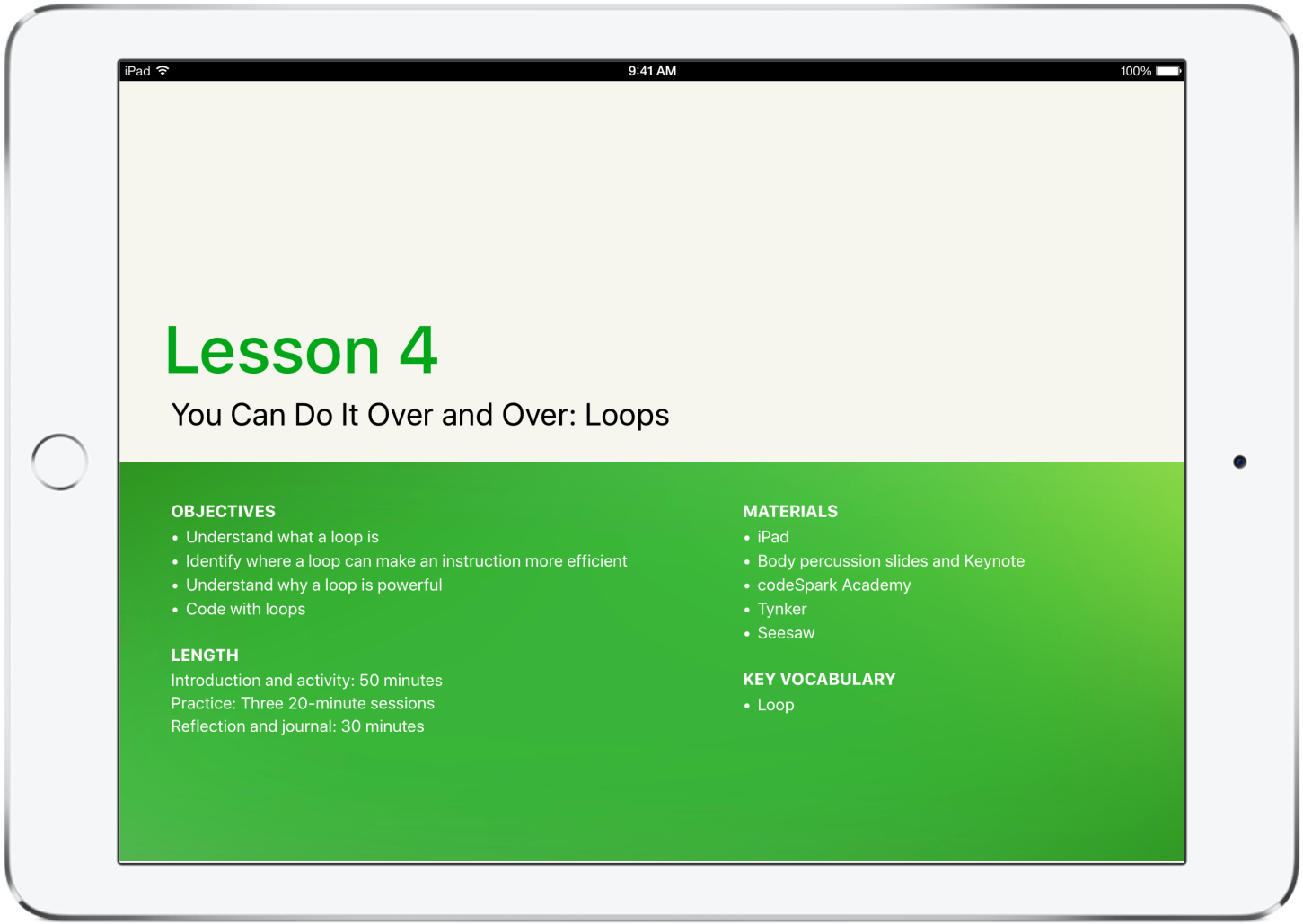
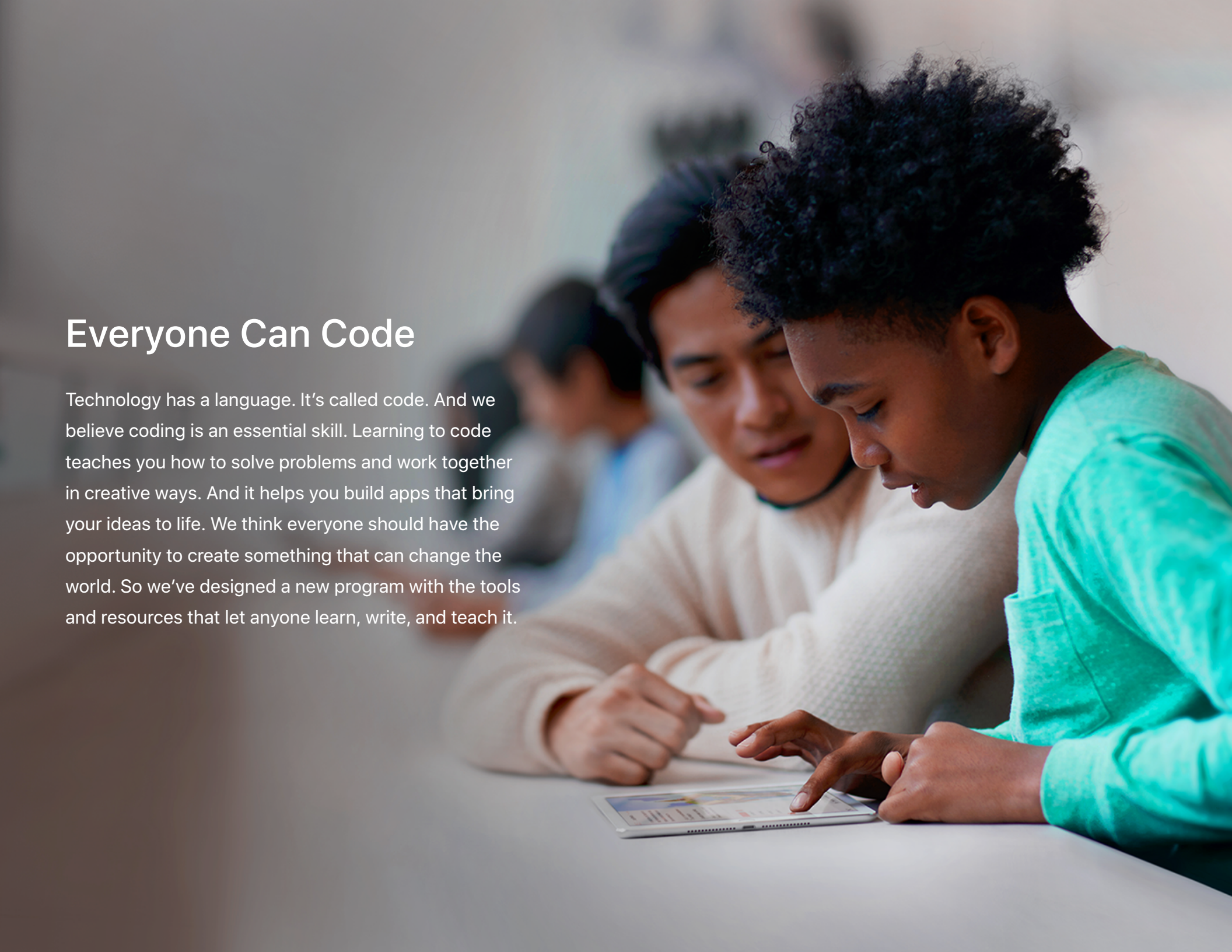
Get Started with Code Curriculum Guide

May 2017



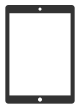
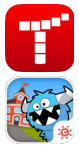
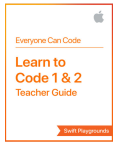
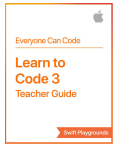
Everyone Can Code

Technology has a language. It's called code. And we believe coding is an essential skill. Learning to code teaches you how to solve problems and work together in creative ways. And it helps you build apps that bring your ideas to life. We think everyone should have the opportunity to create something that can change the world. So we've designed a new program with the tools and resources that let anyone learn, write, and teach it.



Everyone Can Code Curriculum

The Everyone Can Code program includes a range of resources that take students all the way from no coding experience to building their first apps. The table below provides an overview of all the free teaching and learning resources available.

Curriculum	Device	Audience	App	Prerequisites	Overview	Learning materials	Support resources	Number of lesson hours included
		Kindergarten through grade 2		None	Begin to think like coders with hands-on explorations of coding concepts using visual-based apps.	• codeSpark Academy app lessons • Tynker Space Cadet course	• Get Started with Code 1: Teacher Guide	20 hours, including Teacher Guide and app lessons
		Grades 3 through 5		None	Explore fundamental coding concepts and practice thinking like coders using visual-based apps.	• Tynker Dragon Spells course	• Get Started with Code 2: Teacher Guide	20 hours, including Teacher Guide and app lessons
		Middle school and up		None	Learn fundamental coding concepts using real Swift code.	• Swift Playgrounds app • Learn to Code 1 & 2 lessons • iTunes U course	• Learn to Code 1 & 2: Teacher Guide • Apple Teacher Learning Center Swift Playgrounds badges	45 hours, including Teacher Guide and Learn to Code 1 & 2 lessons
		Middle school and up		Learn to Code 1 & 2	Expand coding skills and start thinking more like an app developer.	• Swift Playgrounds app • Learn to Code 3 lessons	• Learn to Code 3: Teacher Guide	20 hours, including Teacher Guide and Learn to Code 3 lessons
		High school and college		None	Get practical experience with the tools, techniques, and concepts needed to build a basic iOS app from scratch.	Intro to App Development with Swift book and project files	• Intro to App Development with Swift: Teacher Guide • MobileMakersEdu professional learning workshops	90 hours

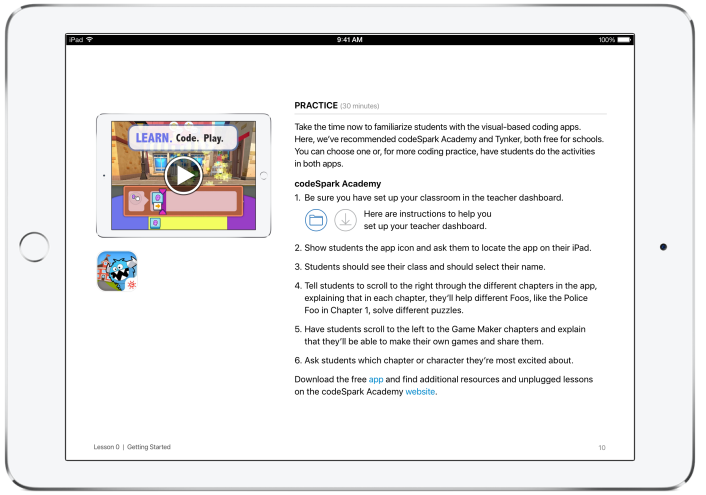
Overview

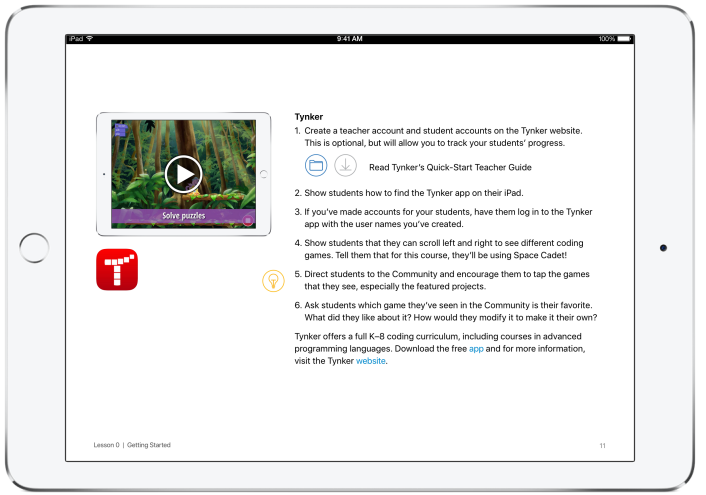
The early years of schooling are a great time to introduce coding concepts as a way of thinking about the everyday and digital worlds, and to develop foundational skills in computational thinking. Apps that are specifically designed for younger learners, such as codeSpark Academy and Tynker, use visual-based coding puzzles to develop problem-solving skills, encourage persistence, and promote creativity. codeSpark Academy is designed for learners ages five to seven. The game's word-free interface allows prereaders, English-language learners, and students with reading challenges to all play. In Tynker, students ages five to eleven begin experimenting with visual blocks, then progress to text-based coding as they solve puzzles, build projects, and program drones and robots.

In the classroom

Tynker and codeSpark Academy, along with the lessons in the Get Started with Code Teacher Guides, are designed to help you bring coding into the early primary classroom. The lessons highlight key coding concepts, while demonstrating how coding is a way of thinking that can be applied to other learning areas and everyday life.

The Teacher Guides provide the support you need to help your students solve the coding puzzles—no matter what your level of coding experience. Extension activities, reflection questions, journal prompts, a grading rubric, and more are included to help you deepen students' understanding of the material. You can teach the lessons in a single block or in sections. Correlations maps in the appendices provide a preliminary alignment of the lessons to the Interim Computer Science Teachers Association K–12 Computer Science Standards for Level 1.





Key Features

codeSpark Academy

Learn. Students solve puzzles in this word-free game to learn basic computer science concepts such as sequencing, looping, conditional statements, and more.

Build. Students then apply their knowledge by coding their own projects in Game Maker.

Self-directed. No coding experience is needed to teach, learn, or play. The curriculum and teacher dashboard for progress reports are available free to educators in 10 languages.

Tynker

Coding environment. Students move at their own pace through scaffolded coding puzzles to learn concepts and apply them creatively.

Automatic assessment. A teacher dashboard assesses students' mastery of skills with puzzles, quizzes, and code analyses.

New Swift feature. As students solve puzzles, they can switch between visual blocks and Swift blocks, allowing them to get familiar with Swift and preparing them for future programming.

Get Started Teacher Guides

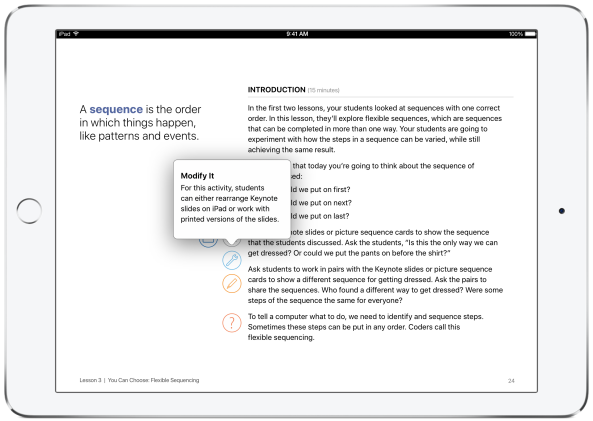
Downloadable files. Template files for student activities and Keynote presentations support in-class student learning.

Answer keys. Tynker and codeSpark Academy puzzle solutions make it easy for you to help students who are stuck.

Student work examples. See what the activities can look like.

Reflections. These questions and prompts for class discussion help you review and reinforce the connection between applying the concept inside and outside of a coding environment.

Tips and examples. Ideas for extending or simplifying lessons are included throughout.



Course Outlines

Get Started with Code 1

By engaging in interactive, hands-on explorations of coding concepts in the context of everyday situations, students will begin to think like coders. They'll learn about commands, sequences, loops, events, and algorithms. Working collaboratively, students will practice predicting the output of their code, as well as debugging their own and others' code. They'll also practice using their skills in visual-based coding apps, solving puzzles and designing their own creations.

Lesson 0—Getting Started. Find out what students already know about apps and coding, set up the class working wall, and introduce them to the key apps they'll use across the lessons.

Lesson 1—You Can Order It: Introduction to Sequencing. Students explore everyday sequences, construct a sequence based on a familiar story, and solve puzzles in visual-based coding apps using simple sequences.

Lesson 2—You Can Step It: Creating Sequences. By examining the importance of order when sequencing instructions, students learn that the same actions can be reordered to create a different sequence, and design their own crazy dance. They discover more commands and solve more complex problems in the coding apps.

Lesson 3—You Can Choose It: Flexible Sequencing. Students learn that some steps in a sequence can be flexibly ordered, and create their own flexible sequences. They explore the different ways a puzzle can be solved and share their code solutions with other students.

Lesson 4—You Can Do It Over and Over: Loops. Students identify loops in everyday contexts and get creative with building their own body percussion loops. They look at how loops are represented in coding and use loops to streamline and simplify their code.

Lesson 5—You Can Fix It: Debugging. Understanding the importance of persistence and debugging in a range of contexts, students review and apply their new coding skills to solve a challenge and debug other students' solutions. They practice predicting the output of their code and identifying bugs when code doesn't execute as planned.

Lesson 6—You Can Prompt It: Events and Actions. Students explore how events can make their app play and code more engaging and responsive, and learn how to code with events. They consider how we prompt events in everyday life and create a robot remote control to practice calling events.

Lesson 7—You Can, If You Follow the Rule: IF Statements. Students are introduced to conditional statements and find out how to recognize IF statements in their everyday lives. They think about the IF statements that work as the rules in familiar board games. Students code with IF statements, making their code more responsive to conditions in the environment.

Lesson 8—You Can Solve It: Algorithms. Bringing together everything they've learned so far, students design algorithms involving a series of steps to solve a problem. They begin with simple recipes and progress to designing and coding their own maze game.

Course Outlines (continued)

Get Started with Code 2

In Get Started with Code 2, students will explore fundamental coding concepts and practice thinking like coders. Along with learning about algorithms, functions, loops, conditional statements, and variables, they'll discover the basics of user interface design. Students will work both collaboratively and individually as they strengthen their coding skills by solving real coding problems, testing each other's code, and designing programs for a range of bots. They'll also practice these skills in Tynker, solving a range of problems and applying the concepts they learn in classroom activities.

Lesson 0—Getting Started. Find out what students already know about apps and coding, familiarize students with a visual-based coding app such as Tynker, and set up students' digital journals using an app like Seesaw.

Lesson 1—Think in Steps: Solving Problems with Algorithms. Students discover algorithms as a set of instructions for solving a problem or performing a task. In the Tynker app, students hatch a dragon of their choice, then build algorithms to solve puzzles while learning sequencing skills. Students design and test algorithms in classroom activities.

Lesson 2—Think in Fixes: Debugging. Students explore finding and fixing errors in their algorithms and in their coding. In Tynker, they modify algorithms that have bugs in order to create a correct program for solving the puzzles.

Lesson 3—Think in Circles: Looking for Loops. Introduced to loops as repetitive patterns, students design and test an algorithm to create a Loopy Snake. In Tynker, they use for loops to solve puzzles by spotting patterns.

Lesson 4—Think in Bits: Composition and Decomposition. To devise an algorithm for performing a Cup Song, students break down the routine into component moves. In Tynker, they solve problems by breaking them down into smaller subproblems.

Lesson 5—Think in Sets: Abstraction. Students explore similarity and generalization as they categorize objects into sets, then explain their rationale. In Tynker, they use abstraction to spot similarities between problems, solving increasingly complex puzzles with all their new coding tools.

Lesson 6—Think in Patterns: Forming Functions. In building a performance routine for a Command Bot, students break it down into functions, then exchange algorithms to test predicted and actual outcomes. In Tynker, students use name and call functions as they reuse sets of instructions to code more efficiently.

Lesson 7—Think in Specifics: Conditional Statements. Students go on a Virtual Travel Adventure, defining their destination with a set of qualifying conditions using if statements. In Tynker, they use if statements to handle decision and alternatives within the puzzles.

Lesson 8—Think in Cycles: While Loops and Nested Loops. Running a virtual donut stand, students use while and nested loops to create algorithms for Donut Bot to ice enough donuts for every customer. In Tynker, students use loops to shorten their code.

Lesson 9—Think In and Outside the Box: Variables, Input, and Output. Using variables to design an algorithm for a Poetry Jam Slam, students perform a song or rap based on audience input. In Tynker, they use variables as they tackle more complex puzzles, using all the coding skills they've learned so far.

Lesson 10—Think in Practice: Design UI. Students analyze what makes a good design, and come up with a sign for their school. In the Tynker activity, they use all the skills they've acquired, completing the lessons of Get Started with Code 2.

Additional Information

Download the Get Started with Code resources

- Tynker
- codeSpark Academy
- Get Started with Code 1
- Get Started with Code 2

Download the Swift Playgrounds resources

- Learn to Code 1 & 2: Teacher Guide
- Learn to Code 3: Teacher Guide
- Swift Playgrounds app

Download the App Development with Swift guides

- Intro to App Development with Swift
- Intro to App Development with Swift: Teacher Guide

Additional resources

- Learn more about the Everyone Can Code program.
- Learn more about Swift.
- Learn more about Xcode.
- Connect with other educators in the Apple Developer Forums.
- Learn more about codeSpark Academy.
- Learn more about Tynker.

Curriculum Alignment: Get Started with Code 1

Get Started with Code 1 lessons align with the algorithms and programs concept of the Computer Science Teachers Association's (CSTA) Interim 2016 Computer Science Standards for Level 1 for grades K–2. As students work through the Get Started with Code 1 lessons and activities, they discover the fundamental concepts underlying computational thinking and take their first steps writing code.

Students work independently and collaboratively both in activities and using the coding apps codeSpark Academy and Tynker.

Alignment Get Started with Code 1—CSTA K–12 Computer Science Standards Level 1 for Grades K–2

Get Started with Code 1 Chapters	CSTA Standard	1A-A-7-1 Credit	1A-A-5-2 Construct Programs	1A-A-5-3 Design Document	1A-A-4-4 Representing Data	1A-A-3-5 Decompose	1A-A-3-6 Categorizing Items	1A-A-3-7 Algorithms	1A-A-6-8 Analyze and Debug
	You Can Order It: Introduction to Sequencing		●	●	●	●		●	●
	You Can Step It: Creating Sequences		●	●	●			●	●
	You Can Choose It: Flexible Sequencing		●	●	●	●		●	●
	You Can Do It Over and Over: Loops		●	●	●			●	●
	You Can Fix It: Debugging		●	●	●	●		●	●
	You Can Prompt It: Events and Actions		●	●	●	●		●	●
	You Can, If You Follow the Rule: IF Statements		●	●		●		●	●
	You Can Solve It: Algorithms		●	●	●	●		●	●

Key: ● Aligns to standard

Curriculum Alignment: Get Started with Code 2

Get Started with Code 2 lessons align with the algorithms and programs concept of the Computer Science Teachers Association's (CSTA) Interim 2016 Computer Science Standards for Level 1 for grades 3–5. As students work through the Get Started with Code 2 lessons and activities, they strengthen their understanding of the fundamental concepts underlying computational thinking and develop their code-writing skills.

Students work independently and collaboratively both in activities and using the coding app Tynker.

Alignment Get Started with Code 2—CSTA K–12 Computer Science Standards Level 1 for Grades 3–5

	CSTA Standard	1B-A-2-1 Collaboration Strategies	1B-A-7-2 Citation and Documentation	1B-A-5-3 Plan	1B-A-5-4 Construct Programs	1B-A-5-5 Mathematical Operations	1B-A-3-6 Decompose	1B-A-3-7 Algorithms	1B-A-6-8 Analyze and Debug
Get Started with Code 2 Chapters	**Think in Steps: Solving Problems with Algorithms**	●		●	●		●	●	●
	Think in Fixes: Debugging	●		●	●		●	●	●
	Think in Circles: Looking For Loops	●		●	●		●	●	●
	Think in Bits: Composition and Decomposition	●		●	●		●	●	●
	Think in Sets: Abstraction	●			●			●	●
	Think in Patterns: Forming Functions	●	●	●	●		●	●	●
	Think in Specifics: Conditional Statements	●	●	●	●		●	●	●
	Think in Cycles: While Loops and Nested Loops	●		●	●		●	●	●
	Think in and Outside the Box: Variables, Input and Output	●		●	●	●	●	●	●
	Think In Practice: Design - UI	●	●	●	●		●		●

Key: ● Aligns to standard

Features are subject to change. Some features may not be available in all regions or all languages.

© 2017 Apple Inc. All rights reserved. Apple, the Apple logo, iTunes U, Keynote, and Xcode are trademarks of Apple Inc., registered in the U.S. and other countries. Swift and Swift Playgrounds are trademarks of Apple Inc. Other product and company names mentioned herein may be trademarks of their respective companies. Product specifications are subject to change without notice. This material is provided for information purposes only; Apple assumes no liability related to its use. May 2017