edhesive

UTeach CS Principles

Course Syllabus:
Advanced Placement® Computer Science Principles

AP® Computer Science Principles introduces students to the foundational concepts of computer science and explores the impact computing and technology have on our society.

Edhesive has partnered with the University of Texas at Austin's UTeach Institute to launch an online version of the esteemed UTeach CS Principles curriculum.

Course Prerequisites

It is recommended that a student will have successfully completed Algebra I prior to taking AP Computer Science Principles. Completion or concurrent enrollment in Algebra II is recommended. No previous programming experience is required.

Learning Environment

Edhesive's online version of the UTeach CS Principles course was designed to be used in a blended classroom. The course uses a mix of web-based and in-person instruction. Students will work online where they will watch videos, complete interactive activities, quizzes and exams, and participate in an online, moderated discussion forum with their peers located around the country. Students will also work in-person with their classmates on unit projects, activities and class discussions. Teachers use the course lesson guides and data and analytics reports to manage their classrooms, facilitate collaborative learning amongst students, and give focused attention to individual students.

Course Assessments

This courses uses three types of assessments:

1) There are short formative assessments to check for understanding after each sub-lesson. These are low stakes quizzes that are typically either multiple choice or fill-in-the-blank, and are automatically graded by the online courseware.
2) A summative exam at the end of each unit assesses students' knowledge of concepts covered in the unit. These exams are multiple choice only, written in the same style and format as the multiple choice questions students will see on the AP exam. The summative assessments are also automatically graded by the online courseware.
3) Each unit has a unit project that presents an overarching challenge for students to investigate, research, and solve. Over the course of the unit, students will collaborate in groups to complete their projects. Detailed grading rubrics are given to students to set expectations and guide their research and learning, and also to teachers to assess the final project grade.

The following pages detail the UTeach CS Principles curriculum in detail. Edhesive's AP Computer Science Principles course strictly follows UTeach's curriculum, while including additional video, interactives, and assessments to support student instruction.

UTeach
CS Principles

Course Syllabus and Planning Guide

(2017 - 2018)

College Board Syllabus ID #1648112v1

Course Description

Developers (http://uteachcs.org)

UTeach CS Principles has been developed by The UTeach Institute through a grant from the National Science Foundation (award #1543014).

Course Overview

UTeach CS Principles has been designed as a year-long high school course that fully addresses the seven "Big Ideas" of computer science and six "Computational Thinking Practices", as specified by the College Board's *AP Computer Science Principles* 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 own programming and computational thinking skills.

Big Ideas [CR2a-g]	
Topics	**Perspectives**
Abstraction [Big Idea 2] Data and Information [Big Idea 3] Programming [Big Idea 5] The Internet [Big Idea 6]	Creativity [Big Idea 1] Algorithms [Big Idea 4] Global Impact [Big Idea 7]

Computational Thinking Practices [CR1a-f]					
P1	**P2**	**P3**	**P4**	**P5**	**P6**
Connecting Computing	Creating Computational Artifacts	Abstracting	Analyzing Problems and Artifacts	Communicating	Collaborating

Course Bibliography

Abelson, H., Ledeen, K., and Lewis, H. R. *Blown to Bits: your life, liberty, and happiness after the digital explosion.* Upper Saddle River, N.J.: Addison-Wesley, 2008.

Programming Resources

Throughout the course, students will explore the coding process through the context of two different programming environments – Scratch and Processing. Each of these platforms has been designed to provide beginning students with a simplified and novice-friendly interface with which to write their first dynamic and highly engaging programs. The tools for both environments are platform-independent and freely available online, 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.

Scratch (https://scratch.mit.edu)

Developed by the MIT Media Lab, Scratch offers students an introduction to coding through the use of a visual programming interface. By dragging and dropping labeled programming components (a.k.a., "blocks") that snap together into syntactically correct compositions, students can quickly construct robust and fully functional programs with very little prior programming knowledge or skill. This block-based programming environment is ideally suited to first-time programmers as it abstracts away much of the low-level implementation details and allows students to clearly focus on the more generalized concepts that are so fundamental to the art of computational thinking.

Processing (https://processing.org)

Built atop the Java programming language, Processing offers a simplified syntax and graphical programming model that allows novice programmers to easily develop visually dynamic programs using a high-level programming language. For students who have already been exposed to the drag-and-drop programming of a blocks-based language like Scratch, the introduction of Processing helps them make the transition to the text-based experience of a procedural programming language. And by leveraging the syntax and structure of an industrial language like Java, Processing makes it easier for students to later more easily adopt Java (as is used in *AP Computer Science A*), C++, or Python a number of other, similar languages that they might encounter in industry or continued studies in computer science.

Project-Based Learning

The *UTeach CS Principles* curriculum utilizes Project-Based Learning (PBL) 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. This project-based approach is particularly effective in engaging girls and other historically underrepresented groups as well as broadening participation in computing overall. Teachers who are unfamiliar with the goals, methods, and techniques of PBL can learn more at the Buck Institute for Education website (http://bie.org/).

In teaching this course, educators are encouraged to utilize the range of PBL techniques that have been incorporated into each unit, including *driving questions*, 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 strategies can help students improve their retention of learned experiences and develop stronger problem solving, critical thinking, and group communication skills.

Driving Questions

Every unit is ultimately guided by one or more driving questions that serve to specify the unifying goal for student inquiry and learning. These questions, which ground each unit in an authentic, real-world context, will be introduced at the start of each unit and then later revisited and reiterated throughout the ensuing instructional modules. Through this regular repetition, teachers can ensure that students always have a clear sense of what they are trying to solve, what they still need to know, and where they stand in terms of achieving their goals for the unit.

Unit Projects

The opening module of each unit also 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 want to find a solution. Teachers should then use subsequent classroom discussions, lessons, and activities to help guide students in identifying ways they might approach the project and what they will need to study and learn in order to develop a complete solution to the stated challenge.

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 conduct research 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. Specifically, *topic lessons* focus on extending existing knowledge of unit concepts through direct instruction and inquiry-based investigations. Meanwhile the *Unit Project*, *Coding Skills*, and *Big Picture* exercises create opportunities for student-centered, collaborative discussions and exercises that encourage students to explore unit topics from the broader perspectives of creativity, algorithms, and global impact.

Final Products and Student Portfolio

At the culmination of each unit, students are expected to present a final product that represents the body of their work and research on the unit project. Through a combination of individual products and collaborative group products, students demonstrate their mastery of

the core content and skills for the unit by exhibiting the authentic and purposeful artifacts that they have created. While the exact format of the final product may vary from unit to unit, a key component 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.

In addition, teachers should encourage students to add artifacts of their final product to an ongoing student portfolio that they maintain throughout the course. The contents of this portfolio can later be used during *Unit 7: Performance Tasks* as a point of inspiration for the "Create" and "Explore" tasks that the students will need to produce and submit to the College Board as part of the AP Computer Science Principles assessment.

Reflections

Finally, at the end of each unit, teachers should allow their students an opportunity to look back on what they have done, experienced, and learned over the course of the unit and reflect on how their perception of computing may have changed as a result of these experiences. Similarly, students are expected to discuss how the unit material relates to their own personal interests and to articulate how they could apply their newly learned skills to authentic tasks within their own lives.

Websites to Whitelist

There are a few websites that will be accessed throughout the duration of the course. You should advise your IT department about these websites prior to beginning the course. Always test site access from your device and the students' devices to ensure a smooth lesson:

> *gitbooks.io, scratch.mit.edu, processing.org, openprocessing.org, studio.sketchpad.cc, youtube.com, collegeboard.org, wired.com, collegexpress.com, tiny.cc, wikipedia.org, ars.userfriendly.org, simonsingh.net, csunplugged.org, uscyberpatriot.org, puzzles.com, 20q.net, en.akinator.com, utexas.edu, base64.wutils.com, mathisfun.com, laweekly.com, slate.com, vt.edu, informationisbeautiful.net, tonakai.aki.gs, wfu.edu, jumk.de, vimeo.com, creativecommons.org, bitsbook.com, ted.com, data.gov, ncdc.noaa.gov, knoema.com, geocommons.com, statsilk.com, betterworldflux.com, gapminder.org, wordle.net, textcompactor.com, mapstory.org, heatmaptool.com, mapalist.com, mturk.com, kickstarter.com, fold.it, setiathome.berkeley.edu, starwarsuncut.com, npr.org, maximumedge.com, archive.org, dontbubble.us, duckduckgo.com, niemanlab.org, purplemath.com, whatshouldireadnext.com, forbes.com, communicationnation.blogspot.com, spyrestudios.com, tagxedo.com, apple.com, photopin.com, freeimages.com, openclipart.org, clker.com, piktochart.com, infogr.am, designify.me, duolingo.com, nytimes.com, journalism.org, tools.ietf.org, amazon.com, ieee.org, howtofindmyipaddress.com, ipv6-test.com, info.cern.ch, oracleofbacon.org, berkeley.edu, browserling.com*

Instructional Units

Each four-week unit focuses on one or more of the four Big Idea "Topics" (e.g., *Abstraction*, *Data and Information*, *Programming*, and *The Internet*) while simultaneously presenting these topics in the context of the three Big Idea "Perspectives" (e.g., *Creativity*, *Algorithms*, and *Global Impact*) through the use of structured in-class activities, project extensions, and/or classroom discussions.

Introduction

The first module of each instructional unit leads off with an anchor video and/or engagement activity designed to introduce the driving questions, major project(s), 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, exercises, quizzes, and daily activities will allow students to explore and practice applying relevant skills and concepts in greater detail.

Unit "Perspective" Modules		
Unit Project *Creativity* [Big Idea 1]	**Coding Skills** *Algorithms* [Big Idea 4]	**The Big Picture** *Global Impact* [Big Idea 7]
Exercises that encourage students to explore and create artifacts and build new knowledge from unit-specific projects.	Exercises that encourage students to deconstruct and describe unit-specific processes procedurally.	Classroom discussions and investigations that examine the cultural and societal impact of emerging technologies.

Unit Project

Within each unit, select modules are set apart to encourage students to work independently and/or collaboratively in more open-ended, student-directed, hands-on projects and activities. While the current unit's driving question dictates the content of the Unit Project modules, they primarily address the curricular standards for the "Creativity" (Big Idea 1) component of the course and allow students to explore the process of creating computational artifacts.

Coding Skills

While students are expected to actively employ computational thinking techniques and practices throughout all of their work, two of the modules in each unit will specifically address the development and reinforcement of these skills. In particular, students will be encouraged to practice thinking about the logic and sequencing of a solution to a problem and to then express that solution with clarity and precision using code, pseudocode, and/or natural language, as appropriate to the situation. In doing so, these modules address the curricular standards for the "Algorithms" (Big Idea 4) component of the course and reinforce the importance of thorough analysis, detailed preparation, and clear communication when solving a computational task.

The Big Picture

Midway through each unit, students are asked to step back and consider the broader implications of the unit's main topic and its impact on society at large. Through in-class discussions, debates, and creative activities, students will extrapolate from the ideas and concepts presented in class to explore the implications of the use of and advances in computational technology. These *Big Picture* modules specifically address the curricular standards for the "Global Impact" (Big Idea 7) component of the course and encourage students to always consider the consequences of their digital interactions and creations on the world around them.

Assessments

In addition to minor, informal assessments throughout each unit, student learning and progress will also be monitored at the end of each unit through formal assessments and an evaluation of their independent and collaborative efforts.

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

Likewise, as preparation for the *Performance Tasks* that the students will submit to the *College Board* in May, each student will be required to maintain and document a portfolio of their independent and collaborative work throughout each unit. During Unit 7 (*Performance Tasks*), students are encouraged to draw upon this body of work to produce their final submissions for the College Board.

Course Sequencing

The year-long course consists of seven units that have been carefully structured to gently guide novice students through the study of computational technology by first establishing a *context* for the course material, then teaching the *core* skills for creating and using computational tools, followed by demonstrating real-world *applications* of digital technology, and finally allowing the students to *exhibit* the skills they have developed.

Course Units [CR2a-g]		
Core		
Unit 1: Computational Thinking Introduction to computational thinking, logical reasoning, and describing processes through algorithms and pseudocode.	**Big Ideas:** Abstraction [2] Algorithms [4] Programming [5] The Internet [6] Global Impact [7]	**Computational Thinking Practices (CTP):** P1, P2, P3, P4, P5 **Enduring Understandings (EU):** 2.2, 4.1, 4.2, 5.2, 6.3, 7.2
Unit 2: Programming Use *Scratch* to explore sequencing, selection, and iteration as part of the goal to create programs that serve useful functions.	**Big Ideas:** Creativity [1] Algorithms [4] Programming [5] Global Impact [7]	**Computational Thinking Practices (CTP):** P2, P3, P4, P5, P6 **Enduring Understandings (EU):** 1.1, 1.2, 4.1, 5.1, 5.2, 7.3
Unit 3: Data Representation Explore the different means of representing information digitally.	**Big Ideas:** Abstraction [2] Data and Information [3] Algorithms [4] Programming [5]	**Computational Thinking Practices (CTP):** P1, P2, P3, P4, P5, P6 **Enduring Understandings (EU):** 2.1, 2.2, 2.3, 3.3, 4.1, 5.1, 5.3, 5.5
Application		
Unit 4: Digital Media Processing Use *Processing* to programmatically manipulate digital images and audio.	**Big Ideas:** Creativity [1] Abstraction [2] Data and Information [3] Algorithms [4] Programming [5] Global Impact [7]	**Computational Thinking Practices (CTP):** P2, P3, P4, P5, P6 **Enduring Understandings (EU):** 1.2, 1.3, 2.2, 3.3, 4.1, 5.1, 5.3, 5.4, 7.3

Unit 5: Big Data Discover new knowledge through the use of large data sets.	**Big Ideas:** Creativity [1] Abstraction [2] Data and Information [3] Algorithms [4] Programming [5] Global Impact [7]	**Computational Thinking Practices (CTP):** P1, P2, P3, P4, P5, P6 **Enduring Understandings (EU):** 1.2, 2.3, 3.1, 3.2, 3.3, 4.2, 5.1, 7.1, 7.2, 7.3, 7.5
Unit 6: Innovative Technologies Explore the current state of technology and its role in our everyday lives.	**Big Ideas:** Creativity [1] Programming [5] The Internet [6] Global Impact [7]	**Computational Thinking Practices (CTP):** P1, P2, P3, P4, P5, P6 **Enduring Understandings (EU):** 1.1, 1.2, 5.1, 6.1, 6.2, 7.1, 7.4
Exhibition		
Performance Tasks Students demonstrate their learning by creating a portfolio of their work for submission to the College Board.	**Big Ideas:** Creativity [1] Abstraction [2] Data and Information [3] Algorithms [4] Programming [5] Global Impact [7]	**Computational Thinking Practices (CTP):** P1, P2, P3, P4, P6 **Enduring Understandings (EU):** 1.2, 2.2, 3.3, 4.1, 5.1, 5.2, 5.3, 5.4, 5.5, 7.1, 7.2, 7.3, 7.4

Sequencing and Pacing of Units		
Core (3 units / 12 weeks)	**Application** (3 units / 12 weeks)	**Exhibition** (1 unit / 5 weeks)
Introduction to traditional computer science and programming	Examples of applied use-cases for course content throughout society and industries	Student-directed projects for their *Performance Tasks*

Core

The *Computational Thinking*, *Programming*, and *Data Representation* units introduce students to the computational thinking skills that will enable them to fully exploit the power of digital technology and help them to develop a strong foundation in core programming and problem-solving skills. In addition, students will develop a profound appreciation for the key role that information plays in computing and the many ways information can be codified, expressed, stored, and manipulated.

Application

Once students are armed with the necessary skills to create computational programs and artifacts, the *Digital Media Processing*, *Big Data*, and *Innovative Technologies* units allow students to further explore a variety of ways digital computing can and has been applied to revolutionize industries and enable new forms of expression, communication, and discovery.

Exhibition

Finally, serving as a capstone to the course, the *Performance Task* unit encourages students to demonstrate what they've learned by designing, developing, and further refining a number of student-directed projects, both individually and collaboratively.

Composition of Instructional Units		
Topic Lessons/Activities	**Projects/Discussions**	**Assessments**
Each unit addresses one or more related, Big Idea "Topics".	Each unit includes three modules whose activities frame the unit content within the contexts of the three Big Idea "Perspectives".	Formal Assessments (modeled after AP Multiple-Choice format) and Projects/Performance Assessments (modeled after the AP "Create" and "Explore" Performance Task rubrics).