, MD

Frances Trees, Rutgers University, Piscataway, NJ

Paul Tymann, Rochester Institute of Technology, Rochester, NY

Michael Vlach, Dixon High School, Dixon, CA

Mindy Walski, Montgomery Catholic Preparatory School, Montgomery, AL

College Board Staff

Daniel Klag, Senior Director, AP Math, Computer Science, and Cyber Generalist, Curriculum & Assessment

Cynthia Mitchell, Senior Director, Instructional Design

Beth Brown, Director, Product Development and Editorial

Valerie Barron, Executive Director, New Course Design

Kim Vinh, Senior Director, Instructional Design

Andrea Pellerito, Senior Director, Course Product Manager

Daniel McDonough, Senior Director, AP Content and Assessment Publications

Dana Kopelman, Vice President, Program Enablement

Allison Thurber, Vice President, AP Curriculum and Assessment

Ben Dougherty, Senior Director, AP Computer Science and Cyber Curriculum and Assessment

Jason VanBilliard, Executive Director, Assessment Innovation & Department Head, AP Math, CS, and Cyber Curriculum and Assessment

© 2026 College Board

About AP

The Advanced Placement® Program (AP®) enables willing and academically prepared students to pursue college-level studies—with the opportunity to earn college credit, advanced placement, or both—while still in high school. Through AP courses in more than 40 subjects, each culminating in a challenging exam, students learn to think critically, construct solid arguments, and see many sides of an issue—skills that prepare them for college and beyond. Taking AP courses demonstrates to college admission officers that students have sought the most challenging curriculum available to them, and research indicates that students who score a 3 or higher on an AP Exam typically experience greater academic success in college and are more likely to earn a college degree than non-AP students. Each AP teacher's syllabus is evaluated and approved by faculty from some of the nation's leading colleges and universities, and AP Exams are developed and scored by college faculty and experienced AP teachers. Most four-year colleges and universities in the United States grant credit, advanced placement, or both on the basis of successful AP Exam scores—more than 3,300 institutions worldwide annually receive AP scores.

AP Course Development

In an ongoing effort to maintain alignment with best practices in college-level learning, AP courses and exams emphasize challenging, research-based curricula aligned with higher education expectations.

Individual teachers are responsible for designing their own curriculum for AP courses, selecting appropriate college-level readings, assignments, and resources. This course and exam description presents the content and skills that are the focus of the corresponding college course and that appear on the AP Exam. It also organizes the content and skills into a series of units that represent a sequence found in widely adopted college textbooks and that many AP teachers have told us they follow to focus their instruction. The intention of this publication is to respect teachers' time and expertise by providing a roadmap that they can modify and adapt to their local priorities and preferences. Moreover, by organizing the AP course content and skills into units, the AP Program is able to provide teachers and students with free formative assessments—Progress Checks—that teachers can assign throughout the year to measure student progress as they acquire content knowledge and develop skills.

Enrolling Students: Access, Opportunity, and Readiness

The AP Program welcomes all students willing to challenge themselves with college-level coursework and career preparation. We strongly encourage educators to invite students into AP classes, including students from ethnic, racial, socioeconomic, geographic, or other groups not broadly participating in a school's AP program. We believe that readiness for AP is attainable, and that educators can expand readiness by opening access to Pre-AP course work. We commit to support educators and communities in their efforts to make AP courses widely available, advancing students in their plans for college and careers.

© 2026 College Board

Offering AP Courses: The AP Course Audit

The AP Program unequivocally supports the principle that each school implements its own curriculum that will enable students to develop the content understandings and skills described in the course framework.

While the unit sequence represented in this publication is optional, the AP Program does have a short list of curricular and resource requirements that must be fulfilled before a school can label a course "Advanced Placement" or "AP." Schools wishing to offer AP courses must participate in the AP Course Audit, a process through which AP teachers' course materials are reviewed by college faculty. The AP Course Audit was created to provide teachers and administrators with clear guidelines on curricular and resource requirements for AP courses and to help colleges and universities validate courses marked "AP" on students' transcripts. This process ensures that AP teachers' courses meet or exceed the curricular and resource expectations that college and secondary school faculty have established for college-level courses.

The AP Course Audit form is submitted by the AP teacher and the school principal (or designated administrator) to confirm awareness and understanding of the curricular and resource requirements. A syllabus or course outline, detailing how course requirements are met, is submitted by the AP teacher for review by college faculty.

Please visit the **AP Course Audit** website for more information to support the preparation and submission of materials for the AP Course Audit.

How the AP Program Is Developed

The scope of content for an AP course and exam is derived from an analysis of hundreds of syllabi and course offerings of colleges and universities. Using this research and data, a committee of college faculty and expert AP teachers work within the scope of the corresponding college course to articulate what students should know and be able to do upon the completion of the AP course. The resulting course framework is the heart of this course and exam description and serves as a blueprint of the content and skills that can appear on an AP Exam.

The AP Test Development Committees are responsible for developing each AP Exam, ensuring the exam questions are aligned to the course framework. The AP Exam development process is a multiyear endeavor; all AP Exams undergo extensive review, revision, piloting, and analysis to ensure that questions are accurate, fair, and valid, and that there is an appropriate spread of difficulty across the questions.

Committee members are selected to represent a variety of perspectives and institutions (public and private, small and large schools and colleges), and a range of gender, racial/ethnic, and regional groups. A list of each subject's current AP Test Development Committee members is available on **AP Central®**.

Throughout AP course and exam development, College Board gathers feedback from various stakeholders in both secondary schools and higher education institutions. This feedback is carefully considered to ensure that AP courses and exams are able to provide students with a college-level learning experience and the opportunity to demonstrate their qualifications for advanced placement and/or college credit.

How AP Exams Are Scored

The exam scoring process, like the course and exam development process, relies on the expertise of both AP teachers and college faculty. While multiple-choice questions are scored by machine, the free-response questions and through-course

© 2026 College Board

performance assessments, as applicable, are scored by thousands of college faculty and expert AP teachers. Most are scored at the annual AP Reading, while a small portion is scored online. All AP Readers are thoroughly trained, and their work is monitored throughout the Reading for fairness and consistency. In each subject, a highly respected college faculty member serves as Chief Faculty Consultant and, with the help of AP Readers in leadership positions, maintains the accuracy of the scoring standards. Scores on the free-response questions and performance assessments are weighted and combined with the results of the computer-scored multiple-choice questions, and this raw score is converted into a composite AP score on a 1–5 scale.

AP Exams are **not** norm referenced or graded on a curve. Instead, they are criterion referenced, which means that every student who meets the criteria for an AP score of 2, 3, 4, or 5 will receive that score, no matter how many students that is. The criteria for the number of points a student must earn on the AP Exam to receive scores of 3, 4, or 5—the scores that research consistently validates for credit and placement purposes—include:

- The number of points successful college students earn when their professors administer AP Exam questions to them
- Performance that researchers have found to be predictive of an AP student succeeding when placed into a subsequent higher-level college course
- The number of points college faculty indicate, after reviewing each AP question, that they expect is necessary to achieve each AP grade level

Using and Interpreting AP Scores

The extensive work done by college faculty and AP teachers in the development of the course and exam and throughout the scoring process ensures that AP Exam scores accurately represent students' achievement in the equivalent college course. Frequent and regular research studies establish the validity of AP scores as follows:

AP Score	Credit Recommendation	College Grade Equivalent
5	Extremely well qualified	A
4	Well qualified	A-, B+, B
3	Qualified	B-, C+, C
2	Possibly qualified	n/a
1	No recommendation	n/a

While colleges and universities are responsible for setting their own credit and placement policies, most private colleges and universities award credit and/or advanced placement for AP scores of 3 or higher. Additionally, most states in the U.S. have adopted statewide credit policies that ensure college credit for scores of 3 or higher at public colleges and universities. To confirm a specific college's AP credit/ placement policy, use the search engine available on the **AP Credit Policy Search** page.

BECOMING AN AP READER

Each June, thousands of AP teachers and college faculty members from around the world gather for seven days in multiple locations to evaluate and score the free-response sections of the AP Exams. Ninety-eight percent of surveyed educators who took part in the AP Reading say it was a positive experience.

© 2026 College Board

There are many reasons to consider becoming an AP Reader, including opportunities to:

- **Bring positive changes to the classroom:** Surveys show that the vast majority of returning AP Readers—both high school and college educators—make improvements to the way they teach or score because of their experience at the AP Reading.
- **Gain in-depth understanding of AP Exam and AP scoring standards:** AP Readers gain exposure to the quality and depth of the responses from the entire pool of AP Exam takers and thus are better able to assess their students' work in the classroom.
- **Receive compensation:** AP Readers are compensated for their work during the Reading. Expenses, lodging, and meals are covered for Readers who travel.
- **Score from home:** AP Readers have online distributed scoring opportunities for certain subjects. Check the **AP Reader** site for details.
- **Earn Continuing Education Units (CEUs):** AP Readers earn professional development hours and CEUs that can be applied to PD requirements by states, districts, and schools.

HOW TO APPLY

Visit the **Become an AP Reader** site for eligibility requirements and to start the application process.

© 2026 College Board

About the Computer Science Principles Course

Computer science involves problem-solving, hardware, and algorithms that help people use computers and incorporate multiple perspectives to address real-world problems in contemporary life. As the application of computer science is integrated into more aspects of daily life, people must understand the impact of computer science and how to maintain privacy, safety, and security not only when using computers but also while innovating new computing applications. This course strives to engage all students, including those who have traditionally been underrepresented in computer science, by allowing them to discover the power of computer science through rewarding yet challenging concepts.

Guidance and requirements for programming languages are available on **AP Central**.

College Course Equivalent

AP Computer Science Principles is equivalent to an introductory, college-level breadth course in computer science.

Prerequisites

There are no specific course prerequisites for AP Computer Science Principles. Students should be motivated and willing to work both individually and collaboratively on college-level programming projects.

Course Alignment

The AP Computer Science Principles course aligns with the core concepts in the **Computer Science Teachers Association (CSTA) K–12 Computer Science Framework**.

AP Computer Science Principles is one of two computer science courses available to AP students. The AP Computer Science A course complements AP Computer Science Principles through a focused study of algorithms and programming. Students can take these two courses in either order or concurrently, as allowed by their school.

Resource Requirements

Students are required to have access to a computer system that contains the software needed to create and edit programs and to practice, complete, and submit the AP Computer Science Principles project. The computer must have Internet access and be able to reach the sites students need to be successful in the course and on assessments. Students should also be provided with the AI tools necessary to complete required AI application activities and labs. Details about AI policies are available on AP Central.

Because AP Computer Science Principles does not require a particular programming language for instruction, exam questions are written in a pseudocode designed to be readable by students regardless of the programming language used in their classroom. Each student must have access

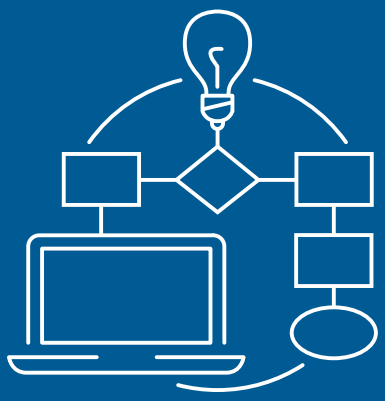
© 2026 College Board

to the AP Computer Science Principles Exam Reference Sheet and project directions. Each student should also receive a college-level textbook—or other curricular resources the teacher deems necessary—for individual use both in class and outside the classroom.

Pseudocode

The pseudocode used in this framework provides a common notation for instruction and assessment in AP Computer Science Principles. It is not intended to represent a complete programming language. Although the pseudocode and related Essential Knowledge statements reflect common programming concepts, students need to understand that programming languages, including the one used in their classroom, may vary in syntax and rules. The syntax and rules described in this framework are consistent with many programming languages but are not universal to all languages.

© 2026 College Board



AP COMPUTER SCIENCE PRINCIPLES

Course Framework

Course Framework Components

Course Units

Unit 1: Creating Purposeful Apps and Programs

Unit 2: Building Solutions Using Selection and Iteration

Unit 3: Developing with Abstraction Using Lists and Procedures

Unit 4: Making Data-Driven Decisions Using the Internet and AI

Unit 5: Exploring Computing Technologies

Curriculum Framework Overview

This course framework provides a clear and detailed description of the course requirements necessary for student success. The framework specifies what students must know, be able to do, and understand to qualify for college credit and/or placement.

The course framework includes two essential components:

- **Computational Thinking Practices**
  The computational thinking practices are central to the study and practice of computer science. Students should practice and develop these skills on a regular basis over the span of the course.
- **Course Content**
  The course content is organized into commonly taught units that reflect key domains of computer science knowledge and practice. These units comprise the content and skills in which colleges and universities typically expect students to be proficient to qualify for college credit and/or placement. Units 1–4 introduce students to computing and present the knowledge and skills needed to design and develop programs, build and refine algorithmic solutions, and examine the impacts of computing and artificial intelligence on individuals and society. An optional, nonassessed Unit 5 extends the course into foundational computing topics and may be taught at any point during the course.

© 2026 College Board

Computational Thinking Practices

Practice 1	Practice 2	Practice 3	Practice 4	Practice 5
Design Code 1 Determine an appropriate program design and develop algorithms.	**Develop Code** 2 Write and implement program code.	**Analyze Code** 3 Determine the output or result of given program code or explain why code may not work as intended.	**Document Code and Computing Systems** 4 Describe the behavior and conditions that produce specified results in a program.	**Use Computers Responsibly** 5 Understand the ethical and social implications of computer use.
SKILLS				
1.A Determine an appropriate program design to solve a problem or accomplish a task.	2.A Write program code, with and without AI tools, to implement an algorithm.	3.A Determine the result or output based on statement execution order in an algorithm.	4.A Describe the behavior of a code segment or program.	5.A Explain how computing impacts society, the economy, and culture.
1.B Determine what knowledge can be extracted from data.	2.B Write program code, with and without AI tools, involving data abstractions.	3.B Determine the result or output based on code that contains data abstractions.	4.B Describe the initial conditions that must be met for a code segment to work as intended or described.	5.B Explain how legal, ethical, and security concerns influence the use of computing.
1.C Determine the level of AI automation appropriate to a task.	2.C Write program code, with and without AI tools, involving procedural abstractions.	3.C Determine the result or output based on code that contains procedural abstractions.	4.C Describe appropriate attribution and documentation when using AI and external sources.	5.C Evaluate data sets for privacy considerations and potential bias.
1.D Represent algorithmic processes without using a programming language.		3.D Explain why a code segment will not compile or work as intended and modify the code to correct the error.	4.D Explain how computing systems work.	5.D Explain how collaboration affects the development of a solution.
		3.E Explain how specific abstractions manage complexity in code.		5.E Explain how probabilistic algorithms in AI models can lead to variable outcomes.
		3.F Evaluate the accuracy of program code, generated with and without AI tools, and the resulting outputs.		

© 2026 College Board

THIS PAGE HAS BEEN INTENTIONALLY LEFT BLANK

AP COMPUTER SCIENCE PRINCIPLES

UNIT 1

Creating Purposeful Apps and Programs

TOPIC 1.1
Purpose and Behavior

Required Course Content

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
1.1.A Determine the purpose and behavior of an app by interacting with it.	**1.1.A.1** Applications (apps) are programs that run on many kinds of computing devices. Apps can run on smartphones, tablets, laptops, desktops, game consoles, and web browsers, and they are designed to help users accomplish tasks or goals. **1.1.A.2** People use apps to help solve real problems and accomplish everyday tasks, such as communicating, learning, navigating, creating and sharing media, staying organized, and being entertained. **1.1.A.3** An app is successful when it uses information to produce an output that helps a user achieve a goal. An app's value to the user depends on how well the results meet the user's needs. **1.1.A.4** Because people rely on apps for everyday tasks, app behavior matters. When an app is confusing, unreliable, or does not work as expected, users can become frustrated. **1.1.A.5** An app's behavior is what it does while it is running, including how it responds to user interactions. The behavior of an app is determined by its program code. **1.1.A.6** An app should behave as intended across a range of conditions, including the different ways people may interact with it. **1.1.A.7** Users can understand an app's purpose and behavior by using it and observing how it responds to their actions.
1.1.B Identify inputs and outputs of an app while using it.	**1.1.B.1** Apps are designed to take in information from a variety of sources, process that information, and respond in useful ways. **1.1.B.2** When users interact with apps, they often tap or click buttons, type letters or numbers, speak, or move the device. Each of these actions provides input: data sent to a computer for a program to process. **1.1.B.3** Apps should provide clear feedback when they receive input, such as highlighting a button, showing a loading icon, playing a sound, vibrating, or displaying an error message. This feedback is a form of output that helps users know the app received the input, the app is working, and what step users should take next.

© 2026 College Board

LEARNING OBJECTIVE

1.1.B

Identify inputs and outputs of an app while using it.

ESSENTIAL KNOWLEDGE

1.1.B.4

Input often comes from a user's direct actions, but it can also be captured automatically from sources such as sensors (e.g., an accelerometer), location, time, or network signals. Apps may also receive audio or images from microphones and cameras, often through user actions (e.g., a voice command or face scan). These inputs allow apps to provide features like maps, step counting, face filters, or recommendations.

1.1.B.5

In many apps, the program responds to events, such as button clicks, key presses, or sensor updates, by running specific code. Events are one way input is delivered to a program. An event signals that something happened, and an event handler processes the input and determines what happens next.

1.1.B.6

Outputs are what an app produces. Outputs may include text, images, sounds, or something the user feels, like a vibration. Some outputs are directly perceived by the user, while others are not, such as data written to a file or sent over a network.

1.1.B.7

Users can identify inputs and outputs by trying different actions in an app and observing what the app displays or does in response.

© 2026 College Board

TOPIC 1.2
Benefits of Collaboration

Required Course Content

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
1.2.A Explain how apps and programs are improved through collaboration and user testing.	**1.2.A.1** User testing with a variety of users helps programmers design an app with a clear purpose that is intuitive for its intended users and minimizes confusion. **1.2.A.2** Programmers can conduct user testing to understand how people use an app, where users are confused, or if users have unmet needs. Observing participants during development provides evidence programmers can use to refine the app's purpose and design. **1.2.A.3** Working collaboratively on an app brings together a variety of talents and perspectives that can improve the final product. Teams can divide tasks, share ideas, and review and document decisions as they develop the app. **1.2.A.4** Working collaboratively gives programmers opportunities to share roles throughout development. Pair programming is a collaboration practice in which two programmers work together at one computer: one writes the code while the other reviews the code as it is written, checks for errors, and keeps the bigger picture in mind. Partners switch roles frequently so both contribute to writing code and problem-solving. **1.2.A.5** Apps and programs are improved through an iterative process. Teams use feedback from user testing and collaboration to make changes and test again to verify an app more accurately meets user needs.
1.2.B Explain how collaboration can reduce bias in apps.	**1.2.B.1** Bias occurs when an app consistently favors certain people, values, or outcomes over others. Bias can be harmful because it can make results less accurate for some users and can contribute to unfair treatment or unequal access. **1.2.B.2** Apps can reflect human bias when programmers' assumptions, rules, or interface choices favor some users over others. Apps can also reflect data bias when the data used to train or inform the app is incomplete or unrepresentative of the people and situations the app is intended to work well for. Human bias and data bias often interact. Choices about what data to collect and how to label it can shape how an app behaves for different users. **1.2.B.3** Collaboration can help reduce bias by bringing multiple perspectives into design and testing. Teams can review design decisions and data together, look for missing perspectives, and gather feedback from a variety of users. **1.2.B.4** When potential bias is identified, teams can revise the rules, data, or design and then test again to check whether outcomes are more accurate and fair across different users and situations.

© 2026 College Board

TOPIC 1.3

Responsible AI and Human Decision Making

Required Course Content

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
1.3.A Explain how human decisions shape the development, deployment, and maintenance of AI tools.	**1.3.A.1** AI tools can affect everyday choices such as what people watch, buy, and learn. Understanding the human decisions behind AI helps users evaluate outputs critically and helps developers design and maintain tools that are safer, more accurate, and more fair. **1.3.A.2** Artificial intelligence (AI) refers to computer systems that use data and patterns to perform tasks that are often associated with human intelligence, producing outputs such as predictions, recommendations, decisions, or generated content. **1.3.A.3** In AI systems, a model is the part of the system used to produce outputs from inputs. Models may follow rules created by people or learn patterns from data. **1.3.A.4** Many modern AI models are created by training on examples. The quality and representativeness of the training data can affect how accurate or fair the model's outputs are. **1.3.A.5** Generative AI is a type of AI that creates new content, such as text, images, music, or code, based on patterns it has learned from examples. Some generative AI tools are powered by large language models (LLMs), which generate text by predicting likely sequences of words based on patterns learned from large collections of text. **1.3.A.6** Humans decide what happens with an AI tool's output, including when to trust it, what actions to take based on it, and when a human should review or make the final decision. **1.3.A.7** When AI tools are used to create or transform content, humans must consider what content may be used and how outputs may be shared. Existing licenses must be respected and attribution given when required. **1.3.A.8** AI tools can generate realistic images, audio, or video that may be mistaken for authentic media (known as deepfakes). Humans must make decisions about safeguards, such as disclosure, verification, and responsible sharing, to reduce misinformation and harm. **1.3.A.9** Model drift occurs when a model's performance changes over time because the real-world data or conditions it encounters shift, potentially making the model less accurate. AI systems should be monitored by humans to detect model drift. When drift is detected, humans can update data, adjust the model, or retrain the system to reduce inaccurate or harmful outcomes.

© 2026 College Board

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
1.3.A Explain how human decisions shape the development, deployment, and maintenance of AI tools.	**1.3.A.10** Developers can increase transparency by clearly describing an AI tool's purpose, development, training, and evaluation, as well as its limitations and appropriate uses. This allows users to make informed choices.
1.3.B Evaluate the appropriate level of automation or augmentation for a task.	**1.3.B.1** Decisions about automation and augmentation affect everyday experiences, such as recommendations, autocorrect suggestions, and navigation directions. Understanding whether a system is acting automatically or supporting human decision making helps users decide how much to rely on its output and when they should verify it, especially when errors, bias, or other limitations could affect the result. **1.3.B.2** Automation allows computers to carry out tasks with limited human involvement, often making work faster, more consistent, and more efficient. Automated systems can handle routine tasks and, in some cases, more complex ones. They may still produce inaccurate results when inputs are ambiguous, conditions change, or unusual cases are not handled well. **1.3.B.3** Augmentation uses tools or systems to support human work while humans remain responsible for decisions and outcomes. In some cases, augmentation involves AI, such as in human-in-the-loop systems where AI outputs are reviewed by humans to improve accuracy and accountability. **1.3.B.4** The appropriateness of automation or augmentation depends on the context and the consequences of error. Although automation can be faster, more consistent, or easier to scale, it may be less appropriate in situations that are high stakes, ambiguous, or require human judgment, empathy, or discretion. In those situations, augmentation may be preferred so that humans remain responsible for the final decision. **1.3.B.5** Evaluating the appropriate level of automation or augmentation requires weighing benefits such as speed or efficiency against risks such as bias, inaccuracy, privacy, or overreliance, including how thresholds affect outcomes. It also means considering social, environmental, and economic trade-offs, such as impacts on people, resource and energy use, costs, labor, and access. **1.3.B.6** Evaluating automation or augmentation should include consideration of how the choice affects user experience, including trust, transparency, and whether users can reasonably decline or reduce automation (e.g., choosing manual controls or requesting human review when needed).

© 2026 College Board

TOPIC 1.4
Binary Representation

Required Course Content

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
1.4.A Explain how computers store and process data as bits and explain how those bits are represented as meaningful digital information for users.	**1.4.A.1** Computing devices represent data digitally. The fundamental unit of digital data is the bit (short for binary digit), which is commonly represented as a value of 0 or 1. **1.4.A.2** Computers store and transmit data as bits, but layers of abstraction allow users to work with meaningful digital information (e.g., text, images, and sound) without dealing directly with 0s and 1s. Abstraction is the process of reducing complexity by focusing on the most important features of something while hiding details that are not needed for a given purpose. Layers of abstraction are designed so people can use technology effectively without needing to understand how the bits are stored or processed. **1.4.A.3** When a user interacts with a computing device, inputs are encoded as bits so a program can process them. The program then produces outputs by translating bits back into forms that users can see or hear. **1.4.A.4** The same sequence of bits can represent different kinds of data depending on the context. An encoding is a set of rules for mapping bits to meaning. Encodings are a layer of abstraction; they let programs interpret and present bits in a user-friendly form. File types and formats signal which encoding rules to use, such as interpreting bits as letters in a text file or colors in an image file.
1.4.B Explain why representing numbers with limited bits can produce inaccurate results.	**1.4.B.1** In many programming languages, numbers can be represented in different ways depending on the need. Some number representations use a fixed number of bits, which limits the range and precision of values that can be stored. As a result, some calculations may produce values that cannot be represented exactly. **1.4.B.2** A round-off error can occur when a computer represents a real number with a fixed number of bits. Because some real numbers cannot be represented exactly, they may be stored as approximations. Calculations that use these approximations can produce results that differ from the mathematically exact value. Over many calculations, small differences can accumulate into a noticeable error, such as totals being off by a few cents or a location estimate being slightly shifted. **EXCLUSION STATEMENT** Specific range limitations for real numbers are outside the scope of the AP Computer Science Principles course and exam.

© 2026 College Board

LEARNING OBJECTIVE

1.4.B

Explain why representing numbers with limited bits can produce inaccurate results.

ESSENTIAL KNOWLEDGE

1.4.B.3

An overflow error occurs when a calculation produces a value outside the range that can be represented with the available bits. This can produce an incorrect result, such as a counter that unexpectedly resets or becomes negative.

1.4.B.4

Some programming languages support an abstraction in which integers can grow to any size limited only by available memory, which can prevent errors caused by overflow.

NOTE

The language on the exam reference sheet uses this abstraction.

© 2026 College Board

TOPIC 1.5
Sequencing

Required Course Content

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
1.5.A Represent an algorithm that uses sequencing using written steps or a flowchart.	**1.5.A.1** Many tasks can be completed by following step-by-step instructions. An algorithm describes these steps in an order that can be followed to solve a problem. **1.5.A.2** Algorithms can be represented in ways people can read and follow, such as written steps or flowcharts. **1.5.A.3** When representing an algorithm in written steps or a flowchart, each step should be clear and specific so the process can be followed correctly. **1.5.A.4** Sequencing means the steps of an algorithm are carried out in a specific order, one step at a time. Changing the order of the steps can change what happens or whether the task is completed correctly.
1.5.B Develop code that runs sequentially, and determine the result of sequential statements.	**1.5.B.1** When an app runs, a computer executes algorithms that are expressed as program code. **1.5.B.2** Programmers write code using tools such as text editors or integrated development environments (IDEs) that help them write and run programs. Some tools include AI features that can generate or autocomplete code, which programmers can adapt and test as they develop programs. **1.5.B.3** Clarity and readability are important when expressing an algorithm in a programming language so others can understand and modify the code. **1.5.B.4** Sequential statements execute in the order they appear in the program, and changing the order of statements can change the program's behavior or what the user sees. **1.5.B.5** Tracing program execution means following which statements are executed and what values are produced and stored as the program runs.

© 2026 College Board

LEARNING OBJECTIVE

1.5.B

Develop code that runs sequentially, and determine the result of sequential statements.

ESSENTIAL KNOWLEDGE

1.5.B.6

Programmers can trace execution by using debuggers or print statements to display values at different points in a program. When the following code runs, the program displays the value of `expression` followed by a new line.

```
print(expression)
```

NOTE

The exam reference sheet provides `print(expression)` to display the value of `expression` followed by a new line.

1.5.B.7

The result of sequential statements can be determined by tracing code in order and keeping track of the values and output.

1.5.B.8

Programmers trace sequential code to confirm an app works as intended and the information it displays is correct and understandable.

© 2026 College Board

TOPIC 1.6
Variables

Required Course Content

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
1.6.A Identify the use of variables in a program.	**1.6.A.1** A variable is a named storage location used to hold a value that can change while the program is running. It is an abstraction that lets programmers use a name to represent a value.
	1.6.A.2 Programs use variables to keep track of changing information by updating their values as the user interacts with the program. Programs can then use a variable's current value to decide what to do next and what feedback to provide, such as updating the screen, playing a sound, or triggering a vibration.
	1.6.A.3 Every value in a program has a data type, such as a number, Boolean, or string. Some programming languages require programmers to specify the type, while others determine it automatically by how the data is used.
	1.6.A.4 The choice of data type depends on what a value represents. Values used to count or measure are best represented as numbers, values that record a true or false condition are best represented as Booleans, and values that store text are best represented as strings. The data type affects what operations can be performed on the variable and how accurately the output is represented in an app.
	1.6.A.5 Variables are often used to store information about the user or a program's state, such as a username, a selected theme, a score, progress toward a goal, a shopping cart total, or whether notifications are on. The program can use the stored value to adjust its behavior or display the value to the user.
	1.6.A.6 Program state is the set of values and information a program is storing at a specific moment, including variable values and any inputs it is using. As the program runs, the state changes when inputs are received and variables are updated.
	1.6.A.7 Users can infer the use of variables in an app by observing information that changes over time, such as counters, totals, settings, progress, or status, and reasoning about what values the app must store and update to produce those changes.

© 2026 College Board

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
1.6.B Develop code that uses assignment statements, and determine the result of these statements.	**1.6.B.1** The assignment operator allows a program to initialize or change the value stored in a variable. When the following code runs, `expression` is evaluated, and the resulting value is stored in the variable `a`. `a = expression` **NOTE** The exam reference sheet provides the assignment statement `a = expression` to assign a value to a variable. It evaluates `expression` and assigns a copy of the result to `a`. **1.6.B.2** Variables are initialized by assigning an initial value before they are used. After initialization, later statements can use the variable's value and update it as the program runs. **1.6.B.3** Using meaningful variable names improves readability and helps programmers understand what the stored values represent. **1.6.B.4** A variable stores its most recently assigned value. **1.6.B.5** The value stored in a variable after a sequence of assignment statements can be determined by tracing statements in order and updating the variable's value after each assignment. **1.6.B.6** Assignment statements are often used to update information an app remembers about the user or the program's state, such as a score or setting, so the app responds appropriately as the user interacts with it.
1.6.C Develop code to store and modify strings, and determine the output produced.	**1.6.C.1** A string is an ordered sequence of characters used to represent text in a program. **1.6.C.2** Apps often use strings to store and display user-facing information, such as names, messages, labels, or search terms. **1.6.C.3** To initialize a string variable, programmers assign it an initial text value or an empty string before it is used so the program starts with a known value. An empty string has zero characters (`""`). It is not the same as a string that contains spaces (`" "`). **1.6.C.4** String concatenation joins two or more strings to create a new string, such as combining a greeting with a user's name. **1.6.C.5** A substring is part of an existing string. Programs can create substrings to extract or use part of a text value, such as extracting the month from a date, reading a portion of a message, or checking for a word within a string.

© 2026 College Board

LEARNING OBJECTIVE

1.6.C

Develop code to store and modify strings, and determine the output produced.

ESSENTIAL KNOWLEDGE

1.6.C.6

The output of code that uses string operations can be determined by tracing statements in order and keeping track of how string values change and what text is displayed.

1.6.C.7

To communicate clearly with users, programmers use strings to store and generate text that apps display, such as prompts, feedback messages, and personalized responses.

© 2026 College Board

TOPIC 1.7
Mathematical Expressions

Required Course Content

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
1.7.A Develop code that uses mathematical expressions, and determine the result of these expressions.	**1.7.A.1** Apps use mathematical expressions to calculate values that help users make decisions, such as prices, time estimates, or progress toward a goal.
	1.7.A.2 Mathematical expressions can be used to update the state of a program by changing values stored in variables, such as scores, timers, levels, and resource usage.
	1.7.A.3 An integer is a numeric data type that represents positive and negative whole numbers, including zero. Integers are often used for counts or scores in a program.
	EXCLUSION STATEMENT Other numeric data types are outside the scope of the AP Computer Science Principles course and exam.
	1.7.A.4 Arithmetic operators appear in most programming languages, but the notation used for those operations is determined by the language. Common arithmetic operators include addition `+`, subtraction `-`, multiplication `*`, division `/`, and modulo `%`.
	NOTE The exam reference sheet includes these arithmetic operators.
	1.7.A.5 The modulo operation `%` evaluates to the remainder when one integer is divided by another: `a % b` is the remainder when `a` is divided by `b`. Modulo can be used to identify patterns in numbers, such as whether a number is even or odd, and to help format values in user-friendly ways, such as converting total minutes into hours and minutes.
	EXCLUSION STATEMENT The use of values less than `0` for `a` and the use of values less than or equal to `0` for `b` is outside the scope of the AP Computer Science Principles course and exam.
	1.7.A.6 Expressions enclosed in parentheses are evaluated first. Then, multiplication `*`, division `/`, and modulo `%` operations are evaluated from left to right. Then, addition `+` and subtraction `-` operations are evaluated from left to right.
	1.7.A.7 The result of a mathematical expression can be determined by evaluating it step by step using the order of operations and operators.

© 2026 College Board

LEARNING OBJECTIVE

1.7.A
Develop code that uses mathematical expressions, and determine the result of these expressions.

ESSENTIAL KNOWLEDGE

1.7.A.8
Mathematical expressions can include variables. To determine the result, substitute the current values of variables and then evaluate the expression using the order of operations.

LEARNING OBJECTIVE

1.7.B
Develop code to generate random numbers, and determine the possible values that could be produced as a result.

ESSENTIAL KNOWLEDGE

1.7.B.1
Apps use random number generation to create variety and unpredictability, such as in games, simulations, or randomly generated prompts.

1.7.B.2
Random values can support variation in app behavior, such as shuffling a playlist, selecting a random item, or assigning outcomes like rolling a die.

1.7.B.3
Programmers can determine the set of possible values produced by a random command by identifying the endpoints and whether they are included. When the following code runs, a random integer from `a` to `b`, including `a` and `b`, is generated and returned. Each result is equally likely to occur. The possible results of `random(a, b)` are all integers from `a` through `b`.

```
random(a, b)
```

NOTE
The exam reference sheet provides `random(a, b)`, which generates and returns a random integer from `a` to `b`, including `a` and `b`. Each result is equally likely to occur. The possible results of `random(a, b)` are all integers from `a` through `b`.

1.7.B.4
When a program uses random number generation, the same code can produce different results each time it runs.

© 2026 College Board

THIS PAGE HAS BEEN INTENTIONALLY LEFT BLANK

AP COMPUTER SCIENCE PRINCIPLES

UNIT 2

Building Solutions Using Selection and Iteration

TOPIC 2.1

Program Design and Development

Required Course Content

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
2.1.A Develop a program using an iterative or incremental process.	**2.1.A.1** Programmers use a development process to plan and build apps and programs that work well for the people who will use them. A typical process includes understanding the problem and the users, designing a solution, building a first version or prototype, testing it, and then reflecting on what worked and what needs improvement. These phases may occur in different orders and may be repeated. **2.1.A.2** An iterative development process involves revising a program based on feedback, testing, or reflection throughout development. This often requires returning to earlier phases to improve the design or behavior. **2.1.A.3** An incremental development process breaks a problem into smaller features and builds a program in steps. Each part is checked to ensure it works before it is combined into the whole. **2.1.A.4** Development processes often include creating and updating design documentation to help track decisions, guide what to build next, and support testing, debugging, maintenance, and improvement. **2.1.A.5** When developing programs with others, teams can use a defined workflow by assigning roles, sharing work, and keeping track of versions so changes can be reviewed and combined responsibly. **2.1.A.6** Programmers can gather evidence from users during development by observing where users get stuck or confused. This evidence can guide revisions that improve the usability of apps or programs. **2.1.A.7** Many apps and programs go through revisions and updates to stay useful for users. Maintaining a program involves making changes to a working version, such as fixing bugs, improving usability, or adapting the program as user needs or conditions change.

© 2026 College Board

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
2.1.B Design an app or program and its user interface based on user research and program requirements.	**2.1.B.1** Programmers often begin by identifying intended users and their goals rather than starting with a list of features. Focusing on users and goals helps teams prioritize features and design an app or program that meets real needs. **2.1.B.2** User research means learning from the people who will use an app or program. Programmers may gather information from intended users to understand what they need, what limits or challenges they have, and what matters most to them. **2.1.B.3** Program requirements describe what a program must do and how it should behave. Requirements may be based on user research and may include user interactions a program must support, information a user interface should display, and feedback the app or program should provide. **2.1.B.4** Programmers can use user stories to connect requirements to real tasks and to guide interface design and testing. User stories are hypothetical descriptions of who a user is, what they want to do, and why it is important. Teams may also represent user needs with use cases or storyboards, which show how someone would use the app or program step by step. **2.1.B.5** User research can be performed in multiple ways, such as surveys, interviews, direct observations, and user testing. **2.1.B.6** The design phase may include activities such as brainstorming and storyboarding, decomposing a problem into subproblems, creating diagrams or sketches of a user interface layout, and developing a testing strategy based on program requirements and user needs.
2.1.C Describe the purpose and behavior of a code segment or program by writing documentation.	**2.1.C.1** Program documentation is a written description of the purpose and behavior of a code segment or program, including how it is intended to be used, how it accomplishes its task, and how it was developed. **2.1.C.2** Comments are a form of documentation included in code for humans to read. Comments do not affect how a program runs. **2.1.C.3** Not all programming environments support comments within the code. Documentation may also be written in other formats, such as design notes, a test plan, or a change log. **2.1.C.4** Documentation helps programmers understand code that has already been written. Clear descriptions of purpose and behavior make it easier for a programmer to understand and update code without having to figure out how it works. **2.1.C.5** Useful documentation may describe important details, such as a program's overall purpose, key variables and algorithms, expected inputs, and expected outputs. Comments can clarify the purpose or behavior of specific abstractions or sections of code. Clear and concise documentation helps others understand, use, and update a program correctly.

© 2026 College Board

LEARNING OBJECTIVE

2.1.C
Describe the purpose and behavior of a code segment or program by writing documentation.

ESSENTIAL KNOWLEDGE

2.1.C.6
Programmers should document a program throughout development to improve clarity, support collaboration, and record decisions and resources used, such as code written by others and AI tool use.

2.1.D
Determine when it is appropriate to reuse, cite, and share digital content.

2.1.D.1
Programmers often reuse or build on existing code, images, sounds, and other digital content because it can save time, improve reliability, and help them meet user needs without reinventing common solutions. Because programmers often build with existing online tools and code, they should acknowledge code that is reused, adapted, or co-developed, including code from other people, online sources, or AI tools.

2.1.D.2
Digital works are considered the intellectual property of their copyright owner, who is often the work's creator but may be another person or organization. This means the copyright owner has legal control over the original work and can decide whether others may use, share, modify, or distribute it. To respect intellectual property, people should follow a work's license or obtain permission before reusing material and provide attribution when required.

2.1.D.3
Digital content can be copied, edited, and shared quickly, which can make it harder to determine who owns it and what permissions are required. Using another person's work without permission or proper credit may have legal consequences.

2.1.D.4
Despite how easy it is to share content online without crediting the creator or getting permission, presenting someone else's material as your own is plagiarism.

2.1.D.5
Knowing how to reuse and cite digital content correctly helps programmers build projects more efficiently while avoiding plagiarism and misuse. Following reuse and attribution requirements helps others understand what was reused, where it came from, and what permissions apply, which supports transparency and responsible sharing.

2.1.D.6
Some licenses and tools make reuse easier by clearly stating permissions. Open source and Creative Commons licenses provide standardized terms that tell others how they may use, modify, and redistribute software or creative works.

2.1.D.7
License terms vary in what they allow and require. Some licenses require attribution, while others allow unrestricted reuse. Creative Commons Zero (CC0) allows creators to waive copyright restrictions to the extent possible so others can reuse the work.

2.1.D.8
To decide whether reuse or sharing is appropriate, people should identify the source and its license or permissions, follow any requirements, and avoid using material in ways the license does not allow.

© 2026 College Board

TOPIC 2.2
Boolean Expressions

Required Course Content

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
2.2.A Identify the use of Boolean expressions in program code to control a program's behavior.	**2.2.A.1** Apps often check conditions that have only two outcomes, such as whether a user is logged in, a checkbox is selected, or a password is correct. **2.2.A.2** Programs make decisions by evaluating conditions that result in a Boolean value. A Boolean value has two possible values: `true` or `false`. **2.2.A.3** Apps can use Boolean expressions to decide what happens next, such as showing a message, allowing access, or changing what a user sees. A Boolean expression is an expression that evaluates to either `true` or `false`. **2.2.A.4** Many Boolean expressions use comparisons, such as checking whether a score is high enough or whether an input meets a requirement like password length.
2.2.B Develop code that uses relational operators, and determine the result of relational expressions.	**2.2.B.1** Apps and programs often compare values to enforce rules, personalize experiences, and prevent mistakes. Relational operators allow programs to evaluate questions such as whether a value is too low or whether one value matches another. **2.2.B.2** Relational operators compare two values and produce a Boolean result. ▪ The expression `a == b` evaluates to `true` when the values of `a` and `b` are equal and evaluates to `false` otherwise. ▪ The expression `a > b` evaluates to `true` when the value of `a` is greater than the value of `b` and evaluates to `false` otherwise. ▪ The expression `a < b` evaluates to `true` when the value of `a` is less than the value of `b` and evaluates to `false` otherwise. ▪ The expression `a >= b` evaluates to `true` when the value of `a` is greater than or equal to the value of `b` and evaluates to `false` otherwise. ▪ The expression `a <= b` evaluates to `true` when the value of `a` is less than or equal to the value of `b` and evaluates to `false` otherwise. **NOTE** The exam reference sheet includes these relational operators. **2.2.B.3** The result of a relational expression can be determined by substituting the current values, including variable values, and evaluating whether the comparison is `true` or `false`.

© 2026 College Board

LEARNING OBJECTIVE

2.2.C
Develop code that uses logical operators, and determine the result of logical expressions.

ESSENTIAL KNOWLEDGE

2.2.C.1
Apps and programs often combine multiple conditions to make decisions that feel predictable and personalized. Logical operators allow programs to answer questions such as whether all conditions are true, whether at least one condition is true, or whether a condition is not true. Logical operators can also support user-centered rules, such as allowing an action only when multiple conditions are met, like when all required fields are complete and the user agrees to terms.

2.2.C.2
Logical operators combine or modify Boolean values and produce a Boolean result.

- The expression `condition1 and condition2` evaluates to `true` when both conditions are true and evaluates to `false` otherwise.
- The expression `condition1 or condition2` evaluates to `true` when at least one condition is true and evaluates to `false` when both conditions are false.
- The expression `not condition` evaluates to `true` when the condition is false and evaluates to `false` when the condition is true.

NOTE
The exam reference sheet includes these logical operators.

2.2.C.3
The result of a logical expression can be determined by evaluating each condition and then applying `not`, `and`, and `or` to determine whether the overall expression is `true` or `false`.

2.2.C.4
Logical expressions can include multiple conditions. Parentheses can be used to group conditions and make the intended order of evaluation clear when combining `not`, `and`, and `or`.

© 2026 College Board

TOPIC 2.3
Selection

Required Course Content

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
2.3.A Represent an algorithm that includes selection using written steps or a flowchart.	**2.3.A.1** Some tasks require making a choice based on a condition. Selection allows an algorithm to choose different steps depending on whether a condition is `true` or `false`. **2.3.A.2** Algorithms that use selection can be represented in ways people can read and follow, such as written steps or flowcharts. **2.3.A.3** Selection changes the path an algorithm follows. Different conditions can lead to different results because the algorithm may perform different steps based on the choice made.
2.3.B Develop code that uses conditional statements, and determine the result of these statements.	**2.3.B.1** Selection allows apps to respond differently depending on the information they receive. As a result, apps can handle user choices and provide helpful output instead of doing the same thing every time. **2.3.B.2** Programmers implement selection in code using conditional statements. Programmers use conditional statements to check a condition using a Boolean expression and run different code depending on the result. **2.3.B.3** An if-statement is a type of conditional statement used when a program needs to check a condition and take an action only when that condition is true. When the following code runs, the statements in `block of statements` are executed if `condition` evaluates to `true`. Otherwise, `block of statements` is skipped and the program continues. `if (condition)` `{` `  <block of statements>` `}` **NOTE** The exam reference sheet provides this if statement.

© 2026 College Board

LEARNING OBJECTIVE

2.3.B
Develop code that uses conditional statements, and determine the result of these statements.

ESSENTIAL KNOWLEDGE

2.3.B.4
An if-else statement is a type of conditional statement used when a program needs to choose between two different actions based on whether a condition is true or false. When the following code runs, the statements in the `first block of statements` execute if the `condition` evaluates to `true`. Otherwise, the statements in the `second block of statements` execute.

```
if (condition)
{
   <first block of statements>
}
else
{
   <second block of statements>
}
```

NOTE
The exam reference sheet provides this if-else statement.

2.3.B.5
Choosing between an if and an if-else statement depends on whether the program should act only when a condition is true or should take one of two actions.

2.3.B.6
The result of conditional statements can be determined by evaluating the conditions and tracing which statements execute, including any changes to variable values or output.

2.3.B.7
When multiple if-statements are written sequentially, each one is evaluated independently, so more than one block may execute if more than one condition is true.

2.3.C
Develop code that uses nested conditional statements, and determine the result of these statements.

2.3.C.1
Apps often make layered decisions by checking a basic condition first and then checking a more specific condition. For example, an app might first check whether a user is logged in and then check whether the user has permission to access a feature. Nested conditionals allow programs to handle these multistep checks.

2.3.C.2
A nested conditional is a conditional statement inside the block of another conditional statement. The inner conditional is evaluated only when the outer condition evaluates to true and execution reaches the inner conditional.

2.3.C.3
Nested conditional statements can be traced by evaluating each conditional statement's Boolean expression in order, following the branch of code that runs, and keeping track of any updates to variable values or output that is produced.

© 2026 College Board

TOPIC 2.4
Iteration

Required Course Content

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
2.4.A Represent an algorithm that includes iteration using written steps or a flowchart.	**2.4.A.1** Some tasks require repeating steps multiple times, such as continuing until a goal is reached or repeating for each item in a collection of items. Iteration allows an algorithm to repeat a sequence of steps. **2.4.A.2** Algorithms that use iteration can be represented in ways people can read and follow, such as written steps or flowcharts. **2.4.A.3** When representing iteration, clearly state the repeated steps and the condition that determines whether repetition continues or ends. **2.4.A.4** Iteration allows an algorithm to solve problems, such as counting items or searching until a match is found, by repeating steps. The outcome depends on what happens during each repetition and on the stopping condition.
2.4.B Develop code that uses iteration, and determine the result of iteration statements.	**2.4.B.1** Iteration allows programs to perform repeated actions without writing the same statements multiple times. Programs can repeat steps until a condition is met, continue updating while something is happening, or repeat actions across many items so the user does not have to do each step manually. **2.4.B.2** Programs use loops to perform iteration. Loop statements change the flow of a program by repeating a block of statements. A loop may run zero or more times depending on its condition or count. **2.4.B.3** A while loop is used when a program needs to repeat a set of statements as long as a condition remains true. When the following code runs, the statements in `block of statements` repeat while `condition` evaluates to `true`. When `condition` evaluates to `false`, the loop stops and the program continues. `while (condition)` `{` `  <block of statements>` `}` **NOTE** The exam reference sheet provides this while loop.

© 2026 College Board

LEARNING OBJECTIVE

2.4.B

Develop code that uses iteration, and determine the result of iteration statements.

ESSENTIAL KNOWLEDGE

2.4.B.4

In a while loop, the condition is checked before the loop body runs. If the condition evaluates to false initially, the loop body is not executed.

2.4.B.5

In a while loop, an infinite loop occurs when the condition never becomes false, so the loop never stops. Infinite loops can be avoided by updating values that affect the condition or adding a stopping condition.

2.4.B.6

A for loop is used when a program needs to repeat a set of statements a specific number of times. When the following code runs, the variable `j` is assigned the value of each integer in the range `0` to `n - 1` sequentially. The code in `block of statements` is executed `n` times—once for each assignment of `j`.

```
for j in range(n)
{
   <block of statements>
}
```

NOTE

The exam reference sheet provides this for loop.

2.4.B.7

A for loop is often used when the number of repetitions is known before the loop starts running, such as repeating a block a specific number of times or iterating through a specific range. A while loop is often used when the number of repetitions cannot be determined before the loop starts running and the loop should continue as long as a condition remains true.

2.4.B.8

The result of iteration statements can be determined by tracing the loop step by step, keeping track of how many times the loop runs and how variable values and output change each time.

© 2026 College Board

TOPIC 2.5
Algorithm Development

Required Course Content

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
2.5.A Develop algorithms by building new solutions or by reusing and combining existing solutions.	**2.5.A.1** Algorithms use sequencing, selection, and iteration to organize steps, choose actions based on conditions, and repeat actions. **2.5.A.2** Programmers can develop an algorithm from scratch, or they can reuse, combine, or adapt existing algorithms to solve a problem. **2.5.A.3** Programmers often build new algorithms by reusing parts of known algorithms and adjusting them to fit a new problem, such as handling different inputs or producing a different output. They test the updated algorithm to confirm it meets its requirements. **2.5.A.4** Knowing common algorithms and patterns can make problem solving more efficient. Many algorithms reuse common patterns, such as: ▪ finding the maximum or minimum value in a group of numbers ▪ computing the sum or average of values ▪ checking whether one integer is evenly divisible by another **2.5.A.5** Using well-understood algorithms as building blocks can reduce development time and support testing, debugging, and maintenance. **2.5.A.6** Programmers select, combine, and adapt algorithms to solve problems and meet user goals, especially as feedback or conditions change. For example, an app might combine algorithms to recommend content, plan a driving route, or calculate a total price with discounts and taxes.
2.5.B Compare multiple algorithms to determine if they produce the same side effect or output.	**2.5.B.1** There is often more than one way to solve a task. Different algorithms can use different steps, yet still produce the same output for the same inputs. By comparing algorithms, programmers can decide which approach is clearer, easier to maintain, or better suited to the situation. **2.5.B.2** When comparing algorithms, programmers consider both the output and any side effects. **2.5.B.3** A side effect occurs when an algorithm changes the program's state while it runs. For example, an algorithm may update a variable's value, add or remove an item from stored data, or change what the user sees on the screen.

© 2026 College Board

LEARNING OBJECTIVE

2.5.B

Compare multiple algorithms to determine if they produce the same side effect or output.

ESSENTIAL KNOWLEDGE

2.5.B.4

Side effects can impact the user experience. An unintended side effect, such as an incorrect variable update, an unexpected screen change, or an inconsistent program state, can confuse or frustrate users.

2.5.B.5

Algorithms that look similar may behave differently with certain inputs or may produce different side effects. To determine whether two algorithms are equivalent, programmers compare their outputs and side effects using the same inputs, often by tracing the steps and tracking changes to variables and displayed output.

2.5.B.6

Some conditional statements can be rewritten as equivalent Boolean expressions, and some Boolean expressions can be rewritten as equivalent conditional statements, without changing the program's behavior.

2.5.B.7

Loops can often be written in different ways and still be equivalent. A for loop that repeats a fixed number of times can often be rewritten as a while loop that uses a counter variable. When comparing a for and while loop, programmers check that they repeat the same number of times, update variables the same way, and produce the same output.

© 2026 College Board

TOPIC 2.6
Errors

Required Course Content

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
2.6.A Correct errors in programs.	**2.6.A.1** Errors are a normal part of programming. As programmers develop code, they repeatedly encounter errors, identify their cause, and revise the program to correct them.
	2.6.A.2 Programmers often discover errors through user testing or feedback, such as when users receive confusing messages, see inconsistent results, or cannot complete a task. Programmers use observations of what the user did and what the app displayed to clarify what the behavior should be for the user. To fix the error, they try to reproduce it by following the same user actions and inputs, which helps confirm the cause and verify that the correction solves the real user problem.
	2.6.A.3 Knowing common types of errors helps programmers diagnose problems more quickly and choose effective strategies to fix them when they occur.
	2.6.A.4 A logic error occurs when a program runs but produces incorrect or unexpected behavior because the algorithm or reasoning is incorrect.
	2.6.A.5 A syntax error occurs when the code breaks the rules of the programming language, preventing the program from running as written.
	2.6.A.6 A runtime error occurs while a program is executing when it encounters a situation it cannot handle, such as dividing by zero. Runtime errors may cause a program to stop running unless the error is handled. Programming languages define the specific runtime errors they produce.
	2.6.A.7 Programmers use error messages to identify and correct problems because they describe what went wrong and often point to where it occurred. They may include the type of error, such as syntax or runtime, the line or location in the code, and details about the operation or value that caused the problem. Programmers use error messages to decide what to check first, make a targeted change, and then rerun the program to confirm the error is resolved.
	2.6.A.8 Programmers trace a program's execution to determine how variable values change and which statements execute. They may trace manually or use debugging tools that step through code and display current variable values. Tracing helps identify where a program's behavior differs from what was expected and is especially useful for finding logic errors. It can also be used to determine what conditions led to a runtime error.

© 2026 College Board

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
2.6.A Correct errors in programs.	**2.6.A.9** Programmers can add output statements to temporarily display intermediate values or messages while a program runs, such as printing variable values at key points. This helps confirm which parts of the code are executing and whether variables and conditions have the expected values. By comparing the observed output to the expected output, programmers can narrow down where the program's behavior first begins to differ from what was expected and identify the source of an error. **2.6.A.10** AI tools can support programmers in interpreting error messages, explaining what a piece of code is doing, and suggesting possible causes and fixes for unexpected behavior. Programmers can describe the expected vs. actual output and share relevant code so the tool can point out likely logic mistakes or recommend ways of making the code easier to trace. AI suggestions should be verified by running the program, checking results against requirements, and using other debugging methods, because AI-generated explanations or fixes may be incomplete or incorrect.
2.6.B Develop test cases that can be used to check the correctness of an algorithm.	**2.6.B.1** In the development process, testing uses defined inputs to check whether an algorithm or program produces the expected outcomes. Programmers compare actual results to expected results and use what they learn to revise the algorithm or program. A test case consists of a specific input and the expected output or behavior that should result from that input. AI tools can help suggest test cases and predict expected results, but programmers are responsible for validating expected outcomes and using test results to confirm correctness. **2.6.B.2** Programmers often derive test cases from user stories and user testing by translating real user tasks into inputs and expected outcomes, including what the user should see at each step. This helps ensure the algorithm is correct in ways that matter to users. **2.6.B.3** An effective set of test cases includes inputs that produce different expected outputs or behaviors, including typical, boundary, and invalid cases. Boundary test cases use values at or near important limits, such as the minimum and maximum allowed inputs or decision thresholds in conditional expressions. Invalid test cases may use values just beyond the allowed range to check how the program handles likely user mistakes. **2.6.B.4** Some algorithms appear correct for common inputs but fail for other inputs, so programmers test multiple cases that cover different paths through the algorithm, such as different branches of a conditional or different numbers of loop repetitions. AI tools can help identify distinct execution paths and suggest test cases for each, but programmers should confirm that the suggested tests actually reach those paths. **2.6.B.5** Testing reflects real users: people mistype, submit blanks, use unexpected values, click repeatedly, or do steps out of order. Including these inputs as test cases helps ensure the app or program responds clearly and consistently, including giving useful feedback when input is invalid.

© 2026 College Board

AP COMPUTER SCIENCE PRINCIPLES

UNIT 3

Developing with Abstraction Using Lists and Procedures

TOPIC 3.1
Lists

Required Course Content

LEARNING OBJECTIVE

3.1.A
Develop code that stores multiple related values in a single variable.

ESSENTIAL KNOWLEDGE

3.1.A.1
A data collection is a way to store multiple related values together so a program can treat them as one unit, such as storing multiple names, scores, or messages. Different programming languages support different kinds of collections, such as lists, arrays, or dictionaries.

NOTE
The exam reference sheet uses lists to store multiple values in a single variable.

EXCLUSION STATEMENT
Data collections other than lists are outside the scope of the AP Computer Science Principles course and exam.

3.1.A.2
Data collections help programs support user-facing features that involve many items, such as a list of saved favorites, recent searches, or a playlist. Storing related values in one variable makes it easier to manage, update, and process the information as users interact with an app.

3.1.A.3
A value stored inside a data collection is an element. Programs can work with the entire collection of items as one unit or with individual elements.

3.1.A.4
In a list, elements are stored in an ordered sequence, meaning the position of each element matters (first, second, third, etc.). When the following code runs, a list called `aList` contains the elements `v1`, `v2`, `v3`, and so on.

```
aList = [v1, v2, v3, ...]
```

NOTE
The exam reference sheet provides this list.

3.1.A.5
Lists can start empty, which allows a program to collect information, such as items a user adds, results found during a search, or responses entered in a form. When the following code runs, an empty list is assigned to `aList`.

```
aList = []
```

NOTE
The exam reference sheet provides this way of creating an empty list.

© 2026 College Board

LEARNING OBJECTIVE

3.1.A
Develop code that stores multiple related values in a single variable.

ESSENTIAL KNOWLEDGE

3.1.A.6
Lists can be built over time by adding new elements, often as the user enters information or the program collects it. When the following code runs, `value` is added to the end of `aList` and the length of `aList` is increased by 1.

```
aList.append(value)
```

NOTE
The exam reference sheet provides this append procedure.

3.1.A.7
An index is a number used to reference a specific element by position in a list or in a string. Many programming languages use zero-based indexing, meaning the first element is at index 0, the second at index 1, and so on.

NOTE
The language on the exam reference sheet uses zero-based indexing.

3.1.A.8
Programs can access a specific list element using its index. This allows programs to read or use stored values in a list. When the following code runs, the element in `aList` at index `j` is accessed.

```
aList[j]
```

NOTE
The exam reference sheet provides this way of accessing a list index.

3.1.A.9
Lists support apps that respond to users by remembering and reusing information. For example, a music app can store a user's queued songs in a list. Each time the user taps "Add to Queue," the song title is appended to the list, and the app uses the list to display the upcoming songs in order and play the next track.

3.1.B
Develop code to manipulate a list, and determine the result produced.

3.1.B.1
Programmers sometimes make a copy of a list so they can modify the copy, such as removing or reordering elements, while keeping the original list unchanged. Preserving the original list can make a program easier to understand, test, and debug because the program can still refer back to the original data if needed. When the following code runs, a copy of the list `bList` is assigned to the list `aList`.

```
aList = bList
```

NOTE
The exam reference sheet provides this way of assigning a copy of a list.

© 2026 College Board

LEARNING OBJECTIVE

3.1.B

Develop code to manipulate a list, and determine the result produced.

ESSENTIAL KNOWLEDGE

3.1.B.2

Elements in a list can be changed to different values, which allows a program to update stored information without rebuilding the entire list. When the following code runs, the value of `x` is assigned to the element at index `j` in `aList`. The previous value at that index is overwritten, and the length of the list does not change.

```
aList[j] = x
```

NOTE
The exam reference sheet provides this way of updating list values.

3.1.B.3

The length of a list can change while a program runs as elements are added or removed. Knowing the length of a list helps programmers write code that works correctly as the list changes. When the following code runs, it evaluates to the number of items in `aList`.

```
length(aList)
```

NOTE
The exam reference sheet provides this way of finding the length of a list.

3.1.B.4

Elements can be inserted at particular positions in a list, which allows a program to keep items in a meaningful order, such as priority or ranking, as new information is added. When the following code runs, any elements at indices greater than or equal to `j` are shifted so that their indices are increased by 1 and `value` is placed at index `j`. The length of the list is increased by 1.

```
aList.insert(j, value)
```

NOTE
The exam reference sheet provides this insert procedure.

3.1.B.5

Elements at particular positions can be removed from a list, which allows a program to delete outdated, incorrect, or unwanted information while keeping the remaining elements usable. When the following code runs, the element at index `j` is removed and the elements after it shift one position so that their indices are decreased by 1. The length of the list is decreased by 1.

```
aList.remove(j)
```

NOTE
The exam reference sheet provides this remove procedure.

© 2026 College Board

LEARNING OBJECTIVE

3.1.B
Develop code to manipulate a list, and determine the result produced.

ESSENTIAL KNOWLEDGE

3.1.B.6
To select a random element from a list, a program can generate a random index and use it to access an element. This allows for features like selecting a random song from a playlist, choosing a random prompt in a game, or assigning a random partner from a list of names. In programming languages that use zero-based indexing, if `aList` has `length(aList)` items, the valid indices are `0` through `length(aList) - 1`. When the following code runs, the program generates one of the valid indices with equal likelihood and accesses the element at that index.

```
aList[random(0, length(aList) - 1)]
```

3.1.B.7
Apps often update lists in response to user actions. Adding or removing items supports user goals such as saving favorites, managing a to-do list, updating a roster, or keeping a recent-search history. Programmers choose list operations that match how users expect information to change as they interact with the app.

3.1.B.8
To determine what output a program produces after list operations, programmers trace statements in order and update the list after each step. This includes tracking how element positions change when items are inserted or removed and using the updated list to determine what values are accessed or displayed next.

3.1.B.9
List operations that use an index depend on the index being valid for the current list. When a program accesses, updates, inserts, or removes using an index, programmers must consider the current length of the list and how earlier operations may have shifted element positions. In many programming languages, using an index that is outside the allowed range produces an error and may cause the program to stop.

3.1.C
Develop code that uses iteration statements to traverse a list, and determine the result produced.

3.1.C.1
Traversing a list is often used to accomplish a goal, such as counting elements that meet a condition, finding a value (like a maximum), computing a total or average, or building an output (e.g., as a message that includes all elements that match a rule).

3.1.C.2
Traversals often update an accumulator variable, such as a count or total, each time an element is processed.

3.1.C.3
Iteration statements can be used to traverse a list, which allows a program to process each element in order without writing repeated code for every element. When the following code runs, the variable `item` is assigned the value of each element of `aList` sequentially, in order, from the first element to the last element. The code in `block of statements` is executed once for each element in `aList`.

```
for item in aList
{
   <block of statements>
}
```

NOTE
The exam reference sheet provides this iteration statement.

© 2026 College Board

LEARNING OBJECTIVE

3.1.C
Develop code that uses iteration statements to traverse a list, and determine the result produced.

ESSENTIAL KNOWLEDGE

3.1.C.4
A linear or sequential search checks elements in order until the desired value is found or the list ends. Programs often store the index where a match was found or store a Boolean value to indicate whether a match was found.

3.1.C.5
Traversing a list can be a complete traversal, where every element is accessed, or a partial traversal, where only some elements are examined. Some partial traversals stop early when a goal is reached, such as finding the first match in a search, which may require fewer steps than traversing the entire list.

EXCLUSION STATEMENT
Traversing multiple lists at the same time using the same index for both (parallel traversals) is outside the scope of the AP Computer Science Principles course and exam.

3.1.C.6
Traversing a list supports user-facing features like calculating a total in a shopping cart, counting votes in a poll, or checking whether a username is already taken. Programmers choose complete vs. partial traversal based on whether a user needs to check every element or find a particular match in the list.

3.1.C.7
To determine the result of a traversal, programmers trace the loop and track how variables update each time an element is processed.

3.1.D
Explain how the use of data abstraction manages complexity in program code.

3.1.D.1
Data abstraction manages complexity by allowing programmers to use data based on its name and purpose without needing to focus on how the data is represented. This separation helps programmers reason about what the data represents and what operations can be done with it.

3.1.D.2
A list is a type of data abstraction because it allows many related values to be treated as a single variable. Instead of writing code for many individual variables, programmers can use the list's name to perform the same kinds of operations on its elements, such as accessing, updating, inserting, removing, or traversing.

3.1.D.3
Using data abstractions such as lists can make programs easier to develop, read, test, and maintain because the same algorithm can often work even when the amount of data changes. When elements are added or removed, the code structure often stays the same.

3.1.D.4
Data abstraction makes it easier to adapt a program as user needs change (e.g., handling more saved favorites, more songs on a playlist, or more messages) without redesigning the entire program.

3.1.D.5
Data abstraction reduces complexity by reducing duplication in code. For example, using a single list and a loop can process any number of elements, instead of requiring many separate variables and repeated statements. This means fewer places to update code during development, which lowers the chance of errors.

© 2026 College Board

TOPIC 3.2
Procedures

Required Course Content

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
3.2.A Develop code to define procedures with and without parameters.	**3.2.A.1** A procedure is a named group of programming instructions that performs a task. Using procedures helps programmers organize code, avoid unnecessary code, and make programs easier to read, test, and update. Procedures may also be called functions or methods depending on the programming language.
	3.2.A.2 If a procedure does not need any input information to complete its intended task, it can be defined with its name followed by parentheses and a block of statements. Each time the procedure is called, the statements in the procedure execute, and then the program continues with the next statement after the call.
	3.2.A.3 Procedures can use parameters to receive input. A parameter is a variable name listed in the procedure definition. An argument is the actual value provided when the procedure is called.
	3.2.A.4 A procedure may not need parameters when it always performs the same steps with no changing input (e.g., displaying an instructions screen or resetting a game to a default starting state). Procedures need parameters when they need input information to complete the intended task. If a procedure should work for many different inputs (e.g., greeting different users or calculating totals for different numbers), parameters make it flexible and reusable because the program can pass different arguments each time it is called. Using parameters when input varies helps avoid duplicating code and keeps the program easier to update.
	3.2.A.5 Procedures with parameters make it easier to adapt an app's behavior or output to different users and situations without rewriting code. A single procedure with parameters can generate many clear, personalized messages for different users by using different argument values each time it is called.
	3.2.A.6 A procedure is defined with a name and a list of zero or more parameters. When the following code runs, a procedure named `pName` is defined with zero or more parameters `(p1, p2, ...)`. When the procedure is called, the statements in `<block of statements>` execute.

```
procedure pName(p1, p2, ...)
{
   <block of statements>
}
```

NOTE
The exam reference sheet provides this procedure definition.

© 2026 College Board

LEARNING OBJECTIVE

3.2.A
Develop code to define procedures with and without parameters.

ESSENTIAL KNOWLEDGE

3.2.A.7
Some procedures use a return statement to send a result back to the part of the program that called the procedure, which allows the returned value to be stored in a variable, used in an expression, or displayed as output. The return statement ends the procedure immediately, even if more statements appear later in the procedure. When the following code runs in the body of a procedure, the value of `expression` is returned to the place in the program where the procedure was called.

```
return expression
```

NOTE
The exam reference sheet provides this return statement.

3.2.A.8
Some procedures perform an action, such as displaying output or updating variables, and do not return a value. Other procedures compute a value and use a return statement so the result can be used elsewhere in the program.

3.2.A.9
Defining a procedure creates the named steps, but the code inside runs only when the procedure is called.

3.2.B
Develop code for procedure calls, and determine the result of procedure calls.

3.2.B.1
Calling a procedure runs a named sequence of steps, which helps programs reuse code and keep the overall program easier to read and follow.

3.2.B.2
A procedure is called by using its name in a statement so the program runs the steps stored in that procedure. When the call happens, the program pauses the current sequence of statements, executes the procedure's instructions, and then resumes at the next statement after the call.

3.2.B.3
In a procedure call, the values inside the parentheses are arguments. When the call runs, each argument value is assigned to the matching parameter in the procedure definition. This allows the same procedure to run with different input values. When the following code runs, `a1` is assigned to the first parameter and `a2` is assigned to the second parameter before the procedure's statements execute.

```
pName(a1, a2)
```

NOTE
The exam reference sheet provides this procedure call.

3.2.B.4
A procedure can be called with no arguments (empty parentheses) when it has no parameters. The number of arguments in a call must match the number of parameters in the procedure definition.

3.2.B.5
To determine the result of a procedure call, programmers trace the call by matching arguments to parameters, stepping through the procedure's statements in order, tracking any variable updates and outputs, and then returning to where the procedure was called to continue program execution.

© 2026 College Board

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
3.2.B Develop code for procedure calls, and determine the result of procedure calls.	**3.2.B.6** If a procedure returns a value, the procedure call evaluates to that value. The returned value can be stored in a variable, used in an expression, or displayed as output. If a procedure does not return a value, the call is used to perform actions such as updating variable values or producing output.
3.2.C Explain how the use of procedural abstraction manages complexity in a program.	**3.2.C.1** Procedural abstraction manages complexity by grouping related code into named procedures that can be called from other parts of a program. **3.2.C.2** Procedural abstraction allows programmers to use a procedure by knowing what it does without needing to know how it works internally. Programmers often use built-in procedures, such as print or random, by relying on the procedure's name, purpose, and expected behavior while the implementation details stay hidden. **3.2.C.3** Procedural abstraction helps manage complexity by breaking a large problem into smaller subproblems and creating procedures to solve each part. **3.2.C.4** The subdivision of a program into smaller, independent procedures or components is called modularity. Modularity helps programmers focus on one part at a time and makes programs easier to build, test, and revise. **3.2.C.5** Procedural abstraction supports code reuse by extracting shared behavior into a single procedure instead of duplicating code in multiple places. **3.2.C.6** Using parameters generalizes a procedure so it can be reused with different inputs, allowing one procedure to handle many situations without rewriting code. **3.2.C.7** Procedural abstraction allows programmers to change a procedure's internal steps (e.g., to improve efficiency or fix a bug) without changing the rest of the program. This makes larger programs easier to maintain because updates can be made in one place. **3.2.C.8** Using procedural abstraction improves readability because the rest of the program can call procedures with descriptive names, so the overall steps of the program are easier to understand at a glance.

© 2026 College Board

TOPIC 3.3

Using Existing Code and Libraries

Required Course Content

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
3.3.A Develop code using appropriate libraries or existing code segments to create a program.	**3.3.A.1** A software library is a collection of prewritten procedures that programmers can use to add functionality to a program without writing everything from scratch (e.g., drawing graphics, working with data, or generating random values). **3.3.A.2** Existing code segments can come from internal sources (e.g., one's own prior projects or team's code) or external sources (e.g., libraries, sample code, or online repositories). Reusing code can save time and reduce errors when the code is well tested and well understood. **3.3.A.3** Libraries and reusable code simplify complex tasks by providing procedures that handle common problems, allowing programmers to focus on the parts of the program that are unique to their users and goals. **3.3.A.4** An application programming interface (API) is a set of rules that specifies how one piece of software can interact with another. An API describes what inputs are accepted, what output is returned (if any), and what side effects may occur. **3.3.A.5** Library documentation describes the API and is necessary for correct use. Programmers use documentation to learn required inputs, possible outputs, examples, limitations, and common errors. **3.3.A.6** Selecting appropriate libraries or code segments requires checking whether they fit the program's purpose and user needs, are compatible with the programming environment, come from a trustworthy source, and are licensed for reuse. **3.3.A.7** When using existing code, programmers test how reused procedures work within their own program. They compare actual results to expected results using both typical inputs and boundary cases to confirm the integration works as intended.

© 2026 College Board

AP COMPUTER SCIENCE PRINCIPLES

UNIT 4

Making Data-Driven Decisions Using the Internet and AI

TOPIC 4.1
The Internet

Required Course Content

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
4.1.A Explain how the Internet exchanges data and scales with demand.	**4.1.A.1** The Internet is a global network of networks that communicates using standardized protocols. **4.1.A.2** A protocol is an agreed-upon set of rules for communication, such as how data is formatted, sent, received, or acknowledged, so different devices and networks can work together reliably. **4.1.A.3** Because Internet protocols are open and nonproprietary (not owned or controlled by a single company), devices and networks from different companies can communicate and exchange data. This makes widespread connectivity possible and allows new devices and networks to join the Internet without relying on proprietary systems (owned or controlled by one company). **4.1.A.4** Devices access the Internet by connecting to a local network (like home or school Wi-Fi) or directly to a cellular provider's network. In both cases, the connection is carried by an Internet service provider (ISP) to the broader Internet. **4.1.A.5** The Internet was designed to be scalable, meaning it can expand over time to meet demand. Scalability is a system's ability to grow or change to meet new demands, such as more users, more devices, or higher traffic, while continuing to function efficiently and reliably.
4.1.B Explain how computing devices communicate in a network and explain how network conditions affect communication and user experience.	**4.1.B.1** A networked computing device, such as a computer, tablet, server, router, or smart sensor, is a physical artifact that can run programs and can send and/or receive data over a network. **4.1.B.2** A computing system is a group of computing devices and programs that work together to accomplish a shared purpose. A messaging service may rely on an app on a phone, a Wi-Fi router, and cloud servers working together to send and receive messages. **4.1.B.3** A computer network is a group of interconnected computing devices that communicate by sending and receiving data. It is a type of computing system because multiple devices coordinate to exchange data and provide services. **4.1.B.4** Data sent across a network typically travels through multiple intermediate devices, such as switches and routers, before reaching its destination. A path from sender to receiver is the sequence of directly connected devices that the data follows from start to finish.

© 2026 College Board

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
4.1.B Explain how computing devices communicate in a network and explain how network conditions affect communication and user experience.	**4.1.B.5** Bandwidth commonly refers to the maximum rate at which data can be transmitted over a connection, usually measured in bits per second. More available bandwidth generally allows higher data rates. Higher data-rate bandwidth can support faster downloads and higher-quality streaming. **4.1.B.6** Latency is the time it takes for data to travel from one device to another across a network. Latency is typically measured in milliseconds. High latency creates noticeable delays, like slow page loads, lag in video calls, or delayed responses in online games, even if bandwidth is high. **4.1.B.7** Network conditions shape user experience. When bandwidth is limited, latency is high, or networks are congested, apps may lag, buffer, or fail to load. Developers design for these situations by reducing file sizes, loading content in chunks, saving progress, and showing clear status messages like “Loading…” or “Try again.” **4.1.B.8** Understanding bandwidth, latency, and routing helps programmers design apps that stay usable under common situations such as crowded public Wi-Fi, weak cellular signals, or unreliable connections, so users can complete tasks reliably or receive clear feedback when connectivity prevents it.
4.1.C Explain how network failures can occur and explain how fault-tolerant networks use packets, dynamic routing, and redundancy to keep communication working.	**4.1.C.1** Data sent over the Internet is broken into smaller pieces called packets. Sending data as packets lets routers send each packet along available paths, and the receiving device reassembles the data at the destination, which improves efficiency. **4.1.C.2** A packet contains a chunk of data (part of a message) and metadata (information such as where it came from and where it is going). This metadata helps networks route packets to the correct destination and may help the receiving device reassemble the original data. **4.1.C.3** Packets may arrive out of order, be delayed, or fail to arrive. Certain Internet protocols help handle these situations by organizing packets correctly and requesting missing data when needed. **4.1.C.4** Routing is the process of selecting a path for data to travel from a sender to a receiver. Routing on the Internet is usually dynamic, meaning data can take different paths when network conditions change, especially when links or devices fail or become unavailable. **4.1.C.5** The Internet is designed to be fault tolerant, meaning it can continue to function even when some devices or connections fail. This matters because failures happen regularly, and fault tolerance helps users continue to access information. **4.1.C.6** Redundancy is the inclusion of extra components or alternate paths that can reduce the impact of failure when part of a system stops working. Redundancy often requires additional resources, but it increases reliability. **4.1.C.7** Because data is sent as packets and routing is dynamic, packets can be redirected when a path fails. Redundant routes can allow the network to continue delivering data even if some devices or connections stop working.

© 2026 College Board

LEARNING OBJECTIVE

4.1.C

Explain how network failures can occur and explain how fault-tolerant networks use packets, dynamic routing, and redundancy to keep communication working.

ESSENTIAL KNOWLEDGE

4.1.C.8

To predict the impact of a failure, identify the intermediate devices and connections used along a path. If removing one device or connection leaves no remaining path between the sender and receiver, that failure can stop communication. Redundancy reduces this risk by providing alternate paths.

© 2026 College Board

TOPIC 4.2
Effects of Technology

Required Course Content

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
4.2.A Explain how effects of technology can be both beneficial and harmful.	**4.2.A.1** Technology is created and used by people, and human choices shape how it affects others. **4.2.A.2** When a new technology is adopted, it often changes how people complete tasks by making some actions easier, faster, or more convenient, which sometimes changes habits and expectations. **4.2.A.3** A new technology can have both beneficial and harmful effects depending on the user, the situation, and what the person values. The same person may view an effect as beneficial in one context and harmful in another. **4.2.A.4** Advances in technology can increase creativity and capabilities in many fields (e.g., medicine, engineering, communication, and the arts). They can also create risks when tools are misused to cause harm, when people overly rely on outputs without checking accuracy, when misinformation spreads faster or seems more believable, or when unequal access limits opportunities and widens gaps in education, jobs, or services.
4.2.B Explain how technology can have an impact beyond its intended purpose.	**4.2.B.1** Digital tools are often used in ways their creators did not intend, especially when users adapt features for new goals or when the tool spreads to new contexts. **4.2.B.2** Technologies can have impacts beyond their intended use because they can change people's behavior, what information is shared, and who has access to opportunities or influence. **4.2.B.3** As technologies spread, they can also create unequal impacts. The digital divide refers to differences in access to computing devices and the Internet based on socioeconomic, geographic, or demographic factors. **4.2.B.4** The digital divide affects both groups and individuals by shaping who can participate fully in digital spaces and benefit from technology. It can be affected by the actions of individuals, organizations, and governments. **4.2.B.5** Because impacts can be uneven, responsible developers consider who may be unable to access or use the technology and who may be negatively affected by its design or use. They design features and supports to reduce barriers to access, usability, and participation.

© 2026 College Board

LEARNING OBJECTIVE

4.2.B

Explain how technology can have an impact beyond its intended purpose.

ESSENTIAL KNOWLEDGE

4.2.B.6

Developers cannot predict every future use of a tool or system, especially once it is widely shared. However, they can anticipate likely risks, test with real users, and update the system over time as new impacts appear.

4.2.B.7

Technology tools can have effects beyond their main purpose. When many people use a tool, its features can reshape habits, relationships, and opportunities in ways that were not planned or predicted. Recognizing these broader effects helps people make more informed choices about how they use a tool and helps developers revise designs more responsibly.

© 2026 College Board

TOPIC 4.3

Privacy and Security

Required Course Content

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
4.3.A Explain how personal data is collected, tracked, and aggregated in digital systems and explain how these practices can create privacy risks.	**4.3.A.1** Websites, apps, and devices can collect data about user activity, including searches, clicks, page views, purchases, and device information. Some systems also collect or infer location and other details based on signals from the device or network.
	4.3.A.2 Collected activity data can be used to personalize experiences, such as search results or recommendations, and may also be used for advertising or analytics, depending on user settings, company practices, and applicable laws.
	4.3.A.3 Cookies are small pieces of data that a website stores in a user's browser or device to remember information such as login status, preferences, or items in a shopping cart.
	4.3.A.4 Cookies and other identifiers can be used to recognize a user across visits and in some cases across different sites or apps, enabling measurement, personalization, and targeted advertising.
	4.3.A.5 Geolocation data estimates or identifies a device's physical location. It can enable useful services, like navigation or nearby results, but can also reveal sensitive patterns, such as where someone lives, goes to school, works, or spends time.
	4.3.A.6 Data from different sources, such as geolocation, browsing history, app activity, purchases, and social media, can be aggregated to create a detailed profile of an individual. Even if a single data point seems harmless, combining many points can reveal identity, habits, and relationships.
	4.3.A.7 Some data is personally identifiable information (PII). Some PII is directly identifying, like a name or phone number, and some becomes identifying when combined with other data. Examples include full name, phone number, email address, Social Security number, government ID numbers, precise location, and biometrics.
	4.3.A.8 Privacy risks increase when data is stored, shared, or reused beyond the user's expectations. Once information is posted or shared digitally, it can be copied, forwarded, or saved by others, making it difficult to fully control or remove.
4.3.B Explain how digital systems and information can be compromised.	**4.3.B.1** A data breach occurs when sensitive data is accessed, disclosed, or stolen by an unauthorized party, often because data was stored insecurely, shared improperly, or protected by weak security controls.

© 2026 College Board

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
4.3.B Explain how digital systems and information can be compromised.	**4.3.B.2** When PII is exposed in a data breach, individuals may be at risk of harms such as identity theft, account takeover, or fraud, especially if the data can be used to impersonate them or access their accounts. **4.3.B.3** Reusing the same password across multiple accounts increases the risk of account compromise because if one account's password is exposed, an adversary can try the same username and password on other services to gain unauthorized access. **4.3.B.4** Social engineering is a strategy adversaries use to compromise systems by targeting people rather than exploiting a technical weakness. An adversary influences a user into taking an action that compromises security, often by impersonating a trusted source or creating urgency, fear, or curiosity. **4.3.B.5** Phishing is a type of social engineering in which an adversary attempts to influence a user into revealing personal information, such as passwords or verification codes. Adversaries can use that information to access accounts, financial services, or other protected systems. **4.3.B.6** Unsolicited emails, attachments, links, and forms can be used to gain unauthorized access by influencing a user into opening a file, entering information, or installing harmful software. These messages may come from unknown senders or from accounts that appear familiar because they were compromised. **4.3.B.7** Malicious links can be disguised to look trustworthy, such as by using misleading text, shortened URLs, or look-alike website addresses, leading users to fake pages or unsafe downloads. **4.3.B.8** Email is not inherently private. Email messages can be forwarded, stored, intercepted, or accessed by people other than the sender and receiver, which can expose sensitive information if an account or message is accessed by an unauthorized party. **4.3.B.9** Malware is software designed to harm a device or enable unauthorized access, such as by stealing data, monitoring user activity, or controlling parts of the system. Malware can be installed through unsafe downloads, malicious attachments, or compromised websites. **4.3.B.10** Adversaries may try to confuse users or get them to act quickly, such as by sending urgent "account locked" messages. Such situations increase the chances a user will click a link, share information, or approve access without noticing warning signs.
4.3.C Explain how digital systems and information can be protected.	**4.3.C.1** Authentication is how a system verifies a user's identity, such as with passwords, passkeys, or biometrics. Strong authentication can reduce unauthorized access to devices and accounts. **4.3.C.2** Multifactor authentication (MFA) requires two or more different pieces of evidence, typically in at least two of the following categories: knowledge (something a user knows), possession (something a user has), and biometrics (something a user is). MFA adds protection because an attacker must compromise multiple factors, not just a single password, to access an account.

© 2026 College Board

LEARNING OBJECTIVE

4.3.C

Explain how digital systems and information can be protected.

ESSENTIAL KNOWLEDGE

4.3.C.3

Encryption encodes data so it cannot be read without a key. Decryption decodes it so it can be read. Encryption helps protect data stored on devices and sent over networks.

EXCLUSION STATEMENT

Specific mathematical procedures for encryption and decryption are beyond the scope of the AP Computer Science Principles course and exam.

4.3.C.4

End-to-end encryption is a method of secure communication in which data are encrypted on the sender's device and can only be decrypted on the intended recipient's device, so services that transmit or store the data along the way cannot read the contents without the decryption key.

4.3.C.5

Systems can be compromised when adversaries exploit software flaws. Installing software updates helps reduce risk by fixing known vulnerabilities that adversaries may target.

4.3.C.6

Malware scanning tools can help detect or block some malicious software, reducing the chance that a device is infected or controlled without the user's knowledge.

4.3.C.7

Apps and websites sometimes request permissions like access to location, contacts, microphone, camera, or photos. Reviewing and limiting permissions helps protect privacy and reduces the chance of misuse if an app is untrustworthy or becomes compromised.

4.3.C.8

Security measures reduce risk in different ways: authentication and MFA limit who can access an account, encryption limits who can read data if it is intercepted or stolen, updates reduce known weaknesses that adversaries can exploit, and permissions limit what an app can access. No single measure prevents all compromise, so systems often use multiple protections together.

© 2026 College Board

TOPIC 4.4
Data Analysis

Required Course Content

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
4.4.A Determine what people can learn from analyzing data.	**4.4.A.1** Data-driven features, such as feeds, dashboards, recommendations, and filters, are used by people every day. Many apps generate annual profiles by organizing a user's activity into categories, such as top items, totals, and trends over time, revealing patterns in how the user engaged with the app.
	4.4.A.2 Data are raw values or observations. Information is what people learn from data after analyzing it to identify patterns or meaning. One common process is to collect or select data, clean it if needed, and then transform, filter, aggregate, and visualize it to answer questions and reveal patterns.
	4.4.A.3 Depending on how data is collected, it may not be uniform. When people enter data into an open text field, they may represent the same idea in different ways (e.g., using abbreviations, different spellings, or different capitalization). Real-world data is often messy because people use different terms, type differently, or make mistakes.
	4.4.A.4 Cleaning data is the process of making data more consistent and usable without changing its meaning, such as standardizing equivalent entries like "1/5," "01-05," and "Jan 5" to a single agreed-upon date format.
	4.4.A.5 Filtering data keeps only the items that meet a condition so patterns are easier to see, such as filtering for trips under 10 miles or survey responses that selected "strongly agree".
	4.4.A.6 Aggregating data combines many values into a summary, such as counting how many items meet a condition or computing a total or average. Comparing data values can identify key results, such as finding the maximum or minimum value or determining which category occurs most often.
	4.4.A.7 Visualizations, such as tables, charts, and graphs, communicate insights by making trends and comparisons easier to notice than raw data alone. Different visual formats highlight different patterns.
	4.4.A.8 Tools like spreadsheets can efficiently organize data, apply filters, sort records, and compute summaries, making it easier to explore patterns quickly.
	4.4.A.9 Programmers should choose data processes and visualizations that match user goals and avoid misleading conclusions. This includes using clear labels, selecting a visualization that fits the type of data and the question being asked, and ensuring filters match what users want to compare.

© 2026 College Board

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
4.4.B Determine what people can learn from metadata.	**4.4.B.1** Metadata are data about data. They describe properties of a digital item, such as when it was created, who created it, where it came from, its type or format, its size, or what device or app produced it. **4.4.B.2** Changing metadata does not change the primary data. Editing the date a photo was taken, for instance, does not alter the image itself. However, changing metadata can affect how the data is searched, sorted, grouped, or interpreted. **4.4.B.3** Metadata helps systems find, organize, and manage information by providing labels and structure, such as titles, tags, categories, timestamps, or file types. **4.4.B.4** Metadata adds context that makes data more useful for analysis, such as identifying the source of information, when it was recorded, or settings used to collect it, which helps people and programs compare data and interpret results more accurately. **4.4.B.5** People can use metadata to detect patterns or support conclusions without examining the full content, such as how often events occur or how items can be grouped or filtered. **4.4.B.6** Programmers use metadata to support user tasks such as sorting by most recent, filtering by location, and organizing items into groups like favorites. Clear metadata and labels help users find what they need quickly and reduce confusion or mistakes.
4.4.C Develop algorithms and code to analyze data.	**4.4.C.1** Programs can analyze data to reveal patterns, such as counts, totals, averages, trends, comparisons, and unusual values. This helps people answer questions and make decisions based on evidence. **4.4.C.2** To analyze data with a program, programmers often begin by stating a question or goal and defining what counts as relevant data and correct output. They then choose processes that match the goal, such as transforming values, filtering records, combining values to compute summaries, or comparing items to find a best match or extreme. **4.4.C.3** Programmers verify extracted information by checking sample rows, comparing against a small hand-checked subset, and confirming that filters and transformations match intended results and the conclusions match the data. **4.4.C.4** Many data analysis programs follow a common pattern: traverse a data set, apply a condition to keep or skip items, update an accumulator (e.g., a count, total, or maximum), and then produce output (e.g., a summary value, a new filtered collection, or a visualization-ready result).

© 2026 College Board

TOPIC 4.5

AI Algorithm Development

Required Course Content

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
4.5.A Describe the differences between deterministic algorithms and probabilistic algorithms.	**4.5.A.1** Many tools people use every day include both predictable steps and best-guess behavior. A calculator performing arithmetic produces the same output every time for the same input. Statistical AI systems (like spam filters and chatbots) use probability to choose what output is most likely under uncertainty, which means they can still make mistakes even when they behave consistently. Understanding these differences helps users decide when results should be checked or verified.
	4.5.A.2 A deterministic algorithm produces the same output every time it is given the same input, so its behavior is predictable and repeatable.
	4.5.A.3 A probabilistic algorithm makes decisions based on likelihoods rather than fixed rules. Because it models uncertainty, a probabilistic algorithm may produce outputs that include a confidence score or probability estimate. For a given input, the system will often produce the same result when the model and decision threshold stay the same, but the outcome can change if the threshold is adjusted or the model is updated.
	4.5.A.4 Probabilistic algorithms are often used for problems where inputs are incomplete, variable, or ambiguous and there is no single clear rule-based solution, such as recognizing objects in images, interpreting speech, filtering spam, or generating text. Because these systems make best-guess decisions, programmers evaluate them by testing performance across many examples and checking how often outputs are accurate, useful, and reliable for the intended context.
	4.5.A.5 Testing deterministic and probabilistic algorithms requires different approaches. Deterministic algorithms can be checked with fixed expected outputs, while probabilistic algorithms are evaluated by testing performance across many examples and checking whether outputs meet success criteria for the intended context.
4.5.B Compare the strengths and limitations of rule-based systems and AI models trained on data.	**4.5.B.1** AI systems can produce outputs using different approaches, such as rule-based systems and models trained on data. Each approach has different strengths and limitations, so the best choice depends on the task, available data, and the need for transparency.
	4.5.B.2 Rule-based systems use explicit rules created by people. They are predictable and easier to test, but they can struggle with complex real-world situations that are variable, ambiguous, or hard to capture with rules, such as identifying spam messages that change over time. A spam filter that relies only on fixed keyword rules may miss new scams or incorrectly flag legitimate messages because the same words can appear in both spam and nonspam messages.

© 2026 College Board

LEARNING OBJECTIVE

4.5.B

Compare the strengths and limitations of rule-based systems and AI models trained on data.

ESSENTIAL KNOWLEDGE

4.5.B.3

Some AI models use mathematical relationships to estimate likelihoods and make a best guess when information is uncertain or incomplete. They can be transparent and consistent, but their accuracy depends on whether the model's assumptions fit the real situation. For example, a weather forecast model might estimate the chance of rain based on pressure, humidity, and past patterns.

4.5.B.4

Machine learning models learn patterns from training data to make predictions, classify inputs, or generate outputs. They can perform well on complex pattern-based tasks, but results depend heavily on the quality and representativeness of the training data, and it may be difficult to explain why a particular output was produced. For example, an image classifier trained mostly on bright, high-quality photos may perform poorly on blurry or low-light images.

4.5.B.5

Choosing between rule-based systems and AI models trained on data depends on the task and constraints, such as the type of output needed, acceptable error rate, speed, cost, and whether people need a clear explanation of how the result was produced.

4.5.B.6

People interact with AI-enabled features every day, such as autocorrect, recommendations, spam filters, and face unlock. These tools produce outputs in different ways: some follow fixed rules, and others use models trained on data to estimate the most likely result. Rule-driven tools usually produce more predictable outputs. Model-driven tools may produce outputs that are uncertain, context dependent, or affected by changes to data, settings, or the model itself.

© 2026 College Board

TOPIC 4.6

Ethics and Effects of AI

Required Course Content

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
4.6.A Evaluate the implementation of AI systems using ethical design criteria.	**4.6.A.1** Ethical design uses criteria to evaluate AI systems, including fairness, transparency, privacy, safety, and accountability. Developers use these criteria to guide choices about data, model behavior, user experience, and safeguards.
	4.6.A.2 Developers use evidence to evaluate whether a system is meeting ethical design criteria. Methods include testing performance across different groups and contexts, reviewing how data is collected and used, observing users to identify confusion or barriers, and examining how the system's outputs are used in decision making.
	4.6.A.3 Ethical design often involves trade-offs, such as how personalization can conflict with privacy. Developers should document decisions, explain constraints, and choose designs that reduce risk and protect users.
	4.6.A.4 Developers assess impacts before deployment and revise designs based on user feedback, then continue monitoring after release because real-world use can reveal new issues or model drift.
	4.6.A.5 Documentation and testing support accountability by making it clear what the system does, what data it relies on, what limitations it has, and what safeguards or human-review steps are in place when the risks are high.
	4.6.A.6 To evaluate an AI system, people apply ethical design criteria by asking questions such as the following: Is the system fair across different users and contexts? Is the system transparent about when it is used and what it is doing? Does the system minimize data collection and protect privacy? Who is responsible when the system fails? What protections in the system reduce harm?
	4.6.A.7 Ethical evaluation includes proposing changes to reduce harm, such as improving disclosure, reducing data collection, adding human review for high-risk cases, expanding testing across groups, or giving users meaningful control like the ability to opt out or adjust settings.
4.6.B Explain how AI impacts society, the economy, and culture.	**4.6.B.1** AI systems are designed to optimize specific goals, such as efficiency, engagement, accuracy, or generating options. Because the goal shapes what the system prioritizes, it helps explain the kinds of outcomes the system produces in areas like health, transportation, education, and media.

© 2026 College Board

LEARNING OBJECTIVE

4.6.B

Explain how AI impacts society, the economy, and culture.

ESSENTIAL KNOWLEDGE

4.6.B.2

AI impacts can change when a tool is deployed in a new context or used beyond its original purpose. In these cases, mismatches between training data, design assumptions, and real-world use can lead to different outcomes.

4.6.B.3

The scale of an AI system refers to how widely it is used and how much it influences outcomes, such as the number of users, the volume of decisions it helps make, how often it is used, and the number of contexts in which it is used, including different schools, jobs, regions, or languages.

4.6.B.4

When an AI tool is used on a large scale, even small design choices, errors, or biased patterns can affect many people, and impacts that seem minor for one user can become significant across a large population. AI tools can also influence individual behavior and habits. When many people rely on the same tool, those shifts can accumulate and reshape group norms and expectations, such as how people write, search, or communicate, even if the tool was originally intended as only a helper.

4.6.B.5

AI can produce uneven outcomes through multiple pathways. These may include training data and labels that shape which users the system works well for, design choices that determine what errors are tolerated and what feedback users receive, and decision-making practices that determine how outputs are applied in real life.

4.6.B.6

AI can reshape work and economic systems by changing the cost, speed, and quality of tasks. This can increase productivity and create new jobs and skills needs, while also contributing to job displacement or changes in wages and working conditions.

4.6.B.7

AI shapes culture and communication by filtering what people see and what content gets created and shared. When these systems are widely used, they can influence what spreads, what is rewarded with attention, and what people consider normal or trustworthy.

4.6.B.8

AI systems can have environmental impacts due to energy use, computing infrastructure, and material consumption.

4.6.B.9

To analyze AI's broader impact, consider how the system is used: identify who uses it and who is affected, where it is deployed and at what scale, and how outputs influence decisions and behavior. Use these observations to determine who benefits, who may be harmed or left out, and what changes when context or scale changes.

© 2026 College Board

TOPIC 4.7
Machine Learning

Required Course Content

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
4.7.A Explain how data and probabilistic algorithms shape machine learning outputs.	**4.7.A.1** Training data for machine learning can come from sources such as sensors, surveys, simulations, or existing data sets. Depending on the task, the data may include labels, which are the correct outputs paired with example data inputs.
	4.7.A.2 Machine learning models use training data to learn patterns that connect inputs to outputs. These patterns come from the examples in the training data and, for supervised learning, the labels attached to them.
	4.7.A.3 Training data is often split into separate sets so the model can learn from one set and then be evaluated on different data it has not seen before. The training set is used to build the model, the validation set is used to adjust model choices and check performance during development, and the test set is used for a final check of how well the model performs on new examples.
	4.7.A.4 In supervised learning, training examples are paired with labels that represent the correct output. Using both examples that match the target and nonexamples that do not match the target helps the model learn boundaries and make more accurate distinctions.
	4.7.A.5 Many machine learning models produce probabilistic outputs, such as a likelihood or confidence score, rather than a guaranteed answer. People decide how to use these scores by choosing a threshold, which determines when the system accepts a prediction automatically and when it requires more checking or human review.
	4.7.A.6 When using a threshold, a system can make two kinds of mistakes. A false positive occurs when the system incorrectly accepts something that should be rejected. A false negative occurs when the system incorrectly rejects something that should be accepted. Changing the threshold usually reduces one kind of error while increasing the other, so developers choose thresholds by considering which error is more harmful for the user and the situation. Developers may also add human review or additional checks to reduce harm.
	4.7.A.7 Training sets that include diverse examples and counterexamples help a model generalize, meaning it performs well on new inputs and varied real-world situations, not just the examples it learned from.
	4.7.A.8 A model can appear to work well during development but still perform poorly on new inputs if it learns patterns that do not generalize beyond the training examples. Testing a model on new data helps evaluate whether its patterns transfer beyond the training sets.

© 2026 College Board

LEARNING OBJECTIVE

4.7.B

Explain how training data decisions can affect accuracy, privacy, and bias.

ESSENTIAL KNOWLEDGE

4.7.B.1

Training data may be processed before a model is trained to decide which examples are included and how they are organized. Sampling can reduce data set size or balance the number of examples across categories. Sampling decisions can improve performance for underrepresented cases, but they can also reduce accuracy if important cases are removed or under sampled.

4.7.B.2

Training data can expose sensitive information. Privacy protection should include minimizing the amount of, limiting access to, and protecting sensitive data during storage and sharing.

4.7.B.3

Bias can occur when training data does not represent the people or situations the model will be used for or when labels reflect human assumptions. Collecting more data does not necessarily reduce bias if the same sampling methods or labeling choices continue.

4.7.B.4

Training data should reflect the intended users and real-world contexts where the model will be used. If the data set differs from actual use, the model can perform poorly or unevenly across users even if it appears accurate during testing. Evaluating training data includes checking whether it represents the people, environments, languages, and situations the system is meant to support, and testing with representative user scenarios to reduce confusion and unequal performance.

© 2026 College Board

THIS PAGE HAS BEEN INTENTIONALLY LEFT BLANK

AP COMPUTER SCIENCE PRINCIPLES

UNIT 5

Exploring Computing Technologies

TOPIC 5.1 Hardware

LEARNING OBJECTIVE

5.1.A
Explain how basic computer hardware components work together to run programs and interact with users.

ESSENTIAL KNOWLEDGE

5.1.A.1
Computer hardware includes physical components that provide input, processing, storage, and output. Many input and output components are peripherals: hardware devices connected to the computer to extend what it can do. Some peripherals provide input, some provide output, and some do both, like touchscreens or headsets.

5.1.A.2
Input devices, such as keyboards, cameras, and sensors, capture data or user actions so a computer can respond.

5.1.A.3
The CPU (central processing unit) executes program instructions and performs calculations, using memory to keep track of needed values.

5.1.A.4
Primary memory, also known as random access memory (RAM), stores data and instructions that are currently in use. It is fast but temporary: its contents are lost when the power is turned off.

5.1.A.5
Secondary storage, such as solid-state drives (SSDs) or hard drives, stores programs and files long term. It is usually slower than RAM but keeps data when the power is off.

5.1.A.6
A computing system uses both memory (RAM) and storage to manage data. Memory (RAM) provides fast, temporary access to data and instructions the CPU is actively using and loses its contents when power is turned off. Storage provides longer-term, nonvolatile access to programs and files and keeps its contents when power is off, but is typically slower than RAM.

5.1.A.7
Output devices, such as monitors, speakers, and printers, present results to users or to other systems.

5.1.A.8
Hardware components communicate by sending data between parts of the system. Data moves between input devices, memory, the CPU, storage, and output devices through internal pathways called buses so the computer can complete tasks.

5.1.A.9
Hardware capabilities affect what problems a computer can handle and how efficiently it can run programs. Key factors include processing speed, memory size, storage capacity, and available inputs and outputs.

5.1.A.10
Some computers include specialized processors, such as graphics processing units (GPUs), to speed up graphics, video, or parallel tasks. Adding specialized hardware can increase cost and power use.

© 2026 College Board

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
5.1.B Explain how computing devices are used in industry to solve problems.	**5.1.B.1** Industries use a wide range of computing devices, such as robots, vehicle control units, medical imaging systems, manufacturing automation, and agricultural sensors, to accomplish tasks and address industry-specific needs.
	5.1.B.2 Industry devices often sense conditions (input), process readings, store values, and trigger actions (output), including controlling physical parts like motors, valves, or lights.
	5.1.B.3 In industry, computing devices are used to automate tasks, collect and process data, and support decision making. The effectiveness of a device depends on whether its hardware capabilities match the problem being solved and how well the software and system design use those capabilities.
	5.1.B.4 The design and capabilities of a computing system can be compared based on user needs and system requirements, such as storage, connectivity, and accessibility. Choices can involve societal, environmental, and ethical impacts, such as cost, energy use, and privacy.

© 2026 College Board

TOPIC 5.2
Software

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
5.2.A Explain how software manages hardware resources and other software components.	**5.2.A.1** Software is a set of instructions that a computer follows to perform tasks using hardware. **5.2.A.2** Many computing tasks depend on multiple software components, such as applications, libraries, and services, that work together to provide features. **5.2.A.3** An operating system (OS) is software that manages shared hardware resources, such as the CPU, memory, storage, and peripherals. The OS coordinates how programs use these resources by scheduling work, allocating memory, handling input and output, and coordinating access to devices so multiple programs can run reliably without interfering with each other. **5.2.A.4** Abstractions, such as files, windows, and device drivers, let software use hardware, like printers, cameras, and storage, without managing the electrical or device-specific details directly. **5.2.A.5** Hardware and software work together: hardware provides input and output, processing, and storage, and software controls how those capabilities are used to complete tasks.

© 2026 College Board

TOPIC 5.3
Troubleshooting

LEARNING OBJECTIVE

5.3.A
Describe systematic troubleshooting strategies used to identify and fix problems in computing systems.

ESSENTIAL KNOWLEDGE

5.3.A.1
Troubleshooting is a systematic process: identify symptoms, describe the problem clearly, form hypotheses about possible causes, test those hypotheses, apply a solution, and verify the fix by repeating the original task to confirm the user's goal is met.

5.3.A.2
Computing problems can result from many sources, including hardware failures, software errors, network issues, configuration settings, or user actions.

5.3.A.3
Effective troubleshooting starts by reproducing the problem and gathering evidence, such as the user's steps, recent changes, and any error messages.

5.3.A.4
Testing one change at a time and keeping notes helps isolate the cause of a problem and prevents introducing new problems.

5.3.A.5
Documentation, diagnostic tools, logs, and error messages can provide clues about what failed and where to focus next.

5.3.A.6
Troubleshooters isolate the problem by checking one layer at a time (device, network, account or settings, application), using simple tests to narrow where the failure occurs.

5.3.A.7
Troubleshooters may use documentation, online forums, or AI tools to research similar problems. They evaluate the credibility of sources, adapt suggested solutions to their specific system, and test changes systematically, using evidence to confirm whether each change helps.

5.3.A.8
If a fix does not work, troubleshooters revise their hypothesis, try a different test, and use what they learned to narrow the possibilities.

© 2026 College Board

TOPIC 5.4

Emerging Technology

LEARNING OBJECTIVE

5.4.A
Explain how emerging technologies impact existing technologies or projects.

ESSENTIAL KNOWLEDGE

5.4.A.1
Emerging technologies can change what is possible by improving efficiency, expanding functionality, increasing automation, or enabling new ways to solve problems.

5.4.A.2
New technologies can affect existing projects by changing requirements, constraints, and trade-offs, such as cost, time, performance, privacy, security, accessibility, or reliability.

5.4.A.3
Integrating an emerging technology often requires updating or redesigning parts of a system, like software and hardware, so components remain compatible.

5.4.A.4
Adoption may require changes to workflows and training because people may use tools differently and new skills may be needed to operate, maintain, or troubleshoot systems.

5.4.A.5
New technologies can introduce new risks or unintended impacts (e.g., bias, misinformation, security vulnerabilities, or increased energy use), so teams test, monitor, and revise systems after adoption.

5.4.A.6
Teams decide whether to use an emerging technology by comparing its capabilities and limitations to the project's requirements and constraints, including reliability, cost, and environmental impacts.

© 2026 College Board

TOPIC 5.5
Career Exploration

LEARNING OBJECTIVE

5.5.A
Explain how computing knowledge and skills support a range of different career pathways.

ESSENTIAL KNOWLEDGE

5.5.A.1
Many careers, such as those in healthcare, education, journalism, science, and art, require or benefit from computing knowledge, even when computing is not the primary focus of the career.

5.5.A.2
Computing knowledge includes concepts such as data, networks, cybersecurity and privacy, and how software systems work. Computing skills include being able to use problem-solving strategies, such as decomposition, pattern recognition, and abstraction, as well as analyzing data, automating tasks, and designing or evaluating systems. This knowledge and these skills are transferable across disciplines and can support work in many fields.

5.5.A.3
Understanding how computing is used in different fields can help people compare career pathways and plan what knowledge and skills to develop.

5.5.A.4
Career pathways differ in how they use computing to solve domain-specific problems. Examples may include building software, using tools to analyze data, managing systems, or developing or applying AI systems.

© 2026 College Board

THIS PAGE HAS BEEN INTENTIONALLY LEFT BLANK