Space Quest

A Beginner Coding Activity for Grades 1 and Up

Teacher Guide

Table of Contents

Quick Start

Space Quest Facts

Web Address:	**tynker.com/hour-of-code/space-quest**
Coding skill level:	**Beginner**
Recommended grade level:	**Grade 1+ (Reading skills required)**
Time Required:	**60 Minutes**
Number of modules:	**28**
Coding language:	**Block-based, Python, JavaScript**
Localization:	**English, Simplified Chinese, Spanish, German, French, Japanese**

Welcome!

If you're new to Tynker, check out our Hour of Code page to see all Tynker activities. Here, you'll find Space Quest and Dragon Blast, the first Tynker Hour of Code activities available in Spanish, Chinese, German, French, and Japanese as well as English. Note that the following activities include exciting concept videos that explain computer science concepts: *Candy Quest*, *Space Quest*, and *Dragon Dash*.

What is Space Quest?

Space Quest is a puzzle-based activity where students complete simple coding puzzles and learn computational thinking skills along the way. They play as an astronaut and trek across the surface of an alien planet.

Space Quest is a simplified version of a curriculum created by Tynker for the Everyone Can Code program by Apple. For more information about using the full course in your classroom, check out the iBook Get Started with Code 1 from Apple.

Who is this activity for?

Space Quest is intended for students in grades 1 and up with no coding experience. Students who are older and have more coding experience should check out Dragon Blast. Strong reading skills are also recommended.

What will my students learn?

The coding concepts covered in this activity are Sequencing, Loops, Debugging, and Conditionals. The puzzle sequence maps to CSTA Computer Science standards 1A-AP-09, 1A-AP-11, 1B-A-5-4, 1B-A-3-7, and 1B-A-6-8. For a complete list of standards, see the Standards Alignment section of this guide.

What devices do I need?

Each student needs to have a desktop computer, laptop computer, or Chromebook with an internet connection and an up-to-date browser. No downloads are required. If not enough devices are available, students can work in pairs on the same device.

How should I prepare to teach Space Quest?

The best way to prepare for an Hour of Code is to know the activity. Take half an hour to try Space Quest on your own beforehand. If you need help, feel free to consult the Activity Guide, which contains a full answer key and block glossary.

How can Tynker help me manage my Hour of Code?

Tynker has several free features for registered teachers that will help you manage your Hour of Code. If you set your students up with a Tynker classroom, you will be able to track their progress through Space Quest and print certificates for them to keep. For more information, see the Classroom Setup section of this guide.

How do I change my language settings?

Tynker will automatically detect your language preferences through your browser. If you are using a language that is supported by Space Quest, Tynker will use that language. A language selection feature will be implemented soon.

Activity Guide

This activity is designed for self-directed learning. The puzzles are as engaging as they are challenging. Aside from the optional Warm-up Activity, you will not have to lead your class through Space Quest. Your goal will be to help students out individually.

Tynker puzzles increase in complexity, and students may need help occasionally. The best way to help students proceed is to give them hints or clues that allow them to arrive at the answers themselves. If you give correct answers directly, your students won't learn as much as they otherwise could. If you need any help answering student questions, see the General Information section, which explains the code blocks and the Puzzle interface.

Before your Hour of Code, if you have extra time, get your students in the coding mindset with a Warm-up Activity. This is a slightly modified version of Simon Says.

Here is a sample schedule:

- 10 minutes: Warm up with Simon Says
- 50 minutes: Complete Space Quest

- If you have more time, choose from over 30 Tynker activities based on interest, grade and experience.

Learning Objectives

Your students will be using logical skills and computational thinking to manipulate code. Space Quest is organized into sections of 4-8 puzzles, corresponding to the following concepts:

Sequencing: A sequence is the order in which instructions are performed. By altering the sequence of code blocks, your students will learn that the order of events matters.

Loops: A loop is a set of code that gets repeated a certain number of times or until a certain condition is met. As they work with loops, your students will learn to analyze patterns.

Debugging: A bug is an error in a piece of code and debugging is the process of finding and fixing that error. As they spot bugs, your students will learn to critically examine code.

Conditionals: A condition is something that can be checked to see if it is true or false. As they use conditional statements, your students will learn to consider multiple scenarios.

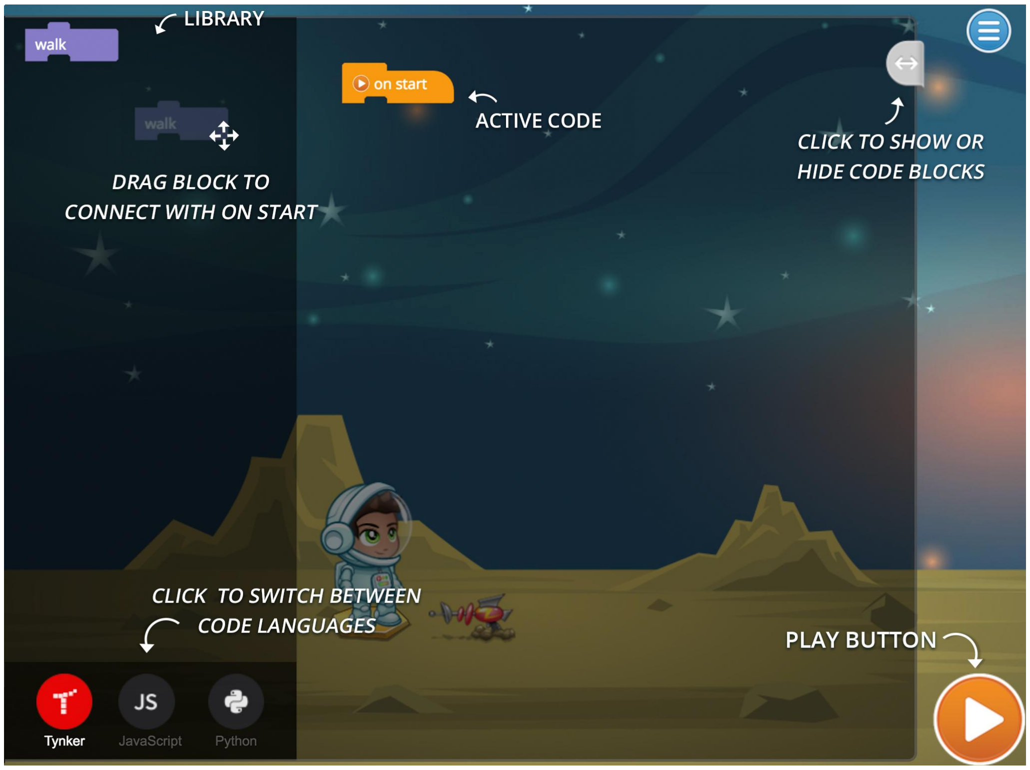
Click the links above to jump to the Answer Keys for each section.

How to Complete the Puzzles

Getting Started

To begin Space Quest, have your students open a browser tab to this URL:
tynker.com/hour-of-code/space-quest

The Workspace


LIBRARY
walk
walk
DRAG BLOCK TO CONNECT WITH ON START
on start
ACTIVE CODE
CLICK TO SHOW OR HIDE CODE BLOCKS
CLICK TO SWITCH BETWEEN CODE LANGUAGES
Tynker
JS
JavaScript
Python
PLAY BUTTON

In each puzzle, an overlay will appear containing the workspace. The darker section on the left is the library and shows which types of blocks are available for use for this particular puzzle. In this situation, the only block available is the “walk” block. The lighter section on the right is the active code area that your students will place their working code in. In this situation, there is already a “on start” code block on the active code area. To add a code block to the active code, simply drag the block from the library and connect it to a block in the active code area.

Visual Code Blocks, JavaScript, and Python

A code block is a block representing a piece of code that your students can use to make their program. The text in the code block has three modes that can be selected at the bottom left of the workspace: **Tynker visual blocks, JavaScript,** and **Python**. For younger students, we recommend using Tynker visual blocks, but students who want a challenge may want to use the other languages. Note that answers we provide use the Tynker blocks, but if you need a reference for JavaScript and Python, we have included the corresponding code in the Blocks Guide.

Running Your Code

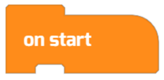
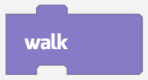
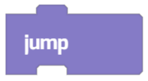
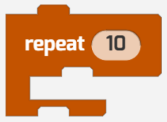
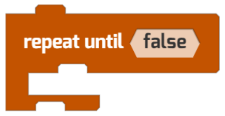
To run the code, your students should click on the play button on the bottom right of the screen. This will remove the workspace and the cadet should follow the code blocks that your students added to the active code area. While the cadet is moving, the code will be shown at the upper left. Your students can watch the code blocks execute by following either the green outlines. The green will move as each code block is executed.

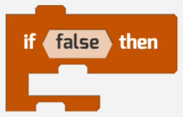
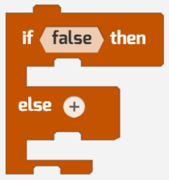
Common Issues

- Disconnected blocks
    - Make sure your students have connected all of their blocks and that the blocks are connected to the “on start” block. Code that is not connected to the “on start” block will not be executed.
- Using too few or too many blocks
    - Drag more blocks over if the cadet does not reach the end. If your students accidentally add too many blocks or a block that they do not want, then they can drag the block from the right to the left and a trash can symbol will show up. Once they release the block, the block will be removed from their code.
- Deleting the “on start” block
    - If students accidentally delete the “on start” block, they will need to restart the puzzle by clicking the button on the top right with the three lines and then selecting the refresh button to reset their code.
- Incorrect sequencing
    - For all of the puzzles, there are particular actions that have to happen at particular times. For some of the puzzles there is only one solution for the puzzle. For others there may be multiple ways to organize the code blocks to get the solution, especially when loops are involved. If your students are

struggling, suggest that they read through the code blocks one by one and trace what will happen to their cadet with their finger.

Block Guide

On Start  **JavaScript:** _on_start() **Python:** _on_start()	This is an event block that will run all code attached to it the play button is pressed. Students must attach their code to the bottom of the “on start” block for it to run.
Walk  **JavaScript:** walk(); **Python:** walk()	Moves the cadet one space forward.
Jump  **JavaScript:** jump(); **Python:** jump()	Makes the cadet jump over the obstacle and land on the opposite side of the obstacle.
Repeat  **JavaScript:** for (var i = 0; i < 10; i++){ } **Python:** for i in range(0, 10):	Everything inside of this block is repeated a certain number of times. The amount of times that the code repeats is the number that is next to the word repeat (the count). To change the number of repeats, click on the count and enter the number that you want. This is called a counting loop because it counts the number of times that the loop has been repeated and stops when it hits the inputted number.
Repeat Until  **JavaScript:** while (! _Med_Kit()){ } **Python:** while (not _Med_Kit()):	Everything inside of this block is repeated until a goal represented by the false is reached. If the goal is never reached, the code will be repeated forever. This is called a conditional loop because it relies on a condition being true to stop.

If  **JavaScript:** if (_Stacked_Aliens()){ } **Python:** if (_Stacked_Aliens()):	If the statement in the between the “if” and the “then” is true, then the code inside the block will execute. Otherwise, the code inside the block will be ignored. This block will perform the check every time it is executed.
Super Jump **JavaScript:** super_jump(); **Python:** super_jump()	Makes the cadet do a high jump over a really tall obstacle and land on the opposite side of the obstacle.
If-Else  **JavaScript:** if (_Stacked_Aliens()){ } else { } **Python:** if (_Stacked_Aliens()): else:	If the statement in the between the “if” and the “then” is true, then the code blocks in the indented section immediately below it will execute and the code blocks in the next indented section will be ignored. Otherwise, the code blocks in the indented section below the “else” statement will execute and the earlier code blocks will be ignored.

Standards Alignment

Space Quest is mapped to the following standards:

U.S. Standards

CSTA Computer Science:

- 1A-AP-09, 1A-AP-11, 1B-A-5-4, 1B-A-3-7, 1B-A-6-8

Computer Science for California-CS CA:

- K-2.AP.12, K-2.AP.13, K-2.AP.14, K-2.AP.16

Common Core CCSS-Math:

- 1.OA.1, 2.OA.1, 1.OA.2, 2.OA.2, 1.OA.3, 2.OA.3, 1.MD.4, MP.1, 5.G.1, 5.G.2, 6.NS.6

Common Core CCSS-ELA:

- 1.RI.3, 2.RI.3, 1.RI.6, 2.RI.6, 1.RI.7, 2.RI.7, 1.RI.10, 2.RI.10, 3.RI.3, 4.RI.3, 3.RI.5, 3.RI.7, 4.RI.7, 1.RF.1, 2.RF.1, 1.RF.4, 2.RF.4, 5.RF.4, 3.RF.3, 4.RF.3, 3.RF.4, 4.RF.4, 1.L.3, 2.L.3, 2.L.6, 6-8.RST.3, 6-8.RST.4, 6-8.RST.7, 3.W.3, 4.W.3, 3.W.4, 4.W.4, 3.W.6, 4.W.6, 3.L.1, 4.L.1, 3.L.2, 4.L.2, 3.L.3, 4.L.3, 3.L.4, 4.L.4

International Society for Technology in Education-ISTE:

- 1.1.c, 1.1.d, 1.4.d, 1.5.c, 1.5.d, 1.6.b, 1.7.c

U.K. Standards

National Curriculum in England (computing):

Key Stage 1

- Use logical reasoning to predict the behaviour of simple programs
- Use technology safely and respectfully, keeping personal information private; identify where to go for help and support when they have concerns about content or contact on the internet or other online technologies

Key Stage 2

- Design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts
- Use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs
- Understand computer networks, including the internet; how they can provide multiple services, such as the World Wide Web, and the opportunities they offer for communication and collaboration
- Use technology safely, respectfully and responsibly; recognise acceptable/unacceptable behaviour; identify a range of ways to report concerns about content and contact

Optional Warm-up Activity: Simon Says

If you have more than 50 minutes available, try Simon Says as an unplugged coding activity.

Simon Says is a classic game for kids in which one person gives simple instructions such as “touch your nose”, “reach for the sky”, or “jump in the air”. If you are unfamiliar with the game or need to review, here are the rules:

- Only follow instructions that come after the phrase “Simon Says”. Players are “out”, or eliminated, when they fail to follow an instruction preceded by “Simon Says”, or follow an instruction NOT preceded by “Simon Says.”
- The object of the game for the player acting as Simon is to get the other players out by giving tricky commands, and the object of the game for everyone else is to be the last one still in the game. If one player remains at the end of the game, that player wins; if the last few players are eliminated at once, Simon wins.

In your Simon Says game, give some specific instructions that will help demonstrate coding concepts.

- Sequencing: “Simon Says, step forward, then step back.”
- Loops: “Simon Says, pat your head three times.”
- Conditions: “Simon Says, If you are wearing green, touch your head.”

As you transition into playing Space Quest, explain to your students that this activity actually has a lot in common with Simon Says:

- They are Simon, and their astronaut is doing what they say-- but only if they say it just right!
- Leaving "on start" without a "walk" block is like saying “Simon Says” with no instructions. The astronaut will not know what to do!
- Placing a "walk" block by itself in the center is like saying “Touch your nose” instead of “Simon Says touch your nose”. The astronaut won’t follow those instructions.

As your students progress, remind them of the specific coding concepts they used in this warm-up activity.

Modules 1-9

Concepts: Sequencing

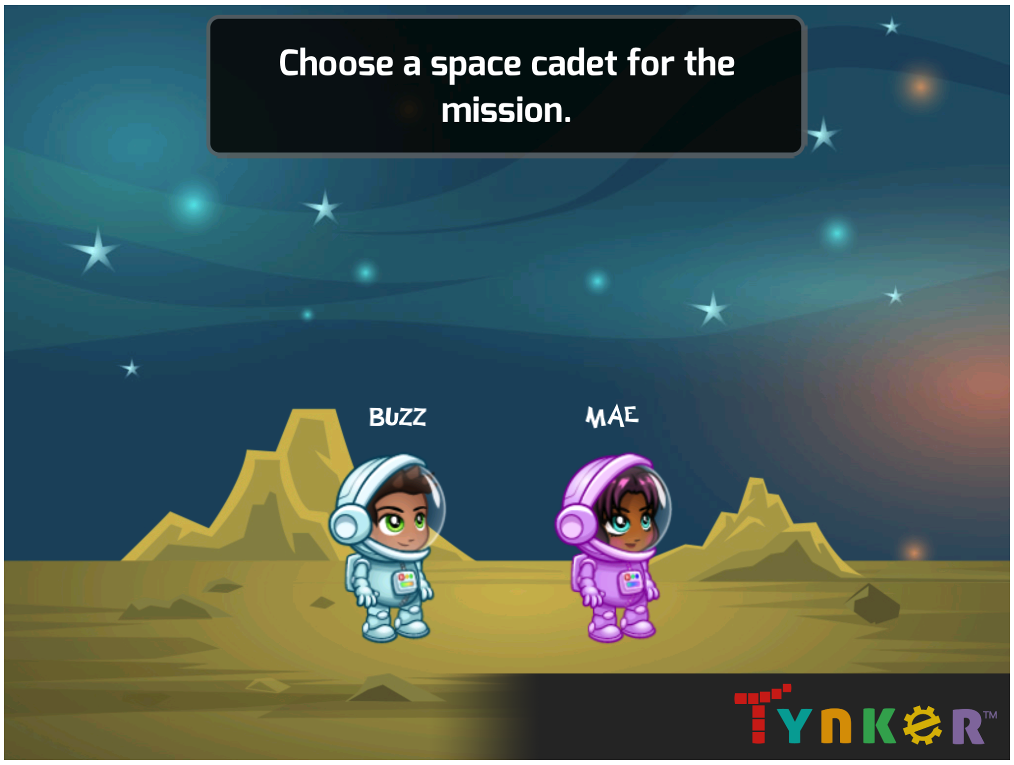
Overview

These puzzles are designed to introduce your students to the Tynker coding environment and to practice with basic coding blocks. Students will also be exposed to sequencing. A **sequence** is the order in which instructions are performed. Your students will need to use the code blocks to define specific sequence of instructions for the cadet to follow to reach the goal at the end of the puzzle.

Modules 8 and 9 are slightly different than the previous puzzles. Module 8 contains a short cutscene that your students will need to click through as well as an introduction to some code blocks that make the cadets dance. They will have the chance to test out some of the dancing blocks before moving on to module 9. Module 9 will give your students a chance to create a dance routine for the two cadets. They can link together as many or as few dance moves as they would like. When your students want to program Buzz, they need to select Buzz on the left side of the screen and use that workspace to program Buzz. When your students want to program Mae, they need to select Mae on the left side of the screen and use that workspace to program Mae.

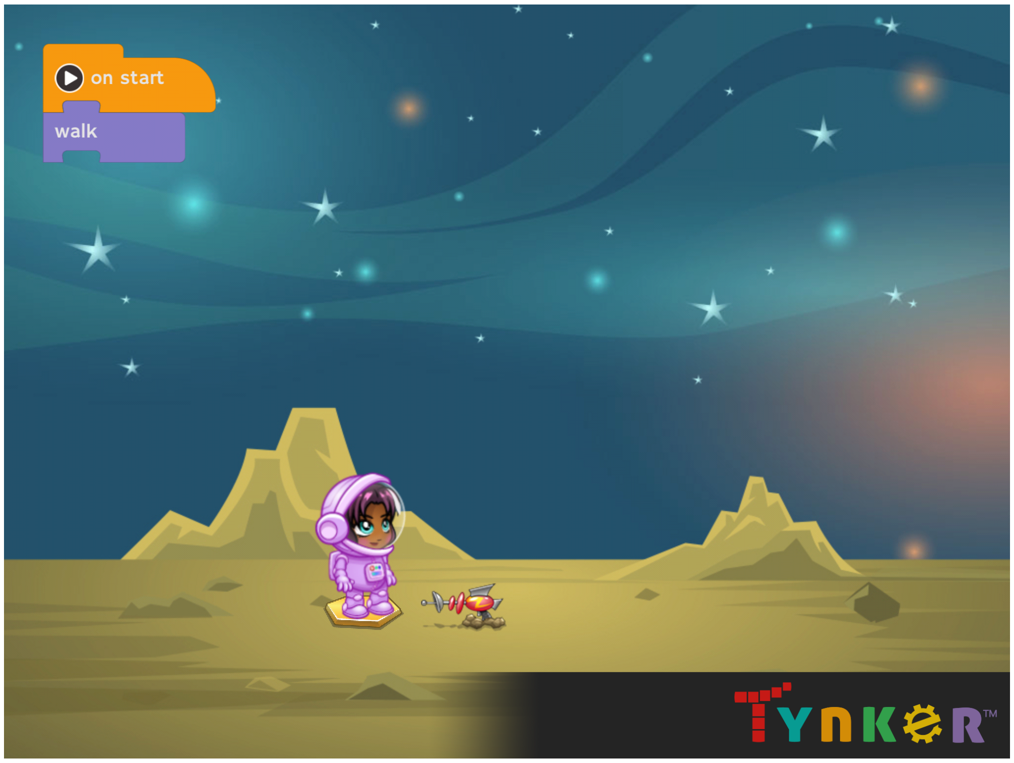
Module 1: Select a Space Cadet and Introduction

Your students must choose their character. They have the choice between two space cadets, Buzz and Mae. After that, an introduction sequence will play.


Choose a space cadet for the mission.

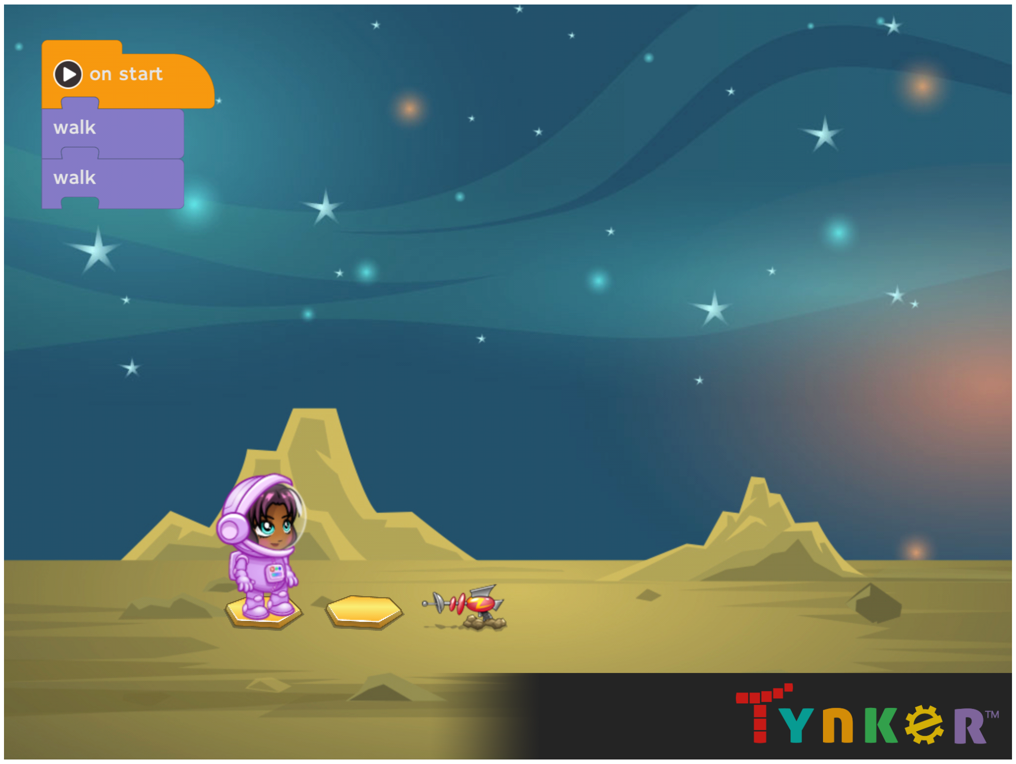
Module 3: Introduction

Move the cadet one space forward to collect the ray gun.



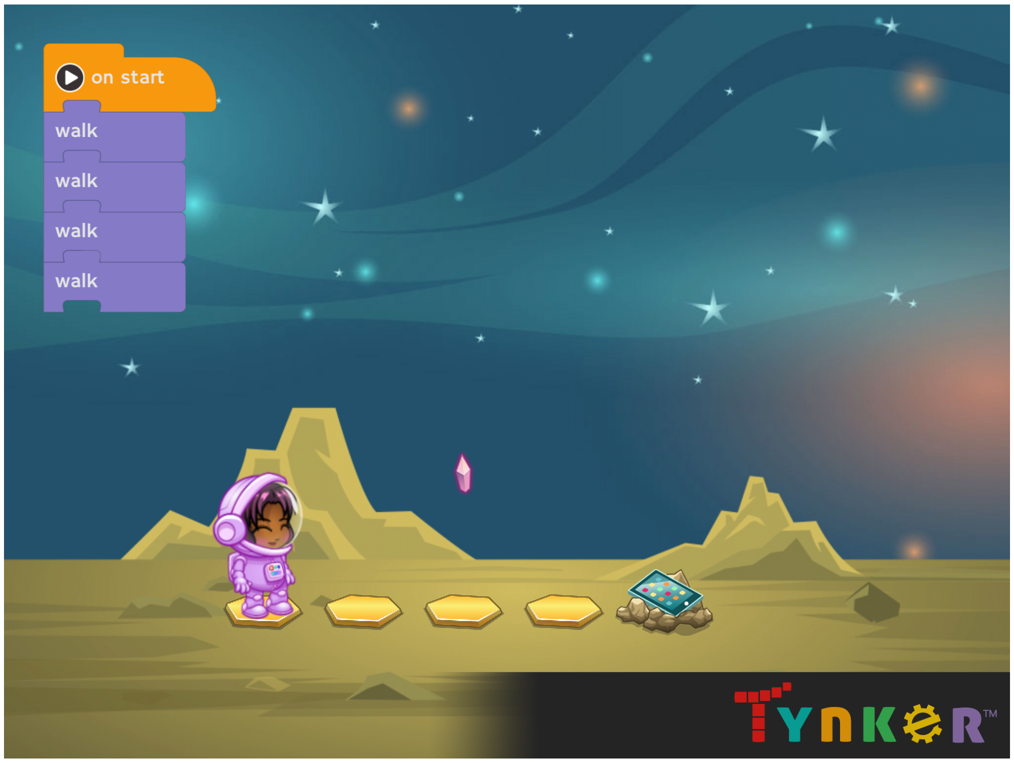
Module 4: Collect the Ray Gun

Move the cadet two spaces forward to collect the ray gun.



Module 5: Collect the Tablet

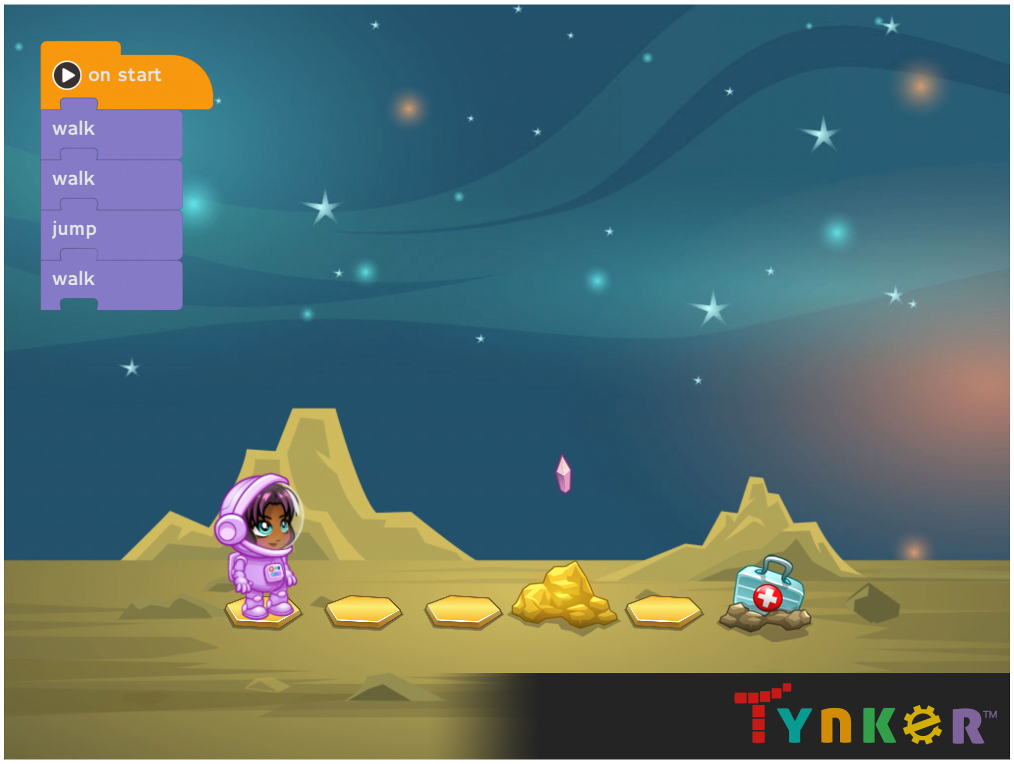
Move the cadet four spaces forward to collect the tablet. This puzzle is very similar to Collect the Ray Gun.



```
on start
walk
walk
walk
walk
```

Module 6: Jump the Rock

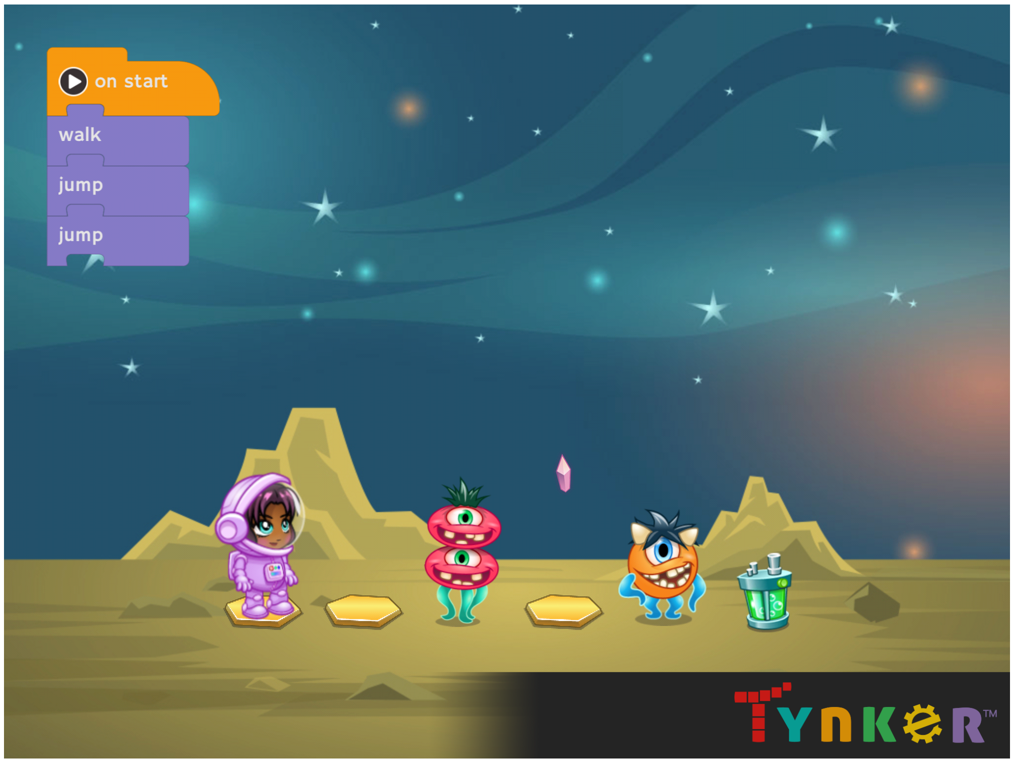
Avoid the rock by jumping and move the cadet to the med kit.



```
on start
walk
walk
jump
walk
```

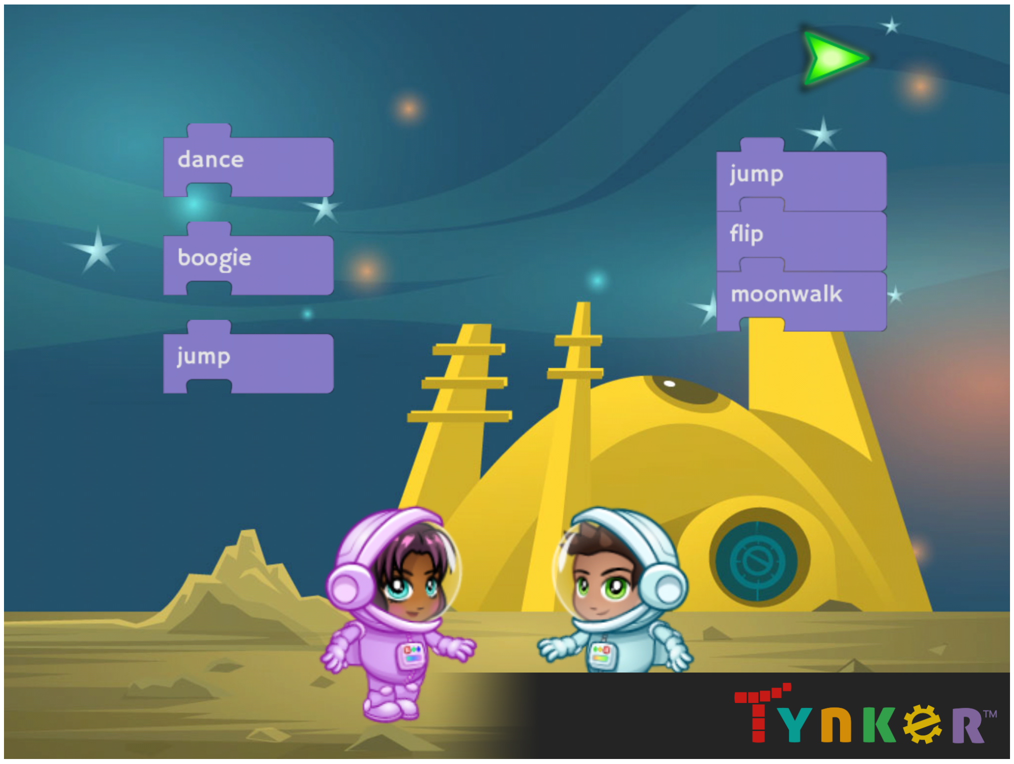
Module 7: Beware of Aliens

Avoid the aliens by jumping and move the cadet to the power cell.



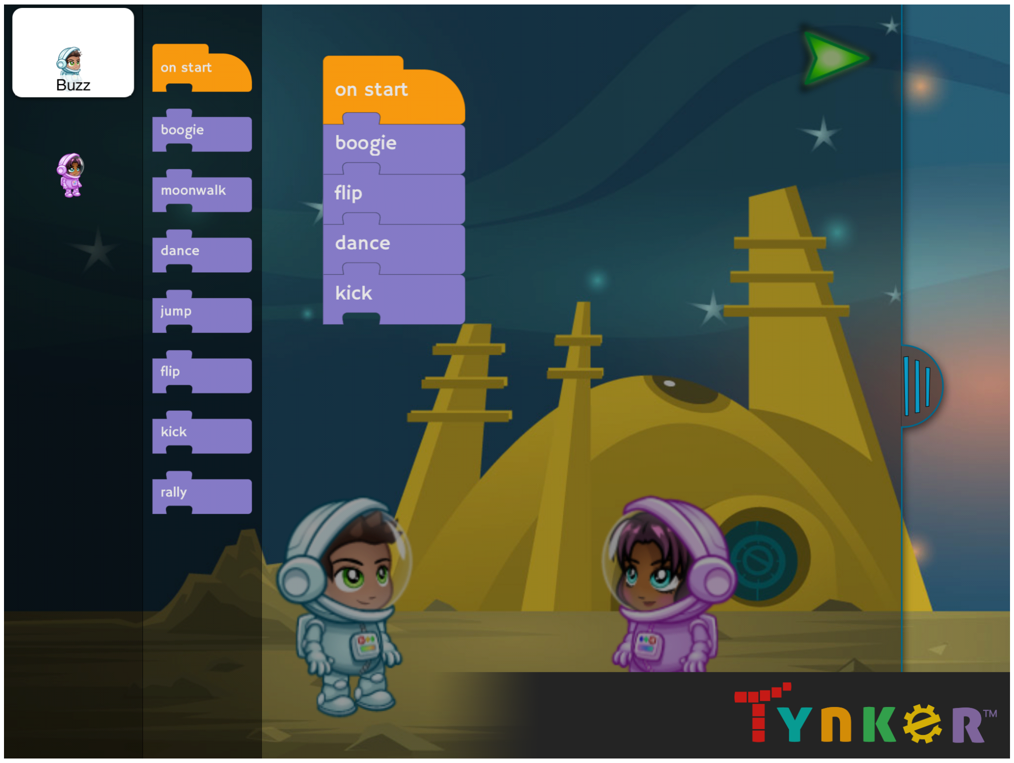
Module 8: Intro to Space Dance Party

Click through the introduction and click on the dance code blocks to see what each one does.



Module 9: Space Dance Party

Program Mae and Buzz to dance.



```
on start
boogie
flip
dance
kick
```

Modules 10-16

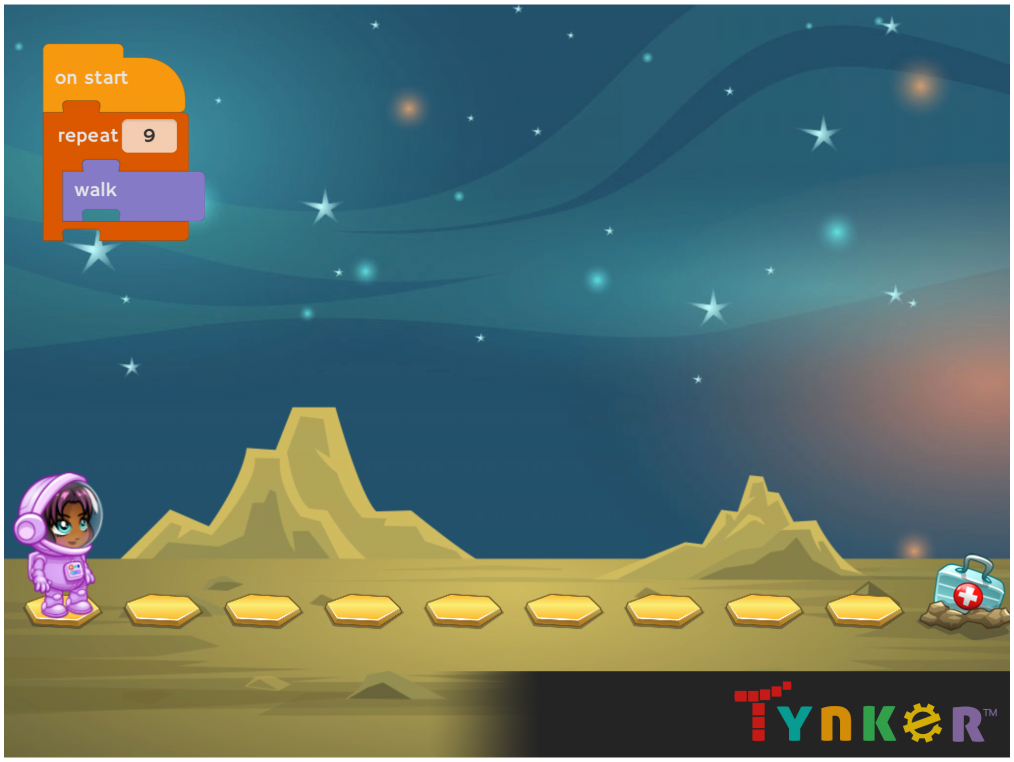
Concepts: Sequencing, Loops

Overview

These puzzles are designed to introduce your students to loops. A **loop** is a set of code that gets repeated a certain number of times or until a certain condition is met. Loops allow programmers to reduce the amount of code that they have to write. To complete the puzzles, your students will need to look for a repeating pattern that can be used to reach the end goal. That repeating pattern will be what they place inside the loop block.

Module 11: Use a Loop

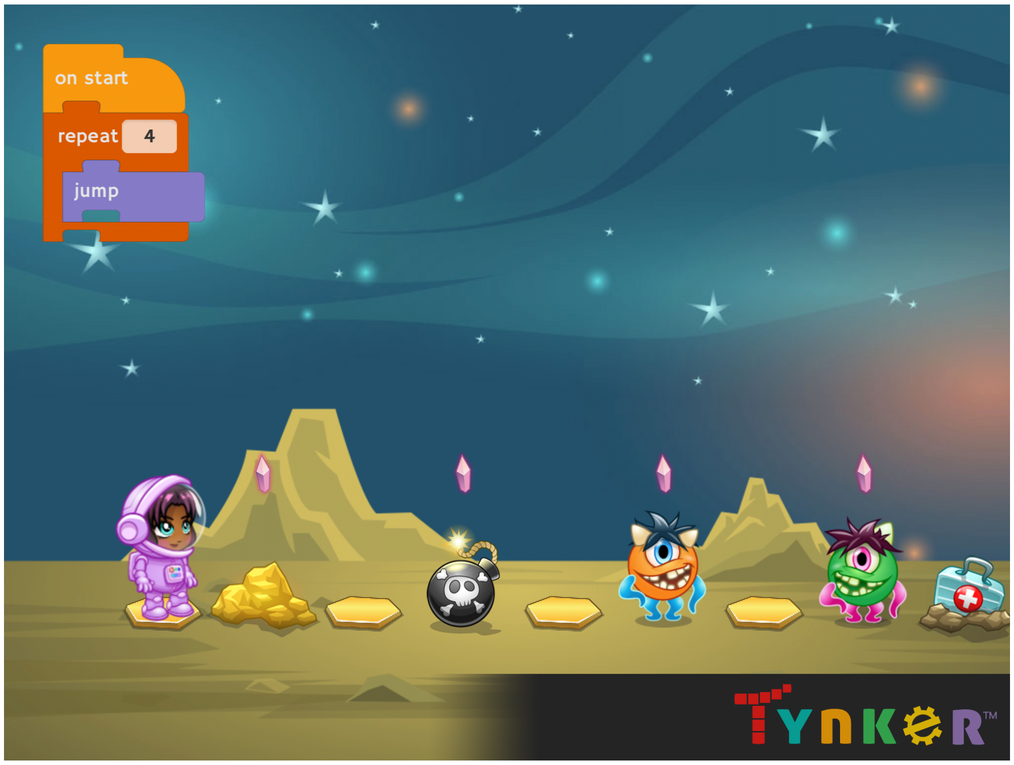
Use a "repeat" block to make the cadet walk to the med kit.



```
on start
repeat 9
    walk
```

Module 12: Jump Loop

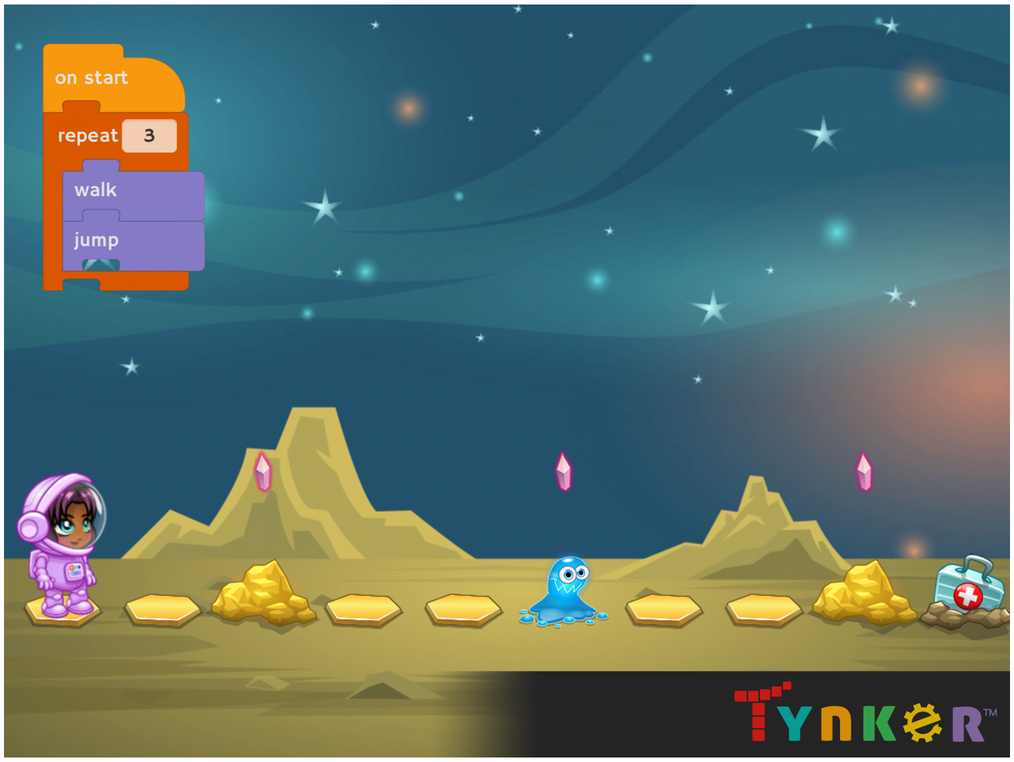
Use a “repeat” block to make the cadet jump over all of the obstacles.



```
on start
repeat 4
    jump
```

Module 13: Detect the Pattern 1

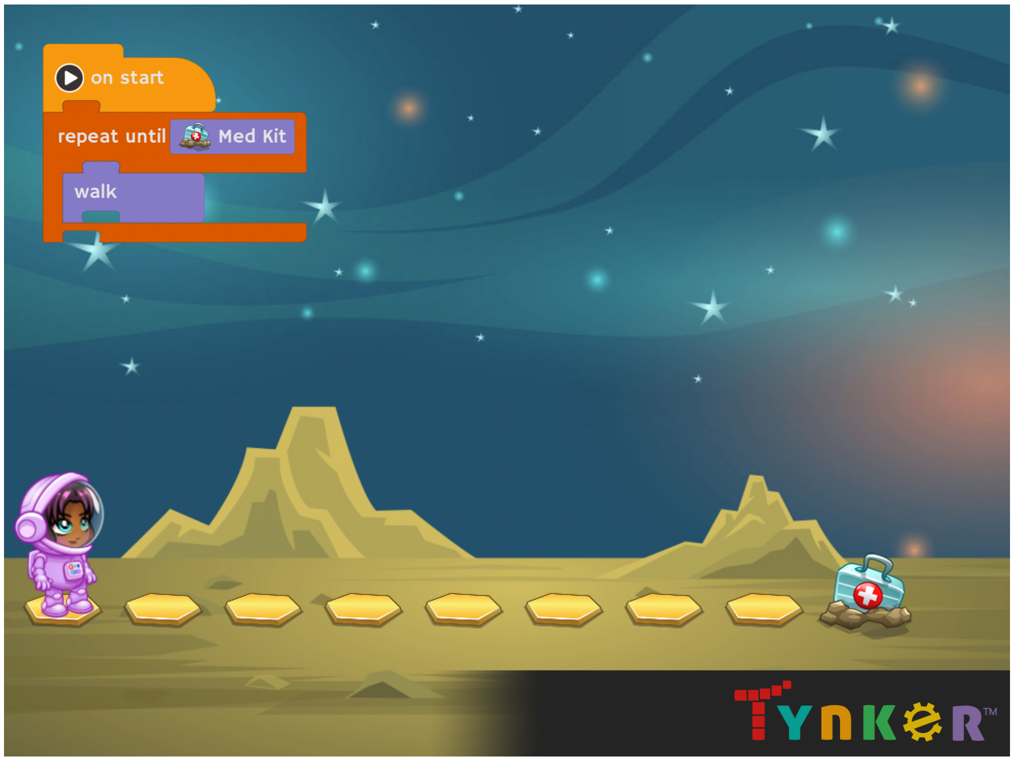
Use a combination of “repeat”, “walk”, and “jump” blocks to make the cadet avoid the obstacles and get to the med kit.



```
on start
repeat 3
    walk
    jump
```

Module 15: Another Loop

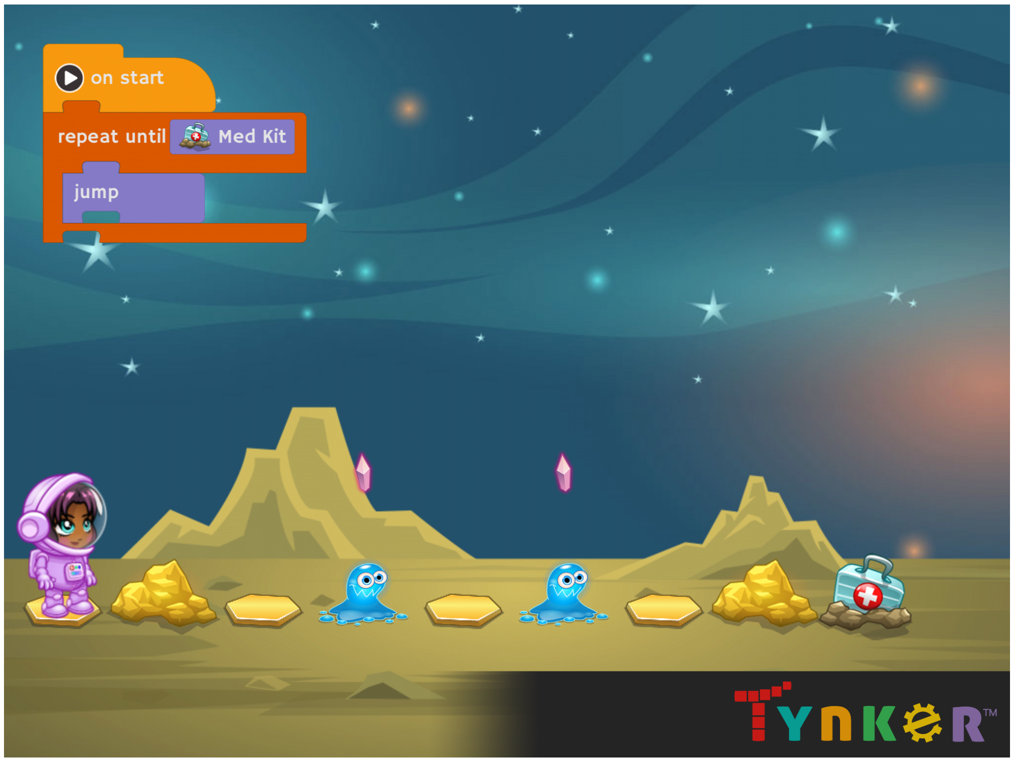
Use a “repeat until” block to make the cadet walk to the med kit.



```
on start
repeat until Med Kit
    walk
```

Module 16: Get the Med Kit

Use the “repeat until” block to make the cadet avoid the obstacles and get to the med kit.



```
on start
repeat until Med Kit
    jump
```

Modules 17-21

Concepts: Sequencing, Loops, Debugging

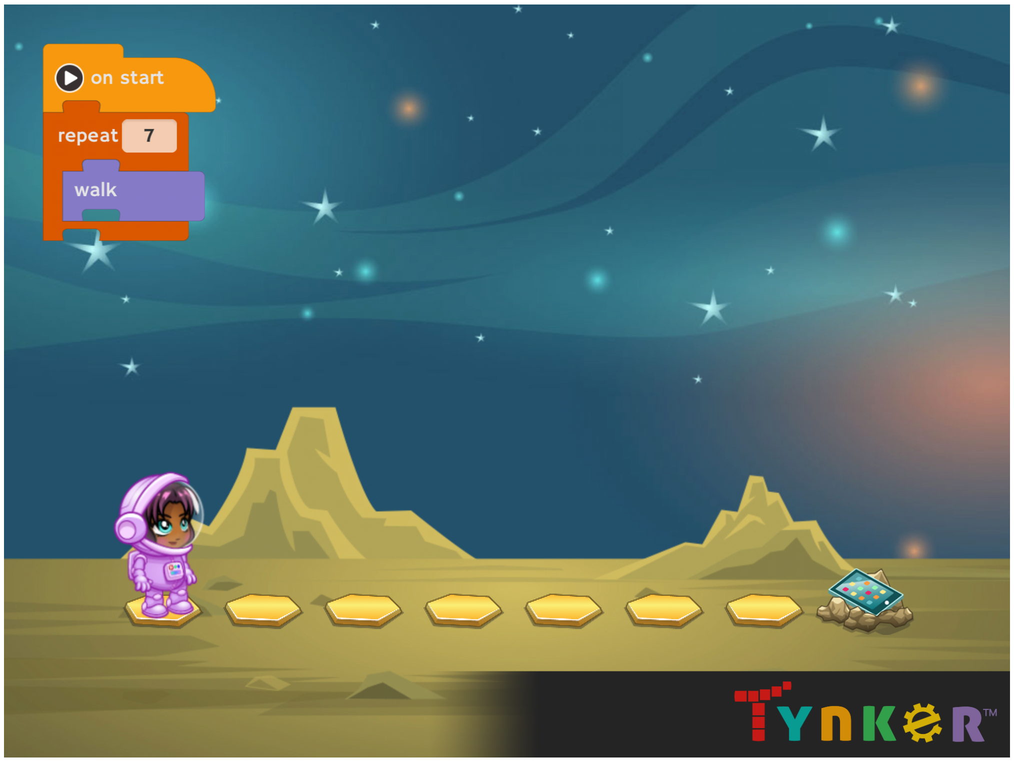
Overview

These puzzles are designed to introduce your students to debugging. A **bug** is an error in a piece of code and **debugging** is the process of finding and fixing that error. In each puzzle, your students will be given a puzzle and a piece of code that has a bug in it. They will need to fix the bug so that the cadet can do what they are supposed to do. There are a couple of ways your students can approach debugging:

- Run the buggy code to see what happens before modifying the code.
- Read through the buggy code and use a finger to trace what should be happening to the cadet as each code block is executed. They may also want to say the steps outloud to understand what is happening.
- Ignore the buggy code and write out by hand what they would do if they were approaching the puzzle as they did with previous puzzles. Then compare that code to the buggy code.

Module 17: Broken Path

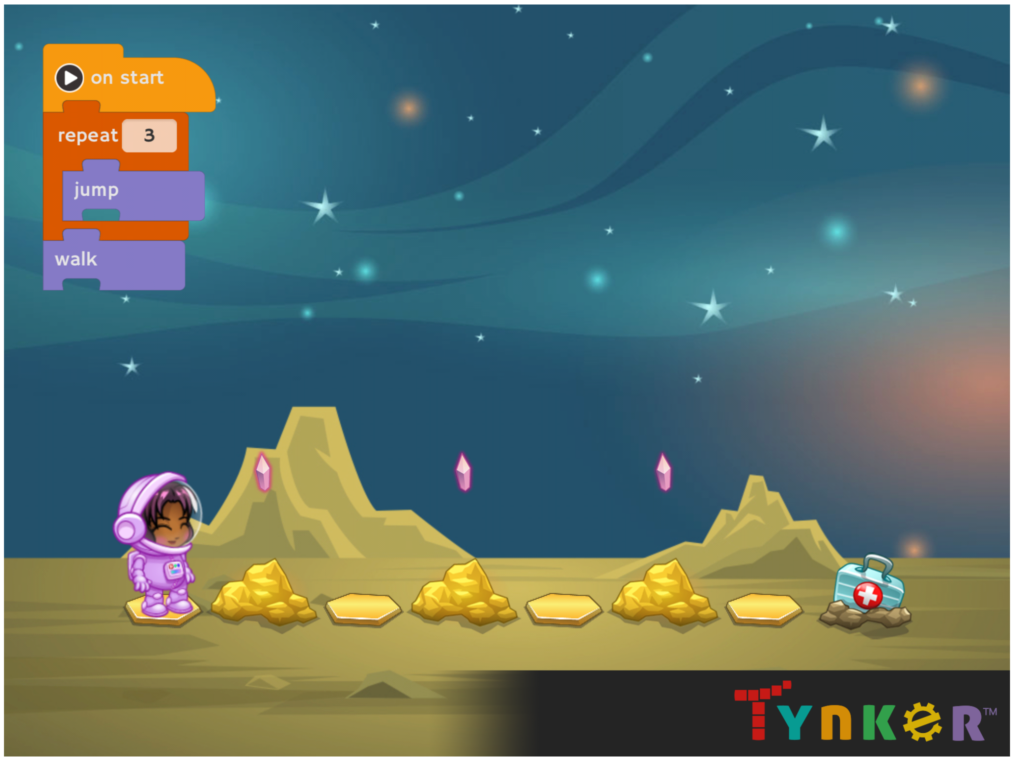
Debug the code so that the cadet can reach the tablet.
Debugging Steps:
Increase the count in the "repeat" block from 5 to 7.



```
on start
repeat 7
    walk
```

Module 18: Fix the Loop

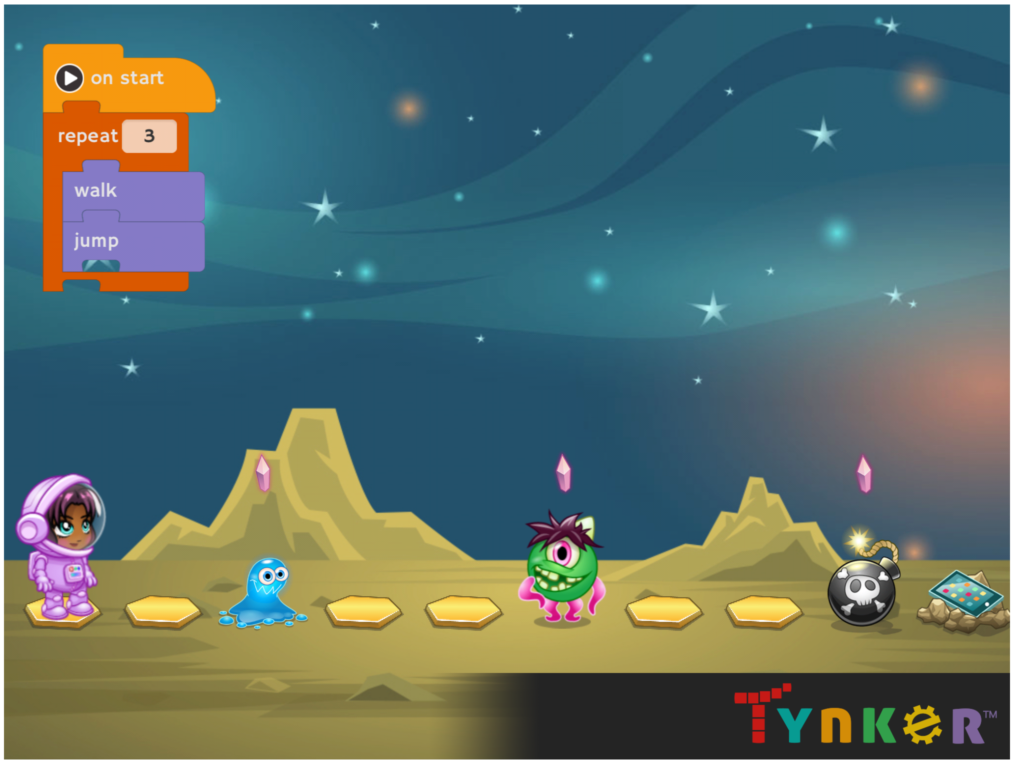
Debug the code so the cadet can avoid the obstacles and reach the med kit. Debugging Steps: Decrease the count in the “repeat” block from 4 to 3. Add a “walk” block after the “repeat” block.



```
on start
repeat 3
    jump
walk
```

Module 19: Ouch! Avoid the Obstacles

Debug the code so that the cadet can avoid the obstacles and reach the tablet. Answer: Increase the count in the “repeat” block from 2 to 3. Switch the “jump” and the “walk” blocks inside the “repeat” block by dragging the “walk” block up.

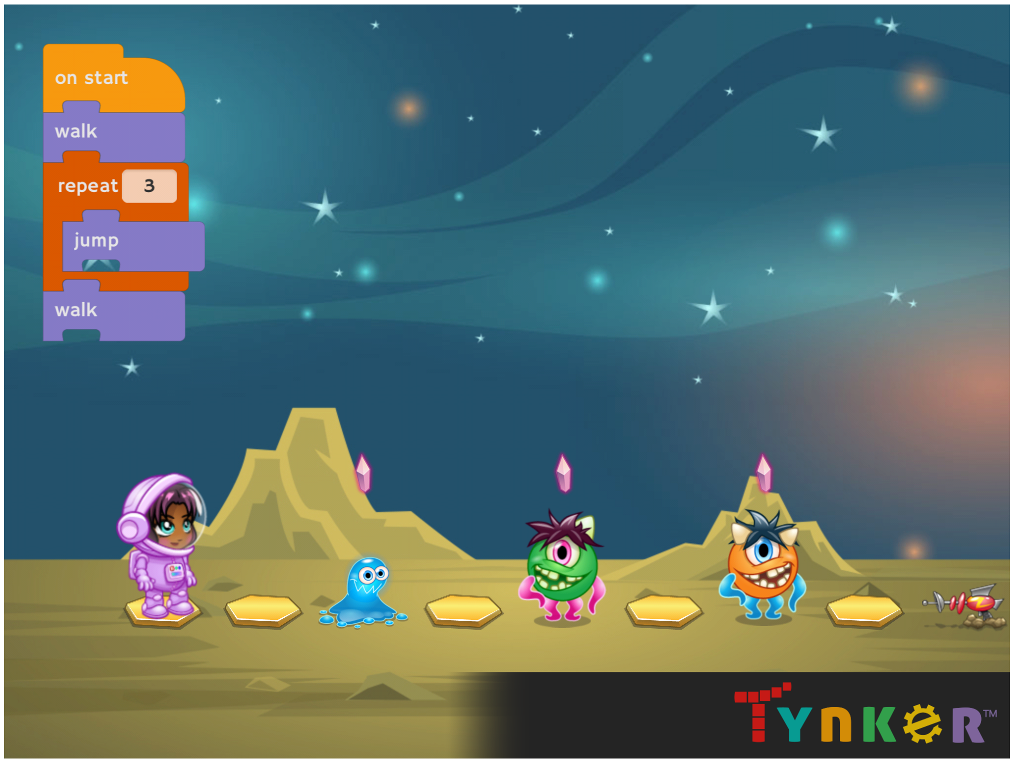


```
on start
repeat 3
    walk
    jump
```

Module 20: Broken Loop

Debug the code so that the cadet can avoid the aliens and reach the ray gun.

Debugging Steps: Increase the count in the “repeat” block from 2 to 3. Add a “walk” block before and after the “repeat” block.

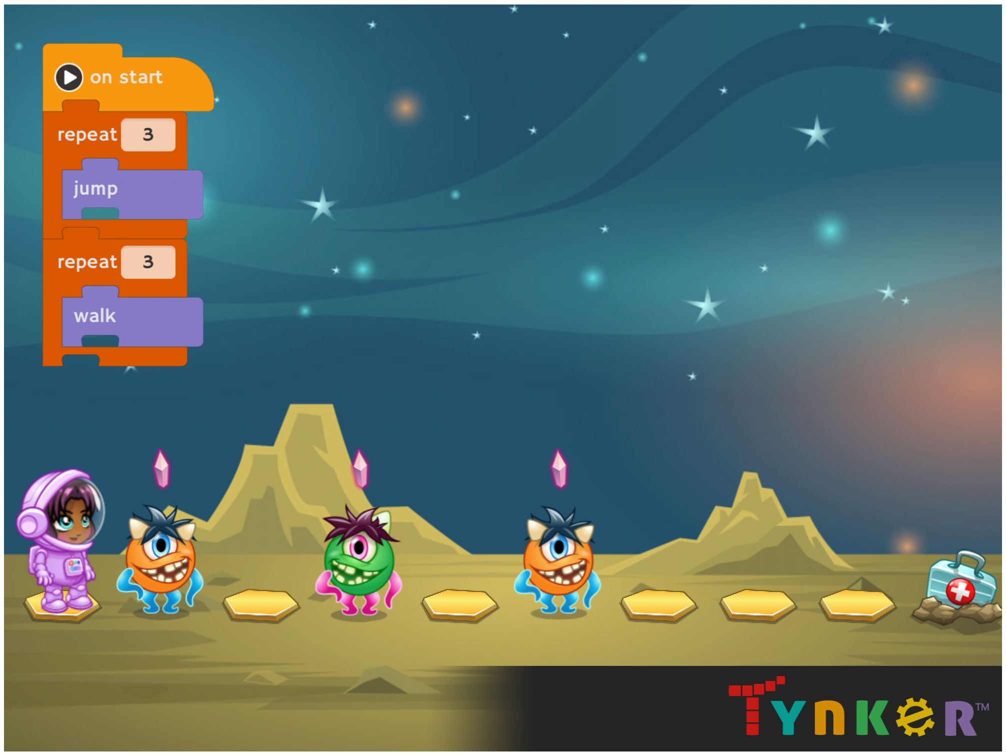


```
on start
walk
repeat 3
    jump
walk
```

Module 21: More Debugging

Debug the code so that the space cadet can reach the med kit while avoiding the aliens.

Debugging Steps: Increase the count in the “repeat” block from 2 to 3. Add a second “repeat” block with count 3 and containing a “walk” block.



```
on start
repeat 3
    jump
repeat 3
    walk
```

Modules 22-28

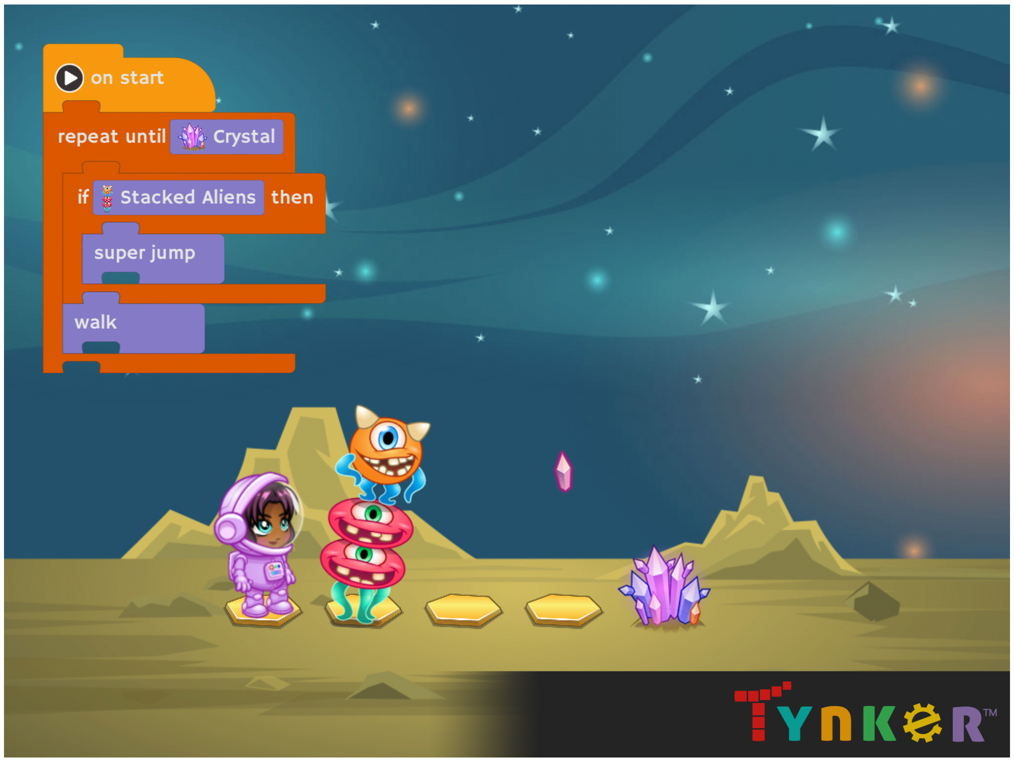
Concepts: Sequencing, Loops, Conditionals

Overview

These puzzles are designed to introduce your students to conditions and conditional statements. A **condition** is something that can be checked to see if it is true or false. For example, the computer can check if the cadet has reached the med kit. If the cadet is at the med kit, the condition is true, if not the condition is false. A **conditional statement** gives certain instructions when certain conditions are true. In these puzzles, your students will be using if statements. An **if statement** is a conditional statement that gives instruction only if the condition is true. For example, if you said the if statement "If you're happy, clap your hands", your students would only clap their hands if they are happy. Otherwise, the command "clap your hands" would be ignored. There are also if-else statements. **If-else statements** are the same as if statements but the have the extra "else" component which contains the instructions that are to be followed if the condition is not true.

Module 23: Shifty Stacked Aliens

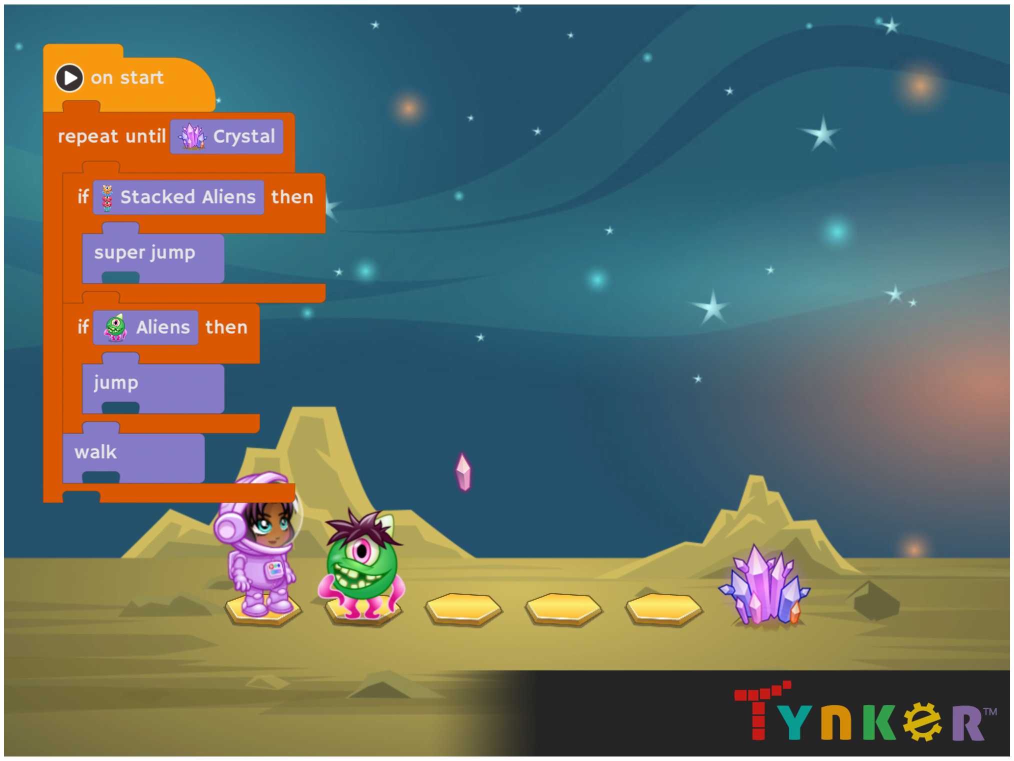
Use the "repeat until" and "if" blocks to move the cadet to the crystal.



```
on start
repeat until Crystal
    if Stacked Aliens then
        super jump
    walk
```

Module 24: Different Shifty Aliens

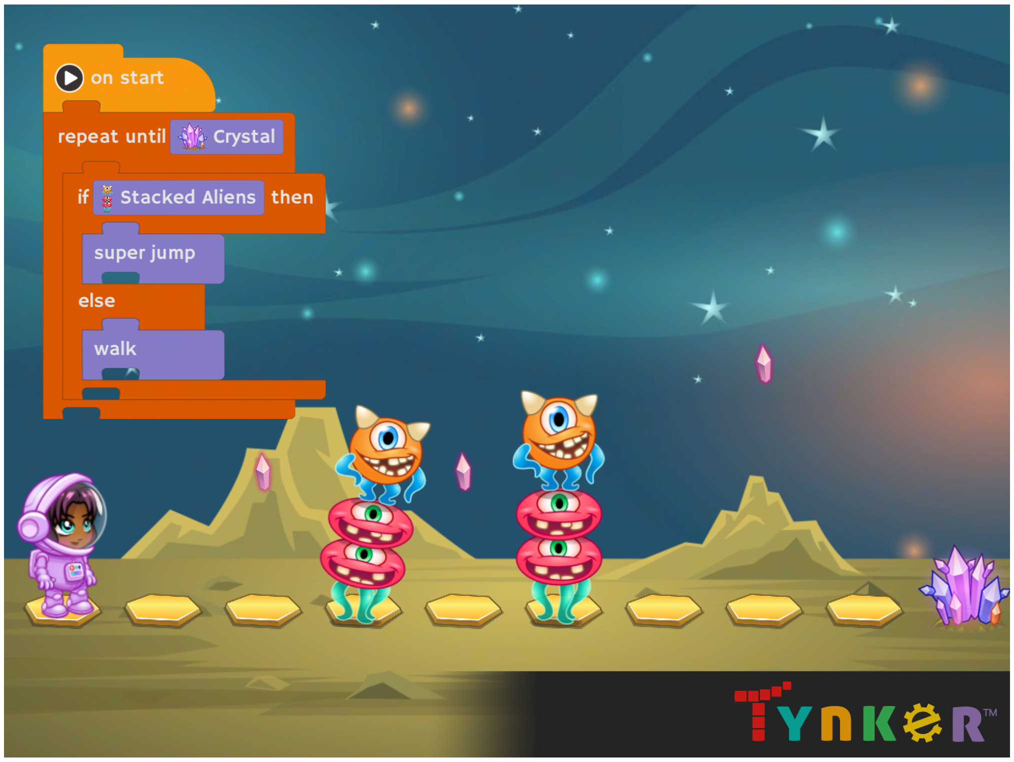
Avoid the moving aliens and move the cadet to the crystal. Use the conditional “if” blocks to check if there is either an alien or a set of stacked aliens in front of you before walking forward.



```
on start
repeat until Crystal
    if Stacked Aliens then
        super jump
    if Aliens then
        jump
    walk
```

Module 25: Loop with Shifty Aliens

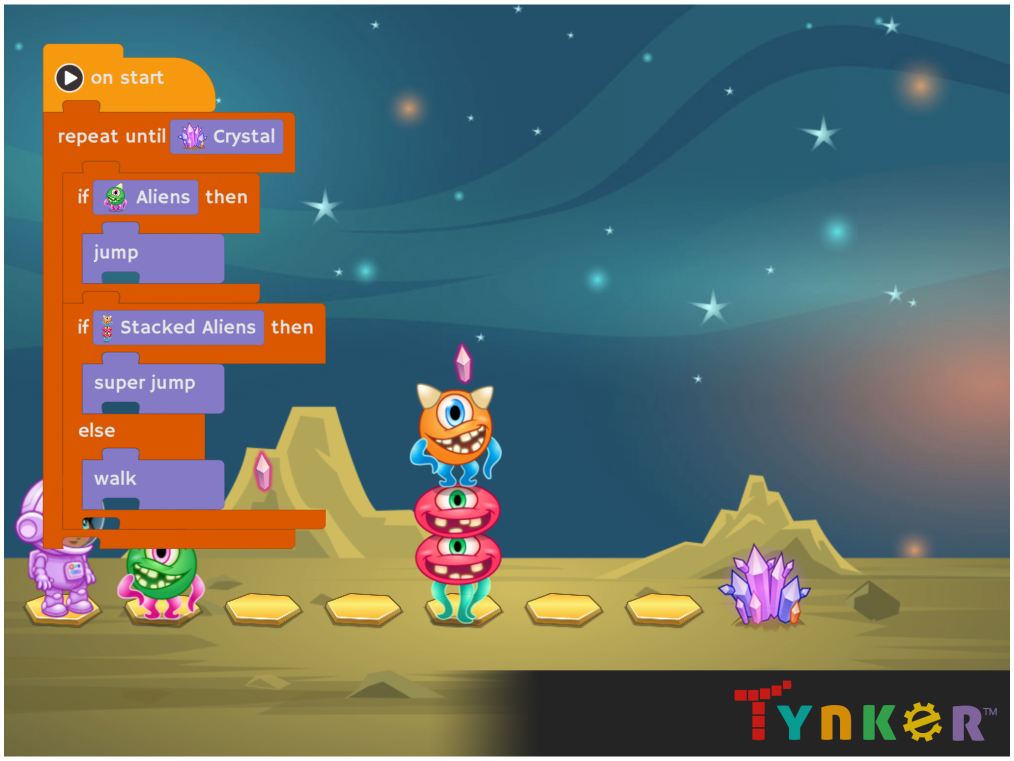
Use a “repeat-until” block to move the cadet to the crystal. Use the conditional “if” blocks to check if there is a set of stacked aliens in front of you or if you can just walk forward.



```
on start
repeat until Crystal
    if Stacked Aliens then
        super jump
    else
        walk
```

Module 26: To Jump or to Super Jump

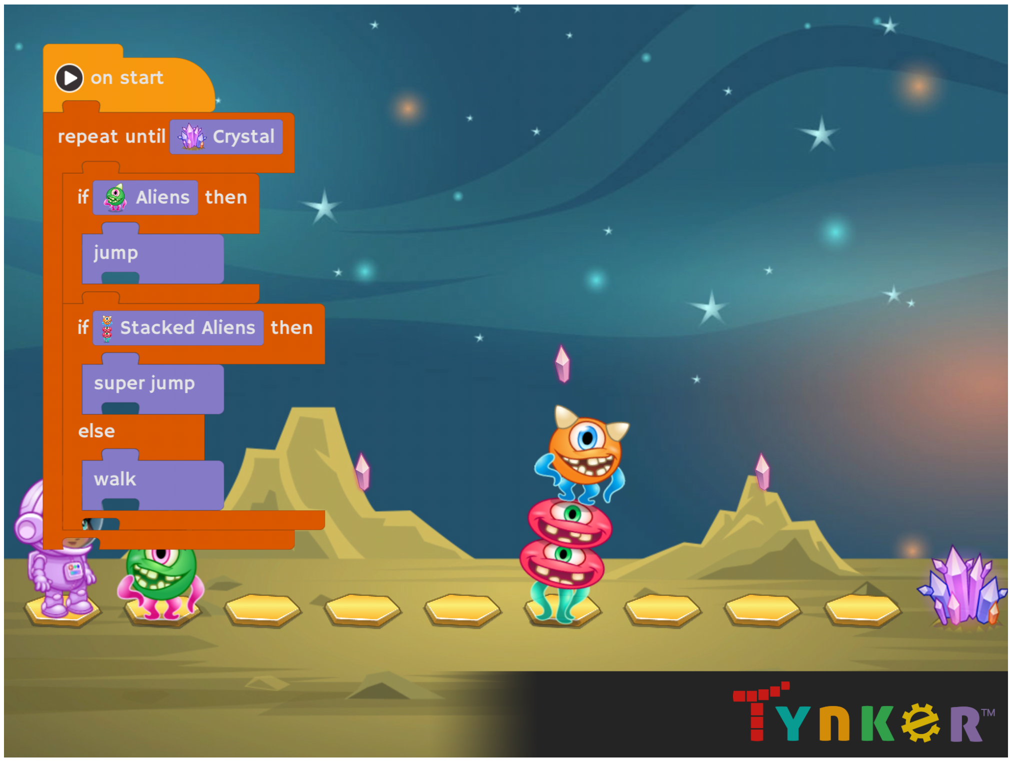
Use a “repeat-until” block to move the cadet to the crystal. Use the conditional “if” blocks to check if there is either an alien or a set of stacked aliens in front of you before walking forward.



```
on start
repeat until Crystal
    if Aliens then
        jump
    if Stacked Aliens then
        super jump
    else
        walk
```

Module 27: More Shifty Aliens

Use a “repeat-until” block to move the cadet to the crystal. Use the conditional “if” blocks to check if there is either an alien or a set of stacked aliens in front of you before walking forward. Note that the aliens can change from a single alien to a set of stacked aliens at any time.



```
on start
repeat until Crystal
    if Aliens then
        jump
    if Stacked Aliens then
        super jump
    else
        walk
```

After your students have completed all of the puzzles, there will be a short cutscene of the cadet blasting off.

Activity Wrap-Up

Once your students have seen the final cutscene, they will have completed their hour of code! Over the course of the 23 puzzles, your students will have explored the importance of sequences, improved their code by finding patterns and using loops, analyzed and tested code through debugging, and learned about how computers use conditionals to make decisions. These skills are fundamental to programming and your students will now have a good programming foundation with which to build on.

Other than their importance in programming, the concepts covered in this Hour of Code activity can be carried over into other subjects such as math, science, music, and literature. Scientists must perform the steps in an experiment in a particular sequence. Composers condense the length of sheet music by creating loops with phrases that indicate to the performer to repeat a specific part of music. Mathematicians correct their work by analyzing what their calculations should do and correcting errors just as programmers debug their code. Authors can create a large number of stories from a small set of conditional statements by writing adventure stories where different events happen depending on the choices the reader makes. Check out Tynker's Hour of Code STEM Activities page to get started using computer science throughout your curriculum.

We hope that your students enjoyed their programming with Tynker and will continue to develop their coding skills outside of Hour of Code. Read on for more information about how to continue your students' coding journey.

Classroom Setup

Click here to access Tynker's Quick Start guide for teachers. It only takes a few minutes to make a free Teacher account and a Tynker classroom for your students. If you are already set up with Google Classroom or Clever, you can use those services to automatically sync student accounts and classroom information with Tynker.

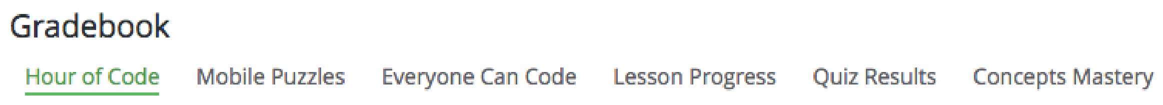
If you set your students up with a Tynker classroom, you will be able to:

- Track your students' progress through Space Quest
- Print certificates of completion for your students to keep
- Save students' progress to their accounts, so that they can continue coding at home
- View teacher guides and answer keys for all Tynker Hour of Code activities
- Access a free introductory coding course for your class
- Give your students access to all of Tynker's free content

Tracking Student Progress

Once you setup your students with a Tynker classroom, you'll be able to observe their progress in Space Quest using your Teacher Dashboard.

- Go to your Teacher Dashboard and select your classroom.
- Navigate to the "Gradebook" tab, then choose "Hour of Code."



- You will be able to see the amount of puzzles your students have completed for each Hour of Code activity, including Space Quest. Refresh the page to update it.

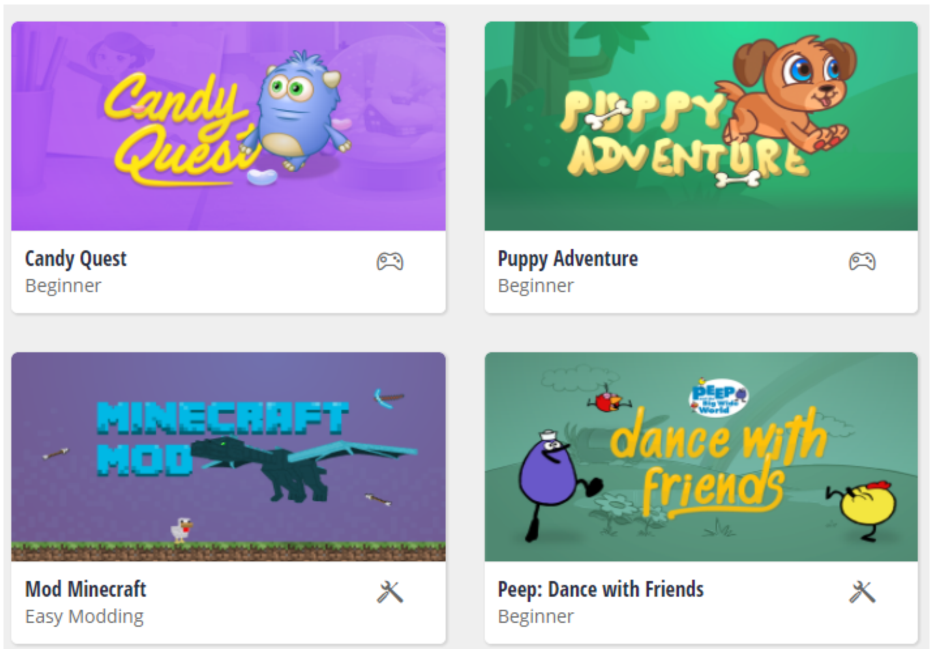
Student Certificates

Every time a student completes an Hour of Code coding activity in Tynker, they earn a badge that is added to their certificate. There are 29 badges that they can earn, so they'll be really excited to complete all the activities!

While signed in to a Teacher account, you can print certificates by clicking on a classroom from your Teacher dashboard, clicking the "Gradebook" tab, going to "Hour of Code", and clicking the "Print All Certificates" button. This will only print certificates for student accounts assigned to the selected classroom.

Other Hour of Code Activities

Tynker offers many other tutorials for the Hour of Code, including two other puzzle-based adventures for beginners: Candy Quest and Puppy Adventure. These activities are your best options if you are interested in having another Hour of Code with the same group of students. For teachers who are interested in the creative side of coding, we recommend Spin Draw and Peep: Dance with Friends. Check out the main Tynker Hour of Code page to see all the tutorials!



Going Beyond an Hour

Do More with Tynker

If your students enjoyed an Hour of Code, they're sure to enjoy the rest of what Tynker has to offer! Tynker offers a complete premium solution for schools to teach Computer Science. Over 300 hours of lessons are available to take K-8 students from block coding to advanced text coding. We offer tons of resources for teachers, including comprehensive guides, free webinars, and a forum to connect with other educators.

Learning Pathways

With Tynker, kids don't just acquire programming skills-- they can explore the world of possibilities that coding opens up. Tynker has several interest-driven learning paths that make coding fun, both inside and outside the classroom.

- **Coding and Game Design:** Your students can use Tynker Workshop, a powerful tool for crafting original programs to make games, stories, animations and other projects . They can even share their work with other kids in the Tynker Community.
- **Drones and Robotics:** Tynker integrates with connected toys, including Parrot drones and Lego WeDo robotics kits, so kids can see their code come to life.
- **Minecraft:** Tynker integrates with Minecraft so your students can learn coding through a game they love. Tynker offers skin and texture editing, as well as a custom Mod Workshop that lets kids try their original code in Minecraft.

Tynker for Schools

Used in over 60,000 schools, our award-winning platform has flexible plans to meet your classroom, school, or district needs. All solutions include:

- Grade-specific courses that teach visual coding, JavaScript, Python, robotics and drones
- A library of NGSS and Common Core compliant STEM courses that are great for project-based learning
- Automatic assessment and mastery charts for the school, class and student level
- Easy classroom management with Google Classroom and Clever integration
- Professional training, free webinars and other teacher training resources

About Tynker

Tynker is a creative platform designed to make coding fun to learn and easy to teach. Tynker's mission is to empower kids to become makers and equip them with computing skills for today's digital world. Over 60 million kids have begun their coding journey with Tynker!

About the Hour of Code

The Hour of Code is a global learning event in which schools and other organizations set aside an hour to teach coding. The event is held every December during Computer Science Education Week. You can also organize an Hour of Code year-round.

The goal of the Hour of Code is to expand access to computer science education for people of all backgrounds. Learning computer science helps students develop logic and creativity, and prepares them for the changing demands of the 21st century.

Tynker has been a leading provider of lessons for the Hour of Code since the event began in 2013. Since then, over 100 million students from 180 countries have finished an Hour of Code. For more information, visit the Hour of Code website.

If you have any issues or questions, just send us an email at support@tynker.com.

Happy coding!