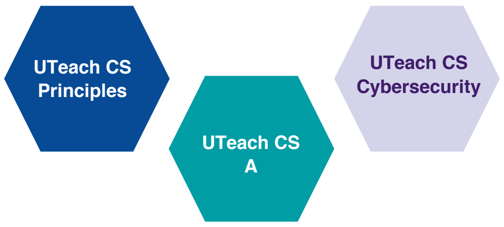
AP Computer Science Principles

Computer Science Principles is an inclusive course designed for all students to explore programming, computational thinking, and the impacts of computing on our lives.

The CS Principles curriculum was developed with the explicit intention of broadening participation among young women and others historically underrepresented in the field of computing.

Computer Science Pathways

- **AP Computer Science Principles** is often offered as a foundational course, followed by **AP Computer Science A**. The courses can also be offered in any order, since the course content is complementary rather than overlapping.
- Whether or not these courses are offered for AP credit, students in UTeach CS courses will discover how to apply programming and computational thinking skills to personally relevant projects.



Who should take CS Principles?

- 9th - 12th grade students
- Students new to programming
- Students with previous experience in computer programming
- Recommended Algebra I prerequisite (required by some schools)
- No prior coding experience needed

This course is designed to engage all students, while also providing rigorous preparation for the AP Computer Science Principles exam.

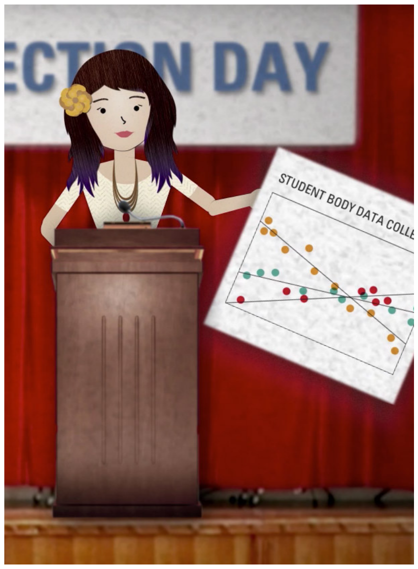
UTeach Computer Science

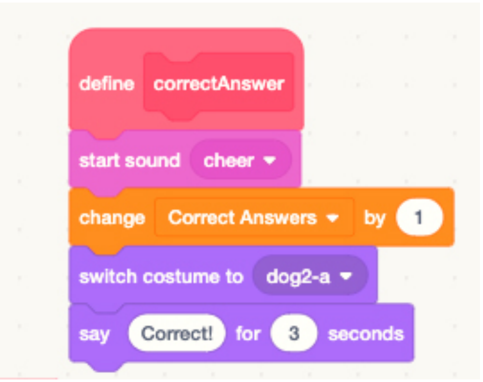
View more student recruitment resources:
cs.uteach.utexas.edu/student-recruitment

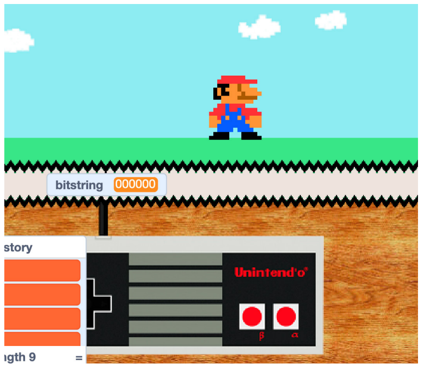
UTeach CS Principles Course Overview

In this engaging, project-based course, students develop computational thinking and programming skills through collaborative, open-ended projects that are personally meaningful to students' lives.

Students examine how computing shapes our society by investigating and debating issues such as cybersecurity, data privacy, and digital literacy. Assignments grow in complexity as students explore programming concepts first through block-based, drag-and-drop coding in Scratch, followed by text-based coding in Python.




```
define correctAnswer
start sound cheer
change Correct Answers by 1
switch costume to dog2-a
say Correct! for 3 seconds
```



Unit 1: Algorithmic Thinking

Students apply the principles of algorithmic thinking (including sequencing, selection, and repetition) to generate secure, unique passwords that are easy to remember but difficult to guess. The unit culminates with a challenge for students to try to "hack" their classmates' password algorithms.

Unit 2: Programming

Students dive into block-based programming to create a personally meaningful Scratch program, game, or movie. Through an iterative development process, students construct blocks of executable code, explore debugging techniques, and ensure that their program functions properly.

Unit 3: Data Representation

Students delve deeper into computational abstraction while programming the interface to a classic video game controller using Scratch. Students discover how all digital information is ultimately represented by "bits" (binary 0s and 1s) and map game actions to button presses.

Unit 4: Digital Media Processing

Students create Instagram-like photo filters after exploring how the RGB color model is used to encode digital images and represent images on computer displays. Students gain familiarity with text-based programming environments (Python) that will prepare them for continuing studies in computer science beyond the scope of this course.

Unit 5: Big Data

Students discover how data analytics can uncover hidden patterns and drive decision-making in fields as disparate as business, science, and government. Students examine open-source databases and communicate insights through short, "TED talk"-style presentations.

Unit 6: Innovative Technologies

Students become visionaries of "future technologies" as they prototype new products that could be realized within their lifetimes. This unit encourages students to think about the entrepreneurial process behind the creation of new technologies, as well as their own role in shaping the future.

Learn more about UTeach Computer Science Principles: **cs.uteach.utexas.edu/csp**

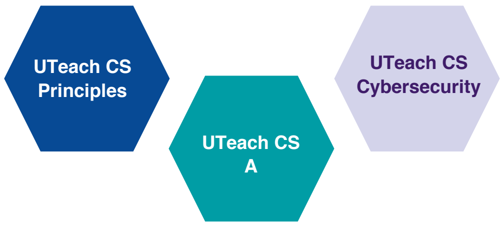
AP Computer Science A

Computer Science A is a rigorous, college-level course designed to prepare students for advanced coursework and potential career pathways in computing.

The CS A curriculum was developed with the explicit intention of broadening participation among young women and others historically underrepresented in the field of computing.

Computer Science Pathways

- **AP Computer Science A** is an advanced course, preceded by more foundational courses like **AP Computer Science Principles**. However, these courses can be taken in any order, since the content is complementary rather than sequential.
- Students in UTeach CS courses will discover how to apply programming skills to solve real-world and socially relevant problems.



Who should take CS A?

- 10th - 12th grade students
- Students **with some previous experience** in computer programming (recommended)
- Students interested in pursuing future computing or STEM careers
- Algebra I prerequisite required

This course is designed to engage **all students**, while also providing rigorous preparation for the AP Computer Science A exam.

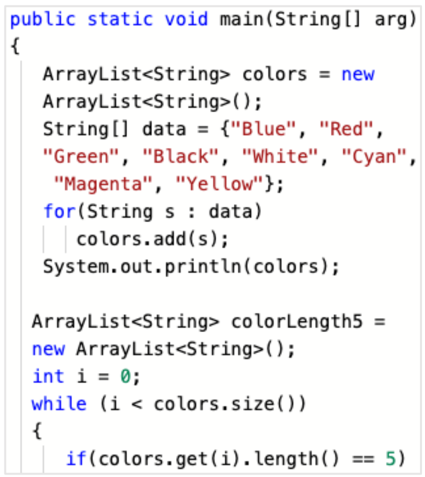
UTeach Computer Science

View more student recruitment resources:
cs.uteach.utexas.edu/student-recruitment

UTeach CS A Course Overview

In this rigorous, project-based course, students deep-dive into programming concepts while collaborating to solve a series of socially relevant challenges. Each unit introduces a unique, real-world problem following an overarching course narrative as students “travel the globe” in search of a missing scientist.

Students are immersed in Java programming topics (e.g. abstraction, algorithms, data structures, object-oriented programming), preparing for advanced college coursework and potential career pathways in computing.



```
public static void main(String[] arg)
{
   ArrayList<String> colors = new
   ArrayList<String>();
   String[] data = {"Blue", "Red",
   "Green", "Black", "White", "Cyan",
    "Magenta", "Yellow"};
   for(String s : data)
      colors.add(s);
   System.out.println(colors);

  ArrayList<String> colorLength5 =
  new ArrayList<String>();
  int i = 0;
  while (i < colors.size())
  {
     if(colors.get(i).length() == 5)
```

Unit 1: Introductions Are In Order

Students use programming algorithms to design unique personal avatars connecting to the the course narrative. Students explore foundational coding concepts such as algorithms, abstraction, modularity, and control.

Unit 2: Primitive Control

Students learn and apply basic Java concepts (e.g., primitive data types, mathematical expressions and operators, conditionals, objects, methods) to create an algorithm identifying the ideal location for a new Antarctic research station.

Unit 3: Strings and Iteration

Students create a language interpreter program to collect a speech sample of an unknown indigenous language in South America, then use iteration and String class methods to translate the new language into English.

Unit 4: Objects, Classes, and Methods

Students design a program using classes and methods in order to diagnose different diseases based on patient symptoms in a clinic in Africa. By the end of the unit, students create a complete class and instantiate objects of that class to determine if symptoms match a specific disease.

Unit 5: Arrays, ArrayLists, and 2D Arrays

Students develop data structures and algorithms to analyze real-world pollution data from cities in Asia. After applying algorithms to process and analyze the data, students compare data structures and determine the scalability of their programs.

Unit 6: Inheritance

Students rebuild a database of hospitals in Australia that has been compromised by cyberhackers. Students create a new class from an existing class, design their own superclasses and related subclasses, and search the database for a specific entry.

Unit 7: Searching, Sorting, and Recursion

Students apply searching and sorting algorithms to compile data for a presentation on climate change to the World Meteorological Organization in Europe. Students predict the efficiency of their sorting algorithms, test their predictions against a larger database, and optimize their algorithms for greater efficiency.

Learn more about UTeach Computer Science A: **cs.uteach.utexas.edu/csa**