ROBOT BRAINS: PROGRAMMING A ROBOT

AT A GLANCE Students will explore the intricacies of robotic programming through an activity where they act as robots and programmers.

OBJECTIVES

Students will:

- Understand how robots complete simple tasks.
- Learn the basics of how programming works.
- Explore how difficult it can be to create a robot that mimics a human.

KEY VOCABULARY

Programming, Computational Thinking

NEXT GENERATION SCIENCE STANDARDS

Science and Engineering Practices:

- Asking questions and defining problems
- Developing and using models
- Planning and carrying out investigations
- Constructing explanations and designing solutions
- Obtaining, evaluating and communicating information

Crosscutting Concepts:

- Patterns
- Cause and effect

Disciplinary Core Ideas:

- PS4: Waves and their applications in technologies for information transfer
- ETS1: Engineering design

PACE YOURSELF

- 45 minutes

ADVANCE PREPARATION

1. Gather 12 index cards per team of two students, along with extras for the entire class.
2. Decide how to divide the class into pairs.
3. Create a sample set of programing directions.
4. Prepare a list of example tasks students will need to program their robots to complete.
5. Use the internet to find and print out various types of robotic coding.
6. If classroom computers are available, explore the digital resources to determine which ones would best fit the needs of the class.

MATERIALS

Per student:

- At least 10 index cards (or other small sheets of paper)
- A pencil or pen

WHAT YOU NEED TO KNOW

To complete their tasks, robots have to sense, plan and act. Robots use different kinds of sensors to collect the information they need. Software processes this information so the robot can plan a response. Then they act to get the job done.

Robots need to have someone tell them what to do. This process is referred to as **programming**: the way that we can make robots or computers follow instructions. Any action a robot is going to do needs to be specifically programmed for it to complete the task. A robot "thinks" and learns by processing data and then uses this information to plan its actions.

Programming is the source of instructions for the robot. A robot's program is a set of instructions that tells it what to do, how to do it and when to do it. In order for the robot to complete a task it must be programmed to do so. Programming requires defining a robot's task as a series of logically based step-by-step instructions that can be followed sequentially in order to reach a goal.

A robot's program will also contain a library of simple commands that allow a programmer to describe things in the same way every time. Robots cannot interpret variability in commands the way humans do. For example, a robot will not be able to differentiate between a command to "sit down" versus "please sit."

Emphasizing the specificity of the command is important. At times, the robot's environment may be strewn with obstacles or unexpected events. Programmers need to think through the possibilities of these various scenarios and plan accordingly so they can communicate in any situation.

The tasks a robot completes must be defined up front. For example, if the robot does not know how to respond to a command to "sit", then giving the robot that command will result in no action from the robot. A robot will have a library of available actions. Without that action being in its library, the robot is not capable of understanding. One way to think about this is to imagine a programming language that does not have the addition (+) operator that is used in math. In this example, the programmer can never give the computer any math operation that includes addition. Similarly, if the robot does not have a library of actions that says "lift feet," then a programming step that includes "lift feet" will result in the robot doing nothing.

Robot software is the collection of coded commands that tell the robot what tasks to perform. Robot software is used to determine what tasks should be performed and to carry out that action. Programming robots can be an intricate task. Many software systems and frameworks have been proposed to make programming robots easier.

Programming involves computational thinking: a way to analyze and solve problems. **Computational thinking** requires deconstructing the entire decision-making process, the variables involved and all possible solutions, ensuring the right decision is made based on the corresponding parameters and limitations of the problem. Computational thinking can be useful in almost any situation that requires solving a challenge.

WARM UP

Lead students through a discussion about robot programming. Possible questions could include:

- How do people complete a new task they have never done?
- How does the robot know what to do or what types of actions to take?
- In what ways are robots not smart?
- What is holding them back?

ACTIVITY

1. After the discussion, tell they class they are going to explore the idea of robot knowledge and programming.
2. Show the students an example of robotics code. Explain that this is created to tell robots how to complete various tasks. Today they're going to be creating their own programming for human "robots."
3. Divide the class into pairs of students. Explain that one student in each team of two will play the role of the "programmer" and the other will play the role of the "robot." Each robot will be asked to carry out a specific action, such as picking up an object or turning on a computer. The robot cannot carry out any action without being specifically told by the programmer. The student programmers will be programming each of the robots to do specific tasks.
4. Share an example with the students. Read off the programming directions on the prepared example and have a student robot complete the task.
5. Discuss the fact that each step in the programming should be one discrete action, such as walking three steps forward or moving their arm. There should not be more than one action verb in each step.

6. Give each of the programming students six index cards and a list of tasks they can program their robot to complete. Examples include moving a book to a different location or throwing away a piece of paper. To prevent the robot students from just naturally completing a task, you may choose to make sure the robot students don't know what the task is that they'll be performing or have the robot students complete the task blindfolded.
7. Ask the programmer to write directions for the robot to complete the task. Each step should be specifically described on an index card, such as move forward six steps. Students do not need to use all six cards if they're not needed.
8. After filling out the cards have the programmer read the directions step by step. The robot should carry out the instructions for each step.
9. Discuss if the teams were able to carry out their task with six steps. Was it hard to think through all the steps before having the robot do them? How many steps were needed to do the tasks? Did everyone do the exact same steps?
10. If the tasks could not be completed, allow students to change their code or use more cards to create additional code.
11. Use the revised cards to try the task again. Did it work better this time? Why or why not?
12. Have the students switch roles with their partner.
13. After everyone has had a chance to write code for their robot, discuss what happened. Was it easy to make the robot do tasks that would be simple for humans? Why or why not? What happened if the code was not written correctly? Discuss how programming is important in robots.
14. Allow students an opportunity to take turns as they program more complex tasks.

CHECK FOR UNDERSTANDING

- Throughout the lesson have students discuss the detailed nature of instructions needed for a robot to perform any task.
- Have students share a time when a computer or other programmed system didn't do what they wanted it to. What are some of the reasons why?
- Ask students to discuss the various things they utilize that have to be programmed.

WHAT'S HAPPENING?

Without step-by-step instructions, the student acting as a robot did not have enough information to complete whatever task he or she was given. This information, referred to as programming, needs to be explicitly given to the robot. In addition, this information has to match against a library of basic actions or steps that the robot can take. This library is key to understanding the limits of a robot's capabilities. Regardless of instructions, the robot cannot operate outside of the limits of this library of actions. If it does not know how to respond to a particular action, it won't be able to complete its task.

DIFFERENTIATED INSTRUCTION

Depending on the class level, students can be given either more complex or less complex tasks to use for programming.

ROBOT BRAINS: PROGRAMMING A ROBOT

EXTENSIONS

- Complete the activity again with a pre-determined set of robot actions on index cards. In the sample set, show students what happens when they provide an instruction to the robot that the robot is not capable of understanding since it's not in the library. Have students create programmed directions that only have the actions available for that robot.
- Have students do the activity again and create various functions instead of writing out a full line of code. Functions serve as shortcuts, or a way to have a short set of symbols represent a more complex action. These allow programmers to create code without as many lines or information typed out.
- If computers are available, allow students to further explore the basics of coding online using the digital resources.
- Allow students to research more on the history of the Turing test, which was created in 1950 as a way to test a computer's ability to exhibit intelligent behavior. If computers are available, have students use this test with a real chat bot to determine how easy or hard it is for a computer to mimic a human being.

DIGITAL RESOURCES

Free online programming language that makes it easy to create interactive art, stories, simulations and games
http://scratch.mit.edu/

Online community for Scratch educators
http://scratched.gse.harvard.edu/

Computer science non-profit organization
http://code.org/

Online chatbots
http://www.elbot.com/
http://www.cleverbot.com/

Programming-related iPad apps
http://www.daisythedinosaur.com/
https://www.gethopscotch.com/
http://twolivesleft.com/CargoBot/

IN THE EXHIBIT

- Google Self Driving Car
- Programming interactive