Explorations in Coding II

In this second course of the Explorations in Coding series, students advance their knowledge of foundational concepts and skills of programming and computer science (CS). Students build their understanding of programming through coding explorations, practices, and projects in Python, where they create animations, games, and other interactive programs. Supplemental online and unplugged activities foster students’ deeper understanding of big ideas in computer science as well as cultivate creative and critical thinking skills. Explorations in Coding students remain engaged and motivated throughout the course as they collaborate with peers, explore real world applications of computer science, and make personal connections to their own interests and experiences.

Prerequisites
Students taking this course should have previously completed Explorations in Coding I or another introductory coding course. They should have a basic understanding of sequencing algorithms, using variables, and writing and debugging text-based code.

Grade Levels
Appropriate for middle school or early high school grades.

Course Goals
- Introduce computer science as an engaging and relevant discipline.
- Develop foundational skills and knowledge in programming and computer science.
- Strengthen problem solving and critical thinking skills.
- Foster creativity, collaboration, and communication.
- Explore issues raised by present and future societal impacts of computing.
- Demonstrate that all students can be successful in computer science.

Curriculum Overview
This course is organized into ten modules, which begin after Modules 1-9 of the recommended prerequisite course: Explorations in Coding I. Each module builds computer science competency through highly engaging, interactive coding projects as well as a series of online and unplugged activities.

Course Modules
Module 10: Experimenting with Loops
Module 11: Variables and Loops
Module 12: Intro to User-Centered Design
Module 13: Experimenting with If Statements
Module 14: Improvement through Iteration
Module 15: Intro to Game Design
Module 16: Experimenting with Lists
Module 17: Automation
Module 18: Digital Data
Module 19: Cumulative Review
Module Overviews

Module 10: Experimenting with Loops
Students will experiment with algorithms and programs that use one of the core building blocks of computer programs: loops. Students will connect computer science to their own interests and experiences as they learn about the first computer programmer and other firsts in computer science history. They will experiment with:
- For loops
- Looping algorithms
- Firsts in computer science history

Module 11: Variables and Loops
Students further their learning about the use of repetition in programs. Students explore several ways they can adjust visual effects in animations through the use of variables inside of for loops. Students also expand their ability to understand and explain loops in real world scenarios. Topics include:
- Using variables in a loop
- Incremental change
- Real world looping algorithms

Module 12: Intro to User-Centered Design
Students advance their knowledge about user input through the lens of user-centered design. Students consider how appearance and functionality affect the user’s experience. Students will also design and evaluate user interfaces that harness a variety of types of user input. Topics include:
- User-Centered Design
- User Input
- Click Events
- User Interfaces

Module 13: Experimenting with If Statements
Students will write programs that use a second core building block of computer programs, conditionals, to allow users to make decisions while using a program. Students consider how the test statements and formatting of their programs affect the output of their programs. Topics include:
- Conditionals
- Test statements
- Program readability
- Flowcharts and pseudocode

Module 14: Improvement through Iteration
Students consider how hardware and software can be improved through an iteration process. Students will use user feedback practices to inform decisions to improve their ideas and programs. Students will create, debug, and improve programs that use random choice to determine output. Topics include:
- Iteration
- Debugging
- User Feedback
- Random selection

Module 15: Intro to Game Design
Students will examine basic principles of game design, using them to collaboratively create their own video game concepts. Students will combine their conceptual knowledge and coding skills with conditionals and user input to program and iterate upon a simple game. Topics include:
- Combining conditionals and user input
- AND statements
- Game design
- Prototyping
Module 16: Experimenting with Lists
Students learn how lists can be used in Python to hold a set of data or variables. Students will consider real world software examples that manage and edit lists of items. Students will create a user-friendly interface for list management. Topics include:
- Applications of lists
- Display function
- Append and remove commands

Module 17: Automation
Students will create games that combine loops and conditionals to automate play and feedback for the user. Students will consider real world examples of automation and automation’s impact on everyday activities and career opportunities. Topics include:
- Automation
- Combining loops and conditionals
- Nesting
- Impacts of Computing

Module 18: Digital Data
Students examine how lists can be used within loops to repeat a set of commands with a set of data. Students extend their thinking about digital data through real world considerations of social media, branding, privacy, and cybersecurity. Topics include:
- Combining loops and lists
- Data representation
- Data privacy and cybersecurity

Module 19: Cumulative Review
Students demonstrate the ability to use and combine computer science and coding concepts from throughout the course. Topics include:
- Cumulative review
- Culminating projects
- Cumulative test

Course Materials
All digital course materials and resources are provided for this class. Additional materials may include:
- Screen-cast or projection device to project provided lesson slides or videos
- Printed copies of provided activity sheets and materials
- Paper, pens/pencils, and other basic classroom supplies

Optional:
- Pocketed folder or binder
- Headphones/earbuds (for watching instructional videos)

Differentiation
The Explorations in Coding courses provide adaptive scaffolding, indicators of common misconceptions, and diverse activity types to provide a dynamic curriculum strategically designed to support students with varying academic backgrounds or previous experience with technology. Coding extensions and other challenge activities are additionally provided throughout the course materials to engage and motivate advanced or accelerated students.
Standards-aligned
The Explorations in Coding courses are aligned with the Computer Science Teachers Association (CSTA) K-12 Computer Science Standards and the Common Core State Standards (CCSS): Standards for Mathematical Practices. The CSTA K-12 Computer Science Standards are the national benchmark for introducing fundamental concepts in computer science at all grade levels. The CCSS Standards for Mathematical Practice outline key processes and proficiencies for effective problem solving. The Explorations in Coding lesson guides indicate each module’s alignment to both sets of standards.

About Explorations in Coding
Computer science is the science behind computing and learning how to use the power of computers to solve big problems. It is the study of the ideas, ways of thinking, programming languages, hardware and software needed to solve problems with computers, and encompasses the study of computers and algorithmic processes, their principles, their designs, their applications, and their impact on society.

The Explorations in Coding courses seek to reinforce the understanding that computer science is intellectually important and that all students should have a basic understanding of computer science and programming principles, technology enabled applications from calculators to simulations, and awareness of the societal impacts of computing. Some key values for advancing computer science education for students include:

Computer science leads to multiple career paths. There are tremendous job opportunities in computer science. Professionals in every 21st century discipline need to understand computing to be productive and competitive in their fields.

Computer science teaches problem solving. Computer science requires students to apply critical thinking skills to consider the context of the situation, work cooperatively on solutions, and balance the robustness, user-friendliness, and efficiency of solutions to real-world issues.

Computer science supports and links to other disciplines. Computer science is not simply one more discipline to learn; rather, it is a methodology that enables the study of and innovation in other disciplines.

Computer science can engage all students. It allows hands-on practice and offers students opportunities to solve computational problems relevant to their own interests, passions, and experiences.

Essential CS Practices
The Explorations in Coding courses seeks to reinforce the following essential computer science practices:

- Problem-solve: find solutions to challenging, real-world problems
- Persevere: try again and again, even when something is very hard
- Collaborate: work together to achieve something greater than could be done alone
- Create: design and develop interesting, relevant computational artifacts
- Communicate: strengthen written and verbal skills to describe computing and one’s work
- Think critically: identify impacts of computing; draw connections to the real world
Student Collaboration
This course is designed to allow teachers to encourage and support student cooperation and collaboration. Students will work on many of the coding and unplugged activities in pairs, groups or teams. Even if students do not choose their own partners, they should be responsible for working well together. For examples, if using pair programming, students should frequently trade roles as “driver” (controlling the computer) and “navigator” (assisting the driver and keeping him/her focused on the big picture), and students should view their teammates as sources of primary support during those activities. To encourage this, students can be encouraged to ask their partners and teammates for help before asking the teacher.

Student Guidelines
The following are sample guidelines for student expectations for the course. These may be modified or adapted by teachers. Many aspects of this course will be exciting and extremely rewarding, though some parts will not be easy. Those parts may be very challenging and even frustrating at times, and this is okay. We consider mistakes and failure as our first attempts in learning, and we persevere and learn from these. In this class, teachers and students work and learn together to achieve more than they could do alone.

Students are expected to follow all school and classroom prescribed rules. Some sample classroom rules can include:

- Follow teachers’ directions.
- Complete all assigned work (to the best of one’s ability).
- Come to class prepared and on time.
- Keep the classroom clean and orderly.
- Use computers and the Internet responsibly.

Sample classroom norms can include:

- Be respectful.
- Be safe and responsible.
- It’s okay to make mistakes.
- It’s okay to not have the answers (yet).
- Never give up. (And, don’t let your teammates give up.)

Syllabus Note
This syllabus is subject to change; and should be considered a template that teachers may use and adapt to their classrooms.