**Goal:** Use Swift commands to tell Byte to move and collect a gem.

Your character, Byte, loves to collect gems but can’t do it alone. In this first puzzle, you’ll need to write Swift commands to move Byte across the puzzle world to collect a gem.

1. Look for the gem in the puzzle world.
2. Enter the correct combination of the `moveForward()` and `collectGem()` commands.
3. Tap Run My Code.

```swift
moveForward()
moveForward()
moveForward()
collectGem()
```
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.

<table>
<thead>
<tr>
<th>Curriculum</th>
<th>Device</th>
<th>Audience</th>
<th>App</th>
<th>Prerequisites</th>
<th>Overview</th>
<th>Learning materials</th>
<th>Support resources</th>
<th>Number of lesson hours included</th>
</tr>
</thead>
<tbody>
<tr>
<td>Years K through 2</td>
<td>None</td>
<td>None</td>
<td>codeSpark Academy app lessons</td>
<td>Get Started with Code 1 Teacher Guide</td>
<td>30 hours, including Teacher Guide and app lessons</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Years 3 through 5</td>
<td>None</td>
<td>None</td>
<td>Tynker Space Cadet course</td>
<td>Get Started with Code 2 Teacher Guide</td>
<td>36 hours, including Teacher Guide and app lessons</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Years 6 through 10</td>
<td>None</td>
<td>None</td>
<td>Tynker Dragon Spells course</td>
<td>Get Started with Code 2 Teacher Guide</td>
<td>36 hours, including Teacher Guide and app lessons</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Years 6 through 10</td>
<td>None</td>
<td>None</td>
<td>Swift Playgrounds app</td>
<td>Up to 85 hours, including Teacher Guide and Learn to Code 1 &amp; 2 lessons</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Senior high school and university</td>
<td>None</td>
<td>None</td>
<td>Swift Playgrounds app</td>
<td>Intro to App Development with Swift Teacher Guide</td>
<td>90 hours</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Senior high school and university</td>
<td>None</td>
<td>None</td>
<td>App Development with Swift book and project files</td>
<td>App Development with Swift Teacher Guide</td>
<td>180 hours</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Overview

Swift Playgrounds is a free iPad app from Apple that makes learning and experimenting with code interactive and fun. Students can solve puzzles to master the basics using Swift — a powerful programming language created by Apple and used by the pros to build many of today’s most popular apps.

The app comes with a complete set of Apple-designed lessons called Learn to Code. Using real Swift code, students solve puzzles and meet characters they can control with just a tap. By exploring and solving rich puzzle worlds, students develop coding skills that become the foundation of their programming knowledge. Additional challenges and connected device playgrounds are also included to let students apply what they’ve learned to new contexts.

In Learn to Code 1 & 2, students learn concepts such as commands, debugging, functions, loops, algorithms and more. The lessons require no previous experience, so they’re perfect for first-time coders. Learn to Code 3 helps students expand their coding skills to start thinking more like an app developer. An optional app design section in the Teacher Guide helps teachers lead students through an app design process.

In the classroom
Learn to Code 1, 2 & 3, along with the lessons in the Teacher Guides, are for use with older primary school and junior high school students. The materials are flexible and usable in any learning environment, and can be used in a stand-alone coding class or as part of an intro to coding program. Lessons are designed for 45- to 60-minute classes, and some span multiple periods. The suggested amount of time needed to complete each section in a lesson is included, so if you teach a less-structured class, like an after-school program, you can divide up the lesson.

The Teacher Guides provide support that allows teachers with or without coding experience to teach with them. It’s recommended that students and teachers have fundamental knowledge of the coding concepts taught in Learn to Code 1 & 2 before moving on to Learn to Code 3.
Key Features

Real Swift, real iOS code. At the heart of Swift Playgrounds is the same Swift programming language that’s used to build many of the leading apps in the App Store today. The skills students learn in Swift Playgrounds don’t just translate into useful skills elsewhere, they’re the exact skills they need to build apps.

Interactive environment. Create code on the left side of the screen and instantly see the results on the right, with just a tap.

Immersive animations. Each section starts with an immersive animation that relates coding concepts to real life, aiding student understanding.

Built-in glossary. Definitions help students understand specific terms.

Helpful hints. Students can get help along the way if they get stuck. In many cases, hints change dynamically as they enter code.

Shortcut Bar. QuickType suggestions for code appear at the bottom of the screen that let students enter the code they need by just tapping the Shortcut Bar.

Onscreen keyboard. A keyboard dedicated to Swift provides quick access to the numbers and symbols most commonly used in Swift.

Accessibility. Swift Playgrounds was designed with accessibility in mind from day one. It takes advantage of the many powerful accessibility features of iOS, including Switch Control and VoiceOver, and even provides additional voice commentary on the actions of characters as students control them with code.

Record and share. Students can record what they do on screen to demonstrate their work.

Review code. Run code faster or slower, or step through it to highlight the lines of code as they execute, making it easier for students to identify where errors might occur.

Touch to edit. Drag complex structures that wrap other code, such as loops and function definitions, around existing code. Just touch the keyword (such as ‘for’) and the drag controls appear on screen.

Edit in place. Edit numeric values, colours and operators quickly and easily using a pop-over keypad.
Support Resources

Learn to Code 1 & 2 Teacher Guide
Designed for use with older primary school and junior high school students, this Teacher Guide will help any teacher bring Learn to Code 1 & 2 into their classroom. The lessons highlight key coding concepts while demonstrating how coding is a way of thinking that can be applied to other subjects and everyday life. Enhanced activities, review and reflection activities, a grading rubric and Keynote presentations are included. The guide represents 40 to 45 hours of core coding lessons, with up to 45 hours of supplemental activities that help students apply what they’ve learned and start designing their very own app. Curriculum correlations are included, showing alignment with various national and international curriculum standards for computer science.

Learn to Code 3 Teacher Guide
The guide is geared toward older primary school and junior high school students. It includes 20 hours of core coding lessons, with up to 25 hours of supplemental activities that help students apply what they’ve learned and start designing their very own app. Building on coding skills from Learn to Code 1 & 2, it includes story activities, code review lessons, Keynote presentations, journal prompts, a grading rubric and more, to help teachers bring these lessons into the classroom. Curriculum correlations are included, showing alignment with various national and international curriculum standards for computer science.

Learn to Code 1 & 2: iTunes U course
This iTunes U course brings the Learn to Code 1 & 2 Teacher Guide to life through video lessons and additional resources. The videos are also a great way for teachers to see how they can bring the Teacher Guide lessons to life in a classroom.

Apple Teacher Program: Earn Swift Playgrounds badges
The Apple Teacher Program is a free professional learning program designed to support and celebrate teachers. It offers self-paced learning materials, tips, inspiration and news. Apple Teachers can visit the Apple Teacher Learning Center to complete quizzes on learning and teaching with Swift Playgrounds, and earn four new badges. They’ll then receive an updated Apple Teacher logo featuring Swift Playgrounds to share their accomplishment.
Course Outlines

Learn to Code 1

By solving puzzles in a dynamic 3D puzzle world, students will develop a set of coding skills to build up their basic programming vocabulary. Their coding journey begins with simple commands, functions and loops. From the start, they’ll write real Swift code — the same code used by real programmers.

Lesson 0 — Getting Started. Students get an introduction to computer science and the goals of the course.

Lesson 1 — Think Like a Computer: Commands and Sequences. Students learn about using commands and sequences in an everyday situation, then code using commands and sequences.

Lesson 2 — Think Like a Detective: Debugging. Students learn about debugging in an everyday situation, then how to debug with code.

Lesson 3 — Think Efficiently: Functions and a Bit of Loops. Students learn about using functions and for-loops in an everyday situation, then how to code using functions and for-loops.

Review and Reflect. Students review lessons 1 through 3, review their portfolios and create a community with peer-to-peer review.

Lesson 4 — Thinking Logically: Conditional Code. Students learn about using conditional code, Booleans and logical operators, then how to code using conditional code, Booleans and logical operators.

Lesson 5 — Think Again and Again: While Loops. Students learn about using while-loops in an everyday situation, then how to code using while-loops.

Lesson 6 — Think the Same Idea: Algorithms. Students learn about using algorithms in an everyday situation, then how to code using algorithms.

Review and Reflect. Students review coding concepts from lessons 3 through 6, continue reflecting on their portfolios and continue their community experience.

Learn to Code 2

Students will build on their fundamental knowledge of Swift. They’ll journey beyond simply solving puzzles and create worlds of their own. And they’ll learn about variables and types, the coding constructs that allow them to store and access information. These new skills, along with initialisation and parameters, will give them even more ways to use code to interact with their characters and the puzzle world, allowing them to change the rules of the world itself.

Lesson 7 — Think Like a NewsBot: Variables. Students learn about using variables in an everyday situation, then how to code using variables.

Lesson 8 — Think Like an Architect: Types. Students learn about using types in an everyday situation, then how to code using types and initialisation.

Lesson 9 — Think Specifically: Parameters. Students learn about using parameters in an everyday situation, then how to code using parameters.

Lesson 10 — Think Organised: Arrays. Students learn about using arrays in an everyday situation, then how to code using arrays.

Milestone Project. Students build their own worlds using the concepts learned throughout the program, creating a story to go with the world. Then they reflect on what they’ve learned using their portfolios and the community peer-to-peer review.

App Design. Students go through a design cycle that focuses on prototyping, much like the process professional app developers use.
Learn to Code 3

Learn to Code 3 helps students expand the coding skills they learned in previous lessons to start thinking more like an app developer. Learn to Code 1 & 2 is a recommended prerequisite for Learn to Code 3.

Encountering the interstellar space of Blu’s universe, students build a set of creative tools as they explore powerful coding concepts that professional developers use. As they learn about graphics and coordinates, they’ll be able to place and manipulate images, then combine these techniques with touch events to paint artistic shapes in space.

After learning about touch events, students dive into strings, giving them a way to bring their voice into Blu’s silent universe. Finally, they’ll explore event handlers as they use real events such as finger movements or taps to trigger their code. With event handlers, they’ll create animated aliens or turn the universe into a giant musical instrument. By the time they finish, they’ll be combining their skills expertly, writing their most advanced code yet!

Lesson 1 — Introduction to Learn to Code 3: Coordinates. Students learn about coordinates; review algorithms, for-loops and arrays; and then code using a combination of concepts. They also discuss what makes a great visual story.

Lesson 2 — Think Like an App Designer: Touch Events. Students review variables, types and initialisation by analysing their favourite apps, then create and initialise their own image tools in Swift Playgrounds. They also research how images influence visual stories.

Lesson 3 — Think Like an Editor: Strings. Students learn about strings in an everyday situation, then create their own text tools in Swift Playgrounds. They also research how text influences visual stories.

Lesson 4 — Think Like an Animator: Event Handlers. Students learn about event handlers by designing their own games. They then create their own action tools in Swift Playgrounds and research how interactivity impacts visual stories.

Milestone Project. Students code their own visual stories in Swift Playgrounds.

App Design. Students go through a design cycle that focuses on prototyping, much like the process professional app developers use.
Additional Information

**Swift Playgrounds requires iOS 10 or later and works on:**
- iPad Pro (9.7-inch)
- iPad Pro (12.9-inch)
- iPad
- iPad Air 2
- iPad Air
- iPad mini 4
- iPad mini 3
- iPad mini 2

**Download the Swift Playgrounds resources**
- Learn to Code 1 & 2 iTunes U Course
- Learn to Code 1 & 2 Teacher Guide
- Learn to Code 3 Teacher Guide
- Swift Playgrounds app

**Download the Get Started with Code resources**
- Tynker Coding for Kids
- codeSpark Academy
- Get Started with Code 1
- Get Started with Code 2

**Additional resources**
- Learn more about Swift Playgrounds.
- Learn more about the Everyone Can Code program.
- Learn more about Swift.
- Connect with other educators in the Apple Developer Forums.
Curriculum Correlations: Learn to Code 1, 2 & 3

The Learn to Code 1 & 2 and Swift Playgrounds exercises are targeted at older primary school and junior high school students, and align with the Processes and Production Skills strand of the Australian Curriculum: Digital Technologies for years 5–10. Teachers can follow a structured pathway through the Teacher Guides while mapping to the corresponding Digital Technologies content description.

Alignment between Learn to Code 1 & 2 and Australian Curriculum: Digital Technologies — Years 5–10 Processes and Production Skills Strand

<table>
<thead>
<tr>
<th>Content description</th>
<th>Years 5–6</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTDIP016收购，存储和验证不同类型的数据，并使用一系列软件来解释和可视化数据并创建信息</td>
<td>☑</td>
</tr>
<tr>
<td>ACTDIP017定义问题，在术语和功能要求之间，参考以前解决的问题</td>
<td>☑</td>
</tr>
<tr>
<td>ACTDIP018设计一个适用于数字系统的界面</td>
<td>☑</td>
</tr>
<tr>
<td>ACTDIP019设计、修改和自动生成简单算法，涉及步骤序列、分支和迭代（重复）</td>
<td>☑</td>
</tr>
<tr>
<td>ACTDIP020实现简单的可视化程序，涉及分支、迭代（重复）和用户输入</td>
<td>☑</td>
</tr>
<tr>
<td>ACTDIP021解释学生解决方案和现有信息系统如何可持续并满足当前和未来本地社区需求</td>
<td>☑</td>
</tr>
<tr>
<td>ACTDIP022计划、创建和传达想法和信息，包括协作地应用已同意的道德、社会和技术在线协议</td>
<td>☑</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Content description</th>
<th>Years 7–8</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTDIP026分析和可视化数据，使用一系列软件来创建信息，并使用结构化数据来建模对象或事件</td>
<td>☑</td>
</tr>
<tr>
<td>ACTDIP027定义和分解现实世界问题，考虑功能要求和经济、环境、社会、技术和可用性约束</td>
<td>☑</td>
</tr>
<tr>
<td>ACTDIP028设计数字系统体验，生成和传达替代设计</td>
<td>☑</td>
</tr>
<tr>
<td>ACTDIP029设计算法表示图形化和用英文，然后跟踪算法来预测给定输入和识别错误</td>
<td>☑</td>
</tr>
<tr>
<td>ACTDIP030实现和修改具有用户界面的程序，涉及分支、迭代和函数在通用编程语言中</td>
<td>☑</td>
</tr>
<tr>
<td>ACTDIP031评估学生解决方案和现有信息系统是否符合需求，是否创新，考虑未来风险和可持续性</td>
<td>☑</td>
</tr>
<tr>
<td>ACTDIP032计划和管理项目，创建和传达想法和信息，包括协作地应用已同意的道德、社会和技术在线协议</td>
<td>☑</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Content description</th>
<th>Years 9–10</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTDIP037分析和可视化数据来创建信息，以及解决复杂问题，然后模型过程、实体和它们的关系使用结构化数据</td>
<td>☑</td>
</tr>
<tr>
<td>ACTDIP038定义和分解现实世界问题，考虑功能和非功能要求，并采访利益相关者来确定需求</td>
<td>☑</td>
</tr>
<tr>
<td>ACTDIP039设计数字系统体验，根据包括功能性的、可访问性的、可用性和美学的评估来设计替代方案</td>
<td>☑</td>
</tr>
<tr>
<td>ACTDIP040设计算法表示图形化并且在结构化英语中，验证算法和程序使用跟踪和测试用例</td>
<td>☑</td>
</tr>
<tr>
<td>ACTDIP041实现模块化程序，并应用选择算法和数据结构，包括使用面向对象的编程语言</td>
<td>☑</td>
</tr>
<tr>
<td>ACTDIP042批判性地评估学生解决方案和现有信息系统和政策考虑未来风险和可持续性，并提供为创新和商业机会提供机会</td>
<td>☑</td>
</tr>
<tr>
<td>ACTDIP043创建交互式解决方案，分享想法和信息在线，考虑安全、社会背景和法律责任</td>
<td>☑</td>
</tr>
<tr>
<td>ACTDIP044计划和管理项目，使用迭代和协作方法，识别风险并考虑安全和可持续性</td>
<td>☑</td>
</tr>
</tbody>
</table>

Key: ☑ Aligns with curriculum