

SPRINGFIELD PUBLIC SCHOOLS
ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Springfield Public Schools
Springfield, New Jersey 07081

AP Computer Science Principles

Grade 9-12

Curriculum Guide

2022

This curriculum may be modified through varying materials, techniques, and strategies as per an individual student's IEP.

Springfield Board of Education
Approved July 25, 2022



Springfield Public Schools
Springfield, New Jersey

Dr. Rachel Goldberg
Superintendent of Schools

Erica Scudero
Director of Curriculum and Instruction

Curriculum Developed By:

Dr. Norman Francis ~ Principal of Jonathan Dayton High School

Gregory Salmon ~ STEM Supervisor

Mary Ellen Powers ~ Math and Computer Science Teacher

Springfield Board of Education
PO Box 210
Springfield, New Jersey 07081

"Cultivating compassionate and extraordinary learners"

Rachel Goldberg
Superintendent of Schools
(973) 376-1025

Michelle Calas
School Business Administrator/
Board Secretary
(973) 376-1025

Resolution: School Government
Introduced By: Mr. Miller
Seconded By: Mrs. Gamarekian
Adopted: July 25, 2022

Curriculum Adoption

To adopt the following curriculum and courses for the 2022-2023 school year:

- Revised Elementary Curriculum due to revised state standards.
  - K-5 Social Studies
- Revised Middle School Curriculum due to revised state standards.
  - Grades 6-8 Social Studies
- Revised High School Curriculum due to revised state standards.
  - AP Macroeconomics
  - AP European History
  - AP US Government & Politics
  - AP US History
  - Contemporary Global Studies
  - Diversity Studies
  - Holocaust & Genocide
  - Sociology
  - US History to 1900
  - World History
  - World History Honors
  - 20th Century & Beyond
- Revised Middle School Courses due to new standards in Computer Science
  - PLTW Gateway: Computer Science for Innovators and Makers
  - PLTW Gateway: App Creators
- Revised High School Courses due to new standards in Computer Science
  - Introduction to Java
  - AP Computer Science Principles
  - Advanced Computer Science - Data Structures: Java
- New Elementary Courses
  - Kindergarten Computer Science
  - Grade 1 Computer Science
  - Grade 2 Computer Science
  - Anticipated - moving PLTW Launch down to Grade 3
  - PLTW Launch Grade 4
  - PLTW Launch Grade 5
- New Middle School Elective Course
  - Forensics

- New Elective Courses at FMG
  - Creative Writing
  - Drama/Theater/Playwriting
  - Journalism
  - Mock Trial
  - Model UN
  - Mysteries in History
  - Project Citizen
- New ELA Courses at FMG
  - Accelerated Communication Arts Grade 7
  - Accelerated Communication Arts Grade 8
- New Elective Courses at JDHS:
  - African American Studies

I, Michelle Calas, School Business Administrator/Board Secretary of the Springfield Board of Education, hereby certify that the extract from the minutes of a meeting of the Board of Education duly called and held on July 25, 2022 has been compared by me with the original minute book of said Board of Education, and is a true, complete and exact copy therefor, as far as the same relates to the subject matter referred to said contract.

IN WITNESS THEREOF, I have hereunto set my hand and seal of the Springfield Board of Education, this 26th of July 2022.

SEAL

Michelle Calas
School Business Administrator/Board Secretary

Course Standards

New Jersey Student Learning Standards

Computer Science and Design Thinking (Standard 8):

- **Computing Systems**
  - 8.1.12.CS.1: Describe ways in which integrated systems hide underlying implementation details to simplify user experiences.
  - 8.1.12.CS.2: Model interactions between application software, system software, and hardware.
  - 8.1.12.CS.3: Compare the functions of application software, system software, and hardware.
  - 8.1.12.CS.4: Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.
- **Networks and the Internet**
  - 8.1.12.NI.1: Evaluate the scalability and reliability of networks, by describing the relationship between routers, switches, servers, topology, and addressing.
  - 8.1.12.NI.2: Evaluate security measures to address various common security threats.
  - 8.1.12.NI.3: Explain how the needs of users and the sensitivity of data determine the level of security implemented.
  - 8.1.12.NI.4: Explain how decisions on methods to protect data are influenced by whether the data is at rest, in transit, or in use.
- **Impacts of Computing**
  - 8.1.12.IC.1: Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices.
  - 8.1.12.IC.2: Test and refine computational artifacts to reduce bias and equity deficits.
  - 8.1.12.IC.3: Predict the potential impacts and implications of emerging technologies on larger social, economic, and political structures, using evidence from credible sources.
- **Data and Analysis**
  - 8.1.12.DA.1: Create interactive data visualizations using software tools to help others better understand real world phenomena, including climate change.
  - 8.1.12.DA.2: Describe the trade-offs in how and where data is organized and stored.
  - 8.1.12.DA.3: Translate between decimal numbers and binary numbers.
  - 8.1.12.DA.4: Explain the relationship between binary numbers and the storage and use of data in a computing device.
  - 8.1.12.DA.5: Create data visualizations from large data sets to summarize, communicate, and support different interpretations of real-world phenomena.
  - 8.1.12.DA.6: Create and refine computational models to better represent the relationships among different elements of data collected from a phenomenon or process.
- **Algorithms and Programming**
  - 8.1.12.AP.1: Design algorithms to solve computational problems using a combination of original

and existing algorithms.
- 8.1.12.AP.2: Create generalized computational solutions using collections instead of repeatedly using simple variables.
- 8.1.12.AP.3: Select and combine control structures for a specific application based upon performance and readability, and identify trade-offs to justify the choice.
- 8.1.12.AP.4: Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue.
- 8.1.12.AP.5: Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
- 8.1.12.AP.6: Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.
- 8.1.12.AP.7: Collaboratively design and develop programs and artifacts for broad audiences by
- incorporating feedback from users.
- 8.1.12.AP.8: Evaluate and refine computational artifacts to make them more usable and accessible.
- 8.1.12.AP.9: Collaboratively document and present design decisions in the development of complex programs.

- **Interaction of Technology and Humans**
  - 8.2.12.ITH.1: Analyze a product to determine the impact that economic, political, social, and/or cultural factors have had on its design, including its design constraints.
  - 8.2.12.ITH.2: Propose an innovation to meet future demands supported by an analysis of the potential costs, benefits, trade-offs, and risks related to the use of the innovation.
  - 8.2.12.ITH.3: Analyze the impact that globalization, social media, and access to open source technologies has had on innovation and on a society's economy, politics, and culture.

2020 Career Readiness, Life Literacies, and Key Skills (Standard 9):

- **Career Awareness and Planning**
  - 9.2.12.CAP.3: Investigate how continuing education contributes to one's career and personal growth.
  - 9.2.12.CAP.4: Evaluate different careers and develop various plans (e.g., costs of public, private, training schools) and timetables for achieving them, including educational/training requirements, costs, loans, and debt repayment.
  - 9.2.12.CAP.5: Assess and modify a personal plan to support current interests and postsecondary plans. •
  - 9.2.12.CAP.6: Identify transferable skills in career choices and design alternative career plans based on those skills.
  - 9.2.12.CAP.7: Use online resources to examine licensing, certification, and credentialing requirements at the local, state, and national levels to maintain compliance with industry requirements in areas of career interest.
  - 9.2.12.CAP.8: Determine job entrance criteria (e.g., education credentials, math/writing/reading

comprehension tests, drug tests) used by employers in various industry sectors.

- **Creativity and Innovation**
  - 9.4.12.CI.1: Demonstrate the ability to reflect, analyze, and use creative skills and ideas.
  - 9.4.12.CI.2: Identify career pathways that highlight personal talents, skills, and abilities.
  - 9.4.12.CI.3: Investigate new challenges and opportunities for personal growth, advancement, and transition.
- **Critical Thinking and Problem-solving**
  - 9.4.12.CT.1: Identify problem-solving strategies used in the development of an innovative product or practice.
  - 9.4.12.CT.2: Explain the potential benefits of collaborating to enhance critical thinking and problem solving.
  - 9.4.12.CT.3: Enlist input from a variety of stakeholders (e.g., community members, experts in the field) to design a service learning activity that addresses a local or global issue (e.g., environmental justice).
  - 9.4.12.CT.4: Participate in online strategy and planning sessions for course-based, school-based, or other projects and determine the strategies that contribute to effective outcomes.
  - 9.4.12.DC.7: Evaluate the influence of digital communities on the nature, content and responsibilities of careers, and other aspects of society.
  - 9.4.12.DC.8: Explain how increased network connectivity and computing capabilities of everyday objects allow for innovative technological approaches to climate protection.
- **Technology Literacy**
  - 9.4.12.TL.1: Assess digital tools based on features such as accessibility options, capacities, and utility for accomplishing a specified task
  - 9.4.12.TL.2: Generate data using formula-based calculations in a spreadsheet and draw conclusions about the data.
  - 9.4.12.TL.3: Analyze the effectiveness of the process and quality of collaborative environments.
  - 9.4.12.TL.4: Collaborate in online learning communities or social networks or virtual worlds to analyze and propose a resolution to a real-world problem.



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

AP CollegeBoard

AP® Computer Science Principles

COURSE AND EXAM DESCRIPTION

Effective
Fall 2020

AP COURSE AND EXAM DESCRIPTIONS ARE UPDATED PERIODICALLY

Please visit AP Central (apcentral.collegeboard.org) to determine whether a more recent course and exam description is available.



About the AP Computer Science Principles Course

AP Computer Science Principles introduces students to the breadth of the field of computer science. In this course, students will learn to design and evaluate solutions and to apply computer science to solve problems through the development of algorithms and programs. They will incorporate abstraction into programs and use data to discover new knowledge. Students will also explain how computing innovations and computing systems, including the Internet, work, explore their potential impacts, and contribute to a computing culture that is collaborative and ethical. It is important to note that the AP Computer Science Principles course does not have a designated programming language. Teachers have the flexibility to choose a programming language(s) that is most appropriate for their students to use in the classroom.

College Course Equivalent

AP Computer Science Principles is equivalent to a first-semester, college-level breadth course in computer science.

Prerequisites

It is recommended that students in the AP Computer Science Principles course have successfully completed a first-year high school algebra course with a strong foundation of basic linear functions, composition of functions, and problem-solving strategies that require multiple approaches and collaborative efforts. In addition, students should be able to use a Cartesian (x, y) coordinate system to represent points on a plane. It is important that students and their advisers understand that any significant computer science course builds upon a foundation of mathematical reasoning that should be acquired before attempting such a course. Prior computer science experience is not required to take this course.



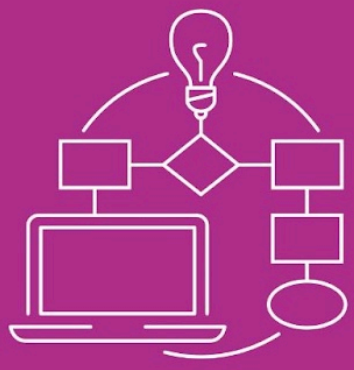
AP COMPUTER SCIENCE PRINCIPLES

Course Framework



Springfield Public Schools

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS



Introduction

Computer science involves problem-solving, hardware, and algorithms that help people utilize 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 our lives, it is important to understand the impact of computer science and how to maintain privacy, safety, and security not only when using computers but also while being the innovators of new computing applications. The course strives to engage all students, including those who have traditionally been underrepresented in computer science—such as female students, students of color, students with disabilities, and rural students—by allowing them to discover the power of computer science through rewarding yet challenging concepts.

A well-designed AP Computer Science Principles course that includes opportunities for students to collaborate to solve problems of their choice can help address traditional issues of equity and access. Such a course can broaden participation in computing while providing a strong and engaging introduction to the breadth of topics in the discipline.

The AP Computer Science Principles course reflects what computer science teachers, professors, and researchers have indicated are the main goals of an introductory, college-level computer science course:

- **Computational Solution Design**—Design and evaluate computational solutions for a purpose.
- **Algorithms and Program Development**—Develop and implement algorithms.
- **Abstraction in Program Development**—Develop programs that incorporate abstractions.
- **Code Analysis**—Evaluate and test algorithms and programs.
- **Computing Innovations**—Investigate computing innovations.
- **Responsible Computing**—Contribute to an inclusive, safe, collaborative, and ethical computing culture.

Students practice their computer science skills when designing and developing programs that address real-world problems and when investigating computing innovations they use or are interested in better understanding.

Compatible Curricula

The AP Computer Science Principles course surveys topics across several knowledge areas recommended by the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers Computing Society (IEEE-CS). Topics from the following knowledge areas have been included in this breadth-first approach to computer science: "Networking and Communication", "Parallel and Distributed Computing," "Software Development Fundamentals," "Programming Languages," "Architecture and Organization," "Computational Science," "Information Assurance and Security," and "Algorithms and Complexity." Teachers can review the **Computer Science Curricula from ACM and IEEE-CS** to see their complete curriculum guidelines.

The AP Computer Science Principles course vertically aligns with the core concepts in the **Computer Science Teachers Association (CSTA) K–12 Computer Science Framework**. Most of the K–12 Core Practices and Concepts are extended in the AP Computer Science Principles course, and the expected depth of knowledge is significantly higher in the AP course. This vertical alignment allows CS teachers to make connections from earlier courses to the college-equivalent AP Computer Science Principles course.

AP Computer Science Program

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

Resource Requirements

Students should have access to a computer system(s) that contains appropriate software to create and edit programs and allows students to practice, complete, and submit the AP Computer Science Principles performance task. The computer must have Internet access and be able to access the sites necessary for students to be successful in the course and assessment. The school ensures that each student has access to the AP Computer Science Principles Exam Reference Sheet (see Appendix), as well as performance task directions and scoring guidelines. The school ensures that each student has a college-level text or curricular resources deemed necessary by the teacher for individual use inside and outside of the classroom.

Course Framework Components

Overview

This course framework provides a description of what students should know and be able to do to qualify for college credit or placement.

The course framework includes two essential components:

1 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.

2 COURSE CONTENT

The course content is organized into big ideas, which are cross-cutting concepts that build conceptual understanding and spiral throughout the course. The content and conceptual understandings within the big ideas reflect what colleges and universities typically expect students to master to qualify for college credit and/or placement.

THIS PAGE IS INTENTIONALLY LEFT BLANK.

1

AP COMPUTER SCIENCE PRINCIPLES

Computational Thinking Practices

The table that follows presents the computational thinking practices that students should develop during the AP Computer Science Principles course. The practices form the basis of tasks on the AP Exam.

The learning objectives found in the big idea guides are each aligned to one of the skills from a practice. Teachers will want to be sure to integrate the practices and the course content with enough repetition to prepare students to transfer these skills when taking the AP Exam.

More detailed information about teaching the computational thinking practices can be found in the Instructional Approaches section of this publication.



AP COMPUTER SCIENCE PRINCIPLES

Computational Thinking Practices: Skills

Practice 1	Practice 2	Practice 3	Practice 4	Practice 5	Practice 6
Computational Solution Design 1 Design and evaluate computational solutions for a purpose.	***Algorithms and Program Development*** 2 Develop and implement algorithms.	***Abstraction in Program Development*** 3 Develop programs that incorporate abstractions.	***Code Analysis*** 4 Evaluate and test algorithms and programs.	***Computing Innovations*** 5 Investigate computing innovations.	***Responsible Computing*** 6 Contribute to an inclusive, safe, collaborative, and ethical computing culture.
SKILLS					
1.A Investigate the situation, context, or task. 1.B Determine and design an appropriate method or approach to achieve the purpose. 1.C Explain how collaboration affects the development of a solution. 1.D Evaluate solution options.	2.A Represent algorithmic processes without using a programming language. 2.B Implement and apply an algorithm.	3.A Generalize data sources through variables. 3.B Use abstraction to manage complexity in a program. 3.C Explain how abstraction manages complexity.	4.A Explain how a code segment or program functions. 4.B Determine the result of code segments. 4.C Identify and correct errors in algorithms and programs, including error discovery through testing.	5.A Explain how computing systems work. 5.B Explain how knowledge can be generated from data. 5.C Describe the impact of a computing innovation. 5.D Describe the impact of gathering data. 5.E Evaluate the use of computing based on legal and ethical factors.	6.A Collaborate in the development of solutions. 6.B Use safe and secure methods when using computing devices. 6.C Acknowledge the intellectual property of others.

***NOTE:** All computational thinking practices except Computational Thinking Practice 6 are assessed in the multiple-choice section of the AP Exam.

2

AP COMPUTER SCIENCE PRINCIPLES

Course Content

Based on the Understanding by Design® (Wiggins and McTighe) model, this course framework provides a description of the course requirements necessary for student success, with a focus on big ideas that encompass core principles, theories, and processes of the discipline. The framework also encourages instruction that prepares students for advanced computer science coursework and its integration into a wide array of STEM-related fields.

Big Ideas

The big ideas serve as the foundation of the course and help students create meaningful connections among concepts. They are often overarching concepts or themes that become threads that run throughout the course. Revisiting the big ideas and applying them in a variety of contexts enables students to develop deeper conceptual understanding. Below are the big ideas of the course and a brief description of each.

BIG IDEA 1: CREATIVE DEVELOPMENT (CRD)
When developing computing innovations, developers can use a formal, iterative design process or a less rigid process of experimentation. While using either approach, developers will encounter phases of investigating and reflecting, designing, prototyping, and testing. Additionally, collaboration is an important tool at any phase of development, because considering multiple perspectives allows for improvement of innovations.

BIG IDEA 2: DATA (DAT)
Data are central to computing innovations because they communicate initial conditions to programs and represent new knowledge. Computers consume data, transform data, and produce new data, allowing users to create new information or knowledge to solve problems through the interpretation of those data. Computers store data digitally, which means that the data must be manipulated in order to be presented in a useful way to the user.

continued on next page

BIG IDEA 3: ALGORITHMS AND PROGRAMMING (AAP)

Programmers integrate algorithms and abstraction to create programs for creative purposes and to solve problems. Using multiple program statements in a specified order, making decisions, and repeating the same process multiple times are the building blocks of programs. Incorporating elements of abstraction—by breaking problems down into interacting pieces, each with their own purpose—makes writing complex programs easier. Programmers need to think algorithmically and use abstraction to define and interpret processes that are used in a program.

BIG IDEA 4: COMPUTING SYSTEMS AND NETWORKS (CSN)

Computer systems and networks are used to transfer data. One of the largest and most commonly used networks is the Internet. Through a series of protocols, the Internet can be used to send and receive information and ideas throughout the world. Transferring and processing information can be slow when done on a single computer, but leveraging multiple computers to do the work at the same time can significantly shorten the time it takes to complete tasks or solve problems.

BIG IDEA 5: IMPACT OF COMPUTING (IOC)

Computers and computing have revolutionized our lives. To use computing safely and responsibly, we need to be aware of privacy, security, and ethical issues. As programmers, we need to understand the potential impacts of our programs and be responsible for the consequences. As computer users, we need to understand any potential beneficial or harmful effects and how to protect ourselves and our privacy when using a computer.

The five big ideas in AP Computer Science Principles, and their weighting on the multiple-choice section of the AP Exam, are listed below.

TOPICS

Each big idea is broken down into teachable segments called *topics*. The topic pages (starting on page 32) contain all required content for each topic. Although most topics can be taught in one or two class periods, teachers are encouraged to pace their course to suit the needs of their students and school.

Big Ideas	Exam Weighting
Big Idea 1: Creative Development	**10–13%**
Big Idea 2: Data	**17–22%**
Big Idea 3: Algorithms and Programming	**30–35%**
Big Idea 4: Computer Systems and Networks	**11–15%**
Big Idea 5: Impact of Computing	**21–26%**



Course at a Glance

Plan

The Course at a Glance provides a useful visual organization of the AP Computer Science Principles curricular components, including the following:

- Big ideas, along with approximate weighting
- Progression of topics within each big idea
- Spiraling of practices across big ideas

This Course at a Glance is organized by big ideas rather than units of instruction. Within each big idea are topics. Teachers and AP endorsed providers can group topics together to create units or modules.

Teach

COMPUTATIONAL THINKING PRACTICES
Practices spiral across big ideas.

1 Computational Solution Design
2 Algorithms and Program Development
3 Abstraction in Program Development
4 Code Analysis
5 Computing Innovations
6 Responsible Computing

Assess

Assign the Topic Questions—either as homework or in class—for each big idea. The Topic Questions are formative AP questions that provide feedback to students on the areas where they need to focus.

BIG IDEA 1 Creative Development

10–13% AP Exam Weighting

- 1, 6 — 1.1 Collaboration
- 1, 3, 4 — 1.2 Program Function and Purpose
- 1, 4, 6 — 1.3 Program Design and Development
- 1, 4 — 1.4 Identifying and Correcting Errors

Topic Questions
Multiple-choice: ~20 questions

BIG IDEA 2 Data

17–22% AP Exam Weighting

- 1, 2, 3 — 2.1 Binary Numbers
- 1 — 2.2 Data Compression
- 5 — 2.3 Extracting Information from Data
- 2, 5 — 2.4 Using Programs with Data

Topic Questions
Multiple-choice: ~20 questions



BIG IDEA 3 — Algorithms and Programming

30–35% AP Exam Weighting

3, 4	3.1 Variables and Assignments
3	3.2 Data Abstraction
2, 4	3.3 Mathematical Expressions
4	3.4 Strings
2, 4	3.5 Boolean Expressions
2, 4	3.6 Conditionals
2, 4	3.7 Nested Conditionals
2, 4	3.8 Iteration
1, 2	3.9 Developing Algorithms
2, 4	3.10 Lists
1	3.11 Binary Search
3, 4	3.12 Calling Procedures
3	3.13 Developing Procedures
2	3.14 Libraries
2, 4	3.15 Random Values
1	3.16 Simulations
1	3.17 Algorithmic Efficiency
1	3.18 Undecidable Problems

Topic Questions

Multiple-choice: ~90 questions
Performance Task: ~20 prompts

BIG IDEA 4 — Computer Systems and Networks

11-15% AP Exam Weighting

5	4.1 The Internet
1, 5	4.2 Fault Tolerance
1	4.3 Parallel and Distributed Computing

Topic Questions

Multiple-choice: ~10 questions

BIG IDEA 5 — Impact of Computing

21–26% AP Exam Weighting

5	5.1 Beneficial and Harmful Effects
5	5.2 Digital Divide
5	5.3 Computing Bias
1	5.4 Crowdsourcing
5	5.5 Legal and Ethical Concerns
5	5.6 Safe Computing

Topic Questions

Multiple-choice: ~20 questions

THIS PAGE IS INTENTIONALLY LEFT BLANK.



AP COMPUTER SCIENCE PRINCIPLES

Big Idea Guides

Designed with input from the community of AP Computer Science Principles educators, the big idea guides offer all teachers helpful guidance in building students' skills and knowledge. It is important to remember that big ideas are themes that run across the entire school year, rather than units of instruction. Within each big idea are topics. Teachers and AP endorsed providers can group topics together to create units or modules for instruction. In some cases, it may be appropriate for a topic to appear in more than one unit or module.

A benefit of topics is that they enable the AP Program to provide interested teachers with formative Topic Questions that they can assign to their students, either throughout a unit or module or at the end, to gauge progress toward success on the AP Exam.



Using the Big Idea Guides



BIG IDEA 1 — 10–13% AP EXAM WEIGHTING

Creative Development

ESSENTIAL QUESTIONS

EQ1.1
- How has working collaboratively with other students improved an overall project?
- What are some ways you can collect additional feedback on your program to use for improvements?

EQ1.2
- What are some ways you currently plan your work before starting a project?
- What apps or programs have you stopped using because you didn't like the design of how you interacted with it?

Developing Understanding

Collaboration is crucial when developing computing innovations, because having multiple perspectives offers opportunities to improve the design of innovations. In this big idea, students work collaboratively to design and develop programs using an iterative development process. They identify the needs of all users by gathering input from people from different backgrounds and demographics. Once the program is developed, they test it to ensure it meets these needs.

Effective collaboration can often differ from group work, because it requires equal participation and voice from all members of the group. Early in the school year, it may be helpful for teachers to establish practices and norms that facilitate a collaborative environment and provide students with time to practice working together. Content in this big idea is often paired with Big Idea 3: Algorithms and Programming.

Building Computational Thinking Practices

1.1 1.3 4.4 4.5

When designing a solution to a problem, programmers consider both the program itself and the way the user will interact with the program: the user interface. A well-designed user interface makes it easy for the user to understand what data are required as input for the program to complete its tasks.

When creating diagrams of their programs, students will benefit from considering how they want their program to behave based on identified inputs. Planning ahead may help them determine what abstractions can be developed and can help identify logic errors early in development.

When implementing program design, programmers often use documentation to explain the purpose of various code segments and describe how they function together in the program. Students' diagrams can be a good place to start writing their documentation. It may be more helpful to concentrate on documenting smaller code segments rather than trying to describe larger sections all at once.

Preparing for the AP Exam

Students will be expected to design and implement a program of their choice for the Create performance task. While students select their own topic for this task, they are required to include certain elements, such as lists and procedures, in their program code. Providing students with exemplars may help them consider the types of programs that can be developed while still meeting the requirement.

Students will need practice identifying and correcting errors to prepare for the AP Exam. One way to give students this practice is to provide them with prewritten program code to correct.

AP Computer Science Principles Course and Exam Description — Course Framework V.1 | 29

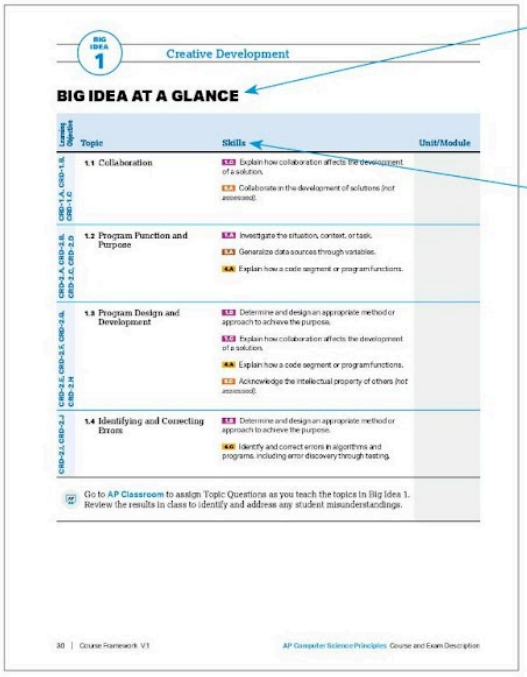
BIG IDEA OPENERS

The **Essential Questions** are thought-provoking questions that motivate students and inspire inquiry. They are organized by the enduring understandings that appear in the big idea.

Developing Understanding provides an overview that contextualizes and situates the key content of the big idea within the scope of the course. **Big Ideas** are overarching concepts or themes that become threads that run throughout the course.

Building Computational Thinking Practices describes specific skills within the practices that are appropriate to focus on in that big idea.

Preparing for the AP Exam provides helpful tips and common student misunderstandings identified from prior exam data.



BIG IDEA 1 — Creative Development

BIG IDEA AT A GLANCE

Learning Objective	Topic	Skills	Unit/Module
CRD-1.A, CRD-1.B, CRD-1.C	1.1 Collaboration	1.1 Explain how collaboration affects the development of a solution. 1.2 Collaborate in the development of solutions (not assessed).	
CRD-2.A, CRD-2.B, CRD-2.C, CRD-2.D	1.2 Program Function and Purpose	1.3 Investigate the situation, context, or task. 3.3 Generalize data sources through variables. 4.4 Explain how a code segment or program functions.	
CRD-2.E, CRD-2.F, CRD-2.G, CRD-2.H	1.3 Program Design and Development	1.3 Determine and design an appropriate method or approach to achieve the purpose. 1.1 Explain how collaboration affects the development of a solution. 4.4 Explain how a code segment or program functions. 5.3 Acknowledge the intellectual property of others (not assessed).	
CRD-2.I, CRD-2.J	1.4 Identifying and Correcting Errors	1.3 Determine and design an appropriate method or approach to achieve the purpose. 4.6 Identify and correct errors in algorithms and programs, including error discovery through testing.	

Go to AP Classroom to assign Topic Questions as you teach the topics in Big Idea 1. Review the results in class to identify and address any student misunderstandings.

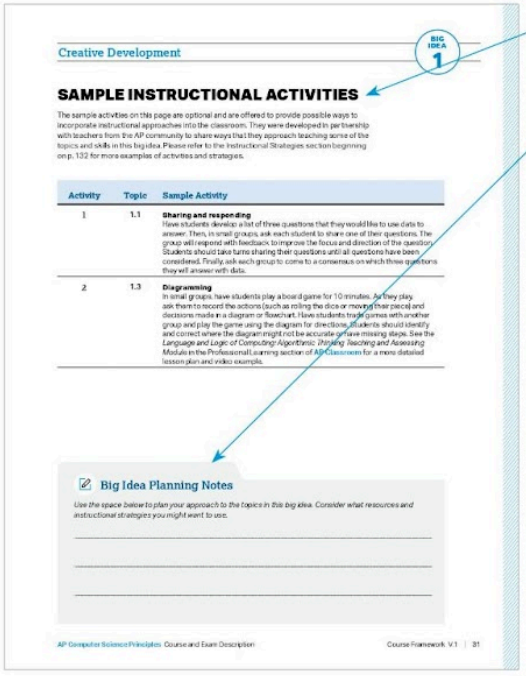
30 | Course Framework V.1 — AP Computer Science Principles Course and Exam Description

The **Big Idea at a Glance** table shows the topics and related learning objectives and skills. The Unit/Module column has been left blank so that teachers can indicate where they are going to place each topic in their course. Teachers who are using an AP endorsed provider can determine where each topic is being taught in their provider's curriculum.

The **Skills** for each topic show how the content in that topic has been linked to specific AP Computer Science Principles skills.

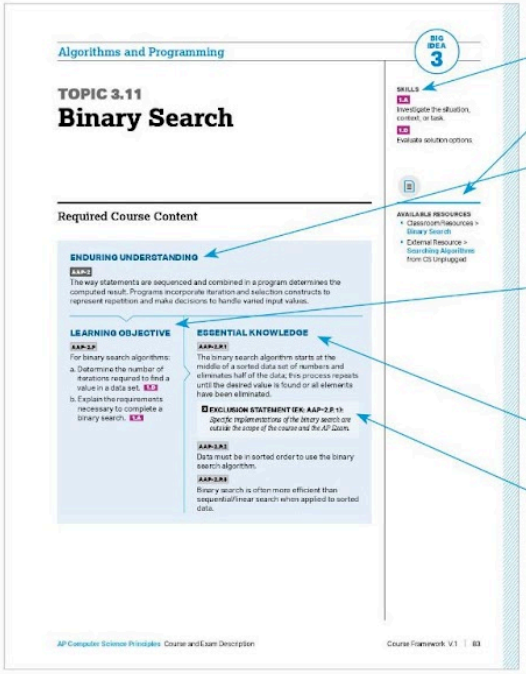


Using the Big Idea Guides



The **Sample Instructional Activities** page includes optional activities that can help teachers tie together the content and skill of a particular topic.

The **Big Idea Planning Notes** section provides space for teachers to make notes on their approach to the individual topics and the big idea as a whole.



TOPIC PAGES

The **Skills** section offers one or more skills related to the topic.

Where possible, **available resources** are listed that might help teachers address a particular topic in their classroom.

Enduring understandings are the long-term takeaways related to the big ideas that leave a lasting impression on students. Students build and earn these understandings over time by exploring and applying course content throughout the year.

Learning objectives define what a student should be able to do with content knowledge to progress toward the enduring understandings. Each learning objective is aligned to a particular skill, and that skill appears in a colored box after the learning objective.

Essential knowledge statements describe the knowledge required to perform the learning objective.

Exclusion statements define content or specific details about content that does not need to be included in the course. The content in the exclusion statements will not be assessed on the AP Computer Science Principles Exam.



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

REQUIRED COURSE CONTENT LABELING SYSTEM

BIG IDEA	ENDURING UNDERSTANDING	LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
DAT Data	**DAT-1** The way a computer represents data internally is different from the way the data are interpreted and displayed for the user. Programs are used to translate data into a representation more easily understood by people.	**DAT-1.A** Explain how data can be represented using bits. 3.C	**DAT-1.A.1** Data values can be stored in variables, lists of items, or standalone constants and can be passed as input to (or output from) procedures.

NOTE: Labels are used to distinguish each element of the required course content and are used throughout this course and exam description. Additionally, they are used in the AP Question Bank and other resources found on AP Classroom. Enduring understandings are labeled sequentially according to the big idea that they are related to. Learning objectives are labeled to correspond with the enduring understanding they relate to. Finally, essential knowledge statements are labeled to correspond with the learning objective they relate to.



AP COMPUTER SCIENCE PRINCIPLES

BIG IDEA 1

Creative Development

AP

10–13%
AP EXAM WEIGHTING

AP

Remember to go to **AP Classroom** to assign students the online Topic Questions for this big idea.

Whether assigned as homework or completed in class, the Topic Questions can be used to spot-check student understanding and help identify content and skills to emphasize in lessons.

Topic Questions

Multiple-choice: ~20 questions

BIG IDEA 1

10–13% AP EXAM WEIGHTING

Creative Development

ESSENTIAL QUESTIONS

CRD-1

- How has working collaboratively with other students improved an overall project?
- What are some ways you can collect additional feedback on your program to use for improvements?

CRD-2

- What are some ways you currently plan your work before starting a project?
- What apps or programs have you stopped using because you didn't like the design of how you interacted with it?

Developing Understanding

Collaboration is crucial when developing computing innovations, because having multiple perspectives offers opportunities to improve the design of innovations. In this big idea, students work collaboratively to design and develop programs using an iterative development process. They identify the needs of all users by gathering input from people from different backgrounds and demographics. Once the program is developed, they test it to ensure it meets these needs.

Effective collaboration can often differ from group work, because it requires equal participation and voice from all members of the group. Early in the school year, it may be helpful for teachers to establish practices and norms that facilitate a collaborative environment and provide students with time to practice working together. Content in this big idea is often paired with Big Idea 3: Algorithms and Programming.

Building Computational Thinking Practices

1.B 3.A 4.A 4.C

When designing a solution to a problem, programmers consider both the program itself and the way the user will interact with the program: the user interface. A well-designed user interface makes it easy for the user to understand what data are required as input for the program to complete its tasks.

When creating diagrams of their programs, students will benefit from considering how they want their program to behave based on identified inputs. Planning ahead may help them determine what abstractions can be developed and can help identify logic errors early in development.

When implementing program design, programmers often use documentation to explain the purpose of various code segments and describe how they function together in the program. Students' diagrams can be a good place to start writing their documentation. It may be more helpful to concentrate on documenting smaller code segments rather than trying to describe larger sections all at once.

Preparing for the AP Exam

Students will be expected to design and implement a program of their choice for the Create performance task. While students select their own topic for this task, they are required to include certain elements, such as lists and procedures, in their program code. Providing students with exemplars may help them consider the types of programs that can be developed while still meeting this requirement.

Students will need practice identifying and correcting errors to prepare for the AP Exam. One way to give students this practice is to provide them with prewritten program code to correct.



BIG IDEA 1

Creative Development

BIG IDEA AT A GLANCE

Learning Objective	Topic	Skills	Unit/Module
CRD-1.A, CRD-1.B, CRD-1.C	**1.1 Collaboration**	1.C Explain how collaboration affects the development of a solution. 6.A Collaborate in the development of solutions *(not assessed)*.	
CRD-2.A, CRD-2.B, CRD-2.C, CRD-2.D	**1.2 Program Function and Purpose**	1.A Investigate the situation, context, or task. 3.A Generalize data sources through variables. 4.A Explain how a code segment or program functions.	
CRD-2.E, CRD-2.F, CRD-2.G, CRD-2.H	**1.3 Program Design and Development**	1.B Determine and design an appropriate method or approach to achieve the purpose. 1.C Explain how collaboration affects the development of a solution. 4.A Explain how a code segment or program functions. 6.C Acknowledge the intellectual property of others *(not assessed)*.	
CRD-2.I, CRD-2.J	**1.4 Identifying and Correcting Errors**	1.B Determine and design an appropriate method or approach to achieve the purpose. 4.C Identify and correct errors in algorithms and programs, including error discovery through testing.	

AP Go to **AP Classroom** to assign Topic Questions as you teach the topics in Big Idea 1. Review the results in class to identify and address any student misunderstandings.

Creative Development

BIG IDEA 1

SAMPLE INSTRUCTIONAL ACTIVITIES

The sample activities on this page are optional and are offered to provide possible ways to incorporate instructional approaches into the classroom. They were developed in partnership with teachers from the AP community to share ways that they approach teaching some of the topics and skills in this big idea. Please refer to the Instructional Strategies section beginning on p. 132 for more examples of activities and strategies.

Activity	Topic	Sample Activity
1	1.1	**Sharing and responding** Have students develop a list of three questions that they would like to use data to answer. Then, in small groups, ask each student to share one of their questions. The group will respond with feedback to improve the focus and direction of the question. Students should take turns sharing their questions until all questions have been considered. Finally, ask each group to come to a consensus on which three questions they will answer with data.
2	1.3	**Diagramming** In small groups, have students play a board game for 10 minutes. As they play, ask them to record the actions (such as rolling the dice or moving their piece) and decisions made in a diagram or flowchart. Have students trade games with another group and play the game using the diagram for directions. Students should identify and correct where the diagram might not be accurate or have missing steps. See the *Language and Logic of Computing: Algorithmic Thinking Teaching and Assessing Module* in the Professional Learning section of **AP Classroom** for a more detailed lesson plan and video example.

Big Idea Planning Notes

Use the space below to plan your approach to the topics in this big idea. Consider what resources and instructional strategies you might want to use.

BIG IDEA 1

Creative Development

SKILLS

1.C
Explain how collaboration affects the development of a solution.

6.A
Collaborate in the development of solutions.

AVAILABLE RESOURCES

- External Resources >
  - **Collaboration Tools** from Cornell University Center for Teaching Innovation
  - **4 Methods to Enhance Student Collaboration in the Classroom** from Concordia University-Portland

TOPIC 1.1
Collaboration

Required Course Content

ENDURING UNDERSTANDING

CRD-1
Incorporating multiple perspectives through collaboration improves computing innovations as they are developed.

LEARNING OBJECTIVE

CRD-1.A
Explain how computing innovations are improved through collaboration. **1.C**

ESSENTIAL KNOWLEDGE

CRD-1.A.1
A computing innovation includes a program as an integral part of its function.

CRD-1.A.2
A computing innovation can be physical (e.g., self-driving car), nonphysical computing software (e.g., picture editing software), or a nonphysical computing concept (e.g., e-commerce).

CRD-1.A.3
Effective collaboration produces a computing innovation that reflects the diversity of talents and perspectives of those who designed it.

CRD-1.A.4
Collaboration that includes diverse perspectives helps avoid bias in the development of computing innovations.

CRD-1.A.5
Consultation and communication with users are important aspects of the development of computing innovations.

continued on next page

Creative Development

BIG IDEA 1

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
CRD-1.A Explain how computing innovations are improved through collaboration. 1.C	CRD-1.A.6 Information gathered from potential users can be used to understand the purpose of a program from diverse perspectives and to develop a program that fully incorporates these perspectives.
CRD-1.B Explain how computing innovations are developed by groups of people. 1.C	CRD-1.B.1 Online tools support collaboration by allowing programmers to share and provide feedback on ideas and documents. CRD-1.B.2 Common models such as pair programming exist to facilitate collaboration.
CRD-1.C Demonstrate effective interpersonal skills during collaboration. 1.C	CRD-1.C.1 Effective collaborative teams practice interpersonal skills, including but not limited to: ▪ communication ▪ consensus building ▪ conflict resolution ▪ negotiation

BIG IDEA 1

Creative Development

SKILLS

1.A Investigate the situation, context, or task.

3.A Generalize data sources through variables.

4.A Explain how a code segment or program functions.

AVAILABLE RESOURCES

- Professional Development > Teaching and Assessing Module: Explaining Processes
- External Resource > ACM Tech News

TOPIC 1.2
Program Function and Purpose

Required Course Content

ENDURING UNDERSTANDING

CRD-2

Developers create and innovate using an iterative design process that is user-focused, that incorporates implementation/feedback cycles, and that leaves ample room for experimentation and risk-taking.

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
CRD-2.A Describe the purpose of a computing innovation. **1.A**	**CRD-2.A.1** The purpose of computing innovations is to solve problems or to pursue interests through creative expression.
	CRD-2.A.2 An understanding of the purpose of a computing innovation provides developers with an improved ability to develop that computing innovation.
CRD-2.B Explain how a program or code segment functions. **4.A**	**CRD-2.B.1** A *program* is a collection of program statements that performs a specific task when run by a computer. A program is often referred to as *software*.
	CRD-2.B.2 A *code segment* is a collection of program statements that is part of a program.
	CRD-2.B.3 A program needs to work for a variety of inputs and situations.
	CRD-2.B.4 The *behavior* of a program is how a program functions during execution and is often described by how a user interacts with it.

continued on next page



Creative Development

BIG IDEA 1

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
CRD-2.B Explain how a program or code segment functions. 4.A	CRD-2.B.5 A program can be described broadly by what it does, or in more detail by both what the program does and how the program statements accomplish this function.
CRD-2.C Identify input(s) to a program. 3.A	CRD-2.C.1 *Program inputs* are data sent to a computer for processing by a program. Input can come in a variety of forms, such as tactile, audio, visual, or text. CRD-2.C.2 An *event* is associated with an action and supplies input data to a program. CRD-2.C.3 Events can be generated when a key is pressed, a mouse is clicked, a program is started, or any other defined action occurs that affects the flow of execution. CRD-2.C.4 Inputs usually affect the output produced by a program. CRD-2.C.5 In event-driven programming, program statements are executed when triggered rather than through the sequential flow of control. CRD-2.C.6 Input can come from a user or other programs.
CRD-2.D Identify output(s) produced by a program. 3.A	CRD-2.D.1 *Program outputs* are any data sent from a program to a device. Program output can come in a variety of forms, such as tactile, audio, visual, or text. CRD-2.D.2 Program output is usually based on a program's input or prior state (e.g., internal values).

BIG IDEA 1

Creative Development

SKILLS

1.B Determine and design an appropriate method or approach to achieve the purpose.

1.C Explain how collaboration affects the development of a solution.

4.A Explain how a code segment or program functions.

6.C Acknowledge the intellectual property of others.

AVAILABLE RESOURCE

- Professional Development > Teaching and Assessing Module: Explaining Processes

TOPIC 1.3
Program Design and Development

Required Course Content

ENDURING UNDERSTANDING

CRD-2

Developers create and innovate using an iterative design process that is user-focused, that incorporates implementation/feedback cycles, and that leaves ample room for experimentation and risk-taking.

LEARNING OBJECTIVE

CRD-2.E

Develop a program using a development process. **1.B**

ESSENTIAL KNOWLEDGE

CRD-2.E.1

A development process can be ordered and intentional, or exploratory in nature.

CRD-2.E.2

There are multiple development processes. The following phases are commonly used when developing a program:

- investigating and reflecting
- designing
- prototyping
- testing

CRD-2.E.3

A development process that is iterative requires refinement and revision based on feedback, testing, or reflection throughout the process. This may require revisiting earlier phases of the process.

CRD-2.E.4

A development process that is incremental is one that breaks the problem into smaller pieces and makes sure each piece works before adding it to the whole.

continued on next page

Creative Development

BIG IDEA 1

LEARNING OBJECTIVE

CRD-2.F
Design a program and its user interface. 1.B

ESSENTIAL KNOWLEDGE

CRD-2.F.1
The design of a program incorporates investigation to determine its requirements.

CRD-2.F.2
Investigation in a development process is useful for understanding and identifying the program constraints, as well as the concerns and interests of the people who will use the program.

CRD-2.F.3
Some ways investigation can be performed are as follows:

- collecting data through surveys
- user testing
- interviews
- direct observations

CRD-2.F.4
Program requirements describe how a program functions and may include a description of user interactions that a program must provide.

CRD-2.F.5
A program's specification defines the requirements for the program.

CRD-2.F.6
In a development process, the design phase outlines how to accomplish a given program specification.

CRD-2.F.7
The design phase of a program may include:

- brainstorming
- planning and storyboarding
- organizing the program into modules and functional components
- creation of diagrams that represent the layouts of the user interface
- development of a testing strategy for the program

continued on next page



BIG IDEA 1

Creative Development

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
CRD-2.G Describe the purpose of a code segment or program by writing documentation. 4.A	**CRD-2.G.1** *Program documentation* is a written description of the function of a code segment, event, procedure, or program and how it was developed. **CRD-2.G.2** *Comments* are a form of program documentation written into the program to be read by people and do not affect how a program runs. **CRD-2.G.3** Programmers should document a program throughout its development. **CRD-2.G.4** Program documentation helps in developing and maintaining correct programs when working individually or in collaborative programming environments. **CRD-2.G.5** Not all programming environments support comments, so other methods of documentation may be required.
CRD-2.H Acknowledge code segments used from other sources. 1.C	**CRD-2.H.1** It is important to acknowledge any code segments that were developed collaboratively or by another source. **CRD-2.H.2** Acknowledgement of a code segment(s) written by someone else and used in a program can be in the program documentation. The acknowledgement should include the origin or original author's name.

Creative Development

BIG IDEA 1

TOPIC 1.4
Identifying and Correcting Errors

Required Course Content

ENDURING UNDERSTANDING

CRD-2

Developers create and innovate using an iterative design process that is user-focused, that incorporates implementation/feedback cycles, and that leaves ample room for experimentation and risk-taking.

LEARNING OBJECTIVE

CRD-2.I

For errors in an algorithm or program:

a. Identify the error. 4.C

b. Correct the error. 4.C

ESSENTIAL KNOWLEDGE

CRD-2.I.1

A *logic error* is a mistake in the algorithm or program that causes it to behave incorrectly or unexpectedly.

CRD-2.I.2

A *syntax error* is a mistake in the program where the rules of the programming language are not followed.

CRD-2.I.3

A *run-time error* is a mistake in the program that occurs during the execution of a program. Programming languages define their own run-time errors.

CRD-2.I.4

An *overflow error* is an error that occurs when a computer attempts to handle a number that is outside of the defined range of values.

CRD-2.I.5

The following are effective ways to find and correct errors:

- test cases
- hand tracing
- visualizations
- debuggers
- adding extra output statement(s)

continued on next page

SKILLS

1.B
Determine and design an appropriate method or approach to achieve the purpose.

4.C
Identify and correct errors in algorithms and programs, including error discovery through testing.

AVAILABLE RESOURCE

- Professional Development > Teaching and Assessing Module: Explaining Processes

BIG IDEA 1

Creative Development

LEARNING OBJECTIVE

CRD-2.J

Identify inputs and corresponding expected outputs or behaviors that can be used to check the correctness of an algorithm or program. **4.C**

ESSENTIAL KNOWLEDGE

CRD-2.J.1

In the development process, *testing* uses defined inputs to ensure that an algorithm or program is producing the expected outcomes. Programmers use the results from testing to revise their algorithms or programs.

CRD-2.J.2

Defined inputs used to test a program should demonstrate the different expected outcomes that are at or just beyond the extremes (minimum and maximum) of input data.

CRD-2.J.3

Program requirements are needed to identify appropriate defined inputs for testing.



AP COMPUTER SCIENCE PRINCIPLES

BIG IDEA 2

Data

AP

17–22%
AP EXAM WEIGHTING

AP

Remember to go to **AP Classroom** to assign students the online Topic Questions for this big idea.

Whether assigned as homework or completed in class, the Topic Questions can be used to spot-check student understanding and help identify content and skills to emphasize in lessons.

Topic Questions

Multiple-choice: ~20 questions

BIG IDEA 2

17–22% AP EXAM WEIGHTING

Data

ESSENTIAL QUESTIONS

DAT-1

- How can we use 1s and 0s to represent something complex like a video of the marching band playing a song?

DAT-2

- How can you predict the attendance at a school event using data gathered from social media?
- When is it more appropriate to use a computer to analyze data than to complete the analysis by hand?

Developing Understanding

Because essentially everything we do with a computer is being broken down into some form of data, it is important for students to develop a breadth of understanding of how computers handle data and how students can use those same data to solve problems such as raising awareness for a cause, using census data to determine which state will gain seats in the House of Representatives, or using traffic and cost data to determine the ideal location for prom. In this big idea, students will gain a deep understanding of how information is stored on a computer in binary and seamlessly translated into what is seen on the screen or heard through speakers. Students will also learn how data are processed to learn something new. This big idea is often paired with Big Idea 3: Algorithms and Programming and Big Idea 5: Impact of Computing.

Building Computational Thinking Practices

2.B 3.C 5.B

On the end-of-course exam, students will be presented with the way data for text or media, such as color, are represented by a computer and will be asked to convert values from binary to decimal or vice versa. The idea that there are number systems other than the decimal system is often new to students. Connecting the foundational principles of how number systems operate to the decimal number system is likely to help students lean on their prior knowledge when asked to work with binary numbers.

Some real-world problems and simulations involve the use of large data sets that cannot be easily analyzed by hand and require a programming solution that manipulates or combines the data with other sources to generate new knowledge and find a solution. When working with large data sets, programmers use data abstraction to write programs that can be flexible enough to handle a change in the number of data entries. Providing students with practice using data sets that are too large to manipulate by hand will motivate them to develop more general solutions and data abstractions. Because the solution is generalized, an explanation of the solution through documentation within the program may be necessary.

Preparing for the AP Exam

Data compression algorithms are often used to maximize storage space or to transmit data over the Internet, but sometimes at a cost to the quality of the data. Students will need to compare data compression algorithms and determine which one would be best to use in a given situation. Providing students with examples of different compression algorithms along with how each might work may deepen students' understanding. Examples of how data might be restored to their uncompressed state—or may be unable to be restored—may improve students' ability to distinguish between compression algorithms.

When presented with scenarios that describe data and metadata for analysis on the end-of-course exam, students will be asked to determine what information can be found, as well as a potential programming process that can be used to extract information or modify the existing data. Students might benefit from practice identifying a problem they could solve using data, such as the best route to take to school, gathering the necessary data to analyze—either by using a public data set or developing a survey to gather the data—and then implementing a program that will manipulate the data to find an answer.



BIG IDEA 2

Data

BIG IDEA AT A GLANCE

Learning Objective	Topic	Skills	Unit/Module
DAT-1.A, DAT-1.B, DAT-1.C	**2.1 Binary Numbers**	1.D Evaluate solution options. 2.B Implement and apply an algorithm. 3.C Explain how abstraction manages complexity.	
DAT-1.D	**2.2 Data Compression**	1.D Evaluate solution options.	
DAT-2.A, DAT-2.B, DAT-2.C	**2.3 Extracting Information from Data**	5.B Explain how knowledge can be generated from data. 5.D Describe the impact of gathering data.	
DAT-2.D, DAT-2.E	**2.4 Using Programs with Data**	2.B Implement and apply an algorithm. 5.B Explain how knowledge can be generated from data.	

AP Go to **AP Classroom** to assign Topic Questions as you teach the topics in Big Idea 2. Review the results in class to identify and address any student misunderstandings.

Data

BIG IDEA 2

SAMPLE INSTRUCTIONAL ACTIVITIES

The sample activities on this page are optional and are offered to provide possible ways to incorporate instructional approaches into the classroom. They were developed in partnership with teachers from the AP community to share ways that they approach teaching some of the topics and skills in this big idea. Please refer to the Instructional Strategies section beginning on p. 132 for more examples of activities and strategies.

Activity	Topic	Sample Activity
1	2.1	**Look for a pattern** Provide students with a sentence or paragraph of compressed lossless text and a key. Have them look for patterns in their process of retrieving the original text and evaluate whether this is the best compression algorithm to use. Have them write down the patterns they see along with their evaluation and share these in a large group.
2	2.4	**Diagramming** Give students a question and a list of data. Have them diagram a process that could be used to answer the question using the data, making sure to include the input(s) of information and the output of the transformed data. Have students include an explanation of how the process represented in their diagram would work to find the solution.

Big Idea Planning Notes

Use the space below to plan your approach to the topics in this big idea. Consider what resources and instructional strategies you might want to use.



BIG IDEA 2

Data

SKILLS

1.D
Evaluate solution options.

2.B
Implement and apply an algorithm.

3.C
Explain how abstraction manages complexity.

AVAILABLE RESOURCES

- External Resources >
  - **Binary Numbers** from CS Unplugged
  - **Blown to Bits: Chapter 1**

TOPIC 2.1
Binary Numbers

Required Course Content

ENDURING UNDERSTANDING

DAT-1

The way a computer represents data internally is different from the way the data are interpreted and displayed for the user. Programs are used to translate data into a representation more easily understood by people.

LEARNING OBJECTIVE

DAT-1.A

Explain how data can be represented using bits. 3.C

ESSENTIAL KNOWLEDGE

DAT-1.A.1

Data values can be stored in variables, lists of items, or standalone constants and can be passed as input to (or output from) procedures.

DAT-1.A.2

Computing devices represent data digitally, meaning that the lowest-level components of any value are bits.

DAT-1.A.3

Bit is shorthand for *binary digit* and is either 0 or 1.

DAT-1.A.4

A *byte* is 8 bits.

DAT-1.A.5

Abstraction is the process of reducing complexity by focusing on the main idea. By hiding details irrelevant to the question at hand and bringing together related and useful details, abstraction reduces complexity and allows one to focus on the idea.

DAT-1.A.6

Bits are grouped to represent abstractions. These abstractions include, but are not limited to, numbers, characters, and color.

continued on next page



Data

BIG IDEA 2

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
DAT-1.A Explain how data can be represented using bits. 3.C	**DAT-1.A.7** The same sequence of bits may represent different types of data in different contexts. **DAT-1.A.8** Analog data have values that change smoothly, rather than in discrete intervals, over time. Some examples of analog data include pitch and volume of music, colors of a painting, or position of a sprinter during a race. **DAT-1.A.9** The use of digital data to approximate real-world analog data is an example of abstraction. **DAT-1.A.10** Analog data can be closely approximated digitally using a *sampling technique*, which means measuring values of the analog signal at regular intervals called *samples*. The samples are measured to figure out the exact bits required to store each sample.
DAT-1.B Explain the consequences of using bits to represent data. 1.D	**DAT-1.B.1** In many programming languages, integers are represented by a fixed number of bits, which limits the range of integer values and mathematical operations on those values. This limitation can result in overflow or other errors. **DAT-1.B.2** Other programming languages provide an abstraction through which the size of representable integers is limited only by the size of the computer's memory; this is the case for the language defined in the exam reference sheet. **DAT-1.B.3** In programming languages, the fixed number of bits used to represent real numbers limits the range and mathematical operations on these values; this limitation can result in round-off and other errors. Some real numbers are represented as approximations in computer storage. **EXCLUSION STATEMENT (EK DAT-1.B.3):** *Specific range limitations for real numbers are outside the scope of this course and the AP Exam.*

continued on next page



BIG IDEA 2

Data

LEARNING OBJECTIVE

DAT-1.C

For binary numbers:

a. Calculate the binary (base 2) equivalent of a positive integer (base 10) and vice versa. 2.B

b. Compare and order binary numbers. 2.B

ESSENTIAL KNOWLEDGE

DAT-1.C.1

Number bases, including binary and decimal, are used to represent data.

DAT-1.C.2

Binary (base 2) uses only combinations of the digits zero and one.

DAT-1.C.3

Decimal (base 10) uses only combinations of the digits 0 – 9.

DAT-1.C.4

As with decimal, a digit's position in the binary sequence determines its numeric value. The numeric value is equal to the bit's value (0 or 1) multiplied by the place value of its position.

DAT-1.C.5

The place value of each position is determined by the base raised to the power of the position. Positions are numbered starting at the rightmost position with 0 and increasing by 1 for each subsequent position to the left.

Data

BIG IDEA 2

TOPIC 2.2
Data Compression

SKILL
1.D
Evaluate solution options.

AVAILABLE RESOURCE
- External Resource > Text Compression from CS Unplugged

Required Course Content

ENDURING UNDERSTANDING

DAT-1
The way a computer represents data internally is different from the way the data are interpreted and displayed for the user. Programs are used to translate data into a representation more easily understood by people.

LEARNING OBJECTIVE

DAT-1.D
Compare data compression algorithms to determine which is best in a particular context. 1.D

ESSENTIAL KNOWLEDGE

DAT-1.D.1
Data compression can reduce the size (number of bits) of transmitted or stored data.

DAT-1.D.2
Fewer bits does not necessarily mean less information.

DAT-1.D.3
The amount of size reduction from compression depends on both the amount of redundancy in the original data representation and the compression algorithm applied.

DAT-1.D.4
Lossless data compression algorithms can usually reduce the number of bits stored or transmitted while guaranteeing complete reconstruction of the original data.

DAT-1.D.5
Lossy data compression algorithms can significantly reduce the number of bits stored or transmitted but only allow reconstruction of an approximation of the original data.

continued on next page



BIG IDEA 2

Data

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
DAT-1.D Compare data compression algorithms to determine which is best in a particular context. 1.D	**DAT-1.D.6** Lossy data compression algorithms can usually reduce the number of bits stored or transmitted more than lossless compression algorithms. **DAT-1.D.7** In situations where quality or ability to reconstruct the original is maximally important, lossless compression algorithms are typically chosen. **DAT-1.D.8** In situations where minimizing data size or transmission time is maximally important, lossy compression algorithms are typically chosen.

Data

BIG IDEA 2

TOPIC 2.3
Extracting Information from Data

SKILLS

5.B
Explain how knowledge can be generated from data.

5.D
Describe the impact of gathering data.

Required Course Content

ENDURING UNDERSTANDING

DAT-2
Programs can be used to process data, which allows users to discover information and create new knowledge.

LEARNING OBJECTIVE

DAT-2.A
Describe what information can be extracted from data.
5.B

ESSENTIAL KNOWLEDGE

DAT-2.A.1
Information is the collection of facts and patterns extracted from data.

DAT-2.A.2
Data provide opportunities for identifying trends, making connections, and addressing problems.

DAT-2.A.3
Digitally processed data may show correlation between variables. A correlation found in data does not necessarily indicate that a causal relationship exists. Additional research is needed to understand the exact nature of the relationship.

DAT-2.A.4
Often, a single source does not contain the data needed to draw a conclusion. It may be necessary to combine data from a variety of sources to formulate a conclusion.

continued on next page



BIG IDEA 2

Data

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
DAT-2.B Describe what information can be extracted from metadata. 5.B	**DAT-2.B.1** *Metadata* are data about data. For example, the piece of *data* may be an image, while the *metadata* may include the date of creation or the file size of the image. **DAT-2.B.2** Changes and deletions made to metadata do not change the primary data. **DAT-2.B.3** Metadata are used for finding, organizing, and managing information. **DAT-2.B.4** Metadata can increase the effective use of data or data sets by providing additional information. **DAT-2.B.5** Metadata allow data to be structured and organized.
DAT-2.C Identify the challenges associated with processing data. 5.D	**DAT-2.C.1** The ability to process data depends on the capabilities of the users and their tools. **DAT-2.C.2** Data sets pose challenges regardless of size, such as: ▪ the need to clean data ▪ incomplete data ▪ invalid data ▪ the need to combine data sources **DAT-2.C.3** Depending on how data were collected, they may not be uniform. For example, if users enter data into an open field, the way they choose to abbreviate, spell, or capitalize something may vary from user to user. **DAT-2.C.4** *Cleaning data* is a process that makes the data uniform without changing their meaning (e.g., replacing all equivalent abbreviations, spellings, and capitalizations with the same word). **DAT-2.C.5** Problems of bias are often created by the type or source of data being collected. Bias is not eliminated by simply collecting more data.

continued on next page

Data

BIG IDEA 2

LEARNING OBJECTIVE

DAT-2.C
Identify the challenges associated with processing data. 5.D

ESSENTIAL KNOWLEDGE

DAT-2.C.6
The size of a data set affects the amount of information that can be extracted from it.

DAT-2.C.7
Large data sets are difficult to process using a single computer and may require parallel systems.

DAT-2.C.8
Scalability of systems is an important consideration when working with data sets, as the computational capacity of a system affects how data sets can be processed and stored.

BIG IDEA 2

Data

SKILLS

2.B
Implement and apply an algorithm.

5.B
Explain how knowledge can be generated from data.

TOPIC 2.4
Using Programs with Data

Required Course Content

ENDURING UNDERSTANDING

DAT-2
Programs can be used to process data, which allows users to discover information and create new knowledge.

LEARNING OBJECTIVE

DAT-2.D
Extract information from data using a program. 2.B

ESSENTIAL KNOWLEDGE

DAT-2.D.1
Programs can be used to process data to acquire information.

DAT-2.D.2
Tables, diagrams, text, and other visual tools can be used to communicate insight and knowledge gained from data.

DAT-2.D.3
Search tools are useful for efficiently finding information.

DAT-2.D.4
Data filtering systems are important tools for finding information and recognizing patterns in data.

DAT-2.D.5
Programs such as spreadsheets help efficiently organize and find trends in information.

continued on next page



Data

BIG IDEA 2

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
DAT-2.D Extract information from data using a program. 2.B	**DAT-2.D.6** Some processes that can be used to extract or modify information from data include the following: ▪ transforming every element of a data set, such as doubling every element in a list, or adding a parent's email to every student record ▪ filtering a data set, such as keeping only the positive numbers from a list, or keeping only students who signed up for band from a record of all the students ▪ combining or comparing data in some way, such as adding up a list of numbers, or finding the student who has the highest GPA ▪ visualizing a data set through a chart, graph, or other visual representation
DAT-2.E Explain how programs can be used to gain insight and knowledge from data. 5.B	**DAT-2.E.1** Programs are used in an iterative and interactive way when processing information to allow users to gain insight and knowledge about data. **DAT-2.E.2** Programmers can use programs to filter and clean digital data, thereby gaining insight and knowledge. **DAT-2.E.3** Combining data sources, clustering data, and classifying data are parts of the process of using programs to gain insight and knowledge from data. **DAT-2.E.4** Insight and knowledge can be obtained from translating and transforming digitally represented information. **DAT-2.E.5** Patterns can emerge when data are transformed using programs.

THIS PAGE IS INTENTIONALLY LEFT BLANK.



AP COMPUTER SCIENCE PRINCIPLES

BIG IDEA 3

Algorithms and Programming

AP

30–35%
AP EXAM WEIGHTING

AP

Remember to go to **AP Classroom** to assign students the online Topic Questions for this big idea.

Whether assigned as homework or completed in class, the Topic Questions can be used to spot-check student understanding and help identify content and skills to emphasize in lessons.

Topic Questions

Multiple-choice: ~90 questions
Performance Tasks: ~20 prompts

BIG IDEA 3

30–35% AP EXAM WEIGHTING

Algorithms and Programming

ESSENTIAL QUESTIONS

AAP-1
- How can we store data in a program to solve problems?

AAP-2
- What might happen if you completed the steps in your regular morning routine to get ready and go to school in a different order? How might the reordering affect the decisions you make each morning?

AAP-3
- How do video games group the different actions for a player based on what key is pressed on the keyboard or controller? How do apps group different actions together based on user interaction, such as pressing buttons?

AAP-4
- What types of problems can be solved more easily with a computer, and what types can be solved more easily without a computer? Why?

Developing Understanding

All programming languages, whether block-based or text-based, use similar programming structures and commands. Having a basic understanding of how these building blocks are combined to form algorithms and abstractions in one language makes it easier to apply these same understandings to other programming languages. This big idea focuses on determining the efficiency of algorithms, as well as writing and implementing algorithms in a program. This big idea can be paired with any of the other big ideas and taught throughout the school year.

Building Computational Thinking Practices

1.A 2.A 2.B 3.B 4.B

Some problems are so simple that writing a program would be more time-consuming than accomplishing the task by hand, while others are so complicated that it would take a computer an unreasonable amount of time to solve, if it even is possible. It is important for students to understand that not all problems can or should be solved using a program.

In procedural abstraction, programmers group program code into a procedure to make the code more readable and reusable. This type of abstraction often means using parameters to make the procedure more general and allow it to be called multiple times with varying inputs. One way to help students create abstractions is to have them write multiple versions of a procedure that each accomplish a specific instance of a task. After identifying the commonalities, students may find it easier to write a more abstract, general procedure that uses parameters to accommodate the differences.

In data abstraction, programmers use a list as a representation of something else, such as a grocery list or seating chart. When students explain how their abstractions manage complexity, they should use the context of their specific program to support why the abstraction is necessary or helpful, rather than writing in general about how abstraction manages program complexity. One way to do this is to consider how the program would be written differently, or if it could be written at all, without the use of the abstraction.

Preparing for the AP Exam

On the AP Exam, students will be asked to determine the result or functionality of code. Students can practice analyzing code segments by first predicting the result of provided code and then checking their hypothesis by running the program on a computer.

Because the Create performance task requires students to create their own program, they will need scaffolded practice that moves them from representing algorithmic processes through diagrams or pseudocode, to writing simple programs that perform calculations, to more complex programs that leverage abstraction. While students can collaborate on the development of the program, each student should be a contributing member of the program development. When completing the written responses, which must be completed independently, an in-depth understanding of how the code segments function will allow each student to answer the prompts on their own.



BIG IDEA 3

Algorithms and Programming

BIG IDEA AT A GLANCE

Learning Objectives	Topic	Skills	Unit/Module
AAP-1-A, AAP-1.B	**3.1** Variables and Assignments	3.A Generalize data sources through variables. 4.B Determine the result of code segments.	
AAP-1.C, AAP-1.D	**3.2** Data Abstraction	3.A Generalize data sources through variables. 3.B Use abstraction to manage complexity in a program. 3.C Explain how abstraction manages complexity.	
AAP-2.A, AAP-2.B, AAP-2.C	**3.3** Mathematical Expressions	2.A Represent algorithmic processes without using a programming language. 2.B Implement and apply an algorithm. 4.B Determine the result of code segments.	
AAP-2.D	**3.4** Strings	4.B Determine the result of code segments.	
AAP-2.E, AAP-2.F	**3.5** Boolean Expressions	2.B Implement and apply an algorithm. 4.B Determine the result of code segments.	
AAP-2.G, AAP-2.H	**3.6** Conditionals	2.A Represent algorithmic processes without using a programming language. 2.B Implement and apply an algorithm. 4.B Determine the result of code segments.	
AAP-2.I	**3.7** Nested Conditionals	2.B Implement and apply an algorithm. 4.B Determine the result of code segments.	
AAP-2.J, AAP-2.K	**3.8** Iteration	2.A Represent algorithmic processes without using a programming language. 2.B Implement and apply an algorithm. 4.B Determine the result of code segments.	
AAP-2.L, AAP-2.M	**3.9** Developing Algorithms	1.D Evaluate solution options. 2.A Represent algorithmic processes without using a programming language. 2.B Implement and apply an algorithm.	

continued on next page

BIG IDEA AT A GLANCE *(cont'd)*

Learning Objectives	Topic	Skills	Unit/Module
AAP-2.N, AAP-2.O	**3.10** Lists	2.B Implement and apply an algorithm. 4.B Determine the result of code segments.	
AAP-2.P	**3.11** Binary Search	1.A Investigate the situation, context, or task. 1.D Evaluate solution options.	
AAP-3.A	**3.12** Calling Procedures	3.B Use abstraction to manage complexity in a program. 4.B Determine the result of code segments.	
AAP-3.B, AAP-3.C	**3.13** Developing Procedures	3.B Use abstraction to manage complexity in a program. 3.C Explain how abstraction manages complexity.	
AAP-3.D	**3.14** Libraries	2.B Implement and apply an algorithm.	
AAP-3.E	**3.15** Random Values	2.B Implement and apply an algorithm. 4.B Determine the result of code segments.	
AAP-3.F	**3.16** Simulations	1.A Investigate the situation, context, or task. 1.D Evaluate solution options.	
AAP-4.A	**3.17** Algorithmic Efficiency	1.D Evaluate solution options.	
AAP-4.B	**3.18** Undecidable Problems	1.A Investigate the situation, context, or task.	

Go to **AP Classroom** to assign Topic Questions as you teach the topics in Big Idea 3. Review the results in class to identify and address any student misunderstandings.

BIG IDEA 3

Algorithms and Programming

SAMPLE INSTRUCTIONAL ACTIVITIES

The sample activities on this page are optional and are offered to provide possible ways to incorporate instructional approaches into the classroom. They were developed in partnership with teachers from the AP community to share ways that they approach teaching some of the topics and skills in this big idea. Please refer to the Instructional Strategies section beginning on p. 132 for more examples of activities and strategies.

Activity	Topic	Sample Activity
1	3.1	**Predict and compare** Provide students with a list of expressions with assignments. Ask them to predict the value of each variable after the assignment and then compare their answers to the output produced when these statements are put into a program.
2	3.6	**Using manipulatives** When learning about conditionals, take a printout of a simple conditional statement and cut it into multiple sections. As students enter the classroom, hand them an envelope full of the paper strips, and ask them to reassemble the conditional in the proper order.
3	3.12	**Marking the text** Provide students with program code that draws a square of side length 10 and a separate set of program code that uses side length 100. Ask students to mark up the sets of code to identify where they are different and to create a generalization by using parameters. Ask them to write a procedure that uses parameters to draw a square of any size.
4	3.18	**Think-pair-share** Have students work in pairs to consider what factors would be the most important to prioritize in writing an algorithm to build the perfect master schedule for the school. Some considerations may include maximum class size, student preferences, and teacher availability. Have the pairs discuss and then report their results. Finally, discuss as a class how such programs may have to settle for a "good enough" solution when an exact solution may not be possible in a reasonable amount of time.

Big Idea Planning Notes

Use the space below to plan your approach to the topics in this big idea. Consider what resources and instructional strategies you might want to use.

Algorithms and Programming

BIG IDEA 3

TOPIC 3.1
Variables and Assignments

Required Course Content

ENDURING UNDERSTANDING

AAP-1

To find specific solutions to generalizable problems, programmers represent and organize data in multiple ways.

LEARNING OBJECTIVE

AAP-1.A

Represent a value with a variable. 3.A

ESSENTIAL KNOWLEDGE

AAP-1.A.1

A *variable* is an abstraction inside a program that can hold a value. Each variable has associated data storage that represents one value at a time, but that value can be a list or other collection that in turn contains multiple values.

AAP-1.A.2

Using meaningful variable names helps with the readability of program code and understanding of what values are represented by the variables.

AAP-1.A.3

Some programming languages provide *types* to represent data, which are referenced using variables. These types include numbers, Booleans, lists, and strings.

AAP-1.A.4

Some values are better suited to representation using one type of datum rather than another.

continued on next page

SKILLS

3.A
Generalize data sources through variables.

4.B
Determine the result of code segments.

AVAILABLE RESOURCE

- AP CSP Exam Reference Sheet (see Appendix)



BIG IDEA 3

Algorithms and Programming

LEARNING OBJECTIVE

AAP-1.B

Determine the value of a variable as a result of an assignment. **4.B**

ESSENTIAL KNOWLEDGE

AAP-1.B.1

The assignment operator allows a program to change the value represented by a variable.

AAP-1.B.2

The exam reference sheet provides the "←" operator to use for assignment. For example,

Text:

```
a ← expression
```

Block:



evaluates `expression` and then assigns a copy of the result to the variable `a`.

AAP-1.B.3

The value stored in a variable will be the most recent value assigned. For example:

```
a ← 1
b ← a
a ← 2
display(b)
```

still displays `1`.

Algorithms and Programming

BIG IDEA 3

TOPIC 3.2
Data Abstraction

SKILLS

3.A Generalize data sources through variables.

3.B Use abstraction to manage complexity in a program.

3.C Explain how abstraction manages complexity.

AVAILABLE RESOURCE

- AP CSP Exam Reference Sheet (see Appendix)

Required Course Content

ENDURING UNDERSTANDING

AAP-1

To find specific solutions to generalizable problems, programmers represent and organize data in multiple ways.

LEARNING OBJECTIVE

AAP-1.C

Represent a list or string using a variable. **3.A**

ESSENTIAL KNOWLEDGE

AAP-1.C.1

A *list* is an ordered sequence of elements. For example,

```
[value1, value2, value3, ...]
```

describes a list where `value1` is the first element, `value2` is the second element, `value3` is the third element, and so on.

AAP-1.C.2

An *element* is an individual value in a list that is assigned a unique index.

AAP-1.C.3

An *index* is a common method for referencing the elements in a list or string using natural numbers.

AAP-1.C.4

A *string* is an ordered sequence of characters.

continued on next page



BIG IDEA 3

Algorithms and Programming

LEARNING OBJECTIVE

AAP-1.D

For data abstraction:

a. Develop data abstraction using lists to store multiple elements. 3.B

b. Explain how the use of data abstraction manages complexity in program code. 3.C

ESSENTIAL KNOWLEDGE

AAP-1.D.1

Data abstraction provides a separation between the abstract properties of a data type and the concrete details of its representation.

AAP-1.D.2

Data abstractions manage complexity in programs by giving a collection of data a name without referencing the specific details of the representation.

AAP-1.D.3

Data abstractions can be created using lists.

AAP-1.D.4

Developing a data abstraction to implement in a program can result in a program that is easier to develop and maintain.

AAP-1.D.5

Data abstractions often contain different types of elements.

AAP-1.D.6

The use of lists allows multiple related items to be treated as a single value. Lists are referred to by different names, such as *array*, depending on the programming language.

EXCLUSION STATEMENT (EK APP-1.D.6):
The use of linked lists is outside the scope of this course and the AP Exam.

AAP-1.D.7

The exam reference sheet provides the notation

```
[value1, value2, value3, ...]
```

to create a list with those values as the first, second, third, and so on items. For example,

- Text:

  ```
  aList ← [value1, value2,
  value3, ...]
  ```

  Block:

  creates a new list that contains the values `value1`, `value2`, `value3`, and `...` at indices `1`, `2`, `3`, and `...` respectively and assigns it to `aList`.

- Text:

  ```
  aList ← []
  ```

  Block:

  creates a new empty list and assigns it to `aList`.

continued on next page

Algorithms and Programming

BIG IDEA 3

LEARNING OBJECTIVE

AAP-1.D

For data abstraction:

a. Develop data abstraction using lists to store multiple elements. 3.B

b. Explain how the use of data abstraction manages complexity in program code. 3.C

ESSENTIAL KNOWLEDGE

- Text:

  `aList ← bList`

  Block:

  `aList ← bList`

  assigns a copy of the list `bList` to the list `aList`. For example, if `bList` contains `[20, 40, 60]`, then `aList` will also contain `[20, 40, 60]` after the assignment.

AAP-1.D.8

The exam reference sheet describes a list structure whose index values are 1 through the number of elements in the list, inclusive. For all list operations, if a list index is less than 1 or greater than the length of the list, an error message is produced and the program will terminate.

BIG IDEA 3

Algorithms and Programming

SKILLS

2.A Represent algorithmic processes without using a programming language.

2.B Implement and apply an algorithm.

4.B Determine the result of code segments.

AVAILABLE RESOURCE

- AP CSP Exam Reference Sheet (see Appendix)

TOPIC 3.3 Mathematical Expressions

Required Course Content

ENDURING UNDERSTANDING

AAP-2

The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
AAP-2.A Express an algorithm that uses sequencing without using a programming language. **2.A**	**AAP-2.A.1** An *algorithm* is a finite set of instructions that accomplish a specific task.
	AAP-2.A.2 Beyond visual and textual programming languages, algorithms can be expressed in a variety of ways, such as natural language, diagrams, and pseudocode.
	AAP-2.A.3 Algorithms executed by programs are implemented using programming languages.
	AAP-2.A.4 Every algorithm can be constructed using combinations of sequencing, selection, and iteration.
AAP-2.B Represent a step-by-step algorithmic process using sequential code statements. **2.B**	**AAP-2.B.1** *Sequencing* is the application of each step of an algorithm in the order in which the code statements are given.
	AAP-2.B.2 A *code statement* is a part of program code that expresses an action to be carried out.

continued on next page



Algorithms and Programming

BIG IDEA 3

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
AAP-2.B Represent a step-by-step algorithmic process using sequential code statements. 2.B	**AAP-2.B.3** An *expression* can consist of a value, a variable, an operator, or a procedure call that returns a value. **AAP-2.B.4** Expressions are evaluated to produce a single value. **AAP-2.B.5** The evaluation of expressions follows a set order of operations defined by the programming language. **AAP-2.B.6** Sequential statements execute in the order they appear in the code segment. **AAP-2.B.7** Clarity and readability are important considerations when expressing an algorithm in a programming language.
AAP-2.C Evaluate expressions that use arithmetic operators. 4.B	**AAP-2.C.1** Arithmetic operators are part of most programming languages and include addition, subtraction, multiplication, division, and modulus operators. **AAP-2.C.2** The exam reference sheet provides `a MOD b`, which evaluates to the remainder when `a` is divided by `b`. Assume that `a` is an integer greater than or equal to `0` and `b` is an integer greater than `0`. For example, `17 MOD 5` evaluates to `2`. **AAP-2.C.3** The exam reference sheet provides the arithmetic operators `+`, `–`, `*`, `/`, and `MOD`. Text and Block: ▪ `a + b` ▪ `a – b` ▪ `a * b` ▪ `a / b` ▪ `a MOD b` These are used to perform arithmetic on `a` and `b`. For example, `17 / 5` evaluates to `3.4`. **AAP-2.C.4** The order of operations used in mathematics applies when evaluating expressions. The `MOD` operator has the same precedence as the `*` and `/` operators.



BIG IDEA 3

Algorithms and Programming

SKILL
4.B
Determine the result of code segments.

TOPIC 3.4
Strings

Required Course Content

ENDURING UNDERSTANDING

AAP-2

The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.

LEARNING OBJECTIVE

AAP-2.D

Evaluate expressions that manipulate strings. 4.B

ESSENTIAL KNOWLEDGE

AAP-2.D.1

String concatenation joins together two or more strings end-to-end to make a new string.

AAP-2.D.2

A *substring* is part of an existing string.

Algorithms and Programming

BIG IDEA 3

TOPIC 3.5
Boolean Expressions

SKILLS

2.B
Implement and apply an algorithm.

4.B
Determine the result of code segments.

AVAILABLE RESOURCE

- AP CSP Exam Reference Sheet (see Appendix)

Required Course Content

ENDURING UNDERSTANDING

AAP-2

The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.

LEARNING OBJECTIVE

AAP-2.E

For relationships between two variables, expressions, or values:

a. Write expressions using relational operators. 2.B

b. Evaluate expressions that use relational operators. 4.B

ESSENTIAL KNOWLEDGE

AAP-2.E.1

A *Boolean value* is either true or false.

AAP-2.E.2

The exam reference sheet provides the following relational operators: =, ≠, >, <, ≥, and ≤.

Text and Block:

- `a = b`
- `a ≠ b`
- `a > b`
- `a < b`
- `a ≥ b`
- `a ≤ b`

These are used to test the relationship between two variables, expressions, or values. A comparison using a relational operator evaluates to a Boolean value. For example, `a = b` evaluates to `true` if `a` and `b` are equal; otherwise, it evaluates to `false`.

continued on next page



BIG IDEA 3

Algorithms and Programming

LEARNING OBJECTIVE

AAP-2.F

For relationships between Boolean values:

a. Write expressions using logical operators. 2.B

b. Evaluate expressions that use logic operators. 4.B

ESSENTIAL KNOWLEDGE

AAP-2.F.1

The exam reference sheet provides the logical operators `NOT`, `AND`, and `OR`, which evaluate to a Boolean value.

AAP-2.F.2

The exam reference sheet provides

Text:

```
NOT condition
```

Block:

NOT (condition)

which evaluates to `true` if `condition` is `false`; otherwise it evaluates to `false`.

AAP-2.F.3

The exam reference sheet provides

Text:

```
condition1 AND condition2
```

Block:

(condition1) AND (condition2)

which evaluates to `true` if both `condition1` and `condition2` are `true`; otherwise it evaluates to `false`.

AAP-2.F.4

The exam reference sheet provides

Text:

```
condition1 OR condition2
```

Block:

(condition1) OR (condition2)

which evaluates to `true` if `condition1` is `true` or if `condition2` is `true` or if both `condition1` and `condition2` are `true`; otherwise it evaluates to `false`.

AAP-2.F.5

The operand for a logical operator is either a Boolean expression or a single Boolean value.



Algorithms and Programming

BIG IDEA 3

TOPIC 3.6
Conditionals

Required Course Content

ENDURING UNDERSTANDING

AAP-2

The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
AAP-2.G Express an algorithm that uses selection without using a programming language. 2.A	AAP-2.G.1 *Selection* determines which parts of an algorithm are executed based on a condition being `true` or `false`.
AAP-2.H For selection: a. Write conditional statements. 2.B b. Determine the result of conditional statements. 4.B	AAP-2.H.1 Conditional statements, or "if-statements," affect the sequential flow of control by executing different statements based on the value of a Boolean expression. AAP-2.H.2 The exam reference sheet provides Text: `IF(condition)` `{` `  <block of statements>` `}` Block: IF condition block of statements

continued on next page

SKILLS

2.A
Represent algorithmic processes without using a programming language.

2.B
Implement and apply an algorithm.

4.B
Determine the result of code segments.

AVAILABLE RESOURCES

- Professional Development > Teaching and Assessing: Algorithmic Thinking
- AP CSP Exam Reference Sheet (see Appendix)

BIG IDEA 3

Algorithms and Programming

LEARNING OBJECTIVE

AAP-2.H

For selection:

a. Write conditional statements. 2.B

b. Determine the result of conditional statements. 4.B

ESSENTIAL KNOWLEDGE

in which the code in `block of statements` is executed if the Boolean expression `condition` evaluates to `true`; no action is taken if `condition` evaluates to `false`.

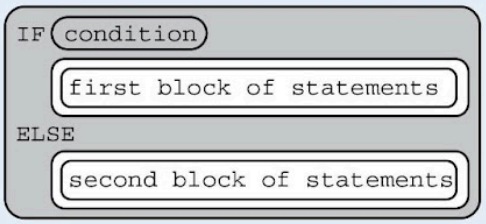
AAP-2.H.3

The exam reference sheet provides

Text:

```
IF(condition)
{
   <first block of statements>
}
ELSE
{
   <second block of statements>
}
```

Block:



in which the code in `first block of statements` is executed if the Boolean expression `condition` evaluates to `true`; otherwise, the code in `second block of statements` is executed.

Algorithms and Programming

BIG IDEA 3

TOPIC 3.7
Nested Conditionals

SKILLS

2.B
Implement and apply an algorithm.

4.B
Determine the result of code segments.

Required Course Content

ENDURING UNDERSTANDING

AAP-2

The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.

LEARNING OBJECTIVE

AAP-2.I

For nested selection:

a. Write nested conditional statements. 2.B

b. Determine the result of nested conditional statements. 4.B

ESSENTIAL KNOWLEDGE

AAP-2.I.1

Nested conditional statements consist of conditional statements within conditional statements.



BIG IDEA 3

Algorithms and Programming

SKILLS

2.A Represent algorithmic processes without using a programming language.

2.B Implement and apply an algorithm.

4.B Determine the result of code segments.

AVAILABLE RESOURCES

- Professional Development > Teaching and Assessing: Algorithmic Thinking
- AP CSP Exam Reference Sheet (see Appendix)

TOPIC 3.8
Iteration

Required Course Content

ENDURING UNDERSTANDING

AAP-2

The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.

LEARNING OBJECTIVE

AAP-2.J

Express an algorithm that uses iteration without using a programming language. **2.A**

AAP-2.K

For iteration:

a. Write iteration statements. **2.B**

b. Determine the result or side effect of iteration statements. **4.B**

ESSENTIAL KNOWLEDGE

AAP-2.J.1

Iteration is a repeating portion of an algorithm. Iteration repeats a specified number of times or until a given condition is met.

AAP-2.K.1

Iteration statements change the sequential flow of control by repeating a set of statements zero or more times, until a stopping condition is met.

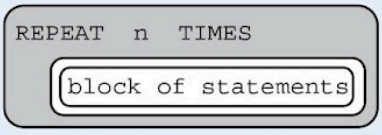
AAP-2.K.2

The exam reference sheet provides

Text:

```
REPEAT n TIMES
{
   <block of statements>
}
```

Block:



in which the `block of statements` is executed `n` times.

continued on next page

Algorithms and Programming

BIG IDEA 3

LEARNING OBJECTIVE

AAP-2.K

For iteration:

a. Write iteration statements. 2.B

b. Determine the result or side effect of iteration statements. 4.B

ESSENTIAL KNOWLEDGE

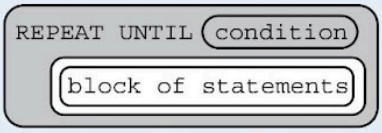
AAP-2.K.3

The exam reference sheet provides

Text:

```
REPEAT UNTIL(condition)
{
   <block of statements>
}
```

Block:



in which the code in `block of statements` is repeated until the Boolean expression `condition` evaluates to `true`.

AAP-2.K.4

In `REPEAT UNTIL(condition)` iteration, an infinite loop occurs when the ending condition will never evaluate to `true`.

AAP-2.K.5

In `REPEAT UNTIL(condition)` iteration, if the conditional evaluates to `true` initially, the loop body is not executed at all, due to the condition being checked before the loop.

BIG IDEA 3

Algorithms and Programming

SKILLS

1.D Evaluate solution options.

2.A Represent algorithmic processes without using a programming language.

2.B Implement and apply an algorithm.

TOPIC 3.9
Developing Algorithms

Required Course Content

ENDURING UNDERSTANDING

AAP-2

The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.

LEARNING OBJECTIVE

AAP-2.L

Compare multiple algorithms to determine if they yield the same side effect or result. **1.D**

ESSENTIAL KNOWLEDGE

AAP-2.L.1
Algorithms can be written in different ways and still accomplish the same tasks.

AAP-2.L.2
Algorithms that appear similar can yield different side effects or results.

AAP-2.L.3
Some conditional statements can be written as equivalent Boolean expressions.

AAP-2.L.4
Some Boolean expressions can be written as equivalent conditional statements.

AAP-2.L.5
Different algorithms can be developed or used to solve the same problem.

continued on next page

Algorithms and Programming

BIG IDEA 3

LEARNING OBJECTIVE

AAP-2.M

For algorithms:

a. Create algorithms. **2.A**

b. Combine and modify existing algorithms. **2.B**

ESSENTIAL KNOWLEDGE

AAP-2.M.1

Algorithms can be created from an idea, by combining existing algorithms, or by modifying existing algorithms.

AAP-2.M.2

Knowledge of existing algorithms can help in constructing new ones. Some existing algorithms include:

- determining the maximum or minimum value of two or more numbers
- computing the sum or average of two or more numbers
- identifying if an integer is or is not evenly divisible by another integer
- determining a robot's path through a maze

AAP-2.M.3

Using existing correct algorithms as building blocks for constructing another algorithm has benefits such as reducing development time, reducing testing, and simplifying the identification of errors.

BIG IDEA 3

Algorithms and Programming

SKILLS

2.B
Implement and apply an algorithm.

4.B
Determine the result of code segments.

AVAILABLE RESOURCE
- AP CSP Exam Reference Sheet (see Appendix)

TOPIC 3.10
Lists

Required Course Content

ENDURING UNDERSTANDING

AAP-2

The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.

LEARNING OBJECTIVE

AAP-2.N

For list operations:

a. Write expressions that use list indexing and list procedures. 2.B

b. Evaluate expressions that use list indexing and list procedures. 4.B

ESSENTIAL KNOWLEDGE

AAP-2.N.1

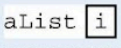
The exam reference sheet provides basic operations on lists, including:

- accessing an element by index

  Text:

  `aList[i]`

  Block:

  accesses the element of `aList` at index `i`. The first element of `aList` is at index `1` and is accessed using the notation `aList[1]`.

- assigning a value of an element of a list to a variable

  Text:

  `x ← aList [i]`

  Block:

  assigns the value of `aList[i]` to the variable `x`.

- assigning a value to an element of a list

  Text:

  `aList[i] ← x`

  Block:

  assigns the value of `x` to `aList[i]`.

Algorithms and Programming

BIG IDEA 3

LEARNING OBJECTIVE

AAP-2.N

For list operations:

a. Write expressions that use list indexing and list procedures. 2.B

b. Evaluate expressions that use list indexing and list procedures. 4.B

ESSENTIAL KNOWLEDGE

Text:

```
aList[i] ← aList[j]
```

Block:



assigns the value of `aList[j]` to `aList[i]`.

- inserting elements at a given index

Text:

```
INSERT(aList, i, value)
```

Block:



shifts to the right any values in `aList` at indices greater than or equal to `i`. The length of the list is increased by 1, and `value` is placed at index `i` in `aList`.

- adding elements to the end of the list

Text:

```
APPEND(aList, value)
```

Block:



increases the length of `aList` by 1, and `value` is placed at the end of `aList`.

- removing elements

Text:

```
REMOVE(aList, i)
```

Block:



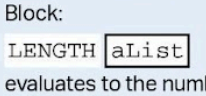
removes the item at index `i` in `aList` and shifts to the left any values at indices greater than `i`. The length of `aList` is decreased by 1.

- determining the length of a list

Text:

```
LENGTH(aList)
```

Block:



evaluates to the number of elements currently in `aList`.

AAP-2.N.2

List procedures are implemented in accordance with the syntax rules of the programming language.

continued on next page



BIG IDEA 3

Algorithms and Programming

LEARNING OBJECTIVE

AAP-2.0

For algorithms involving elements of a list:

a. Write iteration statements to traverse a list. 2.B

b. Determine the result of an algorithm that includes list traversals. 4.B

ESSENTIAL KNOWLEDGE

AAP-2.0.1

Traversing a list can be a complete traversal, where all elements in the list are accessed, or a partial traversal, where only a portion of elements are accessed.

EXCLUSION STATEMENT (EK AAP-2.0.1):
Traversing multiple lists at the same time using the same index for both (parallel traversals) is outside the scope of this course and the AP Exam.

AAP-2.0.2

Iteration statements can be used to traverse a list.

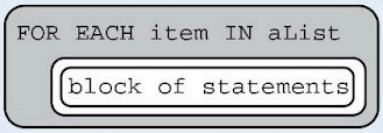
AAP-2.0.3

The exam reference sheet provides

Text:

```
FOR EACH item IN aList
{
 <block of statements>
}
```

Block:



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 assignment of `item`.

AAP-2.0.4

Knowledge of existing algorithms that use iteration can help in constructing new algorithms. Some examples of existing algorithms that are often used with lists include:

- determining a minimum or maximum value in a list
- computing a sum or average of a list of numbers

AAP-2.0.5

Linear search or sequential search algorithms check each element of a list, in order, until the desired value is found or all elements in the list have been checked.



Algorithms and Programming

BIG IDEA 3

TOPIC 3.11
Binary Search

SKILLS

1.A
Investigate the situation, context, or task.

1.D
Evaluate solution options.

AVAILABLE RESOURCES

- Classroom Resources > **Binary Search**
- External Resource > **Searching Algorithms** from CS Unplugged

Required Course Content

ENDURING UNDERSTANDING

AAP-2

The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.

LEARNING OBJECTIVE

AAP-2.P

For binary search algorithms:

a. Determine the number of iterations required to find a value in a data set. 1.D

b. Explain the requirements necessary to complete a binary search. 1.A

ESSENTIAL KNOWLEDGE

AAP-2.P.1

The binary search algorithm starts at the middle of a sorted data set of numbers and eliminates half of the data; this process repeats until the desired value is found or all elements have been eliminated.

EXCLUSION STATEMENT (EK: AAP-2.P.1):
Specific implementations of the binary search are outside the scope of the course and the AP Exam.

AAP-2.P.2

Data must be in sorted order to use the binary search algorithm.

AAP-2.P.3

Binary search is often more efficient than sequential/linear search when applied to sorted data.

BIG IDEA 3

Algorithms and Programming

SKILLS

3.B
Use abstraction to manage complexity in a program.

4.B
Determine the result of code segments.

AVAILABLE RESOURCE

- AP CSP Exam Reference Sheet (see Appendix)

TOPIC 3.12
Calling Procedures

Required Course Content

ENDURING UNDERSTANDING

AAP-3

Programmers break down problems into smaller and more manageable pieces. By creating procedures and leveraging parameters, programmers generalize processes that can be reused. Procedures allow programmers to draw upon existing code that has already been tested, allowing them to write programs more quickly and with more confidence.

LEARNING OBJECTIVE

AAP-3.A

For procedure calls:

a. Write statements to call procedures. 3.B

b. Determine the result or effect of a procedure call. 4.B

ESSENTIAL KNOWLEDGE

AAP-3.A.1

A *procedure* is a named group of programming instructions that may have parameters and return values.

AAP-3.A.2

Procedures are referred to by different names, such as *method* or *function*, depending on the programming language.

AAP-3.A.3

Parameters are input variables of a procedure. *Arguments* specify the values of the parameters when a procedure is called.

AAP-3.A.4

A procedure call interrupts the sequential execution of statements, causing the program to execute the statements within the procedure before continuing. Once the last statement in the procedure (or a return statement) has executed, flow of control is returned to the point immediately following where the procedure was called.

AAP-3.A.5

The exam reference sheet provides

```
procName(arg1, arg2, ...)
```

continued on next page

Algorithms and Programming

BIG IDEA 3

LEARNING OBJECTIVE

AAP-3.A

For procedure calls:

a. Write statements to call procedures. 3.B

b. Determine the result or effect of a procedure call. 4.B

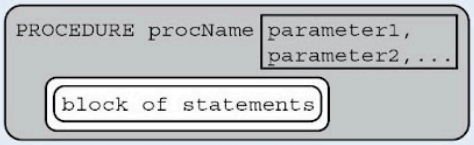
ESSENTIAL KNOWLEDGE

as a way to call

Text:

```
PROCEDURE procName(parameter1,
                   parameter2, ...)
{
    <block of statements>
}
```

Block:



which takes zero or more arguments; `arg1` is assigned to `parameter1`, `arg2` is assigned to `parameter2`, and so on.

AAP-3.A.6

The exam reference sheet provides the procedure

Text:

```
DISPLAY(expression)
```

Block:



to display the value of `expression`, followed by a space.

AAP-3.A.7

The exam reference sheet provides the

Text:

```
RETURN(expression)
```

Block:



statement, which is used to return the flow of control to the point where the procedure was called and to return the value of `expression`.

AAP-3.A.8

The exam reference sheet provides

```
result ← procName(arg1,
                  arg2, ...)
```

to assign to `result` the "value of the procedure" being returned by calling

continued on next page



BIG IDEA 3

Algorithms and Programming

LEARNING OBJECTIVE

AAP-3.A

For procedure calls:

a. Write statements to call procedures. 3.B

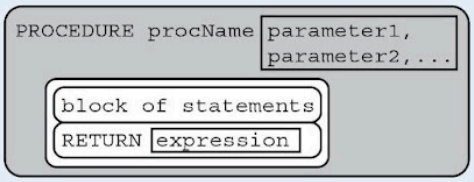
b. Determine the result or effect of a procedure call. 4.B

ESSENTIAL KNOWLEDGE

Text:

```
PROCEDURE procName(parameter1,
                   parameter2, ...)
{
   <block of statements>
   RETURN(expression)
}
```

Block:



AAP-3.A.9

The exam reference sheet provides procedure

Text:

`INPUT()`

Block:

`INPUT`

which accepts a value from the user and returns the input value.

Algorithms and Programming

BIG IDEA 3

TOPIC 3.13
Developing Procedures

Required Course Content

ENDURING UNDERSTANDING

AAP-3

Programmers break down problems into smaller and more manageable pieces. By creating procedures and leveraging parameters, programmers generalize processes that can be reused. Procedures allow programmers to draw upon existing code that has already been tested, allowing them to write programs more quickly and with more confidence.

LEARNING OBJECTIVE

AAP-3.B

Explain how the use of procedural abstraction manages complexity in a program. 3.C

ESSENTIAL KNOWLEDGE

AAP-3.B.1

One common type of abstraction is procedural abstraction, which provides a name for a process and allows a procedure to be used only knowing what it does, not how it does it.

AAP-3.B.2

Procedural abstraction allows a solution to a large problem to be based on the solutions of smaller subproblems. This is accomplished by creating procedures to solve each of the subproblems.

AAP-3.B.3

The subdivision of a computer program into separate subprograms is called *modularity*.

AAP-3.B.4

A procedural abstraction may extract shared features to generalize functionality instead of duplicating code. This allows for program code reuse, which helps manage complexity.

AAP-3.B.5

Using parameters allows procedures to be generalized, enabling the procedures to be reused with a range of input values or arguments.

SKILLS

3.B
Use abstraction to manage complexity in a program.

3.C
Explain how abstraction manages complexity.

AVAILABLE RESOURCE

- AP CSP Exam Reference Sheet (see Appendix)

continued on next page



BIG IDEA 3

Algorithms and Programming

LEARNING OBJECTIVE

AAP-3.B
Explain how the use of procedural abstraction manages complexity in a program. 3.C

AAP-3.C
Develop procedural abstractions to manage complexity in a program by writing procedures. 3.B

ESSENTIAL KNOWLEDGE

AAP-3.B.6
Using procedural abstraction helps improve code readability.

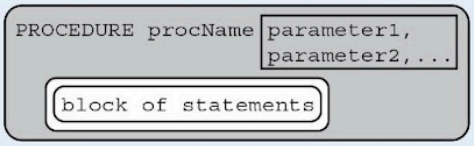
AAP-3.B.7
Using procedural abstraction in a program allows programmers to change the internals of the procedure (to make it faster, more efficient, use less storage, etc.) without needing to notify users of the change as long as what the procedure does is preserved.

AAP-3.C.1
The exam reference sheet provides

Text:

```
PROCEDURE procName(parameter1,
                   parameter2, ...)
{
   <block of statements>
}
```

Block:



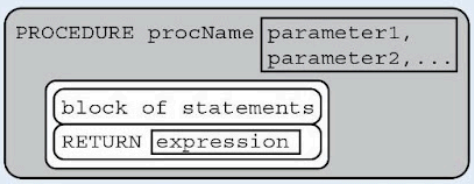
which is used to define a procedure that takes zero or more arguments. The procedure contains `block of statements`.

AAP-3.C.2
The exam reference sheet provides

Text:

```
PROCEDURE procName(parameter1,
                   parameter2, ...)
{
   <block of statements>
   RETURN(expression)
}
```

Block:



continued on next page

Algorithms and Programming

BIG IDEA 3

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
AAP-3.C Develop procedural abstractions to manage complexity in a program by writing procedures. 3.B	which is used to define a procedure that takes zero or more arguments. The procedure contains `block of statements` and returns the value of `expression`. The `RETURN` statement may appear at any point inside the procedure and causes an immediate return from the procedure back to the calling statement.



BIG IDEA 3

Algorithms and Programming

SKILL
2.B
Implement and apply an algorithm.

TOPIC 3.14
Libraries

Required Course Content

ENDURING UNDERSTANDING

AAP-3

Programmers break down problems into smaller and more manageable pieces. By creating procedures and leveraging parameters, programmers generalize processes that can be reused. Procedures allow programmers to draw upon existing code that has already been tested, allowing them to write programs more quickly and with more confidence.

LEARNING OBJECTIVE

AAP-3.D

Select appropriate libraries or existing code segments to use in creating new programs. 2.B

ESSENTIAL KNOWLEDGE

AAP-3.D.1

A software library contains procedures that may be used in creating new programs.

AAP-3.D.2

Existing code segments can come from internal or external sources, such as libraries or previously written code.

AAP-3.D.3

The use of libraries simplifies the task of creating complex programs.

AAP-3.D.4

Application program interfaces (APIs) are specifications for how the procedures in a library behave and can be used.

AAP-3.D.5

Documentation for an API/library is necessary in understanding the behaviors provided by the API/library and how to use them.

Algorithms and Programming

BIG IDEA 3

TOPIC 3.15
Random Values

SKILLS

2.B
Implement and apply an algorithm.

4.B
Determine the result of code segments.

AVAILABLE RESOURCES

- Professional Development > **Teaching and Assessing Module: Algorithmic Thinking**
- AP CSP Exam Reference Sheet (see Appendix)

Required Course Content

ENDURING UNDERSTANDING

AAP-3

Programmers break down problems into smaller and more manageable pieces. By creating procedures and leveraging parameters, programmers generalize processes that can be reused. Procedures allow programmers to draw upon existing code that has already been tested, allowing them to write programs more quickly and with more confidence.

LEARNING OBJECTIVE

AAP-3.E

For generating random values:

a. Write expressions to generate possible values. 2.B

b. Evaluate expressions to determine the possible results. 4.B

ESSENTIAL KNOWLEDGE

AAP-3.E.1

The exam reference sheet provides

Text:

```
RANDOM(a, b)
```

Block:

RANDOM [a, b]

which generates and returns a random integer from `a` to `b`, inclusive. Each result is equally likely to occur. For example, `RANDOM(1, 3)` could return `1`, `2`, or `3`.

AAP-3.E.2

Using random number generation in a program means each execution may produce a different result.

BIG IDEA 3

Algorithms and Programming

SKILLS

1.A
Investigate the situation, context, or task.

1.D
Evaluate solution options.

TOPIC 3.16
Simulations

Required Course Content

ENDURING UNDERSTANDING

AAP-3

Programmers break down problems into smaller and more manageable pieces. By creating procedures and leveraging parameters, programmers generalize processes that can be reused. Procedures allow programmers to draw upon existing code that has already been tested, allowing them to write programs more quickly and with more confidence.

LEARNING OBJECTIVE

AAP-3.F

For simulations:

a. Explain how computers can be used to represent real-world phenomena or outcomes. 1.A

b. Compare simulations with real-world contexts. 1.D

ESSENTIAL KNOWLEDGE

AAP-3.F.1
Simulations are abstractions of more complex objects or phenomena for a specific purpose.

AAP-3.F.2
A *simulation* is a representation that uses varying sets of values to reflect the changing state of a phenomenon.

AAP-3.F.3
Simulations often mimic real-world events with the purpose of drawing inferences, allowing investigation of a phenomenon without the constraints of the real world.

AAP-3.F.4
The process of developing an abstract simulation involves removing specific details or simplifying functionality.

AAP-3.F.5
Simulations can contain bias derived from the choices of real-world elements that were included or excluded.

AAP-3.F.6
Simulations are most useful when real-world events are impractical for experiments (e.g., too big, too small, too fast, too slow, too expensive, or too dangerous).

continued on next page

Algorithms and Programming

BIG IDEA 3

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
AAP-3.F For simulations: a. Explain how computers can be used to represent real-world phenomena or outcomes. 1.A b. Compare simulations with real-world contexts. 1.D	**AAP-3.F.7** Simulations facilitate the formulation and refinement of hypotheses related to the objects or phenomena under consideration. **AAP-3.F.8** Random number generators can be used to simulate the variability that exists in the real world.



BIG IDEA 3

Algorithms and Programming

SKILL
1.D
Evaluate solution options.

TOPIC 3.17
Algorithmic Efficiency

Required Course Content

ENDURING UNDERSTANDING

AAP-4

There exist problems that computers cannot solve, and even when a computer can solve a problem, it may not be able to do so in a reasonable amount of time.

LEARNING OBJECTIVE

AAP-4.A

For determining the efficiency of an algorithm:

a. Explain the difference between algorithms that run in reasonable time and those that do not. 1.D

b. Identify situations where a heuristic solution may be more appropriate. 1.D

ESSENTIAL KNOWLEDGE

AAP-4.A.1

A *problem* is a general description of a task that can (or cannot) be solved algorithmically. An *instance* of a problem also includes specific input. For example, sorting is a problem; sorting the list (2,3,1,7) is an instance of the problem.

AAP-4.A.2

A *decision problem* is a problem with a yes/no answer (e.g., is there a path from A to B?). An *optimization problem* is a problem with the goal of finding the "best" solution among many (e.g., what is the shortest path from A to B?).

AAP-4.A.3

Efficiency is an estimation of the amount of computational resources used by an algorithm. Efficiency is typically expressed as a function of the size of the input.

EXCLUSION STATEMENT (EK AAP-4.A.3):
Formal analysis of algorithms (Big-O) and formal reasoning using mathematical formulas are outside the scope of this course and the AP Exam.

AAP-4.A.4

An algorithm's efficiency is determined through formal or mathematical reasoning.

continued on next page

Algorithms and Programming

BIG IDEA 3

LEARNING OBJECTIVE

AAP-4.A

For determining the efficiency of an algorithm:

a. Explain the difference between algorithms that run in reasonable time and those that do not. 1.D

b. Identify situations where a heuristic solution may be more appropriate. 1.D

ESSENTIAL KNOWLEDGE

AAP-4.A.5

An algorithm's efficiency can be informally measured by determining the number of times a statement or group of statements executes.

AAP-4.A.6

Different correct algorithms for the same problem can have different efficiencies.

AAP-4.A.7

Algorithms with a polynomial efficiency or slower (constant, linear, square, cube, etc.) are said to run in a *reasonable amount of time*. Algorithms with exponential or factorial efficiencies are examples of algorithms that run in an *unreasonable amount of time*.

AAP-4.A.8

Some problems cannot be solved in a reasonable amount of time because there is no efficient algorithm for solving them. In these cases, approximate solutions are sought.

AAP-4.A.9

A *heuristic* is an approach to a problem that produces a solution that is not guaranteed to be optimal but may be used when techniques that are guaranteed to always find an optimal solution are impractical.

EXCLUSION STATEMENT (AAP-4.A.9):
Specific heuristic solutions are outside the scope of this course and the AP Exam.

BIG IDEA 3

Algorithms and Programming

SKILL
1.A
Investigate the situation, context, or task.

TOPIC 3.18
Undecidable Problems

Required Course Content

ENDURING UNDERSTANDING

AAP-4

There exist problems that computers cannot solve, and even when a computer can solve a problem, it may not be able to do so in a reasonable amount of time.

LEARNING OBJECTIVE

AAP-4.B

Explain the existence of undecidable problems in computer science. 1.A

ESSENTIAL KNOWLEDGE

AAP-4.B.1

A *decidable problem* is a decision problem for which an algorithm can be written to produce a correct output for all inputs (e.g., "Is the number even?").

AAP-4.B.2

An *undecidable problem* is one for which no algorithm can be constructed that is always capable of providing a correct yes-or-no answer.

EXCLUSION STATEMENT (EK AAP-4.B.2):
Determining whether a given problem is undecidable is outside the scope of this course and the AP Exam.

AAP-4.B.3

An undecidable problem may have some instances that have an algorithmic solution, but there is no algorithmic solution that could solve all instances of the problem.



AP COMPUTER SCIENCE PRINCIPLES

BIG IDEA 4

Computer Systems and Networks

AP

11–15%
AP EXAM WEIGHTING

AP

Remember to go to **AP Classroom** to assign students the online Topic Questions for this big idea.

Whether assigned as homework or completed in class, the Topic Questions can be used to spot-check student understanding and help identify content and skills to emphasize in lessons.

Topic Questions

Multiple-choice: ~10 questions

BIG IDEA 4

11–15% AP EXAM WEIGHTING

Computer Systems and Networks

ESSENTIAL QUESTIONS

CSN-1

- Why are long text messages sometime delivered out of order?
- When an Internet service outage occurs in a different part of your town or city, how are you still able to access the Internet?

CSN-2

- What are the benefits of dividing tasks among group members?
- Is there a point where adding another group member would not make completing the task faster? Why?

Developing Understanding

The Internet is a network that most students use on a regular basis to look up information, to socialize with friends, and in many cases to complete their school work. In this big idea, students will learn how computer systems and networks, primarily the Internet, work. Students will learn about how information is transmitted on the Internet and about the safeguards that have been put in place to keep this system from breaking down. In addition, students will learn the effect that dividing tasks across multiple computing devices can have on the speed at which processes can occur. This big idea is often taught in conjunction with Big Idea 5: Impact of Computing.

Building Computational Thinking Practices

1.D 5.A

Some programs are so large or complex that they end up taking a long time to run, making them impractical to use. In order to minimize this runtime, the individual processes that comprise a program can be run on multiple processors simultaneously. Programmers need to evaluate the options of running these processes sequentially or in parallel across multiple processors to optimize the solution time. When introducing students to distributed processes, use real-world examples, such as dividing up household chores or passing back classroom papers.

To create more stable access to the Internet, additional connections and routers can be added. Redundant routing options help ensure that the Internet is more reliable and stable. To determine where additional connections and routers need to be added to the systems, evaluation of existing infrastructure is beneficial. Using kinesthetic learning techniques to simulate the connections and routers in a system can help students make this abstract idea more concrete. As student understanding increases, propose problems for the students to solve, such as broken connections or the need to expand the system to include newly added routers or devices.

Preparing for the AP Exam

On the end-of-course exam, students will be presented with scenarios for how information could be passed via the Internet along with illustrations of interconnected computers in given networks. Students will be asked to select which choice best explains how information is passed through these networks from one computing device to another or how designing systems to include redundancy helps make them fault-tolerant. To aid in understanding these diagrams, students should be encouraged to use the Marking the Text strategy (see page 133). They can mark the diagrams with the information provided to ensure they have an accurate visual and can then match that visual to the explanations provided.

BIG IDEA 4

Computer Systems and Networks

BIG IDEA AT A GLANCE

Learning Objective	Topic	Skills	Unit/Module
CSN-1.A, CSN-1.B, CSN-1.C, CSN-1.D	4.1 The Internet	5.A Explain how computing systems work.	
CSN-1.E	4.2 Fault Tolerance	1.D Evaluate solution options. 5.A Explain how computing systems work.	
CSN-2.A, CSN-2.B	4.3 Parallel and Distributed Computing	1.D Evaluate solution options.	
	Go to AP Classroom to assign Topic Questions as you teach the topics in Big Idea 4. Review the results in class to identify and address any student misunderstandings.		

SAMPLE INSTRUCTIONAL ACTIVITIES

The sample activities on this page are optional and are offered to provide possible ways to incorporate instructional approaches into the classroom. They were developed in partnership with teachers from the AP community to share ways that they approach teaching some of the topics and skills in this big idea. Please refer to the Instructional Strategies section beginning on p. 132 for more examples of activities and strategies.

Activity	Topic	Sample Activity
1	4.1	**Journaling** Ask students to read about the Internet and packet switching in Blown to Bits. Pose several prompts related to the Internet, such as the following, and have students add their answers to their journals: ▪ How is the Internet like the US Post Office? ▪ Explain the difference between circuit switching and packet switching. Ask students to use what they learned from reading to make a drawing showing how they think an email travels from one place to another.
2	4.3	**Predict and compare** When introducing parallel and distributed computing, present students with a set of processes and several distributed models. Ask students to compare the models and predict which one is the most efficient, least efficient, or equivalent to other models in the set. Then show students how to determine the efficiency of each model to check if their predictions were correct.

Big Idea Planning Notes

Use the space below to plan your approach to the topics in this big idea. Consider what resources and instructional strategies you might want to use.

BIG IDEA 4

Computer Systems and Networks

SKILL
5.A
Explain how computing systems work.

TOPIC 4.1
The Internet

AVAILABLE RESOURCES
- Classroom Resources > **The Computer and the Internet: TCP/IP: How Messages Get Delivered Across the Internet**
- External Resources >
  - **Routing and Deadlock** from CS Unplugged
  - **Network Protocols** from CS Unplugged
  - **Blown to Bits: Appendix**

Required Course Content

ENDURING UNDERSTANDING

CSN-1
Computer systems and networks facilitate the transfer of data.

LEARNING OBJECTIVE

CSN-1.A
Explain how computing devices work together in a network. 5.A

ESSENTIAL KNOWLEDGE

CSN-1.A.1
A *computing device* is a physical artifact that can run a program. Some examples include computers, tablets, servers, routers, and smart sensors.

CSN-1.A.2
A *computing system* is a group of computing devices and programs working together for a common purpose.

CSN-1.A.3
A *computer network* is a group of interconnected computing devices capable of sending or receiving data.

CSN-1.A.4
A computer network is a type of computing system.

CSN-1.A.5
A *path* between two computing devices on a computer network (a sender and a receiver) is a sequence of directly connected computing devices that begins at the sender and ends at the receiver.

CSN-1.A.6
Routing is the process of finding a path from sender to receiver.

continued on next page

Computer Systems and Networks

BIG IDEA 4

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
CSN-1.A Explain how computing devices work together in a network. 5.A	CSN-1.A.7 The *bandwidth* of a computer network is the maximum amount of data that can be sent in a fixed amount of time. CSN-1.A.8 Bandwidth is usually measured in bits per second.
CSN-1.B Explain how the Internet works. 5.A	CSN-1.B.1 The Internet is a computer network consisting of interconnected networks that use standardized, open (nonproprietary) communication protocols. CSN-1.B.2 Access to the Internet depends on the ability to connect a computing device to an Internet-connected device. CSN-1.B.3 A *protocol* is an agreed-upon set of rules that specify the behavior of a system. CSN-1.B.4 The protocols used in the Internet are *open*, which allows users to easily connect additional computing devices to the Internet. CSN-1.B.5 Routing on the Internet is usually dynamic; it is not specified in advance. CSN-1.B.6 The *scalability* of a system is the capacity for the system to change in size and scale to meet new demands. CSN-1.B.7 The Internet was designed to be scalable.

continued on next page



BIG IDEA 4

Computer Systems and Networks

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
CSN-1.C Explain how data are sent through the Internet via packets. 5.A	**CSN-1.C.1** Information is passed through the Internet as a *data stream*. Data streams contain chunks of data, which are encapsulated in *packets*.
	CSN-1.C.2 Packets contain a chunk of data and metadata used for routing the packet between the origin and the destination on the Internet, as well as for data reassembly.
	CSN-1.C.3 Packets may arrive at the destination in order, out of order, or not at all.
	CSN-1.C.4 IP, TCP, and UDP are common protocols used on the Internet.
CSN-1.D Describe the differences between the Internet and the World Wide Web. 5.A	**CSN-1.D.1** The World Wide Web is a system of linked pages, programs, and files.
	CSN-1.D.2 HTTP is a protocol used by the World Wide Web.
	CSN-1.D.3 The World Wide Web uses the Internet.

Computer Systems and Networks

BIG IDEA 4

TOPIC 4.2
Fault Tolerance

SKILLS

1.D
Evaluate solution options.

5.A
Explain how computing systems work.

Required Course Content

ENDURING UNDERSTANDING

CSN-1
Computer systems and networks facilitate the transfer of data.

LEARNING OBJECTIVE

CSN-1.E
For fault-tolerant systems, like the Internet:

a. Describe the benefits of fault tolerance. 1.D

b. Explain how a given system is fault-tolerant. 5.A

c. Identify vulnerabilities to failure in a system. 1.D

ESSENTIAL KNOWLEDGE

CSN-1.E.1
The Internet has been engineered to be fault-tolerant, with abstractions for routing and transmitting data.

CSN-1.E.2
Redundancy is the inclusion of extra components that can be used to mitigate failure of a system if other components fail.

CSN-1.E.3
One way to accomplish network redundancy is by having more than one path between any two connected devices.

CSN-1.E.4
If a particular device or connection on the Internet fails, subsequent data will be sent via a different route, if possible.

CSN-1.E.5
When a system can support failures and still continue to function, it is called *fault-tolerant*. This is important because elements of complex systems fail at unexpected times, often in groups, and fault tolerance allows users to continue to use the network.

CSN-1.E.6
Redundancy within a system often requires additional resources but can provide the benefit of fault tolerance.

continued on next page

BIG IDEA 4

Computer Systems and Networks

LEARNING OBJECTIVE

CSN-1.E

For fault-tolerant systems, like the Internet:

a. Describe the benefits of fault tolerance. 1.D

b. Explain how a given system is fault-tolerant. 5.A

c. Identify vulnerabilities to failure in a system. 1.D

ESSENTIAL KNOWLEDGE

CSN-1.E.7

The redundancy of routing options between two points increases the reliability of the Internet and helps it scale to more devices and more people.

TOPIC 4.3
Parallel and Distributed Computing

SKILL
1.D
Evaluate solution options.

Required Course Content

ENDURING UNDERSTANDING

CSN-2
Parallel and distributed computing leverage multiple computers to more quickly solve complex problems or process large data sets.

LEARNING OBJECTIVE

CSN-2.A
For sequential, parallel, and distributed computing:

a. Compare problem solutions. 1.D

b. Determine the efficiency of solutions. 1.D

ESSENTIAL KNOWLEDGE

CSN-2.A.1
Sequential computing is a computational model in which operations are performed in order one at a time.

CSN-2.A.2
Parallel computing is a computational model where the program is broken into multiple smaller sequential computing operations, some of which are performed simultaneously.

CSN-2.A.3
Distributed computing is a computational model in which multiple devices are used to run a program.

CSN-2.A.4
Comparing efficiency of solutions can be done by comparing the time it takes them to perform the same task.

CSN-2.A.5
A sequential solution takes as long as the sum of all of its steps.

CSN-2.A.6
A parallel computing solution takes as long as its sequential tasks plus the longest of its parallel tasks.

continued on next page

BIG IDEA 4

Computer Systems and Networks

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
CSN-2.A For sequential, parallel, and distributed computing: a. Compare problem solutions. 1.D b. Determine the efficiency of solutions. 1.D	CSN-2.A.7 The "speedup" of a parallel solution is measured in the time it took to complete the task sequentially divided by the time it took to complete the task when done in parallel.
CSN-2.B Describe benefits and challenges of parallel and distributed computing. 1.D	CSN-2.B.1 Parallel computing consists of a parallel portion and a sequential portion. CSN-2.B.2 Solutions that use parallel computing can scale more effectively than solutions that use sequential computing. CSN-2.B.3 Distributed computing allows problems to be solved that could not be solved on a single computer because of either the processing time or storage needs involved. CSN-2.B.4 Distributed computing allows much larger problems to be solved quicker than they could be solved using a single computer. CSN-2.B.5 When increasing the use of parallel computing in a solution, the efficiency of the solution is still limited by the sequential portion. This means that at some point, adding parallel portions will no longer meaningfully increase efficiency.



AP COMPUTER SCIENCE PRINCIPLES

BIG IDEA 5

Impact of Computing

AP

21–26%
AP EXAM WEIGHTING

AP

Remember to go to **AP Classroom** to assign students the online Topic Questions for this big idea.

Whether assigned as homework or completed in class, the Topic Questions can be used to spot-check student understanding and help identify content and skills to emphasize in lessons.

Topic Questions

Multiple-choice: ~20 questions

BIG IDEA 5

21–26% AP EXAM WEIGHTING

Impact of Computing

ESSENTIAL QUESTIONS

IOC-1

- What app or computer software do you use most often and would have a hard time going without? How does this software solve a problem for you or benefit you?
- Are innovators responsible for the harmful effects of their computing innovations, even if those effects were unintentional? Why or why not?

IOC-2

- What data are generated by smart phones, and what are they being used for?

Developing Understanding

The creation of computer programs can have extensive impacts, some unintended, on societies, economies, and cultures. In this big idea, students explore these effects, the legal and ethical concerns that come with programs, and the responsibilities of programmers. When using computing innovations and transmitting information via the Internet, students should be aware of the risk of sharing personal identifiable information about themselves, such as their age or address, and actively take steps to keep this information safe. This big idea can be integrated throughout the course and works well with the Creative Development, Data, and Computing Systems and Networks big ideas.

Building Computational Thinking Practices

1.C 5.C 5.D 5.E

Computing innovations and programs are often developed in teams. A good, collaborative team starts with a group that is made up of people from different backgrounds, genders, ages, and races so that the perspectives of all potential users are being represented. By creating diverse groups where each person's opinion is considered, we help avoid unintentional bias and potential negative effects, such as contributing to the digital divide, that can creep into innovations.

Investigating the impact of existing computing innovations can help students avoid unintentional negative effects of their own innovations. Consumers should be aware of the impact that a new computing innovation might have before beginning to use it, as well as what data are being gathered and how the product owner intends to use those data. While students may find it relatively easy to describe how the gathering of data would impact them, it is sometimes more difficult for students to understand the impacts that computing innovations and the gathering of personal data might have on people who are different from them, or on society as a whole. As students investigate computing innovations, provide opportunities for students to learn from others' perspectives by allowing time for viewpoints and potential impacts to be shared during a group discussion, like a debate.

Preparing for the AP Exam

Students will be asked to complete three investigations into computing innovations during the school year. Through these investigations, students will look at the data the computing innovation uses to complete its task; any data privacy, security, or storage concerns that might be associated with the innovation; and beneficial and harmful effects the computing innovation might have on society, the economy, or culture.

On the end-of-course exam, students will be presented with a passage about a computing innovation and will be asked a series of questions about data and the effects of the computing innovation. While the computing innovations that need to be investigated are not specified in the curricular requirement, students will benefit from investigating a large range of computing innovations.

BIG IDEA 5

Impact of Computing

BIG IDEA AT A GLANCE

Learning Objective	Topic	Skills	Unit/Module
IOC-1.A, IOC-1.B	5.1 Beneficial and Harmful Effects	5.C Describe the impact of a computing innovation.	
IOC-1.C	5.2 Digital Divide	5.C Describe the impact of a computing innovation.	
IOC-1.D	5.3 Computing Bias	5.E Evaluate the use of computing based on legal and ethical factors.	
IOC-1.E	5.4 Crowdsourcing	1.C Explain how collaboration affects the development of a solution.	
IOC-1.F	5.5 Legal and Ethical Concerns	5.E Evaluate the use of computing based on legal and ethical factors.	
IOC-2.A, IOC-2.B, IOC-2.C	5.6 Safe Computing	5.D Describe the impact of gathering data. 5.E Evaluate the use of computing based on legal and ethical factors.	

Go to **AP Classroom** to assign Topic Questions as you teach the topics in Big Idea 5. Review the results in class to identify and address any student misunderstandings.

SAMPLE INSTRUCTIONAL ACTIVITIES

The sample activities on this page are optional and are offered to provide possible ways to incorporate instructional approaches into the classroom. They were developed in partnership with teachers from the AP community to share ways that they approach teaching some of the topics and skills in this big idea. Please refer to the Instructional Strategies section beginning on p. 132 for more examples of activities and strategies.

Activity	Topic	Sample Activity
1	5.1	**Marking the text** Provide students with an article that highlights both beneficial and harmful effects of a specific computing innovation, and have them mark which effects are beneficial and which are harmful. For each effect the students mark as harmful, have them add notes about whether they think these effects should have been anticipated in advance. For each effect the students mark as beneficial, have the students make notes indicating if they think these benefits were intended or unintended.
2	5.6	**Kinesthetic learning** In small groups, have students create and act out a play or a scene involving privacy and security risks, especially when it comes to personally identifiable information (PII) and the impact of collecting such data. Sample topics might include not recognizing a phishing email, being careless with passwords, downloading a virus accidentally, or not being aware of a search history being kept on a computer. Students could extend their play to include best practices or ways to stay safer when using computing innovations.

Big Idea Planning Notes

Use the space below to plan your approach to the topics in this big idea. Consider what resources and instructional strategies you might want to use.

BIG IDEA 5

Impact of Computing

SKILL
5.C
Describe the impact of a computing innovation.

AVAILABLE RESOURCE
- External Resource > ACM Tech News

TOPIC 5.1
Beneficial and Harmful Effects

Required Course Content

ENDURING UNDERSTANDING

IOC-1
While computing innovations are typically designed to achieve a specific purpose, they may have unintended consequences.

LEARNING OBJECTIVE

IOC-1.A
Explain how an effect of a computing innovation can be both beneficial and harmful.
5.C

ESSENTIAL KNOWLEDGE

IOC-1.A.1
People create computing innovations.

IOC-1.A.2
The way people complete tasks often changes to incorporate new computing innovations.

IOC-1.A.3
Not every effect of a computing innovation is anticipated in advance.

IOC-1.A.4
A single effect can be viewed as both beneficial and harmful by different people, or even by the same person.

IOC-1.A.5
Advances in computing have generated and increased creativity in other fields, such as medicine, engineering, communications, and the arts.

continued on next page

Impact of Computing

BIG IDEA 5

LEARNING OBJECTIVE

IOC-1.B
Explain how a computing innovation can have an impact beyond its intended purpose.
5.C

ESSENTIAL KNOWLEDGE

IOC-1.B.1
Computing innovations can be used in ways that their creators had not originally intended:

- The World Wide Web was originally intended only for rapid and easy exchange of information within the scientific community.
- Targeted advertising is used to help businesses, but it can be misused at both individual and aggregate levels.
- Machine learning and data mining have enabled innovation in medicine, business, and science, but information discovered in this way has also been used to discriminate against groups of individuals.

IOC-1.B.2
Some of the ways computing innovations can be used may have a harmful impact on society, the economy, or culture.

IOC-1.B.3
Responsible programmers try to consider the unintended ways their computing innovations can be used and the potential beneficial and harmful effects of these new uses.

IOC-1.B.4
It is not possible for a programmer to consider all the ways a computing innovation can be used.

IOC-1.B.5
Computing innovations have often had unintended beneficial effects by leading to advances in other fields.

IOC-1.B.6
Rapid sharing of a program or running a program with a large number of users can result in significant impacts beyond the intended purpose or control of the programmer.

BIG IDEA 5

Impact of Computing

SKILL
5.C
Describe the impact of a computing innovation.

AVAILABLE RESOURCE
- External Resource > ACM Tech News

TOPIC 5.2
Digital Divide

Required Course Content

ENDURING UNDERSTANDING

IOC-1
While computing innovations are typically designed to achieve a specific purpose, they may have unintended consequences.

LEARNING OBJECTIVE

IOC-1.C
Describe issues that contribute to the digital divide.
5.C

ESSENTIAL KNOWLEDGE

IOC-1.C.1
Internet access varies between socioeconomic, geographic, and demographic characteristics, as well as between countries.

IOC-1.C.2
The "digital divide" refers to differing access to computing devices and the Internet, based on socioeconomic, geographic, or demographic characteristics.

IOC-1.C.3
The digital divide can affect both groups and individuals.

IOC-1.C.4
The digital divide raises issues of equity, access, and influence, both globally and locally.

IOC-1.C.5
The digital divide is affected by the actions of individuals, organizations, and governments.

Impact of Computing

BIG IDEA 5

TOPIC 5.3
Computing Bias

SKILL
5.E
Evaluate the use of computing based on legal and ethical factors.

AVAILABLE RESOURCE
- External Resource > ACM Tech News

Required Course Content

ENDURING UNDERSTANDING

IOC-1
While computing innovations are typically designed to achieve a specific purpose, they may have unintended consequences.

LEARNING OBJECTIVE

IOC-1.D
Explain how bias exists in computing innovations. 5.E

ESSENTIAL KNOWLEDGE

IOC-1.D.1
Computing innovations can reflect existing human biases because of biases written into the algorithms or biases in the data used by the innovation.

IOC-1.D.2
Programmers should take action to reduce bias in algorithms used for computing innovations as a way of combating existing human biases.

IOC-1.D.3
Biases can be embedded at all levels of software development.

BIG IDEA 5

Impact of Computing

SKILL

1.C

Explain how collaboration affects the development of a solution.

TOPIC 5.4
Crowdsourcing

Required Course Content

ENDURING UNDERSTANDING

IOC-1

While computing innovations are typically designed to achieve a specific purpose, they may have unintended consequences.

LEARNING OBJECTIVE

IOC-1.E

Explain how people participate in problem-solving processes at scale. 1.C

ESSENTIAL KNOWLEDGE

IOC-1.E.1

Widespread access to information and public data facilitates the identification of problems, development of solutions, and dissemination of results.

IOC-1.E.2

Science has been affected by using distributed and "citizen science" to solve scientific problems.

IOC-1.E.3

Citizen science is scientific research conducted in whole or part by distributed individuals, many of whom may not be scientists, who contribute relevant data to research using their own computing devices.

IOC-1.E.4

Crowdsourcing is the practice of obtaining input or information from a large number of people via the Internet.

IOC-1.E.5

Human capabilities can be enhanced by collaboration via computing.

IOC-1.E.6

Crowdsourcing offers new models for collaboration, such as connecting businesses or social causes with funding.

Impact of Computing

BIG IDEA 5

TOPIC 5.5
Legal and Ethical Concerns

SKILL
5.E
Evaluate the use of computing based on legal and ethical factors.

AVAILABLE RESOURCES
- Classroom Resources > Ethical Use of the Computer
- External Resources >
  - ACM Tech News
  - Blown to Bits: Chapter 7

Required Course Content

ENDURING UNDERSTANDING

IOC-1
While computing innovations are typically designed to achieve a specific purpose, they may have unintended consequences.

LEARNING OBJECTIVE

IOC-1.F
Explain how the use of computing can raise legal and ethical concerns. 5.E

ESSENTIAL KNOWLEDGE

IOC-1.F.1
Material created on a computer is the intellectual property of the creator or an organization.

IOC-1.F.2
Ease of access and distribution of digitized information raises intellectual property concerns regarding ownership, value, and use.

IOC-1.F.3
Measures should be taken to safeguard intellectual property.

IOC-1.F.4
The use of material created by someone else without permission and presented as one's own is plagiarism and may have legal consequences.

continued on next page

BIG IDEA 5

Impact of Computing

LEARNING OBJECTIVE

IOC-1.F

Explain how the use of computing can raise legal and ethical concerns. 5.E

ESSENTIAL KNOWLEDGE

IOC-1.F.5

Some examples of legal ways to use materials created by someone else include:

- Creative Commons—a public copyright license that enables the free distribution of an otherwise copyrighted work. This is used when the content creator wants to give others the right to share, use, and build upon the work they have created.
- open source—programs that are made freely available and may be redistributed and modified
- open access—online research output free of any and all restrictions on access and free of many restrictions on use, such as copyright or license restrictions

IOC-1.F.6

The use of material created by someone other than you should always be cited.

IOC-1.F.7

Creative Commons, open source, and open access have enabled broad access to digital information.

IOC-1.F.8

As with any technology or medium, using computing to harm individuals or groups of people raises legal and ethical concerns.

IOC-1.F.9

Computing can play a role in social and political issues, which in turn often raises legal and ethical concerns.

IOC-1.F.10

The digital divide raises ethical concerns around computing.

IOC-1.F.11

Computing innovations can raise legal and ethical concerns. Some examples of these include:

- the development of software that allows access to digital media downloads and streaming
- the development of algorithms that include bias
- the existence of computing devices that collect and analyze data by continuously monitoring activities

TOPIC 5.6
Safe Computing

Required Course Content

ENDURING UNDERSTANDING

IOC-2

The use of computing innovations may involve risks to personal safety and identity.

LEARNING OBJECTIVE

IOC-2.A

Describe the risks to privacy from collecting and storing personal data on a computer system. 5.D

ESSENTIAL KNOWLEDGE

IOC-2.A.1

Personally identifiable information (PII) is information about an individual that identifies, links, relates, or describes them. Examples of PII include:

- Social Security number
- age
- race
- phone number(s)
- medical information
- financial information
- biometric data

IOC-2.A.2

Search engines can record and maintain a history of searches made by users.

IOC-2.A.3

Websites can record and maintain a history of individuals who have viewed their pages.

IOC-2.A.4

Devices, websites, and networks can collect information about a user's location.

IOC-2.A.5

Technology enables the collection, use, and exploitation of information about, by, and for individuals, groups, and institutions.

continued on next page

SKILLS

5.D
Describe the impact of gathering data.

5.E
Evaluate the use of computing based on legal and ethical factors.

AVAILABLE RESOURCES

- External Resources >
  - ACM Tech News
  - Public Key Encryption from CS Unplugged
  - Blown to Bits: Chapter 2 and 5

BIG IDEA 5

Impact of Computing

LEARNING OBJECTIVE

IOC-2.A

Describe the risks to privacy from collecting and storing personal data on a computer system. 5.D

ESSENTIAL KNOWLEDGE

IOC-2.A.6

Search engines can use search history to suggest websites or for targeted marketing.

IOC-2.A.7

Disparate personal data, such as geolocation, cookies, and browsing history, can be aggregated to create knowledge about an individual.

IOC-2.A.8

PII and other information placed online can be used to enhance a user's online experiences.

IOC-2.A.9

PII stored online can be used to simplify making online purchases.

IOC-2.A.10

Commercial and governmental curation of information may be exploited if privacy and other protections are ignored.

IOC-2.A.11

Information placed online can be used in ways that were not intended and that may have a harmful impact. For example, an email message may be forwarded, tweets can be retweeted, and social media posts can be viewed by potential employers.

IOC-2.A.12

PII can be used to stalk or steal the identity of a person or to aid in the planning of other criminal acts.

IOC-2.A.13

Once information is placed online, it is difficult to delete.

IOC-2.A.14

Programs can collect your location and record where you have been, how you got there, and how long you were at a given location.

IOC-2.A.15

Information posted to social media services can be used by others. Combining information posted on social media and other sources can be used to deduce private information about you.

continued on next page

Impact of Computing

BIG IDEA 5

LEARNING OBJECTIVE

IOC-2.B
Explain how computing resources can be protected and can be misused. **5.E**

ESSENTIAL KNOWLEDGE

IOC-2.B.1
Authentication measures protect devices and information from unauthorized access. Examples of authentication measures include strong passwords and multifactor authentication.

IOC-2.B.2
A strong password is something that is easy for a user to remember but would be difficult for someone else to guess based on knowledge of that user.

IOC-2.B.3
Multifactor authentication is a method of computer access control in which a user is only granted access after successfully presenting several separate pieces of evidence to an authentication mechanism, typically in at least two of the following categories: knowledge (something they know), possession (something they have), and inherence (something they are).

IOC-2.B.4
Multifactor authentication requires at least two steps to unlock protected information; each step adds a new layer of security that must be broken to gain unauthorized access.

IOC-2.B.5
Encryption is the process of encoding data to prevent unauthorized access. *Decryption* is the process of decoding the data. Two common encryption approaches are:

- Symmetric key encryption involves one key for both encryption and decryption.
- Public key encryption pairs a public key for encryption and a private key for decryption. The sender does not need the receiver's private key to encrypt a message, but the receiver's private key is required to decrypt the message.

EXCLUSION STATEMENT (EK IOC-2.B.5):
Specific mathematical procedures for encryption and decryption are beyond the scope of this course and the AP Exam.

IOC-2.B.6
Certificate authorities issue digital certificates that validate the ownership of encryption keys used in secure communications and are based on a trust model.

continued on next page

BIG IDEA 5

Impact of Computing

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
IOC-2.B Explain how computing resources can be protected and can be misused. 5.E	**IOC-2.B.7** Computer virus and malware scanning software can help protect a computing system against infection. **IOC-2.B.8** A *computer virus* is a malicious program that can copy itself and gain access to a computer in an unauthorized way. Computer viruses often attach themselves to legitimate programs and start running independently on a computer. **IOC-2.B.9** *Malware* is software intended to damage a computing system or to take partial control over its operation. **IOC-2.B.10** All real-world systems have errors or design flaws that can be exploited to compromise them. Regular software updates help fix errors that could compromise a computing system. **IOC-2.B.11** Users can control the permissions programs have for collecting user information. Users should review the permission settings of programs to protect their privacy.
IOC-2.C Explain how unauthorized access to computing resources is gained. 5.E	**IOC-2.C.1** *Phishing* is a technique that attempts to trick a user into providing personal information. That personal information can then be used to access sensitive online resources, such as bank accounts and emails. **IOC-2.C.2** *Keylogging* is the use of a program to record every keystroke made by a computer user in order to gain fraudulent access to passwords and other confidential information. **IOC-2.C.3** Data sent over public networks can be intercepted, analyzed, and modified. One way that this can happen is through a rogue access point. **IOC-2.C.4** A *rogue access point* is a wireless access point that gives unauthorized access to secure networks.

continued on next page



Impact of Computing

BIG IDEA 5

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
IOC-2.C Explain how unauthorized access to computing resources is gained. 5.E	**IOC-2.C.5** A malicious link can be disguised on a web page or in an email message. **IOC-2.C.6** Unsolicited emails, attachments, links, and forms in emails can be used to compromise the security of a computing system. These can come from unknown senders or from known senders whose security has been compromised. **IOC-2.C.7** Untrustworthy (often free) downloads from freeware or shareware sites can contain malware.



IEP AND 504 ACCOMMODATIONS AND MODIFICATIONS FOR EARLY CHILDHOOD AND ELEMENTARY SCHOOL STUDENTS:

Large print textbooks
Textbooks for at-home use
Additional time for assignments
Review of directions
Use of mnemonics
Have student restate information
Concrete examples
Adaptive writing utensils
Support auditory presentations with visuals
Use of a study carrel
Assistance in maintaining uncluttered space
Weekly home-school communication tools (notebook, daily log, phone calls or email messages)
Space for movement or breaks
Study sheets and teacher outlines
Extra visual and verbal cues and prompts
Math sheets with highlighted instructions
Graph paper to assist in organizing or lining up math problems
Use of computers and calculators
Books on tape
Graphic organizers
Quiet corner or room to calm down and relax when anxious
Preferential seating
Alteration of the classroom arrangement
Reduction of distractions
Answers to be dictated
Hands-on activities
Use of manipulatives
No penalty for spelling errors or sloppy handwriting
Follow a routine/schedule
Alternate quiet and active time
Teach time management skills
Rest breaks
Verbal and visual cues regarding directions and staying on task
Agenda book and checklists
Adjusted assignment timelines
Visual daily schedule

Varied reinforcement procedures
Immediate feedback
Work-in-progress check
Personalized examples

TESTING AND ASSESSMENT ACCOMMODATIONS AND MODIFICATIONS:

Answers to be dictated
Frequent rest breaks
Additional time
Oral testing
Untimed tests
Choice of test format (multiple-choice, essay, true-false)
Alternate ways to evaluate (projects or oral presentations instead of written tests)
Accept short answers
Open-book tests
Read test and directions to student
Provide study outline prior to tests
Highlight key directions
Test in alternative site
Use of calculator
Extra credit option
Preview test procedures
Simplified test wording; rephrased test questions and/or directions

GENERAL ACCOMMODATIONS AND MODIFICATIONS:

Use of alternative books or materials on the topic being studied
Computerized spell-check support
Word bank of choices for answers to test questions
Provision of calculator and/or number line for math tests
Reworded questions in simpler language
Projects instead of written reports
Highlighting important words or phrases in reading assignments
Modified workload or length of assignments/tests
Modified time demands
Modified grades based on IEP

BEHAVIOR ACCOMMODATIONS MODIFICATIONS:

Breaks between tasks
Cue expected behavior
Daily feedback to student
Have contingency plans
Use de-escalating strategies
Use positive reinforcement
Use proximity/touch control
Use peer supports and mentoring
Model expected behavior by adults
Have parent sign homework
Have parent sign behavior chart
Set and post class rules
Chart progress and maintain data



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

IEP AND 504 ACCOMODATIONS AND MODIFICATIONS FOR MIDDLE AND HIGH SCHOOL STUDENTS:

Large print textbooks
Textbooks for at-home use
Additional time for assignments
A locker with adapted lock
Review of directions
Review sessions
Use of mnemonics
Have student restate information
Provision of notes or outlines
Concrete examples
Adaptive writing utensils
Support auditory presentations with visuals
Weekly or bi-weekly home-school communication
Space for movement or breaks
Study sheets and teacher outlines
Extra visual and verbal cues and prompts
Lab and math sheets with highlighted instructions
Graph paper to assist in organizing or lining up math problems
Use of tape recorder for lectures
Use of Interactive notebook
Use of a scribe
Use of computers and calculators
Books on tape
Graphic organizers
Quiet corner or room to calm down and relax when anxious
Preferential seating
Alteration of the classroom arrangement
Reduction of distractions
Answers to be dictated
Hands-on activities
Ability to type all assignments
Follow a routine/schedule
Teach time management and organizational skills
Rest breaks
Verbal and visual cues regarding directions and staying on task
Agenda book and checklists
Daily check-in with case manager or special education teacher
Adjusted assignment timelines
Visual daily schedule

Varied reinforcement procedures
Immediate feedback
Work-in-progress check
Personalized examples

TESTING AND ASSESSMENT ACCOMMODATIONS AND MODIFICATIONS:

Answers to be dictated
Frequent rest breaks
Additional time
Oral testing
Choice of test format (multiple-choice, essay, true-false)
Alternate ways to evaluate (projects or oral presentations instead of written tests)
Accept short answers
Open-book or open-note tests
Read test and directions to student
Provide study guides prior to tests
Highlight key directions
Test in alternative site
Use of calculator
Extra credit option
Pace long-term projects
Preview test procedures
Simplified test wording; rephrased test questions and/or directions

GENERAL ACCOMMODATIONS AND MODIFICATIONS:

Allow outlining, instead of writing for an essay or major project
Use of alternative books or materials on the topic being studied
Computerized spell-check support
Word bank of choices for answers to test questions
Provision of calculator for math tests
Film or video supplements in place of reading text
Reworded questions in simpler language
Projects instead of written reports
Highlighting important words or phrases in reading assignments
Modified workload or length of assignments/tests
Modified time demands

Pass/no pass option
Modified grades based on IEP

BEHAVIOR ACCOMMODATIONS FOR STUDENTS:

Breaks between tasks
Cue expected behavior
Daily feedback to student
Have contingency plans
Use de-escalating strategies
Use positive reinforcement
Use proximity/touch control
Use peer supports and mentoring
Model expected behavior by adults
Have parent sign homework
Set and post class rules
Chart progress and maintain data



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Accommodations and Modifications for ESL Students

Give tests orally
Give instructions/directions in writing and orally
Assign a peer tutor, same language or English speaking
Allow use of bilingual/translation dictionary
Provide summaries of text or study guides of particular chapters
Allow errors in speaking
Allow errors in writing
Accept writing in first language
Highlight key vocabulary
Reduce amount of work required
Assess comprehension through demonstration or other alternative means (gestures, drawings)
Rephrase questions, directions, and explanations
Reduce multiple choices on assessments
Provide study guides and/ or outlines
Shorten reading assignments
Allow extended time for test/project completion
Use of audio books
Use of graphic organizers
Use reduced text, so that print is not dense
Use visuals/pictures throughout lesson/unit/assignemt
Adapt homework to reflect language proficiency/home support
Adapt homework to reflect home support
Provide hands-on activities and explanations
Allow extended time to answer questions, and permit drawing, as an explanation
Use of translation dictionaries to locate words in the native language
Use anecdotal records and portfolio of work as forms of assessment
Use story retellings to assess comprehension