**ACTIVITY: Intro Programming**

BY: Sam Germain adapted from Carter Hill

GRADE and CAMP: Grade 7-9, Codemakers (Love 2D)

TOPIC(s): Game Design, binary

|  |
| --- |
| TIME: 90 mins |

OBJECTIVE: Learning the basics of programming in Lua

**MATERIALS:**

* ヽ(ຈل͜ຈ)ﾉ︵ ┻━┻

**SCIENTIFIC BASIS (learning outcomes - teach this):**

**What is syntax?** All coding languages are unique in how they operate, and the word syntax is used to describe them. For example the “syntax” for writing something to the console in Python or Lua is “print(“something”)” whereas with c++ it’s more like: cout << “something” << endl; In the same way, one could say the “syntax” for saying hello in French is bonjour.

**Variables:** Variables, put simply, are placeholders for data. They have an assigned name and a value that can be changed at almost any time. Variables in coding are not very different from variables in math. If we say that x – 3 = 1, we know that x = 4, x would be its name, and 4 would be its value. In Lua, variables can be assigned really easy, just by typing name = value. If there is already a variable called name, its value would be changed, and if not, it would simple create a value for it.

Variables can hold the values of all kinds of different data types. The most obvious answer would be numbers. What other types of data can be used in computer code? If you want a string (a string of characters, like a sentence) you would say myString = “Your sentence in quotes”. The quotations make the computer know that it’s a string variable type. Another important data type is Boolean, which can only have two values, can you guess what they are? (It’s True and False). myBoolean = True

**Conditionals:** Are bits of code that are only executed if a given condition is true. This is where computer and human language are most comparable. Let’s say you’re about to go get groceries you check your fridge and see that you’re out of milk, so you add it to your list. The next time you go to get groceries you still have some left, so you decide not to get milk. Next, somebody else in your house is going shopping, so you tell them “If we’re out of milk buy some!” which is very close to how conditionals are written in computer science. Let’s say we have a variable that is true if the milk jug is empty, so “outOfMilk = True”. Then, our if statement would look like: “if outOfMilk then // end” or “if outOfMilk == True”. Or we could have a variable that is true if we have milk, so: “if milkNotEmpty == False then // end”.

You can also use numbers in conditionals. For example, if 4 == 4 would always turn true, and thus the code in that block will always execute. Or you could have a variable that shows how much milk is left, like milkAmount = 10. You could say that if the milk is almost gone, you should buy more. You can do that by using < and > which mean less than or greater than. So “if milkAmount < 3 then” would execute if your “milkAmount” is two or less.

**Functions:** Functions, or procedures, are kind of like variables but for code. Let’s say we want a function that adds four to a given number. It would look like:

function add(x, y)

z = x + y

return z

end

And then if I wanted to add two numbers together and assign it to a variable I would simply type myVariable = add(4, 3). Functions can have as many arguments as you want them to, or they could have 0 arguments at all. If they have no arguments, you have to remember to use the brackets anyway, there just won’t be anything between them.

**Objects**

Objects behave like a container that can have variables(called attributes) and functions(called methods) attached to them. Lets use the analogy of a bike as our object. If we refer to samsbike.height we are referring to the height of sams bike, and if we refer to samsbike.speed we are referring to the speed of sams bike. If we say mattsbike.speed we are referring to the speed of matts bike.

**Loops**

Our for loops are only going to loop through objects, because of this, they don’t have to really learn what for loops are, just how to loop through an object. In looping through an object we repeat the same series of instructions for each attribute of the objects. We LOOP through each of the attributes of the object and perform some code using each attribute

**PROCEDURE:**

Get the kids to watch this video <https://www.youtube.com/watch?v=l26oaHV7D40&index=13&list=PL8dPuuaLjXtNlUrzyH5r6jN9ulIgZBpdo> on crash course programming.

The following exercises will take place on <https://repl.it/repls/GloomyGummyGenericsoftware>. Get the kids to visit this site

Write each of the following examples on the board and get the kids to try to code their own example

Write this example of variable’s on the board and get them to try to code their own example

**Variables**

x = 3

print(x) -- should print 3

y = 4 + x

print(y) --should print 7

**Conditional**

x = 3

if (x<5) then

print("less than 5")

elseif (x<10) then

print("less than 10")

else

print("greater than 10")

end

**Functions**  
function add(x, y)

z = x + y

return z

end

x = 2

y = 3

w = add(x,y)

print(w) --Should print 5

**Objects**

mattsbike = {}

samsbike = {}

samsbike.height = 172

mattsbike.height = 184

print(samsbike.height) --Should print 172

print(mattsbike.height) --Should print 184

--If the kids are too confused, you can skip this next part on methods

function samsbike.setheight(x)

samsbike.height = x

end

samsbike.setheight(150)

print(samsbike.height) --Should print 150

**For Loop**

We are going to create an object that has 3 string variables. We are then going to loop through each variable and print the string.

**Pseudocode**

bike = {}

bike.height = "1.2 m"

bike.speed = "20 km/hr"

bike.terrain = "Mountain"

for each variable in objects

print the variable

end for loop

**Lua Code**

strings = {}

bike.height = "1.2 m"

bike.speed = "20 km/hr"

bike.terrain = "Mountain"

for \_, item in pairs(objects) do

print(item)

end

--Should print

--1.2 m

--20 km/hr

--bike.terrain