Computer Science Curriculum: Course Descriptions

IDEAS: Iterate, Design, Engage, Apply, Synthesize

- 6 courses that progress through higher levels of complexity
- Courses available starting with 2nd grade (7 year olds) through high school

IDEAS 1: Exploring Computer Science ●

Students begin learning the basics of the computer, its parts, and their functions. They then learn how to use a variety of software applications to draw and to communicate, how to program animations using Scratch's block-based programming language, and how to design games both physically and digitally.

IDEAS 2: Creativity in Computer Science

Students review the basics of game design and create their own levels of Robomatter's Resolvers game. They then learn about the internet, cybersecurity, and how computer technologies are increasingly accessible. Students' programming skills are enhanced by debugging procedures, and by learning how to write code for fully animating sprites and having sprites engage in dialogue.

IDEAS 3: Interactive Quizzes and Games

Students learn to use Scratch as a tool for communication and entertainment through cross-disciplinary team challenges. Those challenges can be expanded to include research and/or content from any STEM-related topic.

IDEAS Accelerator: Accelerator ●

The Computer Science Accelerator course provides an entry point for older students to begin the IDEAS curriculum. Students learn the computer and programming skills that were taught in previous courses, including the game design and cross-disciplinary challenges.

IDEAS 4: Introduction to Algorithms

Introduction to Algorithms positions Scratch as a tool for computations by showing students how to use lists, functions, and algorithms to create basic, descriptive statistical applications within Scratch. Students then use those applications to analyze the data from a brief survey of their own design.

IDEAS 5: Structure and Function of Programming ●

Now that they know the structures of programming concepts, students learn about the functions in how those programming concepts are applied. The functions of programming concepts are highlighted in a new block-based programming language, Sandbloqs. Sandbloqs allows for a transition from block-based to text-based coding. Students are also introduced to object-oriented thinking which is foundational to object-oriented programming.

IDEAS 6: Creating Computational Artifacts

Students learn about the fundamentals of computational thinking and about the process of software development while working collaboratively to incorporate feedback into their development and testing of computational artifacts. Additionally, students transition from block-based to text-based programming using Sandbloqs, as well as beginning object-oriented programming.

Robomatter INCORPORATED

Courses subject to change

● *Indicates an entry-point course*

412-312-3100 • robomatter.com • STEMsolutions@robomatter.com