

UTeach Computer Science

**AP Computer Science A
Course Syllabus and Planning Guide
2020–2021**

Table of Contents

Curricular Requirements	3
Course Description	4
Course Framework	5
Unit Guides	9
Unit 1: Introductions Are in Order	11
Unit 1 Description & Topics	11
Unit 1 Schedule	12
Unit 2: Primitive Control	13
Unit 2 Description & Topics	13
Unit 2 Schedule	15
Unit 3: Strings and Iteration	17
Unit 3 Description & Topics	17
Unit 3 Schedule	18
Unit 4: Objects, Classes, and Methods	19
Unit 4 Description & Topics	19
Unit 4 Schedule	20
Unit 5: Arrays, ArrayLists, and 2D Arrays	23
Unit 5 Description & Topics	23
Unit 5 Schedule	24
Unit 6: Inheritance	26
Unit 6 Description & Topics	26
Unit 6 Schedule	27
Unit 7: Searching, Sorting, and Recursion	29
Unit 7 Description & Topics	29
Unit 7 Schedule	30
Pedagogical Approaches	31
Resources and Technical Requirements	35

Curricular Requirements

Curricular Requirements		Pages
CR-1	The teacher and students have access to a college-level computer science textbook, in print or electronic format.	4
CR-2	The course provides opportunities to deepen student understanding of the required content outlined in each of the units described in the AP Course and Exam description.	9-10
CR-3	The course provides opportunities to deepen student understanding of the Big Ideas.	7-8
CR-4	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 1: Program Design and Algorithm Development.	5
CR-5	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 2: Code Logic.	5
CR-6	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 3: Code Implementation.	6
CR-7	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 4: Code Testing.	6
CR-8	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 5: Documentation.	6
CR-9	The course provides students with hands-on lab experiences to practice programming through designing and implementing computer-based solutions to problems.	32-33

Course Description

Developers

UTeach Computer Science (uteachcs.org)

UTeach AP Computer Science A has been developed by the UTeach Institute in collaboration with A+ College Ready Alabama.

Course Overview

UTeach AP Computer Science A has been designed as a year-long high school course that fully addresses the big ideas, computational thinking practices and skills, and sequenced curriculum units, as specified by the College Board's *AP Computer Science A* curriculum framework.

The lessons and materials used throughout this course incorporate Project-Based Learning (PBL), a pedagogical approach that actively engages students in the educational process, improves retention, and develops problem solving, critical thinking, and group communication skills. Through this collaborative, learner-centric approach, students are encouraged to explore the advantages and societal impact of computational technology while developing their programming and computational thinking skills through Java.

It is recommended that students have daily access to the Internet. Students are also required to have access to a computer for a minimum of three hours a week.

Course Textbook

[CR-1] *UTeach AP CS A* has an online textbook available for students and teachers. The textbook is hosted by Canvas and is publicly available with no account login or password needed at <https://uteachpd.instructure.com/courses/386/modules>.

Programming Language Requirements

Students will use the Java and Python with Turtle (Unit 1 only) programming languages throughout the course activities and assignments.

Repl.it (www.repl.it)

Repl.it is the preferred programming environment for the course, as it provides a simplified and friendly interface supporting multiple programming languages within a single integrated development environment (IDE). This IDE is freely available online and is platform-independent, so schools and students can run these applications and develop their own programs on any available computer without having to purchase any additional software or licenses.

Course Framework

The course framework consists of two components: 1) Computational Thinking Practices, and 2) Course Content, which includes Big Ideas, Enduring Understandings, Learning Objectives, and Essential Knowledge Statements.

Computational Thinking Practices

Computational Thinking Practices: Skills [CR 4-8]				
1 Program Design and Algorithm Development: Determine required code segments to produce a given output.	2 Code Logic: Determine the output, value, or result of given program code given initial values.	3 Code Implementation: Write and implement program code.	4 Code Testing: Analyze program code for correctness, equivalence, and errors.	5 Documentation: Describe the behavior and conditions that produce identified results in a program.
1.A Determine an appropriate program design to solve a problem or accomplish a task (not assessed).	2.A Apply the meaning of specific operators.	3.A Write program code to create the objects of a class and call methods.	4.A Use test-cases to find errors or validate results.	5.A Describe the behavior of a given segment of program code.
1.B Determine code that would be used to complete code segments.	2.B Determine the result or output based on statement execution order in a code segment without method calls (other than output).	3.B Write program code to define a new type by creating a class.	4.B Identify errors in program code.	5.B Explain why a code segment will not compile or work as intended.
1.C Determine code that would be used to interact with completed program code.	2.C Determine the result or output based on the statement execution order in a code segment containing method calls.	3.C Write program code to satisfy method specifications using expressions, conditional statements and iterative statements.	4.C Determine if two or more code segments yield equivalent results.	5.C Explain how the result of program code changes, given a change to the initial code.
	2.D Determine the number of times a code segment will execute.	3.D Write program code to create, traverse, and manipulate elements in a 1D array or ArrayList objects.		5.D Describe the initial conditions that must be met for a program segment to work as intended or described.
		3.E Write program code to create, traverse, and manipulate elements in 2D array objects.		

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

CR-4: 1.B For most of the programming assignments and the unit projects, students build on starter code in Repl.it using instructions on how to complete the code in order to make a working solution to a given problem which assesses mastery of the topics presented in the lesson.

CR-5: 2.D Students engage in an activity in Unit 7 during the lesson on Compare Big O Informally in which they read through a code segment and determine how many times the data structure is being accessed. Then they open a program with the same code and run the program to compare their prediction with the actual statement execution times.

CR-6: **3.B** The Unit 4 project – Disease Diagnoser and the Unit 6 project – Hospital Locator ask students to create a class and write the code to test the implementation of the class.

3.D **3.E** The Unit 5 project – Air Quality Analyzer has students create, traverse and manipulate data in 1D arrays, ArrayLists, and a 2D array.

CR-7: **4.A** Throughout the programming activities, students are asked to test their program code using various test cases. One example is the Palindrome program that students write in Unit 3 in which students are to test their code using at least 12 different words to check if they are palindromes. For the Unit 7 project in which students write searching and sorting algorithms, students use a small data set for the initial testing and then are given a much larger data set to test the efficiency of their algorithms.

CR-8: **5.B** One of the instructional strategies used as a check for understanding is Thumbs Up/Thumbs Down. Students review a code segment and show a thumbs up if the code will work as intended or thumbs down if the code will not compile or work as intended. The students explain the syntax or logic error for thumbs down code segments. This strategy is used in Unit 2 for conditionals and Unit 3 for iteration.





Course Content - Big Ideas




The big ideas serve as the foundation of the course. They are overarching concepts or themes that are spiraled throughout all seven of the curriculum units and connected to the topics and activities within each unit.

Big Ideas [CR-3]	
<p>Big Idea 1: MODULARITY (MOD) Incorporating elements of abstraction, by breaking problems down into interacting pieces, each with their own purposes, makes writing complex programs easier. Abstracting simplifies concepts and processes by looking at the big picture rather than being overwhelmed by the details. Modularity in object-oriented programming allows us to use abstraction to break complex programs down into individual classes and methods.</p>	<p>Along with several smaller programming assignments within Unit 6, the unit 6 project – Hospital Locator - allows students to demonstrate mastery of the Modularity Big Idea by constructing a database consisting of a superclass and several subclasses and populating the database with real-world data and searching the data.</p>
<p>Big Idea 2: VARIABLES (VAR) Information used as a basis for reasoning, discussion, or calculation is referred to as <i>data</i>. Programs rely on variables to store data, on data structures to organize multiple values when program complexity increases, and on algorithms to sort, access, and manipulate this data. Variables create data abstractions, as they can represent a set of possible values or a group of related values.</p>	<p>Along with several smaller programming assignments within Unit 5, the Unit 5 project – Air Quality Analyzer - allows students to demonstrate mastery of the Variables Big Idea by analyzing real-world data stored in each of the 3 data structures: 1D array, ArrayList, and 2D array.</p>
<p>Big Idea 3: CONTROL (CON) Doing things in order; making decisions, and doing the same process multiple times are represented in code by using control structures and specifying the order in which instructions are executed. Programmers need to think algorithmically in order to define and interpret processes that are used in a program.</p>	<p>Students will have many opportunities to demonstrate the use of control structures throughout the units. Some example activities are:</p> <ul style="list-style-type: none"> • In Unit 2, students write a program using conditionals and nested if statements in order to create an interactive story using user input • In Unit 3, students create iterative algorithms such as reversing all the characters in a word and taking the average of all digits in a number.
<p>Big Idea 4: IMPACT OF COMPUTING (IOC) Computers and computing have revolutionized our lives. To use computing safely and responsibly, we need to be aware of privacy, security, and ethical issues. As programmers, we need to understand how our programs will be used and be responsible for the consequences.</p>	<p>In Unit 1, students research and report on computing innovations, then discuss as a class the legal issues and intellectual property concerns surrounding the development and creation of programs and software. In Unit 4, students discuss the importance of system reliability and how programmers should make every effort to maximize system reliability and how using classes and objects (object-oriented programming) can help with this.</p>

	<p>In Unit 5, students discuss the ethical and security issues of gathering and storing personal information such as names and birthdates.</p>
--	--

Unit Guides

Course Units [CR-2]		
<p>Unit 1: Introductions Are in Order</p> <p>Unit 1 is a gentle introduction to the AP Computer Science A course and text-based programming languages. The programming activities in Unit 1 use the Python with Turtle graphical programming language and Repl.it is the online IDE used throughout the course.</p>	<p>Big Ideas: Modularity Control Impact of Computing</p> <p>Enduring Understandings: MOD-1, MOD-2, CON-2, IOC-1</p>	<p>Computational Thinking Practices:</p>  <p>Aligned to College Board CED Units: None-this unit is an introduction to the UTeach CSA course narrative and using a text-based programming language.</p>
<p>Unit 2: Primitive Control</p> <p>This unit provides an introduction to object-oriented programming, primitive data types in Java, mathematical expressions and operators, Boolean expressions, conditionals, using objects, and the Math class.</p>	<p>Big Ideas: Modularity Variables Control</p> <p>Enduring Understandings: MOD-1, VAR-1, CON-1, CON-2</p>	<p>Computational Thinking Practices:</p>  <p>Aligned to College Board CED Units: 1, 2 and 3</p>
<p>Unit 3: Strings and Iteration</p> <p>Unit 3 concentrates on the String class and the String class methods. This unit introduces many of the most common string manipulating algorithms as well as iteration and the Java control structures while loop and for loop that allow for the use of repetition in programs.</p>	<p>Big Ideas: Modularity Variables Control</p> <p>Enduring Understandings: MOD-1, VAR-1, CON-2</p>	<p>Computational Thinking Practices:</p>  <p>Aligned to College Board CED Units: 2 and 4</p>
<p>Unit 4: Objects, Classes, and Methods</p> <p>Unit 4 takes a deep dive into Object-Oriented Programming by introducing all the components of a class including private instance variables, constructors, accessor methods, mutator methods, and helper methods. This unit also covers accessibility and the different categories of methods from static and non-static to void and non-void.</p>	<p>Big Ideas: Modularity Variables Impact of Computing</p> <p>Enduring Understandings: MOD-1, MOD-2, MOD-3, VAR-1, IOC-1</p>	<p>Computational Thinking Practices:</p>  <p>Aligned to College Board CED Unit: 5</p>

<p>Unit 5: Arrays, ArrayLists, and 2D Arrays</p> <p>Unit 5 introduces 3 different data structures used to store data in Java: the array, ArrayList, and 2D array, and the syntax and traversal methods for the most common algorithms for processing data.</p>	<p>Big Ideas: Variables Control Impact of Computing</p> <p>Enduring Understandings: VAR-1, VAR-2, CON-2, IOC-1</p>	<p>Computational Thinking Practices:</p>  <p>Aligned to College Board CED Units: 6, 7 and 8</p>
<p>Unit 6: Inheritance</p> <p>Unit 6 takes a comprehensive look at two relationships between classes in Java: inheritance and composition. Topics covered in depth are superclasses and subclasses, constructors of the subclasses, overriding methods, composition, the Object class, and polymorphism.</p>	<p>Big Ideas: Modularity</p> <p>Enduring Understandings: MOD-2, MOD-3</p>	<p>Computational Thinking Practices:</p>  <p>Aligned to College Board CED Unit: 9</p>
<p>Unit 7: Searching, Sorting, and Recursion</p> <p>This unit covers different search algorithms, the selection sort and insertion sort, Big O, recursion, recursive search and sort algorithms.</p>	<p>Big Ideas: Control</p> <p>Enduring Understanding: CON-2</p>	<p>Computational Thinking Practices:</p>  <p>Aligned to College Board CED Units: 6, 7 and 10</p>

NOTES: The following topics are taught throughout the course as appropriate:

- Exception (CED Units 2, 6, 8)
- Ethical and Social Implications (CED Units 5, 7)

Unit 1: Introductions Are in Order

Unit 1 is designed to be a gentle introduction to text-based programming using Python and Python with Turtle as a graphical language. Throughout Unit 1, students master basic coding tools and concepts, such as algorithms and abstraction, which can be easily applied in later units using Java.

The Big Ideas of Modularity, Control, and Impact of Computing are all covered in Unit 1 using Python. These are the Enduring Understandings covered in this unit:

- MOD-1: Some objects or concepts are so frequently represented that programmers can draw upon existing code that has already been tested, enabling them to write solutions more quickly and with a greater degree of confidence.
- MOD-2: Programmers use code to represent a physical object or nonphysical concept, real or imagined, by defining a class based on the attributes and/or behaviors of the object or concept.
- CON-2: Programmers incorporate iteration and selection into code as a way of providing instructions for the computer to process each of the many possible input values.
- IOC-1: While programs are typically designed to achieve a specific purpose, they may have unintended consequences.

Unit 1 Topics

Over the course of this unit and the unit project, students will learn to:

- Collaborate with classmates and establish group norms.
- Use an online IDE to create programs.
- Write code to draw graphics using Python with Turtle.
- Write an algorithm.
- Implement an algorithm in Python code.
- Debug a program through error detection and testing.
- Use abstraction by incorporating built-in functions from Python to draw shapes.
- Create modules in a program to aid in readability.
- Comment code to aid in readability.

Unit 1 Schedule (14 Days)

Day	Topic	Lessons	Learning Objectives	Essential Knowledge
U1D1	Unit Project	Avatar Creator Project (1 of 5)	Non-AP	Non-AP
U1D2	Introduction to APCS A	What is Computer Science?	IOC-1.A	IOC-1.A.2, IOC-1.A.3
U1D3		Introduction to APCS A		
U1D4		My First Python Program		
U1D5	Algorithms	Writing Algorithms	CON-2.D, MOD-2.C	MOD-2.C.1
U1D6		Implementing an Algorithm		
U1D7	Abstraction	Draw an Emoji	MOD-1.E	MOD-1.E.2
U1D8	Unit Project	Avatar Creator Project (2 of 5)	CON-2.D, MOD-1.E, MOD-2.C	MOD-1.E.2, MOD-2.C.1
U1D9	Modular Programming	Modular Programming	MOD-1.E	MOD-1.E.2, MOD-1.E.3
U1D10	Unit Project	Avatar Creator Project (3 of 5)	CON-2.D, MOD-1.E, MOD-2.C	MOD-1.E.2, MOD-1.E.3, MOD-2.C.1
U1D11	Unit Test Review	Unit 1 Test Review	None	None
U1D12	Unit Project	Avatar Creator Project (4 of 5)	CON-2.D, MOD-1.E, MOD-2.C	MOD-1.E.2, MOD-1.E.3, MOD-2.C.1
U1D13	Unit Test	Unit 1 Test	None	None
U1D14	Unit Project	Avatar Creator Presentation (5 of 5)	None	None

Unit 2: Primitive Control

Unit 2 covers the College Board Units 1-3. Unit 2 jumps right into programming in Java. Over the course of this unit, object-oriented programming, a style of programming that uses classes and objects, is introduced. Also, primitive data types in Java, mathematical expressions operators, Boolean expressions, conditionals, using objects, and a few of the Math class methods are studied.

The Big Ideas covered in this unit are Modularity, Variables, and Control. These are the Enduring Understandings covered in this unit:

- MOD-1: Some objects or concepts are so frequently represented that programmers can draw upon existing code that has already been tested, enabling them to write solutions more quickly and with a greater degree of confidence.
- VAR-1: To find specific solutions to generalizable problems, programmers include variables in their code so that the same algorithm runs using different input values.
- CON-1: The way variables and operators are sequenced and combined in an expression determines the computed result.
- CON-2: Programmers incorporate iteration and selection into code as a way of providing instructions for the computer to process each of the many possible input values.

These are the Computational Thinking Skills addressed:

- 1.A Determine an appropriate program design to solve a problem or accomplish a task.
- 1.B Determine code that would be used to complete code segments.
- 1.C Determine code that would be used to interact with completed program code.
- 2.A Apply the meaning of specific operators.
- 2.B Determine the result or output based on statement execution order in a code segment without method calls (other than output).
- 2.C Determine the result or output based on the statement execution order in a code segment containing method calls.
- 3.A Write program code to create objects of a class and call methods.
- 3.C Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.
- 4.A Use test-cases to find errors or validate results.
- 4.B Identify errors in program code.
- 4.C Determine if two or more code segments yield equivalent results.
- 5.A Describe the behavior of a given segment of program code.
- 5.B Explain why a code segment will not compile or work as intended.

Unit 2 Topics

Our Unit 2 covers the College Boards Units 1, 2, and 3. Over the course of this unit and the unit project, the following topics will be covered and teachers may use these in the AP Classroom to filter questions for formative and summative assessments:

- 1.1 Why Programming? Why Java?
- 1.2 Variables and Data Types
- 1.3 Expressions and Assignment Statements
- 1.4 Compound Assignment Operators
- 1.5 Casting and Ranges of Variables
- 2.1 Objects: Instances of Classes
- 2.2 Creating and Storing Objects (Instantiation)
- 2.3 Calling a Void Method
- 2.4 Calling a Void Method with Parameters
- 2.5 Calling a Non-void Method
- 2.6 `String` Objects: Concatenation, Literals, and More
- 2.7 `String` Methods
- 2.8 `Wrapper` Classes: `Integer` and `Double`
- 2.9 Using the `Math` Class
- 3.1 Boolean Expressions
- 3.2 `if` Statements and Control Flow
- 3.3 `if-else` Statements
- 3.4 `else if` Statements
- 3.5 Compound Boolean Expressions
- 3.6 Equivalent Boolean Expressions
- 3.7 Comparing Objects

Unit 2 Schedule (22 Days)

Day	Topic	Lessons	Learning Objectives	Essential Knowledge
U2D1	Unit Project	Resource Finder Project (1 of 5)	MOD-1.A, VAR-1.A	MOD-1.A.1, MOD-1.A.2, VAR-1.A.1
U2D2	Variables and Data Types	Variable and Data Types	VAR-1.B, VAR-1.C	VAR-1.B.1, VAR-1.B.2, VAR-1.B.3, VAR-1.C.1, VAR-1.C.2, VAR-1.C.3, VAR-1.C.4
U2D3	Expressions and Assignment Statements	Simple Assignment Statements	CON-1.A, CON-1.B	CON-1.A.1, CON-1.A.2, CON-1.A.3, CON-1.A.4, CON-1.A.5, CON-1.A.6, CON-1.A.7, CON-1.A.8, CON-1.B.1, CON-1.B.2, CON-1.B.3
U2D4		Compound Assignment Operators	CON-1.B	CON-1.B.4, CON-1.B.5
U2D5	Unit Project	Resource Finder Project (2 of 5)	CON-1.A, CON-1.B, CON-1.C	CON-1.A.1, CON-1.A.2, CON-1.A.3, CON-1.A.5, CON-1.B.1, CON-1.B.2, CON-1.B.3, CON-1.C.4, CON-1.C.5, CON-1.C.6
U2D6	Using Objects	Overview of Objects and Classes	MOD-1.B, MOD-1.C, MOD-1.D, VAR-1.D	MOD-1.B.1, MOD-1.B.2, MOD-1.C.1, MOD-1.C.2, MOD-1.C.3, MOD-1.C.4, MOD-1.C.5, MOD-1.C.6, MOD-1.D.1, MOD-1.D.2, MOD-1.D.3, MOD-1.D.4, VAR-1.D.1, VAR-1.D.2
U2D7		Calling Methods	MOD-1.E, MOD-1.F, MOD-1.G	MOD-1.E.1, MOD-1.E.2, MOD-1.E.3, MOD-1.E.4, MOD-1.E.5, MOD-1.E.6, MOD-1.E.7, MOD-1.E.8, MOD-1.F.1, MOD-1.F.2, MOD-1.F.3, MOD-1.G.1
U2D8-9	Introduction to Strings	String Objects & Methods	VAR-1.E	VAR-1.E.1, VAR-1.E.2, VAR-1.E.6, VAR-1.E.7, VAR-1.E.8, VAR-1.E.9, VAR-1.E.12
U2D10	Wrapper Classes/Math Class	Wrapper Classes/Math Class	VAR-1.F, CON-1.D, MOD-1.H	VAR-1.F.1, VAR-1.F.2, VAR-1.F.3, VAR-1.F.4, VAR-1.F.5, VAR-1.F.6, VAR-1.F.7, CON-1.D.1, CON-1.D.2, CON-1.D.3, CON-1.D.4, MOD-1.H

U2D11	Boolean Expressions	Truth Tables and Booleans	CON-1.E	CON-1.E.1, CON-1.E.2, CON-1.E.3
U2D12		Compound Boolean Expressions	CON-1.F	CON-1.F.1, CON-1.F.2, CON-1.F.3
U2D13		Equivalent Boolean Expressions	CON-1.G	CON-1.G.1, CON-1.G.2, CON-1.G.3
U2D14	Conditionals	If Statements	CON-2.A	CON-2.A.1, CON-2.A.2, CON-2.A.3
U2D15		If/else Statements	CON-2.A	CON-2.A.4
U2D16		Else/if Statements	CON-2.A, CON-2.B	CON-2.A.5, CON-2.B.1
U2D17	Unit Project	Resource Finder Project (3 of 5)	MOD-1.A, VAR-1.A, VAR-1.B, VAR-1.C, CON-1.A, CON-1.B, CON-1.C, CON-2.A	MOD-1.A.1, VAR-1.A.1, VAR-1.B.1, VAR-1.B.2, VAR-1.B.3, VAR-1.C.1, VAR-1.C.2, VAR-1.C.3, CON-1.A.1, CON-1.A.2, CON-1.A.3, CON-1.A.5, CON-1.B.1, CON-1.B.2, CON-1.B.3, CON-1.C.4, CON-2.A.1, CON-2.A.2, CON-2.A.3
U2D18	Comparing Objects	Comparing Objects	CON-1.H	CON-1.H.1, CON-1.H.2, CON-1.H.3, CON-1.H.4
U2D19	Unit Project	Resource Finder Project (4 of 5)	MOD-1.A, VAR-1.A, VAR-1.B, VAR-1.C, CON-1.A, CON-1.B, CON-1.CON-2.A	MOD-1.A.1, VAR-1.A.1, VAR-1.B.1, VAR-1.B.2, VAR-1.B.3, VAR-1.C.1, VAR-1.C.2, VAR-1.C.3, CON-1.A.1, CON-1.A.2, CON-1.A.3, CON-1.A.5, CON-1.B.1, CON-1.B.2, CON-1.B.3, CON-1.C.4, CON-2.A.1,, CON-2.C.2, CON-2.C.3
U2D20	Unit Test Review	Unit 2 Test Review	None	None
U2D21	Unit Test	Unit 2 Test	None	None
U2D22	Unit Project	Resource Finder Presentation (5 of 5)	None	None

Unit 3: Strings and Iteration

Unit 3 concentrates on the String class and the String class methods that are tested on the AP CSA exam. Manipulating strings is a very common task in programming and students will learn many of the most common string manipulating algorithms. This unit also introduces iteration and the Java control structures that allow for the use of repetition in programs.

The Big Ideas covered in this unit are Modularity, Variables, and Control. These are the Enduring Understandings covered in this unit:

- MOD-1: Some objects or concepts are so frequently represented that programmers can draw upon existing code that has already been tested, enabling them to write solutions more quickly and with a greater degree of confidence.
- VAR-1: To find specific solutions to generalizable problems, programmers include variables in their code so that the same algorithm runs using different input values.
- CON-2: Programmers incorporate iteration and selection into code as a way of providing instructions for the computer to process each of the many possible input values.

These are the Computational Thinking Skills addressed:

- 1.B Determine code that would be used to complete code segments.
- 1.C Determine code that would be used to interact with completed program code.
- 2.A Apply the meaning of specific operators.
- 2.B Determine the result or output based on statement execution order in a code segment without method calls (other than output).
- 2.C Determine the result or output based on the statement execution order in a code segment containing method calls.
- 3.A Write program code to create objects of a class and call methods.
- 3.C Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.
- 4.C Determine if two or more code segments yield equivalent results.
- 5.A Describe the behavior of a given segment of program code.
- 5.C Explain how the result of program code changes, given a change of the initial code.

Unit 3 Topics

Our Unit 3 covers the College Board Unit 4 and a review of Strings. Over the course of this unit and the unit project, the following topics will be covered and teachers may use these in the AP Classroom to filter questions for formative and summative assessments:

- 2.6 String Objects: Concatenation, Literals, and More
- 2.7 String Methods

- 4.1 while Loops
- 4.2 for Loops
- 4.3 Developing Algorithms using Strings
- 4.4 Nested Iteration
- 4.5 Informal Code Analysis

Unit 3 Schedule (19 Days)

Day	Topic	Lessons	Learning Objectives	Essential Knowledge
U3D1	Unit Project	Language Interpreter Project (1 of 6)	None	None
U3D2	Strings	Instantiation, Literals, and More	VAR-1.A, VAR-1.E, MOD-1.G	VAR-1.A.1, VAR-1.E.1, VAR-1.E.2, VAR-1.E.3, VAR-1.E.4, VAR-1.E.5, VAR-1.E.6, VAR-1.E.7, VAR-1.E.8, VAR-1.E.9, VAR-1.E.10, VAR-1.E.11, VAR-1.E.12, VAR-1.E.13, MOD-1.G.1
U3D3-D4		String Methods		
U3D5	Unit Project	Language Interpreter Project (2 of 6)	VAR-1.E, MOD-1.G	VAR-1.E.1, VAR-1.E.3, VAR-1.E.10, VAR-1.E.11, VAR-1.E.12, MOD-1.G.1
U3D6	Iteration	For Loops	CON-2.D, CON-2.E	CON-2.D.1, CON-2.E.1, CON-2.E.2, CON-2.E.3, CON-2.E.5
U3D7-D8		While Loops	CON-2.C, CON-2.D, CON-2.E	CON-2.C.1, CON-2.C.2, CON-2.C.3, CON-2.D.4, CON-2.D.5, CON-2.D.1, CON-2.D.2, CON-2.E.4
U3D9-D10		Nested Iteration	CON-2.G	CON-2.G.1, CON-2.G.2
U3D11	Unit Project	Language Interpreter Project (3 of 6)	VAR-1.E, MOD-1.G	VAR-1.E.1, VAR-1.E.3, VAR-1.E.10, VAR-1.E.11, VAR-1.E.12, MOD-1.G.1
U3D12-D14	Algorithms	Developing Algorithms using Strings	CON-2.F, CON-2.H	CON-2.F.1, CON-2.H.1
U3D15-D16	Unit Project	Language Interpreter Project (4-5 of 6)	VAR-1.E, MOD-1.G	VAR-1.E.1, VAR-1.E.3, VAR-1.E.10, VAR-1.E.11, VAR-1.E.12, MOD-1.G.1
U3D17	Unit Test Review	Unit 3 Test Review	None	None
U3D18	Unit Test	Unit 3 Test	None	None
U3D19	Unit Project	Language Interpreter Presentation (6 of 6)	None	None

Unit 4: Object, Classes and Methods

Unit 4 takes a deep dive into Object-Oriented Programming by introducing all the components of a class including private instance variables, constructors, accessor methods, mutator methods, and helper methods. This unit also covers private vs public access to data and methods, the different categories of methods from static and non-static to void and non-void. Within the unit, students will create entire classes and then instantiate objects of that class.

The Big Ideas covered in this unit are Modularity, Variables, and Impact of Computing. These are the Enduring Understandings covered in this unit:

- MOD-1: Some objects or concepts are so frequently represented that programmers can draw upon existing code that has already been tested, enabling them to write solutions more quickly and with a greater degree of confidence.
- MOD 2: Programmers use code to represent a physical object or nonphysical concept, real or imagined, by defining a class based on the attributes and/or behaviors of the object or concept.
- MOD-3: When multiple classes contain common attributes and behaviors, programmers create a new class containing the shared attributes and behaviors forming a hierarchy. Modifications made at the highest level of the hierarchy apply to the subclasses.
- VAR-1: To find specific solutions to generalizable problems, programmers include variables in their code so that the same algorithm runs using different input values.
- IOC-1: While programs are typically designed to achieve a specific purpose, they may have unintended consequences.

These are the Computational Thinking Skills addressed:

- 1.A Determine an appropriate program design to solve a problem or accomplish a task.
- 1.B Determine code that would be used to complete code segments.
- 1.C Determine code that would be used to interact with completed program code.
- 2.C Determine the result or output based on the statement execution order in a code segment containing method calls.
- 3.B Write program code to define a new type by creating a class.
- 4.B Identify errors in program code.
- 5.A Describe the behavior of a given segment of program code.
- 5.B Explain why a code segment will not compile or work as intended.
- 5.D Describe the initial conditions that must be met for a program segment to work as intended or described.

Unit 4 Topics

Our Unit 4 covers the College Board Unit 5. Over the course of this unit and the unit project, the following topics will be covered and teachers may use these in the AP Classroom to filter questions for formative and summative assessments:

- 5.1 Anatomy of a Class
- 5.2 Constructors
- 5.3 Documentation with Comments
- 5.4 Accessor Methods
- 5.5 Mutator Methods
- 5.6 Writing Methods
- 5.7 Static Variables and Methods
- 5.8 Scope and Access
- 5.9 this Keyword
- 5.10 Ethical and Social Implications of Computing Systems

Unit 4 Schedule (21 Days)

Day	Topic	Lessons	Learning Objectives	Essential Knowledge
U4D1	Unit Project	Disease Diagnoser Project (1 of 6)	None	None
U4D2-D3	Objects	Object-Oriented Programming	MOD-1.B, MOD-2.D, MOD-2.E, MOD-2.F, MOD-3.A, VAR-1.G, IOC-1.A	MOD-1.B.1, MOD-1.B.2, MOD-2.D.1, MOD-2.D.2, MOD-2.D.3, MOD-2.D.4, MOD-2.D.5, MOD-2.D.6, MOD-2.D.7, MOD-2.E.1, MOD-2.E.2, MOD-2.F.1, MOD-2.F.2, MOD-2.F.3, MOD-2.F.4, MOD-2.F.5, MOD-2.F.6, MOD-3.A.1, MOD-3.A.2, MOD-3.A.3, MOD-3.A.4, VAR-1.G.1, VAR-1.G.2, VAR-1.G.3, VAR-1.G.4, IOC-1.A.1
U4D4		The Object Superclass	MOD-3.E	MOD-3.E.1, MOD-3.E.2, MOD-3.E.3, MOD-3.E.4
U4D5-D6		Writing Classes	MOD-2.B, MOD-2.C, MOD-2.D, MOD-2.E, MOD-2.F	MOD-2.B.1, MOD-2.B.2, MOD-2.B.3, MOD-2.B.4, MOD-2.C.1, MOD-2.C.2, MOD-2.C.3, MOD-2.C.4, MOD-2.C.5, MOD-2.D.1, MOD-2.D.2, MOD-2.D.3, MOD-2.D.4, MOD-2.D.5, MOD-2.D.6, MOD-2.D.7, MOD-2.E.1, MOD-2.E.2, MOD-2.F.1, MOD-2.F.2, MOD-2.F.3, MOD-2.F.4, MOD-2.F.5, MOD-2.F.6
U4D7		Creating Instances of Classes	MOD-2.B	MOD-2.B.1, MOD-2.B.2, MOD-2.B.3, MOD-2.B.4, MOD-2.B.5
U4D8		Accessibility (private/public)	MOD-2.A, MOD-3.A	MOD-2.A.1, MOD-2.A.2, MOD-

				2.A.3, MOD-2.A.4, MOD-2.A.5, MOD-2.A.6, MOD-3.A.1, MOD-2.A.2, MOD-3.A.3, MOD-3.A.4
U4D9		Static Variables and Methods	MOD-1.H, MOD-2.G, MOD-2.H	MOD-1.H.1, MOD-2.G.1, MOD-2.G.2, MOD-2.G.3, MOD-2.G.4, MOD-2.G.5, MOD-2.H.1, MOD-2.H.2, MOD-2.H.3
U4D10		this and super	VAR.1.G, VAR.1.H	VAR-1.G.1, VAR-1.G.2, VAR-1.G.3, VAR-1.G.4, VAR-1.H.1, VAR-1.H.2
U4D11-D12	Unit Project	Language Interpreter Project (2-3 of 6)	MOD-2.A, MOD-2.B, MOD-2.D, MOD-2.E, MOD-2.F, MOD-3.A, VAR-1.G, VAR-1.H	MOD-2.A.1, MOD-2.A.2, MOD-2.A.3, MOD-2.A.4, MOD-2.A.5, MOD-2.A.6, MOD-2.B.1, MOD-2.B.2, MOD-2.B.3, MOD-2.B.5, MOD-2.D.1, MOD-2.D.2, MOD-2.D.3, MOD-2.D.4, MOD-2.D.5, MOD-2.D.6, MOD-2.D.7, MOD-2.E.1, MOD-2.E.2, MOD-2.F.1, MOD-2.F.2, MOD-2.F.4, MOD-3.A.3, MOD-3.A.4, VAR-1.G.1, VAR-1.G.2, VAR-1.G.3, VAR-1.H.1
U4D13	Methods	Void Methods	MOD-1.E, MOD-2.E	MOD-1.E.1, MOD-1.E.2, MOD-1.E.3, MOD-1.E.4, MOD-1.E.5, MOD-1.E.6, MOD-1.E.7, MOD-2.E.1
U4D14		Non-Void Methods	MOD.1.G, MOD-2.D, MOD-2.F	MOD-1.G.1, MOD-2.D.2, MOD-2.D.3, MOD-2.D.4, MOD-2.D.5, MOD-2.F.1, MOD-2.F.2, MOD-2.F.3, MOD-2.F.4
U4D15	Unit Project	Disease Diagnoser Project (4 of 6)	MOD-2.A, MOD-2.B, MOD-2.D, MOD-2.E, MOD-2.F, MOD-3.A, VAR-1.G, VAR-1.H	MOD-2.A.1, MOD-2.A.2, MOD-2.A.3, MOD-2.A.4, MOD-2.A.5, MOD-2.A.6, MOD-2.B.1, MOD-2.B.2, MOD-2.B.3, MOD-2.B.5, MOD-2.D.1, MOD-2.D.2, MOD-2.D.3, MOD-2.D.4, MOD-2.D.5, MOD-2.D.6, MOD-2.D.7, MOD-2.E.1, MOD-2.E.2, MOD-2.F.1, MOD-2.F.2, MOD-2.F.4, MOD-3.A.3, MOD-3.A.4, VAR-1.G.1, VAR-1.G.2, VAR-1.G.3, VAR-1.H.1
U4D16-17	Overloaded Methods	Overloaded Methods	MOD-1.C, MOD-1.F	MOD-1.C.4, MOD-1.F.3
U4D18	Unit Project	Disease Diagnoser Project (5 of 6)	MOD-2.A, MOD-2.B, MOD-2.D, MOD-2.E, MOD-2.F, MOD-3.A, VAR-1.G, VAR-1.H	MOD-2.A.1, MOD-2.A.2, MOD-2.A.3, MOD-2.A.4, MOD-2.A.5, MOD-2.A.6, MOD-2.B.1, MOD-2.B.2, MOD-2.B.3, MOD-2.B.5, MOD-2.D.1, MOD-2.D.2, MOD-2.D.3, MOD-2.D.4, MOD-2.D.5, MOD-2.D.6, MOD-2.D.7, MOD-2.E.1, MOD-2.E.2, MOD-2.F.1, MOD-2.F.2, MOD-2.F.4, MOD-3.A.3, MOD-3.A.4, VAR-1.G.1, VAR-1.G.2, VAR-1.G.3, VAR-

				1.H.1
U4D19	Unit Test Review	Unit 4 Test Review	None	None
U4D20	Unit Test	Unit 4 Test	None	None
U4D21	Unit Project	Disease Diagnoser Presentation (6 of 6)	None	None

Unit 5: Arrays, ArrayLists, and 2D Arrays

Unit 5 focuses on three different data structures used to store data in Java: the array, ArrayList, and 2D array. Arrays are static, one-dimensional data structures. ArrayList is a built-in Java class that stores data in a dynamic list and 2D arrays store data in a two-dimensional structure like a table or spreadsheet. Topics include the syntax for creation and access and traversal methods for each data structure and the most common algorithms for processing data.

The Big Ideas covered in this unit are Variables, Control, and Impact of Computing. These are the Enduring Understandings covered in this unit:

- MOD-1: Some objects or concepts are so frequently represented that programmers can draw upon existing code that has already been tested, enabling them to write solutions more quickly and with a greater degree of confidence.
- MOD 2: Programmers use code to represent a physical object or nonphysical concept, real or imagined, by defining a class based on the attributes and/or behaviors of the object or concept.
- MOD-3: When multiple classes contain common attributes and behaviors, programmers create a new class containing the shared attributes and behaviors forming a hierarchy. Modifications made at the highest level of the hierarchy apply to the subclasses.
- VAR-1: To find specific solutions to generalizable problems, programmers include variables in their code so that the same algorithm runs using different input values.
- VAR-2: To manage large amounts of data or complex relationships in data, programmers write code that groups the data together into a single data structure without creating individual variables for each value.
- CON-2: Programmers incorporate iteration and selection into code as a way of providing instructions for the computer to process each of the many possible input values.
- IOC-1: While programs are typically designed to achieve a specific purpose, they may have unintended consequences.

These are the Computational Thinking Skills addressed:

- 1.B Determine code that would be used to complete code segments.
- 1.C Determine code that would be used to interact with completed program code.
- 2.B Determine the result or output based on statement execution order in a code segment without method calls (other than output).
- 2.C Determine the result or output based on the statement execution order in a code segment containing method calls.
- 3.D Write program code to create, traverse, and manipulate elements in 1D array or ArrayList objects.

- 3.E Write program code to create, traverse, and manipulate elements in 2D array objects.
- 4.A Use test-cases to find errors in program code.
- 4.B Identify errors in program code.
- 4.C Determine if two or more code segments yield equivalent results.
- 5.D Describe the initial conditions that must be met for a program segment to work as intended or described.

Unit 5 Topics

Our Unit 5 covers the College Board Units 6, 7, and 8. Over the course of this unit and the unit project, the following topics will be covered and teachers may use these in the AP Classroom to filter questions for formative and summative assessments:

- 6.1 Array Creation and Access
- 6.2 Traversing Arrays
- 6.3 Enhanced `for` Loop for Arrays
- 6.4 Developing Algorithms using Arrays
- 7.1 Introduction to `ArrayList`
- 7.2 `ArrayList` Methods
- 7.3 Traversing `ArrayLists`
- 7.4 Developing Algorithms using `ArrayLists`
- 7.7 Ethical Issues Around Data Collection
- 8.1 2D Arrays
- 8.2 Traversing 2D Arrays

Unit 5 Schedule (22 Days)

Day	Topic	Lessons	Learning Objectives	Essential Knowledge
U5D1	Unit Project	Air Quality Analyzer Project (1 of 6)	None	None
U5D2-D3	Arrays	Array Creation and Access	IOC-1.B, VAR-2.A	IOC-1.B.1, IOC-1.B.2, VAR-2.A.1, VAR-2.A.2, VAR-2.A.3, VAR-2.A.4, VAR-2.A.5, VAR-2.A.6, VAR-2.A.7
U5D4		Traversing Arrays	VAR-2.B	VAR-2.B.1, VAR-2.B.2, VAR-2.B.3
U5D5		Enhanced For Loop for Arrays	VAR-2.C	VAR-2.C.1, VAR-2.C.2, VAR-2.C.3, VAR-2.C.4
U5D6-D7		Developing Algorithms Using Arrays	CON-2.I	CON-2.I.1, CON-2.I.2
U5D8	Unit Project	Air Quality Analyzer Project (2 of 6)	VAR-2.A, VAR-2.B, VAR-2.C, CON-2.D, CON-2.I	VAR-2.A.1, VAR-2.A.2, VAR-2.A.3, VAR-2.A.4, VAR-2.A.5,

				VAR-2.A.6, VAR-2.A.7, VAR-2.B.1, VAR-2.B.2, VAR-2.B.3, VAR-2.C.1, VAR-2.C.2, VAR-2.C.3, VAR-2.C.4, CON-2.D.2, CON-2.I.1, CON-2.I.2
U5D9	ArrayLists	Introduction to ArrayList	VAR-2.D	VAR-2.D.1, VAR-2.D.2, VAR-2.D.3, VAR-2.D.4, VAR-2.D.5
U5D10-D11		ArrayList Methods	VAR-2.D	VAR-2.D.6, VAR-2.D.7
U5D12		Wrapper Classes: Integer and Double	VAR-1.F	VAR-1.F.1, VAR-1.F.2, VAR-1.F.3, VAR-1.F.4, VAR-1.F.5, VAR-1.F.6, VAR-1.F.7
U5D13		Traversing ArrayLists	VAR-2.E	VAR-2.E.1, VAR-2.E.2, VAR-2.E.3, VAR-2.E.4
U5D14		Developing Algorithms Using ArrayLists	CON-2.J	CON-2.J.1, CON-2.J.2
U5D15	Unit Project	Air Quality Analyzer Project (3 of 6)	VAR-1.F, VAR-2.D, VAR-2.E, CON-2.J	VAR-1.F.1, VAR-1.F.2, VAR-1.F.3, VAR-1.F.4, VAR-1.F.5, VAR-1.F.6, VAR-1.F.7, VAR-2.D.1, VAR-2.D.2, VAR-2.D.3, VAR-2.D.4, VAR-2.D.5, VAR-2.D.6, VAR-2.D.7, VAR-2.E.1, VAR-2.E.2, VAR-2.E.3, VAR-2.E.4, CON-2.J.1, CON-2.J.2
U5D16	2D Arrays	2D Arrays	VAR-2.F, VAR-2.G, CON-2.N	VAR-2.F.1, VAR-2.F.2, VAR-2.F.3, VAR-2.F.4, VAR-2.F.5, VAR-2.G.1, VAR-2.G.2, VAR-2.G.3, CON-2.N.2
U5D17		Traversing 2D Arrays	VAR-2.F, VAR-2.G, CON-2.N	VAR-2.F.1, VAR-2.F.2, VAR-2.F.3, VAR-2.F.4, VAR-2.F.5, VAR-2.G.1, VAR-2.G.2, VAR-2.G.3, CON-2.N.2
U5D18	Unit Test Review	Unit 5 Test Review	None	None
U5D19-D20	Unit Project	Air Quality Analyzer Project (4-5 of 6)	VAR-2.F, VAR-2.G, CON-2.N	VAR-2.F.1, VAR-2.F.2, VAR-2.F.3, VAR-2.F.4, VAR-2.F.5, VAR-2.G.1, VAR-2.G.2, VAR-2.G.3, CON-2.N.2
U5D21	Unit Test	Unit 5 Test	None	None
U5D22	Unit Project	Air Quality Analyzer Presentation (6 of 6)	None	None

Unit 6: Inheritance

Unit 6 takes a comprehensive look at two relationships between classes in Java: inheritance and composition. Class interaction is a fundamental concept of OOP - building a new class from an existing class or using features of one class in another class leads to reliability and reliability is the main goal of OOP. Topics covered in depth are superclasses and subclasses, constructors of the subclasses, overriding methods, composition, the Object class, and polymorphism.

The Big Idea covered in this unit is Modularity. These are the Enduring Understandings covered in this unit:

- MOD 2: Programmers use code to represent a physical object or nonphysical concept, real or imagined, by defining a class based on the attributes and/or behaviors of the object or concept.
- MOD-3: When multiple classes contain common attributes and behaviors, programmers create a new class containing the shared attributes and behaviors forming a hierarchy. Modifications made at the highest level of the hierarchy apply to the subclasses.

These are the Computational Thinking Skills addressed:

- 1.A Determine an appropriate program design to solve a problem or accomplish a task.
- 1.C Determine code that would be used to interact with completed program code.
- 3.A Write program code to create objects of a class and call methods.
- 3.B Write program code to define a new type by creating a class.
- 5.A Describe the behavior of a given segment of program code.
- 5.B Explain why a code segment will not compile or work as intended.
- 5.D Describe the initial conditions that must be met for a program segment to work as intended or described.

Unit 6 Topics

Our Unit 6 covers the College Board Unit 9. Over the course of this unit and the unit project, the following topics will be covered and teachers may use these in the AP Classroom to filter questions for formative and summative assessments:

- 9.1 Creating Superclasses and Subclasses
- 9.2 Writing Constructors for Subclasses
- 9.3 Overriding Methods
- 9.4 `super` Keyword
- 9.5 Creating References using Inheritance Hierarchies
- 9.6 Polymorphism
- 9.7 `Object` Superclass

Unit 6 Schedule (19 Days)

Day	Topic	Lessons	Learning Objectives	Essential Knowledge
U6D1	Unit Project	Hospital Locator Project (1 of 6)	None	None
U6D2-D3	Creating Superclasses and Subclasses	Class Hierarchies	MOD-3.B, MOD-3.E	MOD-3.B.1, MOD-3.B.2, MOD-3.E.3, MOD-3.E.4
U6D4		Extending Classes	MOD-3.B	MOD-3.B.1, MOD-3.B.2, MOD-3.B.4, MOD-3.B.11, MOD-3.B.12, MOD-3.B.13
U6D5		The “is-a” Inheritance Relationship	MOD-3.B	MOD-3.B.3
U6D6		The “has-a” Composition Relationship	MOD-2.B	MOD-2.B.1
U6D7		Overriding Methods	MOD-3.B	MOD-3.B.10, MOD-3.B.11, MOD-3.B.12, MOD-3.B.13, MOD-3.B.14, MOD-3.B.15
U6D8	Unit Project	Hospital Locator Project (2 of 6)	MOD-3.B, MOD-3.E	MOD-3.B.1, MOD-3.B.2, MOD-3.B.3, MOD-3.B.4, MOD-3.B.10, MOD-3.B.11, MOD-3.B.12, MOD-3.B.13, MOD-3.B.14, MOD-3.B.15, MOD-3.E.3, MOD-3.E.4
U6D9-D10	Constructors	Writing Constructors for Subclasses	MOD-3.B	MOD-3.B.5, MOD-3.B.6, MOD-3.B.7, MOD-3.B.8, MOD-3.B.9, MOD-3.B.14, MOD-3.B.15
U6D11	Reference vs Object Type	Declaring Objects of Parent Type	MOD-3.C	MOD-3.C.1, MOD-3.C.2, MOD-3.C.3, MOD-3.C.4
U6D12	Unit Project	Hospital Locator Project (3 of 6)	MOD-3.B, MOD-3.C, MOD-3.E	MOD-3.B.1, MOD-3.B.2, MOD-3.B.3, MOD-3.B.4, MOD-3.B.5, MOD-3.B.6, MOD-3.B.7, MOD-3.B.8, MOD-3.B.9, MOD-3.B.10, MOD-3.B.11, MOD-3.B.12, MOD-3.B.13, MOD-3.B.14, MOD-3.B.15, MOD-3.C.1, MOD-3.C.2, MOD-3.C.3, MOD-3.C.4, MOD-3.E.3, MOD-3.E.4
U6D13-D14	Polymorphism	Polymorphism	MOD-3.D	MOD-3.D.1, MOD-3.D.2, MOD-3.D.3
U6D15	Unit Project	Hospital Locator Project (4 of 6)	MOD-3.B, MOD-3.C, MOD-3.D, MOD-3.E	MOD-3.B.1, MOD-3.B.2, MOD-3.B.3, MOD-3.B.4, MOD-3.B.5, MOD-3.B.6, MOD-3.B.7, MOD-3.B.8, MOD-3.B.9, MOD-3.B.10, MOD-3.B.11, MOD-3.B.12, MOD-3.B.13, MOD-3.B.14, MOD-3.B.15, MOD-3.C.1, MOD-

				3.C.2, MOD-3.C.3, MOD-3.C.4, MOD-3.D.1, MOD-3.D.2, MOD-3.D.3, MOD-3.E.3, MOD-3.E.4
U6D16	Unit Test Review	Unit 6 Test Review	None	None
U6D17	Unit Project	Hospital Locator Project (5 of 6)	MOD-3.B, MOD-3.C, MOD-3.D, MOD-3.E	MOD-3.B.1, MOD-3.B.2, MOD-3.B.3, MOD-3.B.4, MOD-3.B.5, MOD-3.B.6, MOD-3.B.7, MOD-3.B.8, MOD-3.B.9, MOD-3.B.10, MOD-3.B.11, MOD-3.B.12, MOD-3.B.13, MOD-3.B.14, MOD-3.B.15, MOD-3.C.1, MOD-3.C.2, MOD-3.C.3, MOD-3.C.4, MOD-3.D.1, MOD-3.D.2, MOD-3.D.3, MOD-3.E.3, MOD-3.E.4
U6D18	Unit Test	Unit 6 Test	None	None
U6D19	Unit Project	Hospital Locator Presentation (6 of 6)	None	None

Unit 7: Searching, Sorting, and Recursion

Unit 7 covers different search algorithms, the selection sort and insertion sort, Big O, recursion, recursive search and sort algorithms.

The Big Idea of Control is covered in Unit 7. These are the Enduring Understandings covered in this unit:

- CON 2: Programmers incorporate iteration and selection into code as a way of providing instructions for the computer to process each of the many possible input values.

These are the Computational Thinking Skills addressed:

- 1.B Determine code that would be used to complete code segments.
- 2.C Determine the result or output based on the statement execution order in a code segment containing method calls.
- 2.D Determine the number of times a code segment will execute.
- 3.D Write program code to create, traverse and manipulate elements in 1D array or ArrayList objects.
- 5.A Describe the behavior of a given segment of program code.
- 5.C Explain how the result of program code changes, given a change of the initial code.

Unit 7 Topics

Our Unit 7 covers the College Board Unit 10 and the searching and sorting topics from Unit 7. Over the course of this unit and the unit project, the following topics will be covered and teachers may use these in the AP Classroom to filter questions for formative and summative assessments:

- 7.5 Searching
- 7.6 Sorting
- 10.1 Recursion
- 10.2 Recursive Searching and Sorting

Unit 7 Schedule (20 Days)

Day	Topic	Lessons	Learning Objectives	Essential Knowledge
U7D1	Unit Project	Data Decoder Project (1 of 6)	None	None
U7D2	Searching	Sequential and Iterative Binary Search	CON-2.K, CON-2.N	CON-2.K.1, CON-2.K.2, CON-2.N.1
U7D3	Iterative Sorts	Selection Sort	CON-2.L, CON-2.M	CON-2.L.1, CON-2.M.1
U7D4		Insertion Sort	CON-2.L, CON-2.M	CON-2.L.1, CON-2.M.1
U7D5-D6		Create Your Own Sort	CON-2.L, CON-2.M	CON-2.L.1, CON-2.M.1
U7D7-D8	Unit Project	Data Decoder Project (2-3 of 6)	CON-2.K, CON-2.L, CON-2.M	CON-2.K.1, CON-2.K.2, CON-2.L.1, CON-2.M.1
U7D9	Informal Analysis	Compare Big O Informally	CON-2.H, CON-2.M	CON-2.H.1, CON-2.M.1
U7D10	Unit Project	Data Decoder Project (4 of 6)	CON-2.H, CON-2.K, CON-2.L, CON-2.M	CON-2.H.1, CON-2.K.1, CON-2.K.2, CON-2.L.1, CON-2.M.1
U7D11	Recursion	Recursion Defined	CON-2.O	CON-2.O.1, CON-2.O.2, CON-2.O.3, CON-2.O.4, CON-2.O.5, CON-2.O.6
U7D12		Recursive Methods		
U7D13		Recursive Traversal Methods		
U7D14	Recursive Searching and Sorting	Binary Search	CON-2.P	CON-2.P.1, CON-2.P.2, CON-2.P.3, CON-2.P.4
U7D15-D16		Merge Sort	CON-2.Q	CON-2.Q.1
U7DOP	Unit Project	Data Decoder Project (Optional)	CON-2.H, CON-2.M, CON-2.P, CON-2.Q	CON-2.H.1, CON-2.M.1, CON-2.P.1, CON-2.P.2, CON-2.P.3, CON-2.P.4, CON-2.Q.1
U7D17	Unit Test Review	Unit 7 Test Review	None	None
U7D18	Unit Project	Data Decoder Project (5 of 6)	None	None
U7D19	Unit Test	Unit 7 Test	None	None
U7D20	Unit Project	Data Decoder Presentation (6 of 6)	None	None

Pedagogical Approaches

The *UTeach AP Computer Science A* curriculum uses Project-Based Learning (PBL) and Culturally Responsive Teaching (CRT) in order to better engage students in the learning process. By encouraging students to use critical thinking skills and challenging them to solve authentic and meaningful problems, PBL helps students to develop a deeper and more profound understanding of the power of computation in our everyday lives. Built using a student-centric approach, the curriculum leverages many important components of CRT including learning within the context of culture, encouraging the teacher to be a facilitator of learning, and mediating instruction within the culture. Teachers who are unfamiliar with the goals, methods, and techniques of PBL can learn more at the [Buck Institute for Education website](#). Likewise, more information about CRT can be found at the [Education Alliance website](#).

In teaching this course, educators are encouraged to use the range of PBL and CRT techniques that have been incorporated into each unit, including narrative *anchoring videos*, overarching *unit projects*, clear *rubrics*, regular *benchmarks*, *scaffolding activities*, *final products*, and *reflection*. Used together in a coherent, unified manner that actively engages students in the educational process, PBL and CRT strategies can improve comprehension and retention and help students develop stronger problem solving, critical thinking, and group communication skills.

Instructional Sequencing

The year-long course consists of seven instructional units (each approximately lasting four weeks) that have been carefully structured to guide students through an introduction to the Java programming language and the study of object-oriented programming, a style of programming that uses classes and objects mimicking the organization of the real world. The topics are sequenced so that each new unit builds on knowledge and skills from previous units, with a goal to prepare students to successfully complete the College Board AP Computer Science A exam in the spring.

Introduction

The first module of each instructional unit opens with an anchor video designed to introduce the driving questions, unit project, and key topics for the next few weeks of study. Students are expected to participate in small-group and/or whole-class discussion to identify areas of focus that will direct and drive learning throughout the unit.

Topic Lessons/Activities

Distributed throughout each unit, individual lessons, daily activities, programming assignments, and discussions will allow students to explore and practice applying relevant skills and concepts in greater detail.

Project Work Days

Each of the projects has five or six dedicated work periods throughout each unit for students to work on the programming aspect of the project. Project work days are not consecutive but instead are spread throughout the unit, strategically placed after relevant topics have been presented.

Assessments

In addition to informal, formative assessments throughout each unit, student learning and progress is also assessed at the end of each unit through a formal, summative assessment and evaluation of their independent and collaborative efforts on the unit project.

Formal assessments are modeled after the single-select multiple-choice questions and free response questions on the *AP Computer Science A* exam so that students can familiarize themselves with the scope, style, and timing of questions that they can expect to see on the AP exam in the spring.

Unit Projects

The course is narrative-driven, with students following a diverse cast of characters who travel across the globe and solve real-world problems along the way. Students will come to know and love a cast of four main characters, Vee, Hinni, Ed, and Antonio who take them on an adventure around the world. Hinni, Ed, and Antonio are helping Vee search for her mother who is stationed at the McMurdo Station, a research facility in Antarctica, when disaster strikes. As the group of friends fly to each continent on their search, they take time out to help people along the way solving problems using their computational thinking skills. The students in the classroom are actively involved with the characters as they represent a technical team of computer programmers who lend a helping hand in creating effective solutions to the authentic problems the group faces.

The opening module of each unit serves as a formal launch of the unit project, an overarching, product-oriented challenge for students to investigate, research, and solve over the course of the unit. The project launch starts with an anchor video that introduces the fundamental problem or challenge to be solved and is intended to spark the students' imaginations and inspire them to find a solution. Projects are designed to be collaborative with students working together in groups and using the pair programming instructional strategy. These project work days, along with smaller daily programming assignments, exceed the College Board Curricular Requirement 9, requiring students to spend at 20 hours in hands-on lab experiences [CR-9].

Unit Projects [CR-9]
<p>Unit 1: Avatar Creator Project Students design unique, personal avatars, using the Python with Turtle programming language, in order to represent themselves as a member of the technical team assisting the cast of characters in the course narrative.</p>
<p>Unit 2: Resource Finder Project Students create a program using conditional statements to read and analyze data from different rock samples in order to identify an ideal area to rebuild a science research station in Antarctica.</p>
<p>Unit 3: Language Interpreter Project Students create a language interpreter program that collects a speech sample of an unknown indigenous language in South America, then uses iteration and String class methods to translate the new language into English.</p>
<p>Unit 4: Disease Diagnoser Project Students will design a program using classes and methods in order to diagnose different diseases based on patient symptoms in a clinic in South Africa.</p>
<p>Unit 5: Air Quality Analyzer Project Students write a program to store and analyze air quality data from cities in Sri Lanka using arrays, ArrayLists, and 2D arrays. After completing their analysis, students compare the use of the different data structures and decide if their program is scalable for a larger data set.</p>
<p>Unit 6: Hospital Locator Project Students reconstruct a compromised database of hospitals in Australia using superclass and subclass structures. Students will then use a clue to search for a specific hospital.</p>
<p>Unit 7: Data Decoder Project Students write a program that uses searching and sorting algorithms to analyze climate change data. They must optimize the algorithms for greater efficiency in order to create a presentation for the World Meteorological Organization in Switzerland.</p>

Rubrics

Each unit project is accompanied by a clearly defined rubric that specifies the set of expectations for student work throughout the unit, including an exhaustive list of assessment criteria for the artifacts that students will produce and detailed descriptors for each performance level that a student might demonstrate. Teachers should provide students with the rubric at the start of the unit as part of the initial discussion immediately following the anchor video. Giving the students the rubric at the time of the project launch is critical for setting clear student expectations early in the research and learning process. Over the course of

the unit, teachers should regularly refer back to the goals and criteria of the rubric in order to ensure that students remain focused and on pace for meeting the stated requirements.

Benchmarks

Each unit provides the teacher with a number of benchmark activities, or subtasks, that feed into the larger unit project. Each of these subtasks contributes directly to the final product that the students will create. Teachers can use these benchmarks as intermediate, informal assessments to gauge the progress of each student and/or collaborative group in their mastery of the unit goals.

Scaffolding Activities

The bulk of each unit consists of a series of individual topic lessons, activities, discussions, and hands-on applications that allow the teacher to provide instruction, guidance, and support to students and collaborative groups as they design and implement the programming code for the unit project. These scaffolding activities serve to introduce, explain, and encourage the use of the unit's core concepts and skills by providing students with structured opportunities and incentives to explore the material in greater depth.

Final Products and Student Portfolio

At the culmination of each unit, student groups are expected to present a final product that represents the body of their work and code implementation on the unit project. Students demonstrate their mastery of the core content and skills for the unit by exhibiting the authentic and purposeful artifacts, in the form of working program code, that they have created. A key component of the project always includes a public presentation of each student's work before their peers as a way of providing motivation for each student and holding them accountable for their own learning.

Reflections

Students will be required to submit their project to the teacher in the form of a Student Submission Template which provides the teacher with a link to the student's final program code along with thoughtful responses to specific questions about the topics covered in the unit project. This provides students an opportunity to reflect on the challenges and successes during the course of the unit project.

Resources and Technical Requirements

UTeach AP Computer Science A does not require additional materials to be purchased for implementation.

Websites to Whitelist

The following websites will be accessed throughout the duration of the course and should be submitted to the school IT department for whitelisting prior to beginning the course. We also recommend testing access to different sites from your device as well as students' devices to ensure a smooth lesson.

repl.it, youtube.com, vimeo.com, apcentral.collegeboard.org, docs.oracle.com, docs.python.org, codingbat.com, topsoccerblog.com, epa.gov, scijinks.gov, aqicn.org, wunderground.com, wikipedia.org, google.com/maps, cs.joensuu.fi/jeliot

Course Bibliography

Almossawi, A. (2014). *An illustrated book of bad arguments*. New York, NY: The Experiment.

Parlante, Nick. *CodingBat Java*, codingbat.com/java.

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

Williams, L. A., & Kessler, R. R. (2000). All I really need to know about pair programming I learned in kindergarten. *Communications of the ACM*, 43(5), 108–114.

doi:10.1145/332833.332848.