My Robotic Friends

ACTIVITY OVERVIEW

STEM Focus Area: Robotics

Learning Goal: Demonstrate understanding of computational thinking concepts (sequence, algorithm, testing, debugging) & practice giving simple and precise, multi-step directions to solve complex problems.

Youth Learning Target

- "I understand that robots follow commands."
- "I can control a robot with an algorithm."
- "I understand that there are better ways than others to write an algorithm."
- "I understand there is more than on way to write an algorithm to do something."

LEARNING ENVIRONMENT

Activity Duration: 40-60 minutes

Group Size: Large or Small

Age of Youth: Grades 3-5 [ages 9-11]

Guiding Question - What is the question to explore OR the problem or challenge to solve?

How can you guide a robot to complete a task by using computational sequences (called algorithms)?

Throughout this activity, youth will:

- Create an "algorithm" designed to complete a specific task.
- Develop an understanding and observe how robots respond only to the commands given to them.
- Have an opportunity to "debug" or edit/fix a sequence of commands.
- Have an opportunity to utilize "loops" to simplify a code.

Facilitator Prep

- Ensure materials are set up before activity.
- Watch Unplugged - Real-Life Algorithms: Planting a Seed
- Watch My Robotic Friends

Literacy Connection: Great books to get youth support learning

Books to read while "Robots" are waiting for "Coder" to create their code. Books about computer science would be most appropriate. Here are some good examples:

- *How to Code a Sandcastle*
- *Gabi's if/then Garden*
- *Hello Ruby: Adventures in Coding*

PREPARATION

Materials

Per Group:

- Symbol Key *(page 6 of this document)*
- Stack Pack *(pages 7-9)*
- 6-9 disposable cups
- Step Guide *(page 10)*

Per Participant:

- Blank paper/note cards
- Pen/pencil

Room

Ensure each group has enough space to write and make the stacks of cups. Ensure there is a separate area/corner where the "Robot" can wait for the code to be written.

Stem Content Learning

Key concepts:

- **Robots**: a machine that carries out a series of actions automatically, especially one programmable/programmed by a computer.
- **Computational Thinking**: a problem solving process that involves logically ordering and analyzing data to create solutions to problems using a series of ordered steps.
- **Algorithms**: a sequence of instruction intended to complete a particular task.
- **Debugging**: correcting a part of an algorithm to better complete its goal.
- **Loop**: a repetition in the code by an specified amount *(eg. Loop 3x, loop 10x, etc)*

Inquiry

Within the given coding parameters, youth will develop a code to provide their "Robots." Observing the "Robots" read and act upon the code given to them and fixing any mistakes in the code will model the process that computer scientists use in real lie. Youth will guide the alterations/changes to the codes as they progress through the activity.

Potential Question to support inquiry:

- How did you know that the code needed debugging?
- What would you need to do in order for the robot to correct its own code?
- How do you know if you have written the right code? Is there more than one correct way to write a code to complete a particular task?
-

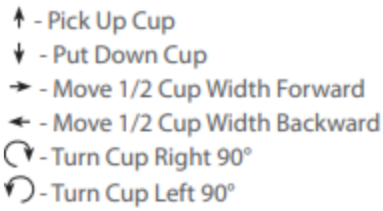
INTRODUCTION TO ACTIVITY (15 MINUTES)

Start with a discussion the participants' experiences with computers and robots:

- "Has anyone here seen or interacted with a robot? How do you interact with a robot? Does the robot really "hear" you speak? Does it "understand" what you are saying?"
- "Robots don't hear and understand the same way people do. They operate off "instructions," specific things they're preprogrammed to do. In order to accomplish a task, they need to have a series of instructions (called an **algorithm**) that they can reference."
- "We rely on algorithms in our everyday lives to complete tasks. Brushing your teeth requires an algorithm. What is are the series of steps you need to brush your teeth? What would happen if you did them out of order? "

Explain the activity

- Youth will be divided into groups of three.
- The facilitator will choose one team member to be the "Robot" and the rest to be "Coders"
- The robot to the "Robot Library" to wait until the "Coder" to write their code.
- The facilitator will choose one image from the Cup Stack Pack for every group each round.
- The other team members will create an algorithm for how the robot should build the select stack.
- Coders will translate their algorithm to arrows, as described by the Symbol Key (page 10).
- Show the participants the Symbol Key (page 10, either on a projector or on printed sheets), as well as the step guide. Explain to them that these are the only symbols that they will be using in their algorithm to instruct their "robot" to build a specified cup stack.


↑ - Pick Up Cup
↓ - Put Down Cup
→ - Move 1/2 Cup Width Forward
← - Move 1/2 Cup Width Backward
↻ - Turn Cup Right 90°
↺ - Turn Cup Left 90°

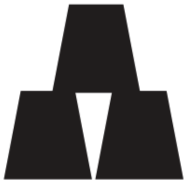
- When the "Coders" have finished coding they can retrieve their robot from the library.
- When the robot returns to the group, the robot reads the symbols from the cards and translates them back into movements.
- The group should watch for incorrect movements, then work together to ***debug*** *(change)* their program before asking the robot to re-run it.

Rules

- Coders should translate all moves using **ONLY** the six arrows.
- Cups should remain with the robot, **NOT** provided to the coders during coding.
- Once robots are back with their groups, there should be **NO TALKING** between the coders and the robots.

Go over an example together

Show them the image of the stack, either on the board or from the distributed images.



Tell them one "step" is only half the width of a cup. Place your stack of cups on the table where they can be seen by everyone. Ask the class to instruct you on the first step from the list of instructions (the correct answer is "↑ - pick up cup"). Whole holding the cup, ask them for the next step. You may have to remind them that one "step" is half the width of a cup.

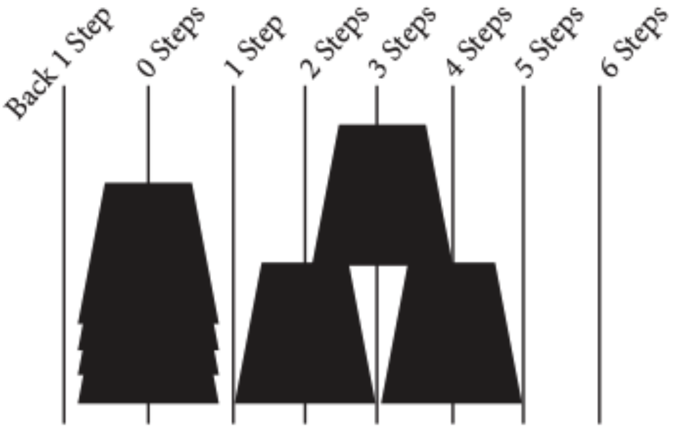


Fig 1: STEP GUIDE

Once you've placed the cup, transition back to the board, and challenge the class to have you write the entire "program" using the symbols. One possible solution looks like this:

↑ → → ↓ ← ←

↑ → → → → ↓ ← ← ← ←

↑ → → → ↓

Once the program is written, call a volunteer to "run" it by reciting the steps out loud. Have them say the arrows out loud as you play the part of the robot and put the cups in place. Ensure it is built correctly before moving on.

ACTIVITY ENGAGEMENT

- Divide the participants into groups of 3.
- Have the groups run through exercise: selecting a robot, coding a program, running the program, and repeating until everyone has an opportunity to fulfill all the roles.
- Each time a group solves a challenge, the group should be given a more difficult cup stack card.

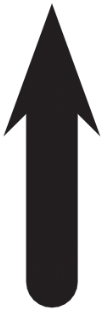
Activity Engagement

- **Participation**: Ensure each member of the group fulfills each role.
- There's a chance that when they start approaching the more complex cup stacks, their code will start to get very long and repetitive. Use this as an opportunity to introduce the concept of loops – either to the whole room or each individual group.
  - Q: What could have made this easier? Was there lots of repetition in the code? How could you have simplified that?
    - A: We can use loops to repeat the same command. If writing "←" four times in a row, try "← (x4)"
- You can introduce them to the concept of functions and encouraging them to break their code into different sections to better organize it.
  - Q: If you were a coder looking at the code someone else wrote, would all these commands in the program be overwhelming to try and read? How could we make this easier to read?
  - Function 1: bottom row, Function 2: middle row, Function 3: top row

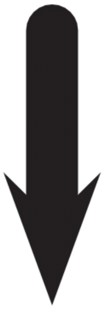
FINAL REFLECTION & RELEVANCE (5 MINUTES):

- “Was this easier or harder than you anticipated? What made it challenging/easy?”
- Have the groups compare code from the larger/more complex cup stacks.
    - “Are these codes the same or different? What does this mean? How do you know if a code is correct? Is there more than one way to write correct code?”
- “Do you ever use algorithms in your everyday life? What are some examples?”
    - *Brushing teeth, making breakfast, taking a shower are all processes that use a sequence of step to accomplish a goal.*
- “What are some professions or circumstances you can see algorithms being useful?”

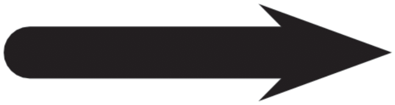
Symbol Key



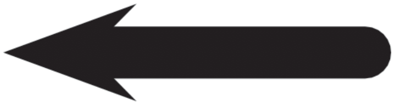
Pick Up Cup



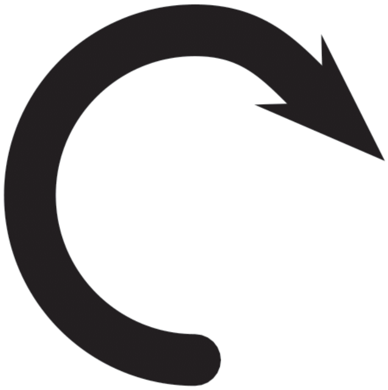
Put Down Cup



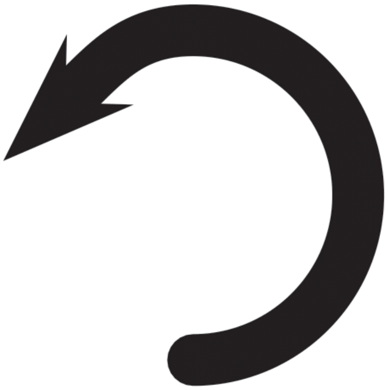
Step Forward



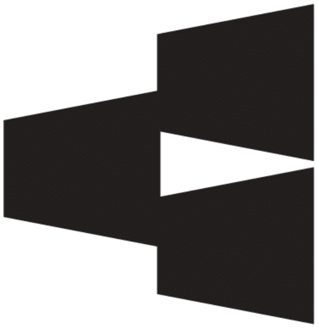
Step Backward

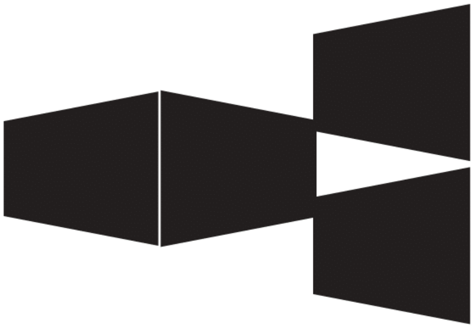


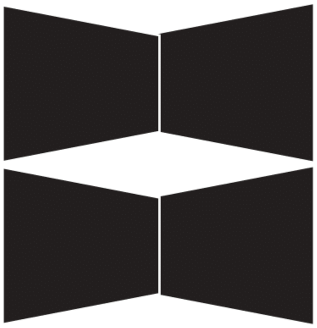
Turn Cup Right 90°

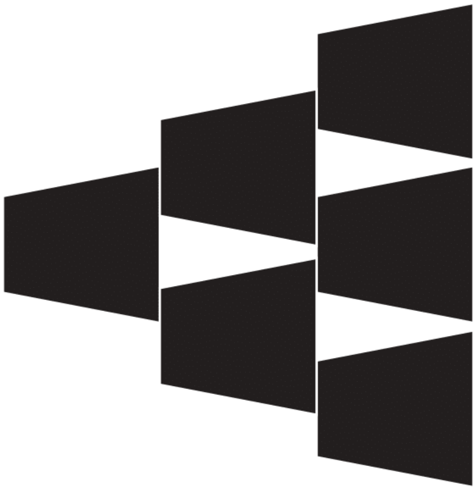


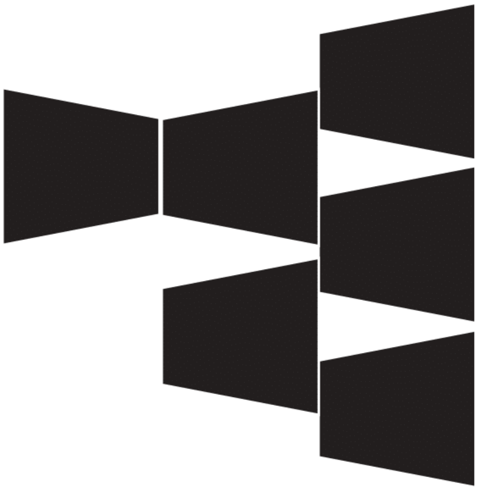
Turn Cup Left 90°

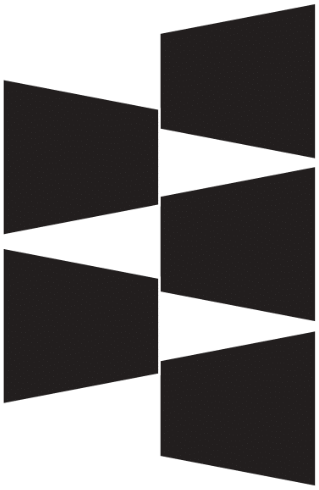


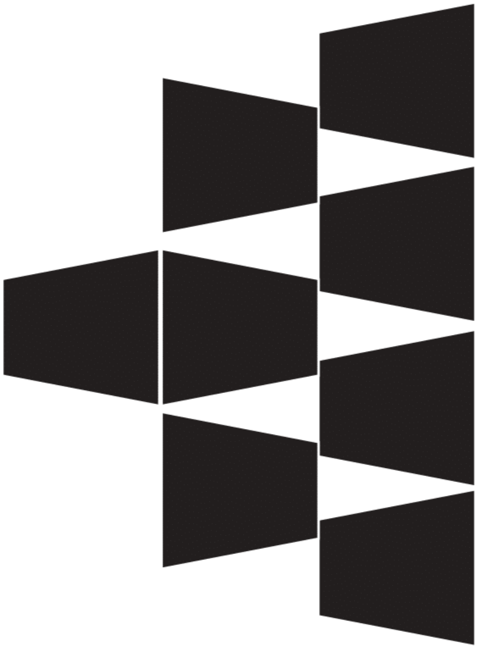


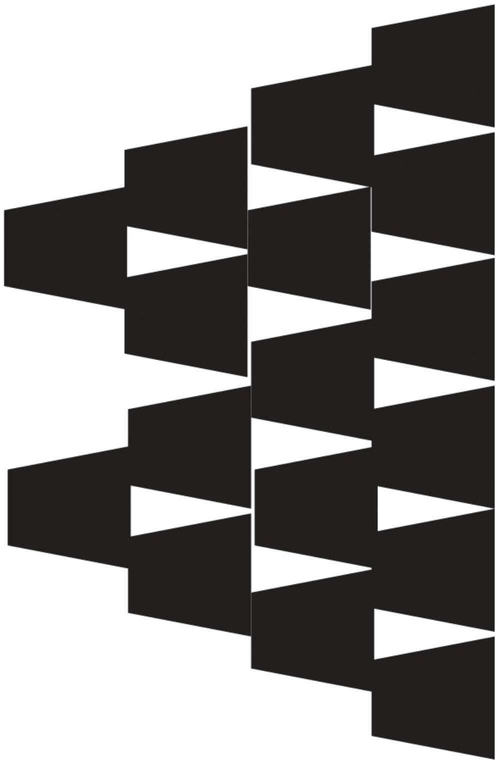


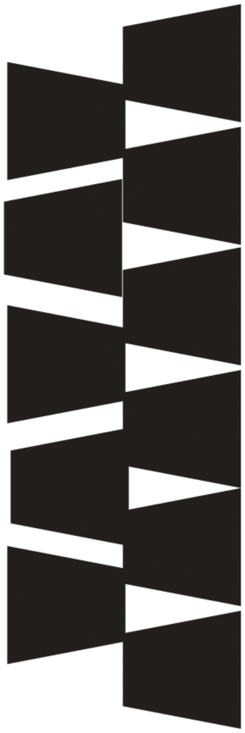


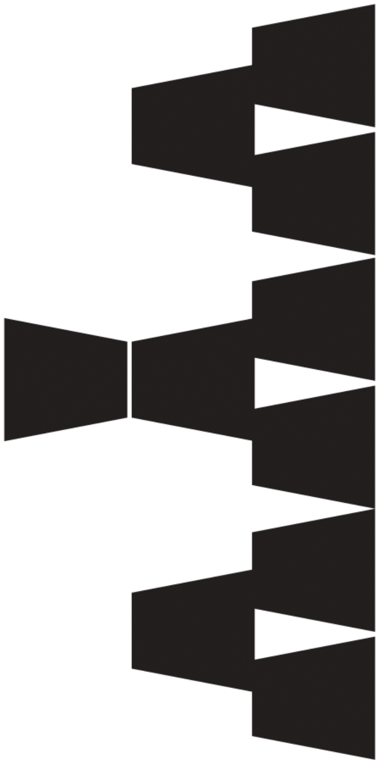


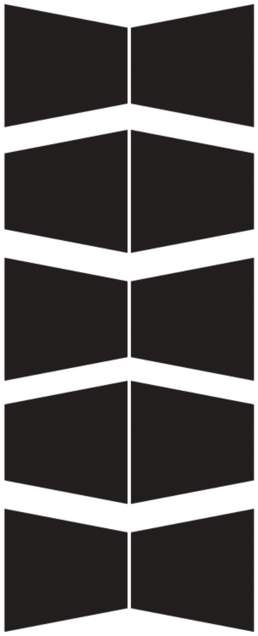


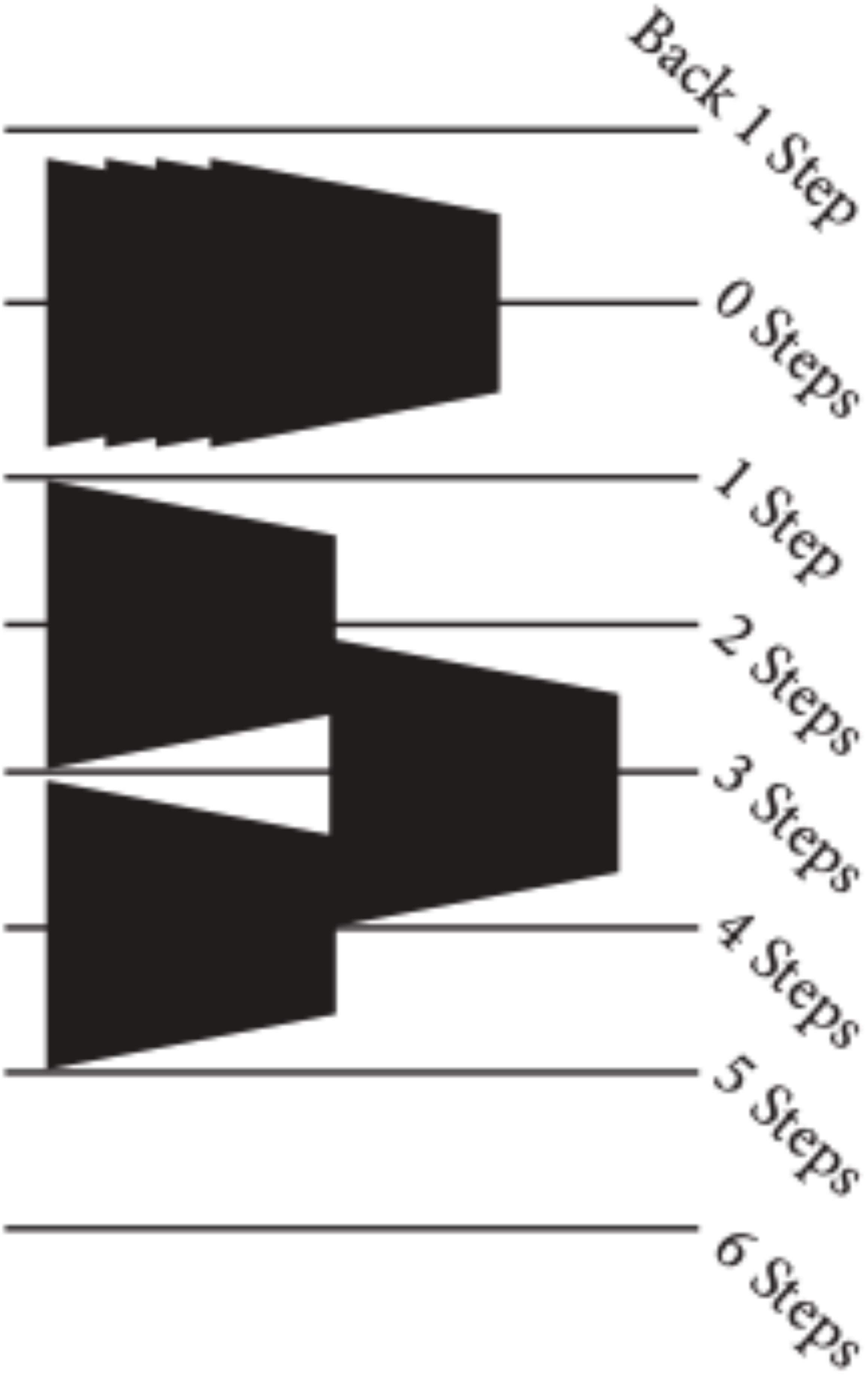











Back 1 Step
0 Steps
1 Step
2 Steps
3 Steps
4 Steps
5 Steps
6 Steps