

Introduction to Basic Coding Course

The range of courses involves six basic concepts of programming: **Event, Sequencing, Loop, Conditional, Function, Variable**. It is designed to motivate students' interest in learning through sessions like Everyday Examples, Fun Games, Task Card, Project Design, Self-review and Project Report, and aims to help students lay a solid foundation for further study in programming. The courses can be divided into 8-10 class sessions (16-20 class periods) and provide projects and task cards with different difficulty levels, enabling teachers to tailor their plans according to the ages and current level of knowledge of students.

Product Overview:

We design the courses on the basis of Codey Rocky (hardware) and the mBlock 5 (software) . The combination of playful hardware and powerful software let children learn to code while playing and allows them to develop themselves in terms of programming skills, logical thinking, computational thinking and collaborative skills.

Codey Rocky: Codey+Rocky. Codey Rocky is an educational programmable robot. It has two parts, Codey and Rocky. Codey is equipped with plenty of built-in electronic modules like sensors, IR receiver, and servo. It welcomes originality from children. Therefore, simply with Codey, children are able to design their own games as a way to reinforce their creativity, logical thinking, and talents of arts and music. When teaching the first three concepts (Event, Sequencing and Loop), educators only need Codey as the teaching aid.



Figure 1 Product Layout of Codey

As the chassis of Codey, Rocky puts more features into Codey Rocky, for instance, color recognition and line-following. When educators move on to the latter three concepts (Conditional, Function and Variable), Rocky is going to play a role in the teaching, bringing diversity into the challenges and making it more fun for students to conquer the challenges.

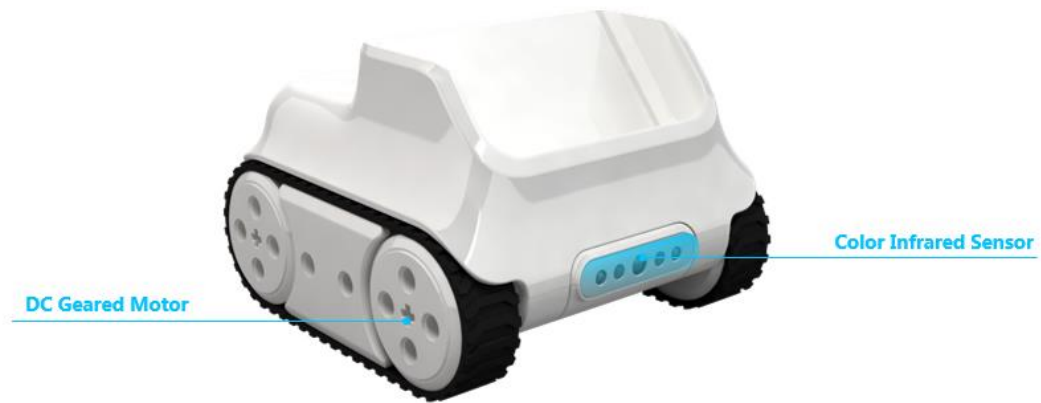


Figure 2 Product Layout of Rocky

mBlock 5: The mBlock 5 is a programming software tool which is inspired by Scratch and aims at STEAM education. It supports graphical and text programming languages, allowing users to design fascinating stories, games and animations. Moreover, with mBlock 5, kids can write programs for massive hardware, including Makeblock robots, Arduino and micro: bit.



Figure 3 mBlock 5

Table of Contents:

(We will post one course every Monday in consideration of the translation progress)

1. Event
2. Sequencing
3. Loop
4. Conditional
5. Function
6. Variable

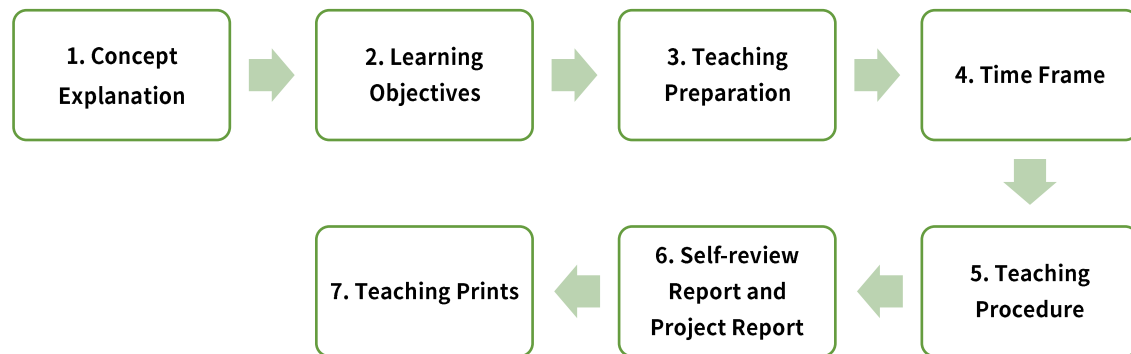
Lesson Grid:

| Course Title | Concept | Time (class period) | Teaching Objectives | Device | Age |
|--|-------------|---------------------|--|----------------|-----|
| Basic Coding Course (Coding + Codey Rocky) | Event | 2 | 1. Understand the concept of Event; 2. Write programs including multiple events. | Codey | 7+ |
| | Sequencing | 2 | 1. Understand the concept of Sequencing; 2. Design a procedure and an animation; 3. Understand the concept of Bug; 4. Find bugs and debug. | Codey | 7+ |
| | Loop | 2 | 1. Understand the concept of Loop; 2. Identify the difference between Counting Loop and Infinite Loop; 3. Design animations by applying the counting loop and the infinite loop. | Codey Rocky | 7+ |
| | Conditional | 4 | 1. Understand the concept of Conditional; 2. Fulfill tasks and create projects by writing programs with the blocks of conditionals. | Codey Rocky | 8+ |
| | Function | 4 | 1. Understand the concept of Function; 2. Fulfill the tasks and create projects by creating functions. | Codey Rocky | 9+ |
| | Variable | 6 | 1. Understand the concept of Variable; 2. Create projects with variables; 3. Tackle the challenges by applying what you've learned about the six concepts. | Codey Rocky | 9+ |

*Note: One class period lasts between 45 and 50 minutes.

There are six documents of courses in the file. Each document can be divided into seven parts, Concept Explanation, Learning Objectives, Teaching Preparation, Time Frame, Teaching Procedure,

Self-review Report and(or) Project Report, and printed materials (used for teaching) .



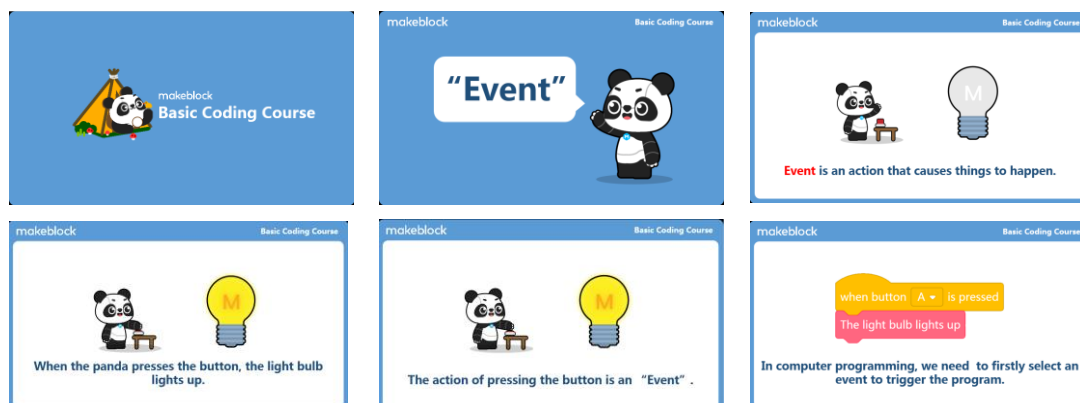
Teaching Procedure:

The teaching procedure is roughly divided into five parts:

| Review | New Knowledge | Lead-in Game | Student Activity | Wrap up |
|--|---|---|--|--|
| <ul style="list-style-type: none"> Review the knowledge by asking questions | <ul style="list-style-type: none"> Explain the knowledge through everyday examples | <ul style="list-style-type: none"> Have students turn off the computer and learn through interactive games | <ul style="list-style-type: none"> Help students master the tasks by giving them instructional scaffolds first and removing them later. | <ul style="list-style-type: none"> Wrap up the new knowledge for today and fill in the self-review report |

Review: Review the knowledge of last class by asking questions.

Explain New Knowledge: Explain the new concept through everyday examples with PowerPoint presentation.



Lead-in Game: Plugging out Power Supply is a teaching method that is commonly used in programming education. It means that children are supposed to turn off the computer and participate in interactive games which center on the concepts they are expected to master. Through engaging

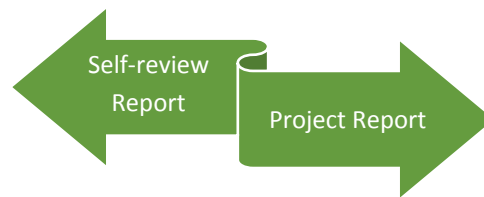
games, children are able to have a clearer picture of important concepts and increase their curiosity for coding. We offer Plugging out Power Supply games for each concept in this course and hope educators can make the best of them to ignite children's passion for coding.

Student Activity: In the Student Activity session, the course will provide a wide range of tasks for students to practice what they've learned. And to help students, teachers can give instructional scaffolds in this session. From beginning to advanced tasks, teachers will try to withdraw the scaffolds progressively and finally let students deliver projects and fulfill tasks on their own.

Wrap up: Teachers and students work together to wrap up what they've learned. Have students fill in the self-review report.

Assessment:

The assessment session consists of two parts: Self-review Report and Project Report:



The self-review report is designed to assist students in reviewing the concept they've learned about at the end of the session and what they like or dislike about the class session. To avoid annoyance among students, it's important to make it clear in the report that **students can give one or two sentences as the answer to each question.**

Project Report: When students are learning about the first three concepts(Event, Sequencing and Loop), they are expected to deliver a simple project at the end of each session. To achieve this goal, the project report is designed to make students: 1) Have a clear design goal 2) Record the inspirations 3) Review the strengths and weaknesses of the project 4) Review the collaborative process 5) Refine the plan. Moreover, children can follow the questions flow of the project report to showcase their projects.

And the courses try to help students have a better understanding of the latter three concepts by taking the form of Challenges Cards. Students can accomplish the challenges independently or collaboratively. So you might notice that there is no project report for last three concepts.

Cooperative Learning Advice:



We suggest that educators encourage students to **Program in Pair** in the Student Activity sessions. That is to say, one student acts as the driver who controls the mouse and another student as the navigator who directs the driver and controls the Codey Rocky. The ideal situation is: The two students learn from each other while accomplishing the task in pairs. Both of them must follow the rule: The navigator is forbidden to touch the mouse and two students should switch roles when they finish one task.

Event

Overview

| | |
|-----------------------------|--|
| Concept | Event ——An action that causes things to happen |
| Explanation | Example: When you press the button, the light is turned on. Event: press the button Result: the light is turned on |
| Learning Objectives | 1. Understand the concept of Event; 2. Write programs that include multiple Event; |
| Teaching Preparation | 1. One green flag and one red flag; 2. A whiteboard and a whiteboard marker(or you can use a blackboard and chalks); 3. One Codey and a Bluetooth dongle(or the USB cable)for each student but it's fine if 2 or 3 students share one set; 4. A computer with installed mBlock 5 for each student but it's fine if 2 or 3 students share a computer; 5. Hand out a self-review report and a project report for each student. |
| Time | 30-60min |

Teaching Procedure

Step 1: Icebreaking Game - Self-introduction

Teachers and student give a self-introduction first.

Game rule: Have everyone repeat the names of all the people who have introduced themselves before. For instance, the first student said, "I'm Laven"; the second one said, "I'm Denial, the previous student is called Laven"; the third one introduced, "I'm Herry, and the previous two students are Laven and Denial"... And the last one said, "I'm Teddy, the previous students are Laven, Denial, Herry, Zoe, Melody, Lily and Nancy." Once again with the same rule in the opposite direction.

Step 2: Explain the Key Point - Event

Explain the concept of Event to students: Event refers to the action that triggers something to happen. Teachers can give an example here. For instance, when you press the button, the light will

be turned on. In this case, pressing the button is the Event and the result is that the light is turned on.

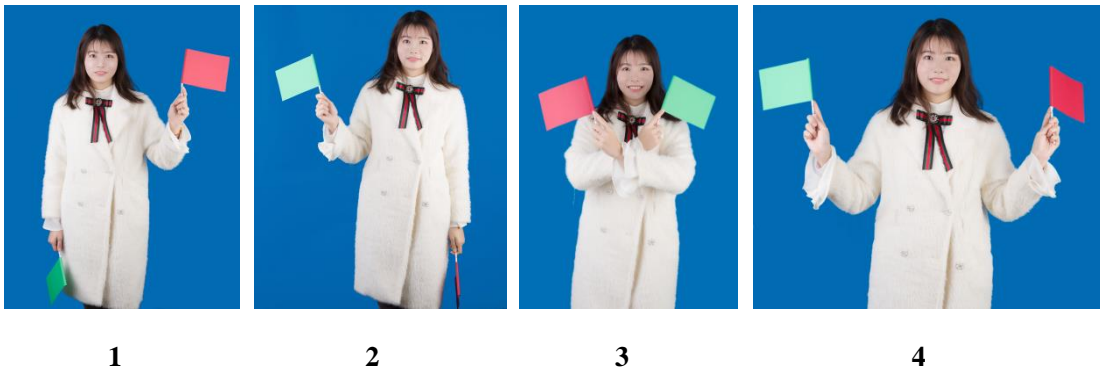
Brainstorming

Ask students questions: Are there any Event in daily life? And what happens due to the Event? Give students time to think and teachers can drop some hints in the process. To name a few, the alarm goes off (Event) so you get up(what happens next); press the switch(Event) and the television is turned on (what happens next).

Step 3: Lead-in Game Red and Green Flag

Teachers can design four events:

1. Raise the red flag with one single hand;
2. Raise the green flag with one single hand;
3. Raise the two flags with both hands and cross the arms over your chest;
4. Raise the two flags with both hands and spread the arms.



Have students stand up first.

The four events trigger the following actions accordingly:

1. Raise the red flag with one single hand - Put both hands on the shoulder;
2. Raise the green flag with one single hand - Put both hands on the waist;
3. Raise the two flags with both hands and cross the arms over your chest -Put both hands on the knees;
4. Raise the two flags with both hands and spread the arms - Clench your fists over your chest.



1

2

3

4

When students have no trouble performing the actions, it is time to have them make some sounds as well:

1. Raise the red flag with one single hand - Put both hands on the shoulder and shout out “Do”;
2. Raise the green flag with one single hand - Put both hands on the waist and shout out “Re”;
3. Raise the two flags with both hands and cross the arms over your chest -Put both hands on the knees and shout out “Mi”;
4. Raise the two flags with both hands and spread the arms - Clench your fists over your chest and shout out “Bingo”.

Teachers are the ones that wave the two flags, and students are the ones that make actions and sounds in response to the events.

Tips:

1. Teachers can personalize the actions and sounds to meet the different teaching purposes or the needs of students;
2. When students are skilled at following the commands, teachers can speed up a little bit and make it more rhythmic.

Step 4: Student Activity

Task 1: Download the software and connect the device.

Navigate students to download mBlock 5 for PC.

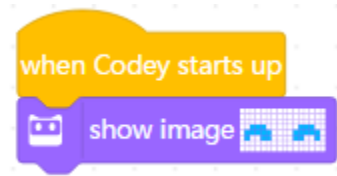
Tips:

1. Teachers can install mBlock 5 on the computer beforehand to save time for the class;
2. Before the class begins, the first thing is to test each of the computers. Make sure mBlock 5 can work well on each of the computer and codes can be uploaded to Codey Rocky;

3. Before the class begins, we should check each Codey Rocky to see whether it functions well or not.

Task 2: Demonstrate

Teachers operate and students follow. (reference case for teachers: *Start up and Smile*)



Program Story:

The yellow-headed block 【when Codey starts up】 is the Event: it means when Codey starts up;

You can edit on the blue block 【show image】 to change the image on the LED display: Codey smiles.

Task 3: Practice

Have students work in a pair to finish the task as the teacher did. Tell them to match different expressions or images to the three events respectively: when button A pressed; when button B pressed; when button C pressed.

Tips:

If students are quick learners, then it's OK to jump to task 4 directly.

Task 4: Let students create

Have students work in a pair to finish the task. Tell them to match different expressions or images to the three events respectively: when button A pressed; when button B pressed; when button C pressed. At the end of this session, teachers should direct students to share their works with the class. Students need to fill in the project report before they share the works and they need to display the works by following the questions from the report.

Tips:

1. If time allows, teachers can let students edit the images as they like;
2. If there are some students who fail to finish the task on time, invite them to share one thing that happened when they were writing programs. It could be something funny, a challenge they confront or a problem they have;

3. Students can display their works to the whole class or they can share the works with groups one by one;

4. Students might not be willing to spend much time filling in the project report so tell them to put down key points;

5. When students are showing their works, they need to give details about the work and elaborate on the project report;

6. Teachers can tailor the time limits in line with different teaching purposes and students characteristics. Recommended time: 10 min

Teachers can refer to the sample project: *Press Button to Change Expression*:



Program Story:

【When button A is pressed】 , then 【play sound switch】 and Codey open its eyes;

【When button B is pressed】 , then 【play sound switch】 and Codey smiles;

【When button C is pressed】 , then 【play sound switch】 and Codey becomes sad.

Extension Task: Add LED lights

1. Add a new action for each event – Different RGB light colors;

2. Challenge: Try to program with other events, like 【when Codey is shaking】 .

Step 4: Wrap up

Summarize what the topic is for today, Event and its concept. Teachers need to highlight one thing: The event serves as the starting point for a program. When we are writing programs, the first thing we should do is to select a proper triggering event.

Self-review Report

Name:

Age:



- Answer the following questions and record your outcome:

Describe what you've learned with one or two sentences.

Describe what you like most and least about this class session briefly

What I like most

What I like least

Draw an event that happened to you today and an action caused by the event

| Event | Action |
|-------|--------|
| | |

Tell a short story to us (Describe how the codes work from top to bottom)





You can paint how you feel about this class session in the upper right corner of the self-review report.

Project Report

Name:

Group Name:

- Follow these questions to show your work:

What events did you use? What's the image, sound or light for each event?

| Event | Action |
|-------|--------|
| | |

Do you like your design? What are you satisfied with? What are you unhappy with? Any improvements in the future?

| |
|---|
| <input type="radio"/> Love it <input type="radio"/> Like it <input type="radio"/> So-so <input type="radio"/> Don't like it <input type="radio"/> Hate it |
| |

What's the most amazing thing for you when you were creating the project?

| |
|--|
| |
|--|

Did you come across any obstacles? How did you overcome?

| |
|--|
| |
|--|

Instructor Assessment

1. Cooperation (30%): Evaluate how the group performs in terms of labor division, collaboration and coordination.
2. Completeness (20%) : Evaluate whether the project is complete enough. Of course, the project must stick to the topic first.
3. Innovation (20%) : Evaluate how creative the project is.
4. Functionality (20%): Evaluate whether the work is functional enough?
5. Difficulty (10%) : Evaluate what is the difficulty level of the work?

CSTA

| Grades | Identifier | Interim CSTA K-12 CS Standards | Framework | Framework |
|------------|------------|--|-------------------------|----------------------------------|
| | | | Concept | Practice |
| K-2 | 1A-A-5-2 | Construct programs, to accomplish a task or as a means of creative expression, which include sequencing, events, and simple loops, using a block-based visual programming language, both independently and collaboratively (e.g., pair programming). | Algorithms and Programs | Creating Computational Artifacts |
| 3-5 | 1B-A-5-4 | Construct programs, in order to solve a problem or for creative expression, that include sequencing, events, loops, conditionals, parallelism, and variables, using a block-based visual programming language or text-based language, both independently and collaboratively (e.g., pair programming). | Algorithms and Programs | Creating Computational Artifacts |
| 3-5 | 1B-A-6-8 | Analyze and debug (fix) an algorithm that includes sequencing, events, loops, conditionals, parallelism, and variables. | Algorithms and Programs | Testing and Refining |

Sequencing

Overview

| | |
|-----------------------------|--|
| Concept | Sequencing —— A set of ordered steps for accomplishing a task |
| Explanation | <p>To ensure that Codey Rocky can fulfill the task as expected, we should give instructions in the correct order.</p> <p>Example: Put the watermelon in the refrigerator.</p> <p>Bug: Mistakes that cause the failure of programs to run as programmed</p> <p>Debug: Find bugs and fix them.</p> |
| Learning Objectives | <ol style="list-style-type: none"> 1. Understand the concept of Sequencing; 2. Design a set of steps for making an animation; 3. Understand the concept of Bug; 4. Find bugs and fix them. |
| Teaching Preparation | <ol style="list-style-type: none"> 1. A whiteboard and a whiteboard marker (or you can use a blackboard and chalks); 2. One Codey and a Bluetooth dongle (or the USB cable) for each student but it's fine if 2 or 3 students share one set; 3. A computer with installed mBlock 5 for each student but it's fine if 2 or 3 students share a computer; 4. Hand out a self-review report and a project report for each student. |
| Time | 90-120 min |

Teaching Procedure

Step 1: Review —— Event

Review:

- What is Event?
- Can students think of any events in daily lives?
- What events did you use in the last class?

Event is an action that causes things to happen. For example, when you press the button, the light is turned on. In this case, pressing the button is the event and the result is that the light is turned on. The events students used in the last class session are: 1) When the program starts up 2) When Button A, B and C pressed.

Step 2: Explain New Knowledge —Sequencing

The new concept for this class session is Sequencing. For instance, if we want to put the watermelon in the refrigerator, then we need to follow these steps:

1. Open the refrigerator door.
2. Put the watermelon in the refrigerator.
3. Close the refrigerator door.

If you follow the wrong sequence, you wouldn't be able to put the watermelon in the refrigerator.

Brainstorming

Ask students if they can think of any cases in which they must follow a set of steps to achieve something. Teachers can give an example here:

You must first uncap the bottle, pour the water into your mouth and then screw the cap of the bottle.

If you don't follow the steps, you won't be able to drink the water. (Tip: Each example should only offer one specific order. It means that only when you follow one specific set of steps, can things be done in the example.)

To fulfill a task, you must follow a set of steps. The order the steps are performed is called as **Sequencing**.

Step 3: Lead-in Game - I'm a Robot

The teacher acts like a robot, walking from somewhere in the classroom to the blackboard and drawing a smiley face on it. Students give instructions to the robot and write them on papers.

Game Instructions:

1. Teachers read the students' instructions from top to bottom.
2. Ask students to leave detailed instructions in the correct order.

Tips:

1. If students write instructions from left to right, teachers should still stick to read instructions from top to bottom. In this case, there is a possibility that instructions can only be read from left.
2. When students' instructions are unclear, teachers still need to follow the instructions to make actions. For example, if the instruction is: turn left, move ahead by 4 meters, then the robot should

execute the instruction like: turn left and move ahead. This is exactly how the software instructions are performed. When there is no specific setting for the time and the angle, the computer will read the simple instruction for turning left promptly and then read the instruction for moving ahead;

3. If it's necessary to make the instructions more specific, you can remind students about the fact that the robot lays down its two hands vertically. Therefore, when students are giving certain instructions, they need to make sure that the instructions are detailed enough. For instance, if the instruction for the robot is to pick up a pen, the instruction must include details: by which hand, the hand gesture, where to draw the smiley face exactly on the blackboard, etc;

4. In consideration of the time limit and the ages of students, teachers can simplify the instructions. Anyway, the key point is clear: you need to make instructions specific and arrange them in the correct order if you want the robot to do things as you program.

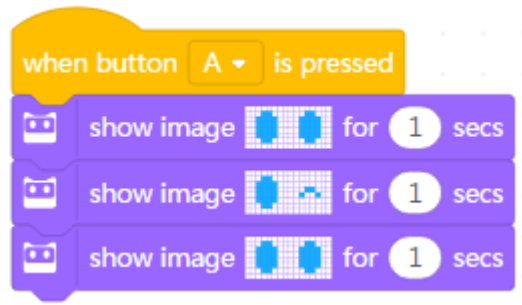
Summary: When we are programming, we arrange the blocks in the order from top to bottom to form a set of steps. Machines can follow the steps one by one to perform a task. We refer to the set of steps as **Sequencing**.

Step 4: Student Activity - Animation Design

Design animations with the block **【show image () for () secs】**. Traditionally, animation designers would make animations in the following way: Put a sheet of static drawing on the table first and unfold a new drawing paper on top of the first paper. Designers will outline the frame and then change the drawing slightly. Then another new paper, outline the frame and change the drawing slightly again. Designers repeat the steps over and over again until they complete a series of pictures that are slightly different from each other. Then, they flip the drawings quickly to make them smooth. Simply speaking, designers animate the drawings.

Task 1: Follow the teachers to design animations

Demonstrate: In this session, teachers will show how to make animations with the block **【show image () for () secs】**. The method is simple: Use the image of the block as the starting picture, duplicate the block, and change the image slightly. Repeat the steps and then arrange those blocks in sequence. **Sample project: Winking Eyes**



Program Story :

Drag out the block **【show image () for () secs】** and edit the first image to be Codey opening eyes;

Duplicate the block **【show image () for () secs】** but edit one eye to be winking;

Duplicate the block **【show image () for () secs】** once again, and this time make the winking eye open.

Add the Event block — “When button A pressed”

Upload the programs to the device. When the button A is pressed, Codey will wink at you.

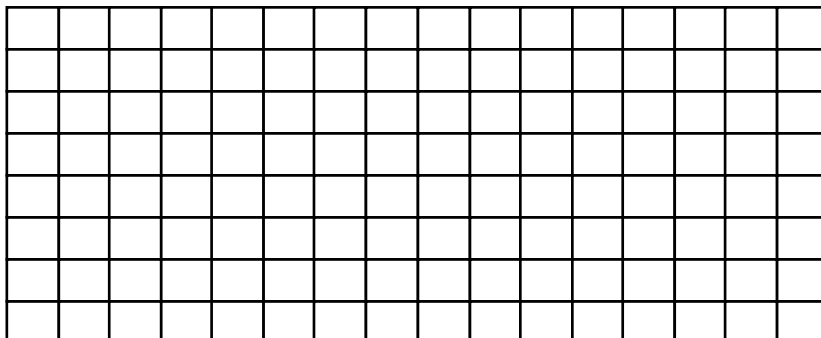
Student Activity:

Students work in pairs to make their animations as the teachers did.

Task 2: Design New Animations

At the end of this session, students need to showcase their works and fill in the project reports.

Students work in pairs and fill the little boxes to form an image first; draw the same image on the block **【show image () for () secs】** ; duplicate the block and change the image slightly; repeat the step until you have a series of blocks that are changing and arranged in sequence.

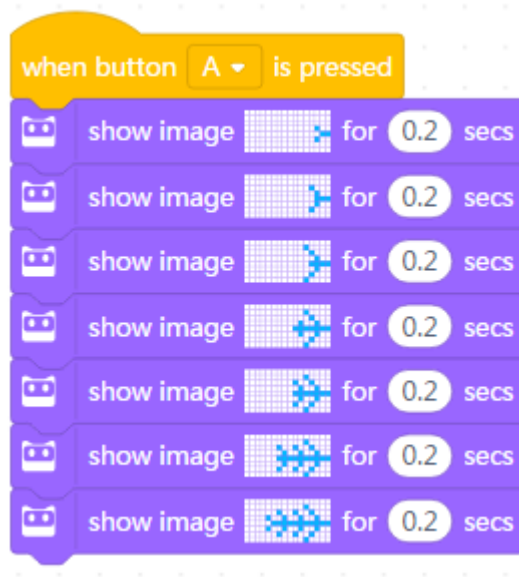


Upload codes to examine the animation effect. Students need to present their works but should complete the project report first. They should display their works by following the questions from the report.

Tips

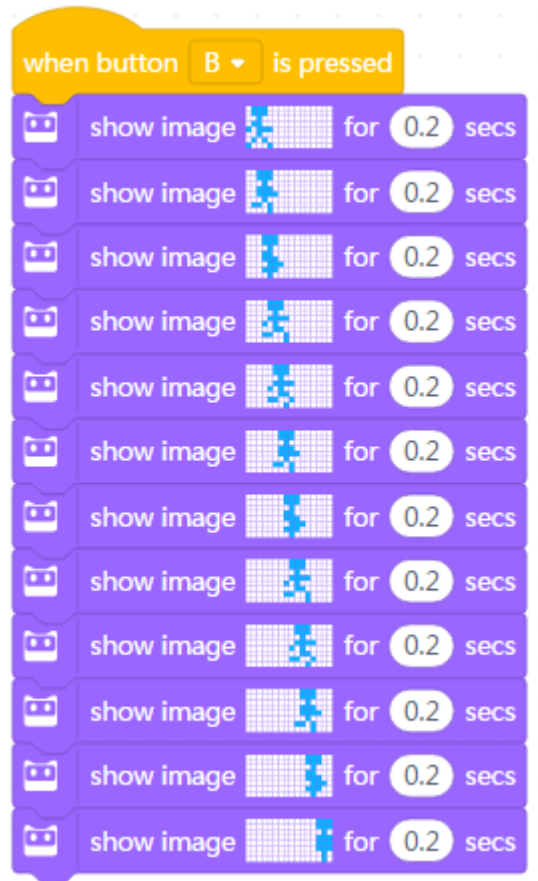
1. If some students accomplish the task ahead of time, tell them to start their animations with multiple events;
2. Or have them change how long they show the image;
3. If some students fail to accomplish the task on time, invite them to share one thing that happened when they were writing programs. It could be something funny, a challenge they confront or a problem they have;
4. Students can share their works with the whole class, or they can take turns presenting the works to other groups ;
5. Teachers can tailor the time limit according to teaching purposes and the personality of the class. Recommended time: 10 min

Teachers can show sample projects: *Growing Tree* and *Walking*



Program Story:

Place the LED matrix display upright. When the button B is pressed, a sapling will be growing.



Program Story:

When the button B is pressed, a child will be running from one side of the LED matrix display to the other side.

Expert Tips: Bug and Debug

Introduce the new concepts Bug and Debug before students work on their tasks.

When we are writing programs, bugs are inevitable. A bug means an insect in its literal sense but refers to a glitch in a computer program here. At the initial stage where computers were invented, unlike the portable computers of today, they were extremely large in size back then. (Picture from Wiki).

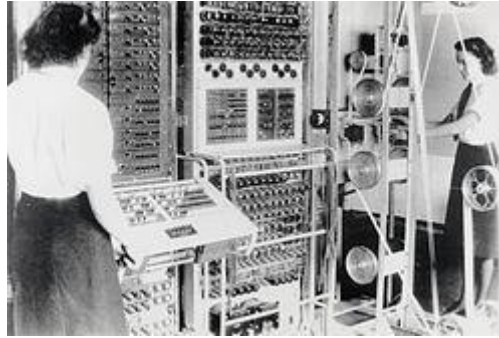


Figure 1 Colossus computers were used to decipher German codes during WW II

It was during this period that a colossus computer failed to work properly. The whole team of programmers was trying so hard to find out the problem but to no avail. In the end, Grace Murray Hopper, a female programmer, identified what the problem was: a moth flew to the inside of the computer and caused the glitch. When they removed the moth, everything was back on the right track. It was the first bug in a computer program that was found and programmers affixed it to the log book (see the picture below). From now on, the term bug is common in use when people refer to the mistakes in a computer program. Naturally, Grace Murray Hopper was since then considered as the Mother of Debug. (Picture from Wiki)

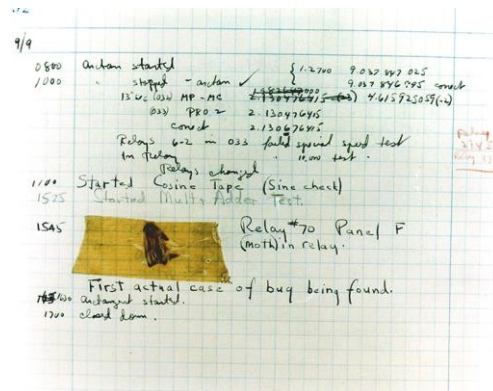


Figure 2 The moth caused the computer mistake and this is the first mistake in a computer program

When writing programs, we need to go through each line of the codes to find bugs and fix them.
 Find bugs in the following programs and try to fix them:

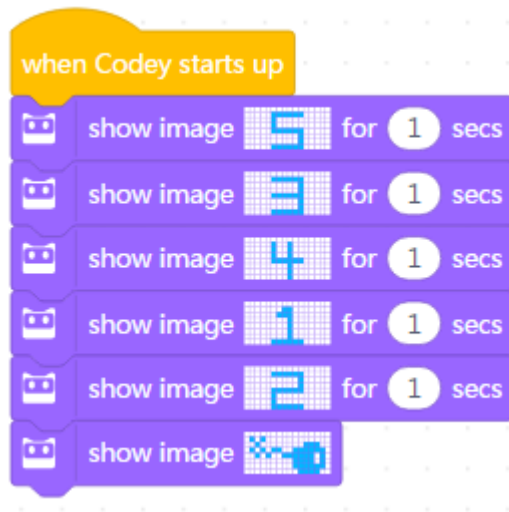
1. Earthworm and Insect



Program Story:

A little earthworm meets a big insect when crawling on the ground. Remove the insect to let the earthworm crawl again (change the insect image to be an image of the earthworm crawling).

2. The Bomb Can't Count Down!!



Program Story:

There is a bomb that can't time. Help it!

3. The car key was stolen!



Program Story:

You can't start the car because the car key was stolen. You have to find the key right now! (a missing event).

Step 5: Wrap up

1. Summarize what the concept of **Sequencing** is. **Sequencing** refers to a set of ordered steps for performing a task.
2. In the mBlock software, the sequencing is **from top to bottom**. If Codey Rocky fails to perform the task as programmed, you can go over each line of codes from top to bottom to find bugs.
3. When you design animations, you can start with a sheet of basic picture, copy the picture and change it slightly every single time. Finally, play the series of pictures in sequence to animate them.
4. Students complete the self-review report.

Self-review Report

Name:

Age:



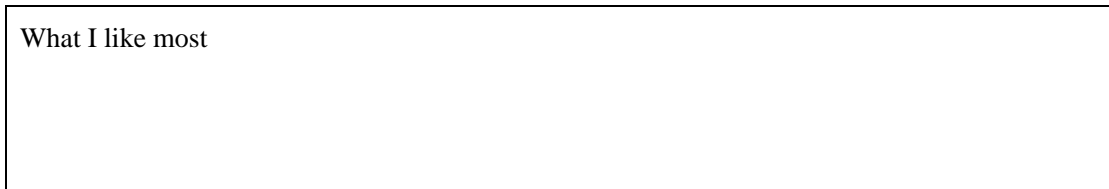
- Answer the following questions and record your outcome:

Describe what you've learned with one or two sentences.



Describe what you like most and least about this class session briefly

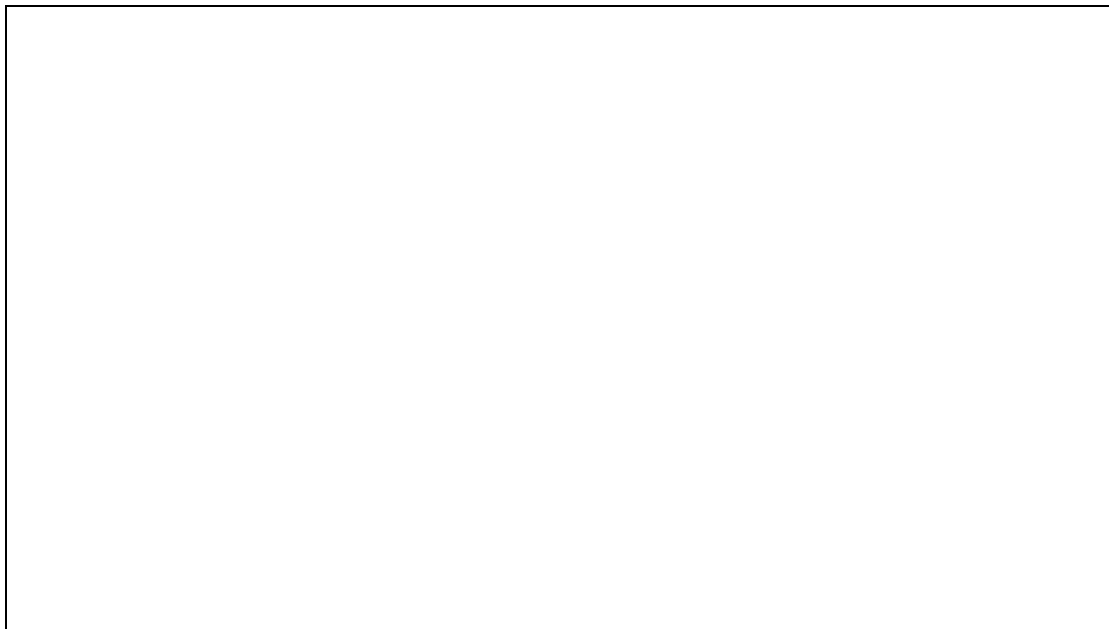
What I like most



What I like least



Draw a **Sequencing** that happened around you today:



You can paint how you feel about this class session in the upper right corner of the self-review report.

Project Report

Name:

Group Name:

- Follow these questions to present your work:

What tasks did you take on? Describe it in one or two sentences.

Did you have any ideas on how to fulfill the tasks?

Write down or draw every single bit of inspiration, being it good or bad. You are doing this to explore more possibilities. Use more papers to record your ideas as you like.

Describe the final effect of your project and why you choose the effect in one or two sentences

Effect

Example: When the button A is pressed, the earthworm starts crawling ahead.

Why you choose this (these) effect?

Example: Because it's funny.

Did you come across any obstacles? Did you have any solutions? Describe it with one or two sentences. (The solution can be a rough plan)

| Difficulties | Solutions |
|--|--|
| <i>Example: Failed to connect Codey Rocky to mBlock; Couldn't upload codes; Failed to achieve consensus...</i> | <i>Example: Power on Codey Rocky; Rock Paper Scissors...</i> |
| | |

Students can answer the following questions after the sharing session.

Do you like your design? Describe what you like most and least about the project with one or two sentences. And any improvements in the future?

| <input type="radio"/> Love it <input type="radio"/> Like it <input type="radio"/> So-so <input type="radio"/> Don't like it <input type="radio"/> Hate it |
|---|
| What I like most <i>Example: The animation is vivid; the sound matches the animation.</i> |
| What I like least <i>Example: It's challenging for people to tell what the animation is about at first glance.</i> |
| Improve <i>Example: Redesign the animation and make it simpler this time; add sounds and lights.</i> |

Instructor Assessment

1. Cooperation (30%): Evaluate how the group performs regarding labor division, collaboration, and coordination.
2. Completeness (20%) : Evaluate whether the project is complete enough. Of course, the project must stick to the topic first.
3. Innovation (20%) : Evaluate how creative the project is.
4. Functionality (20%): Evaluate whether the work is functional enough?
5. Difficulty (10%) : Evaluate what is the difficulty level of the work?

CSTA

| Grades | Identifier | Interim CSTA K-12 CS Standards | Framework Concept | Framework Practice |
|------------|------------|--|-------------------------|---|
| K-2 | 1A-A-5-2 | Construct programs, to accomplish a task or as a means of creative expression, which include sequencing, events, and simple loops, using a block-based visual programming language, both independently and collaboratively (e.g., pair programming). | Algorithms and Programs | Creating Computational Artifacts |
| K-2 | 1A-A-5-3 | Plan and create a design document to illustrate thoughts, ideas, and stories in a sequential (step-by-step) manner (e.g., story map, storyboard, sequential graphic organizer). | Algorithms and Programs | Creating Computational Artifacts |
| K-2 | 1A-A-3-7 | Construct and execute algorithms (sets of step-by-step instructions) that include sequencing and simple loops to accomplish a task, both independently and collaboratively, with or without a computing device. | Algorithms and Programs | Recognizing and Defining Computational Problems |
| K-2 | 1A-A-6-8 | Analyze and debug (fix) an algorithm that includes sequencing and simple loops, with or without a computing device. | Algorithms and Programs | Testing and Refining |
| 3-5 | 1B-A-2-1 | Apply collaboration strategies to support problem solving within the design cycle of a program. | Algorithms and Programs | Collaborating |
| 3-5 | 1B-A-5-4 | Construct programs, in order to solve a problem or for creative expression, that include sequencing, events, loops, conditionals, parallelism, and variables, using a block-based visual programming language or text-based language, both independently and collaboratively (e.g., pair programming). | Algorithms and Programs | Creating Computational Artifacts |
| 3-5 | 1B-A-3-7 | Construct and execute an algorithm (set of step-by-step instructions) that includes sequencing, loops, and conditionals to accomplish a task, both independently and collaboratively, with or without a computing device. | Algorithms and Programs | Recognizing and Defining Computational Problems |
| 3-5 | 1B-A-6-8 | Analyze and debug (fix) an algorithm that includes sequencing, events, loops, conditionals, parallelism, and variables. | Algorithms and Programs | Testing and Refining |



| | | | | |
|-----|-----------|---|----------------------|--|
| 3-5 | 1B-I-1-17 | Seek out and compare diverse perspectives, synchronously or asynchronously, to improve a project. | Impacts of Computing | Fostering an Inclusive Computing Culture |
|-----|-----------|---|----------------------|--|

Loop

Overview

| | |
|-----------------------------|--|
| Concept | Forever ——Repeat over and over again |
| Explanation | Repeat () times ——Repeat specific times Example: Walk, Breathe, Brush teeth, Sunrise, Sunset, etc. |
| Learning Objectives | 1. Understand the concept of Loop; 2. Distinguish between Counting Loop and Infinite Loop; 3. Design animations using counting loops and infinite loops. |
| Teaching Preparation | 1. A whiteboard and a whiteboard marker (or you can use a blackboard and chalks); 2. One Codey and a Bluetooth dongle (or the USB cable) for each student but it's fine if 2 or 3 students share one set; 3. A computer with installed mBlock 5 per student, but it's fine if 2 or 3 students share a computer; 4. Hand out a self-review report and a project report for each student. |
| Time Frame | 60-90min |

Teaching Procedure

Step 1: Review - Sequencing

Review:

- What is **Sequencing**?
- Can students think of any cases of sequencing in everyday lives?
- What are **Bug** and **Debug**?
- How to find bugs?

Sequencing refers to a set of steps for accomplishing a task. For example, putting the watermelon in the refrigerator, brushing teeth, washing hair, etc.

Bugs are the mistakes that cause the failure of a computer program to run as programmed. **Debug** is a process of finding bugs and fixing them. To find bugs, we need to go over the programs from top to bottom.

Step 2: Explain New Knowledge - Loop

To make our environments better, sanitation workers are going to plant trees alongside the pavement. The exact steps to plant a tree are: dig a hole, put down a sapling, cover it with soil and move forward 5 meters to plant another tree. If the road is 20 meters in length, then sanitation workers will have to repeat the same sequence of steps four times.

In computer programs, we sometimes use counting loops to repeat the piece of codes a specific number of times. On other occasions, the loop might be forever, or in other words, having no terminating conditions. In everyday life, infinite loops are happening around us, like sunset and sunrise. Similarly, when writing programs, we use the **forever** blocks to run the piece of codes endlessly.

Loop and **Repeat** have the same meaning. If you repeat doing something, you are performing a loop. Ask students if they can think of any repetitive behaviors and how many times they repeat. Teachers can give cues here: brush teeth, two times each day; breathe, forever.

Step 3: Lead-in Game – Tap to the beat

Game Rule:

1. The letter A represents tapping the table with the left hand; B represents tapping the table with the right hand; C represents tapping the table with both hands. Tap the table by following the beats below:

ABABABC ABABABC

Chaos might ensue when students are required to tap the table at a rapid speed. In this case, we suggest that teachers ask students if it's necessary to make the process smoother by simplifying the beats. Simplify the beats to be:

ABABABC Repeat two times

2. What if we introduce another beat? Add an action, D for clapping the hands:

ABABABC ABABABC ABCABC ABABABC

ABABABD ABABABD ABDABD ABABABD

ABABABC ABABABC ABCABC ABABABC

ABABABD ABABABD ABDABD ABABABD

Simplify the beats:

| | | |
|--------------------------|---|------------------|
| ABABABC Repeat two times | } | Repeat two times |
| ABC Repeat two times | | |
| ABABABC | | |
| ABABABD Repeat two times | | |
| ABD Repeat two times | | |
| ABABABD | | |

Tips:

1. Teachers write beats on the blackboard and can mark the beats while working with students to figure out the rules;
2. Make sure the beats are simple if you are to design the rhythm yourself. Also, don't make the game too long, or otherwise students might lose interest soon.

Step 4: Student Activity

When writing programs, we often use the **Repeat** blocks to make our programs neat. They can save us from dragging the same block over and over again. With the Repeat blocks, Codey Rocky can perform the same sequence of instructions repeatedly. When the Repeat block is executed once, we count it as one cycle or one iteration.

The repeat can be forever or happens specific times. For instance, we brush our teeth twice or three times a day while the sunrise and the sunset happen around us endlessly. The two different loops blocks are as follows:



Ask students a question: what is the difference between **forever** and **repeat () times**. Direct the attention of students to one fact: the **forever** block has no bump at the bottom that enables it to interlock other blocks. The forever block is designed to repeat the codes endlessly, so we can't affix other blocks to it.

Task 1: The Steamed Bread Can't Jump

Story: The steamed bread is persisting in learning how to jumping up. It tried many times but failed to make it. Teachers show the animation of a steamed bread failing to jump and demonstrate how to make the animation. Next, it's time for students to work on their animations in the same way as teachers do. Sample project: *The Steamed Bread Can't Jump*

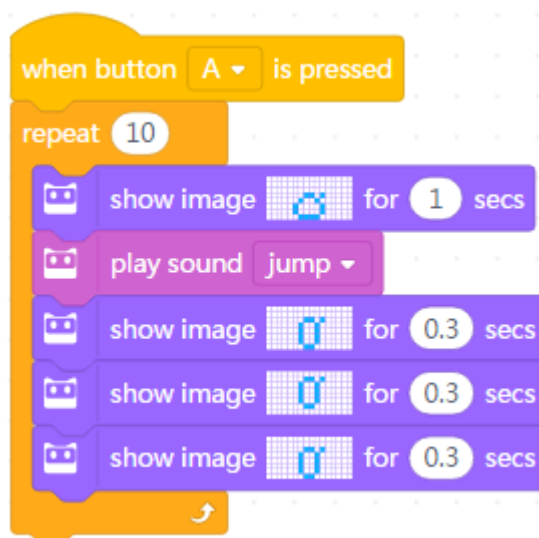


Program Story:

The steamed bread wants to jump up. It tried ten times but still can't make it.

Task 2: The Jumping Steamed Bread

Story: One day, the steamed bread finally knows how to jump. It keeps hopping and can go anywhere it likes. Have student design a program *The Jumping Steamed Bread* using **forever** blocks. Teachers can refer to the sample project: The Jumping Steamed Bread (Not recommended to show it to students too soon. Have them practice by themselves first).



Program Story:

The steamed bread is finally off the ground and keeps bouncing;

Each bounce comes with a sound;

The program will be executed endlessly.

Challenge: Use another event “**when Codey is shaking**”

Task 3: DIY Project

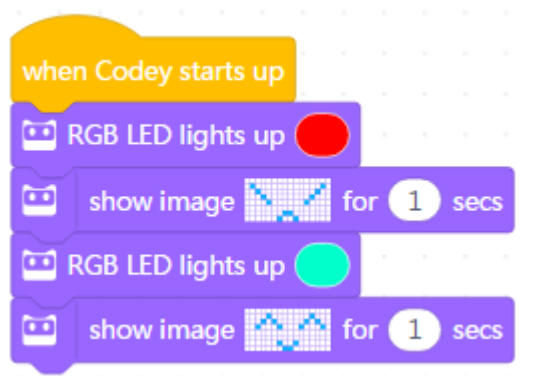
Students work in pairs to design animations using infinite loops or counting loops. They need to present their works but should complete the project report first. Students should display their works by following the questions from the report.

Tips:

1. If students have enough time and are quick learners, tell them to design animations using multiple counting loops or nested loops.
2. Teachers can tailor the time limit according to teaching purposes and the personality of the class. Recommended time: 20 min.

Extensions:

1. Design different animations: triggered by different events(buttons) and made up by a combination of counting loops and infinite loops;
2. Add a mix of sounds and RGB LED effects to make your animation emotional,
for example:



Step 5: Wrap up

Describe what the concept of **Loop** refers to. Loop means that the machine repeats pieces of codes endlessly or a specific number of times.

Students complete the self-review report.

Self-review Report

Name:

Age:



- Answer the following questions and record your outcome:

Describe what you've learned with one or two sentences.

Describe what you like most and least about this class session briefly

What I like most

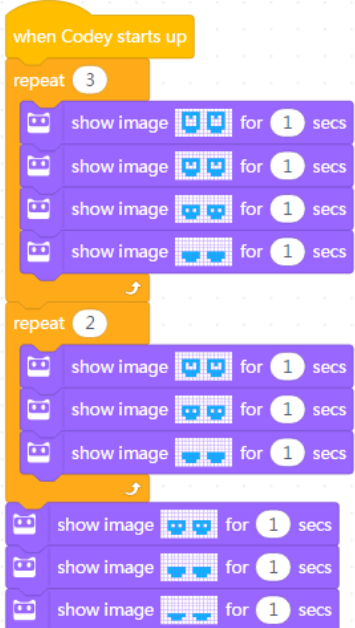
What I like least

Draw a Counting Loop and an Infinite Loop that happened around you today:

Counting Loop

Infinite Loop

Tell us a brief story (Describe how the following codes run from top to bottom)

| | |
|---|--------------------------------------|
|  | <p><i>When Codey is shaking,</i></p> |
|---|--------------------------------------|

You can paint how you feel about this class session in the upper right corner of the self-review report.

Project Report

Name:

Group Name:

- Follow these questions to present your work and you need to give your answers during the presentation.

What tasks did you take on? Describe it in one or two sentences.

Did you have any ideas on how to fulfill the tasks?

Write down or draw every single bit of inspiration, being it good or bad. You are doing this to explore more possibilities. Use more papers to record your ideas as you like.

Describe the final effect of your project and why you choose the effect in one or two sentences

Effect

Example: When button A pressed, the steamed bread bounces ten times.

Why you choose this (these) effect?

Example: It's funny and it's easy to design steamed bread.

Did you come across any obstacles? Did you have any solutions? Describe it with one or two sentences. (The solution can be a rough plan)

| Difficulties | Solutions |
|--|--|
| <i>Example: Failed to connect Codey Rocky to mBlock; Couldn't upload codes; Failed to achieve consensus...</i> | <i>Example: Power on Codey Rocky; Rock Paper Scissors...</i> |
| | |

Students can answer the following questions after the sharing session.

Do you like your design? Describe what you like most and least about the project with one or two sentences. And any improvements in the future?

| |
|---|
| <input type="radio"/> Love it <input type="radio"/> Like it <input type="radio"/> So-so <input type="radio"/> Don't like it <input type="radio"/> Hate it |
| What I like most <i>Example: The animation is vivid; the sound matches the animation.</i> |
| What I like least <i>Example: It's challenging for people to tell what the animation is about at first glance.</i> |
| Improve <i>Example: Redesign the animation and make it simpler this time; add sounds and lights.</i> |

Instructor's Assessment

1. Cooperation (30%): Evaluate how the group performs regarding labor division, collaboration, and coordination.
2. Completeness (20%): Evaluate whether the project is complete enough. Of course, the project must stick to the topic first.
3. Innovation (20%) : Evaluate how creative the project is.
4. Functionality (20%): Evaluate whether the work is functional enough?
5. Difficulty (10%) : Evaluate what is the difficulty level of the work?

CSTA

| Grades | Identifier | Interim CSTA K-12 CS Standards | Framework Concept | Framework Practice |
|------------|------------|--|----------------------------|---|
| K-2 | 1A-A-5-2 | Construct programs, to accomplish a task or as a means of creative expression, which include sequencing, events, and simple loops, using a block-based visual programming language, both independently and collaboratively (e.g., pair programming). | Algorithms and Programs | Creating Computational Artifacts |
| K-2 | 1A-A-3-7 | Construct and execute algorithms (sets of step-by-step instructions) that include sequencing and simple loops to accomplish a task, both independently and collaboratively, with or without a computing device. | Algorithms and Programs | Recognizing and Defining Computational Problems |
| K-2 | 1A-A-6-8 | Analyze and debug (fix) an algorithm that includes sequencing and simple loops, with or without a computing device. | Algorithms and Programs | Testing and Refining |
| 3-5 | 1B-A-5-4 | Construct programs, in order to solve a problem or for creative expression, that include sequencing, events, loops, conditionals, parallelism, and variables, using a block-based visual programming language or text-based language, both independently and collaboratively (e.g., pair programming). | Algorithms and Programs | Creating Computational Artifacts |
| 3-5 | 1B-A-3-7 | Construct and execute an algorithm (set of step-by-step instructions) that includes sequencing, loops, and conditionals to accomplish a task, both independently and collaboratively, with or without a computing device. | Algorithms and Programs | Recognizing and Defining Computational Problems |
| 3-5 | 1B-A-6-8 | Analyze and debug (fix) an algorithm that includes sequencing, events, loops, conditionals, parallelism, and variables. | Algorithms and Programs | Testing and Refining |
| 6-8 | 2-A-5-6 | Develop programs, both independently and collaboratively, that include sequences with nested loops and multiple branches. [Clarification: At this level, students may use block based and/or text-based programming languages.] | Algorithms and Programming | Creating Computational Artifacts |

Conditionals

Overview

| | |
|-----------------------------|---|
| Concept | Conditionals - Make decisions in response to different conditions. Execute the action only if the conditional statement is true; otherwise, skip it or ignore it. |
| Explanations | Example: If it's raining, then you will use an umbrella. |
| Learning Objectives | 1. Understand the concept of Conditionals; 2. Write programs using the conditional blocks to accomplish challenges and create projects. |
| Teaching Preparation | 1. A box of conditionals (Step 3 - Lead-in Game) ; 2. A whiteboard and a whiteboard marker (or you can use a blackboard and chalks); 3. One Codey and a Bluetooth dongle (or the USB cable) per student but it's fine if 2 or 3 students share one set; 4. A computer per student, but it's fine if 2 or 3 students share a computer. And install mBlock 5 on each computer; 5. A self-review report per student. |
| Time | 120-240 min |

Teaching Procedure

Step 1: Review —— Loop

Review:

- What is a **Loop**?
- When will you need to run coding blocks repeatedly?
- How does the **Repeat** block help with your programs?

Loop means running one piece of programs over and over again. **Counting loops** refer to repeating specific times; **infinite loops** are to run one piece of programs endlessly. If we need to repeat one piece of programs specific times or endlessly, we can use the Repeat block to make our programs neat and effective.

Step 2: Explain New Knowledge - Conditionals

In today's lesson, we are to move on to a new concept, **Conditionals**. Conditional statements are common in everyday lives, and we often have to make decisions when we come

across them. For instance, if you want to buy a cup of tea, then you might walk out of your house first to see whether it's raining.

If it's raining, you will then take an umbrella to go; if it's not raining, you will go outside without an umbrella.

Conditionals refer to conditional statements. We make decisions in response to different situations. We execute the action only when the conditional statement is true; otherwise, skip it or ignore it. Ask students if they can think of any conditionals in daily lives.

We use conditionals blocks in programs to enable robots to make decisions in response to different situations. To get a Boolean value(true/false), we can drag a hexagon-shaped block to the hexagon dent in the **If/then** block. If the conditional statement is true, the computer will run the program; if the conditional statement is false, the computer will skip it and directly run the next piece of programs.



Step 3: Lead-in Game- Box of Conditionals

The box is full of paper strips on which there are all kinds of conditional statements. Have students randomly pick papers from the box and read out what it says. Students should make decisions and take actions as requested by the papers they pick. Ask students to write down the conditionals in advance and leave the actions part to teachers. In this way, we can avoid occasions where students might write down actions that they can not finish, like walking out of the classroom to leave school or jumping down from the 2nd floor. The chart attached at the bottom of this document are available for reference. Teachers can modify the actions or add new actions. Print the chart and cut the paper into strips along the dashed lines. Fold the strips in half and put them all into the box of conditionals.

Remind students one thing: write down conditionals that are easy to identify. This is to ensure that the game can go on smoothly. For instance, write conditionals like the following ones: if you have long hair; if you wear glasses; if you are in a black shirt; if your name has a letter A in it; if you were born in June; if someone puts up his or her hand; if someone claps his or her hands. If the

conditional statement on a paper strip is hard to identify (for instance, “if it rains tomorrow” or “if the amount of your hair is an odd number”), ignore that paper strip.

Have students come to the podium to pick papers from the box, or they can pass around the box on their seats.

Steps:

1. Pick a paper strip from the box and open it;
2. Read aloud what the paper says and make decisions. If the conditional statement is true, take action correspondingly; if the conditional statement is false, ignore it;
3. Fold the paper in half and put it back into the box. Go back to your seat or hand the box to the next student.

Suggestions:

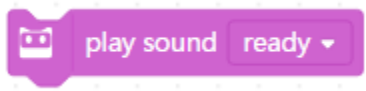
1. Teachers can join the students if there are not enough students;
2. If the paper strips outnumber students, then there is no need for students to put the conditionals papers they’ve used back into the box.

Step 4: Tasks

Have students accomplish the tasks in pairs or by themselves. Each step has a square box. **Tell them to tick the square box when they finish the step (The Tasks Cards are included at the bottom of this document. Print them).**

Task 1: Start off when the flag is waving

Introduce the story background and the task requirements to students. Then hand out the tasks cards.

| |
|---|
| Task 1— Start off when the flag is waving |
| The game is starting now. If Codey Rocky sees the green flag waving, it will run forward at its top speed. |
| <input type="checkbox"/> When button A is pressed, Codey Rocky is getting ready at the starting line (play the sound ready). |
|  |

- ☐ If the color sensor detects **green**, then Codey Rocky will **move forward** at its **top** speed.



- ☐ After the computer assesses the conditional statement and make a decision, the RGB LED will light up with red color.



- ☐ **Challenge: Customize coding blocks to make Codey Rocky show expressions and make sounds when it is running.**

Teachers demonstrate how to write the following programs. Remind students:

1. The “**if/then**” block includes a dent, so we need to interlock a conditional block (for instance, “if it is green?”) to the dent;
2. Remind students to put the the block “**RGB LED lights on with color red**” outside the conditional block to help end the programs.



Students write programs as demonstrated by the teachers.

Discussions:

1. If we want Codey Rocky to move backward when detecting red, how to rewrite the programs?

Task 2: Avoid the Obstacle

Have students finish the task independently.

Task 2—Avoid the Obstacle

When meeting an obstacle, Codey Rocky will avoid it and keep moving forward.

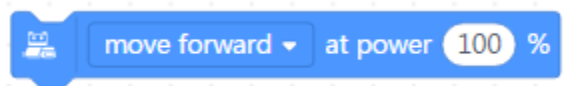
- ☐ Place an obstacle before Codey Rocky.



- ☐ When **the button A** is pressed, if it **detects an obstacle**, Codey Rocky will turn right by 90 degrees, move forward, turn left by 90 degrees and **move forward** at a rapid speed.



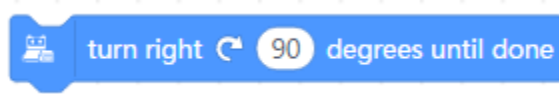
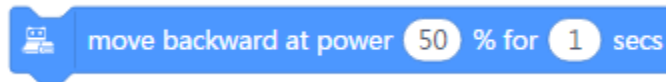
- ☐ If Codey Rocky doesn't detect any obstacles, it will move forward at its top speed.



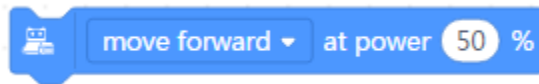
- ☐ **Challenges** 1. Take a detour to avoid the obstacle, that is to say, turn left and then turn right.
2. Let Codey Rocky show expressions and make sounds when it's moving.

Tips:

1. Make sure you put the **“move forward”** block and the **“ turn left/right”** block including time limits in the conditional block.



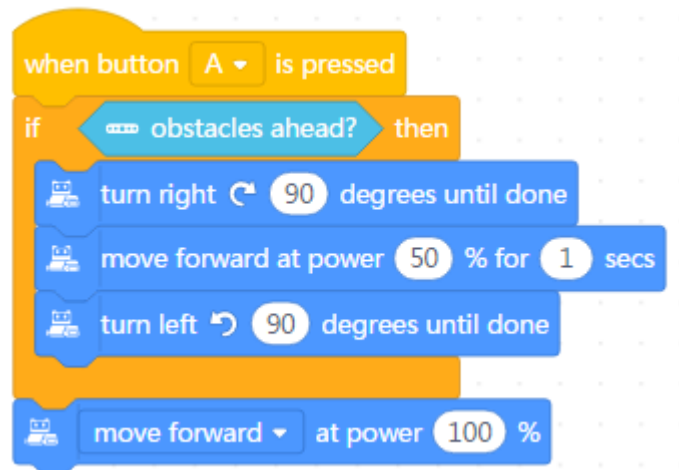
Don't use the following block.



The first two blocks will stop when the time is up and then move on to run the next block. However, the third block will keep running with other blocks unless you use another action block in the programs. In this case, Codey Rocky will keep moving forward at power 50% until someone powers it off.

2. Remind students to put the block “**move forward at power 100%**” outside the conditional block. After Codey Rocky turns left, the piece of programs surrounded by the conditional block will come to an end, and the outer block starts to run. The outer block has no time limits.

Sample Programs:



Discussions:

1. What will happen if you encircle all the programs with a **forever** block?
2. What will happen if you use a **move forward** block **without** time limits instead?
3. What will happen if you put a **move forward** block **including** time limits outside the conditional block instead?

Task 3: Service Station

Have students finish the task independently.

Teachers remind students: In some cases, we might use multiple conditional blocks to accomplish a task.

Task 3— Service Station

During the race, Codey Rocky needs to get fuel and repairs at service stations for plenty of times. However, the ways it pulls into the station might vary from time to time. Write programs to ensure that Codey Rocky is able to re-enter the racetrack whatever.

- ☐ Use books or other objects to encircle your Codey Rocky as illustrated below. This is to simulate a scene where Codey Rocky stops at a service station. Codey Rocky pulls into the station

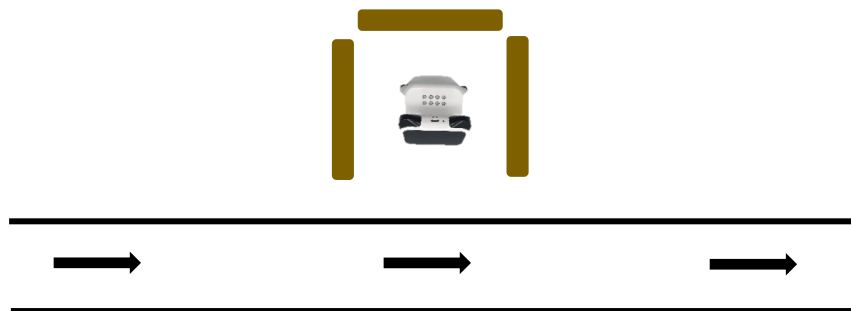


ending up in random positions as shown above.

- ☐ Write programs: when **the button A** is pressed, if Codey Rocky detects **an obstacle** ahead, then it will keep making turns until it finds the exit. You might use more than one conditional blocks.



- ☐ Use chalks or black sticky tapes to make a racetrack in front of the Exist.

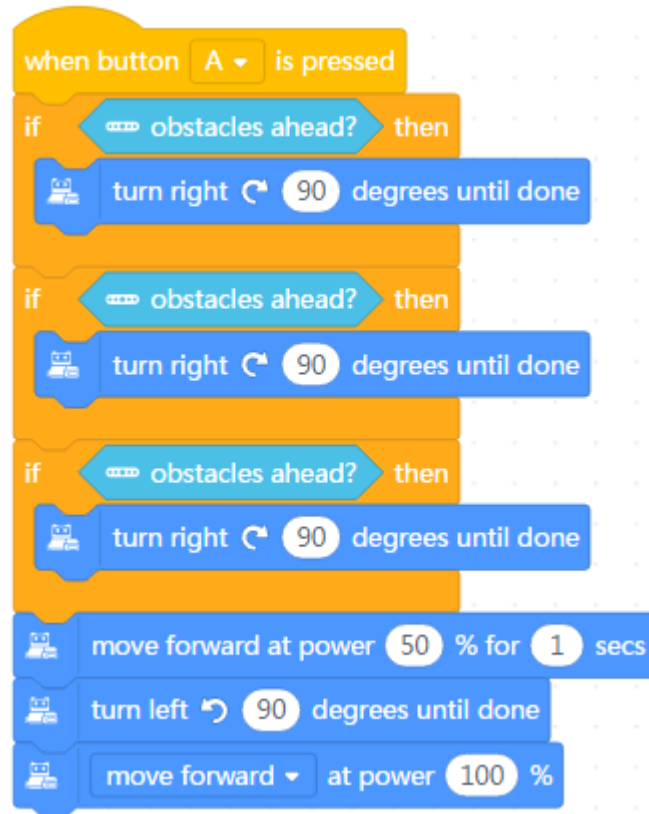


- ☐ When Codey Rocky finds the Exit, it will first **move forward** to leave the garage, **turn left** and then move forward **at its top speed**.
- ☐ **Challenge:** Add expressions, sounds, and lights.

Tips:

1. You might use more than one conditional blocks.
2. You can set the blocks to make Codey Rocky enter the tunnel at different powers under the drive of various events.

Sample Programs:



Discussions:

1. How to use the fewest blocks to make Codey Rocky find the Exit? Any repetitive blocks did I use?

Task 4: Tunnel

Instructions: The hexagon blocks we used in the previous tasks can directly judge whether the conditional statement is true or false, for instance, whether there is an obstacle or not, whether it is red or not. Besides, we use the comparative Operator block to evaluate two values. For example, the block “**when light intensity < 20**” helps us assess whether the light intensity exceeds 20. If the light intensity is below 20, then the result will be true. Otherwise, the result will be false.



The hexagon blocks return Boolean values (with only two possible values: true or false). A Boolean variable has only two values, 1 if true and 0 if false. If the conditional statement is satisfied, then it will return the value, true. Otherwise, it will return the value, false. Therefore, we also refer to the hexagon blocks as Boolean blocks.

Teachers Demonstrate:

1. Drag the following comparison block from the Operators category;



2. Drag the “**light intensity block**” into the left dent;
3. Input a value in the right dent of the comparison block;
4. Finish the following task.

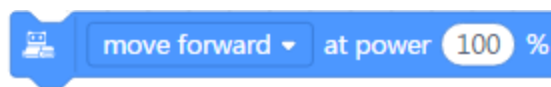
Task 4— Tunnel

When entering a dark tunnel, Codey Rocky will turn on its light and slow its speed.

- ☐ Stick a black paper to the light sensor of Codey Rocky.

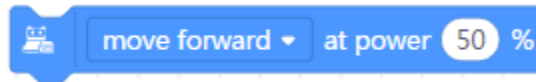


- ☐ When **the button A** is pressed, Codey Rocky will move forward at its top speed.



- ☐ **If Codey** detects that the light intensity is **below 20**, Codey Rocky will turn on its **white RGB indicator** and drive at a lower speed.





Tips:

1. The light sensor is the black dot at the right bottom of Codey.

Sample Programs:



Challenges for Students:

1. By following the teachers, students write programs to let Codey make sounds and show expressions in a dark environment;
2. How to make Codey Rocky more sensitive to a dark environment? How to manipulate Codey Rocky to turn on the RGB indicator instantly the light intensity gets low?

Task 5: Volume

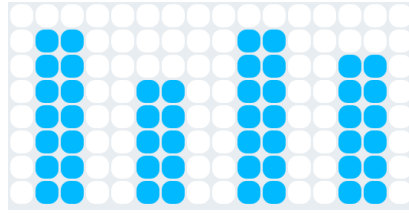
Students finish the task independently.

Task 5—Volume

Codey Rocky wins the game and people are applauding. The volume bar on the screen will change in response to the applause loudness.

- ☐ When Codey Rocky starts up, if the detected applause loudness exceeds 20, the volume bar will reach its top height.





- ❑ If the detected loudness falls between 10 and 20, the volume bar will go down. Now, we need to use the “**and**” block.



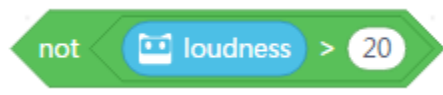
- ❑ If the applause loudness is below 10, the volume bar will fall to its lowest height.
- ❑ Encircle all the programs with a “**forever**” block to make Codey Rocky keep detecting loudness of sounds around it all the time.



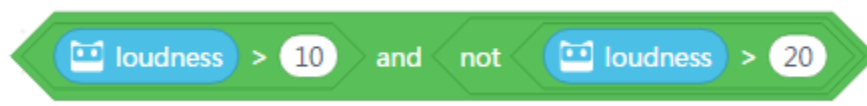
- ❑ **Challenge:** Control the RGB LED indicator to change its color in response to the loudness.

Tips:

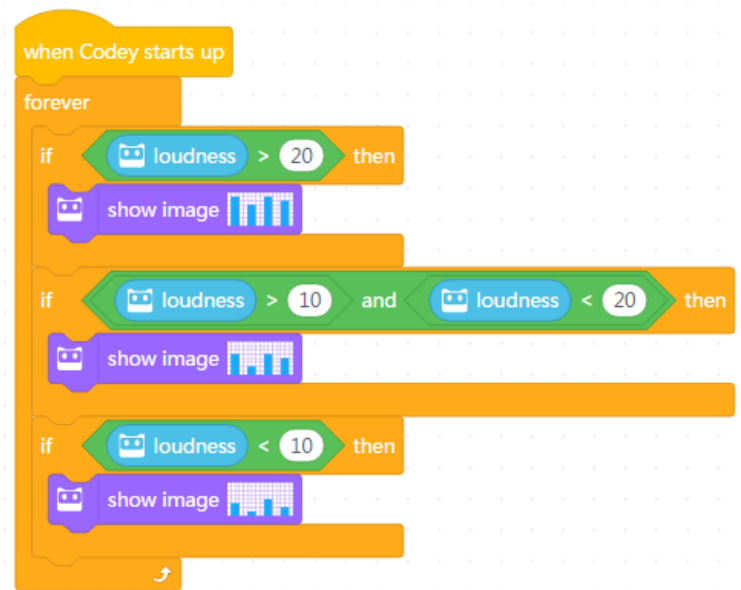
1. The “**and**” block means that both of the two conditional statements should be true;
2. And there is another occasion: if the loudness equals 20 or 10. We need to use the “**not**” block if we want to add this situation. If we want to make Codey Rocky assess whether the loudness is less than or equals 20, we will use the following combination of blocks:



If we want to evaluate whether the loudness falls among a range of $10 < x \leq 20$, write programs using the following combination of blocks:



Sample Programs:



Discussion:

1. How to define different ranges: 1) >20 ; 2) $10 < x \leq 20$; 3) ≤ 10 ;
2. How to subdivide the ranges to make the volume bar more sensitive to the loudness? When the music starts, the volume bar will react to the music.

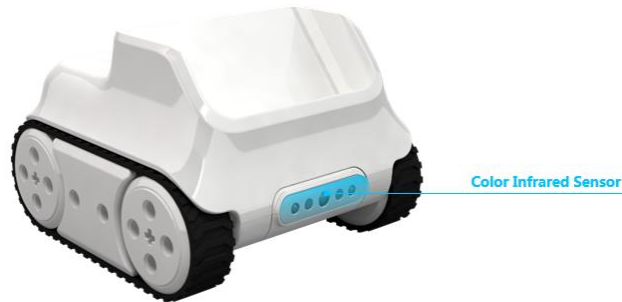
Task 6: Follow

Teachers give students more details about the “if...then...else” block. Then have students finish the task independently.

Task 6 — Follow

Imagine Codey is a cute cat. When you approach, it will follow you; otherwise, it will stay there waiting for you.

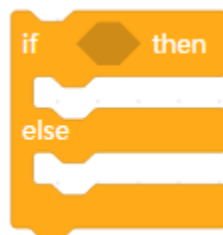
- The IR color sensor detects the intensity of reflected IR lights. The closer an object is to Codey Rocky, the stronger the light intensity will be.



- When Codey Rocky starts up, the LED matrix will display the reflected infrared light intensity value.



- Surround the “show” block with a “forever” block to make Codey Rocky monitor light intensity in real time.
- In the “forever” block, programs first show the reflected infrared light intensity, then make assessments. If the intensity exceeds a specific value (Customized), then Codey Rocky moves forward at its top speed, otherwise it will stop there not moving. We need to use the “if...then...else” block and the Operators block.



- **Challenge:** If the loudness exceeds one specific value, then Codey Rocky moves forward or turns left at the greatest power.

Tips:

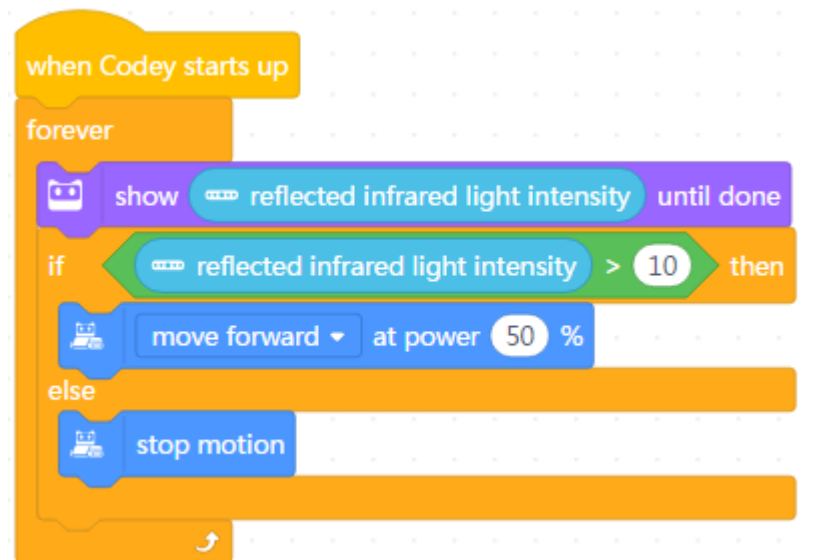
1. When using the **“show”** block to display variables changes on the LED matrix, we need to avoid using blocks including time limits, such as **“move forward for 1 sec”**, **“turn left until done”**, **“play sound until done”**. This is because these blocks will lead to the failure of Codey Rocky to monitor data in real time.

2. Through monitoring and showing the reflected infrared light intensity in real time, we can customize the conditionals. That is to say, only when the reflected infrared light intensity exceeds a certain value will Codey Rocky move forward.

3. Make sure all the programs are surrounded by the **“forever”** block;

4. Make the IR color sensor face forward to detect obstacles.

Sample Programs:



Discussion:

1. What will happen if I make the IR color sensor face down?
2. I want Codey Rocky (cat) to do the following things: keeps searching around until it finds its master and then follows the master. How to rewrite programs?

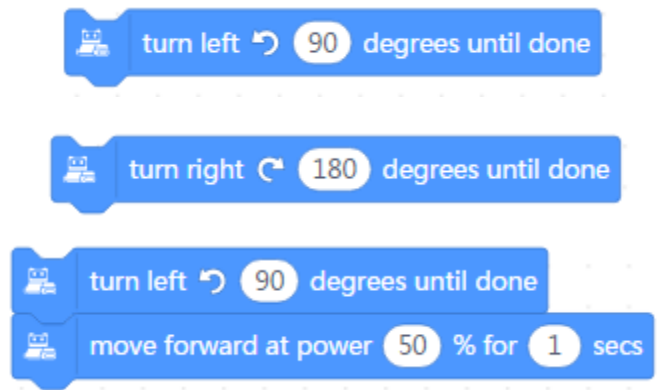
Task 7: Find the Master

Task 7: Find the Master

Rather than stay there waiting for the master, the cat (Codey Rocky) prefers to walk around searching.

☐ If Codey Rocky finds its master (the reflected infrared light intensity exceeds a specific value), it will move forward.

☐ If Codey Rocky fails to find its master, it will first keep turning left, right and left, and then keep moving forward in the original direction to search.



☐ Encircle all the programs with the **forever** block.

☐ **Challenge:** Write programs to make Codey Rocky show expressions and make sounds when it finds the master.

Tips:

1. To ensure that Codey Rocky accurately detects the master, we need to keep our hand or objects level with the IR color sensor;

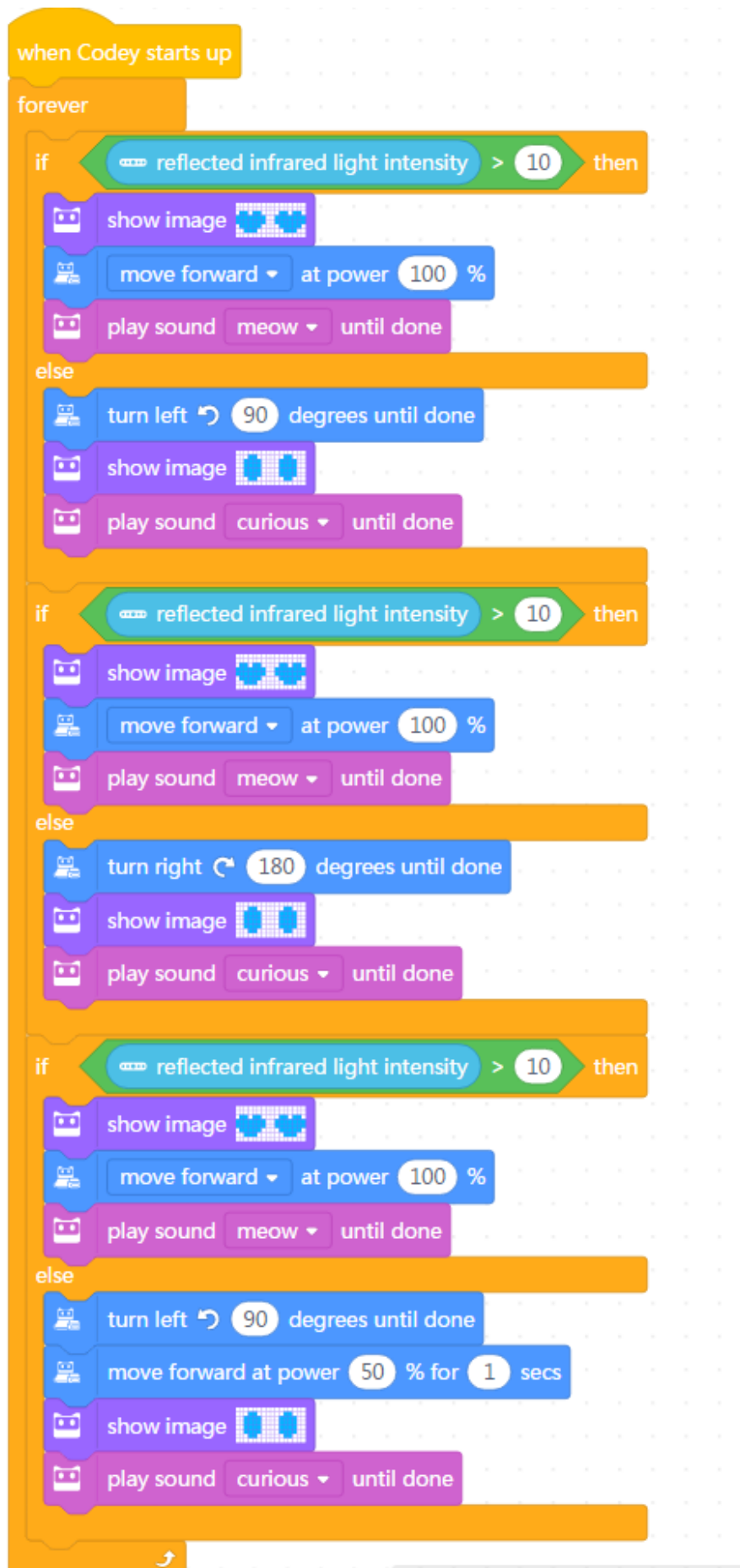
2. Remember to make Codey Rocky move forward in the original direction after it finishes searching around for its master;

3. If Codey Rocky succeeds in finding its master (an obstacle), we should use the “**move forward**” block without time limits; otherwise, we need to use the blocks including time limits, such as “**turn left until done**”, “**turn right until done**” and “**move forward for 1 sec**”.

4. To finish the task, we have to use 3 “**if...then...else**” blocks. One block can just manipulate Codey Rocky to detect the master (an obstacle) only when it’s moving forward. However, what we need is that the cat (Codey Rocky) detects three times respectively when it turns left, turns right and moves forward.

5. We can shorten the time of moving forward in case the cat (Codey Rocky) misses its master (an obstacle).

Sample Programs:



Discussion:

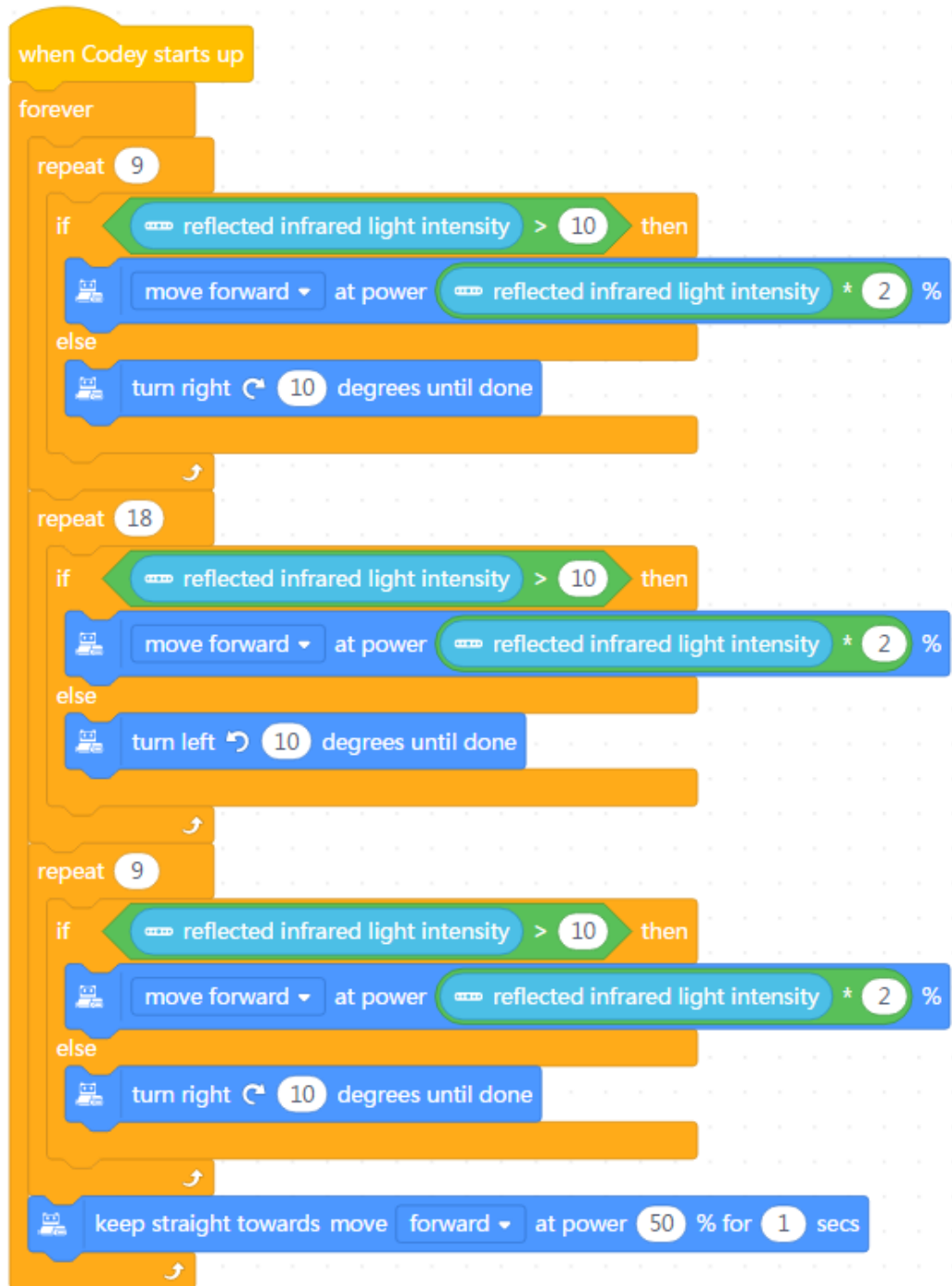
1. If the master (an obstacle) is dressed in black, is the cat (Codey Rocky) still able to detect it?

Why?

2. I want the cat to detect whether there is an obstacle ahead each time it turns by 10 degrees.

How to achieve this by adding the “repeat” blocks?

Sample Solutions:



Self-review Report

Name:

Age:



- Answer the following questions to record your learning outcomes:

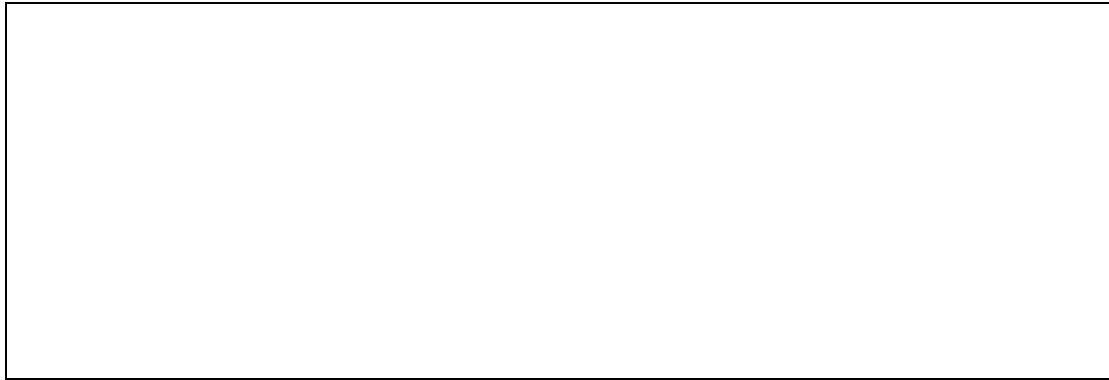
Describe what you've learned with one or two sentences.



Describe what you like most and least about this class session briefly.

| |
|-------------------|
| What I like most |
| What I like least |

Draw a conditional that happens in your life.



You can paint how you feel about this class session at the upper right corner of the self-review report.



CSTA

| Grades | Identifier | Interim CSTA K-12 CS Standards | Framework Concept | Framework Practice |
|------------|------------|--|-------------------------|----------------------------------|
| K-2 | 1A-A-5-2 | Construct programs, to accomplish a task or as a means of creative expression, which include sequencing, events, and simple loops, using a block-based visual programming language, both independently and collaboratively (e.g., pair programming). | Algorithms and Programs | Creating Computational Artifacts |
| 3-5 | 1B-A-5-4 | Construct programs, in order to solve a problem or for creative expression, that include sequencing, events, loops, conditionals, parallelism, and variables, using a block-based visual programming language or text-based language, both independently and collaboratively (e.g., pair programming). | Algorithms and Programs | Creating Computational Artifacts |
| 3-5 | 1B-A-6-8 | Analyze and debug (fix) an algorithm that includes sequencing, events, loops, conditionals, parallelism, and variables. | Algorithms and Programs | Testing and Refining |

If (_____) ,
then press the light button in the classroom.

If (_____) ,
then touch any green object in the classroom.

If (_____) ,
then squat jump three times.

If (_____) ,
then draw a fish on the blackboard.

If (_____) ,
then stretch one hand through underneath of the other hand to hold your nose and turn
around three times with that posture.

If (_____) ,
then say “I’ m great.”

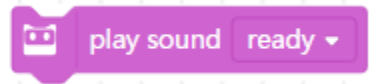
If (_____) ,
then cross arms over the chest and smiles at the teacher.

If (_____) ,
then open a book and turn to page 20.

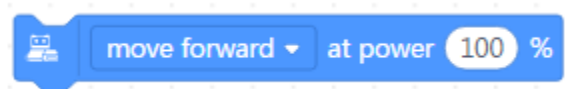
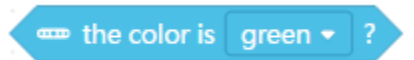
Task 1— Start off when the flag is waving

The game is starting now. If Codey Rocky sees the green flag waving, it will run forward at its top speed.

- ☐ When **button A** is pressed, Codey Rocky is getting ready at the starting line (play the sound ready).



- ☐ If the color sensor detects **green**, then Codey Rocky will **move forward** at its **top** speed.



- ☐ After the computer assesses the conditional statement and make a decision, the RGB LED will light up with red color.



- ☐ **Challenge:** Customize coding blocks to make Codey Rocky show expressions and make sounds when it is running.

Task 2—Avoid the Obstacle

When meeting an obstacle, Codey Rocky will avoid it and keep moving forward.

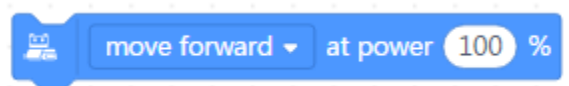
- ☐ Place an obstacle before Codey Rocky.



- ☐ When **the button A** is pressed, if it **detects an obstacle**, Codey Rocky will turn right by 90 degrees, move forward, turn left by 90 degrees and **move forward** at a rapid speed.



- ☐ If Codey Rocky doesn't detect any obstacles, it will move forward at its top speed.



- ☐ **Challenges** 1. Take a detour to avoid the obstacle, that is to say, turn left and then turn right.
2. Let Codey Rocky show expressions and make sounds when it's moving.

Task 3— Service Station

During the race, Codey Rocky needs to get fuel and repairs at service stations for plenty of times. However, the ways it pulls into the station might vary from time to time. Write programs to ensure that Codey Rocky is able to re-enter the racetrack whatsoever.

- ☐ Use books or other objects to encircle your Codey Rocky as illustrated below. This is to simulate a scene where Codey Rocky stops at a service station. Codey Rocky pulls into the station

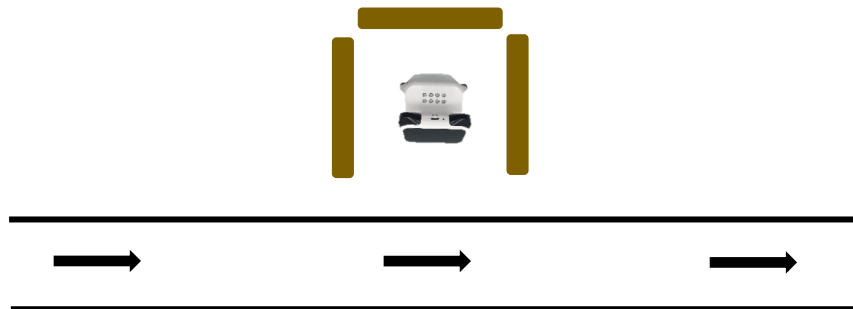


ending up in random positions as shown above.

- ☐ Write programs: when **the button A** is pressed, if Codey Rocky detects **an obstacle** ahead, then it will keep making turns until it finds the exit. You might use more than one conditional blocks.



- ☐ Use chalks or black sticky tapes to make a racetrack in front of the Exist.



- ☐ When Codey Rocky finds the Exit, it will first **move forward** to leave the garage, **turn left** and then move forward **at its top speed**.
- ☐ **Challenge:** Add expressions, sounds, and lights.

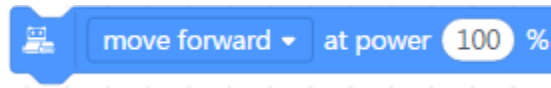
Task 4— Tunnel

When entering a dark tunnel, Codey Rocky will turn on its light and slow its speed.

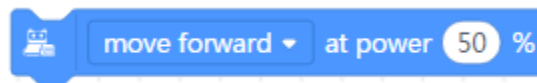
- ☐ Stick a black paper to the light sensor of Codey Rocky.



- ☐ When **the button A** is pressed, Codey Rocky will move forward at its top speed.



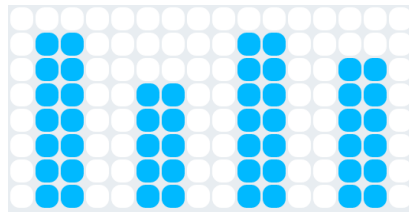
- ☐ **If Codey** detects that the light intensity is **below 20**, Codey Rocky will turn on its **white RGB indicator** and drive at a lower speed.



Task 5—Volume

Codey Rocky wins the game and people are applauding. The volume bar on the screen will change in response to the applause loudness.

- ☐ When Codey Rocky starts up, if the detected applause loudness exceeds 20, the volume bar will reach its top height.



- ☐ If the detected loudness falls between 10 and 20, the volume bar will go down. Now, we need to use the “**and**” block.



- ☐ If the applause loudness is below 10, the volume bar will fall to its lowest height.
- ☐ Encircle all the programs with a “**forever**” block to make Codey Rocky keep detecting loudness of sounds around it all the time.

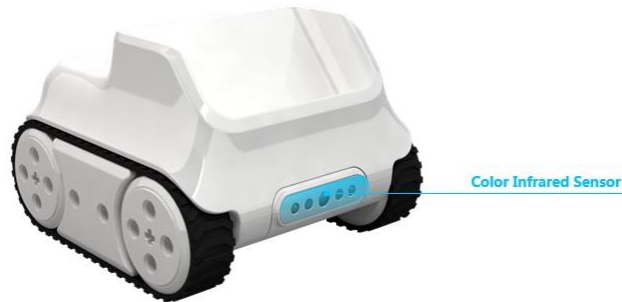


- ☐ **Challenge:** Control the RGB LED indicator to change its color in response to the loudness.

Task 6 — Follow

Imagine Codey is a cute cat. When you approach, it will follow you; otherwise, it will stay there waiting for you.

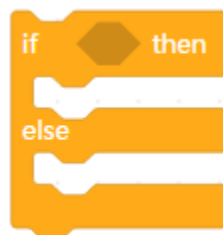
- The IR color sensor detects the intensity of reflected IR lights. The closer an object is to Codey Rocky, the stronger the light intensity will be.



- When Codey Rocky starts up, the LED matrix will display the reflected infrared light intensity value.



- Surround the “show” block with a “forever” block to make Codey Rocky monitor light intensity in real time.
- In the “forever” block, programs first show the reflected infrared light intensity, then make assessments. If the intensity exceeds a specific value (Customized), then Codey Rocky moves forward at its top speed, otherwise it will stop there not moving. We need to use the “if...then...else” block and the Operators block.

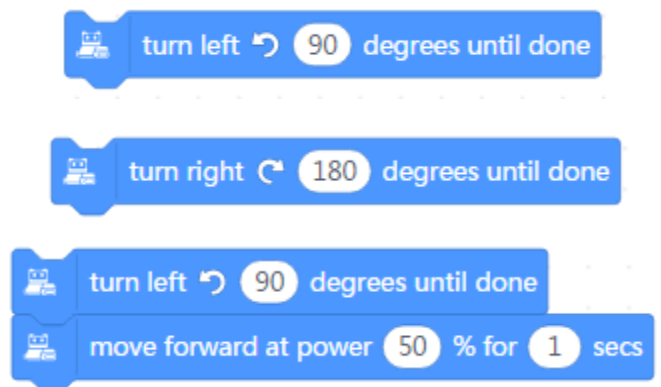


- **Challenge:** If the loudness exceeds one specific value, then Codey Rocky moves forward or turns left at the greatest power.

Task 7: Find the Master

Rather than stay there waiting for the master, the cat (Codey Rocky) prefers to walk around searching.

- ☐ If Codey Rocky finds its master (the reflected infrared light intensity exceeds a specific value), it will move forward.
- ☐ If Codey Rocky fails to find its master, it will first keep turning left, right and left, and then keep moving forward in the original direction to search.



- ☐ Encircle all the programs with the **forever** block.
- ☐ **Challenge:** Write programs to make Codey Rocky show expressions and make sounds when it finds the master.

Functions

Overview

| | |
|-----------------------------|--|
| Concept | Function —— A group of coding instructions that can be called over and over again |
| Explanations | Example: Wash hair |
| Learning Objectives | 1. Understand the concept of Functions; 2. Create functions and call the functions to perform tasks. |
| Teaching Preparation | 1. Paper strips of functions (step 3 - Lead-in game) ; 2. A whiteboard and a whiteboard marker (or you can use a blackboard and chalks); 3. One Codey Rocky and a Bluetooth dongle (or a USB cable) per student but it's fine if 2 or 3 students share one set; 4. A computer per student, but it's fine if 2 or 3 students share a computer. And install mBlock 5 on each computer; 5. A self-review report form per student. |
| Time | 120-240 min |

Teaching Procedure

Step 1: Review——Conditionals

Review:

- What is a conditional block?
- How did we use the conditional blocks to assess conditions in the previous lesson? Ask

students if they could come up with any of those conditions.

The conditional blocks assess various conditions and decide whether the conditional statements are true or not. If the conditional statement is true, then the computer will run the piece of code. Otherwise, it will skip it or ignore it. In the previous lesson, we used the conditional blocks to assess the following situations: 1) whether the color detected by the color sensor is green 2) whether there are obstacles ahead 3) whether the light intensity exceeds 20 4) whether the sound loudness is less than 20 5) whether the reflected infrared light intensity exceeds 10.

Step 2: Explain New Knowledge —— Functions

Today, we are to have a look at **Functions**. In daily life, we often do some things more than once, like washing hair. In most cases, the three steps to wash our hair include: take some shampoo, massage the hair to make the shampoo foam, and rinse the hair with warm water. One of your friends comes to ask you out, but you have to wash hair first. Then you might say, “I’m about to take shampoo, massage the hair to make it foam, and rinse the hair. Allow me some time.” Mom is shouting the breakfast is ready, while you are washing your hair. You then repeat the words, “I’m taking shampoo, massaging the hair to make it foam, and rinsing the hair.” Being caught in the rain, you are murmuring to yourself, “I’ve got to take shampoo, massage my hair to make it foam and rinse the hair.”

Actually, we don’t describe each step in everyday life, instead of which we often give the group of steps one simple name. We use the name to represent the whole sequence of steps and call the name when referring to the steps. For example, we name the sequence of actions above as *wash hair* . When your friend comes to ask you out, you say this time, “I am about to wash hair. Allow me some time. “When your mom is shouting the breakfast is ready, you say, “I’m washing hair, wait.” Being wet in the rain, you murmur to yourself, “I’ve got to wash hair.” Using a simple name to refer to a sequence of steps makes conversations efficient and enjoyable.

In computer programming, we use a function to name a piece of commands and can call the name to run the commands if necessary at any time. However, before that, we have to create a function. The first step is to consider a proper name for a function. Then, we need to input commands under the name of the function to define the function. Here, one thing we must keep in mind is that the function name should be easy to understand, which otherwise could lead to some trouble when we call the function. Take washing hair as an example, if we give the sequence of steps a function name *have dinner* , it could be misleading. How does this sound? When our friends come to ask us out, we tell them “I’m having dinner” while we are washing hair in the bathroom instead. It could be confusing.

After defining the function, we can call the function to run a unit of commands that are stored in the function. For instance, wash hair and then go outside; wash hair and then have breakfast; wet in the rain and then wash hair. Thanks to functions, our programs become neat and easy to understand. We no longer need to repeat the same part of code in programs multiple times.

Step 3: Lead-in Game—— The Morning Functions

Game Steps and Teaching Preparation:

1. Teachers print the first chart appendix in the document, **The Morning Functions**. Cut the print into pieces along the dash lines. These are called paper strips of functions;
2. Teachers put down on the whiteboard “Event -When the alarm clock goes off”;
3. Each student pick one paper strip randomly from teachers;
4. Teachers articulate the game rules: “Everyone has a piece of paper strip in hand, and each of them represents a group of actions. Give the group of actions a name yourself; I will later call the functions under the event of “When the alarm clock goes off”; When the function of yours is called, you need to read aloud the detailed steps and act them out;
5. Teachers write down the functions names on the blackboard;
6. After putting down all the function names on the whiteboard, teachers need to ask students what the order is for the functions to be called. Then write down the function names under the event of “When the alarm clock goes off” in the order;
7. Ask students to think about the functions that are not called or called multiple times. Figure out the reasons and complete the programs.

The following chart demonstrates sample function names:

| | | |
|--|--|---|
| <u>Oversleep</u> cover head with the quilt; wait for 5 seconds; uncover the quilt; stretch out one hand; turn off the alarm clock. | <u>Have a look at the time</u> take up the alarm clock; have a look at the time; put down the clock. | <u>Get up</u> stretch yourself; uncover the quilt; sit up. |
| <u>Put on slippers</u> put on the slipper to the left foot; put on the slipper to the right foot; stand up. | <u>Walk</u> lift the left foot; stretch forward the left foot and step on the ground; lift the right foot; stretch forward the right foot and step on the ground. | <u>Brush teeth</u> take the toothbrush; squeeze the toothpaste; rinse the mouth; brush the teeth all around; rinse the mouth. |
| <u>Wash up</u> turn on the tap; cup the water with hands; wash up with the water; turn off the tap; | <u>Take off the coat</u> If it’s a T-shirt, take it off from top to bottom; If the coat has buttons, unbutton it first, then the left | <u>Take off pants</u> pull down the pants; the left leg comes out of the pants; |

| | | |
|---|---|--|
| dry your face with the towel | arm comes out of the left sleeve, next the right arm comes out of the right sleeve. | the right leg comes out of the pants. |
| <u>Take clothes off</u> take off the coat; take off pants. | <u>Put on the coat</u> get the left arm into the left sleeve; get the right arm into the right sleeve; if it's a T-shirt, get your head into the shirt; if the coat has buttons, fasten the buttons. | <u>Put on pants</u> get the left leg into the pants; get the right leg into the pants. |
| <u>Dress up</u> put on the pants; put on the coat. | <u>Carry schoolbag</u> get the right arm through one strap; get the left arm through the other strap; shoulder the schoolbag. | <u>Lace the shoes</u> make a circle of the shoelace with the left hand; make a circle of the shoelace with the right hand; tie the two circles |
| <u>Put on shoes</u> get the left feet into the left shoe; get the right feet into the right shoe; lace the left shoe. | | |

Suggestions:

1. The functions of “**Take clothes off**”, “**Put on clothes**”, and “**Put on shoes**” include nested functions, so it might be a little bit challenging for students to understand the concept of nested functions. In consideration of this, we could skip the nested functions. That is to say, we only use the function paper strip “**Dress up**” in the game, ignoring the paper strips of “**Put on the coat**” and “**Put on pants**”. However, if students are old enough to understand the concept of nested functions, then it is fine to introduce the concept to students. A nested function is a function that is defined within another function;

2. In some cases, the function names might be inaccurate. For example, students name the group of actions “lift the left foot; stretch forward the left foot and step on ground; lift the right foot and step on ground” as “Gymnastics”. Have students read the steps aloud, act them out and rename it with the help of partners. Next, try to call the function again in the program.

3. Sometimes, the function names can be funny. For example, give the group of actions “cover head with the quilt; wait for 5 seconds; uncover the quilt; stretch out one hand; turn off the alarm clock” an amusing name like “Annoying”. Don’t limit the name to be “Oversleep”.

Rather, encourage students to bring whimsy into the class;

4. If students are not willing to act out the steps or the time is limited, skip the acting out part. Just let them read out the steps.

Game Wrap-up:

Three steps to use a function in a program:

1. Create a function and name it;
2. Define the function;
3. Call the function in a program.

Step 4: Tasks

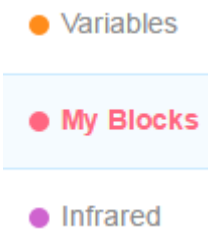
Students work on the following tasks in pairs or on their own. Tell them to tick the square box when they finish the step (The Tasks Cards are included at the end of this document. Print them).

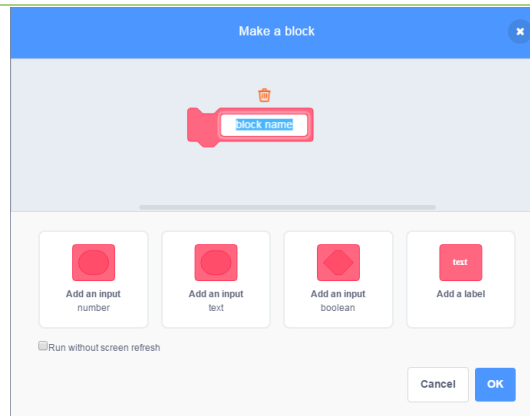
Guided by Teachers: Open the mBlock 5, click **My Blocks**, and select **Make a Block**.

Navigate students to accomplish the task 1.

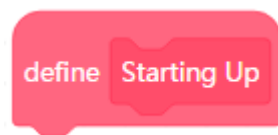
Task 1: Starting up Function

Create a starting up function for Codey Rocky. Each time the Codey Rocky is powered on, the function will be executed.

| Task 1-Starting up Function | |
|---|---|
| Create a starting up function for Codey Rocky | |
| <input type="checkbox"/> | Click My Blocks at the category bar and select Make a Block . |
|  | |
| <input type="checkbox"/> | Click Make a Block . Create a function and give it a name. |

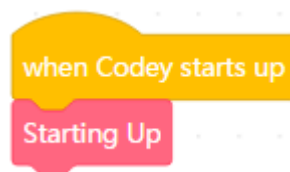


- Then, the **Define** block for the function will appear in the Scripts area.

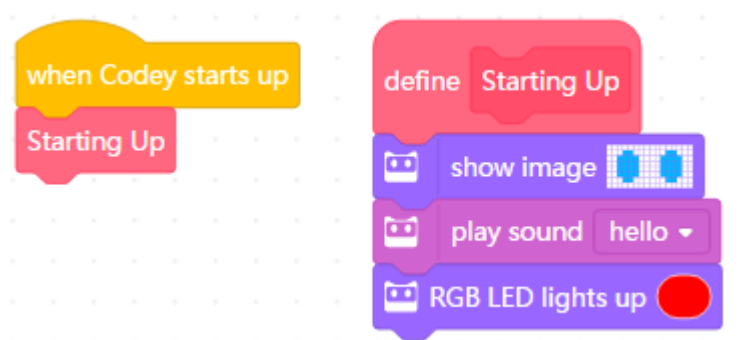


- What programs do you want to run each time Codey Rocky starts up? Design programs under the “**Define start up**” block.

- After they finish defining the function, have students attach the “**Start up**” block to the bottom of the event block “**When Codey Rocky starts up**”.



Sample Programs:

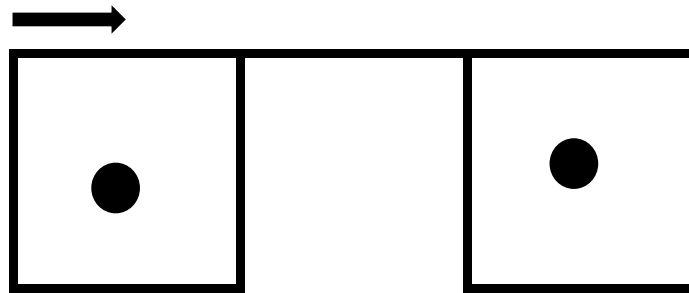


Task 2: Patrol the 1st Floor

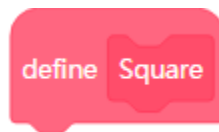
Task 2- Patrol the 1st Floor

Imagine Codey Rocky is a security guard. It has to patrol the passages in the building to keep an eye on properties. Now, it's patrolling the 1st floor.

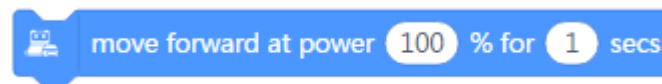
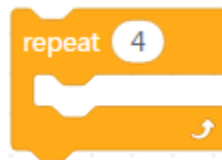
- Design programs to make Codey Rocky patrol the passage by following the black lines as illustrated below.



- Create a function and name it as Square.



- You might need to attach the following blocks to the bottom of the function.



- Under the event block “**When button A pressed**”, call the function Square two times.
- **Challenge:** Add expressions, sounds and lights.

Tips:

1. Before handing out the task cards, teachers need to explain more about the task. Students are likely to miss out details because they seldom enjoy reading texts.

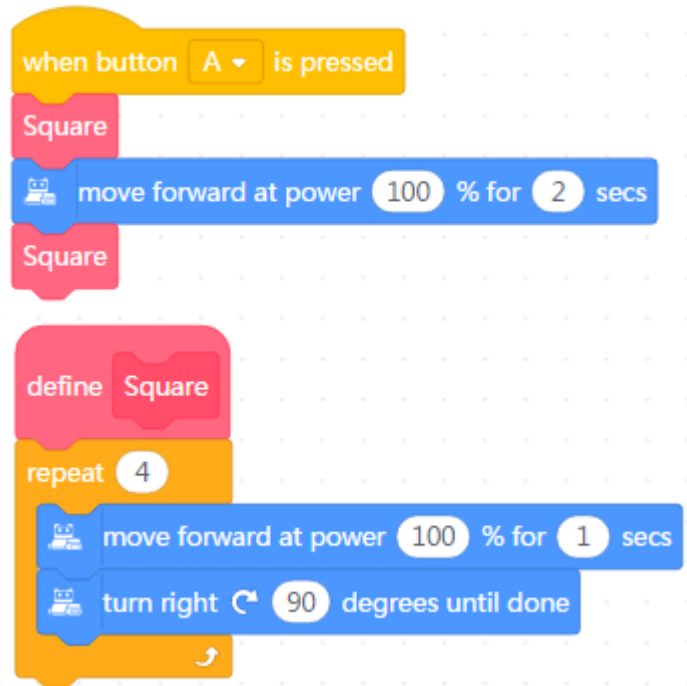
2. There is a chance that Codey Rocky might fail to rotate by accurately 90 degrees. Give Codey Rocky tolerance. If it can patrol in the roughly same route as illustrated in the picture above, then it's fine;

3. The longer distance Codey Rocky walks, the more likely it is to deviate from the original direction. Therefore, it is recommended that you set the walking time to be 1 second or 2 seconds;

starting point;

5. Since there are two squares, we should call the square function twice at least in the program.

Sample Programs:

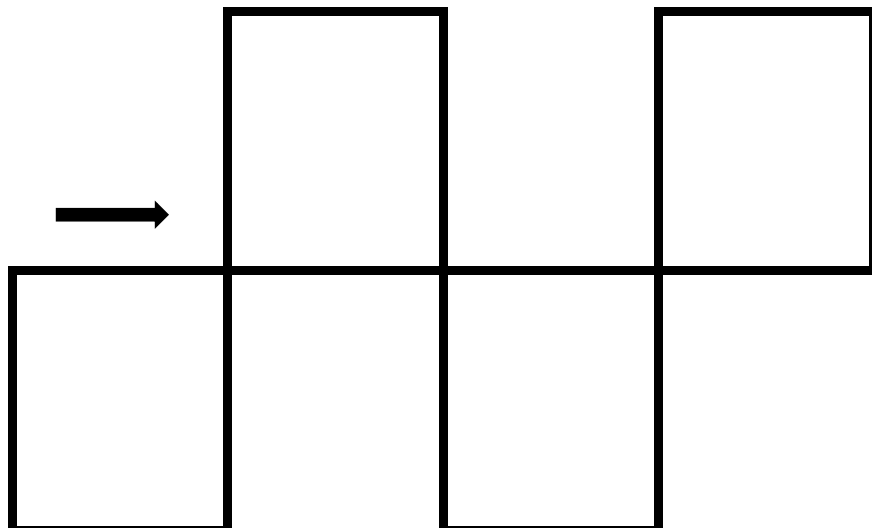


Task 3: Patrol the 2nd Floor

Task 3-Patrol the 2nd Floor

Codey Rocky comes to the 2nd floor. There are more rooms and the passages are more complex than the 1st floor.

- Design programs to make Codey Rocky patrol in the black route as illustrated below.



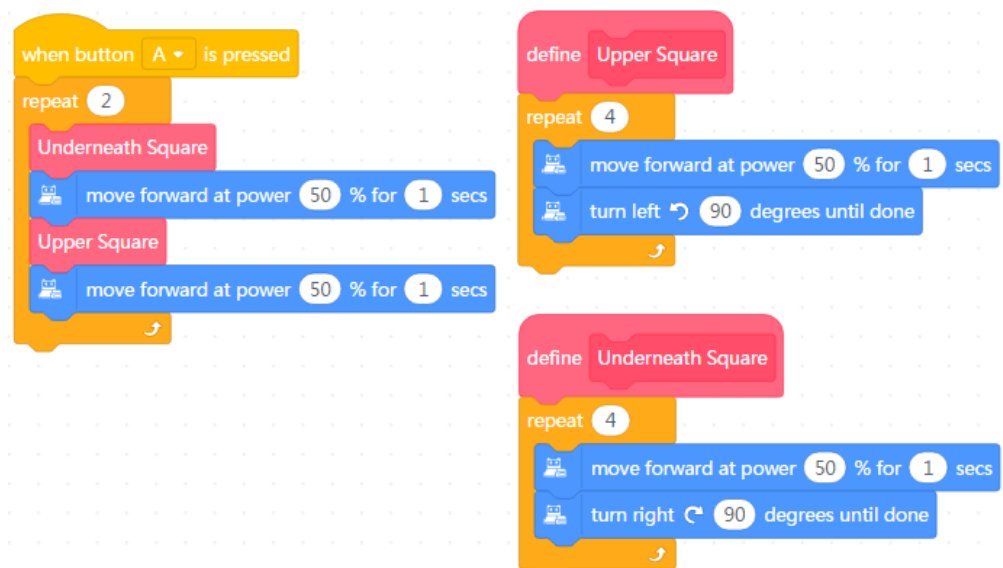
- ☐ You need to create two functions, Upper square, Underneath square.
- ☐ Under the event block “**When button A pressed**”, call the two functions in the program.
Use the **Repeat** block to make your program neat.
- ☐ **Challenge:** Add expressions, sounds and lights.

Tips:

1. Remind students that they should create two functions, like Upper square, Underneath square;
2. There are several ways to make Codey Rocky take the route as shown above. Have students design programs on their own first; then teachers use the following pseudocode to inspire students to complete the task:

| | |
|---|--|
| <p>Underneath square:</p> <div> <div>move forward 1 sec</div> <div>turn right by 90°</div> </div> <p>Repeat four times</p> | <p>Upper square:</p> <div> <div>move forward 1 sec</div> <div>turn left by 90°</div> </div> <p>Repeat four times</p> |
| <p>When button A pressed:</p> <div> <div>underneath square</div> <div>move forward 1 sec</div> <div>upper square</div> <div>move forward 1 sec</div> </div> <p>Repeat two times</p> | |

Sample Programs:

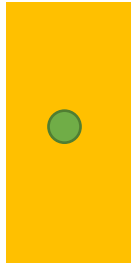


Task 4: Dancing Turtle

Task 4- Dancing Turtle

We are at a dancing party. Animals are going to show their dance moves one by one. The first one is a little turtle.

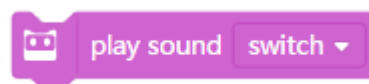
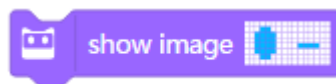
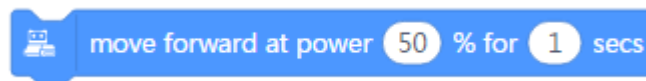
- ☐ The little turtle is standing at the center of the rectangle stage, ready to perform his dance moves.:



- ☐ Use a function block to compose dance steps for the turtle. Name the function as “**Turtle Dance**”.



- ☐ To add dance moves, expressions and sounds, you might need to add the following blocks to the function:

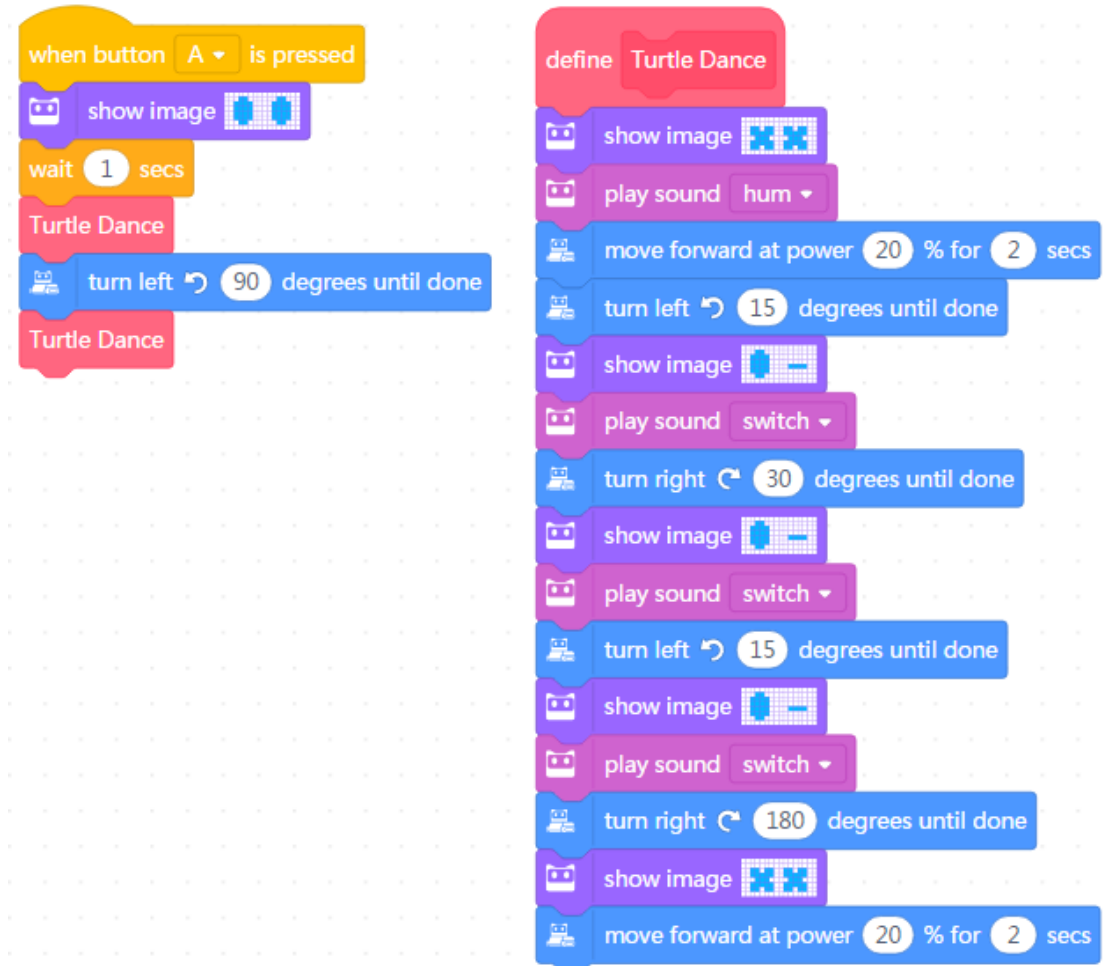


- ☐ The turtle is expected to dance on both sides of the stage. In consideration of this, we need to use the function block two times in different places in the program.

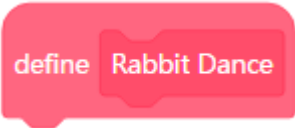
Tips:

1. Some students might use the block “move backward“ to make the turtle return to the center of the stage. This can also work.

Sample Programs:



Task 5: Dancing Rabbit

| Task 5- Dancing Rabbit | |
|--|---|
| The second one is a little rabbit. | |
| <input type="checkbox"/> Use the function block to compose dance steps for the rabbit. Name the function as “ Rabbit Dance ”. |  |
| <input type="checkbox"/> Design dance moves, expressions, and sounds for the rabbit. | |

- The rabbit is expected to start from the center of the cruciform stage and perform the dance on four sides. When finishing dancing on one side, the rabbit needs to return to the center of the stage and faces forward another side. As shown below:



- Use the **Repeat** block to make your program neat.

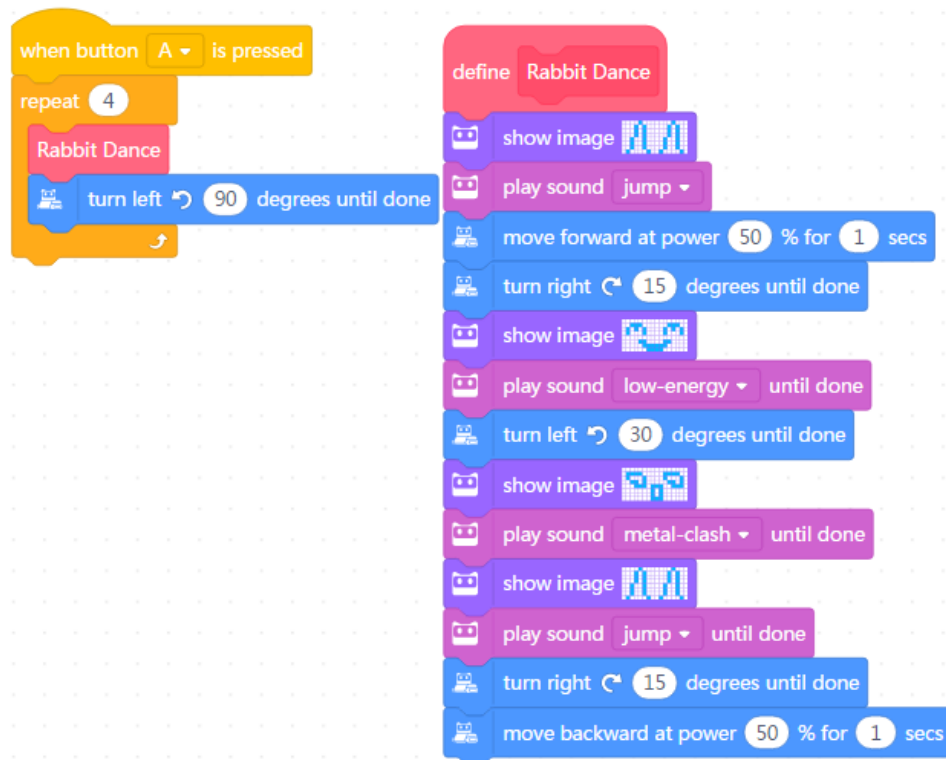


Tips:

1. Some students might call the function in a way that's different from the sample programs.

If students' programs are too complex, show the following sample programs to them.

Sample Programs:

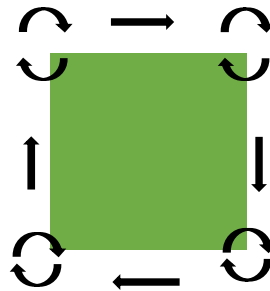


Task 6: Dancing Swan

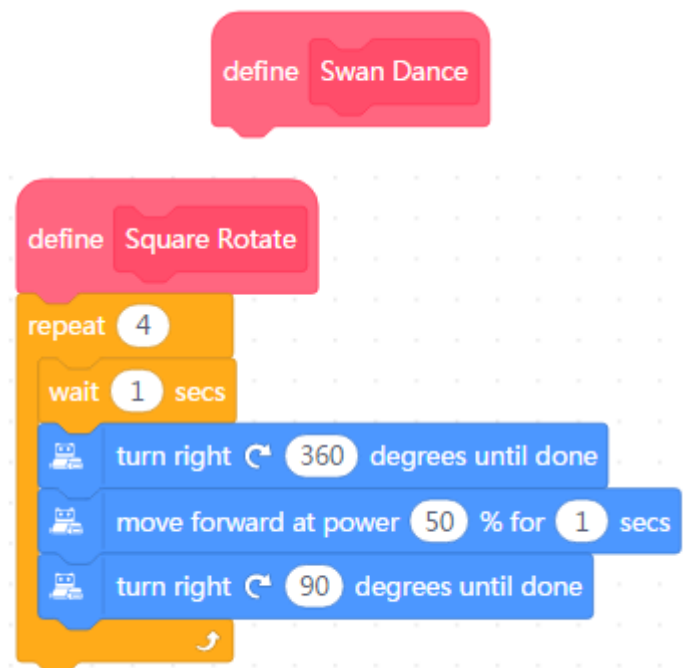
Task 6- Swan Dance

The last one is a swan.

- ☐ Create two functions to compose dance moves for the swan.
- ☐ Like the rabbit, the swan will start from the center of the cruciform stage and dance on four sides of the stage. Make the swan rotate once at each corner of the square (As shown in the picture on the right).



- ☐ You should create two functions. Use one function to represent the dance performances of the swan on each side (Swan Dance). Use the other function to represent the rotations of the swan at each corner (Square Rotate). The second function is used to execute the first function, which forms a nested function. The nested function will make your programs look clean.



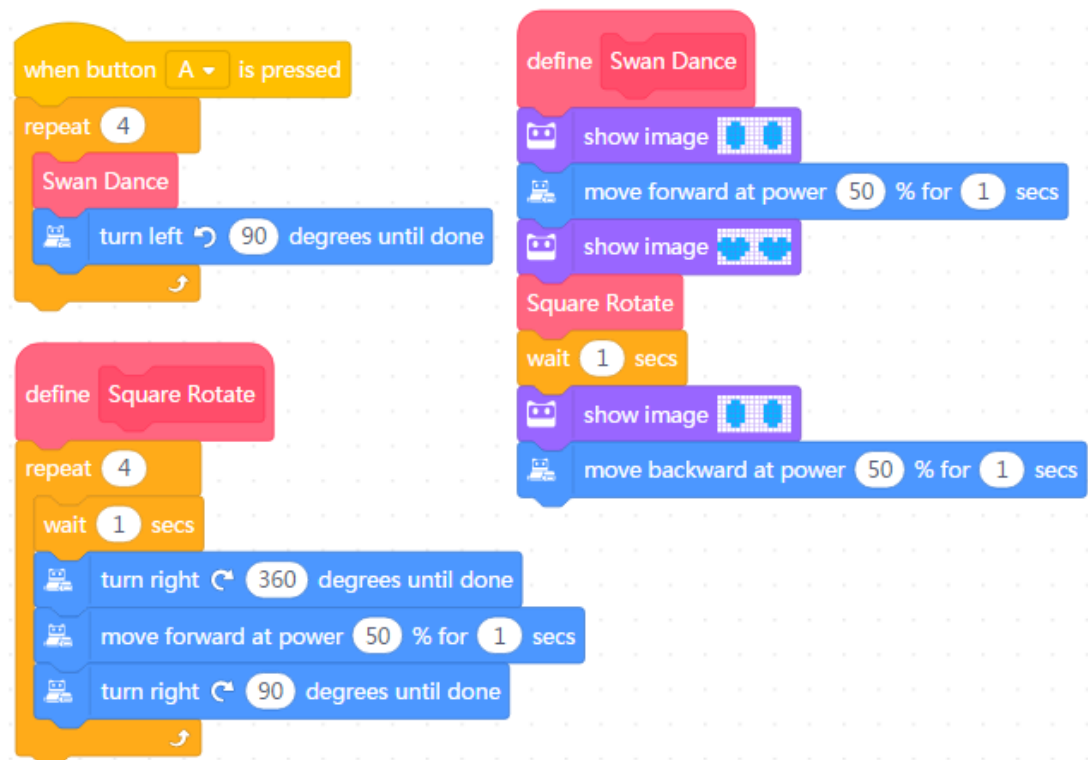
- ☐ Add expressions and sounds.

Tips:

1. You need two functions and one of the function is used to execute the other function;
2. The block “wait 1 sec” in the sample programs makes the swan dance steps clearer;
3. Since the nested function is a complicated concept, teachers could help students complete the task by showing them the following pseudocode:

| | |
|---|--|
| Square Rotate: wait 1 sec rotate 1 round moveforward turn right by 90° <div style="position: absolute; left: 330px; top: 260px;">Repeat four times</div> | Swan Dance: move forward 1 sec square rotate wait 1 sec rotate 180° move forward 1 sec <div style="position: absolute; left: 720px; top: 260px;">Repeat four times</div> |
| When button A pressed: swan rotate turn left by 90° <div style="position: absolute; left: 310px; top: 440px;">Repeat four times</div> | |

Sample Programs:



Self-review Report

Name:

Age:

- Answer the following questions to record your learning outcomes:

Describe what you've learned with one or two sentences.

Describe what you like most and least about this class session briefly.

What I like most

What I like least

Use one or two sentences to articulate why you need to use a function in programs.

How to simplify the following programs using functions? Draw your new programs in the right column.

| | |
|---|--|
|  <pre> when button A is pressed repeat (2) repeat (4) move forward at power 50 % for 1 secs turn left 90 degrees until done move forward at power 50 % for 1 secs repeat (4) move forward at power 50 % for 1 secs turn right 90 degrees until done move forward at power 50 % for 1 secs </pre> | |
|---|--|

You can draw out how you feel about this class session at the upper right corner of the self-review report.

CSTA

| Grades | Identifier | Interim CSTA K-12 CS Standards | Framework Concept | Framework Practice |
|---------|------------|---|----------------------------|-----------------------------------|
| 6 – 8 | 2-A-4-8 | Define and use procedures that hide the complexity of a task and can be reused to solve similar tasks. [Clarification: Students use and modify, but do not necessarily create, procedures with parameters.] | Algorithms and Programming | Developing and Using Abstractions |
| 9 – 10 | 3A-A-4-8 | Deconstruct a complex problem into simpler parts using predefined constructs (e.g., functions and parameters and/or classes). | Algorithms and Programs | Developing and Using Abstractions |
| 11 – 12 | 3B-A-5-7 | Decompose a problem by creating new data types, functions, or classes. | Algorithms and Programs | Creating Computational Artifacts |

| | | |
|--|--|---|
| <p><u>Function: Oversleep</u> cover head with the quilt; wait for 5 seconds; uncover the quilt; stretch out one hand; turn off the alarm clock.</p> | <p><u>Function: Have a look at the time</u> take up the alarm clock; have a look at the time; put down the clock.</p> | <p><u>Function: Get up</u> stretch yourself; uncover the quilt; sit up.</p> |
| <p><u>Function: Put on slippers</u> put on the slipper to the left foot; put on the slipper to the right foot; stand up.</p> | <p><u>Function: Walk</u> lift the left foot; stretch forward the left foot and step on the ground; lift the right foot; stretch forward the right foot and step on the ground.</p> | <p><u>Function: Brush teeth</u> take the toothbrush; squeeze the toothpaste; rinse the mouth; brush the teeth all around; rinse the mouth.</p> |
| <p><u>Function: Wash up</u> turn on the tap; cup the water with hands; wash up with the water; turn off the tap; dry your face with the towel</p> <div style="display: flex; align-items: center;"> <div style="border-left: 1px solid black; padding-left: 5px; margin-left: 10px;"> Repeat three times </div> </div> | <p><u>Function: Take off the coat</u> If it's a T-shirt, take it off from top to bottom; If the coat has buttons, unbutton it first, then the left arm comes out of the left sleeve, next the right arm comes out of the right sleeve.</p> | <p><u>Function: Take off pants</u> pull down the pants; the left leg comes out of the pants; the right leg comes out of the pants.</p> |
| <p><u>Function: Take clothes off</u> take off the coat; take off pants.</p> | <p><u>Function: Put on the coat</u> get the left arm into the left sleeve; get the right arm into the right sleeve; if it's a T-shirt, get your head into the shirt; if the coat has buttons, fasten the buttons.</p> | <p><u>Function: Put on pants</u> get the left leg into the pants; get the right leg into the pants.</p> |
| <p><u>Function: Dress up</u> put on the pants; put on the coat.</p> | <p><u>Function: Carry schoolbag</u> get the right arm through one strap; get the left arm through the other strap; shoulder the schoolbag.</p> | <p><u>Function: Lace the shoes</u> make a circle of the shoelace with the left hand; make a circle of the shoelace with the right hand; tie the two circles</p> |
| <p><u>Function: Put on shoes</u> get the left feet into the left shoe; get the right feet into the right shoe; lace the left shoe.</p> | | |

Task 1-Starting up Function

Create a starting up function for Codey Rocky

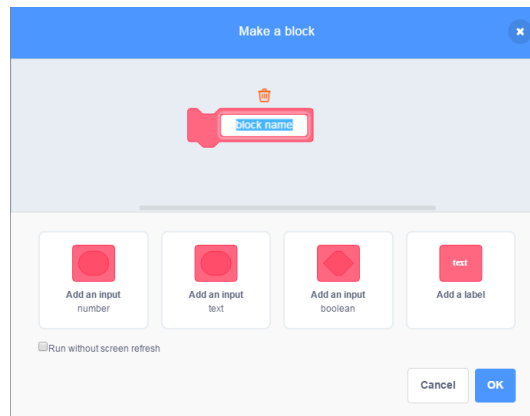
- ❑ Click **My Blocks** at the category bar and select **Make a Block**.

● Variables

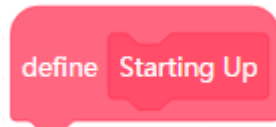
● **My Blocks**

● Infrared

- ❑ Click **Make a Block**. Create a function and give it a name.

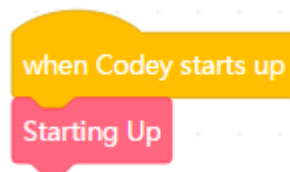


- ❑ Then, the **Define** block for the function will appear in the Scripts area.



- ❑ What programs do you want to run each time Codey Rocky starts up? Design programs under the “**Define start up**” block.

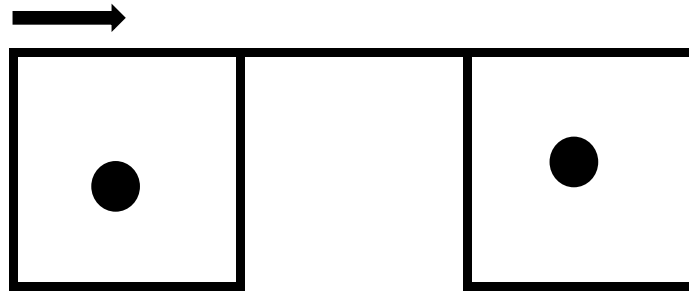
- ❑ After they finish defining the function, have students attach the “**Start up**” block to the bottom of the event block “**When Codey Rocky starts up**”.



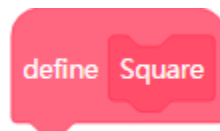
Task 2- Patrol the 1st Floor

Imagine Codey Rocky is a security guard. It has to patrol the passages in the building to keep an eye on properties. Now, it's patrolling the 1st floor.

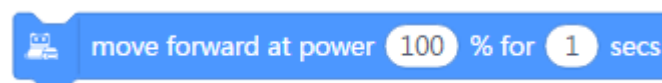
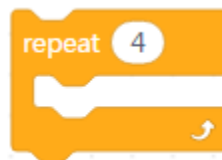
- Design programs to make Codey Rocky patrol the passage by following the black lines as illustrated below.



- Create a function and name it as Square.



- You might need to attach the following blocks to the bottom of the function.

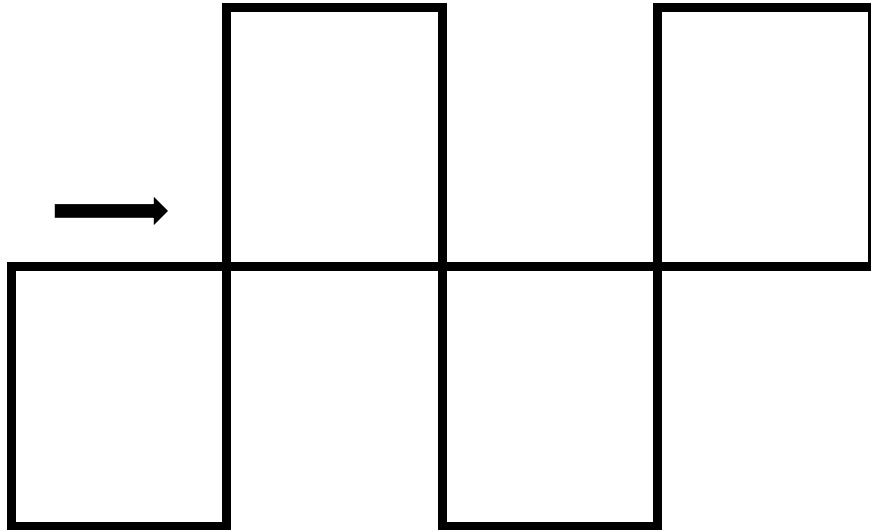


- Under the event block “**When button A pressed**”, call the function Square two times.
- **Challenge:** Add expressions, sounds and lights.

Task 3-Patrol the 2nd Floor

Codey Rocky comes to the 2nd floor. There are more rooms and the passages are more complex than the 1st floor.

- ☐ Design programs to make Codey Rocky patrol in the black route as illustrated below.



- ☐ You need to create two functions, Upper square, Underneath square.
- ☐ Under the event block “**When button A pressed**”, call the two functions in the program.

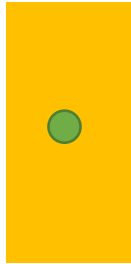
Use the **Repeat** block to make your program neat.

- ☐ **Challenge:** Add expressions, sounds and lights.

Task 4- Dancing Turtle

We are at a dancing party. Animals are going to show their dance moves one by one. The first one is a little turtle.

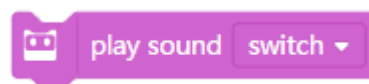
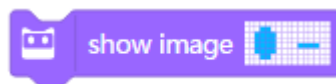
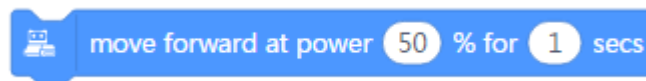
- ☐ The little turtle is standing at the center of the rectangle stage, ready to perform his dance moves.:



- ☐ Use a function block to compose dance steps for the turtle. Name the function as “**Turtle Dance**”.



- ☐ To add dance moves, expressions and sounds, you might need to add the following blocks to the function:



- ☐ The turtle is expected to dance on both sides of the stage. In consideration of this, we need to use the function block two times in different places in the program.

Task 5- Dancing Rabbit

The second one is a little rabbit.

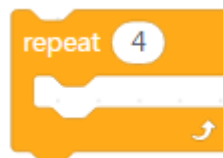
- ☐ Use the function block to compose dance steps for the rabbit. Name the function as “**Rabbit Dance**”.



- ☐ Design dance moves, expressions, and sounds for the rabbit.
- ☐ The rabbit is expected to start from the center of the cruciform stage and perform the dance on four sides. When finishing dancing on one side, the rabbit needs to return to the center of the stage and faces forward another side. As shown below:



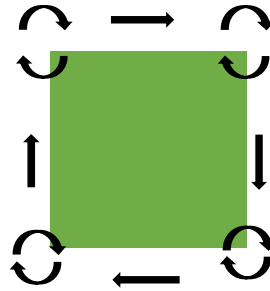
- ☐ Use the **Repeat** block to make your program neat.



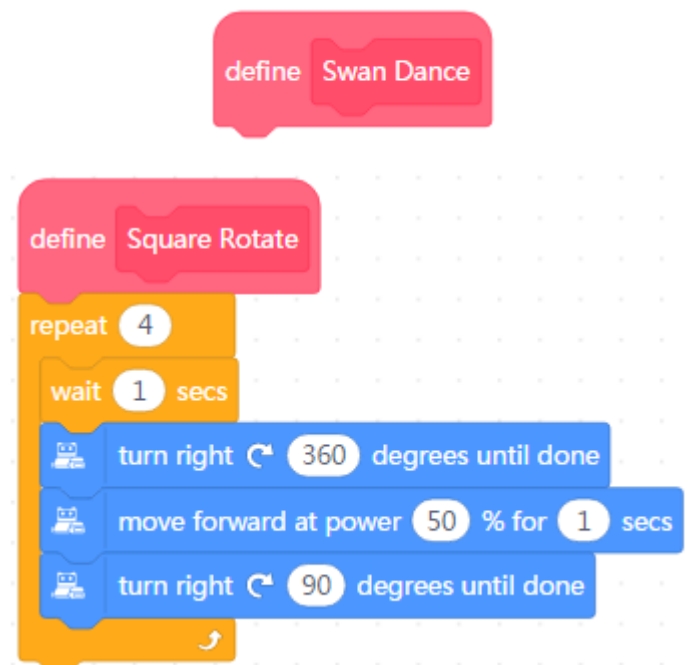
Task 6- Swan Dance

The last one is a swan.

- ☐ Create two functions to compose dance moves for the swan.
- ☐ Like the rabbit, the swan will start from the center of the cruciform stage and dance on four sides of the stage. Make the swan rotate once at each corner of the square (As shown in the picture on the right).



- ☐ You should create two functions. Use one function to represent the dance performances of the swan on each side (Swan Dance). Use the other function to represent the rotations of the swan at each corner (Square Rotate). The second function is used to execute the first function, which forms a nested function. The nested function will make your programs look clean.



- ☐ Add expressions and sounds.

Variables

Overview

| | |
|-----------------------------|---|
| Concept | Variable —— a container storing information that can be changed. |
| Explanations | Examples: Candy jar, Scoreboard, Wallet. |
| Learning Objectives | 1. Understand the concept of Variable ; 2. Create a variable and use the variable in your programs. |
| Teaching Preparation | 1. A box of nuts and lots of variables paper strips (step 3- Lead-in game); 2. A whiteboard and a whiteboard marker (or you can use a blackboard and chalks); 3. One Codey Rocky and a Bluetooth dongle (or a USB cable) per student but it's fine if 2 or 3 students share one set; 4. A computer per student, but it's fine if 2 or 3 students share a computer. And install mBlock 5 on each computer; 5. A self-review report form and project report form per student. |
| Time | 120-240min |

Teaching Procedure

Step 1: Review——Functions

Review:

- How were functions used in a program in the previous lesson? (walking squarely and dancing)
- How did functions help make programs more efficient? (A function can be called multiple times, which makes programs more concise).
- What will the programs be like if we don't use functions? (The programs will be lengthy because some pieces of code might be repeated multiple times. And the logic of the programs might be unclear.)
- What's the difference between a function and a loop? (A loop is repeated in one certain place in a program while a function can be called over and over again in different places in a program.)

A function refers to a group of coding instructions that can be called over and over again. In the mBlock 5, we need to make a block under the category of **My Blocks**. Then, we can call the function block at any time.

Step 2: Explain New Knowledge —— Variables

A variable is a container which stores information that can be changed.

Imagine that a variable is a box. You can put a value into the box and replace it with another value anytime. A scoreboard works in the same way, serving as a container that stores the match scores. Before the match ends, you can change the scores on the scoreboard when necessary. Specifically speaking, a variable is a storage location in a computer, and it is paired with a symbolic name. When a variable is created, the computer will leave a memory location to store the variable and give the location a symbolic name. Through the variable name, programs can read and modify the data in the storage location.

It's important to give a variable a proper and simple name because this helps to ensure the variable can work properly. For instance, if we name the scoreboard as “Red Team”, then the scoreboard will represent the scores that the red team wins in the match. When the red team has a three-point shot, the number on the scoreboard will change accordingly. A new value takes the place of the old value.

How a variable is used can be explained from three aspects:

The first one is **Assignment**. You can copy a value into a variable. For example, a variable “Red Team” is created to store the scores of the red team, and the initial value of the variable is set to be 0. The second one is **Revising Information**, which means you can change the value stored in the variable by adding, subtracting, multiplying or dividing them. For instance, if the red team makes a successful shot, then the value in the variable “Red Team” will increase by 2. The third one is **Comparison**. You can read the current value and compare it with another value. Through comparison, you can tell that when a variable is greater than a certain value, then something will happen. For instance, if the scores of the red team exceed the scores of the blue team, then the red team wins.

There are plenty of examples of variables in real life. The **Health** of characters in games is an example. Similarly, the health points of characters are changing over time in games. The amount of

money in your wallet and the number of candies are both examples of variables. We can read the values and change the amounts of money and candies by adding or subtracting them. Moreover, we can compare the values with other values. For instance, if the amount of money is greater than the price of a certain product, then we can afford to purchase the product; if the amount of candies in the jar exceeds a certain number, then we can share them with friends.

Step 3: Lead-in Game —— Squirrel's Nuts Box

Story and Teaching Preparation:

A little squirrel has a nuts box. Every day, it puts nuts into the box and takes nuts out of the box, so the amount of nuts keeps changing all the time.

It's another day. In the morning, the squirrel gets up and opens its nuts box, finding out there are 10 nuts left.

Teaching Preparation: Teachers prepare a box that's shadow. Put a white paper on the box and writes down a number 10 on the paper. Open the box and show the number to students.

Let's have a look at what happens in the nuts box?

Teaching Preparation: Teachers prepare some paper strips. Each paper strip describes a situation of how the number of nuts changes (as shown below). If there are too many students, invite some of them to pick paper strips randomly; if there are only a few students, then all the students have a chance to pick paper strips. Following the instructions on the paper strips, students should count the current number of the nuts, erase the previous number and write down a new number with pencils.

| |
|--|
| In the morning, the little squirrel eats 2 nuts for breakfast. |
| If the maximum temperature today exceeds 5°C, the squirrel will pick 10 nuts outdoors. |
| At noon, the squirrel eats 3 nuts for lunch. |
| In the afternoon, a little bird pays a visit and gives the squirrel 5 nuts. |
| Today is the monkey's birthday. The little squirrel gives him 3 nuts as a gift. |
| In the evening, a rat steals 4 nuts. |

The squirrel plays the rock-paper-scissors game with a friend three times. If he loses the game, he gives away a nut; if he wins the game, he gets a nut; if it's a draw, then no gain or loss for both sides.

In the evening, the squirrel eats only 1 nut to keep fit.

It's hot today. The squirrel buys a straw hat from Aunt Bear with 7 nuts.

If the amount of the nuts in the box is less than 5, then the squirrel will pick 10 nuts outdoors.

If the amount of the nuts in the box exceeds 12, then the little squirrel will turn in a circle happily.

Ask students:

1. What's the variable in the game? What name will you give it?
2. What's the initial value of the variable?
3. What operations were applied in the game?
4. How did you compare the variable with other values in the game?

Tips:

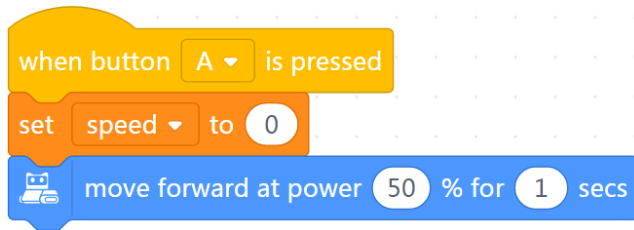
The order of how the paper strips are picked affects the calculated results. In some cases, students might get a negative number after they calculate the nuts. If students have already learned about the concept of a negative number, tell them to replace the previous value with a negative number; if they haven't learned the concept yet, then teachers should revise the conditions on the paper strips to give the squirrel opportunities to gain more nuts.

Summary: Imagine that a variable is a box and you can store a value that can be changed in the box. When you need to use the variable in a program, the box will open and the value will be taken out. Next, you can add, subtract, multiply or divide the value, or compare the value with another value. By doing this, you get a new value and can replace the original value with a new one.

Step 4: Tasks

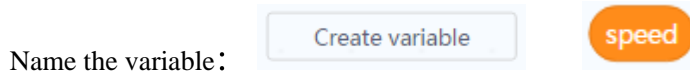
Guided by teachers, students learn how to create, set and use a variable. **Tell them to tick the square box when they finish the step (The Tasks Cards are included at the end of this document. Print them).**

Guided by Teachers: Demonstrate to students how to write the following programs:



Instruction:

1. How to create a variable: Open the mBlock 5, go to the category **Variables** and select **Create Variable**.



2. Set a value:

3. Use the variable:

Task 1: Assign a value to the variable

Have students work in pairs or work on their own to complete the tasks.

| Task 1- Assign a value to the variable | |
|--|---|
| Codey Rocky is standing on the stage to greet the audience. | |
| <input type="checkbox"/> | Codey Rocky turns left 70 degrees to face towards the left and then turns right 140 degrees to face towards the right. Finally, it turns left 70 degrees to return to the starting place. To make Codey Rocky turn specific degrees, we have to create a variable named “ angle ”. |
| <input type="checkbox"/> | Set the value of the variable to be 70. |
| <input type="checkbox"/> | Codey Rocky turns left by the preset degrees. |
| <input type="checkbox"/> | Assign a new value to the variable “ angle ” to make Codey Rocky turn right 140 degrees and then turn left 70 degrees. |
| <input type="checkbox"/> | Add expressions, sounds and lights to each rotation of Codey Rocky. |

Tips:

1. After we create a variable, new blocks will show up in the Scripts area:

Variable block: 

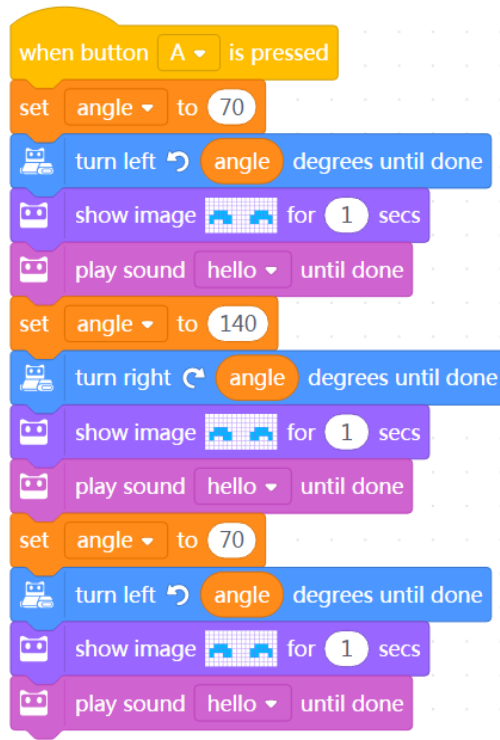
Block for setting the value: 

Block for changing the value:  (a negative number indicates a decrease)

2. Give the variable a proper name. Make it easy to understand;

3. In the program, the variable has three different settings and is used three times.

Sample Program:



Discussion:

1. I want to make Codey Rocky move forward at a certain speed and make it last for 1 second.

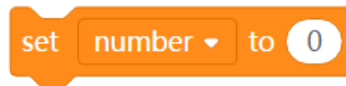
How should I add another variable and how to use it?

Task 2: Add and Subtract

Task 2-Add and Subtract

Codey Rocky have learned the basics of operations and knew how to add 1 to a value and subtract 1 from a value.

- Codey Rocky would like to start with the number 0, so we can set the value of the variable “**number**” to be 0.



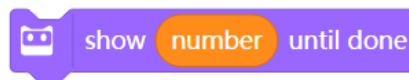
- When the button A is pressed, the number increases by 1.



- When the button B is pressed, the number decreases by 1. A negative number refers to subtracting.



- If you would like to show each calculated result on the LED display, you can add the block “**show () until done**” under each event block.



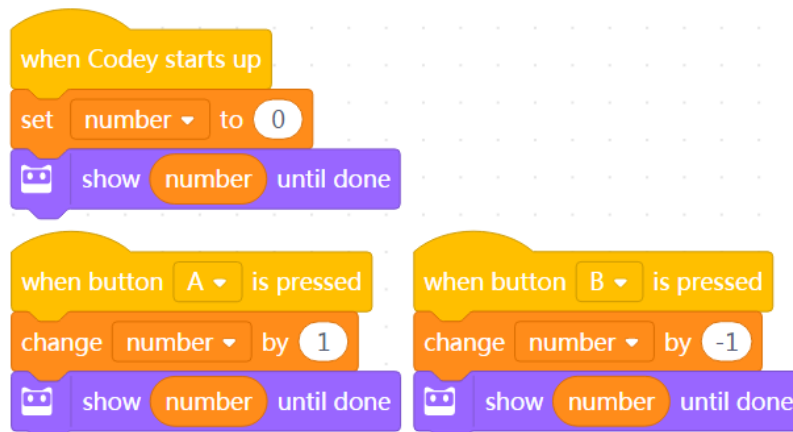
Tips:

1. In the “**change (number) by ()**” block, a negative number refers to subtracting.



2. The parameter dent has three shapes: hexagon, rectangle, and round rectangle. A variable block can only fit into the round rectangle.

Sample Programs:

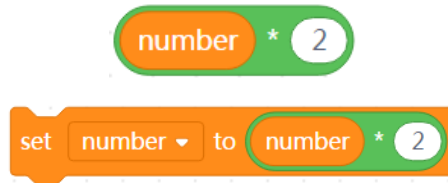


Task 3: Multiply and Divide

Task 3-Multiply and Divide

Codey Rocky also learned how to multiply and divide.

- ☐ Assign an initial value to a variable.
- ☐ When the button A is pressed, the number is multiplied by 2. We need to use the operator block of multiply here to get a calculated result. Then, assign the result to the variable.

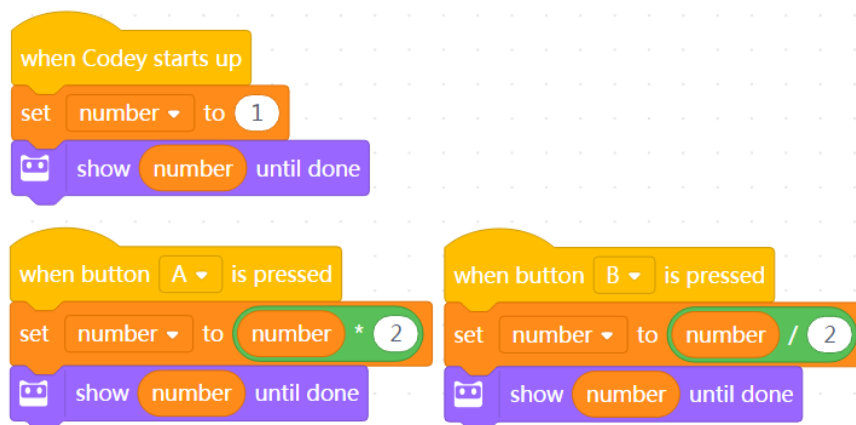


- ☐ When the button B is pressed, the number is divided by 2.
- ☐ Show each calculated result on the LED display. It is worth pointing out that the LED display can only show a value ranging from -999 to 9999.
- ☐ **Challenge:** Change an initial number and a factor (replace the number 2 with another value). Observe the result.

Tips:

1. We need to use the Operators blocks to get calculated results. We replace the previous value with a new value by reassigning the calculated result to the variable.
2. Make sure the variable you use under the three different events (when Codey Rocky starts up; when the button A is pressed; when the button B is pressed) is the same one.

Sample Programs:



Discussion:

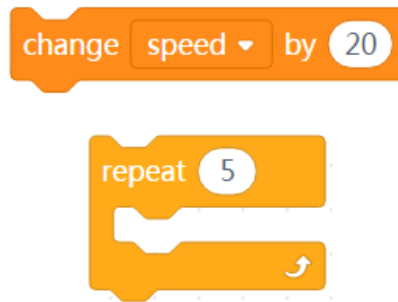
1. If we have a closer look at the values on the LED display, we will notice one thing: if there is a time that the calculated result includes a decimal, then the following calculated results will continue to show up in the form of decimals.

Task 4: Speed up

Task 4-Speed up

Classes are over. Codey Rocky is speeding up and rushing home at its top speed.

- ☐ The initial speed of Codey Rocky is set to be 0, which means that Codey Rocky stays there not moving.
- ☐ When the button A is pressed, the speed will keep rising by 20 at a time. Repeating 5 times, the speed will finally go up to 100.



- ☐ If the speed exceeds 70, the RGB LED will light on with color red to give a warning. In this case, we need to use the **comparison operator block** to tell whether the **conditional statement** is true or not.

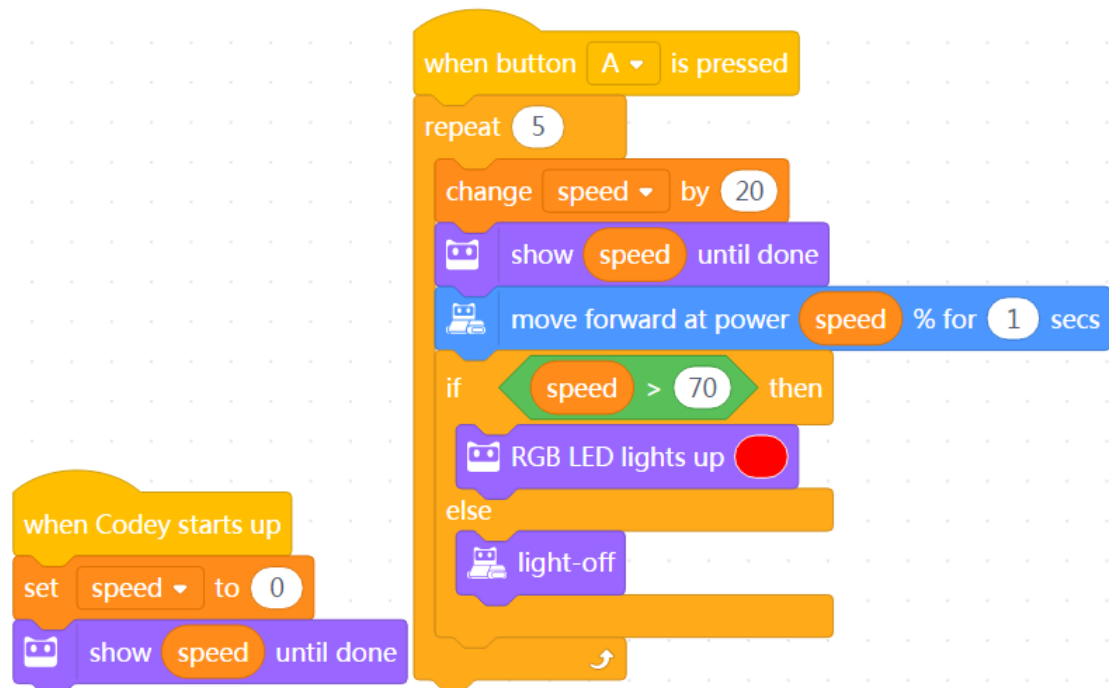


- ☐ Remember to display the value on the LED screen in real time.
- ☐ **Challenge:** Set the speed to be 100 and then keep slowing the speed until making Codey Rocky stop.

Tips:

1. The top power of Codey Rocky is 100% . If a power value exceeds 100%, Codey Rocky will continue to move forward at power 100%.
2. The minimum power of Codey Rocky is 0%. If a power value is a negative number, then Codey Rocky will move in an opposite direction. In other words, it will move backward.
3. Make the LED display show the value promptly the moment the variable changes.

Sample Programs:



Discussion:

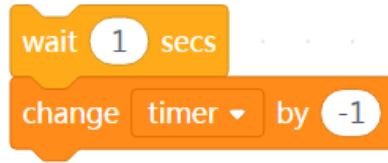
1. Set the initial speed of Codey Rocky to be 100 and keep lowering the speed until it stops. How to revise the programs?
2. Set the power to be a negative number. Observe how Codey Rocky will move?

Task 5: Bomb Countdown

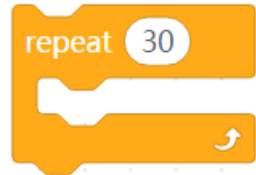
Task 5- Bomb Countdown

Rules: Codey is counting down. During the period, anyone who is holding Codey has to give an animal name. The game ends the moment the bomb blasts.

- ☐ When Codey Rocky starts up, the initial value of the variable “**timer**” is set to be 30.
- ☐ When the button A is pressed, the “timer” will start counting down. After 1 second, the value of “timer” will decrease by 1.



The same step repeats 30 times.



Add sounds to make the countdown more intense and display how much time is left.



- ☐ During the countdown period, anyone who is holding Codey has to give an animal name and then passes Codey to the next person. Animal names cannot be repeated.
- ☐ When the value of “timer” becomes 0, the countdown ends and the bomb blasts. At this time, the RGB LED light on with the color red.
- ☐ **Challenge:** Reset the initial value of the variable “**timer**” to make the game time longer or shorter.

Sample Programs:



Task 6: The Explosive Number

Task 6-The Explosive Number

Rules: When the button is pressed, Codey will generate a number randomly. Have two students play the game rock-paper-scissors. Anyone who loses the game has to press the button. And the number on the LED display will decrease by 1. Repeat the steps until the display shows the number that Codey randomly generates. Boom! The bomb blasts.

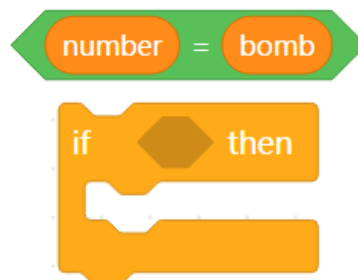
- ☐ We need to create two variables: **bomb** and **number**. The variable “**bomb**” represents the randomly generated number. The variable “**number**” indicates the changing numbers (starting from 0). The bomb blasts when the two variables have the same values.
- ☐ When Codey Rocky starts up, the initial values of the two variables are 0.
- ☐ When the button A is pressed, the value of “**bomb**” is set to be a random number that falls within the range of 1-20.



Meanwhile, Codey puts on his sunglasses and plays the sound “**ready**”.

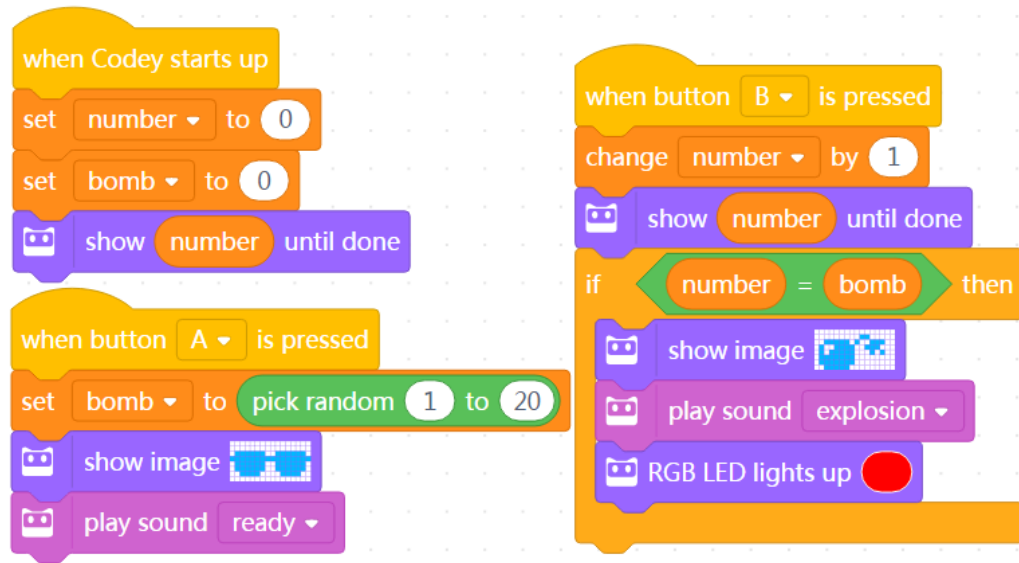
Two students play the game rock-paper-scissors. Anyone who loses the game has to press the button B. When the button is pressed, the value on the LED display increases by 1 and the new value appears on the LED display.

- ☐ If the value of the variable “**number**” is the same one as the “**bomb**” value, an image bomb will appear on the LED display and then comes the sound. You will have to use the operator blocks of comparison and conditional blocks to assess whether the two variables are identical in values.



- ☐ **Challenge:** Revise the value range of the **bomb**.

Sample Programs:



Task 7: Rock-Paper-Scissors

Task 7-Rock-Paper-Scissors

Use Codey Rocky to play the game rock-paper-scissors with friends.

- ☐ When Codey Rocky starts up, the initial values of all the variables are set to be 0.
- ☐ Codey Rocky forms one of the shapes randomly. It uses the number 0, 1 and 2 to represent rock, paper and scissors respectively. When Codey is shaken, the value of the variable “shape” will be set to one of the three numbers randomly.



If the value of the variable “shape” is set to be 0, then an image of a **fist** will appear on the LED display.



If the value of the variable “shape” is set to be 1, then an image of **scissors** will appear on the LED display.

If the value of the variable “shape” is set to be 2, then an image of **paper** will appear on the LED display.

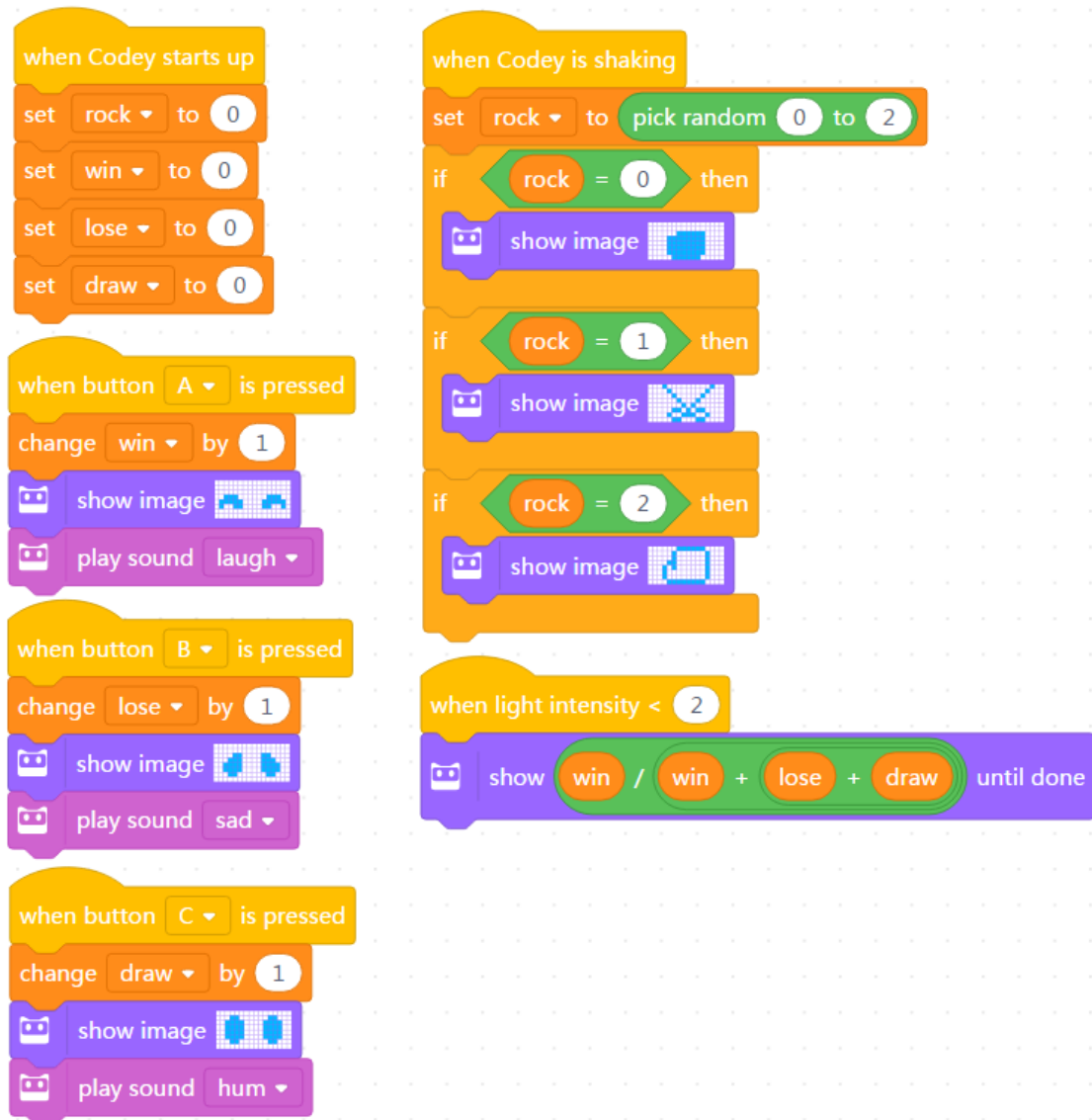
- ❑ If Codey wins the game, then press the button A. The variable “**win**” will increase its value by 1. Meanwhile, Codey shows a smiley face and laughs out.
- ❑ If Codey loses the game, then press the button B. The variable “**lose**” will increase its value by 1. Meanwhile, Codey shows a sad face and cries.
- ❑ If it is a draw, then press the button C. The variable “draw” will increase its value by 1. Meanwhile, Codey looks the same and hums.
- ❑ In some cases, Codey might peep at its winning probability. When the light intensity exceeds 2, the winning probability will appear on the LED display in the form of decimals.



Tips:

1. When we write programs for Codey Rocky, we can use six events at most.

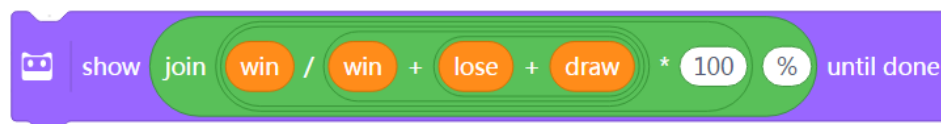
Sample Programs:



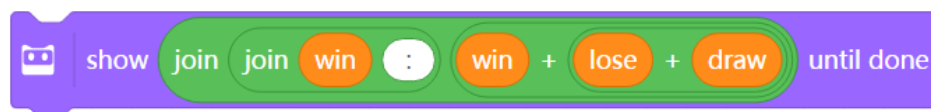
Discussion:

1. How to show the winning probability on the LED display in the form of percentages.

Sample answer:



2. How to show the winning probability in the form of ratio? **Sample answer:**



Task 8: Infrared Battle

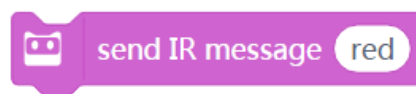
Task 8- Infrared Battle

Rules: When the button is pressed, Codey transmits an infrared signal; if another Codey receives the signal, then the health points go down by 1; the game ends when the health points fall to 0.

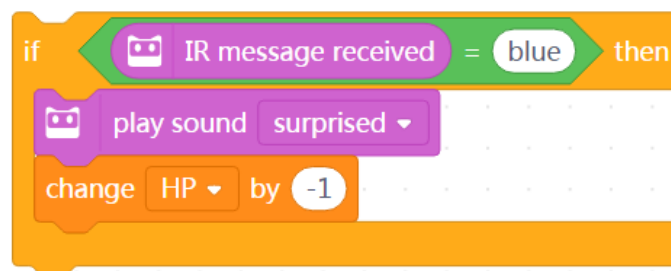
- The two ears of the Codey are preset with an IR receiver and an IR transmitter, which facilitates the wireless communication between two Codeys.



- Select the block “send IR message” in the category “Infrared”. Have two Codeys send each other different messages. In this way, they detect each other. For instance, two Codeys send messages “red” and “blue” respectively. The messages serve as the bullets.



- When one Codey receives the message from another Codey, then we consider the first Codey gets shot and its health points reduce by 1. We use the conditional block here to assess whether the Codey receives an IR message (gets shot). If Codey gets shot, it will make a scary sound and the health points will fall by 1.



- When Codey starts up, the initial value of the health point should be set to 10.
- The next step is to use the **forever** block to keep detecting whether Codey gets “bullets” from its “enemy”. If Codey gets shot, the health point increases by “-1” and plays the sound “scary”. The game is over when the health points fall to 0 and Codey looks so sad.

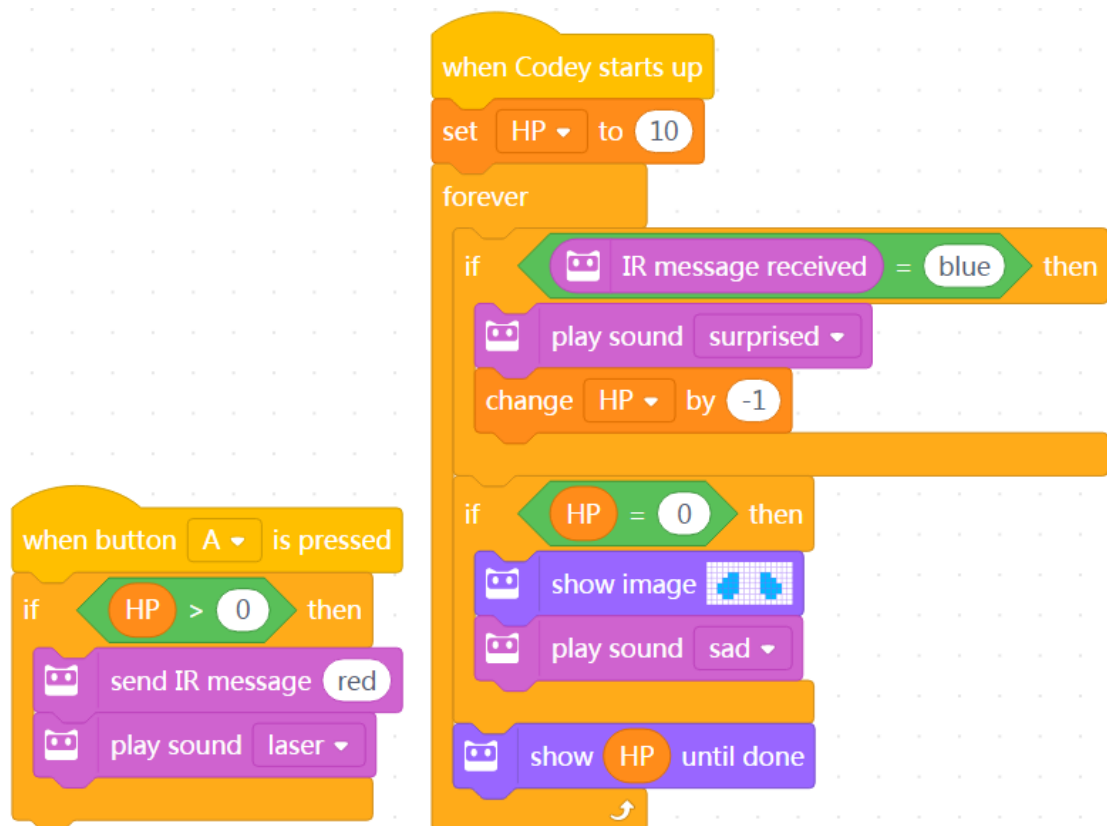
☐ When the button A is pressed, if Codey still has health points (the value is greater than 0), then it can send IR messages and play the sound “laser”.

☐ **Challenge:** If it is a battle of three, then how to revise the programs?

Tips:

1. When we write programs for Codey Rocky, we can use six events at most.

Sample Programs:



Discussion:

1. If there are three teams, then how should we revise the programs?

Self-review Report

Name:

Age:

- Answer the following questions to record your learning outcomes:

Describe what you've learned with one or two sentences.

Describe what you like most and least about this class session briefly.

What I like most

What I like least

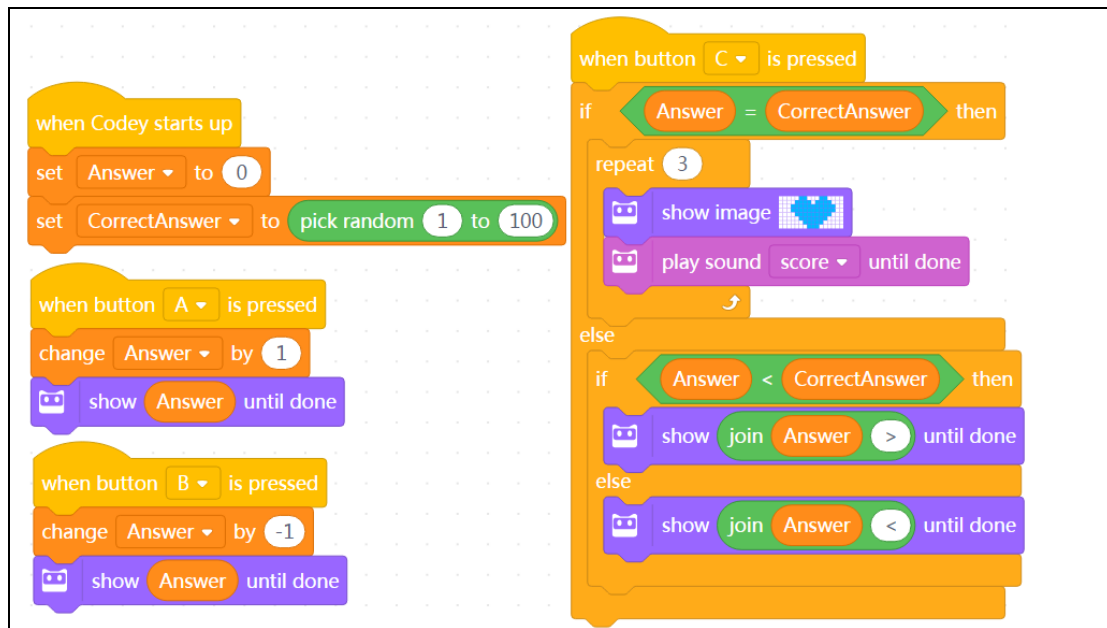
Use one or two sentences to describe variables.

Why do we say a variable is a container?

What does the variable store as a container in this lesson?

How do you use variables in this lesson?

Guess what a game it is? (Answer: _____)



You can draw out how you feel about this class session at the upper right corner of the self-review report.



CSTA

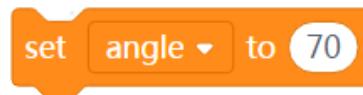
| Grades | Identifier | Interim CSTA K-12 CS Standards | Framework Concept | Framework Practice |
|--------|------------|---|----------------------------|----------------------------------|
| 3 – 5 | 1B-A-5-4 | Construct programs, in order to solve a problem or for creative expression, that include sequencing, events, loops, conditionals, parallelism, and variables, using a block-based visual programming language or text-based language, both independently and collaboratively (e.g., pair programming) | Algorithms and Programming | Creating Computational Artifacts |
| 3 – 5 | 1B-A-5-5 | Use mathematical operations to change a value stored in a variable. | Algorithms and Programs | Creating Computational Artifacts |
| 3 – 5 | 1B-A-6-8 | Analyze and debug (fix) an algorithm that includes sequencing, events, loops, conditionals, parallelism, and variables. | Algorithms and Programs | Testing and Refining |
| 6 – 8 | 2-A-5-7 | Create variables that represent different types of data and manipulate their values. | Algorithms and Programming | Creating Computational Artifacts |

| |
|--|
| In the morning, the little squirrel eats 2 nuts for breakfast. |
| If the maximum temperature today exceeds 5°C, the squirrel will pick 10 nuts outdoors. |
| At noon, the squirrel eats 3 nuts for lunch. |
| In the afternoon, a little bird pays a visit and gives the squirrel 5 nuts. |
| Today is the monkey's birthday. The little squirrel gives him 3 nuts as a gift. |
| In the evening, a rat steals 4 nuts. |
| The squirrel plays the rock-paper-scissors game with a friend three times. If he loses the game, he gives away a nut; if he wins the game, he gets a nut; if it's a draw, then no gain or loss for both sides. . |
| In the evening, the squirrel eats only 1 nut to keep fit. |
| It's hot today. The squirrel buys a straw hat from Aunt Bear with 7 nuts. |
| If the amount of the nuts in the box is less than 5, then the squirrel will pick 10 nuts outdoors. |
| If the amount of the nuts in the box exceeds 12, then the little squirrel will turn in a circle happily. |

Task 1- Assign a value to the variable

Codey Rocky is standing on the stage to greet the audience.

- ☐ Codey Rocky turns left 70 degrees to face towards the left and then turns right 140 degrees to face towards the right. Finally, it turns left 70 degrees to return to the starting place. To make Codey Rocky turn specific degrees, we have to create a variable named “**angle**”.
- ☐ Set the value of the variable to be 70.



- ☐ Codey Rocky turns left by the preset degrees.

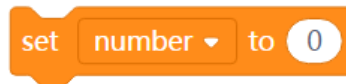


- ☐ Assign a new value to the variable “**angle**” to make Codey Rocky turn right 140 degrees and then turn left 70 degrees.
- ☐ Add expressions, sounds and lights to each rotation of Codey Rocky.

Task 2-Add and Subtract

Codey Rocky have learned the basics of operations and knew how to add 1 to a value and subtract 1 from a value.

- ☐ Codey Rocky would like to start with the number 0, so we can set the value of the variable “**number**” to be 0.



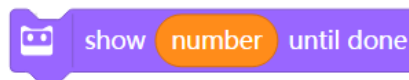
- ☐ When the button A is pressed, the number increases by 1.



- ☐ When the button B is pressed, the number decreases by 1. A negative number refers to subtracting.



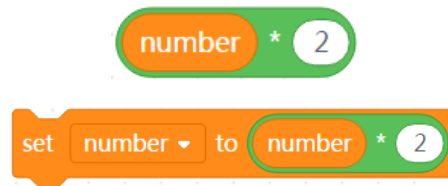
- ☐ If you would like to show each calculated result on the LED display, you can add the block “**show () until done**” under each event block.



Task 3-Multiply and Divide

Codey Rocky also learned how to multiply and divide.

- ☐ Assign an initial value to a variable.
- ☐ When the button A is pressed, the number is multiplied by 2. We need to use the operator block of multiply here to get a calculated result. Then, assign the result to the variable.



- ☐ When the button B is pressed, the number is divided by 2.
- ☐ Show each calculated result on the LED display. It is worth pointing out that the LED display can only show a value ranging from -999 to 9999.
- ☐ **Challenge:** Change an initial number and a factor (replace the number 2 with another value). Observe the result.

Task 4-Speed up

Classes are over. Codey Rocky is speeding up and rushing home at its top speed.

- ☐ The initial speed of Codey Rocky is set to be 0, which means that Codey Rocky stays there not moving.
- ☐ When the button A is pressed, the speed will keep rising by 20 at a time. Repeating 5 times, the speed will finally go up to 100.



- ☐ If the speed exceeds 70, the RGB LED will light on with color red to give a warning. In this case, we need to use the **comparison operator block** to tell whether the **conditional statement** is true or not.

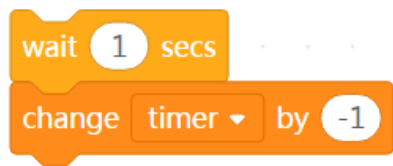


- ☐ Remember to display the value on the LED screen in real time.
- ☐ **Challenge:** Set the speed to be 100 and then keep slowing the speed until making Codey Rocky stop.

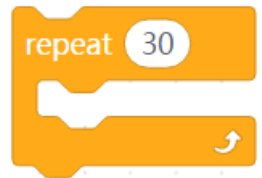
Task 5- Bomb Countdown

Rules: Codey is counting down. During the period, anyone who is holding Codey has to give an animal name. The game ends the moment the bomb blasts.

- ☐ When Codey Rocky starts up, the initial value of the variable “**timer**” is set to be 30.
- ☐ When the button A is pressed, the “timer” will start counting down. After 1 second, the value of “timer” will decrease by 1.



The same step repeats 30 times.



Add sounds to make the countdown more intense and display how much time is left.



- ☐ During the countdown period, anyone who is holding Codey has to give an animal name and then passes Codey to the next person. Animal names cannot be repeated.
- ☐ When the value of “timer” becomes 0, the countdown ends and the bomb blasts. At this time, the RGB LED light on with the color red.
- ☐ **Challenge:** Reset the initial value of the variable “**timer**” to make the game time longer or shorter.

Task 6-The Explosive Number

Rules: When the button is pressed, Codey will generate a number randomly. Have two students play the game rock-paper-scissors. Anyone who loses the game has to press the button. And the number on the LED display will decrease by 1. Repeat the steps until the display shows the number that Codey randomly generates. Boom! The bomb blasts.

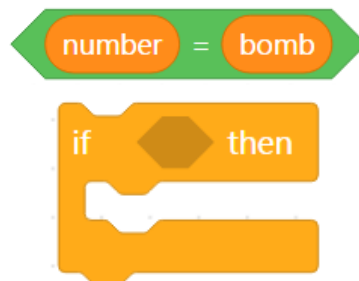
- ☐ We need to create two variables: **bomb** and **number**. The variable “**bomb**” represents the randomly generated number. The variable “**number**” indicates the changing numbers (starting from 0). The bomb blasts when the two variables have the same values.
- ☐ When Codey Rocky starts up, the initial values of the two variables are 0.
- ☐ When the button A is pressed, the value of “**bomb**” is set to be a random number that falls within the range of 1-20.



Meanwhile, Codey puts on his sunglasses and plays the sound “**ready**”.

Two students play the game rock-paper-scissors. Anyone who loses the game has to press the button B. When the button is pressed, the value on the LED display increases by 1 and the new value appears on the LED display.

- ☐ If the value of the variable “**number**” is the same one as the “**bomb**” value, an image bomb will appear on the LED display and then comes the sound. You will have to use the operator blocks of comparison and conditional blocks to assess whether the two variables are identical in values.



- ☐ **Challenge:** Revise the value range of the **bomb**.

Task 7-Rock-Paper-Scissors

Use Codey Rocky to play the game rock-paper-scissors with friends.

- ☐ When Codey Rocky starts up, the initial values of all the variables are set to be 0.
- ☐ Codey Rocky forms one of the shapes randomly. It uses the number 0, 1 and 2 to represent rock, paper and scissors respectively. When Codey is shaken, the value of the variable “shape” will be set to one of the three numbers randomly.



If the value of the variable “shape” is set to be 0, then an image of a **fist** will appear on the LED display.



If the value of the variable “shape” is set to be 1, then an image of **scissors** will appear on the LED display.

If the value of the variable “shape” is set to be 2, then an image of **paper** will appear on the LED display.

- ☐ If Codey wins the game, then press the button A. The variable “**win**” will increase its value by 1. Meanwhile, Codey shows a smiley face and laughs out.
- ☐ If Codey loses the game, then press the button B. The variable “**lose**” will increase its value by 1. Meanwhile, Codey shows a sad face and cries.
- ☐ If it is a draw, then press the button C. The variable “draw” will increase its value by 1. Meanwhile, Codey looks the same and hums.
- ☐ In some cases, Codey might peep at its winning probability. When the light intensity exceeds 2, the winning probability will appear on the LED display in the form of decimals.



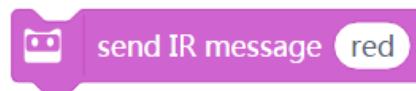
Task 8- Infrared Battle

Rules: When the button is pressed, Codey transmits an infrared signal; if another Codey receives the signal, then the health points go down by 1; the game ends when the health points fall to 0.

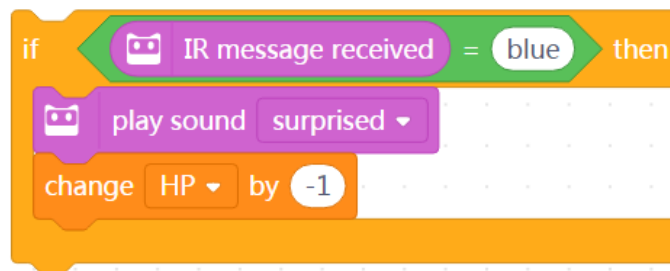
- The two ears of the Codey are preset with an IR receiver and an IR transmitter, which facilitates the wireless communication between two Codeys.



- Select the block “**send IR message**” in the category “**Infrared**”. Have two Codeys send each other different messages. In this way, they detect each other. For instance, two Codeys send messages “red” and “blue” respectively. The messages serve as the bullets.



- When one Codey receives the message from another Codey, then we consider the first Codey gets shot and its health points reduce by 1. We use the conditional block here to assess whether the Codey receives an IR message (gets shot). If Codey gets shot, it will make a scary sound and the health points will fall by 1.



- When Codey starts up, the initial value of the health point should be set to 10. The next step is to use the **forever** block to keep detecting whether Codey gets “bullets” from its “enemy”. If Codey gets shot, the health point increases by “-1” and plays the sound “**scary**”. The game is over when the health points fall to 0 and Codey looks so sad.
- When the button A is pressed, if Codey still has health points (the value is greater than 0), then it can send IR messages and play the sound “**laser**”.
- **Challenge: If it is a battle of three, then how to revise the programs?**