Teaching Coding and Computational Thinking: The Basics

www.samlabs.com

Contents

Introduction

Teaching coding in schools ensures that students can understand the technology they use every day. Generation Alpha (born after 2010) is the first generation born entirely in the 21st century, never knowing a world without sophisticated technology. As a result, they can take technology for granted. As technology and artificial intelligence become more and more intertwined with our daily activities, students will learn to create (rather than just consume) the digital systems that power their lives.

Computational thinking is a very important skill that students need to learn not just for their future careers, but for every area of their lives. This 21st century skill is critical to their future success at work and at home.

Understanding the fundamentals of coding has become as important as learning reading, writing and mathematics. While not every students will grow up to become a computer scientist, it's vital that they understand the basics of how computer systems work.

What is computer science?

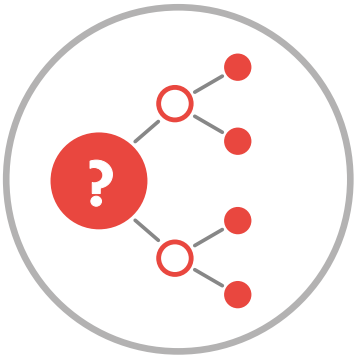
Computer science is the study of the principles and use of computers and how digital systems work. It includes software theory, design and development.

What is computational thinking?

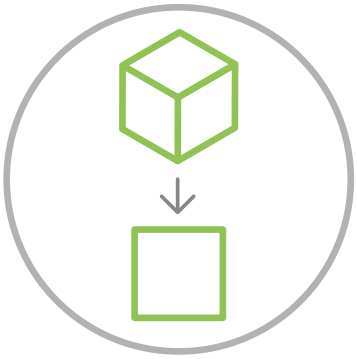
Computational thinking refers to the process used to formulate a problem and express the solution in a way a computer can understand and implement. It's comprised of four parts:

1. Breaking down a complex problem into manageable parts **(decomposition)**
2. Identifying similarities among problems **(pattern recognition)**
3. Focusing on the important part of a problem **(abstraction)**
4. Developing step-by-step instructions to solve the problem **(algorithms)**

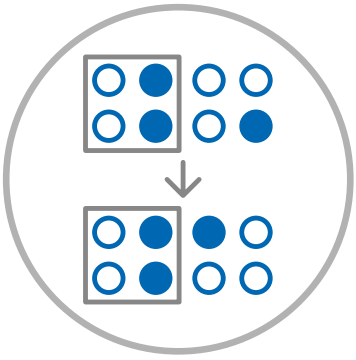
Decomposition



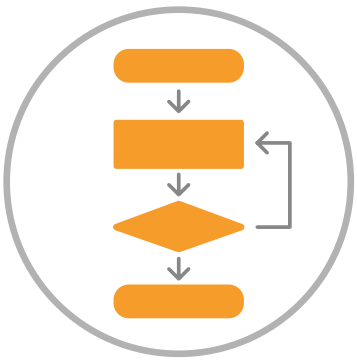
Abstraction



Pattern Recognition



Algorithms



What is computer programming?

Computer programming (or simply, programming) refers to translating desired actions for a computer into terms the computer will understand and activate. In other words, it's how we tell a computer to do something using written code.

What is coding?

Coding is a contemporary term for computer programming, and more specifically refers to writing code. Like programming, it refers to telling a computer how to perform tasks. In recent years, the term 'coding' has become more popular since it's seen to be less intimidating and more open for beginners.

Why should kids learn coding and computational thinking?

Teaching coding in schools isn't just about creating a new generation of software engineers. The fundamentals of computational thinking apply to many areas of development.

Computational thinking is rooted in **problem solving**. While coding is infamous for being frustrating, it's critical to give students the ability to understand that when something isn't working, they have the ability to fix it. This involves breaking complex problems down into sequences of simpler, more manageable tasks. Understanding that some options work to fix the problem, while others won't, enables students to broaden the way they think and persist in looking for a solution. This encourages focus, organization, perseverance and resilience.

Coding is the new **creativity**. Students feel empowered by programming, creating something out of nothing. This encourages experimentation, exploration and discovery and increases student confidence. The possibilities for creation with programming are infinite and students' only limitations are their own imaginations.

Coding encourages **collaboration** and **communication**. Students need to work together to solve problems in the classroom, often sharing tools and devices. They also communicate directly with computers and translate requirements into terms that are actionable and understood.

The history of coding education

Teaching coding in the classroom can be traced all the way back to the theory of Constructivism, created by Jean Piaget in 1936. Constructivism looked at cognitive development to understand how learning happens with children constructing mental models of the world. Piaget didn't believe that knowledge was a fixed trait but rather one that was built based on experiences and interactions. A fundamental part of this theory was based on children learning by doing and exploring.

Educator Seymour Papert later developed these ideas into a programming language called Logo in the 1960s. Logo enabled students to create keyboard commands to produce graphics using a small physical robot, the Logo Turtle. This form of programming utilized major elements of Constructivism: children learn by doing and physical objects support concrete thinking. Papert advocated children being creators and encouraged exploration. He believed that students' curiosity and independence would enable them to take control of their learning experience by using the physical materials around them. This would reduce reliance on teachers and textbooks for answers and encourage independent problem solving. Papert evolved Piaget's theory of Constructivism (learning by doing) into his theory of Constructionism (learning by making).

These key principles were a major part of the development of future hands-on coding technologies. Coding education began appearing in the classroom in the 1980s with the implementation of programming languages like Logo and Basic. This led to the development of Scratch, developed by Mitch Resnick at MIT in 2003. Papert was a mentor to Resnick and Logo's user-friendly nature had a major influence on Scratch.

Fast-forwarding to today, SAM Labs embodies these concepts by encouraging hands-on learning that combines physical and digital elements to create something meaningful. Combining an intuitive, visual app with wireless blocks, SAM Labs introduces the 'learn by making' philosophy in a new way that's highly relevant to Generation Alpha. At the same time, SAM Labs ensures that all activities are rooted in evidence-based learning outcomes that teachers can effectively assess.

About SAM Labs

SAM Labs is an edtech company that empowers teachers with the most engaging STEAM solutions including lesson plans, apps and electronics. Its goal is to inspire every student to discover the fun in coding and creating. By combining highly engaging software with intuitive, user friendly hardware, SAM Labs teaches the fundamentals of coding and computational thinking to children of all skill sets and interests. Offering curriculum-based lesson plans across a variety of subjects — from biology, to probability, to music — SAM Labs creates edtech products to capture every student's imagination.

SAM Labs offers a solution for educators who are teaching computer science by providing in-classroom support for both the early adopter and the tech-hesitant teacher. This is done by offering fully integrated and curriculum-based lesson plans and hands-on educator support. With partners like Intel Education and Microsoft Education, SAM Labs is bringing coding and creating to every classroom.

For more information, please visit www.samlabs.com

©2018 SAM Labs