

Vocab.

Coding Curricula and Lessons

Block Coding: A visual way of representing computer code, where you assemble programs in a drag-and-drop fashion. This makes code easier to learn, interact with, and debug. There are lots of block coding platforms online and tons of interactive robotic toys which utilize block coding as a means of teaching programming. Most of the time, you can also switch over from block code to text, to see how you'd actually write what was block coded in a true programming language like JavaScript. This makes it a great gateway into coding.

Code Club International: This site provides lots of guidance on hosting a coding club. It also has coding projects in 28 different human languages and several different programming languages, including projects for Raspberry Pi. <https://codeclubprojects.org/en-GB/>

Code.org: Code.org is home of the Hour of Code and hosts numerous roughly hour long coding lessons for a variety of ages. Code.org is also home to the App Lab, where you can build and share your own JavaScript app. <https://code.org/learn>

Codecademy: Codecademy has interactive coding lessons on tons of different languages, as well as community support if you get stuck. Great for teens and adults, but younger folks learning to program would be better off starting elsewhere. <https://www.codecademy.com/>

CS First: A lengthy curriculum for kids designed around using the Scratch programming language. It includes full lesson plans and video tutorials. The curriculum was developed by Google and features characters from Cartoon Network. There's a whole lot here and it's a pretty robust place to start. However, it does require having multiple browser tabs open, relies upon Adobe Flash (for now), and doesn't have any self-correcting or in-program guidance that is typical of other early code-learning environments. That might sound like a lot of bad, but there's a ridiculous amount of good on the other side of that equation that more than balances things out. www.cs-first.com/en/home

Hello Ruby: Learning about computers and code through crafts—a very approachable and fun way for young kids to start learning core computer science concepts. <http://www.helloruby.com/>

Kano: Pronounced “Can-oh,” Kano makes a build-it-yourself computer and a build-it-yourself high-definition screen, which are great ways for kids to learn how computer hardware works. As a bonus, once the computer is built, it becomes a super fun and powerful computer programming lab. Their site, <https://kano.me>, also features tons of block coding lessons and projects that can be done in a web browser, even if you're not using your home-made Kano.

Made with Code: Awesome site with projects geared towards inspiring teen girls to learn and use code: <https://www.madewithcode.com/>

Pencil Code: An inspirational way to learn how to code using Coffeescript and an excellent stepping stone to Javascript, HTML, and CSS. <https://pencilcode.net/>

Thimble: An educational web-design environment focused on HTML, CSS, and JavaScript. The site includes projects and lesson plans to get you started: <https://thimble.mozilla.org/en-US/>

Wolfram Alpha Programming Lab: Background: Wolfram Alpha is a self-proclaimed “computational knowledge engine,” and it’s based on the profoundly powerful mathematics platform called Wolfram Mathematica (see, there’s this dude Stephen Wolfram who likes to name stuff after himself). Anyhow, the point I’m trying to make is, there’s this really robust coding environment designed purely for math and science and statistics, and awesomely they have lessons that teach you how to use it in a variety of surprisingly engaging and fun ways called the Wolfram Alpha Programming Lab.
<https://lab.open.wolframcloud.com/app/>

Programming Languages and Related Terms

Bug: A bug is a coding error. Bugs can cause programs to crash (fail catastrophically) or glitch (behave unexpectedly). The process of seeking out and correcting bugs is called “debugging.” Bugs are easier to find in well-commented code.

CoffeeScript: Modified JavaScript that’s easier and more efficient to read and write. Whee! Coffee.

Compiler: A compiler transforms the human-friendly code you write into a set of machine-friendly instructions. You really shouldn’t worry much about them, just be grateful they exist. =)

CSS: Cascading Style Sheets (CSS) set the visual style of a site and allow you to make layout and formatting changes through templates instead of by changing parameters and settings in the HTML of all the different pages. CSS reduces complexity and repetition while improving accessibility over using HTML alone. Super handy!

GitHub: GitHub is a place where people share open-source coding projects and help each other out. Go Git some code, friends, and share your stuff if you’re of a mind to!

HTML: Hypertext Markup Language (HTML) is the base language providing the structure and content of websites. HTML is fairly eye-readable and easy to learn and write. Even if you won’t be writing much raw HTML, it’s useful to be familiar with because of its ubiquity and interaction with CSS and JavaScript.

Java: Just generally avoid Java on anything Internet-connected, really.

JavaScript: Not to be confused with Java, JavaScript makes dynamic and interactive content on websites possible. It’s a bit more complex than HTML or CSC, so it’s typically the third web-related language you’d learn when delving into web development.

JSON: JavaScript Object Notation (JSON) allows JavaScript to do cool web-stuff much more reliably and safely than would otherwise be possible. Often it’s learned right after JavaScript.

Notepad++: A nice open-source text editor to write code in.

Open Source: Open-source code is freely available to review, use, edit, and repurpose. Releasing things open source is very nice and librarian-like.

Python: A popular, powerful, and human-friendly coding language used by Wikipedia, Google, CERN, Reddit, and NASA. Raspberry Pi and the One Laptop per Child XO also utilize Python. Python scripting is common in the Open Source community (numerous Linux distributions and LibreOffice use Python).

Ruby on Rails: A web application framework that plays nicely with JavaScript, JSON, HTML, and CSS. It makes backend database incorporation into websites much more manageable. I'm reasonably certain Ruby on Rails was invented by someone who was tired of mucking about with arrays.

Scratch: An introductory programming language out of MIT in which the code you write controls a cat. It's based on an earlier instructional coding language called Logo. It's easy to learn and a fine stepping stone to other languages. It can also produce algorithmic math geek art. <https://scratch.mit.edu/>

XML: Extensible Markup Language is sort of a meta-language that countless other formats were built on, including the CONTENTdm metadata used in Digital Horizons. It's more like a grammar than a language.

Control Structures (Things that Make Code Flow)

Conditional Statements: Used to execute one set of instructions in certain circumstances and another (or nothing at all) in others. Typically written as an "if" or an "if, else" statement and utilizing sequential or Boolean logic to determine which path to follow.

Event-Handling: When you're writing code with human interaction in mind (and you will be), you'll want to be able to respond to certain events. An event can be something like a mouse click or a robot colliding with a wall; either way, if you wrote code for that event, there will be feedback or a response when it occurs. Note that this is a totally asynchronous way to program—the code is just sitting there waiting for something to happen, and won't otherwise execute. Neat.

Function: A piece of code that, once written, can be invoked elsewhere in a program. This is a way of reusing instead of rewriting script and is part of modular design. Using functions is good practice, efficient, and dead sexy.

Library: From the perspective of computer programming, a library is a collection of commonly useful formulas that can be loaded into a program so you don't have to write them in. Libraries are great!

Logic: Logical operators are used in a lot of programming languages. There are two basic types, Sequential and Boolean. Sequential logic is perhaps more intuitive, as it is based on quantity and inequality. Sequential logic statements are: greater than (>), less than (<), equal to (==), not equal to (!=), greater than or equal to (>=), and less than or equal to (<=). Boolean logic is a little trickier. The most common expressions are: or, and, not, xor (exclusive or), and nand (not and). This probably seems weird, but let's take a look at the truth table below to see how this might be useful (note that typically false equates to 0 and true equates to 1, for coding purposes):

x	y	x OR y	x AND y	NOT X	NOT y	x XOR y	x NAND y
0	0	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE
1	0	TRUE	FALSE	FALSE	TRUE	TRUE	TRUE
0	1	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE
1	1	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE

How to render this in human-friendly terms? OR is true when either condition compared is true. AND is true only when both conditions compared are true. NOT turns things on their head: NOT x is true when x

is false and it's false when x is true (this is why it's also called an inverter). XOR (exclusive or) is true when one and only one of the conditions compared is true. NAND (not and), is true whenever an AND statement would be false.

Looping: A loop is something that repeats or recurs. It's a control structure used to execute repetitive tasks simply and elegantly. Loops are often coded using "while," "do while," or "for" with a specified number of iterations or a condition that must be true for the loop to stop.

Modular Design: The process of breaking a problem into chunks and tackling them separately in reusable pieces of code. Oftentimes, these modules are formulas.

Nesting: Placing conditional statements within conditional statements or loops within loops to make more complex, powerful, useful, (and confusing!) control structures. Parentheses abound.

Recursion: It is possible, using other control structures like conditionals and loops, to have a function call itself. This is known as recursion and it's sort of like nesting for functions.

Numbers and Letters

Array: An array is basically an arrangement of things in rows and columns that can be referred to by coordinates. They become really useful, but are super confusing when you're starting out. Picture a three-by-three matrix containing...actually scratch that, you don't have to, I'm going to illustrate it:

37	"Capybara"	FALSE
"tacos"	42	0
99.44	"Zed"	""

This is a visual representation of a 3x3 array of numbers, strings, and Boolean expressions. When you're coding, you never actually see it like that, so it's much more abstract. Crazy, right? Yep. It's pretty crazy. And in this case not particularly useful for much of anything. Just take my word, if you delve deep enough into coding, you will eventually get to arrays and want to stab yourself with a pencil.

Binary: At their heart, computer instructions break down to binary, a series of ones and zeros, on's and off's. Binary is a 2-based number system using bits, whereas the number system customarily used is 10-based and uses digits. Instead of positionally representing ones (10^0), tens (10^1), hundreds (10^2), and thousands (10^3), etc., bits represent ones (2^0), twos (2^1), fours (2^2), eights (2^3), etc. So a base-10 7 would equate to 111 in binary ($4+2+1$). The binary number 100101 represents one 32, one 4 and one 1, so this equates to the digital number 37.

Comments: Comments are elements in code that are just there for us humans, and they help explain what's going on and how different segments of modular code operate and can be re-deployed. They're usually broken out by // or contained within a /* and a */. These never get compiled into machine instructions and are completely ignored by computers. // *The robots have no idea what I'm saying now!*

Constant: Constants don't change values. They also don't get their own names. They just are. Like 12. Or "hot dog." The values assigned to variables are typically constants, though it's probably not helpful to think about that.

Hexadecimal: 16-based number system (0-F) that is often used as a shorthand for binary, since one hexadecimal or hex character represents four binary bits.

Incrementation: Oftentimes when you write a loop, you want it to execute a certain number of times (specified either by you or a user). To do this, you'd use a variable as a counter and increment it each time the loop is run. The shorthand way of coding this is ++ (plus plus), so for instance if you name your counter variable "c" you'd type c++ into the loop to increase the value of c for each iteration. That's where the ancient (though not abandoned) computer language C++ got its name from. Neat, right?

Scope: The range in which talking about something makes sense is its scope. In complex bits of code, variable and functions have limited scopes and can't be accessed, invoked, or altered from parts of the code that are beyond their scope. This is kinda higher level, but it's good to be aware of.

String: In code-speak, a string is basically data that is textual, as opposed to being numeric or Boolean (true/false). Almost all of the time, strings are contained within quotation marks. "Hello world!" is a string. "I have hamsters in my pockets!" is a string. Strings are fun!

Syntax: A language's syntax consists of the specifications for how statements and formulas must be written and how punctuation and symbols are used. Syntactical errors result in bugs.

Variable: A variable is a thing whose value can change, but which is referred to by an assigned name. This is basically the point where coding gets all algebra on you. I'm sorry if you just threw up in your mouth a little.

Zero: While humans tend to start counting at 1, machines like to start counting at 0. This causes bugs.

Hardware and Technology

Arduino: Open-source computer hardware that is the basis of numerous do-it-yourself (DIY) kits. If you want to learn about and explore the hardware side of things affordably, Arduino is for you. Arduino can also be used in prototyping, so they show up in lots of MakerSpaces, too.

Bluetooth: A means of device-to-device communication across short distances using an ultra-high frequency radio band. For our purposes, it's what allows us to program robots with our tablets and smartphones. This technology also lets you talk to your loved ones through your car, like a boss.

Raspberry Crazy Ants: *Nylanderia fulva*, commonly called the Raspberry Crazy Ant (named after exterminator Tom Raspberry, whom they presumably ate), are a species of ants attracted to electrical equipment. They are known to chew through insulation and wiring, electrocuting themselves and releasing an alarm pheromone in the process. This attracts other ants to the area in search of attackers. Mechanical failures and system short circuits are a common result. These pests, which will eventually end civilization as we know it, are currently only found in parts of Texas, Louisiana, Mississippi, Georgia, and Florida, though they're spreading faster than you'd imagine ants could.

Raspberry Pi: A series of variously-featured ultra-affordable single-board computers developed to promote teaching basic computer science. As the starting price-point is \$5 (\$10 with Bluetooth and WiFi) and the hardware is flexible, powerful, and diminutive, they've found innumerable unanticipated uses and something of a DIY darling, having sold over 11 million units to date.