VIPLE Programming in Middle School Education

by

Emily Belt

has been approved
Spring 2020

APPROVED (printed name, signature)

Yinong Chen Director

Cindy Miller Second Committee Member

ACCEPTED:

_____

Dean, Barrett, The Honors College

Table of Contents

1 Abstract

Learning to code is a skill that is becoming increasing needed as technology advances, yet is absent in traditional education. This thesis aims to provide a resource for middle school teachers to introduce programming skills and concepts to their students over several lessons designed to fit within the constraints of a standard class period. By targeting students in middle school, if they develop an interest, they will have enough time in middle or high school to prepare themselves for a degree in Computer Science or to complete a programming boot camp after they graduate high school. Additionally, middle school students are old enough to understand challenging programming concepts and work together to solve a programming challenge. The programming language and environment, VIPLE, will be used to teach the concepts in the lessons as it is a graphical programming language, which removes many of the common challenges faced by young students in learning to code, like dealing with syntax or remembering keywords for coding blocks.

2 Project Scope

2.1 Preexisting Resource for VIPLE

Before beginning this project, the resources existing for VIPLE were limited to only those found on the VIPLE website and the ASU VIPLE Tutorial book published by Dr. Yinong Chen. While these resources provided a lot of valuable information for those interested in coding with VIPLE, there was no complete package for any teacher who wanted to use VIPLE to teach their students programming—especially teachers without previous programming experience.

2.2 Initial Scope

The initial scope of this project was to develop a set of lesson plans and corresponding activities that a teacher could use to bring programming to their classroom—even if they did not have prior programming experience. I intended to pilot my lessons on a group of middle school students to iron out any kinks and

to get feedback from a student perspective. Then, I would host a focus group with educators to gain feedback about the lesson plans.

2.3 Changes to Scope

Unfortunately, due to COVID-19, I was not able to host in person workshops or a focus group. Therefore, my thesis committee and I made the decision to develop a series of videos to accompany the lesson plans I created so that teachers would have an additional resource, or students could learn on their own, using the videos as a guide.

3 Lessons

Lesson 1: Intro to Programming

Overview

Students will gain an understanding of programming and see a program in action with the first lesson in our series!

Purpose

Students will discover the many uses for programming in our modern world and begin practicing with the coding environment. Here, students will see how they drag and drop components to make their very own programs.

Agenda (55 min)

Warm Up (10 min)
- Introduction

Demonstration (15 min)
- Hello World Program
- Input/Output Program
- Final Project Example

Main Activity (20 min)
- Exploration in VIPLE

Wrap Up (15 min)
- Journaling and Discussion

Objectives

Students will be able to:
- Create a simple program in VIPLE with input and output

Preparation

- ☐ Install VIPLE software on all computers
- ☐ Print journal sheets

Vocabulary

- **Program** - a series of commands that perform a certain task when executed by a computer.

Teaching Guide

Warm Up (10 min)

INTRODUCTION

Introduce the students to programming by going through the “What is Programming?” slides.

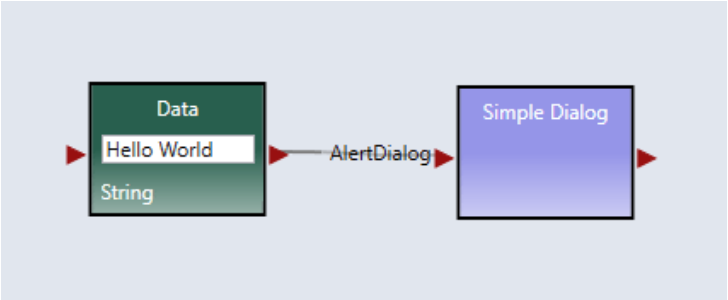
- Invite students to offer other programming languages or uses they have heard of.

Demonstration (15 min)

This demonstration serves to show the basics of VIPLE software. Share your screen on the projector and walk your students through these example programs.

HELLO WORLD PROGRAM

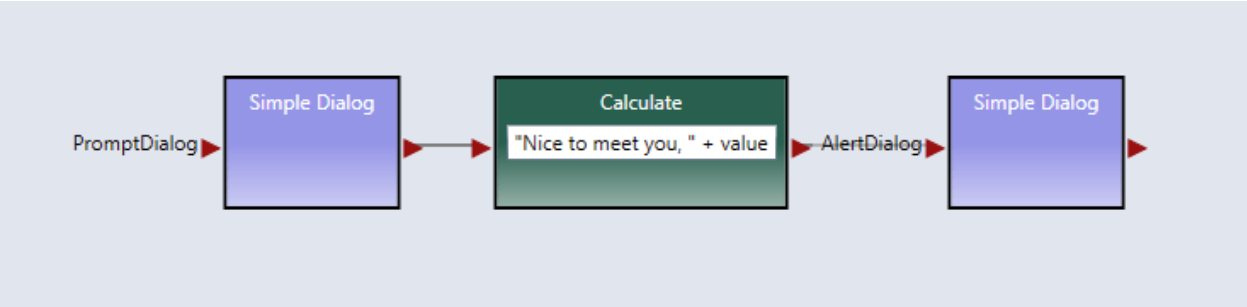
This is the simplest program one can make—in any programming language. In VIPLE, it requires the connection of just two activities. You use a Data activity and type in “Hello World” in the data field; VIPLE automatically recognizes this as a string. Then you connect the Data activity to a Simple Dialog activity and select the option for “AlertDialog” when the Connections popup comes up. When you run the program, the program outputs the “Hello World”



INPUT/OUTPUT PROGRAM

You can expand on the Hello World example by collecting an input from the user before outputting a message. After dragging the first Simple Dialog, right click on the activity and choose “Change dialog type,” then select the “PromptDialog” option. This is a dialog window that prompts a user for input. After pressing OK, enter “What is your name?” for the PromptText and leave DefaultValue blank. Connect the first Simple Dialog to a Calculate activity and the Calculate activity to an Alert Dialog as seen below. The Calculate activity combines the string “Nice to meet you” with the value of the string that was entered in the PromptDialog.

When you run this program, you will be prompted to enter your name. When you press OK, the program outputs “Nice to meet you, *name*.”



FINAL PROJECT IN VIPLE

Here is your chance to get students excited about what they will be able to accomplish by the end of the course. Show students what an example for the final project would look like. Don’t focus so much on the code, but on the result instead.

Main Activity (20 min)

STUDENT EXPLORATION IN VIPLE

Now, you will allow time for the students to replicate the programs you have demonstrated. Invite them to work together and ask each other questions before they ask you.

Once students have created the simple Input/Output program, have them chain together more PromptDialogs and AlertDialogs to create a longer conversation.

If time permits, invite students to explore using numbers instead of letters. Can they create a program that adds numbers using the same basic structure?

Wrap Up (15 min)

JOURNALING

Having students write about what they learned and how they feel about it not only helps the students to retain the information they learned, but also allows for the teacher to check understanding of the students.

Pass out the journaling sheets for this lesson and give students time to answer the following questions on their own.

1. What is a computer program?
2. Is programming easy or hard? Why?

DISCUSSION

Spend a few minutes to allow students to share their thoughts from the journaling activities with their peers—either in small groups or as a class.

Lesson 2: Data Types, Variables, and If Statements

Overview

Students will learn the common data types used in programming, how and why to use variables in a program, and the structure of an if-else statement. The concepts will first be explained to the whole group by writing on the white board. Then, in pairs, the students will practice the concepts on the computer by creating or filling in the provided VIPLE programs. By the end of the lesson, students should have a firm grasp on these fundamental programming concepts.

Purpose

This lesson aims to establish some of the most important concepts in programming that carry across all programming languages. All future lessons in this series will depend on the building blocks learned in this lesson. By learning if-else statements, students will be introduced to beginning algorithms and start thinking like a problem solving programmer!

Agenda (65 min)

Warm Up (5 min)
- Recap

Demonstration (25 min)
- Data Types
- Variables
- If-statements

Main Activity (20 min)
- Exploration in VIPLE

Wrap Up (15 min)
- Journaling and Discussion

Objectives

Students will be able to:
- Differentiate between data types
- Declare and use variables
- Write an if, if-else, and else-if statement

Preparation

- [ ] Print journal sheets
- [ ] Create and familiarize yourself with the example programs shown
- [ ] Load lesson 2 VIPLE programs on student computers

Vocabulary

- **string** - a series of characters ex. “Hello World”
- **int32** - an integer ex. -2, -1, 0, 1, 2, 3
- **double** - a number with decimals ex. -5.32, 0.000, 14.8
- **char** - a letter of the alphabet, capital or lowercase ex. a, b, c, D, E, F
- **boolean** - true or false

Teaching Guide

Warm Up (5 min)

RECAP

Remind students of the concepts learned in the previous lesson. Recall the definition for **computer program**: a series of commands that perform a certain task when executed by a computer. Invite students to share what their favorite parts of the last lesson were.

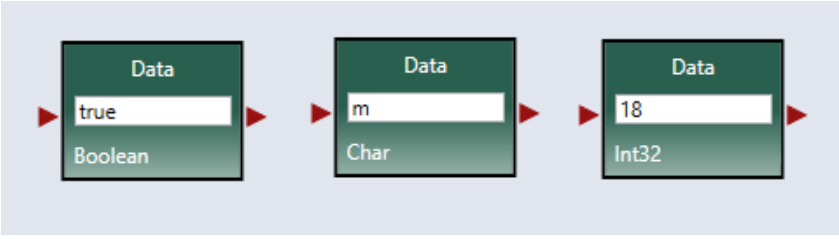
Demonstration (25 min)

Share your screen on the projector and walk your students through these example programs.

DATA TYPES

We used the Data activity in the first lesson, and now we will explain how that box works and what the different data types are. All of the data types supported by VIPLE are listed above in the Vocabulary section. The data activity in VIPLE automatically recognizes the type of data that is entered in it.

As seen in the image below, after entering ‘true,’ ‘m,’ and ’18,’ into different data activities, their data types are recognized by VIPLE.



Consider: What happens when you try to combine two different data types in a Calculate activity? Certain data types are compatible while others are not. Consider the following examples.

8.2 + 3 = 11.2

Here, 8.2 is a double and 3 is an int32. VIPLE converts 3 to a double, then adds the two numbers.

‘c’ + “Hello” = “cHello”

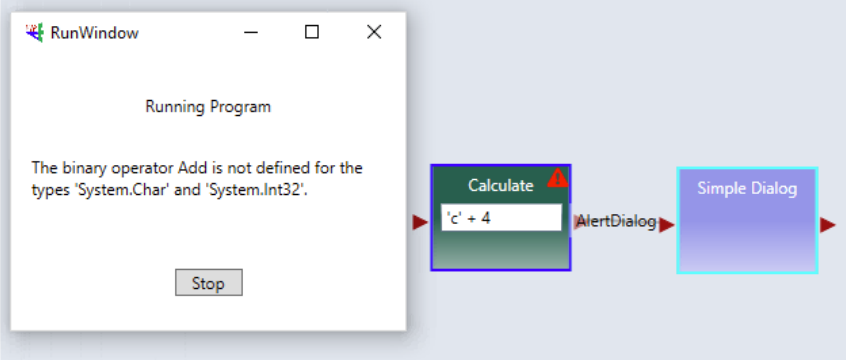
Here, ‘c’ is a char and “Hello” is a string. ‘c’ gets converted to a string, and the two are combined.

“Hello” + 4 = “Hello4”

Here, “Hello” is a string and 4 is an integer. 4 gets converted to a string just like in the previous example and they are combined.

‘c’ + 4 = ERROR

Here, ‘c’ is a char and 4 is an integer. If we try to run this calculation in VIPLE, we get the following error: “The binary operator Add is not defined for the types ‘System.Char’ and ‘System.Int32.” VIPLE also puts a red triangle with an exclamation mark on the activity that is causing the error. This will be helpful in our later lesson about debugging.

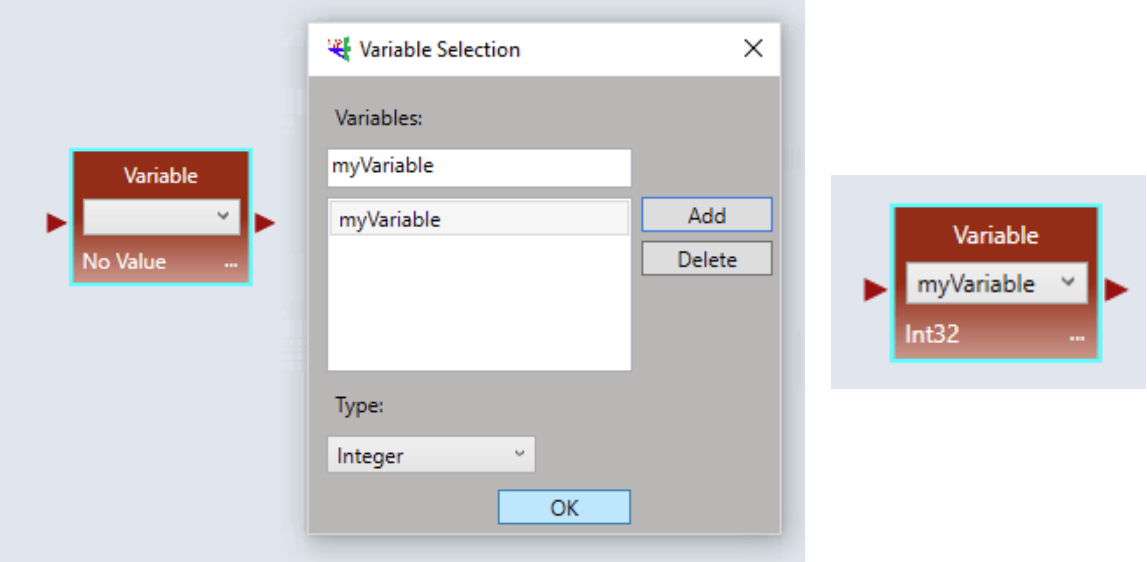


VARIABLES

We use variables when we want to have access to a particular piece of data multiple times in our program or when we want to modify and store a particular piece of data.

To create a variable in VIPLE, drag a Variable activity into the workspace, then press the three dots in the lower right corner to open the Variable Selection

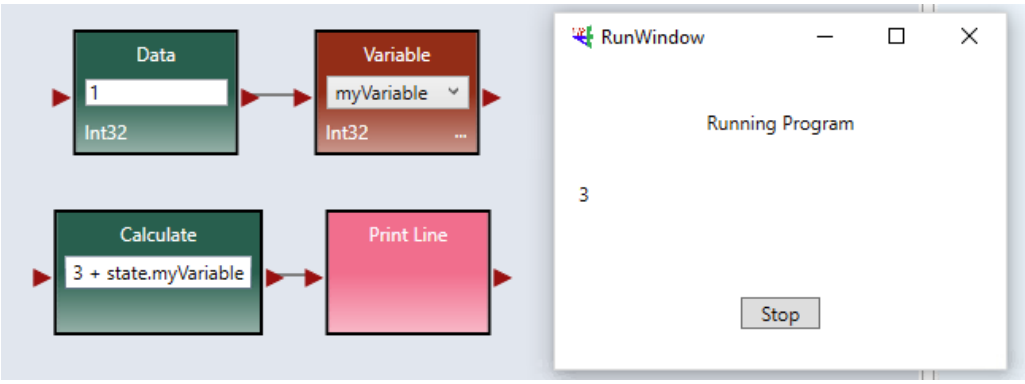
window. Type in the name you would like for your variable, click ‘Add,’ then select the data type from the drop down menu and press ‘OK.’ The Variable activity will then show the name and data type. If you would like to change the name or data type, you can click the three dots in the bottom of your Variable activity to open the Variable Selection window again.



Once a variable has been defined through the procedure above, we can now assign it a default value and use it. If you do not assign a default value, it will be set to NULL which essentially means the variable is empty. It is best practice to always set a default value for the variable, even if it is 0 or an empty string, “”.

As seen below, to set the default value for a variable, drag a Data activity to the workspace, enter the desired default value, and connect the Data activity to the Variable.

To refer to a variable, we use **state.nameOfTheVariable**. State refers to the current state of the variable in the program, and the dot notation followed by the name of the variable is how we refer to a specific variable.



BOOLEAN EXPRESSIONS

Before learning about if-statements, it is important to understand boolean expressions. A boolean expression is an expression that returns **true** or **false**.

VIPLE supports the following **boolean expressions**:

Less than: **a < b**

Less than or equal to: **a <= b**

Greater than: **a > b**

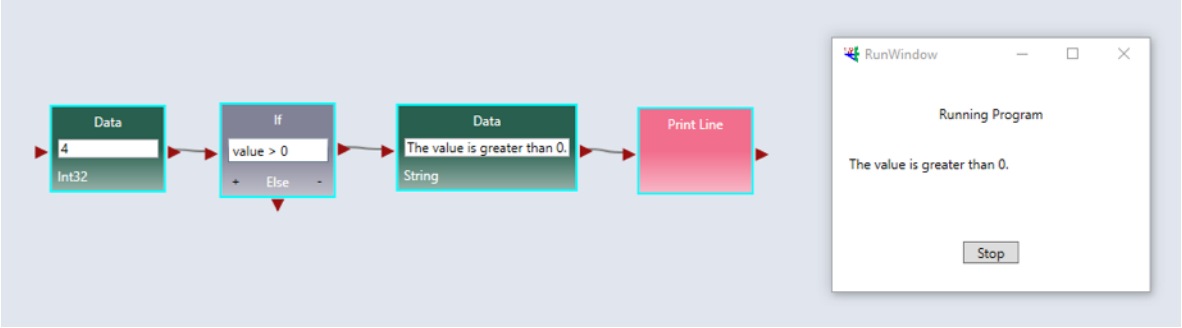
Greater than or equal to: **a >= b**

Equal to **a == b**

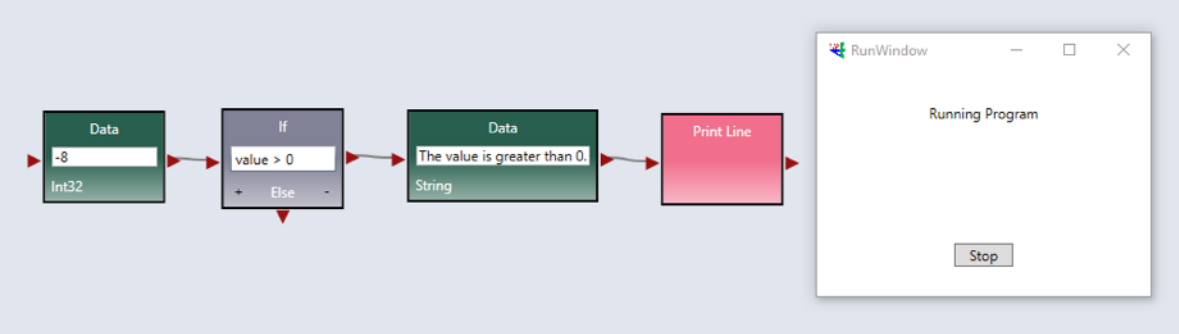
Not Equal to: **a != b**

IF STATEMENTS

The if statement is the simplest form of decision making in computer programming. If a **condition** is true, then the contents of the if block will execute. If the **condition** is false, nothing will happen. A **condition** is just a boolean expression that resolves to either true or false.
In the following example, 4 is greater than 0, so the string is printed.



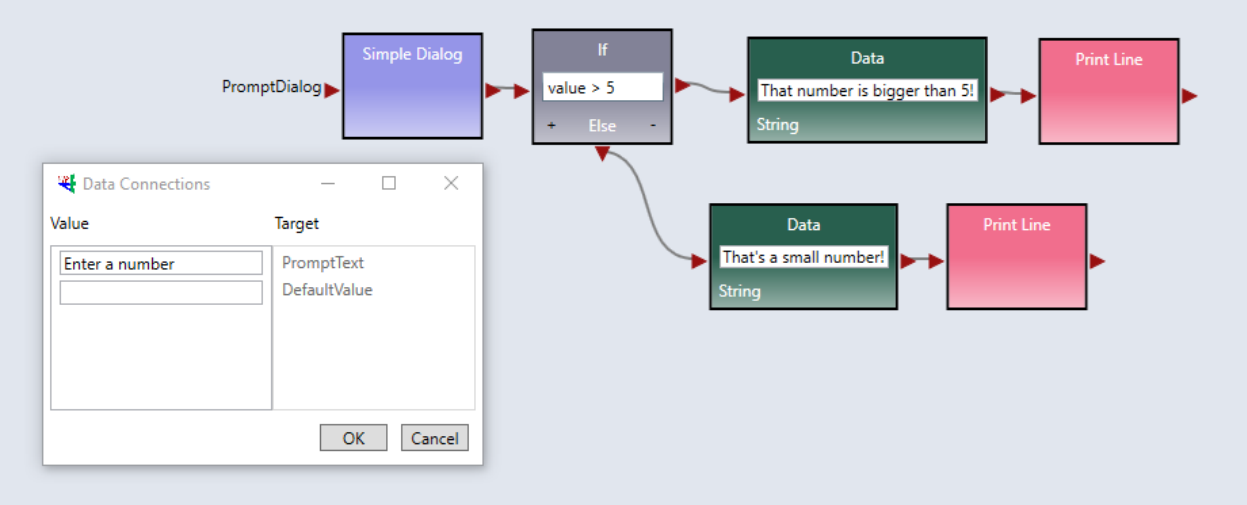
But, in this example, -8 is not greater than 0, so the string is not printed.



IF-ELSE STATEMENTS

If-else statements expand on the simple if-statement by allowing for an alternate set of commands to happen if the **condition** is not met.

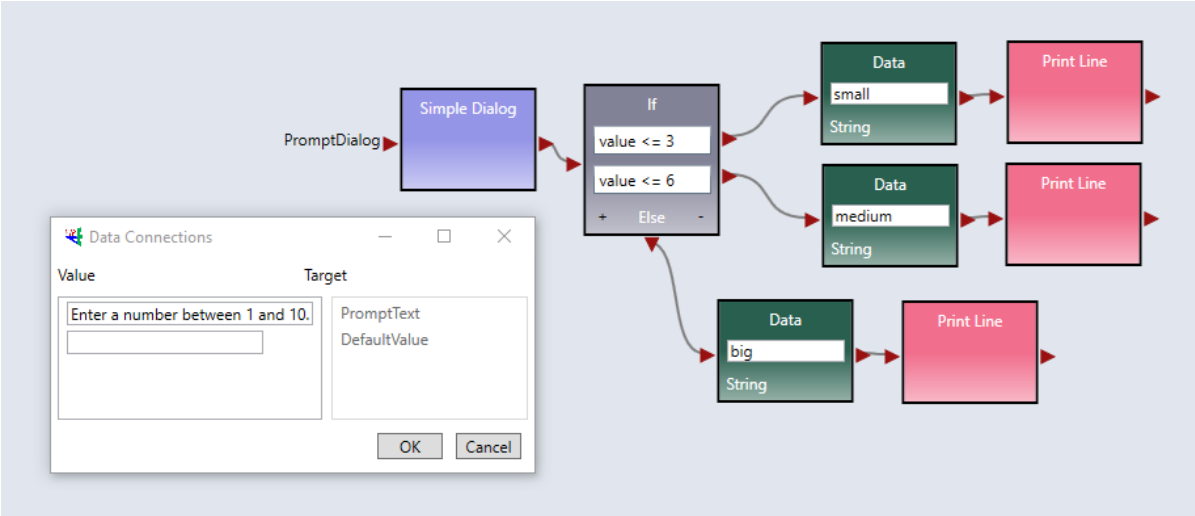
In the following example, there is a prompt for the user to enter a number. The condition of the if-else statement is value > 5. If the number entered is greater than or equal to than five, the string, “That number is bigger than 5!” will print out. If the number is not greater than 5, the string “That’s a small number.” will print.



ELSE-IF STATEMENTS

It is also possible to add a new condition if the first if condition is false by using an else-if statement.

In the following example, the user is asked to enter a number between 1 and 10. If the number is less than or equal to 3, the program prints, “small.” If the number failed the first condition (it is greater than 3), it will move onto the second condition. If the number is less than or equal to 6 —but greater than 3— the program prints, “medium.” If the number failed both conditions, then the program prints, “big.”



Main Activity (20 min)

PAIR PROGRAMMING

Now, you will allow time for the students to replicate the concepts you have demonstrated. Invite them to work together and ask each other questions before they ask you.

For this lesson, students will use all of the concepts learned: data types, variables, and if-else statements to finish putting together the Lesson 2 VIPLE program.

Wrap Up (15 min)

JOURNALING

Having students write about what they learned and how they feel about it not only helps the students to retain the information they learned, but also allows for the teacher to check understanding of the students.

Pass out the journaling sheets for this lesson and give students time to answer the following questions on their own.

1. What are the different data types we learned? Give an example for each.

2. Why do we use variables?
3. What is the difference between an if, if-else, and else-if statement?
4. What part of today's lesson was the most challenging?

DISCUSSION

Spend a few minutes to allow students to share their thoughts from the journaling activities with their peers—either in small groups or as a class.

Lesson 3: Logical Operators and Loops

Overview

This lesson introduces the concept of while loops. The lesson will begin with a demonstration of how do construct a while loop and a discussion on how while loops differ from the various if-statements learned in the previous lesson. Then, students will practice creating while loops and reinforcing concepts like data types and boolean expressions by filling out the lesson 3 VIPLE program. By the end of the lesson, students should feel comfortable with the logic of while loops.

Purpose

Loops are an important part of programming because they allow the programmer to repeat a section of code according to their constraints. Loops are also a large part of algorithms which will be introduced in subsequent lessons.

Agenda (55 min)

Warm Up (5 min)
- Recap

Demonstration (15 min)
- While Loop
- Logical Operators

Main Activity (20 min)
- Exploration in VIPLE

Wrap Up (15 min)
- Journaling and Discussion

Objectives

Students will be able to:
- Write a while loop
- Combine boolean expressions using logical operators

Preparation

❑ Print journal sheets

- [ ] Create and familiarize yourself with the example programs shown
- [ ] Load lesson 3 VIPLE programs on student computers

Vocabulary

- **&&** - Logical AND - returns true if both of the statements are true
- **||** - Logical OR - returns true if one of the statements are true
- **!** - Logical NOT - reverses the result, returns false if the statement is true

Teaching Guide

Warm Up (5 min)

RECAP

Remind students of the concepts learned in the previous lesson. Go over the different data types. Discuss the differences between and if, if-else, and else-if statements. Invite students to share what their favorite parts of the last lesson were.

Demonstration (15 min)

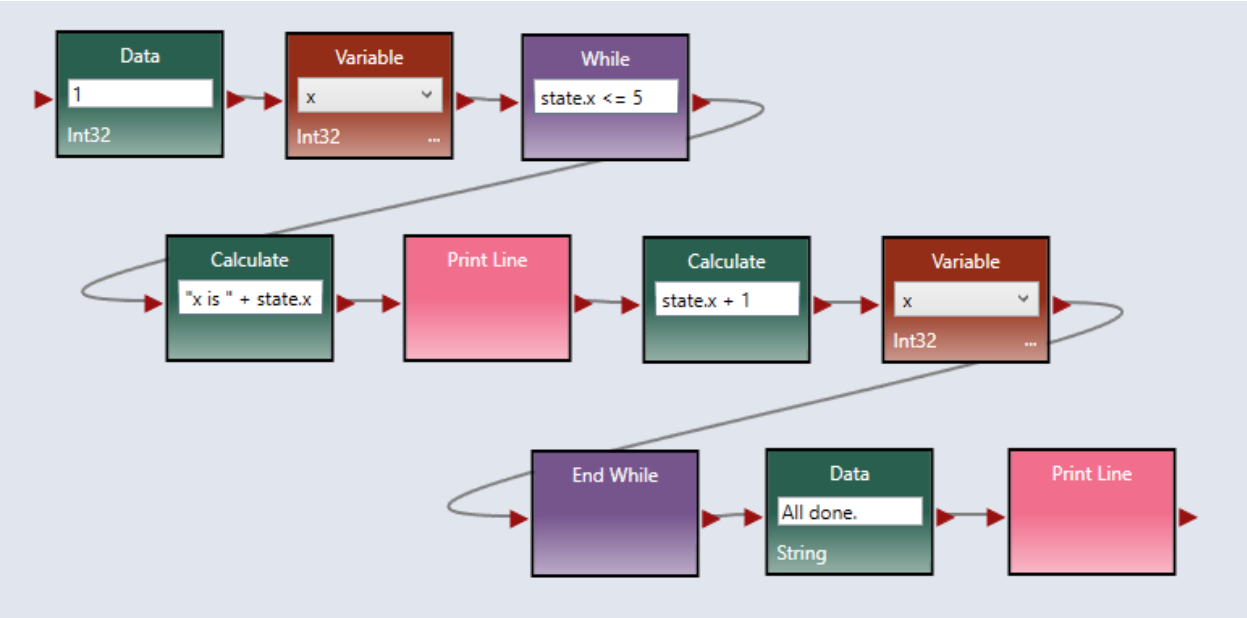
In the previous lesson, we went over different types of if-statements that allow you to execute a section of code depending on if a condition is true or not. But what if you want to execute a section of code more than just once?

WHILE LOOP

While loops allow you to execute a block of code over and over again, as long as the **condition** is still true.

Consider the following example. The variable, x, is initialized to 1. The while loop's condition is (state.x <= 5). The code between the While and End While activities will continue to execute as long as x is less than or equal to 5.

Note: After printing out the value of x, x+1 is calculated, then set as the new value of x. This is important because if you do not increment x, the loop will never end!



Consider: How is a while loop different from an if-statement or and else-if statement?

LOGICAL OPERATORS

In the last lesson, we learned about boolean expressions that are used in the condition of an if-statement or while loop. Boolean expressions can also be combined together with the help of the following **logical operators**:

Logical and: **&&** - returns true if both of the statements are true
Logical or: **||** - returns true if one of the statements are true
Logical not: **!** - reverses the result, returns false if the statement is true

With the help of logical operators, a condition could have more than boolean expression. Consider the following examples.

x = 1, y = 5

(x == 1) && (y > 4)	true. (x == 1) is true AND (y > 4) is true.
(x == 1) || (y < 3)	true. (x == 1) is true, but (y < 3) is false.
!(x > 2) && (y == 5)	true. !(x > 2) is true AND (y == 5) is true.
(x > 3) || !(y > 3)	false. (x < 3) is false and !(y > 3) is false.

Logical operators can be entered in VIPLE just like conditions.

Main Activity (20 min)

PAIR PROGRAMMING

Now, you will allow time for the students to replicate the concepts you have demonstrated. Invite them to work together and ask each other questions before they ask you.

For this lesson, students will use while loops and logical operators, as well as a concepts from the previous lesson, like variables and boolean expressions to complete the Lesson 3 VIPLE program.

Wrap Up (15 min)

JOURNALING

Having students write about what they learned and how they feel about it not only helps the students to retain the information they learned, but also allows for the teacher to check understanding of the students.

Pass out the journaling sheets for this lesson and give students time to answer the following questions on their own.

1. What is the basic structure of a while loop?
2. What part of today's lesson was the most challenging?

DISCUSSION

Spend a few minutes to allow students to share their thoughts from the journaling activities with their peers—either in small groups or as a class.

Lesson 4: Algorithms

Overview

This lesson introduces the concept of algorithms. The lesson will begin with an explanation of algorithms and a discussion about ways to solve a maze. Then, a demonstration of the Robot Maze Simulator is to be presented to show the right-wall following maze-solving algorithm. Finally, students can explore different algorithms on the maze simulator in pairs.

Purpose

Algorithms are an important part of programming because they allow programmers to make solutions for problems. Introducing the right-wall following algorithm in this lesson will allow students to code the algorithm in VIPLE in the final course project.

Agenda (65 min)

Warm Up (5 min)
- Recap

Demonstration (30 min)
- Algorithms
- Maze Simulator

Main Activity (15 min)
- Exploration in Maze Simulator

Wrap Up (15 min)
- Journaling and Discussion

Objectives

Students will be able to:
- write a simple algorithm to solve a maze

Preparation

- ☐ Print journal sheets
- ☐ Familiarize yourself with the examples and online maze simulator

Vocabulary

- **algorithm** - a set of detailed instructions that solve a specific problem

Teaching Guide

Warm Up (5 min)

RECAP

Remind students of the concepts learned in the previous lesson. Go over the three logical operators, AND, Or, NOT — &&, ||, !, and the basic structure and idea of a while loop. Invite students to share what their favorite parts of the last lesson were.

Demonstration (15 min)

Now that we have learned many basic programming structures and concepts, we want to put them into practice by creating a program that can solve a maze. To do this, we will first learn the concept of an algorithm and different maze solving algorithms.

ALGORITHMS

Algorithms are sets of detailed instructions that solve a problem. We write algorithms in simple english before we turn them into code.

Consider the following example algorithm for how to make a peanut butter & jelly sandwich:

1. Place two slices of bread side by side on a plate
2. Open the peanut butter jar
3. Scoop peanut butter out of the jar with a knife
4. Spread the peanut butter on one of the slices of bread
5. Open the jelly jar
6. Scoop jelly out of the jar with a knife
7. Spread the jelly on the other slice of bread
8. Place the slices of bread together, with the filling on the inside

As you can see, each instruction is very specific so that the computer (or person in this case) knows exactly what to do to get the finished product just right. In

fact, if we were coding a PB&J making robot, our instructions would have to be even more specific.

Discuss: How can we change our PB&J algorithm to be even more specific? Are there other ways we could write the instructions and still end up with the same sandwich?

Consider: What if we wanted our PB&J algorithm to take into consideration that someone might not like jelly? Let's rewrite the algorithm.

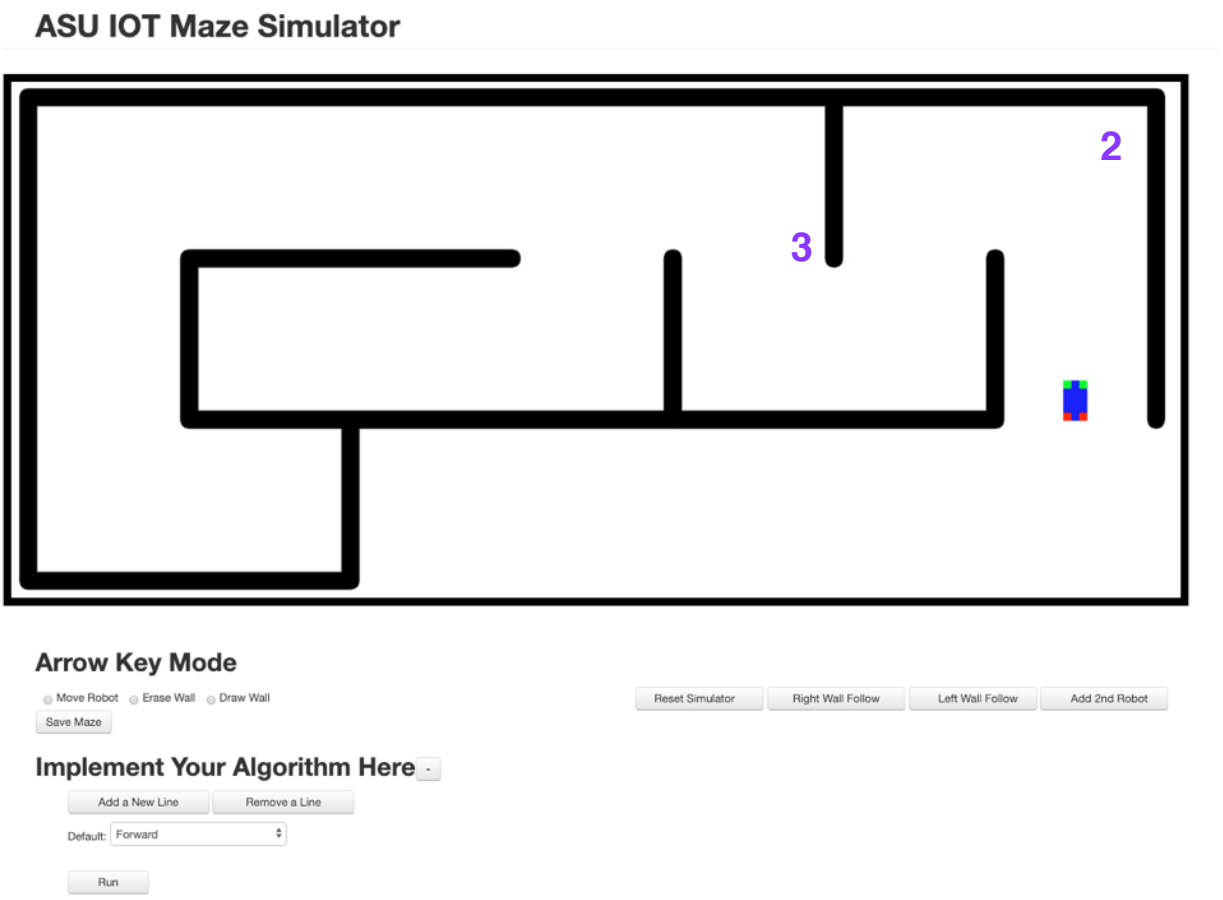
1. Place two slices of bread side by side on a plate
2. Open the peanut butter jar
3. Scoop peanut butter out of the jar with a knife
4. Spread the peanut butter on one of the slices of bread
5. If you don't like jelly, skip to step 9. Else, continue as normal.
6. Open the jelly jar
7. Scoop jelly out of the jar with a knife
8. Spread the jelly on the other slice of bread
9. Place the slices of bread together, with the filling on the inside

With one extra step, we can check for a condition and skip to another step if needed. Algorithms can also repeat steps if they are needed.

MAZE-SOLVING ALGORITHMS

As we have seen from the sandwich example, there is more than one way to solve most problems, the same of which is true for solving a maze.

Navigate to http://neptune.fulton.ad.asu.edu/VIPLE/Web2DSimulator/ in your web browser. You will see the following:



This simulator allows you to implement your own maze-solving algorithm or to try the provided right and left-wall following algorithms.

To introduce you to the simulator, try clicking the button for “Right Wall Follow.” You will see the robot traverse the maze by following the right wall. For the right-wall following algorithm, imagine you placed your right hand on the wall of a maze, and you kept your hand on the wall as you walked. This strategy ensures that you will never get lost, and you will slowly go through every part of a maze! We can watch our robot do the same!

The “Left Wall Follow” does essentially the same things, it just follows the left wall instead of the right.

Now, it’s time to come up with an algorithm to solve the maze! The robot in our simulation has 3 sensors on it, one in the front, one on the left side, and one on the right side.

Using these sensors, we can write the following simple algorithm to describe how our robot should move:

1. Robot moves forward
2. if robot's front sensor shows it is about to hit a wall, turn the robot 90 degrees to the left
3. if robot's right sensor no longer detects a wall, continue forward for a short time, then turn the robot 90 degrees to the right
4. Return to step 1

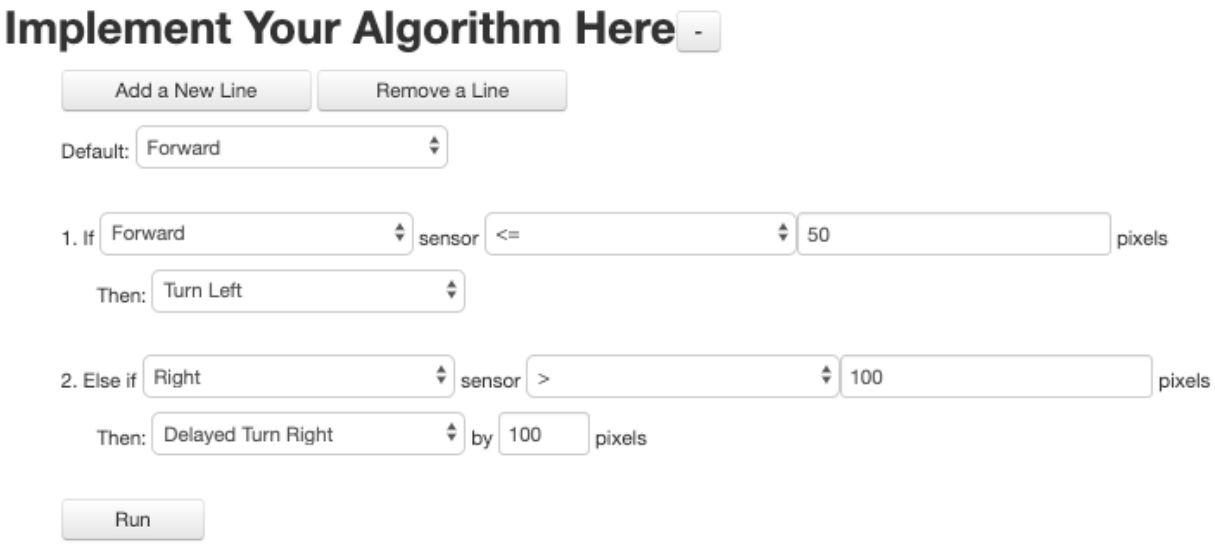
Step 2 of the algorithm takes care of corners in a maze while Step 3 takes care sections of the wall ending, where the robot has to make a U-turn shape. These steps have been labelled in the image above.

To turn the simple english algorithm into one the simulator can understand, we make the following changes:

Default step: Move forward

1. If distance measured by robot's forward sensor <= 50 pixels, then turn left
2. Else if distance measured by robot's right sensor > 100 pixels, then make a delayed right turn of 100 pixels.

The following image shows how the algorithm can be added to the Web Simulator.



Main Activity (15 min)

STUDENT EXPLORATION IN WEB SIMULATOR

Now, allow time for the students to get in pairs and interact with the web simulator. Encourage them to recreate the right-wall following algorithm you went over as a class, then to try and come up with new algorithms that allow the robot to traverse the maze.

Wrap Up (15 min)

JOURNALING

Having students write about what they learned and how they feel about it not only helps the students to retain the information they learned, but also allows for the teacher to check understanding of the students.

Pass out the journaling sheets for this lesson and give students time to answer the following questions on their own.

1. What is an algorithm?
2. If you were lost in a maze, how would you get out?
3. What part of today's lesson was most challenging?

DISCUSSION

Spend a few minutes to allow students to share their thoughts from the journaling activities with their peers—either in small groups or as a class.

Lesson 5: Bring it All Together

Overview

This lesson brings together all of the knowledge students have learned in this series by having students implement a maze-solving algorithm in VIPLE. First, you will show students how to connect VIPLE to the web maze simulator. Then, students will work in pairs to implement the right-wall following algorithm they learned in the last lesson.

Purpose

This final project will be a good way for students to showcase many of the programming concepts taught with a fun, visual representation of their work.

Agenda (60 min)

Warm Up (5 min)
    Recap
Demonstration (10 min)
    How to connect to Web Simulator
Main Activity (30 min)
    Exploration in VIPLE
Wrap Up (15 min)
    Journaling and Discussion

Objectives

Students will be able to:

- code a simulated maze solving robot

Preparation

- ☐ Print journal sheets
- ☐ Practice connecting to the maze simulator
- ☐ Create and familiarize yourself with the final project solution

Teaching Guide

Warm Up (5 min)

RECAP

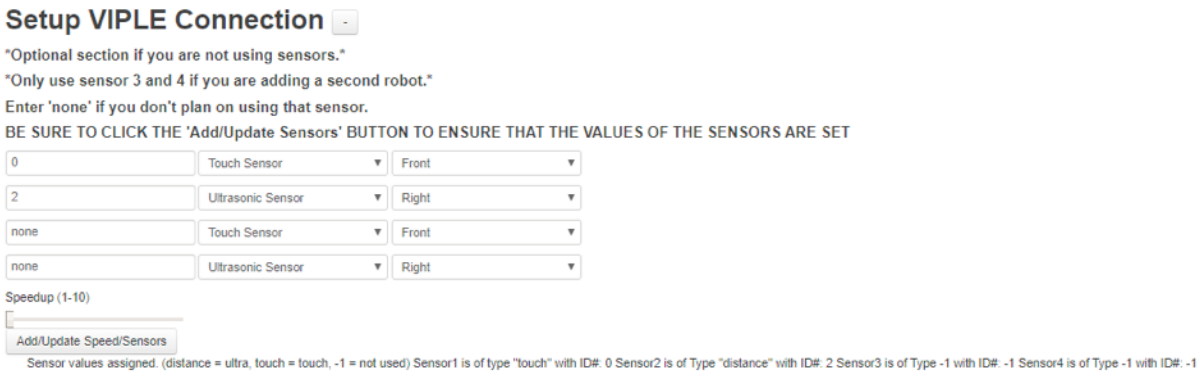
Remind students of the right wall following algorithm taught in the previous lesson. Have students write the algorithm down for reference while coding their project.

Demonstration (10 min)

CONNECTING TO THE SIMULATOR

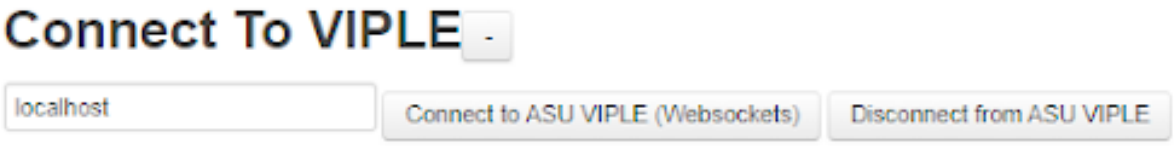
Navigate to http://neptune.fulton.ad.asu.edu/VIPLE/Web2DSimulator/ in your browser.

In the Setup VIPLE Connection section, enter 0 in the first textbox that says “ID of sensor1” and enter 2 in the textbox that says “ID of sensor2”. Then click the “Add/Update Sensors” button to connect the Web Simulator to the sensors in our VIPLE program.



Once the program for the maze-solving algorithm is completed, to run or test the code on the simulator

1. Run the program in VIPLE, using the green arrow button.
2. Click the Connect to ASU VIPLE (Websockets) button on the web simulator
3. Click in the Running Program Window in VIPLE
4. Press ‘r’ on the keyboard to start the robot moving forward



Main Activity (20 min)

STUDENT EXPLORATION IN VIPLE

Now is the time for students to implement their maze-solving algorithm in VIPLE, and control the robot with their own code! Invite them to work with their partner and ask each other questions before they ask you.

Students should have connected their VIPLE environment to the web simulator during the demonstration, so they will be ready to begin coding the Lesson 5 VIPLE program.

Wrap Up (15 min)

JOURNALING

Having students write about what they learned and how they feel about it not only helps the students to retain the information they learned, but also allows for the teacher to check understanding of the students.

Pass out the journaling sheets for this lesson and give students time to answer the following questions on their own.

1. What maze-solving algorithm did you use for your robot?
2. What was the hardest part about coding your algorithm?

DISCUSSION

Spend a few minutes to allow students to share their thoughts from the journaling activities with their peers—either in small groups or as a class.

4 Surveys

4.1 Pre Survey

Study ID Number: __ __ __ __

Pre-Survey

Programming is easy	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I would be good at programming	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I would like to learn about programming	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I know a lot about programming	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree

How much programming experience do you have?	None	A little	Some	A lot

For the following statements, circle how much you agree

For the following questions, circle the answer you think is correct.

1. What is a program?
   a. a series of commands that perform a task when executed by a computer
   b. an application on a computer, like Word or PowerPoint
   c. a way for a computer to talk to the person using the computer

2. Which of the following is **NOT** a use of programming?
   a. building websites
   b. robotics
   c. typing in a word document

3. Match the data types with the information they store.

_____ Boolean	a. 6
_____ Int32	b. Hello World
_____ char	c. b
_____ double	d. false
_____ string	e. 9.45

4. In VIPLE, 4 + 2.3 =
   a. 6
   b. 6.3
   c. ERROR

5. In VIPLE, 4 + 'c' =
   a. 4c
   b. ERROR

6. In VIPLE, "Hello" + 4 =
   a. "Hello4"
   b. ERROR

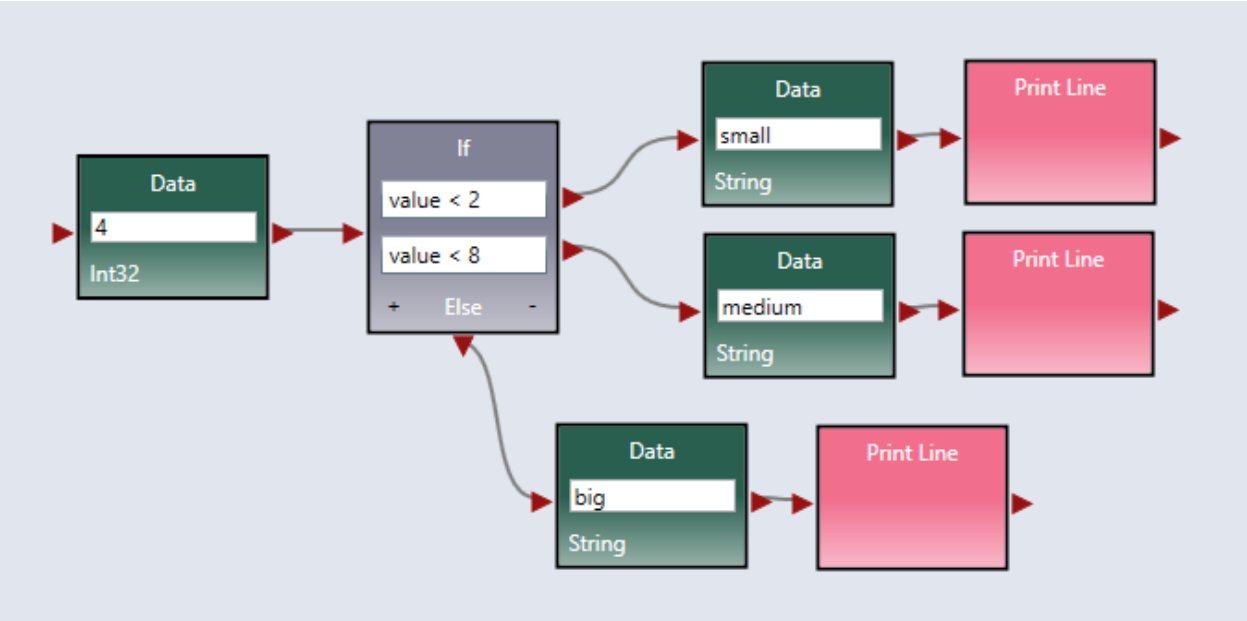
5. Match the logical operators with their definitions

_____ &&	a. returns true if one of the statements are true
_____ ll	b. returns true if both of the statements are true
_____ !	c. reverses the result, returns false if the statement is true

6. Given that **a = 6, b = 3, and c = 1**, what do the following return?

(a == 6) && (b >= 2)	true	false
!(b > 4) ll (c == 0)	true	false
(a <= 5) && !(c == 1)	true	false

7. What is the result of the following program?



a. small
b. medium
c. big
d. ERROR

8. Order the steps of the Right Wall Following Algorithm from 1-4.

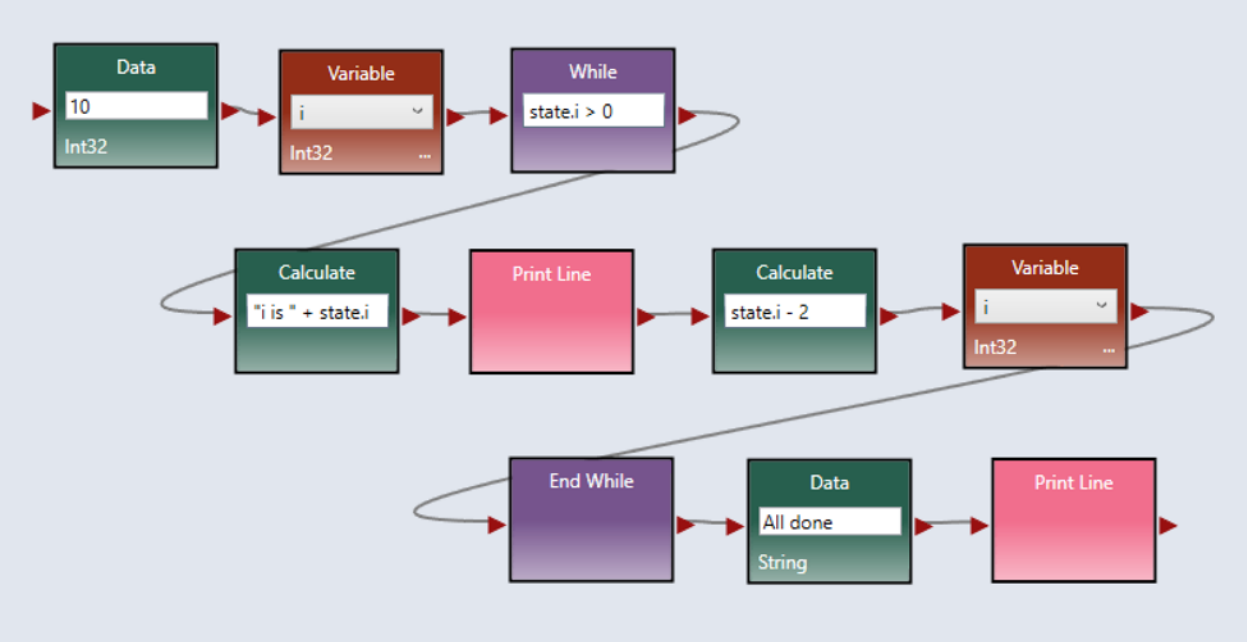
_____ if sensor.forward distance < 50, turn to the left

_____ Robot moves forward

_____ Repeat

_____ if sensor.right distance > 100, delay then turn to the right

9. What is the output of the following program?



Output:

4.2 Post-Lesson Surveys

Study ID Number: __ __ __ __

Lesson 1: Intro to Programming

Post-Lesson Questions

For the following statements, circle how much you agree.

I liked learning the concepts in this lesson	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The concepts in this lesson were hard to understand	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
VIPLE was easy to use for this lesson.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The coding challenge helped me understand the concepts in the lesson	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The coding challenge in this lesson was easy	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree

For the following questions, circle the answer you think is correct.

1. What is a program?
   a. a series of commands that perform a task when executed by a computer
   b. an application on the computer, like Word or PowerPoint
   c. a way for a computer to talk to the person using the computer

2. Which of the following is **NOT** a use of programming?
   a. building websites
   b. robotics
   c. typing in a word document

Study ID Number: __ __ __ __

Lesson 2: Data Types, Variables, and If Statements

Post-Lesson Questions

For the following statements, circle how much you agree.

I liked learning the concepts in this lesson	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The concepts in this lesson were hard to understand	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
VIPLE was easy to use for this lesson.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The coding challenge helped me understand the concepts in the lesson	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The coding challenge in this lesson was easy	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree

1. Match the data types with the information they store.

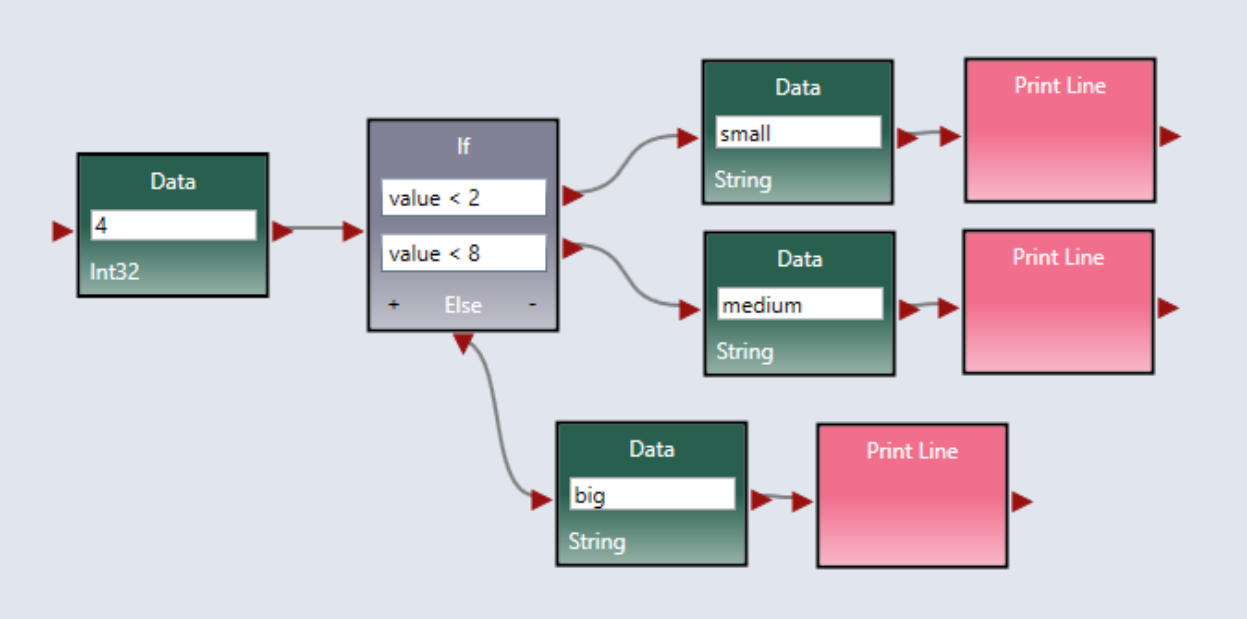
_____	Boolean	a.	6
_____	Int32	b.	Hello World
_____	char	c.	b
_____	double	d.	false
_____	string	e.	9.45

2. In VIPLE, 4 + 2.3 =
   a. 6
   b. 6.3
   c. ERROR

3. In VIPLE, 4 + ‘c’ =
   a. 4c
   b. ERROR

4. In VIPLE, “Hello” + 4 =
   a. “Hello4”
   b. ERROR

5. What is the result of the following program?



   a. small
   b. medium
   c. big
   d. ERROR

Study ID Number: __ __ __ __

Lesson 3: Logical Operators and Loops

Post-Lesson Questions

For the following statements, circle how much you agree

I liked learning the concepts in this lesson	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The concepts in this lesson were hard to understand	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
VIPLE was easy to use for this lesson.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The coding challenge helped me understand the concepts in the lesson	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The coding challenge in this lesson was easy	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree

1. Match the logical operators with their definitions

_____ && a. returns true if one of the statements are true

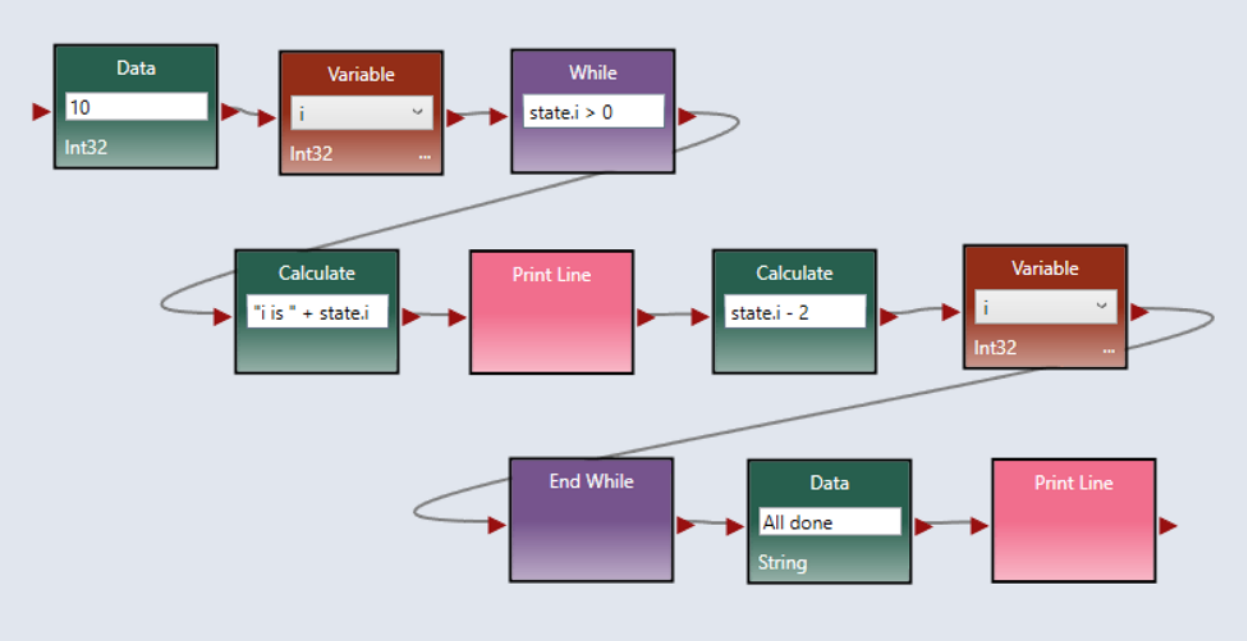
_____ || b. returns true if both of the statements are true

_____ ! c. reverses the result, returns false if the statement is true

2. Given that **a = 6, b = 3, and c = 1**, what do the following return?

(a == 6) && (b >= 2)	true	false
!(b > 4) ll (c == 0)	true	false
(a <= 5) && !(c == 1)	true	false

3. What is the output of the following program?



Output:

Study ID Number: __ __ __ __

Lesson 4: Algorithms

Post-Lesson Questions

For the following statements, circle how much you agree.

I liked learning the concepts in this lesson	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The concepts in this lesson were too hard to understand	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I feel like I learned something important in this lesson	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The coding challenge helped me understand the concepts in the lesson	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The coding challenge in this lesson was easy	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree

1. Order the steps of the Right Wall Following Algorithm from 1-4.

_____ if sensor.forward distance < 50, turn to the left

_____ Robot moves forward

_____ Repeat

_____ if sensor.right distance > 100, delay then turn to the right

4.3 Post-Series Survey

Study ID Number: __ __ __ __

Post-Survey

For the following statements, circle how much you agree

Programming is easy	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I am good at programming	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
VIPLE is easy to use	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I liked learning to program	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I know a lot about programming	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I enjoyed the maze-solving programming challenge	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The maze-solving challenge was easy	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree

5 Accompanying Links

5.1 Link to individual lessons

https://drive.google.com/drive/folders/1JqWpApyUX1SnHcqSqGnCM2-tS34rrNha?usp=sharing

5.2 Link to practice VIPLE programs

https://drive.google.com/drive/folders/119E5Zk05PvD1E8C21EeDBs4OZAGvFbcP?usp=sharing

5.3 Link to video lessons

https://www.youtube.com/playlist?list=PLFXg-cPh1crVgxZKcM7AL6fPNaBURxB8V

6 Future Work

There is great potential for more lessons to be added onto this beginning series of programming lessons. VIPLE supports LEGO EV3 robotics kits which would be a welcome addition for adding a series of lessons regarding robotics.

Additionally, seeing as the lessons are currently untested with students, piloting the lessons would add to the validity of the project. The included surveys in this thesis could be used to get feedback from students in the future.

7 Conclusion

The beginning programming series I created in this project will hopefully prove to be an incredible resource for teachers who want to introduce programming to their students. While there are many tools and software options that exist to teach programming to students, VIPLE is unique in that it has all of the capability of modern programming languages, it can work with the LEGO EV3 kits, yet it is more simple because it is a graphical programming language.

This project also acted as a massive learning experience in project development and showed how much projects and ideas can change over time. Initially, this project started as a way for me to personally teach students in middle school how to program. But realizing what an undertaking that would be, and seeing the longer term value in empowering teachers to teach their own students, my committee and I made the decision to develop lesson plans for teachers. Then, within each lesson, there were new concepts I didn't realized I would need to be included. For example, when developing the lesson about if-statements, I realized I would have to first explain what a boolean expression was, because that is what goes in the condition of an if-statement. Outside of the lesson contents themselves, this thesis taught me that sometimes, very rarely, a global pandemic can occur, and throw your whole world for a loop, so you need to be able to adapt and find ways to accomplish a similar result in a completely different way. I did this by learning how to make videos and creating a video for the Demonstration part of each lesson I developed.

While I am incredibly proud of the work that I created over the last two semesters, my lessons are by no means perfect. There are always ways to explain things to make them more clear, and because I unable to gather feedback from real middle school teachers, I was unable to incorporate alternate explanations for any of the concepts.

8 Acknowledgements

First and foremost, I'd like to thank Dr. Yinong Chen and Dr. Cindy Miller, my thesis director and second committee member respectively, who I've had the pleasure of working with these last two semesters. Both of you have given me so much insight and help throughout this project and I'm grateful for the opportunity to have worked with you.

I'd also like to thank Julie Rotberg who helped my idea come to fruition back when I worked on the Engineering in Action project. This project wouldn't have

happened if I didn't have your constant support and encouragement as I taught all those students.

Lastly, I'd like to thank my family and friends: Mom, Dad, Hayley, Nana, Lucas, Serita, and Ali, who inspire me every day and push me to keep going, even when life gets hard.

9 References

[1] VIPLE Laboratory Manual. *http://neptune.fulton.ad.asu.edu/VIPLE/*

[2] W3Schools.org. W3Schools. http://www.w3schools.com/

[3] CS Discoveries 2019-2020. Code.org. https://curriculum.code.org/csd-19/