FIRIA LABS

LESSON: April Showers Project		Time: 45- 60 minutes
Overview: This lesson will simulate rain showers as the April monthly theme. It is the first part of the project. It continues with May Flowers. The second part can be completed immediately after the first part, or you can wait until May.		**Objectives:** • I can control the CodeX with the accelerometer. • I can use a list of lists for the position of lines. • I can use nested for loops to draw several lines on the display. • I can update values in a list.
Grades 6-8 CS Standards: **2-CS-01** Design projects that combine hardware and software components to collect and exchange data. **2-CS-03** Systematically identify and fix problems with computing devices and their components. **2-AP-11** Create clearly named variables that represent different data types and perform operations on their values. **2-AP-13** Decompose problems and subproblems into parts to facilitate the design, implementation and review of programs. **2-AP-14** Create procedures with parameters to organize code and make it easier to reuse. **2-AP-16** Incorporate existing code, media and libraries into original programs, and give attribution.	**Grades 9-10 CS Standards:** **3A-DA-11** Create interactive data visualizations using software tools to help others better understand real-world phenomena. **3A-AP-13** Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests. **3A-AP-14** Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables. **3A-AP-16** Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions. **3A-AP-17** Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules and/or objects.	**Grades 11-12 CS Standards:** **3B-AP-10** Use and adapt classic algorithms to solve computational problems. **3B-AP-14** Construct solutions to problems using student-created components, such as procedures, modules and/or objects. **3B-AP-16** Demonstrate code reuse by creating programming solutions using libraries and APIs. **3B-AP-17** Plan and develop programs for broad audiences using a software life cycle process. **3B-AP-22** Modify an existing program to add additional functionality and discuss intended and unintended implications.

Preparation:
• Download slides
• Be familiar with the final code
• Read through the teaching guide
	In the folder:
	• April Showers project slides
	• April Showers project code
	  ○ Starter code
	  ○ Code solutions for all steps and final
		Agenda:
		• Complete program using slides (45-60 minutes)

Teacher Notes: • This lesson is designed so that students can work independently by following the slides. However, you can also work together as a class by projecting the slides. See the teaching guide for specific help and hints. • Almost all mistakes made by students are typing mistakes. If students get errors when they run their code, first look over the code for spelling, punctuation and indenting. • The solution code for each step is provided. • The program has two parts – April and May. They can be completed together, or one each month. Each month has its own slides.		

Teaching Guide

Warm-up

A warm-up isn't really needed for this lesson. You can talk about April and the weather, review concepts in Python or discuss debugging techniques. Or, just start the lesson.

Create/Run the Program (~45 minutes)

💻 This project can be completed individually or with pair programming.
It can be completed working independently or as a whole class.

💡 **Teaching tip:**

This project is not included in CodeSpace. Download and follow the slides. They include step-by-step instructions as well as code snippets to guide students through the program code creation.

You can have students complete the project one of two ways:

- Show the slides on a large screen or monitor and have the class work on each step together.
- Give the slides to the students and let them work through the instructions at their own pace. This works well if students are pair programming and have access to one computer for instructions and one computer for programming.

💻 **Slide 3 New File**

Students get into CodeSpace and start a new file. Use the sandbox! If students do not already have an account, they can easily create one using any email address. Or they can log in as a guest, which means their code will not be saved.

💻 **Slides 4-5 Starter Code**

This project has starter code for the students that they will add to. Give the starter code text file to your students. The easiest way is probably to use your LMS. You can also upload the file to the cloud and insert a link to the file in the slides for students to access. Students need to open the text file and copy and paste the code into a new file in CodeSpace. They will not open the starter file in CodeSpace.

Students should run the code and make sure there are no errors.

💻 **Slides 6-12 Step #1 Introduction**

Students explore the given code and learn about the project.

- The final project will simulate three types of rain: light, regular and heavy.
- The type of rain is determined by shaking the CodeX. The accelerometer and some math will then assign values for coding the different types of rain.
- These slides go over the basics of the program and have students find the accelerometer on the CodeX.

💻 **Slides 13-15 Step #1 Assign values to use with the rain**

Students add code to the function.
The if statement will assign values to the variables needed for the different types of rain. It needs to have the same indentation as #TODO.

- The if statement has four branches: three for the types of rain and one for no rain (no shaking).
- The variable "count" is given a value from shaking the CodeX. The more rigorous the shaking, the higher the count. The values that count is compared to can be modified depending on how sensitive the accelerometer is to the student's shaking.
- The #comments do not need to be typed by the student, but they do help them understand the code.
- The return statement returns four values, one for each of the arguments used later for the types of rain.
- The first value indicates the CodeX was shaken, so it should rain. It is a Boolean, so it will be True or False.
- The second value is the gap between lines for the rain. Heavy rain is close together (small gap) and light rain is more spread out (large gap). These numbers can also be adjusted.

- The third number is the number of times to loop the rain. A light rain loops twice, while a heavy rain loops 6 times. These numbers can be adjusted.
- The final number is the delay. A small delay is used for very fast (heavy) rain. A longer delay for a gentle (slow) rain. These numbers can be adjusted.
- The numbers need to remain in the order so they match the variables that get their values.

💻 **Slides 16-19 Step #2 Create a list for the rain**

Students add a function that creates a list of lists. The list is used to place lines across the display for the rain.

- The code can look a little complicated. Slides are added to explain some of the code.
- In range(240/gap), the 240 is used because that is the size of the screen. Dividing by the gap will give the number of lines needed for the type of rain. The // will give an integer as the answer instead of a float.
- If students want to see the list after it is created, they can add this code to the main program and open the Console Panel to see the list:

```
# -- MAIN PROGRAM
fill_lines(15)
print(lines)
```

💻 **Slides 20-25 Step #3 Simulate the rain**

Students add another function to call rain() several times. It will have two loops, one nested inside the other.

- When calling the rain() function, the typing needs to be exact. Don't include spaces between the square brackets, and make sure to use square brackets. The values in the list are being accessed.
- The inner loop that draws lines across the screen should work properly.
- The outer loop that draws the lines going down will need more code.

💻 **Slides 26-30 Step #4 Incrementing y-values**

Students add another function to increment the y-values.

- This function is called in the rainfall() function so the lines move down the display.
- Again, the code can look intimidating because students are updating a value in the list of lists.
- Only the [1] value is changing, which is the y-value. The x-values will stay the same because the lines stay in the same beginning place and just move down but not over.

💻 **Slides 31-34 Step #5 Simulating the rain**

This is the final step. Students will write code for the main program, which brings all the functions together.

- The order of the variables in the while True loop matter. They will get their values from the return statement in rain_type(), so they need to be in the same order.
- Students should be able to shake the CodeX in various amounts to see all three types of rain. If needed, students can adjust the function rain_type(). The values that count is compared to can be changed so each type of rain is detected.
- They can also adjust the values returned if they want the rain to run faster or slower, etc.

Wrap-up

💻 You can wrap-up this project in a variety of ways, depending on your students and your classroom procedures.

- Students can share their projects with other students, especially students not in class. Challenge them to do this!
- Students can fill out a journal entry about their experience or what they learned during the lesson.

May Flowers

💡 **Next Project:** This is Part 1 of the project. The second part is May Flowers. You can choose to continue the project now, or have students wait until May and then add to their project. May Flowers has its own set of slides, and five more steps.