UTeach Computer Science

AP Computer Science A
Syllabus and Planning Guide

2025–2026

Table of Contents

Curricular Requirements

Curricular Requirements		Pages
CR-1	Students and teachers have access to a college-level computer science textbook or resource in print or electronic format.	4
CR-2	The course provides opportunities to develop student understanding of the required content outlined in each unit described in the AP Course and Exam Description (CED).	7
CR-3	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 1: Design Code, as outlined in the AP Course and Exam Description (CED).	5—6
CR-4	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 2: Develop Code, as outlined in the AP Course and Exam Description (CED).	6
CR-5	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 3: Analyze Code, as outlined in the AP Course and Exam Description (CED).	6
CR-6	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 4: Document code and computing systems, as outlined in the AP Course and Exam Description (CED).	6
CR-7	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 5: Use computers responsibly, as outlined in the AP Course and Exam Description (CED).	6
CR-8	The course provides students with hands-on lab experience to practice programming through designing and implementing computer-based solutions to problems.	16

Curriculum Description

Developers

UTeach AP Computer Science A has been developed as part of a collaborative effort between expert faculty from The University of Texas at Austin, successful secondary CS teachers, and UTeach's 25+ years of experience leading nationwide, evidence-based teacher preparation.

UTeach Computer Science provides project-based, student-centered curricula that mirrors the evolving computing industry and AP requirements. Partner teachers receive comprehensive resources, including classroom-ready lesson plans, intentionally scaffolded activities and unit projects, AP-style formative and summative assessments, auto-grading, code feedback tools, academic integrity features, and learner behavior insights. The customizable curriculum includes embedded teacher guidance, timely support videos, and on-demand coaching to empower educators with the instructional strategies needed to be successful in a variety of high school learning environments. UTeach CS continuously refines the program to enhance student learning experiences, build teacher capacity, address teachers' evolving needs, and broaden access to high-quality computing education.

Curriculum Overview

UTeach AP Computer Science A is a year-long high school curriculum that fully addresses the course content as specified by the College Board's AP Computer Science A Course and Exam Description, including all topics, learning objectives, essential knowledge statements, computational thinking practices and skills, and sequenced curriculum units.

This rigorous project-based curriculum allows students to explore programming concepts in depth while collaborating to solve a series of real-world challenges. Each unit introduces a unique interplanetary mission where students tackle hands-on problems as they "travel through space." Units have been thoughtfully scaffolded to guide students through an introduction to the Java programming language and the larger study of object-oriented programming, a style of programming that uses classes and objects mimicking the organization of the real world. Lessons are structured so that subsequent units build on knowledge and skills from previous units, preparing students for the spring College Board AP Computer Science A exam and potential CS career pathways.

Content used in this curriculum incorporate Project-Based Learning (PBL), an evidence-based pedagogical approach that boosts academic achievement and leads to higher learning outcomes, increased student engagement, improved teacher satisfaction, and development of 21st-century competencies. This learner-focused approach encourages students to explore the advantages and societal impact of computing while developing programming and computational thinking skills through Java.

Textbook

[CR-1] The interactive online textbook is an essential component of the UTeach CS A curriculum for both students and teachers**. *It is required for students to have daily access to the internet*.** The textbook, assessments, projects, and built-in programming environments are hosted on the cloud-based Codio platform, which can be integrated with most learning management systems. Codio is FERPA compliant, GDPR compliant, and meets the Web Content Accessibility Guidelines (WCAG) 2.1 Standard at Level AA. Codio provides a built-in dyslexia support feature and supports screen readers for text-based programming languages such as Java and Python. Codio is actively committed to maintaining and continuously improving the accessibility of the Codio experience.

Programming Language Requirements

Students use the Java programming language throughout the curriculum activities and assignments.

AP Classroom

The course guides teachers to assign corresponding 'Personal Progress Checks' in AP Classroom for topic reinforcement and additional content practice.

Resources and Technical Requirements

Aside from the license, which may be purchased or provided through a grant, UTeach AP Computer Science A does not require additional materials for implementation.

Websites Accessed Throughout the Curriculum

The websites below will be accessed in this curriculum. Please advise the school IT department about these websites prior to beginning the course. Always test site access from your device(s) and students' devices to ensure a smooth lesson.

**.codio.com, *. codio.io, https://uteachcs.github.io/CSA-CSS/css/styles.css,*

youtube.com, apcentral.collegeboard.org, docs.oracle.com, codingbat.com

Bibliography

Parlante, Nick. (2017) *CodingBat Java*, CodingBat. https://codingbat.com/java

Touretzky, D. S. (2013). Chapter 8: Recursion. *COMMON LISP: A gentle introduction to symbolic computation* (pp. 281–286). Mineola, NY: Dover Publications.

Course Framework

The course framework consists of two components: 1) Computational Thinking Practices & Skills and 2) Course Content, which includes topics, learning objectives, and essential knowledge statements.

Computational Thinking Practices

The following Computational Thinking Practices are addressed throughout the course in various student assignments. Skill development is sequenced and scaffolded appropriately to allow students to master these skills.

Practice 1: Design Code
- 1.A: Determine an appropriate program design to solve a problem or accomplish a task.
- 1.B: Determine what knowledge can be extracted from data.

Practice 2: Develop Code
- 2.A: Write program code to implement an algorithm.
- 2.B: Write program code involving data abstractions.
- 2.C: Write program code involving procedural abstractions.

Practice 3: Analyze Code
- 3.A: Determine the result or output based on statement execution order in an algorithm.
- 3.B: Determine the result or output based on code that contains data abstractions.
- 3.C: Determine the result or output based on code that contains procedural abstractions.
- 3.D: Explain why a code segment will not compile or work as intended and modify the code to correct the error.

Practice 4: Document Code and Computing Systems
- 4.A: Describe the behavior of a code segment or program.
- 4.B: Describe the initial conditions that must be met for a code segment to work as intended or described.

Practice 5: Use Computers Responsibly
- 5.A: Explain how computing impacts society, economy, and culture.

The following are examples in the curriculum of an instructional approach or activity that describes how students will engage with these skills:

CR-3: Skill 1A: In Unit 3, students design a class by following a structured approach to planning and representing its attributes and behaviors. Students are expected to:
- **Choose a class**: Select a real-world object or concept that can be represented as a class.
- **Identify the purpose of the class**: Write a brief description of what the class represents and its role in a larger program.
- **Determine the attributes (data abstractions)**: List the attributes the class will need to store. For each attribute, include its name, data type, and a brief description.

- **Identify the behaviors (procedural abstractions)**: List at least three methods (behaviors) and describe its purpose, inputs (parameters), and outputs (return type).
- **Represent their design:** Represent the class design as both natural language and class diagram.
- **Write the code for the class:** Using the class design from a previous assignment, students write the Java code for the class.

CR-4: Skill 2A: In Unit 2, students create programs to implement each of the following algorithms using selection and iteration:

- **Checking Divisibility**: Determine if an integer is evenly divisible by another integer.
- **Identifying Individual Digits:** Using iteration, isolate each individual digit of an input number.
- **Determining Frequency**: Using selection, determine how often a specific criterion is met.
- **Computing Sum and Average**: Use iteration to determine the sum and average of a set of numbers.
- **Finding Minimum or Maximum**: Use selection statements to determine the minimum and maximum value in a series of numbers.

CR-5: Skill 3B: In Unit 4, students work through a series of tasks adding code statements that work with `ArrayList` methods and generic `ArrayList` objects and determine the output.

CR-6: Skill 4A: In Unit 1, students are provided with a program to compute the length of the hypotenuse of a right triangle using the Pythagorean theorem. Students must document what each segment of code does by explaining the purpose of the variables, the logic of the program, and the flow of execution. In Unit 4, students are asked to trace the code for two provided recursive methods and determine if they recognize the common mathematical algorithms the two methods perform—calculating the factorial of an input value and the Fibonacci sequence.

CR-7: Skill 5A: In Unit 3, students work in groups to conduct research on one of the following topics to explore how computing systems impact society and ethics:

- **Privacy**: Data collection, surveillance, and personal information security
- **Artificial intelligence:** Bias in algorithms and ethical decision-making
- **Environmental impact:** Energy consumption and electronic waste
- **Accessibility:** Inclusive design for individuals with disabilities
- **Employment:** Automation and its effects on jobs

For their assigned topic, students are to:

- Describe the computing system or technology involved
- Explain the social or ethical challenges related to this technology
- Provide real-world examples or case studies to support their points
- Reflect on the broader implications of the researched topic

Course Content

The course includes the required content organized into the following units based on the AP Course and Exam Description:

Curriculum Unit Overviews [CR-2]
Unit 1: Using Objects and Methods Unit 1 introduces students to fundamental programming concepts in Java, starting with algorithms, programming structures, and the role of compilers. Students learn about variables, data types, expressions, and input/output operations, including assignment statements and type casting. They explore operators, including compound assignment, and gain experience using APIs and libraries. Emphasis is placed on writing clear, well-documented code with comments. The unit covers method signatures, calling class and instance methods, and working with Java's Math class. Students also develop an understanding of objects as instances of classes, focusing on object creation, storage, and string manipulation techniques.
Unit 2: Selection and Iteration Unit 2 explores algorithmic decision-making and repetition using selection and iteration structures in Java. Students learn to construct and evaluate Boolean expressions, using if and nested if statements to control program flow. They develop compound Boolean expressions and compare them for efficiency and correctness. The unit introduces looping structures, including `while` and `for` loops, enabling students to implement selection and iteration algorithms. They apply these concepts to string manipulation and nested iteration, analyzing their efficiency through informal run-time analysis to understand performance trade-offs in algorithm design.
Unit 3: Class Creation Unit 3 focuses on abstraction and effective program design, emphasizing how design choices impact readability, maintainability, and efficiency. Students analyze the structure of a Java class, learning about constructors, instance and class variables, and method creation. They explore how methods pass and return object references, understanding the implications for memory and program behavior. Concepts of scope and access control are introduced to manage data security and encapsulation. The `this` keyword is examined as a tool for distinguishing instance variables from parameters, reinforcing best practices in object-oriented programming.
Unit 4: Data Collections Unit 4 explores data structures, algorithms, and ethical considerations in computing. Students examine the societal impact of data collection and learn to work with datasets, including reading from text files. They develop proficiency with arrays and `ArrayList` objects, implementing traversal techniques and algorithms for data manipulation. The unit introduces two-dimensional arrays, reinforcing multi-level data organization and processing. Students study fundamental searching and sorting algorithms, comparing their efficiency. Recursion is introduced as a powerful problem-solving approach, culminating in recursive implementations of searching and sorting algorithms to deepen algorithmic thinking.

Unit 1: Using Objects and Methods

Unit 1 introduces fundamental Java programming concepts, including algorithms, programming structures, variables, data types, input/output operations, and type casting. Students learn to write clear, well-documented code, use APIs, and work with class and instance methods, the Math class, and string manipulation. They also explore object-oriented principles, such as object creation, storage, and understanding objects as class instances.

Computational Thinking Skills addressed:

- 2.A Write program code to implement an algorithm.
- 2.B Write program code involving data abstractions.
- 3.B Determine the result or output based on code that contains data abstractions.
- 3.C Determine the result of output based on code that contains procedural abstractions.
- 3.D Explain why a code segment will not compile or work as intended and modify the code to correct the error.
- 4.A Describe the behavior of a code segment or program.
- 4.B Describe the initial conditions that must be met for a code segment to work as intended or described.

Unit 1 Schedule (20 Days)

Assignment #	Topic	Learning Objectives
1.1	Introduction to Algorithms, Programming, and Compilers	1.1.A Represent patterns and algorithms found in everyday life using written language or diagrams. 1.1.B Explain the code compilation and execution process. 1.1.C Identify types of programming errors.
1.2	Variables and Data Types	1.2.A Identify the most appropriate data type category for a particular specification. 1.2.B Develop code to declare variables to store numbers and Boolean values.
1.3	Expressions and Output	1.3.A Develop code to generate output and determine the result that would be displayed. 1.3.B Develop code to utilize string literals and determine the result of using string literals. 1.3.C Develop code for arithmetic expressions and determine the result of these expressions.
1.4	Assignment Statements and Input	1.4.A Develop code for assignment statements with expressions and determine the value that is stored in the variable as a result of these statements. 1.4.B Develop code to read input.

Assignment #	Topic	Learning Objectives
1.5	Casting and Range of Variables	1.5.A Develop code to cast primitive values to different primitive types in arithmetic expressions and determine the value that is produced as a result. 1.5.B Describe conditions when an integer expression evaluates to a value out of range. 1.5.C Describe conditions that limit accuracy of expressions.
1.6	Compound Assignment Operators	1.6.A Develop code for assignment statements with compound assignment operators and determine the value that is stored in the variable as a result.
1.7	Application Program Interface (API) and Libraries	1.7.A Identify the attributes and behaviors of a class found in the libraries contained in an API.
1.8	Documentation with Comments	1.8.A Describe the functionality and use of code through comments.
1.9	Method Signatures	1.9.A Identify the correct method to call based on documentation and method signatures. 1.9.B Describe how to call methods.
1.10	Calling Class Methods	1.10.A Develop code to call class methods and determine the result of those calls.
1.11	Math Class	1.11.A Develop code to write expressions that incorporate calls to built-in mathematical libraries and determine the value that is produced as a result.
1.12	Objects: Instances of Classes	1.12.A Explain the relationship between a class and an object. 1.12.B Develop code to declare variables to store reference types.
1.13	Object Creation and Storage (Instantiation)	1.13.A Identify, using its signature, the correct constructor being called. 1.13.B Develop code to declare variables of the correct types to hold object references. 1.13.C Develop code to create an object by calling a constructor.
1.14	Calling Instance Methods	1.14.A Develop code to call instance methods and determine the result of these calls.
1.15	String Manipulation	1.15.A Develop code to create string objects and determine the result of creating and combining strings. 1.15.B Develop code to call methods on string objects and determine the result of calling these methods.
	Unit 1 Review	
	Unit 1 Test	

Unit 2: Selection and Iteration

Unit 2 focuses on algorithmic decision-making and repetition using selection and iteration structures in Java, such as `if` statements, `while` loops, and `for` loops. Students construct and evaluate Boolean expressions, develop efficient compound expressions, and analyze performance trade-offs through informal run-time analysis. These concepts are applied to string manipulation and nested iteration to design and evaluate algorithms.

Computational Thinking Skills addressed:

- 2.A Write program code to implement an algorithm.
- 2.B Write program code involving data abstractions.
- 3.A Determine the result or output based on statement execution order in an algorithm.
- 3.D Explain why a code segment will not compile or work as intended and modify the code to correct the error.

Unit 2 Schedule (18 Days)

Assignment #	Topic	Learning Objectives
2.1	Algorithms with Selection and Repetition	2.1.A Represent patterns and algorithms that involve selection and repetition found in everyday life using written language or diagrams.
2.2	Boolean Expressions	2.2.A Develop code to create Boolean expressions with relational operators and determine the result of these expressions.
2.3	`if` Statements	2.3.A Develop code to represent branching logical processes by using selection statements and determine the result of these processes.
2.4	Nested `if` Statements	2.4.A Develop code to represent nested branching logical processes and determine the result of these processes.
2.5	Compound Boolean Expressions	2.5.A Develop code to represent compound Boolean expressions and determine the result of these expressions.
2.6	Comparing Boolean Expressions	2.6.A Compare equivalent Boolean expressions. 2.6.B Develop code to compare object references using Boolean expressions and determine the result of these expressions.
2.7	`while` Loops	2.7.A Identify when an iterative process is required to achieve a desired result. 2.7.B Develop code to represent iterative processes using `while` loops and determine the result of these processes.
2.8	`for` Loops	2.8.A Develop code to represent iterative processes using `for` loops and determine the result of these processes.
2.9	Implementing Selection and Iteration Algorithms	2.9.A Develop code for standard and original algorithms (without data structures) and determine the result of these algorithms.

Assignment #	Topic	Learning Objectives
2.10	Implementing String Algorithms	2.10.A Develop code for standard and original algorithms that involve strings and determine the result of these algorithms.
2.11	Nested Iteration	2.11.A Develop code to represent nested iterative processes and determine the result of these processes.
2.12	Informal Run-Time Analysis	2.12.A Calculate statement execution counts and informal run-time comparison of iterative statements.
	Unit 2 Review	
	Unit 2 Test	

Unit 3: Class Creation

Unit 3 emphasizes abstraction and program design, highlighting how design choices affect readability, maintainability, and efficiency. Students learn about Java class structure, including constructors, variables, methods, scope, and access control for encapsulation. They explore passing and returning object references, memory implications, and the use of the `this` keyword to distinguish instance variables from parameters.

Computational Thinking Skills addressed:

- 1.A Determine an appropriate program design to solve a problem or accomplish a task.
- 2.C Write program code involving procedural abstractions.
- 3.C Determine the result or output based on code that contains procedural abstractions.
- 5.A Explain how computing impacts society, economy, and culture.

Unit 3 Schedule (12 Days)

Assignment #	Topic	Learning Objectives
3.1	Abstraction and Program Design	3.1.A Represent the design of a program by using natural language or creating diagrams that indicate the classes in the program and the data and procedural abstractions found in each class by including all attributes and behaviors.
3.2	Impact of Program Design	3.2.A Explain the social and ethical implications of computing systems.
3.3	Anatomy of a Class	3.3.A Develop code to designate access and visibility constraints to classes, data, constructors, and methods.
3.4	Constructors	3.4.A Develop code to declare instance variables for the attributes to be initialized in the body of the constructors of a class.
3.5	Methods: How to Write Them	3.5.A Develop code to define behaviors of an object through methods written in a class using primitive values and determine the result of calling these methods.
3.6	Methods: Passing and Returning References of an Object	3.6.A Develop code to define behaviors of an object through methods written in a class using object references and determine the result of calling these methods.
3.7	Class Variables and Methods	3.7.A Develop code to define behaviors of a class through class methods. 3.7.B Develop code to declare the class variables that belong to the class.
3.8	Scope and Access	3.8.A Explain where variables can be used in the code.
3.9	`this` Keyword	3.9.A Develop code for expressions that are self-referencing and determine the result of these expressions.

Assignment #	Topic	Learning Objectives
	Unit 3 Review	
	Unit 3 Test	

Unit 4: Data Collections

This unit covers data structures, algorithms, and ethical considerations in computing, emphasizing the societal impact of data collection. Students work with datasets, arrays, `ArrayLists` objects, and two-dimensional arrays, implementing traversal, searching, and sorting algorithms while analyzing their efficiency. This unit also introduces recursion as a problem-solving tool and culminates with a look at recursive implementations of searching and sorting algorithms to enhance algorithmic thinking.

Computational Thinking Skills addressed:

- 1.B Determine what knowledge can be extracted from data.
- 2.A Write program code to implement an algorithm.
- 2.B Write program code involving data abstractions.
- 3.B Determine the result or output based on code that contains data abstractions.

Unit 4 Schedule (26 Days)

Assignment #	Topic	Learning Objectives
4.1	Ethical and Social Issues Around Data Collection	4.1.A Explain the risks to privacy from collecting and storing personal data on computer systems. 4.1.B Explain the importance of recognizing data quality and potential issues when using a data set. 4.1.C Identify an appropriate data set to use in order to solve a problem or answer a specific question.
4.2	Introduction to Using Data Sets	4.2.A Represent patterns and algorithms that involve data sets found in everyday life using written language or diagrams.
4.3	Array Creation and Access	4.3.A Develop code used to represent collections of related data using one-dimensional (1D) array objects.
4.4	Array Traversals	4.4.A Develop code used to traverse the elements in a 1D array and determine the result of these traversals.
4.5	Implementing Array Algorithms	4.5.A Develop code for standard and original algorithms for a particular context or specification that involves arrays and determine the result of these algorithms.
4.6	Using Text Files	4.6.A Develop code to read data from a text file.
4.7	Wrapper Classes	4.7.A Develop code to use `Integer` and `Double` objects from their primitive counterparts and determine the result of using these objects.
4.8	`ArrayList` Methods	4.8.A Develop code for collections of related objects using `ArrayList` objects and determine the result of calling methods

Assignment #	Topic	Learning Objectives
		on these objects.
4.9	`ArrayList` Traversals	4.9.A Develop code used to traverse the elements of an `ArrayList` and determine the results of these traversals.
4.10	Implementing `ArrayList` Algorithms	4.10.A Develop code for standard and original algorithms for a particular context or specification that involve `ArrayList` objects and determine the result of these algorithms.
4.11	2D Array Creation and Access	4.11.A Develop code used to represent collections of related data using two-dimensional (2D) array objects.
4.12	2D Array Traversals	4.12.A Develop code used to traverse the elements in a 2D array and determine the result of these traversals.
4.13	Implementing 2D Array Algorithms	4.13.A Develop code for standard and original algorithms for a particular context or specification that involves 2D arrays and determine the result of these algorithms.
4.14	Searching Algorithms	4.14.A Develop code used for linear search algorithms to search for specific information in a collection and determine the results of executing a search.
4.15	Sorting Algorithms	4.15.A Determine the result of executing each step of sorting algorithms to sort the elements of a collection.
4.16	Recursion	4.16.A Determine the result of calling recursive methods.
4.17	Recursive Searching and Sorting	4.17.A Determine the result of executing recursive algorithms that use strings or collections. 4.17.B Determine the result of each iteration of a binary search algorithm used to search for information in a collection. 4.17.C Determine the result of each iteration of the merge sort algorithm when used to sort a collection.
	Unit 4 Review	
	Unit 4 Test	

Hands-on Labs

[CR-8] Students spend well over 20 hours throughout the course working on a variety of lab projects, including the following:

Lab #1: Magic Five Lab

Write a program that illustrates a "magic five" number puzzle.

Lab #2: Boolean Detective Lab

Create a program to analyze clues and determine which suspect is guilty in a case involving a stolen artifact. The clues involve logical conditions about suspects, alibis, and evidence.

Lab #3: Virtual Pet Playdate Lab

Develop a program to create and interact with virtual pets. Students will create a **VirtualPet** class with instance variables, a constructor, and methods, as well as a tester class to instantiate objects of the **VirtualPet** class and call the methods.

Lab #4: Analyzing Car Pollution Lab

Complete coding tasks to prepare pollution data (NO2 levels) stored in an array to be analyzed to identify which city intersection has the most measured air pollution.

Lab #5: Seating Chart Lab

Create a 2D array to represent a classroom seating chart for a high school classroom.