Coding and Computational Thinking: Teaching 5-7 Year Olds

Contents

Introduction

Today's students are growing up in a world that is very different from the environment their teachers experienced as children. 5-7 year olds have no comprehension of life before tablets, YouTube and online games.

Understanding the fundamentals of coding is now as critical as learning reading, writing and mathematics. While not every student will grow up to become a computer scientist, it's important for everyone to understand how computer systems work since they will be so influential in their lives. At SAM Labs, we encourage building a *shared vocabulary* that students can use to understand, communicate and collaborate with others about coding and technology throughout their academic and professional lives.

There are three main pillars of teaching computing:

1. **Computer science** (the principles of computation and information, how digital systems work)
2. **Information technology** (creating programs, systems and content)
3. **Digital literacy** (the ability to use information and communication technology)

Teaching the basics of coding and computational thinking to 5-7 year olds should include:

- Understanding the concept of algorithms as instructions
- Understanding how algorithms are implemented as computer programs
- How to create and debug simple programs
- Understanding logical reasoning and predictive behavior
- How to use technology to create, organize, store, manipulate and retrieve digital content
- How to recognize IT implementations outside of school
- How to use technology safely, including how to get help when concerned about digital content and how to keep personal information private

Understanding the fundamentals of coding is now as critical as learning reading, writing and mathematics

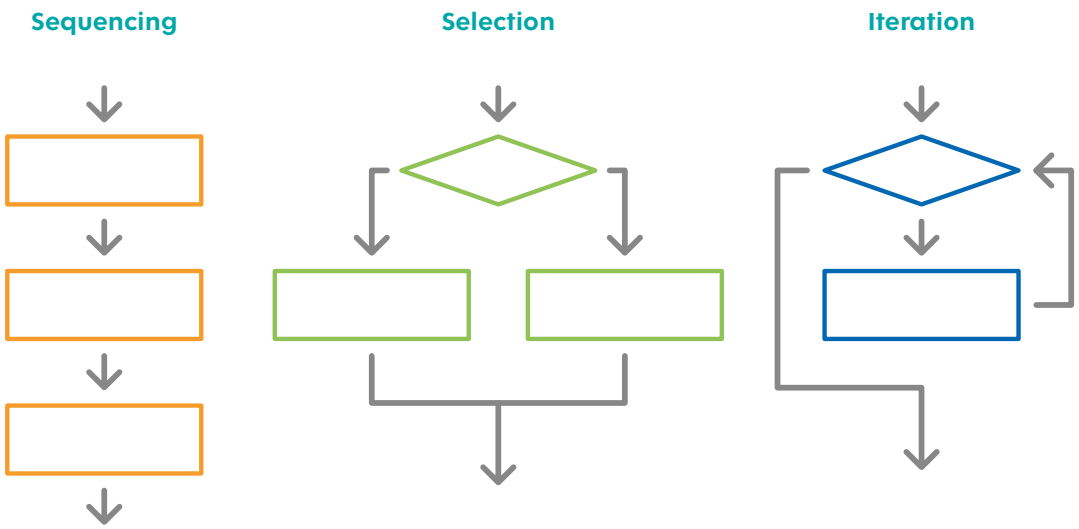
Algorithms

An **algorithm** is a set of instructions or rules to perform a task or solve a problem. For example, this could be a recipe for a meal or instructions to build a chest of drawers. While all correct algorithms should get us to the right solution, we aim to create the most efficient algorithms that get us to the answer simply and quickly.

There are three main components of an algorithm:

- **Sequencing** (the specific order of the instructions in the algorithm)
- **Selection** (a decision made when there are multiple options to choose in the algorithm's steps)
- **Iteration** (repeating instructions over and over again, also known as a **loop** or **repetition**)

With computer programs, we create algorithms to tell a computer what it needs to do. These need to be written clearly, in a language that the computer can understand. Because computers' central processors only understand machine code, computer scientists developed more intuitive coding languages to translate ideas from an algorithm into code that the machine will understand.



Data is made of numbers, words, images, videos and sounds without context. When data is processed by computers, it turns into **information** that can be used to make decisions.

A '**computer**' is not just a traditional PC, it's any device that can accept an **input** (the data a computer receives) and process it using a stored program to produce an **output** (the data a computer sends). These elements are encoded as numbers and this is what makes it a digital device. Digital devices include everything from your tablet to your microwave.

Algorithmic thinking is the ability to define clear steps to solve a problem. It enables students to break problems down into manageable parts (**decomposition**) and come up with solutions as a sequence of steps. This way of thinking is useful across all school subjects and at home.

Creating and debugging simple programs

To create a program, students take an idea for doing something and transform it into a set of instructions that a computer can understand. Students should have a clear idea of what they want the computer to do and how it should be done. Algorithmic thinking is critical here as students need to break down the task into a series of steps.

Usually, programs don't work as they should on the first try. Finding and fixing the mistakes ('bugs') that are causing the problem is called 'debugging.' Students first need to identify that something isn't working, and then determine which part of the program has caused the problem. They then need to use logical reasoning to figure out how to fix it. This is often done through classroom collaboration, with students helping each other identify bugs and solutions.

Debugging code helps students develop critical skills like perseverance and resilience.

Logical reasoning and predictive behavior

Logic enables us to understand and explain why things do the things they do. **Logical reasoning** applies systematic rules to **problem solving**, allowing us understand how computers work and what we can do with them. **Predictive behavior** enables us to predict exactly how computers behave by analyzing historical facts or developing models of how specific software works. We can understand computer programs to understand what they do and how they do it. Students should be encouraged to make predictions about what a program will do before it runs and should be able to logically explain their thinking. Logical reasoning ensures that students are following a set of rules when they are making predictions.

Using technology to create, organize, store, manipulate and retrieve digital content

Students will grow up creating digital content every day. This includes everything from writing messages on their phones to editing selfies to writing computer programs. This content is digitized (converted to numbers) once it's on a computer.

Because we encounter such a huge volume of digital content every day, keeping it organized is incredibly important. Examples of organizing content include tagging photos and organizing files.

Storing content refers to understanding where we've saved anything from a file to a photo to a song to a game on a computer. If we store too much, our computer's memory will fill up. Different types of content take up different amounts of space. Size is measured in bytes, and a byte refers to the volume of information needed to encode a single text character.

Kilobyte (kB) = 1000 bytes

- Example: Short word-processed document = 25 kB

Megabyte (MB) = 1000 kB

- Example: Digital photo = 5 MB

Gigabyte (GB) = 1000 MB

- Example: Digital movie = 4 GB

Terabyte (TB) = 1000 GB

- Example: Data on a computer hard drive = 1 TB

Manipulating content can incorporate more than one application, like photo-editing software or word processors. This includes making changes to digital content to be suitable for its desired purpose.

Retrieving content is the opposite of storing content. This involves knowing where the content was stored and remembering the file name and file type. Computer systems often include search features to make this easier.

Recognizing IT implementations outside of school

Almost every part of our lives is now influenced by technology. Students may experience technology when watching TV, playing a game on an iPad, eating food that is microwaved or traveling in a car that uses GPS. Students should have the ability to recognize that algorithms and programs are powering the technology they experience outside of school.

Using technology safely

Students need to be able to use technology safely and respectfully, understanding the primary risks associated with the internet. They also need to know where to go for help if they have concerns about digital content and should be encouraged to speak to teachers or parents about this. This includes content relating to cyberbullying. Students also need to understand how to keep their personal information private and young children should be taught to avoid sharing any of their personal information online.

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