**Hour of Code**

**Vocabulary**

**abstraction**

Pulling out specific differences to make one solution work for multiple problems students.

*(Course 3: Lesson 1)*

**algorithm**

A list of steps to finish a task. A set of instructions that can be performed with or without a computer. For example, the collection of steps to make a peanut butter and jelly sandwich is an algorithm.

*(Course 1: Lessons 1, 6 | Course 2: Lessons 1, 2 | Course 3: Lessons 1, 10)*

**binary**

A way of representing information using only two options.

*(Course 2: Lesson 14)*

**blockly**

The visual programming language used in Code.org's online learning system for K-5 students. *(Course 1: Lesson 3)*

**bug**

An error in a program that prevents the program from running as expected. *(Course 2: Lesson 9)*

**code**

One or more commands or algorithm(s) designed to be carried out by a computer. See Program. *(Course 1: Lesson 2)*

**command**

An instruction for the computer. Many commands put together make up algorithms and computer programs. *(Course 1: Lesson 2)*

**computational thinking**

Mental processes and strategies that include: decomposition, pattern matching, abstraction, algorithms (decomposing problems into smaller, more manageable problems, finding repeating patterns, abstracting specific differences to make one solution work for multiple problems, and creating step-by-step algorithms). *(Course 3: Lesson 1)*

**computer science**

A field in which people use the power of computers to solve big problems. *(Course 1: Lesson 2)*

**conditionals**

Statements that only run under certain conditions or situations. *(Course 2: Lesson 12)*

**crowdsourcing**

Getting help from a large group of people to finish something faster. *(Course 3: Lesson 19)*

**data**

Quantities, characters, or symbols that are the inputs and outputs of computer programs. *(Course 2: Lesson 14 | Course 3: Lessons 1)*

**debugging**

Finding and fixing errors in programs. *(Course 2: Lesson 9)*

**decompose**

Break a problem down into smaller pieces. *(Course 3: Lesson 1)*

**digital citizen**

Someone who acts safely, responsibly, and respectfully online. *(Course 3: Lesson 20)*

**digital footprint**

The information about someone on the Internet. *(Course 2: Lesson 18)*

**DNS (domain name service)**

The service that translates URLs to IP addresses. *(Course 3: Lesson 18)*

**DSL/cable**

A method of sending information using telephone or television cables. *(Course 3: Lesson 18)*

**event**

An action that causes something to happen. *(Course 1: Lesson 15 | Course 2: Lesson 15)*

**event-handler**

An action or event that is being constantly monitored for by the computer. When you write code for the computer to perform after that an action occurs, that code, the event-handler, will run every time the action is performed without having to put the commands inside of a loop. Many event-handlers are human-initiated. For example: an event handler might respond when the user clicks the mouse by making the bird flap its wings. “When the mouse is clicked" is an event-handler. *(Course 1: Lesson 15 | Course 2: Lesson 15)*

**fiber optic cable**

A connection that uses light to transmit information. *(Course 3: Lesson 18)*

**function**

A piece of code that you can easily call over and over again. Functions are sometimes called ‘procedures.’ A function definition is a segment of code that includes the steps performed in the function. A function call is the code segment, typically within the main logic of the program, which invokes the function. *(Course 3: Lesson 4, 9)*

**Internet**

A group of computers and servers that are connected to each other. *(Course 1: Lesson 17 | Course 3: Lessons 18, 20)*

**IP address**

A number assigned to any item that is connected to the Internet. *(Course 3: Lesson 18)*

**iteration**

A repetitive action or command typically created with programming loops. *(Course 1: Lesson 12 | Course 2: Lesson 5)*

**loop**

The action of doing something over and over again. *(Course 1: Lesson 12 | Course 2: Lesson 5)*

**packets**

Small chunks of information that have been carefully formed from larger chunks of information. *(Course 3: Lesson 18)*

**pattern matching**

Finding similarities between things. *(Course 3: Lesson 1)*

**persistence**

Trying again and again, even when something is very hard. *(Course 1: Lesson 9)*

**program**

A program is an algorithm that has been coded into something that can be run by a machine. *(Course 1: Lesson 2 | Course 2: Lesson 1 | Course 3: Lesson 10)*

**run program**

To have the computer execute the commands you've written in your program. *(Course 1: Lesson 4)*

**servers**

Computers that exist only to provide things to others. *(Course 3: Lesson 18)*

**toolbox**

The tall grey bar in the middle section of Code.org's online learning system where all the commands you can use to write your program are displayed. *(Course 1: Lesson 4)*

**URL (universal resource locator)**

An easy-to-remember address for calling a web page (like [www.code.org](http://www.code.org/)). *(Course 3: Lesson 18)*

**username**

A name you make up so that you can see or do things on a website, sometimes called a “screen name.” *(Course 1: Lesson 18)*

**variable**

A placeholder for a piece of information that can change. *(Course 3: Lesson 4)*

**Wi-Fi**

A wireless method of sending information using radio waves. *(Course 3: Lesson 18)*

**workspace**

The white area on the right side of Code.org's online learning system where you drag and drop commands to build your program. *(Course 1: Lesson 4)*