Welcome to Computer Science Discoveries

Code.org's CS Discoveries is an introductory, classroom-based course appropriate for 6-10th grade students. The course aims to empower students to create authentic artifacts and engage with computer science as a medium for creativity, communication, problem solving, and fun. CS Discoveries (CSD) takes a wide lens on computer science by covering topics such as programming, physical computing, web development, design, and data. The course inspires students as they build their own websites, apps, games, and physical computing devices. Our curriculum is available at no cost for anyone, anywhere in the world and can be accessed at code.org/csd.

Curriculum At-a-Glance

CSD is designed with the new-to-CS student and teacher in mind and can be taught as a year-long course, a semester-long course (3-5 hours per week of instruction for 9+ weeks), or in a more modular approach, teaching the units that fit the teacher's needs. See our Implementation Guide for more details on ways to implement CSD. When teaching these units as a full-year course, the following sequence will satisfy all of the middle school CSTA Standards.

Problem Solving and Computing	Students learn about the problem-solving process, the input-output-store-process model of a computer, and how computers help humans solve problems. Students end the unit by proposing their own app to solve a problem.
Web Development	Students learn to create websites using HTML and CSS inside Code.org's Web Lab environment. Throughout the unit, students consider questions of privacy and ownership on the internet as they develop their own personal websites.
Interactive Animations and Games	Students learn fundamental programming constructs and practices in the JavaScript programming language while developing animations and games in Code.org's Game Lab environment. Students end the unit by designing their own animations and games.
The Design Process	Students apply the problem solving process to the problems of others, learning to empathize with the needs of a user and design solutions to address those needs. During the second half of the unit, students form teams to prototype an app of their own design, first on paper and eventually in Code.org's App Lab environment.
Data and Society	Students explore different systems used to represent information in a computer and the challenges and trade-offs posed by using them. In the second half of the unit, students learn how collections of data are used to solve problems and how computers help to automate the steps of this process.
Creating Apps with Devices	Students use Code.org's App Lab environment, along with either the Adafruit Circuit Playground or the BBC micro:bit, to explore the relationship between hardware and software. Students develop prototypes that mirror existing innovative computing platforms, before ultimately designing and prototyping one of their own

Optional Unit

AI and Machine Learning	Students learn how machine learning can be used to solve problems by preparing data, training a machine learning model, then testing and evaluating the model for accuracy and bias. Students use Code.org's AI Lab environment to train machine learning models, then import their models into App Lab to create apps that solve problems

Our Vision

Code.org's vision is that every student in every school should have the opportunity to learn computer science (code.org/about). Our curriculum is designed so that an empowered teacher can lead a diverse group of students through experiences that are supportive, equitable, engaging, and lead to valuable learning (code.org/educate/curriculum/values).

Historically this vision has contrasted sharply with reality. Until recently, most schools did not offer computer science at all, and what classes there were notoriously lacked in diversity. Additionally, many students found these classes unengaging, intimidating, or simply disconnected from their lived experiences with technology. Thanks to efforts by many organizations and individuals, this world is beginning to change: many more schools now offer computer science courses; more diverse students take those courses; and more engaging, relevant, and equitable pedagogy has become the established norm. Even so, there is much work still to be done. This course is designed to continue this momentum as the collective CS education community moves towards this vision of an equitable CS education system.

How We Support Our Vision

Many aspects of Code.org's CS Discoveries curriculum are designed to bring about the eventual change we aim to see more broadly in CS education. Some of the most significant features are listed below.

Free and open: We make our curriculum, videos, and tools free and open for anyone to adopt.

Prioritize New-to-CS Teachers: Historically only a few schools could hire trained computer scientists as teachers, which severely limited which schools could offer a CS course. Reaching all schools has meant developing our CS Discoveries course with the understanding that most of our teachers are new-to-CS and prioritizing their needs. As such, our course includes some distinctive features.

- Comprehensive lesson plans and resources designed to ensure new-to-CS teachers have everything they need to implement the course
- Clear and consistent pedagogy to help new-to-CS teachers develop best practices as CS teachers
- High-quality videos that help teachers introduce and explain CS concepts
- An associated professional learning program that pays particular attention to the needs of new-to-CS teachers

Equitable Pedagogy: Our curriculum is designed to promote an equitable classroom environment for all students, with particular attention paid to the experiences of historically excluded groups, most notably young women and Black, Hispanic, and Native American students. Drawing from extensive feedback from our classrooms, as well as CS education research, our course includes many features designed to support and prioritize these students:

- Pedagogy that develops a collaborative and supportive classroom environment
- Specific attention paid to language demands of our lessons
- Projects and activities that highlight a variety of applications of computing and frequently ask students to incorporate their own backgrounds and interests.
- Curriculum videos that feature a cast of diverse role models in terms of race, gender, and profession who empower our diverse students to "see themselves" as part of the world of computing
- A professional learning program that highlights these features and helps teachers reflect on how best to implement them within their own classroom

Join Us in this Vision

We think our vision is audacious and deeply motivating. If you feel the same, the best way to join us in this vision is to teach this course! We know that for many teachers this represents a significant undertaking, and we have aimed to do our best to help share the load. Based on the feedback of many teachers we know it will be a challenging, but ultimately gratifying experience. Code.org is here to support you, and we look forward to your feedback so that we can continue to make CS Discoveries an even better experience for our students and teachers.

Provided Materials

Our materials and tools are specifically created with a focus on foundational concepts and are designed to support exploration and discovery. This allows students to develop an understanding of these concepts through “play” and experimentation. From our coding tools to our non-coding tools and videos, all our resources have been engineered to support the lessons in our curriculum, and thus our philosophy about student engagement and learning. Together, these resources typically allow the teacher to act in the role of facilitator and coach when addressing unfamiliar material. In instances when the teacher acts as the primary source of information, generous supports are provided.

All resources below can be accessed free of charge at code.org/csd.

Lesson Plans

- Instructional guides for every lesson
- Activity Guides and handouts for students
- Lesson presentation slides
- Formative and summative assessments
- Exemplars, rubrics, and teacher dashboard

Videos

- Tutorials, instructional videos, and inspirational videos

Tools

- Web Lab - A browser-based tool for creating and publishing HTML and CSS web sites.
- Game Lab - A browser-based JavaScript programming environment designed to create sprite-based drawing, animations, and games. Enables students to switch between programming in blocks or text.
- App Lab - A browser-based JavaScript programming environment for creating interactive apps. Enables students to switch between programming in blocks or text.
- AI Lab - A browser-based tool for creating machine learning models from tabular data.

Technical Requirements

The course requires and assumes a 1:1 computer lab or setup such that each student in the class has access to an Internet-connected computer every day in class. The course is developed to be completed within the classroom - no homework or after-hours computer access is assumed. All provided course tools and resources listed above are available online. Tablets are not currently supported. For more details on the technical requirements, please visit: code.org/educate/it

Suggested Materials and Supplies

One potential cost to consider when teaching this course is printing. Many lessons have handouts that are designed to guide students through activities. While it is not required that all of these handouts be printed, many were designed to be printed and we highly recommend printing when possible.

Beyond printing, some lessons call for typical classroom supplies and manipulatives such as: Student journals, poster paper, markers/colored pencils, scissors, scrap paper, glue or tape, post-it notes or index cards (or similar sized scrap paper), and rulers or a straight edge of some kind.

In addition to those general course materials, the following items are called for in specific units:

- Problem Solving and Computing
    - Aluminum foil, container for water, pennies (note that pennies can be replaced with some other kind of weight of the same size). Alternate activities are available if you do not have access to these supplies.
- Creating Apps with Devices (Option A and Option B)
    - Option A: Classroom set of Circuit Playgrounds. Check out code.org/maker/circuitplayground for more details.
    - Option B: Classroom set of micro:bit devices. Check out code.org/maker/microbit for more details.
    - Maker supplies, such as empty tissue boxes or cardboard rolls or scratch paper

  Note: The Creating Apps with Devices unit has **two implementation options - one for those using circuit playgrounds and one for those using micro:bit. Teachers should **not** buy both sets of devices or teach both unit options as they are equivalent and cover the same content.*

CS Discoveries Curriculum Overview

The following pages provide an overview of each of the units in the CS Discoveries curriculum. For each unit, there is a description of the unit, big questions answered throughout the unit, and learning goals. More information about each unit can be found on the course overview page of our website: http://studio.code.org/courses/csd

Problem Solving and Computing

Problem Solving and Computing is a highly interactive and collaborative introduction to the field of computer science, as framed within the broader pursuit of solving problems. Students practice using a problem solving process to address a series of puzzles, challenges, and real-world scenarios. Next, students learn how computers input, output, store, and process information to help humans solve problems. The unit concludes with a project in which students design an application that helps solve a problem of their choosing.

Big Questions

Chapter 1 - The Problem Solving Process

- What strategies and processes can I use to become a more effective problem solver?

Chapter 2 - Computers and Problem Solving

- How do computers help people to solve problems?
- How do people and computers approach problems differently?
- What does a computer need from people in order to solve problems effectively?

Unit Goals

- Identify the defined characteristics of a computer and how it is used to solve information problems.
- Use a structured problem solving process to address problems and design solutions that use computing technology.
- Create a collaborative classroom environment where students view computer science as relevant, fun, and empowering.

Web Development

In Web Development, students are empowered to create and share content on their own web pages. They begin by thinking about the role of the web and how it can be used as a medium for creative expression. As students develop their pages and begin to see themselves as programmers, they are encouraged to think critically about the impact of sharing information online and how to be more critical consumers of content. They are also introduced to problem solving as it relates to programming while they learn valuable skills such as debugging, using resources, and teamwork. At the conclusion of the unit, students will have created a personal website they can publish and share.

Big Questions

Chapter 1 - Creating Web Pages

- Why do people create websites?
- How can text communicate content and structure on a web page?
- How do I safely and appropriately make use of the content published on the internet?
- What strategies can I use when coding to find and fix issues?

Chapter 2 - Multi-Page Websites

- How can websites be used to address problems in the world?
- What strategies can teams use to work better together?
- How do I know what information can be trusted online?

Unit Goals

- Create digital artifacts that use multiple computer languages to control the structure and style of their content.
- Create a website as a form of personal expression.
- Use different programming languages to solve different problems.
- Examine their role and responsibilities as both creators and consumers of digital media.

Interactive Animations and Games

In the Interactive Animations and Games unit, students build on their coding experience as they create programmatic images, animations, interactive art, and games. Starting off with simple, primitive shapes and building up to more sophisticated sprite-based games, students become familiar with the programming concepts and the design process computer scientists use daily. They then learn how these simpler constructs can be combined to create more complex programs. In the final project, students develop a personalized, interactive program.

Big Questions

Chapter 1 - Images and Animations

- What is a computer program?
- What are the core features of most programming languages?
- How does programming enable creativity and individual expression?
- What practices and strategies will help me as I write programs?

Chapter 2 - Building Games

- How do software developers manage complexity and scale?
- How can programs be organized so that common problems only need to be solved once?
- How can I build on previous solutions to create even more complex behavior?

Unit Goals

- Create an interactive animation or game that includes basic programming concepts such as control structures, variables, user input, and randomness.
- Work with others to break down programming projects using sprites and functions.
- Give and respond constructively to peer feedback, and work with their teammates to complete a project.
- View yourself as a computer programmer, and see programming as a fun and creative form of expression.

The Design Process

The Design Process unit transitions students from thinking about computer science as a tool to solve their own problems towards considering the broader social impacts of computing. Through a series of design challenges, students are asked to consider and understand the needs of others while developing a solution to a problem. The second half of the unit consists of an iterative team project, during which students have the opportunity to identify a need that they care about, prototype solutions both on paper and in App Lab, and test their solutions with real users to get feedback and drive further iteration.

Big Questions

Chapter 1: User Centered Design

- How do computer scientists identify the needs of their users?
- How can we ensure that a user's needs are met by our designs?
- What processes will best allow us to efficiently create, test, and iterate upon our design?

Chapter 2: App Prototyping

- How do teams effectively work together to develop software?
- What roles beyond programming are necessary to design and develop software?
- How do designers incorporate feedback into multiple iterations of a product?

Unit Goals

- See the design process as a form of problem solving that prioritizes the needs of a user.
- Identify user needs and assess how well different designs address them.
- Develop paper and digital prototypes, gather and respond to feedback about a prototype, and consider ways different user interfaces do or do not affect the usability of their apps.
- Understand other roles in software development, such as product management, marketing, design, and testing, and how to use what they have learned about computer science as a tool for social impact.

Data and Society

The Data and Society unit is about the importance of using data to solve problems and it highlights how computers can help in this process. The first chapter explores different systems used to represent information in a computer and the challenges and tradeoffs posed by using them. In the second chapter, students learn how collections of data are used to solve problems, and how computers help to automate the steps of this process. In the final project, students gather their own data and use it to develop an automated solution to a problem.

Big Questions

Chapter 1: Representing Information

- Why is representation important in problem solving?
- What features does a representation system need to be useful?
- What is necessary to create usable binary representation systems?
- How can we combine systems together to get more complex information?

Chapter 2: Solving Data Problems

- How does data help us to solve problems?
- How do computers and humans use data differently?
- What parts of the data problem solving process can be automated?
- What kinds of real world problems do computers solve by using data?

Unit Goals

- Understand the role of data and data representation in solving information problems.
- Explain the necessary components of any data representation scheme, as well as the particulars of binary and the common ways that various types of simple and complex data are represented in binary code.
- Design and implement a data-based solution to a given problem and determine how the different aspects of the problem solving process could be automated.

Creating Apps with Devices

In this unit, students explore the role of physical devices in computing. Using App Lab and *either* an Adafruit's Circuit Playground or BBC micro:bit, students develop programs that utilize the same hardware inputs and outputs that you see in the smart devices, looking at how a simple rough prototype can lead to a finished product. Then, students explore how physical devices can be used to react to the world around them using a "maker" mindset to create prototypes with everyday materials.

This unit has two options: Option A uses the Adafruit Circuit Playground, and Option B uses the BBC micro:bit. Teachers should pick whichever option matches the devices they have in their classroom - they do *not* need to purchase both sets of devices or try to teach both options.

Big Questions

Chapter 1: Inputs and Outputs

- What inputs and outputs are available on a physical device?
- What inputs and outputs are available on an app?
- How can we create apps that use a physical device to control a digital app?

Chapter 2: Building Physical Prototypes

- How can a physical device use sensors to react to a physical environment?
- How can simple hardware be used to develop innovative new products?

Unit Goals

- Design and build a physical computing device that integrates physical inputs and outputs with digital apps.
- Create app prototypes that use a physical device to solve real-world problems
- Use physical computing to solve problems in fun and innovative ways.

AI and Machine Learning

In the AI and Machine Learning unit, students learn how computers can find patterns in data to make decisions. Students use the Problem Solving Process for machine learning to define a problem, prepare their data, train a model, then test and evaluate their model for accuracy and potential bias. Students explore a variety of scenarios and datasets that lend themselves to machine learning. They also explore some of the modern problems with machine learning, especially around bias and impact.

This unit is designed as an *optional* unit within a year-long CS Discoveries course. Teachers can decide to implement this instead of one of the other units. This is especially useful for classrooms that do not have physical devices and are unable to implement the Creating Apps with Devices unit.

Big Questions

Chapter 1: Understanding Machine Learning

- How does machine learning find patterns in data to make decisions?
- How can we avoid bias when training a machine learning model?

Chapter 2: Design a Machine Learning App

- How can machine learning be used to solve problems in our community?

Unit Goals

- Create a machine learning model in AI Lab to solve a problem, and use App Lab to create an app that uses their model.
- Understand how machine learning models make decisions from data
- Create machine learning models from their own data to solve problems in their community.