LESSONS 7 & 8

Now that your students have completed Lessons 1-6, they should have a solid grasp of the core coding concepts we've covered and of Hopscotch as a tool. These lessons are designed to give students an opportunity to practice their skills and apply them in new ways. You can determine the right amount of structure based on your class's experience and needs. Younger kids might need more concrete assignments, whereas older students may relish the opportunity to make something from their imagination.

An important idea to explore in creative coding is identifying which problems you can solve with a computer (a computable problem) and which are better solved by a human. For example, a video game can easily be solved by a computer, but a mind-reading game cannot.

Here are some suggestions of how you can guide your class; do what is best for your goals:

Go deeper with games from previous lessons:

- Refine one of the games you made in lessons 1-6
- Design and make your own game, using pieces from these lessons
- Form a game dev team and make a big game together, assigning roles to each team member
- Have a game showcase with another class

Link the coding experience to other subjects:

- Write a review of someone else's game
- Make a commercial for your game: video, website, magazine ad
- Make a video tutorial for how to use a certain block, or define a term
- Write a persuasive essay about two different ways to do something, and which one is better
- Record your project as an animated gif (make a meme)

Explore Hopscotch in greater detail:

- Do the tutorial videos in the app
- Find a game you like in the Community and remix it
- Design a game as a team and implement it, using the Forum to ask for help

	Unsatisfactory	Competent	Proficient	Distinguished
Execution	Program does not work, or has major flaws that prevent its intended use	Program mostly works, and has only minor flaws	Program works in the way the student intended	Program is functional and refined, with extra features that add functionality or improve upon the original design
Content	Program lacks understanding of core concepts and skills	Program shows some understanding of core concepts and skills	Program reflects understanding of concepts and skills	Program shows synthesis of new and old concepts and skills
Reflection	Student cannot describe how their code works	Student can mostly describe how their code works	Student can describe how their code works and can make changes that have desired effects	Student can describe how their code works and how they wrote it, and help others debug their code
Habits of mind	Student is not aware of the goal of the program, is frequently off-task, does not offer their own ideas, and gives up when it is difficult	Student is aware of the goal of the program, returns to the task when asked, has some ideas when prompted, asks for help when stuck	Student understands the goal of the program, has their own ideas, rarely goes off-task, and attempts to solve problems first before asking for help	Student embraces the goal of the program and chooses to try out new ideas and multiple solutions, even when they are challenging

Inspired by http://www.edutopia.org/pdfs/blogs/edutopia-yokana-maker-rubric.pdf

GLOSSARY FOR YOUNGER STUDENTS

Ability: Code that can be reused

Algorithm: A recipe for a program

Coding: Telling computers what to do

Concurrence: Two things happening at the same time

Conditional: Statements of the form "IF (something is true), THEN (do an action)"

Debugging: Finding mistakes in your code and fixing them

Event: When something happens

Iteration: Having ideas and making mistakes, over and over

Logic: The process of making decisions

Loop: Code that repeats

Operator: A mathematical symbol that makes an equation

Program: A set of instructions a computer can understand

Programmer: A person who writes programs

Programming Language: A set of rules or blocks that can be used to write any program

Random: When there's no pattern

Range: The highest and lowest number random can choose between

Rule: Instructions that tell your computer what to do (the command) and when to do it (the event)

Sequence: The order in which instructions are given to the computer

Object: A character or text with its own rules

Value/Variable: A holder for a number

GLOSSARY FOR OLDER STUDENTS

Ability/Function/Procedure/Subroutine: A saved set of blocks. What we call abilities in Hopscotch are known as functions or subroutines in other programming languages. Easily replicable routines are a key concept in computer programming, and allow you to scale your code and create complex programs

Algorithm: Algorithms are at the heart of computer science; they are the recipes that computers follow to solve problems

Bug: An error that a programmer has made in his or her code

Coding: Writing the rules of behavior for a computer to follow automatically; programming

Concurrency: Two or more things happening at the same time, or triggered by the same event

Conditional: Statements of the form "IF (something is true), THEN (do an action)"

Debugging: Finding mistakes in your code (bugs) and fixing them

Event: A trigger that the computer recognizes and causes it to do some action. In Hopscotch, events include "When the iPad is tapped" or "When the play button is tapped"

Iteration: The repetition of a process

Logic: The science of the formal processes of thinking and reasoning

Loop: A repeating set of instructions

Operator: A mathematical symbol that produces a value

Program: A set of instructions a computer can understand

Programmer: A person who writes programs

Programming Language: A set of words, rules, blocks or instructions that can be used to write a program

Random: Any number or item among a set. The lack of a pattern among items in a set

Range: The highest and lowest number random can choose between

Rule: Rules tell your object what to do and when to do it. When you make an ability and pair it with an event, you create a rule

Sequence: An ordered list of things (instructions, blocks, numbers, etc) which can be triggered by an event or repeated

Object: A character or text with its own rules on screen

Value: A holder for a number. Also known as a variable

REFERENCES

Wiggins, Grant P., and Jay McTighe. Understanding by design. Ascd, 2005. https://books.google.com/books?id=hL9nBwAAQBAJ&dq=understanding+by+design+apple+unit

https://computationalthinkingcourse.withgoogle.com

http://csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf

http://www.corestandards.org/Math/Practice/

http://www.nextgenscience.org/sites/ngss/files/Appendix%20F%20%20Science%20and%20Engineering%20Practices%20in%20the%20NGSS%20-%20FINAL%20060513.pdf

http://www.edutopia.org/pdfs/blogs/edutopia-yokana-maker-rubric.pdf

ACKNOWLEDGMENTS

Huge thanks to Dr. Emily Thomforde, David Dulberger, Jesse Beutow, Thomas Abend, Ashley Gavin, Miranda Gohh, Elizabeth McDonald, Jessica Wertheim, Redwood City Public Library, Taft Community School, and all of the educators who have provided ideas for and feedback on this curriculum.

Most importantly, thanks to Hopscotchers everywhere for making amazing things and reminding us that learning should be fun. This curriculum is inspired by and dedicated to you <3.