Create It With CODE

You can learn basic computer science in just one hour!

Given just one hour, Roshni Srikanth can show an entire class how to use computer code to make a dragon fly through a maze or make a robot dance. Roshni, a 17-year-old senior at Tesla STEM High School in Redmond, Washington, helps run Hour of Code events along with her twin sister Tarini. While the students are bringing their games and apps to life, Roshni and Tarini are also teaching them the basics of computer coding. "Students can see their dragons move and the things they're doing," Roshni says. "They're able to problem solve by themselves."

The Hour of Code is an international event held during Computer Science Education Week, from December 9 through 15 this year. Its goal is to get classrooms around the world to spend one hour on a computer science project. Students of all levels can participate with activities that teach the basics of computer coding.

"The Hour of Code is an opportunity to try computer science. In one hour you can build a game or app or code art. You can get a sense of what computer science is," says Alice Steinglass. She's the president of Code.org, which helps run the Hour of Code. The event started six years ago.

Computers are vital in classrooms and beyond. But as Steinglass points out, many students aren't learning how they work or how to program them in school. "We learn how to read books, but imagine if we only learned how to read and never how to write. Learning computer science in school is enabling everyone to not just be a reader but also a writer," Steinglass says.

The Hour of Code helps students get excited about computer science by letting them try it firsthand. "Our main goal is not for anyone to be scared," says Tarini. "We're saying if you want to be a computer scientist, don't let anyone stop you."

—Jennifer Hackett

VIDEO

Learn more about code at scholastic.com/math

CODE WORDS

These terms are used to describe different parts of code.

`Input:` A value or series of values given to a computer to use in a program

`Output:` A value given by a computer as a result of running a computer program

`Command:` Instructions that a computer follows

`Variable:` A placeholder that represents a value. A variable can also be used as an input.

`Conditional Statement:` An "if, then" statement that tells a computer how to react to different inputs

MEET MATH++

***Scholastic MATH*'s editors invented this programming language called MATH++. It has five commands and one variable.**

`SUM+(x, y, z)`
This command adds the inputs (up to three terms), from left to right.

`MIN-(x, y, z)`
This command subtracts the inputs (up to three terms), from left to right.

`DIV/(x, y)`
This command divides the first input by the second input.

`MULx(x, y, z)`
This command multiplies the inputs (up to three terms), from left to right.

`SOLVE=` This command outputs the final answer for all previous commands.

`RES#` This variable is used to input the result from the previous command's output.

online

PROGRAMMING EXPRESSIONS

2 SKILL BUILDERS

PLAY A GAME

When you write code, you're writing instructions for the computer to follow. Each line of code gives a different order, or command. Use the commands and variable defined in "Meet MATH++" to turn expressions into lines of code that perform one operation per line of code. You will need to follow the order of operations, which states: First perform operations in parentheses, then compute exponents and roots, followed by multiplication and division from left to right, and finally addition and subtraction from left to right.

EXAMPLE: Write a program in MATH++ to evaluate $4 + 2 \times 5 \times 3$

Step 1 Determine the first command using the order of operations. Since multiplication is the first step, **`MULx`** is the first command. Separate each number with a comma.

`Command 1:` **`MULx(2,5,3)`**

What it does: $2 \times 5 \times 3 = 30$

Step 2 Determine the next command using the order of operations.

`Command 2:` **`SUM+(4,RES#)`**

What it does: $4 + 30 = 34$

Step 3 Once your program is finished, use **`SOLVE=`** to output the final answer.

`Command 3:` **`SOLVE=`**

What it does: Outputs the number 34

So this program in MATH++ would be:

```
MULx (2, 5, 3)
SUM+ (4, RES#)
SOLVE=
```

YOUR TURN

On a separate sheet of paper, use the MATH++ programming language to write code to evaluate the following expressions.

1. Write a MATH++ program for: $10 \times (2 + 5)$
2. Write a MATH++ program for: $(10 \times 6 + 2 + 2) \div 8$
3. Write a MATH++ program for: $30 - 33 \div 3 \times 2$
4. Write a MATH++ program for: $(10 \times 2)^2 + 5$
5. For the expression $(2 + 7 \times 4) \div 3 \times 5$ a student writes this program in MATH++:

```
MULx (7, 4)
SUM+ (RES#, 2)
DIV/ (RES#, 15)
SOLVE=
```

Is their program correct? If not, find their error and debug, or fix, the program.