FIRIA LABS

LESSON: Valentine's Day Project		Time: 45 minutes
Overview: This lesson will enable students to send and receive short text messages from each other using a radio signal. It will also use lists as a data abstraction, random selection, and random numbers for a screen location.		**Objectives:** • I can turn on and off the CodeX radio signal. • I can send and receive text messages. • I can create and use lists. • I can display a random bitmap image on the display screen.
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-16** Incorporate existing code, media and libraries into original programs, and give attribution.	**Grades 9-10 CS Standards:** **3A-CS-03** Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors. **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-21** Evaluate and refine computational artifacts to make them more usable and accessible.	**Grades 11-12 CS Standards:** **3B-AP-10** Use and adapt classic algorithms to solve computational problems. **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:** • Valentine's Day project slides • Valentine's Day project code • OPTIONAL: CodeX and RGB slides	**Agenda:** • Optional: Warm-up (5 minutes) • Complete program using slides (30 minutes) • Run program (5 minutes) • Optional: Wrap-up (5 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. See the teaching guide for specific helps and hints. • The program is most interesting if all steps are completed. But if you are short on time, students can stop after any of the five steps and still have a working program. Extensions are also available if you have more time or students work quickly. • 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 code for each step can be found in the folder.		

Extensions:

- Divide students into smaller groups. Assign each group their own channel. One group can be channel 6, but other groups will have different channel numbers. They will send and receive only to students with their same channel.
- Students can create their own custom colors using RGB triplets. If students are already familiar with RGB triplets, they can add them to their colors list. You can also use the CodeX and RGB slides to learn more about RGB colors. The custom colors don't need a name; simply add the triplets in the list, like this:

```
text_colors = [RED, WHITE, PINK, MAGENTA, CYAN, (255, 50, 150)]
```

- Learn more about RGB colors by using the Console panel.

```
>>> from codex import *
>>> print(PINK)
(255, 20, 147)
```

  The RGB triplet is returned. Then students can try different triplets to create their own colors.
- Try different arguments for the random numbers used for hx and hy. See how the arguments affect where the hearts are displayed on the screen.
- After the code is finished, have students add more colors and/or messages. Discuss with the students how they didn't need to revise the code to handle a different number of items. This is an example of data abstraction.
- In this program, the radio stays on until the program quits. Add code for a button to act as a toggle, turning on and off the radio.
- Look at the code for the bitmap heart image. Have students modify the color and/or shape of the heart.
- The bitmap image is always scale=2. Get a random number for the size of the heart.
- Make the starting menu more interesting.

Cross-curricular Connections:

- **LANGUAGE ARTS:** Have students write about their coding experience.
- **LANGUAGE ARTS:** Have students summarize something they learned during the project.
- **LANGUAGE ARTS:** Have students create a tutorial to explain how to do the first part of the program (turning on the radio and sending and receiving a message).
- **LANGUAGE ARTS:** Have students compare and contrast a part of the program. Examples: if statements and while loops, the two different lists, the different ways to get a random selection.
- **MATH:** The program uses random numbers. How random are the numbers? Do some experiments with probability.
- **MATH:** Keep track of how often each message is displayed. Calculate the probability of each message. Try the experiment again. Do the percentages hold up? How random is the random function?
- **SCIENCE:** The program uses red, green and blue lights to form colors. Have a lesson about light.
- **SCIENCE:** The program uses radio signals. Have a lesson about how sound or data travels through radio signals.
- **SOCIAL STUDIES:** Research the history of Valentine's Day. How do people celebrate it? What are the cultural aspects of Valentine's Day?
- **VISUAL ARTS:** Using a simple grid, have students design their own Valentine's Day image.

Teaching Guide

Warm-up / Optional (5 minutes)

This short warm-up is to help students brainstorm possible short messages to send on the CodeX.

💡 **Teaching tip – warm-up**

- Show a bag of Valentine “conversation hearts” candy.
- What are some short messages you might want to send to your friends? The messages should be short and uplifting.

Create/Run the Program (35 minutes)

💻 For this project, it is best if each student has his/her own CodeX. It can be completed in pair programming, but it is more fun with their own CodeX, since they will be creating their own messages and adding their initials to the message.

💡 **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.

This program requires students to test the program with a partner. Students will send a message to at least one other person, and then receive messages from at least one other person. They cannot send and receive to themselves.

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. They will need other students working at about their same pace so they can test their code with a partner.

💻 **Slides 1-3**

Students get into CodeSpace to create their program. 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 4-9 Step #1 Can you hear me now?**

Students start a new file and begin their code. Students should test at the end of each step (or even during the step) and fix any errors before continuing.

- The instructions are set up so that all students are on the same channel. In this scenario, all students will send and receive messages to all other students. If you want, you can divide students into smaller groups and assign each group a different channel. The channel numbers are 0-13. However, if students are assigned adjacent channels, they may get some overlap messages. So if possible, assign only the odd or only the even channels.
- What to watch for: In Python, spelling, punctuation and indenting are key. If students are getting errors in code, the first thing to look for is typing or spelling mistakes. Also, check the indenting. The code indenting must be consistent within blocks of code. And blocks of code start with a colon (:). Almost all mistakes at the beginning level are typing mistakes.
- One other thing to note, if you are new to Python, is that spacing in between elements doesn’t matter. For example, in `display.print(msg, scale=2, color-WHITE)`, you can have a space around the = or not, and after a comma, or not. The spacing doesn’t matter, but the indenting and spelling do matter. But you cannot have a space between print() and parenthesis.
- The final code for the first step is included on slide 9.

💻 **Slides 10-16 Step #2 Make It Personal**

Students personalize their message by adding their initials at the end, and getting random colors for the messages. Students will create a list and get a random color from the list to use when the message is displayed.

- The pre-defined colors are named in ALL CAPS. This is important.
- A list is defined with square brackets. [ ]
- An extension is to let students customize their own colors. See extension bullet #2.
- The full code is provided on slide 16.

💻 **Slides 17-21 Step #3 Mixed Messages**

Students create another list for messages. The program will select a random message to send each time the button is pressed. Students should be creative and type in their own messages, but they should be short so they fit on the screen.

- Each item in the list will be a string. The words must be enclosed within quotation marks. The comma that separates the items is not in the quotation marks.
- The full code is not provided for this step.

💻 **Slides 22-25 Step #4 Steer Clear**

Students add code to clear their screen and to quit the program.

- This code is added to the main program. Students need to be careful with their indenting.
- The full code for the main program is included on slide 25. That is the only part of the code that is changed.

💻 **Slides 26-34 Step #5 Add Some ❤️**

Students add a bitmap heart image to the code. They use random numbers for a location on the screen, and then display a heart every time they receive a new message.

- The code for the heart can be copied and pasted, so the students don't need to re-create it. Students can copy and paste from the slide, or you can give them the code digitally. It needs to be exact, or errors occur.
- The code for this step is a bit tricky compared to the earlier code. Students need to be careful of typing mistakes.

Wrap-up / Optional (5 minutes)

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

- Students can fill out a journal entry about their experience or what they learned during the lesson.
- Students can share with each other or in small groups something they learned, or how they might apply what they learned to a different project.