Welcome to the Everyone Can Code Club!

These coding lessons will teach you basic coding concepts to go along with the app design lessons. With these lessons, you’ll not only build up your coding skills, but you’ll start to understand how apps work. This will help you design better apps.

In each lesson, you will learn about a specific coding concept with a brief introduction activity and then you’ll apply that coding concept to solve puzzles in Tynker.

There are also “Pick and Choose” optional activities for each lesson to help you learn more about that concept. You can decide whether you want to learn more about that concept or if you’re ready to move on to the next lesson.

Have fun!
Lessons

Think in Steps: Solving Problems with Algorithms 4
Think in Fixes: Debugging 6
Think in Circles: Looking for Loops 8
Think in Bits: Composition and Decomposition 10
Think in Sets: Abstraction 12
Think in Patterns: Forming Functions 14
Think in Specifics: Conditional Statements 16
Think in Cycles: While Loops and Nested Loops 18
Think In and Outside the Box: Variables, Input, and Output 20
Think in Practice: Design UI 22
Think in Steps: Solving Problems with Algorithms

Introduction (20 minutes)

Attention, it’s time for a game of Class Commander! Choose one person to stand at the front of the class and call out a set of commands for the rest to follow—for example, “Stand up,” and “touch your ear.” Take turns and be sure to include some silly commands too!

Were the instructions clear? Did everyone perform each command the same way? For example, which ear did they touch?

In computing, a command is an instruction that we give to a computer. Computers need clear, understandable instructions that need to be precise to get a specific result.

Now, imagine a simple Command Bot that follows only simple instructions in the order exactly as it’s been told: “Turn right, turn left, step forward, lift left arm.”

Pick one person to be the Command Bot. Give the Command Bot instructions to perform a specific task, such as fetching a book from the bookshelf or sitting in a chair.

Does the Command Bot succeed? The Command Bot’s instructions have to be clear, but they also have to be in the right order. The order of the commands is called the sequence.

You’ve been communicating an algorithm for Command Bot’s task.

Practice (25 minutes)

Complete all puzzles in Dragon Spells Lesson 1 in the Tynker app.

Think about it: How many steps did their longest algorithm have? How did they figure out the algorithms they needed to use to solve the puzzles?

Command: An instruction to perform a specific action.

Sequence: The order in which things happen, like patterns and events.

Algorithm: A step-by-step set of instructions to solve a problem or complete a task.
Think in Steps: Solving Problems with Algorithms
Pick and Choose

Make My Sandwich! (30 minutes)

Take turns as a super chef and create an algorithm to make your own signature sandwich, then they test your algorithms on each other.

1. Using the Keynote template provided, make a sandwich by dragging your choice of fillings into the sandwich template. Give your signature sandwich a name.

2. On a new slide, write an algorithm for your signature sandwich so that others can follow it to re-create the super chef signature sandwich.

3. Swap instructions, and using the Keynote template, see if you can make each other’s sandwich using the instructions provided.

4. Compare the sandwiches you’ve made following each other’s algorithms with the original signature sandwich.

5. Think about it: Did you successfully re-create the sandwich? Was anything missing from the instructions? Are they clear?
Think in Fixes: Debugging

Introduction (20 minutes)
Choose someone to be the Command Bot and another person to be the Director. The Director should give the Command Bot Tell instructions to walk in a square shape, but there’s a catch. The Director should decide on an error to include in the instructions, like turn right instead of left or take 10 steps instead of 5 so that the Command Bot is unable to complete a square.

Can the rest of the group spot the error? How should the error be corrected? Try it again with different shapes and students.

In coding, an error in instructions is called a bug. Finding and fixing the error is called debugging. As students learned in the last lesson with their algorithms for the Command Bot, their code has to be very accurate. If there’s a bug, the algorithm won’t work.

Practice (25 minutes)
In the Tynker app, complete all puzzles in Dragon Spells Lesson 2. For these puzzles, some code is given, but it’s buggy! First try the code provided and see what happens, then see if you can fix it to make it work correctly.

Think about it: How did the process of debugging improve their programs? How did they go about debugging the code?

Bug: An error in your code.
Debugging: The process of finding and fixing the error.
Pseudocode: An informal description of code or a concept that’s intended for human reading.
Think in Fixes: Debugging
Pick and Choose

Tunnel Bug (30 minutes)
Challenge each other to debug instructions for completing a video game level.

1. In small groups, create your own version of the Tunnel Bug video game and write a pseudocode algorithm for completing the level, thinking through the steps from start to final outcome.

2. Write each line of your pseudocode on a separate slide in Keynote. You should test out your algorithm to make sure it works.

3. Then, mix up the order of your slides and give your Keynote slides to another group who will use the instructions to play the level.

4. With the mixed-up Keynote slides, each group now needs to figure out how to debug the code so they have a correct set of player instructions.

5. Write down each step of the debugging process. For example, “Move slide 4 up to slide 2.”

6. Think about it: How many steps did it take? Could you have debugged the algorithm in fewer steps?
Think in Circles: Looking for Loops

Introduction (15 minutes)
Choose a tongue twister like, “She sells seashells by the seashore.” Everyone should try to perform the tongue twister very quickly, as many times as they can. Who can do the most repetitions? Now as a group, recite it 10 times. You’ve just performed a for loop.

Name the different ways you could describe what you’ve just done. You could write the tongue twister out 10 times. Or you could just say, “Repeat Shell Riff 10 times.”

In coding, we use for loops to give instructions to repeat a set of commands, because it’s more efficient than giving the same instruction 10 times.

Practice (30 minutes)
Complete all puzzles in Dragon Spells Lesson 3. Try to identify a repeating pattern, such as walking and jumping, then shorten their algorithm by putting that code in a loop.

Think about it: How did you use for loops in Tynker? How did you spot where they could use loops?
Think in Circles: Looking for Loops
Pick and Choose

Loopy Snake (30 minutes)
It’s time to get loopy! Identify loops in the patterns of a snake, then create your own patterns using for loops.

1. Download the Loopy Snake Keynote and, as a group, discuss how you would write instructions for someone else to create it.

2. Now create your own Loopy Snakes. In the Keynote template, you should first create a pattern of colors, then drag the colors onto the snake template, creating your own snake.

3. Next, you should give your pattern a name and write an algorithm for how to create your Loopy Snake.

4. Swap the slide with your algorithm with a partner to see if you can create each other’s snakes from the algorithms.

5. Think about it: Who had the simplest looking snake. What about most complex? Then compare the algorithms for those snakes. How different or similar are the algorithms?
Think in Bits: Composition and Decomposition

Introduction (15 minutes)
You’re now ready to get into the rhythm of coding. Pair up to compose a short body percussion routine. The rhythm can be expressed as a list of simple commands or actions—for example: tap, tap, tap, clap.

Choose a class conductor who will then point to each pair in random order and as many times as she wants. Each pair should perform their routine only when pointed to. By the end, the group should have performed a long, complex sound. That very complex sound result came from putting simple parts together.

Creating something more complex from simple parts is called composition. The opposite of composition is decomposition. Decomposition is breaking something down into smaller parts in order to solve a problem.

In coding, we put together smaller pieces of code to create a more complex program through composition. We use decomposition to break down a problem into smaller pieces that we can deal with one by one.

Practice (30 minutes)
In Tynker, complete all puzzles in Dragon Spells Lesson 4.

Think about it: How did you break down the puzzles? For example, first you might need to get past a barrier, then jump to the next level, then walk to the treasure.

Composition: Putting smaller parts of a program together to solve a larger problem.
Decomposition: Breaking a larger problem into smaller, more manageable pieces.
Think in Bits: Composition and Decomposition
Pick and Choose

**Cup Song** (30 minutes)
Learn to execute a complex maneuver by breaking it down into parts.

1. Watch this Cup Song video and try doing the routine just from watching the video.

2. Work in small groups to learn the routine by breaking it into smaller parts.

3. Download this movie and from your iPad Camera Roll, add into Notes or Pages.

4. As you watch the movie, divide the clip into smaller parts and label each section. Are the parts easier to learn? If not, can you cut them into even smaller parts?

5. In Notes, write the commands for each section of the Cup Song. This will help you form an algorithm for performing the routine. You’re using decomposition to create an algorithm for solving a problem.

6. Now perform the Cup Song for the group and explain your process of breaking down the routine into smaller parts. Did everyone get the same algorithm as the original?

7. Think about it: Did groups break down the routine in the same way? Was there a process that seemed better?
Think in Sets: Abstraction

Introduction (15 minutes)
Think of a few simple rules for categorizing things—for example, by shape, color, or purpose. Choose someone to be the picnic commander who is going on a picky picnic. The picnic commander decides on a rule for the items to bring on the picnic, like only desserts, or things that are round.

The group then suggests different things to bring on the picnic and the picnic commander responds yes or no to whether that item can go in the picnic basket.

The first student to guess the underlying common feature becomes the commander for the next round. Play a couple of rounds with different rules.

How did you guess the rule? By figuring out what those things had in common.

This is called abstraction—we’re abstracting from what’s different about the things, and focusing on what they have in common.

Practice (30 minutes)
Complete the puzzles and open-ended projects in Dragon Spells Lesson 5.

Think about it: How did they use abstraction in the app?
**Think in Sets: Abstraction**

**Pick and Choose**

---

**Silly Sets** *(30 minutes)*

Classify objects into sets and explain your rationale using abstraction.

1. Download this [Keynote template](#) that has a group of seemingly random items that seem silly—for example, a guitar and a pair of binoculars.

2. Organize the objects on the template by grouping them into sets in any way you want. You don’t have to include all the objects in their sets, and you can use one object in several sets, but you should have at least three objects in each of your two sets.

3. In the Keynote template, provide a descriptive title for each of your sets so that it’s clear why those items are grouped together.

4. Think about it: Take a look at each other’s work. Are there items in any sets that don’t belong? Are there items that can be added to the set? What about items that are not clear as to whether they should belong in a set?

---

All these things you can eat.

All these things are transportation.
Think in Patterns: Forming Functions

Introduction (15 minutes)
Think of a person or thing in the room. In pairs, take turns describing that person or thing to each other. The catch is, you can’t use the name of the person or thing. How many characteristics did you describe before your partner figured it out? Wouldn’t it have been much easier if you could just say the name of the person or thing?

Think back to Lesson 1. Remember the class commander and Command Bot?

Sometimes we had to give the Command Bot a lot of instructions to do a task. We can make it easier by giving the set of instructions for each task a name. If we give the Command Bot a name for those instructions, like “Get book” or “Do cartwheel,” then any time we want the Command Bot to perform the task, we can just use the name. Now we have a smart way of giving the Command Bot instructions.

When we code, we create functions—a set of commands that has been given a name. Just like with the Command Bot, it makes coding easier and more efficient. We can just call that function by its name whenever we need it.

Practice (30 minutes)
In Tynker, complete all puzzles in Dragon Spells Lesson 6.

Think about it: What tasks, like eating fireflies and blasting through obstacles, can you package in a function to make them easy to reuse.
Think in Patterns: Forming Functions
Pick and Choose

Command Bot for the Win (30 minutes)

Using functions, create a routine for the Command Bot.

1. Work in small groups. Each group is entering the Command Bot in a competition of your choice. The only rule is that it should involve a routine, such as gymnastics, dancing, boxing, cooking, or even a Cup Song competition.

2. Using a presentation app like Keynote or Explain Everything, collaborate to create one routine made up of at least three different moves that you’ll use to program the Command Bot.

3. Then, take photos of each step of the routine and upload them to your presentation. In the presentation, provide the instructions to perform your routine.

4. Using your composition and decomposition skills, create and label functions in your presentation.

5. Once you’ve finished their presentation, swap it with another group and perform each other’s routine. Did they perform the routine correctly? Would they win the gold medal?

6. Think about it: Did you give their routines a name? Is that a function?
Think in Specifics: Conditional Statements

Introduction (15 minutes)

Sing the song, “If You’re Happy and You Know It.” Sing a few rounds, changing the lyrics to something like, “if you’re surprised”

You’ve been using if statements. An if statement tells us what to do if something is true. A condition is something we can check to tell if the statement is true or false. So here, being happy is a condition. If “you’re happy” is true, then clap your hands. “If you’re happy, then clap your hands” is a conditional statement.

What do we do if we’re not happy? With code, each command needs to be very precise. With conditional statements, if the condition isn’t true, you just carry on:

If you’re not happy, then don’t clap your hands.

But we can also decide to be more precise by using an if-else statement:

If you’re happy, clap your hands, else make a sad face

In coding, conditional statements like if statements and if-else statements give instructions for what to do in particular situations.

Practice (40 minutes)

Complete all puzzles in Dragon Spells Lesson 7.

Think about it: What did if statements enable them to do in their coding?
Think in Specifics: Conditional Statements
Pick and Choose

Virtual Travel Adventure (30 minutes)
Games they play are built around IF statements.

1. Work in pairs and plan your ideal travel destination. To help you figure out where to go, come up with at least five conditions that their destination needs to meet—for example, that the destination “is warm” or “is near the seaside.” Brainstorm your list of conditions in Notes.

2. Think of different destinations, such as Paris or New York, and find out if each destination fits your conditions. Use Safari to research the destinations or even ask Siri questions like, “What’s the weather like today in the Grand Canyon?”

3. Test your destination by checking each condition and labeling them TRUE or FALSE. Does the destination meet all five conditions? If the destination passes the test, they’re ready to continue the virtual adventure.

4. Open Maps and search for your tested and approved destination using the search bar, and take a screenshot of the map.

5. Create their own virtual landmark postcard using a Photo Card template in Pages. Upload your screenshots from Maps to the template and write to your family or friends, using if-else statements to tell them how you decided where to go for their virtual adventure. Don’t forget the “else” part of the statement!

6. Think about it: Were your conditions always clear?
Think in Cycles: While Loops and Nested Loops

Introduction (15 minutes)
The Command Bot now wants to enter another competition, like a gymnastics floor routine to music. Here are the first few moves:

Walk onto the mat.
If the audience applauds, bow. Twirl 3 times.
While the jazzy introduction plays, do jumps.
Then perform a double twirl 3 times.
If the applause is loud, repeat the double twirl.
Do a back flip.

Can you spot how to code the routine using the coding tools we've learned so far? Most of the routine can be coded using if statements, for loops, and functions. But what about the jumps? The for loop can’t help us here, because we don’t know how many jumps there will be. Instead we use a while loop, the instruction to just keep doing jumps, as long as the jazzy introduction is playing. A while loop tells us to repeat a set of commands while a condition is true. When the condition is no longer true, we stop.

In the line, “Then perform a double twirl 3 times,” there is a for loop to do a double twirl 3 times. But a double twirl is a loop also by itself. So there is a loop inside a loop, which is called a nested loop. This is one way we can combine the concepts we’ve learned so far to create complex programs in a very efficient way.

Practice (30 minutes)
Complete all puzzles in Dragon Spells Lesson 8.
Think about it: When and how did you use different types of loops?

While loop: Tells us to repeat a set of commands while a condition is true. When the condition is no longer true, we stop.

Nested loop: A loop inside another loop.
Think in Cycles: While Loops and Nested Loops
Pick and Choose

Donut Delight (30 minutes)
Use while loops and nested loops to organize running your own donut stand.

1. You’ve been tasked with running a donut stand and you’ll be serving hundreds of customers. Luckily, you have Donut Bot to help you.

2. To start the day, the Donut Bot’s job is to ice donuts. Using a presentation app like Keynote or Explain Everything, work in groups and create a virtual store that includes pseudocode for the steps needed to ice four packs of donuts. Each pack of donuts has five donuts.

3. The Donut Bot has succeeded in icing four packs, each with five donuts. Twenty donuts are ready! But more customers still wait. How can you revise our pseudocode so that the Donut Bot keeps icing as long as there are customers?

4. Identify what coding concepts they’ve used in their pseudocode. Where did they use functions, algorithms, or for, while, and nested loops?

5. Think about it: Did all groups came up with the same pseudocode. Which group had the most efficient pseudocode? Most interesting?
Think In and Outside the Box: Variables, Input, and Output

Introduction (15 minutes)

Play a rapid-fire game, like multiplication tables. Each time the group answers correctly, they score a point, but you have to answer within five seconds. Tally the score on the board, and stop periodically to check the score with the class. The score changes as you play.

The score is a variable. The input is each new point you add, and the output is the final number you score.

Variables hold values like numbers, words, and other types of data. You use variables when the values in a program are changing. They’re useful because coders can write the code without knowing exactly what number or word will be used.

Practice (30 minutes)

Complete the puzzles and projects in Dragon Spells Lesson 9.

Think about it: How did you use variables in the app? What kinds of input and output did you notice?

Variable: A container that stores a value. The value can change over time, and the container has a name.

Input: Information that’s received.

Output: Information that’s given back.
Think In and Outside the Box: Variables, Input, and Output

Pick and Choose

Poetry Jam Slam (30 minutes)

Create a short poem or rap about your classmates using variables.

1. Form small groups and write a four-line poem or rap in Notes. The poem or rap should contain at least two variables specific to students, such as name, age, favorite food, where they’re from, or favorite celebrity.

2. Now you are ready to perform your poem or rap. But how do you receive the inputs for their variables? You will need to give the class a signal—for example, if you say a student’s name or point to someone, that classmate provides the input.

3. Perform your poem or rap in turn. Repeat the verse several times, with different students providing inputs, to create an ongoing piece with variables that change. The end result is a piece with several verses.

4. Think about it: Did using variables make writing or performing the song easier? Why or why not?
Think in Practice: Design UI

**Introduction** (15 minutes)
Think of a few examples of apps that you use almost every day. This could be your favorite game or video app.

What makes it a great app? Write down your favorite parts—what do you like about it? Did you know how to use it at first? Are any parts of the app confusing?

These are the elements that make up the User Interface (UI).

Now think about what it’s like to use the app. Write down which parts of the app do you use most or least often? Is there a reason for this? Here you’re thinking about the experience that the user has of the app, or User Experience (UX).

Now think about what makes the app easy or difficult to use. Can you easily find their way around the app? Here you’re learning about navigation, which is how the user moves through an app.

**Practice** (30 minutes)
Complete Dragon Spells Lesson 10.

**Think about it:** What elements did you design in Dragon Spells? How did you make your design decisions?

---

**User Interface (UI):** What users see in front of them as they use an app, website, or other coding product.

**User Experience (UX):** The experience users have of the app.

**Navigation:** How users move through an app.
Think in Practice: Design UI
Pick and Choose

Class Commander Spots the Signs (30 minutes)
Develop your understanding of UI and UX and get to design a school sign.

1. Download this Keynote template, and as a group, flash through the set of familiar signs. Quickly shout out what you think the signs mean.

2. Together, discuss which parts of the signs make them easy to understand? What makes the silly signs difficult to understand? Explore the use of color, layout, text, fonts, and images.

3. In small groups, think of a sign that would be useful for the school—for example, indicating where students should line up for lunch, or pointing to a quiet zone for reading.

4. In Notes, brainstorm and describe what you’re trying to communicate, and which of the many available design tools they can use to do this effectively—for example, color, images, and words.

5. Using paper and crayons, or a drawing app like Pages, sketch out your ideas for a sign. You should think about how to make your sign noticeable, attractive, and easy to understand.

6. Present their sign to the group and see if they can guess the meaning and function of each sign.