Intro to Coding

Robious Elementary Winter 2017

Jan 10 - Feb 16 4-5PM

Class Size: 16

1/12/2017 Day 1

- ❑ **Objectives:**
  - ❑ **Intro/goals**
  - ❑ **Understand what technology is**
  - ❑ **What is Programming (coding)**
  - ❑ **hardware/software**
  - ❑ **input/output devices**
  - ❑ **Start Lightbot**

- 4:00 - 4:10: Introductions & Objectives
  - Share names and grades
  - Has anyone programmed (or coded) before?
  - What is your favorite thing to do on the computer (or iPad)?

- 4:10 - 4:20 What is Technology, What is Programming
  - Ask the class what they think technology is
  - Technology is anything created by humans to make our lives easier and to solve problems
  - Discuss some examples of technology

- 4:20 - 4:30 What is Programming, Class Objectives
  - Ask class, have you followed directions before?
  - Some technology comes with directions (step-by-step instructions on how to put it together or use it
  - Some technology comes with directions already inside of it (i.e. computers, phones, iPads)
  - To tell a computer, or phone what to do, someone had to program or code the computer; tell it what to do step-by-step.
  - Discuss to class, we are here to learn how to code!

- 4:30 - 4:40 Draw a Computer
  - Have class draw what they think is a computer, try to label parts of their drawing
  - Ask what is software? What is hardware?
  - Hardware: the physical stuff; the ‘hard’ things
  - Software: programs; instructions to tell a computer what to do
  - Ask if they see hardware in their drawings (should be all of it)

- 4:40 - 445 Input/Output devices
  - Ask how can we interact with a computer?
  - Discuss input/output devices; keyboard, mouse, monitor, printer, etc.

- 4:45 - 5:00 Lightbot
    - Iterate again that to program is to give a computer step-by-step instructions to do something
    - Quickly explain Lightbot; giving a robot instructions to light up the blue squares
        - It cannot do this on its own, you have to 'program' it

Were Objectives met?

1/19/2017 Day 2

- ❑ **Objectives:**
    - ❑ **Simulate a Program; Fetch, Decode, Execute**
    - ❑ **Efficiency**
    - ❑ **Functions**
    - ❑ **Split into different level/age groups if need to**
        - ❑ **Older: Lightbot**
        - ❑ **Younger: Kodable**

- 4:00 - 4:10 Review
    - Ask class difference between hardware and software
    - What is a computer program?
    - I/O devices?

- 4:10 - 4:20 Simulate a Program
    - What happens when you start your favorite game?
    - Simulate a Program exercise
        - Choose three *volunteers*
        - A program, bus, and CPU

    - Explain Fetch, Decode, Execute cycle
        - Fetch: instructions from program are fetched
        - Decode: the instructions are understood by the CPU in binary
        - Execute: The CPU carries out the instructions
        - This happens when you start your favorite game and happens from the moment you turn the computer on.

- 4:20 - 4:30 Simon Says/Efficient
    - Do a simple demonstration of efficiency using the Simon Says game
        - Quickly explain the Simon Says game (they should all know it already? Too old now?)
        - Pick two students
        - Give them instructions to go to the same place:
            - "Simon Says, walk five steps; simon says turn right…"
            - but, have one complete it using a lot more unnecessary steps than the other one
    - Ask class which set of instructions was more efficient?

    - Explain what it means to be efficient

- 4:30 - 4:35 Functions (Procedures, or even a routine)
    - Ask class about their morning routine
        - Do they find themselves doing some of the same tasks every morning?
        - Do they notice they might do them in mostly the same order every morning?
    - Briefly explain procedures in Lightbot
    - If you see a pattern you can reuse, put that pattern of instructions in a procedure folder
    - Procedures = Functions; their purpose is to reuse code to be more efficient

- 4:35 - 5:00 Lightbot/Kodable
    - Lightbot: Students should get to the point of completing a couple levels involving procedures
    - Kodable: Students should get to at least Conditions

Were Objectives met?

1/26/2017 Day 3

- ❑ **Objectives:**
    - ❑ **Conditions**
    - ❑ **Overloading**
    - ❑ **Review Functions**

- 4:00 - 4:10 Conditions
    - Ask the question, Have your parents ever told you,
        - "**IF** you eat your vegetables, **THEN** you can have desert."
        - "**IF** you finish your homework, **THEN** you can play with your friends."
    - Those are condition statements
    - Before one particular thing can happen, something else has to happen before it
    - Think of the sets of instructions you have been coding in Lightbot or Kodable
    - The program always runs in the order you specify; it never jumps around.
    - After the first instruction is carried out, then the second one can be carried out, and so on.

- 4:10 - 4:25 Overloading
    - Ask the class what they can do with their voice
        - If confusion, provide some examples: talk, sing, yell, whisper, etc. with your voice
        - So, your voice can do a range of things, not just one thing, it depends on the situation.
    - Explain that this is overloading
    - Overloading in programming means that one instruction can do more than one thing

- 4:25 - 5:00 Lightbot/Kodable
    - Lightbot: Overloading
    - Kodable: Conditions, Loops

Were Objectives met?

2/2/2017 Day 4

- ❑ **Objectives:**
  - ❑ **Algorithm**
  - ❑ **Sequence**
  - ❑ **Loops**
  - ❑ **Hopscotch**

- 4:00 - 4:15 Algorithm-Sequence Exercise
  - Discuss with the class on steps for pouring a bowl of cereal
    - What if the steps were slightly out of order? For instance, the milk is poured before the bowl is even put out
  - Discuss with class that this is an example of an algorithm; step-by-step instructions to complete a task or solve a problem.
  - Remember, computers (or lightbot) do what they are told, only in the order they were told. This is known as a sequence, an order of events.
  - Just like with pouring a bowl of cereal, with computers, it is important to give instructions in the right order, or else the program won't run as you expected.
  - A Sequence is important in a computer algorithm because the correct order of steps is needed in order to make the algorithm work.
  - Iterate computers do what they are told, in the order they were told.

- 4:15 - 4:25 Hopscotch Intro
  - Explain an Event, a trigger that the computer recognizes and causes it to do something.
  - In Hopscotch, you have to specify a certain event, so the computer knows when to run your program.
  - To quick run-through of the app
    - Create a new project
    - Drag and drop a character
    - Tap the character to 'see code': This is where you give that object a sequence of instructions
    - Have them choose an event to let the computer know to start the program, give their character an instruction (move forward), and test it.

- 4:25 - 4:35 Algorithm to draw a square
  - First challenge: have class finish the instructions for their character to draw a square. Challenge them to do it by giving only a couple of instructions.
    - Explain Loops; they repeat a set of instructions
    - Point out, they can find the Loop command under "Control Flow"

- 4:35 - 5:00 Create an Algorithm to control your character's movements with buttons
  - Second Challenge: Have the students bring drag out a second character and give instructions to that character to move wherever/whenever they tap a particular button

- Point out the arrow button emojis
- Have them drag out the left, right, up, down arrow emojis (setting up a control pad)
- If time is left over have them make their character to collect some object

Were Objectives met?

2/9/2017 Day 5

- ❑ **Objectives:**
  - ❑ **Computational thinking: How to solve problem**
    - ❑ **Divide & conquer (decomposition): break up a large problem into smaller parts**
    - ❑ **Generalize: seeing the bigger problem**
    - ❑ **Recognize Patterns: decide which things to repeat**
    - ❑ **Make a plan (Make an algorithm): the process to solve the problem**

- 4:00 - 4:15 How to Solve a Problem
  - Refer to the slides - **[better results when you draw it out on whiteboard]**
  - Propose the problem: Your whole house is a complete mess, your parents tell you to clean the whole house by yourself and tell you to have it done by a specific time.
  - At first this seems overwhelming!
  - Break the problem up into smaller more manageable parts
  - Think about focusing on completing those small parts and see how they make up the larger problem
  - See if you can reuse some steps
  - Make a plan to totally solve the problem, then carry it out

- 4:15 - 5:00 Make “Crossy Road” game on Hopscotch
    - *10min*
  - Pass out iPads; have students create a new project on Hopscotch
  - Add character object and place at bottom of screen
  - Add direction emoji buttons
  - In Hopscotch (0,0) is at the bottom left of your screen
  - Have kids set up the instructions for each arrow emoji, so they control the main character
  - Encourage using the “change X by..”, ”change Y by..”, etc.

    - *15min*
  - Propose challenge to have cars driving back and forth within the area above the character
  - Explain sequences and loops, again
  - In reference to sandwiches: consider using a loop to repeat the sandwich making process: For the number of sandwiches I need: open the tuna, add mayo, stir, put on bread, put in a bag.
  - As a class, discuss the behavior of these cars and together make a list of the steps they take. Ask students to consider the difference between using “Repeat 10 Times” and

"Repeat Forever". Which is appropriate for the sandwich? Which is appropriate for the car's movement? Also, consider what happens if instructions are out of order.
    - Draw out on the board how the sequence/loop might look like for one of the cars
        - *10min*
    - Discuss Randomness
    - We can make the game more interesting by have the cars move at random speeds
    - Randomness depends on giving the computer options to choose from, or a range.
    - Relate to the random number guessing game: "Pick a number between 1 and 10"
    - Show students the Set Speed function -refer to slides. Have them change up the values and see what happens
        - *10min*
    - Have students create more car obstacles with similar behaviours

Were Objectives met?

2/16/2017 Day 6

- ❑ **Objectives:**
    - ❑ **Finish Computational thinking exercise: How to solve problem**
    - ❑ **Conditions/Events**

- 4:00 - 4:05 Review the challenge
    - The challenge (or problem) is to make a game where you can control a character's movements to cross a busy road with cars driving by at random speeds
    - Let's finish this challenge

- 4:05 - 4:20 Condition Statements
    - Review what conditions/events are
    - Right now, nothing happens when the character collides with a passing car
    - Add some conditions statements to change that
    - Add them to the main character, add them to account for each car

- 4:20 - 4:35 More Conditions
    - Set up something to trigger an event to win the game
    - How do we win? When the character crosses the road safely, maybe have the character collide with an object (that is not a car) on the other side of the road.
    - Challenge is to set-up a *win the game event*

- 4:35 - 4:40 Iterate the problem solving process we took
    - Briefly discuss how we broke up the game into individual tasks to make the game as a whole more manageable
    - We used concepts we've learned throughout i.e. sequences, conditions, loops, events, etc.

- 4:40 - 5:00 Complete the Crossy Road project
    - Have students finish the project
    - Discuss what other things they might add to the game to make it more fun or challenging and implement those changes
    - Students can either continue with this project or make a new project/game
    - For those who are done with Hopscotch, offer Lightbot or Kodable to finish class

- Pass out shirts!

Were Objectives met?
