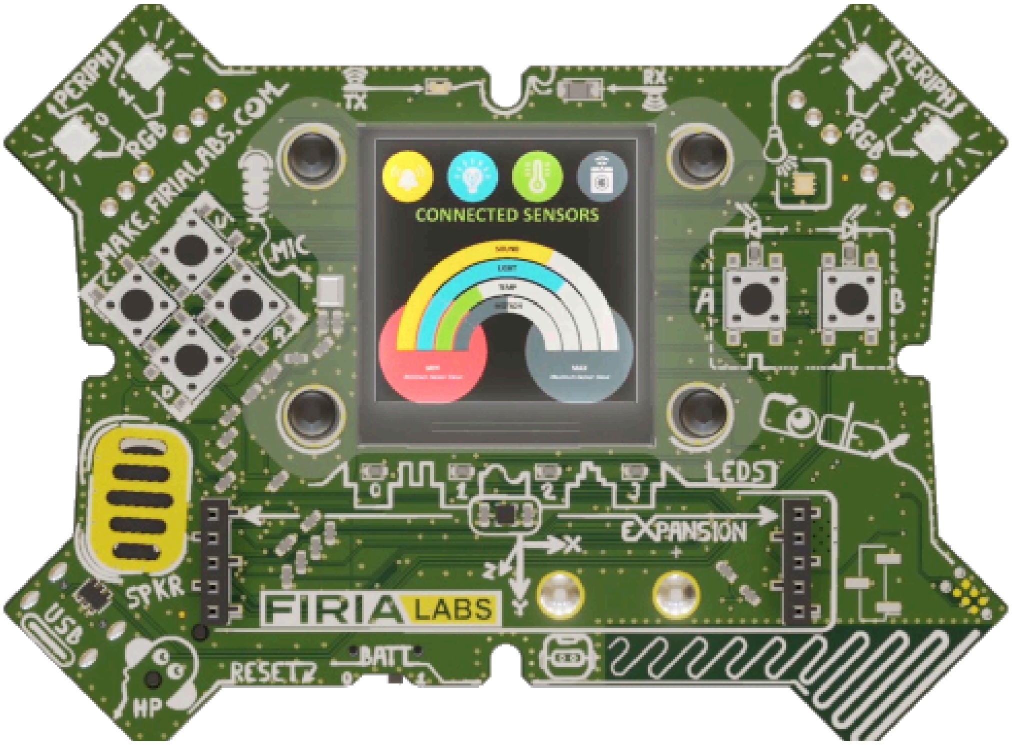
Curriculum Guide



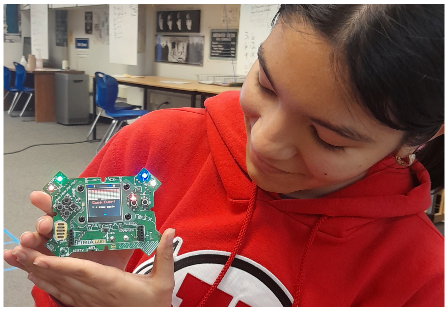
Mission Pack:

Python with CodeX

CodeX™ Python CodeSpace Course

Table of Contents

Python with CodeX Overview



Introducing CodeX, an entry-level physical computing device for a gentle introduction to Python programming. CodeX features built-in graphics, sensors, sound and more in a rugged, expandable one-piece design. Compatible with Chromebook, PC and Mac, this curriculum offers missions for elementary, middle and high school students that can drop-in to elective courses, Computer Science 1, CTE and AP Computer Science Principles.

The Python with CodeX curriculum is easily adaptable for elementary students starting in fourth grade up through high school and beyond. The missions can be completed as part of a unit in an elective, during an after-school club, or as a one- or two-semester elective course in middle or high school. The mission pack course covers the fundamentals of Python programming as students apply each new coding skill and concept to engaging projects with **CodeX**.

The time required, or pacing calendar, is up to you! You can spend as little as a few weeks just completing a subset of the missions. Or for a more complete experience, include remix projects and additional lessons for students to apply their knowledge and newly gained skills that can last a semester or full school year. No prior coding experience is required! CodeX puts the focus on coding, with built-in sensors and programmable controls for *endless* projects and learning opportunities.

Pre-Mission Work (1 - 5 hours)

Build a foundation for programming by utilizing some unplugged activities. If your students come with no Computer Science background, it is important to start by building a foundation of computational thinking. Dedicate some time for students to learn basic terms, such as algorithm, program, and debug. See the Firia Labs collection of Unplugged Activities at https://learn.firialabs.com/curricula/cs-unplugged.

Getting Started (Mission 0)

When you are ready for your students to start the CodeX missions, you will need to set up a class for them in CodeSpace and give them a join code.

Use these resources in the Teacher Resources materials on the learning portal.

- A - Lesson Prep Getting Started
- Getting Started Slides
- Getting Started Workbook
- Teacher Dashboard in CodeSpace

Unit 1: Getting Started (8-20 hours)

Students will learn about the programming environment, the CodeX, and basic commands for programming the CodeX using Python. Students start by turning on the RGB pixels, then displaying images on the LCD and finally by playing mp3 files.

Mission 1 Welcome:

Take a tour of the CodeSpace Development Environment. Students create an account and join the class to access the curriculum. The mission will let them become familiar with CodeSpace.

Mission 2 Introducing CodeX:

An introduction to CodeX where you get your device connected and run some code! Students first learn about CodeX in the simulation. Then they connect the device and right some code.

Mission 3 Light Show:

This project introduces the CodeX pixel LEDs, variables and the sleep function. Students will learn about built-in colors and turn on the pixels in different colors and in different patterns. An optional extension is to create your own colors using RGB values. If time permits, a remix is provided.

Mission 4 Display Games:

Learn some CodeX display basics and create your first game. Students learn how to display built-in bitmap images, as well as text. They will convert integers to strings. Students will also learn about the buttons on CodeX and how to program them to do something, like input for a game.

Mission 5 Micro Musician:

Play music and sounds with the CodeX and learn about code readability. Students learn how to play a built-in mp3 file. They also learn about using comments and blank lines to enhance code readability.

Preparation and Materials:

- Create a class on the teacher dashboard.
- Students need a computer / laptop with the Chrome web browser.
- Make sure the students can successfully login to http://make.firialabs.com,
- Students create a student account and join the class with the code.
- Each student (or pair) needs a CodeX and connecting USB cable.
- Headphones are optional if you don't want to hear all the music files.

Assessment:

Mission 1-3 Review Kahoot	Mission 4 Review Kahoot	Mission 5 Review Kahoot	
U1 Vocab Review Kahoot	U1 Coding Review Kahoot	U1 Vocab Test (MS Form)	U1 Coding Test (MS Form)

Standards addressed in this unit:

CSTA Standards Grades 6-8	CSTA Standards Grades 9-10	CSTA Standards Grades 11-12
● 2-CS-03 ● 2-AP-10 ● 2-AP-11 ● 2-AP-13 ● 2-AP-15 ● 2-AP-16 ● 2-AP-19 ● 2-IC-20	● 3A-CS-03 ● 3A-AP-13 ● 3A-AP-16 ● 3A-AP-19 ● 3A-AP-21	● 3B-AP-16 ● 3B-AP-17 ● 3B-AP-21 ● 3B-AP-22 ● 3B-AP-23

Mission 1: Welcome

Time Frame: 30-60 minutes

Project Goal: Students will learn about the CodeSpace learning environment.

Learning Targets

- I can **navigate CodeSpace**.
- Identify major parts of the Codespace interface: Mission Bar, Objective Panel, text editor, CodeTrek, Toolbox, and Lesson Navigation Controls

Key Concepts

- **Follow instructions** in the Lesson Panel carefully. There is a lot of important reading!
- Look for "**tool icons**" to collect tools in your Toolbox as you go.

Assessment Opportunities

- Quiz after Objective 4.
- Print a picture of CodeSpace and have students label the parts.

Success Criteria

- □ Navigate CodeSpace
- □ Identify major features of the CodeSpace interface: Editor panel, Lesson panel, Toolbox, CodeTrek, Hints

Teacher Resources

- Use the Mission 1 Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, Workbook, and solutions

Vocabulary

- **Browser**: Software that displays web pages
- **Cloud:** A place to save files and data through the Internet
- **Objective:** The steps in the mission; has a goal to accomplish
- **Text editor:** Where you type the code
- **Code:** Instructions to the computer
- **Toolbox:** A place in CodeSpace to keep information you learn about programming concepts so you can use it later when you need the information
- **Debugging:** The process of understanding what the computer is actually doing and then changing the code to do what you want it to do

Real World Applications

Programmers need to use some type of text editor to create their code. CodeSpace is an IDE, or integrated development environment. It is patterned after other popular IDEs.

Teacher Note:

- This lesson is the first lesson in all the mission packs. If your students have completed other mission packs with other physical devices, they will already know the information. You can choose to have them complete the mission as a review and refresher, or you can unlock the next mission.

Mission 2: Introducing CodeX

Time Frame: 30-60 minutes

Project Goal: Students will learn about the peripherals of CodeX and the basics of Python.

Learning Targets

- I can identify the **main components** of CodeX.
- I can safely **connect** and disconnect **the CodeX** using the USB cable.
- I can create a **new file**.
- I can write code using the conventions of capitalization and punctuation specific to Python.

Key Concepts

- The CodeX is a powerful general-purpose computing device you can use to build an infinite number of cool projects.
- Python requires all objects – variables, peripherals, etc. – to be spelled exactly the same; **capitalization matters**!
- Your code is automatically saved to the file name you create.

Assessment Opportunities

- Print a picture of the CodeX and have students label the parts.
- Quiz after Objective 2.
- Quiz after Objective 7.
- Submit the final program.

Success Criteria

□ Identify the parts of the CodeX
□ Create a new file
□ Import the codex module
□ Write a program, load it to CodeX and run it

Teacher Resources

- Use the Mission 2 Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, Workbook, and solution

Vocabulary

- **CPU:** Central Processing Unit, or the brain of the computer
- **Peripherals:** Devices that give input or output to the CodeBot; they include LED lights, speaker, motors, line sensors, proximity sensors, an accelerometer and push buttons

New Python Code

`from codex import *`	Import codex module (library)
`display.show(pics.HEART)`	Display a built-in bitmap image
`(all built-in images)`	pics.HEART pics.HEART_SMALL pics.MUSIC pics.HAPPY pics.SAD pics.SURPRISED pics.ASLEEP pics.TARGET pics.TSHIRT pics.PLANE pics.HOUSE pics.TIARA pics.ARROW_N pics.ARROW_NE pics.ARROW_E pics.ARROW_SE pics.ARROW_S pics.ARROW_SW pics.ARROW_W pics.ARROW_NW

Real World Applications

Make sure each student takes the time to personally inspect their CodeX. Discuss the fact that all the electronic devices they use have similar circuit boards inside. The tools and techniques they're learning apply to all the electronic devices they use every day! Challenge students to name a few devices they use every day that might contain computer chips or "microcontrollers" such as the one on the 'bot. How many of the following do they think of? There are so many more!

Microwave oven Cell phone Automobile Watch or fitness tracker
Video game controller Refrigerator Home thermostat Coffee maker
Bread machine Alarm system Electronic locks Automatic garage doors

Challenge students to describe how our lives are impacted by the above technology, and to compare how related tasks were done before computer technology was invented. (Use this discussion question or a similar one from the CSTA Standards for your grade band.)

Mission 3: Light Show

Time Frame: 45-60 minutes

Project Goal: Students will learn about CodeX pixels.

Learning Targets

- I can use the “Step” feature to **debug** a program.
- I can assign **data** to a **variable**.
- I can use **variables** to make code more efficient.
- I can turn on a NeoPixel LED to a specific color.
- I can use the sleep() function to slow down the execution of a program.

Key Concepts

- Computers execute code in **sequential** steps.
- The CodeSpace debugger lets you *step* through the code one line at a time to understand what the computer is doing.
- Built-in functions come from **modules or libraries**, like **codex** or **time**.
- **Variables** can be defined to hold changing values.

Assessment Opportunities

- Quiz after Objective 4
- Quiz after Objective 8
- Submit the **Pixels1** program
- Mission 1, 2, 3, Review Kahoot

Success Criteria

□ Learn how to use the CodeSpace debugger
□ Define and use a variable delay in sleep()
□ Define and use a variable for color that is changed and used multiple times
□ Write a program, run it, and save it
□ Clear the CodeX of meaningful code

Teacher Resources

- Use the Mission 3 Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, Workbook, and solution

Vocabulary

- **RGB:** Red, Green, Blue; the colors that make up a single pixel on the screen
- **Sequential:** Executing code line by line, one after another, in order
- **Literal:** A specific value, like 1 or “hello”
- **Bug:** An error in the code (like a typing mistake, indenting problem, missing punctuation, etc.)
- **Variable:** A name you assign to some data used in code instead of the literal, or actual, values
- **Assign:** Give a variable a value (bind a name to a value)

New Python Code

`pixels.set(0, GREEN)`	Turn on ONE NeoPixel LED (they are numbered 0, 1, 2, 3)
`(all built-in colors)`	BLACK YELLOW GRAY PINK BROWN GREEN WHITE LIGHT_GRAY RED BLUE CYAN DARK_GREEN ORANGE PURPLE MAGENTA DARK_BLUE

`from time import sleep`
`from time import *`
	Import the time module (library) to use the sleep() function
	Either line of code will work

`sleep(1)`	Cause a pause or delay in the code – this examples pauses for 1 second
`delay = 1  /  color = RED`	Define a variable (and assign a value)
`sleep(delay)`	Use a variable with the sleep() function

Instructions for using the debugger are included in this Mission (Objectives 5 & 6)

Real World Applications

The skills used in this project are used by professional software developers to build:

- Traffic lights
- Sports event scoreboards
- Games, etc.

Discuss the use of lights and controlling them in various aspects of daily life.

Mission Extension: Have the CodeX put on a light show, like fireworks.

Teacher Notes

- Always start a new program by creating a new file and naming it appropriately. If you don't, you will lose all your previous work!
- Using descriptive file names is essential to finding the program later!
- Students are making a project – not just working random problems. Focus on the *project-based* objectives and avoid rushing through the material too quickly.
- Have students test their understanding along the way by "coloring outside the lines". Try stuff!
- Collect all the Tools they find! (They're indicated with a wrench icon)
- Read carefully – usually the answer is right there in front of you!
- You may consider having students (or the class collectively) keep a chart of errors and the ways to fix them.
- You can also add vocabulary to a word wall and keep a document or chart of the Python code learned during each mission.

Mission 3 Remix

Time Frame: 1-3 hours

Project Goal: Students will use the skills and concepts they learned in the first three missions to create their own project.

Project Outline: Follow the five-steps of the design process to design a remix project.

Assessment Opportunities

- Peer reviews / Gallery walk
- Remix Planning Guide (5-step process)
- Remix Rubric Checklist
- Submit Remix Program
- Mission 1, 2, 3, Review Kahoot
- Student Reflection

Teacher Resources

- Use the Mission 3 Remix Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, Workbook, and solutions.

Vocabulary

- **Remix:** Creating something original based on other projects, or using pieces of code from other projects
- **Tuple:** A triplet of numbers that represents an RGB value – example: (47, 147, 181)

New Python Code

`display.fill(BLACK)`	Clear the display
`pixels.set(0, BLACK)`	Clear a NeoPixel (turn it black)
`from random import randrange`	Import the random module (library)

`red = randrange(256)`
`green = randrange(256)`
`blue = randrange(256)`
	Assign a random color (RGB)

`color = (red, green, blue)`	Assign a color variable the random RGB
`pixels.set(0, color)`	Use the color variable to set a pixel

Remix Ideas:

- **MILD**: Select an RGB color for all four pixels and display an image. After a short sleep(delay), select a different RGB color and image. Repeat as many times as you want. Refer to “Using RGB values” for help.
- **MEDIUM**: Light all pixels with an RGB color. Then turn off the pixels and display an image. Then clear the screen and light the pixels with another RGB color. Repeat as many times as you want.
- **SPICY**: Use one of the ideas from Mild or Medium, but instead of using an RGB color that you choose, use random colors. Use the Page 8 challenge for help.
- Think of your own creative project with pixels and images.

Cross-Curricular

- **ART:** Discuss how RGB creates all colors in light.
- **COMPUTATIONAL THINKING:** The lesson requires students to program multiple steps and coordinate pixels with images. They must think algorithmically and step-by-step to plan their project. They will follow the design process.
- **SCIENCE:** The CodeX pixels use light to shine colors. Discuss the science behind colors and light.
- **MATH:** Computers and CodeX do math really well and really fast. Using a variable, gradually increase (slow down) and decrease (speed up) the amount of delay between setting a pixel. Discuss how the math works in this situation.
- **LANGUAGE ARTS:** Have students write about technology in their daily lives. Or have them take a position on technology being helpful or harmful and defend their position.

- Supports **language arts** through peer review and reflection writing.

Rubric Checklist:

□ Write an original program, run it, and save it to the CodeX
□ Follow the design process and document their work in the log assignment
□ Use at least one variable in the program
□ Change the color of at least one pixel
□ Display at least one image on the screen
□ Debug any errors in the code so the program works correctly
□ Complete a review of their original program
□ Clear the CodeX of meaningful code

Mission 4: Display Games

Time Frame: 60-120 minutes

Project Goal: Students will explore the CodeX’s LCD display and push buttons.

Learning Targets

- I can explain the difference between **integer** and **string** data types and use each appropriately.
- I can convert between **string** and **integer** data types.
- I can write an **if:else** conditional statement.

Key Concepts

- Values have different data types, like integer and string
- Computers can make a selection through branching using **if:else** statements.
- Indenting after a colon is very important! This creates a block of code.
- Batteries can make your CodeX portable.

Assessment Opportunities

- Quiz after Objective 6
- Quiz after Objective 9
- Daily exit ticket
- Submit flowchart or pseudocode for the project
- Mission 4 Review Kahoot
- Student reflection
- Submit final **Display** program

Success Criteria

□ Understand and use variable data types, converting types when needed

□ Use a Boolean condition in an if..then statement

□ Receive input from the user through a button

□ Program a button to make a fast-click game

□ Debug any errors in the code

□ Write a program, run it, and save it

Teacher Resources

- Use the Mission 4 Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, Workbook, and solutions

Vocabulary

- **Integer:** A whole number that can be positive, negative or zero
- **String:** A sequence of characters, like words or sentences
- **Conversion Function:** A built-in function that converts a value to a different (and specific) data type
- **Branching:** Decision points in code; a condition
- **Boolean:** True or False data type (values that can be True or False)
- **Indentation:** Structuring blocks of code in Python; statements ending with a colon (:) execute the block of code indented four spaces beneath it

New Python Code

`display.show(“Ahoy”)`	Display a word, text or string data type (use “ “ to indicate text)
`word = str(number)`	Convert a number to a string
`number = int(string)`	Convert a string to an integer

`display.show(str(9))`
`display.show(str(number))`
	Display a number> It can be a literal value, like 9
	Or a variable

`display.print(“Jack and Jill”)`
`display.print(“went up a hill”)`
`display.print(“to fetch a pail”)`
	Display more than one line of text
	<Use “print” instead of “show”>

`pressed = True`
`if pressed:`
`    pixels.set(0, GREEN)`
`else:`
`    pixels.set(0, RED)`
	If:else statement for branching
	Look for the : and the indenting – very important!

`pressed = buttons.is_pressed(BTN_A)`
`pressed = buttons.was_pressed(BTN_B)`
	Assign a value (True or False) if a button is pressed
	The “is_pressed” checks if it is currently being pressed
	The “was_pressed” checks if it was pressed since the last check

Real World Applications

The skills used in this project are used by professional software developers to build:

- Traffic lights
- Sports event scoreboards
- Games, etc.

The Mission controls lights, displays images and text, and accepts input from the user through a button press. Discuss devices that do this in our daily lives.

Mission 4 Remix

Time Frame: 1-3 hours

Project Goal: Students will use the skills and concepts they learned in the first four missions to create their own project.

Project Outline: Follow the five-steps of the design process to design a remix project.

Assessment Opportunities

- Peer reviews / Gallery walk
- Remix Planning Guide (5-step process)
- Remix Rubric Checklist
- Submit Remix Program
- Mission 4 Review Kahoot
- Student Reflection

Teacher Resources

- Use the Mission 4 Remix Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, Workbook, and solutions

Remix Ideas:

- **MILD**: Add images to the code. When a pixel turns green, display an image like a happy face. Then the pixel turns red, display a different image, like a sad face.
- **MEDIUM**: Use images instead of printed instructions. There are arrow images available. Display an arrow and check for the correct button. For example, the north arrow points up and can be used for BTN_U.
- **SPICY**: Use a color of pixel to indicate which button to press. For example, if a pixel turns RED, press BTN_U. If a pixel turns BLUE, press BTN_D. And so forth. You can use six colors and all six buttons. Display an image (like a sad face) if they are incorrect. You may want to include a set of print statements at the beginning as instructions that tell the button and color combination.
- Think of your own creative project with buttons, pixels and images.

Cross-Curricular

- **MATH:** Students learn that computers can do math really fast. Use the computer as a calculator to check their work.
- **COMPUTATIONAL THINKING:** The lesson requires students to program multiple steps and coordinate pixels with images. They must think algorithmically and step-by-step to plan their project. They will follow the design process.
- **LANGUAGE ARTS:** Have students write about how their code works, like a technical guide.
- Supports **language arts** through peer review and reflection writing.

Rubric Checklist:

□ Write an original program, run it, and save it to the CodeX
□ Follow the design process and document their work in the log assignment
□ Use conditions for at least four button presses
□ Debug any errors in the code so the program works correctly
□ Complete a review of their original program
□ Clear the CodeX of meaningful code

Mission 5: Micro Musician

Time Frame: 30-60 minutes

Project Goal: Students will learn how to play mp3 files on CodeX.

Learning Targets

- I can make the CodeX play music through the speaker or headphones.
- I can add comments to make the code readable.
- I can add blank lines to make the code readable.

Key Concepts

- You can import new code modules that have fantastic capabilities (like music) with Python's **import** statement.
- Batteries can make your CodeX portable.
- Code should be readable to everyone. Two ways to accomplish this are to add meaningful comments and blank lines.

Assessment Opportunities

- Quiz after Objective 5
- Daily exit ticket
- Mission 5 Review Kahoot
- Submit final **Music1** program
- Student Reflection

Success Criteria

☐ Create a program that plays an audio file on the CodeX

☐ Add readability to your program by adding blank lines and comments

☐ Debug any errors in the code

☐ Clear the CodeX of meaningful code

Teacher Resources

- Use the Mission 5 Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, Workbook, and solutions.

Vocabulary

- **Readability:** Making code easy to understand for humans
- **Comments:** Notes in code that are ignored by the computer but can explain what the code does
- **Analog (optional):** Smooth and continuous signals that represent a quantity, like sound waves
- **Digital (optional):** A numerical representation of an analog signal, represented in increments

New Python Code

`audio.mp3("sounds/welcome")`	Plays a built-in audio clip
`(all built-in audio clips)`	a.mp3 africa.mp3 b.mp3 funk.mp3 led.mp3 left.mp3 yes.mp3 seven.mp3 eight.mp3 five.mp3 four.mp3 bohemia.mp3 button.mp3 codetrek.mp3 codex.mp3 off.mp3 okay.mp3 on.mp3 one.mp3 power.mp3 right.mp3 mic.mp3 six.mp3 techstyle.mp3 ten.mp3 three.mp3 two.mp3 welcome.mp3 nine.mp3 down.mp3 no.mp3 shire.mp3 zero.mp3 display.mp3 up.mp3 roll.mp3

Real World Applications

- Computer controlled music sequences are very common. You can probably think of many more instances where computer-controlled tones are used for entertainment, alarms and alert messages, etc.

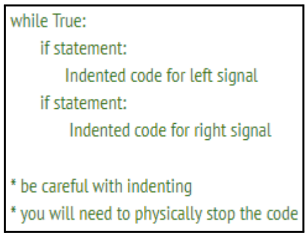
Mission 5 Remix	**Time Frame:** 1-3 hours
Project Goal: Students will use the skills and concepts they learned in the first five missions to create their own project. **Project Outline:** Follow the five-steps of the design process to design a remix project.	**Assessment Opportunities** ● Peer reviews / Gallery walk ● Remix Planning Guide (5-step process) ● Remix Rubric Checklist ● Submit Remix Program ● Mission 5 Review Kahoot ● Student Reflection

Teacher Resources

- Use the Mission 5 Remix Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, Workbook, and solutions

Remix Ideas:

- **MILD**: Add a voice prompt for the game created in Mission 4 (Display). Either replace each *display.print()* with an audio prompt for which button to press, or use both text and audio.
- **MILD**: Open the Mission 4 program (Display). Add an audio file to the if statements that will indicate correct “yes” or wrong “no”.
- **MEDIUM**: Program each button to do a different task. For example, BTN-A could light pixels and display an image. BTN_B could display an image and play audio. BTN_U could cause pixels to blink random colors, and so forth.
- **MEDIUM**: Create turn signals for a bicycle. Press BTN_L to turn left and BTN_R to turn right. Blink the pixels to indicate the turn, and use an audio file to say the direction you are turning.
  EXTENSION: Use a *while True* loop to use the signals more than once.
  (see sidebar)
- **SPICY**: Combine the two mild remixes into one program – audio prompt for the button and audio file for correct or wrong. Also, add print statements at the beginning for an intro and a print statement at the end.
- **SPICY**: Program at least three buttons to display a short poem and play an audio file.
- Think of your own creative project with buttons, pixels and images.



```
while True:
    if statement:
        Indented code for left signal
    if statement:
        Indented code for right signal

* be careful with indenting
* you will need to physically stop the code
```

Cross-Curricular

- **MUSIC:** Students play pre-recorded audio clips. Expand the lesson to discuss sound waves or the difference between mono and stereo sound.
- **TECHNOLOGY:** Go through the additional slide deck on analog and digital (remix). Have students give examples.
- **LANGUAGE ARTS:** A remix idea is to write a short poem and play music while it is displayed.
- **COMPUTATIONAL THINKING:** The lesson requires students to program multiple steps and coordinate pixels with images. They must think algorithmically and step-by-step to plan their project. They will follow the design process.
- **LANGUAGE ARTS:** Have students write a poem about programming or music. Or have students write lyrics for a song.
- Supports **language arts** through peer review and reflection writing.

Rubric Checklist:

- ☐ Write an original program, run it, and save it to the CodeX
- ☐ Follow the design process and document their work in the log assignment
- ☐ Add readability to your program by adding blank lines and comments
- ☐ Use at least two audio clips in the program
- ☐ Debug any errors in the code so the program works correctly
- ☐ Clear the CodeX of meaningful code

Unit 1 Assessments

Time Frame: 2-5 hours

Project Goal: Assess students for mastering Unit 1 Vocabulary, Coding and Concepts.

Learning Targets

- I can create my own original program using concepts from Missions 3, 4 and 5.
- I can define vocabulary words from Missions 3-5.
- I can answer multiple choice questions that test coding and concepts from Missions 3, 4, and 5.

Assessment Opportunities

- Remix Planning Guide (5-step process)
- Remix Rubric Checklist
- Remix Project Rubric
- Submit Remix Program
- Student Reflection
- Unit 1 Vocabulary Exam
- Unit 1 Coding and Concepts Exam

Teacher Resources

- Kahoot Reviews: Unit 1 Vocabulary / Unit 1 Coding and Concepts
- For a Unit 1 Remix, use the Mission 5 Remix Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- **NOTE:** *If students have already created remix programs for the missions in this unit, you can skip the remix here. If they haven't done any remixes so far, they should do at least one before moving to the next unit. Depending on the time you have to spend in the curriculum, you might find value in having students do a Unit 1 Remix, even if they have done remixes for other missions.*
- You can find two rubrics to use for the Unit 1 Remix: CodeX Project Rubric and/or CodeX Sliding Rubric
- Use the planning guide from one of the Unit Remix assignments.
- End the unit with the Unit 1 Vocabulary Exam and Unit 1 Coding and Concepts Exam

Remix Ideas:

- Pick a remix idea from Mission 3, Mission 4 or Mission 5. Try one you haven't done yet.
- Combine remix ideas from the different missions into your own original program.
- Create a "Pandora' Box" program. Make each button cause a different action when pressed. Use all the concepts from Mission 3, Mission 4 and Mission 5 (pixels, flashing lights, random colors, displaying text, displaying images, changing the display color, and adding audio).
- Think of your own creative project. Maybe it will be a game. Maybe it will solve a problem. Maybe it will incorporate science, math or language arts. It is up to you!

Rubric Checklist:

□ Light up at least one pixel
□ Use at least one variable in the program
□ Use sleep() at least once
□ Display at least one image on the LCD
□ Display some text on the LCD
□ Play at least one mp3 audio file
□ Take input from at least 4 buttons
□ Add readability to your program by adding blank lines and comments
□ Use strategies to debug the code
□ Program should work correctly and without any errors

Unit 2: Inputs and Outputs (7-19 hours)

Students continue their programming journey by updating variables and using lists. Students first learn to increment and decrement a variable to adjust the speed of a flashing image. Then they learn about lists and how they can manage complexity. They learn the basics of lists, like how to create a list, access an item in the list, and scroll through a list using a variable for an index. Finally, students learn about random numbers and functions that are in the random module and how they can be used with lists.

Mission 6 Heartbeat:

Students update a variable by incrementing and decrementing the delay. The delay is used to switch between two heart images. The **while True** loop is used to run the code continuously.

Mission 7 Personal Billboard:

Students use a list to scroll through different images, text and colors to display a mood or topic. They learn about data types and converting data types.

Mission 8 Answer Bot:

Students again use a list, and use random functions to pick from the list like a Magic 8 Ball.

Preparation and Materials:

- Students need a computer / laptop with the Chrome web browser.
- Make sure the students can successfully login to http://make.firialabs.com
- Students login with their account (possibly using their gmail account)
- Each student (or pair) needs a CodeX and connecting cable.
- **Teacher Note:** At this point, we suggest introducing the students to features of the Editor they may not be aware of, such as **cut**, **copy**, **paste**, and **undo**. There is a Toolbox tool covering these "Editor Shortcuts", and they're also explained in a video at https://youtu.be/WItk_kpkGiU.

Assessment:

Mission 6 Review Kahoot	Mission 7 Review Kahoot	Mission 8 Review Kahoot	
U2 Vocab Review Kahoot	U2 Coding Review Kahoot	U2 Vocab Test (MS Form)	U2 Coding Test (MS Form)

Standards addressed in this unit:

CSTA Standards Grades 6-8	CSTA Standards Grades 9-10	CSTA Standards Grades 11-12
• 2-CS-03 • 2-AP-10 • 2-AP-11 • 2-AP-13 • 2-AP-15 • 2-AP-16 • 2-AP-19 • 2-IC-20	• 3A-CS-01 • 3A-CS-03 • 3A-DA-09 • 3A-AP-13 • 3A-AP-14 • 3A-AP-16 • 3A-AP-19 • 3A-AP-21 • 3A-AP-22 • 3A-AP-23 • 3A-IC-26	• 3B-CS-02 • 3B-AP-10 • 3B-AP-12 • 3B-AP-16 • 3B-AP-17 • 3B-AP-21 • 3B-AP-22 • 3B-AP-23

Mission 6: Heartbeat

Time Frame: 45-90 minutes

Project Goal: Students will use a variable and images in a loop to create a beating heart animation.

Learning Targets

- I can use a ***while*** loop to make my code more efficient.
- I can apply ***variables*** to a new program.
- I can program using ***float*** values.

Key Concepts

- You create an infinite loop when it is always True.
- Readability is very important in coding. Whitespace can help, and comments are essential.
- Use the BREAK control to halt your program and break out of the loop.
- You can use a variable to hold the *current state* of your program. In this case, it is the speed of the heartbeat, or the *delay* between beats.

Assessment Opportunities

- Quiz after Objective 7
- Quiz after Objective 11
- Daily exit ticket
- Mission 6 Review Kahoot
- Submit final **Heart2** program
- Student Reflection

Success Criteria

□ Create a program that shows a beating heart using an infinite loop

□ Include code in the program that speeds up and slows down the heart

□ Debug errors in the code (except the last one!)

Teacher Resources

- Use the Mission 6 Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, Workbook, and solution

Vocabulary

- **Loop:** Repeats a block of code, subject to a given condition (iteration)
- **Condition:** An expression that evaluates to True or False (example: num < 5 or choice == 1)
- **While Loop:** Repeats a block of indented code as long as the condition is true
- **Infinite Loop:** A loop that never ends because the condition is always true
- **Float:** A real number, or a number with a decimal point (called a floating point)
- **Increment:** Increase the value of a variable by a set amount (example: num = num + 1)
- **Decrement:** Decrease the value of a variable by a set amount (example: num = num - 1)

New Python Code

`while True:`
`    # indent code to loop`
`    display.show(pics.HEART)`
`    sleep(delay)`
	Infinite while loop

`if buttons.was_pressed(BTN_A):`
`    break`
	Halt the program execution by breaking out of a loop.
	(Can be any button)

`delay = delay + 0.2`	Increment a variable (delay)
`delay = delay - 0.2`	Decrement a variable (delay)

Real World Applications

- Ask students if they have seen flashing road signs. Can they think of other continuously blinking or repeating indicators? How can computers make this more efficient?
- You have programmed an embedded computer to monitor push buttons to control speed UP and DOWN. That's real-world code used in many applications you see around you.
  - TV remote, game controllers, push button light dimmers, etc.

Mission 6 Remix	Time Frame: 1-3 hours
Project Goal: Students will use the skills and concepts they learned in the first six missions to create their own project. **Project Outline:** Follow the five-steps of the design process to design a remix project.	**Assessment Opportunities** ● Peer reviews / Gallery walk ● Remix Planning Guide (5-step process) ● Remix Rubric Checklist ● Submit Remix Program ● Mission 6 Review Kahoot ● Student Reflection

Teacher Resources

- Use the Mission 6 Remix Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, Workbook, suggestions and solutions

New Python Code

`audio.pitch(my_sound, 0.5) / audio.pitch(520, delay)`	Play a tone

Remix Ideas:

- **MILD**: Make one or more pixels flash on and off, instead of an image on the screen.
- **MILD**: Add flashing pixels to the heartbeat. Add one, two or all of them.
- **MEDIUM**: Add another if statement for a different button press that will break out of the loop and stop the program. Add text that lets the user know the program has ended.
- **MEDIUM**: Change the flashing image to pixels that roll, turning on one pixel at a time. One sequence will be pixel 0, pixel 1, pixel 2 and then pixel 3. Repeat the sequence in a loop, with the option to slow down or speed up.
- **MEDIUM**: Add two pitch sounds to the heartbeat, one for each image. Use the delay variable in the pitch statement instead of a sleep() statement. Example: audio.pitch(440, delay)
- **SPICY**: Add code that will prevent the runtime error (delay goes below 0).
- **SPICY**: Change the flashing image to a left (west) or right (east) arrow. Use different button presses to change from one image to the other.
- **CHALLENGE**: Add a menu that explains what each button press will do, and another break to hold the menu until the user wants to start. Also, include an ending message. If you include flashing lights, use another variable for the color.

Cross-Curricular

- **MATH:** Students learn the difference between integer and real (float) data types. An extension lesson can have students identify number types and discuss the similarities and differences.
- **MATH:** The program introduces incrementing and decrementing values. This is like counting by a specific number, or multiples. An extension lesson can be built around multiples of numbers, or counting by a specific number (count by 3s, etc.)
- **COMPUTATIONAL THINKING:** The lesson requires students to program multiple steps and coordinate pixels with images. They must think algorithmically and step-by-step to plan their project.
- Supports **language arts** through peer review and reflection writing.

Rubric Checklist:

□ Write an original program, run it, and save it to the CodeX
□ Follow the design process and document your work in the log assignment
□ Add readability to your program by adding blank lines and comments
□ Change the value of at least one variable by pressing a button
□ Use at least two buttons as input to affect the code
□ Debug any errors in the code so the program works correctly
□ Complete a review or reflection of their original remix program

Mission 7: Personal Billboard

Time Frame: 60-120 minutes

Project Goal: Students will synthesize their skills to create their own personal billboard.

Learning Targets

- I can create a list to use abstraction in my code.
- I can distinguish between string and image data types.
- I can apply an **if/else** conditional statement to solve a problem.

Key Concepts

- An infinite loop is a good way to continuously check for button presses. Many programs follow this pattern, with a “main loop” that monitors and acs on events as they occur.
- You can use a number variable to track the state of a program.
- In Python, the double equals sign is used to compare for equality.
- CodeSpace lets you inspect variables in the debugger.
- Python’s list is a powerful way to hold a collection of items.
- You can check the type of a variable and deal with it accordingly!

Assessment Opportunities

- Quiz after Objective 3
- Quiz after Objective 6
- Daily exit ticket
- Mission 7 Review Kahoot
- Submit final **Billboard** program
- Student Reflection

Success Criteria

□ Program the buttons to scroll through a series of images.

□ Change the code by using a list to make the program easy to add lots more images.

□ Mix text messages, colors and images in the list

□ Add an if statement that will correctly display a text string, image or color

□ Debug errors in the code; code works correctly

Teacher Resources

- Use the Mission 7 Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, Workbook, and solution

Vocabulary

- **Index:** A number that keeps track of what choice should be displayed in a list.
- **Comparison Operators**: Operators that let you compare two values; the result is True or False. Comparison operators include: ==, <, >, <=, >=, !=
- **List:** A sequence of items you can access with an index.

Review from Mission 6:

- **Increment:** Increase the value of a variable by a set amount (example: num = num + 1)
- **Decrement:** Decrease the value of a variable by a set amount (example: num = num - 1)

New Python Code

<table>
<tr><td><pre>if choice == 0:
    # do something</pre></td><td>Compare a variable to a specific value</td></tr>
<tr><td><pre>LAST_INDEX = len(my_list) - 1</pre></td><td>Last index of an item in a list is one less than the length (computers start counting at 0, not 1)</td></tr>
<tr><td><pre>if buttons.was_pressed(BTN_L):
    choice = choice + 1
    if choice > LAST_INDEX:
        choice = 0</pre></td><td>List index wrap around scrolling forward (end, back to beginning of list)</td></tr>
</table>

`if buttons.was_pressed(BTN_L):`
`    choice = choice - 1`
`    if choice < 0:`
`        choice = LAST_INDEX`
	List index wrap around scrolling backward
	(beginning, back to end of list)

`my_list = [pics.HAPPY,`
`           pics.SAD,`
`           pics.SURPRISED]`
	Define (or create) a list
	The list definition can list each item going down, or all across in one line. It doesn't matter as long as a comma (,) separates each item.

`index = 3`
`my_item = my_list[index]`
`my_item = my_list[2]`
	Access an item (element) from the list using an index

`IN THE CONSOLE:`
`>>> type(7)`
`<class ‘int’>`
`>>> type(1.15)`
`<class ‘float’)`
`EXAMPLES IN CODE:`
`my_type = type(7)`
`if my_type(my_item) == tuple:`
	Get the data type of a variable
	You can get the data type in your code or by using the Console.

Real World Applications

- Have you ever made a sign to post on a door or wall? How about a name badge to wear? Or a cap or t-shirt with a message or slogan on it? In this project, students will build a device that lets them display images or text, and even use the CodeX's buttons to select what is displayed to suit a particular occasion or mood. This project has practical applications.
    - Imagine if everyone at a party or meeting wore CodeX badges like this, to show their name and interests.
    - Or what if you made this into a cap for a sports team, to give out to all the fans before a big game?
- Have you ever scrolled through a menu system on a website or video game? Software applications of all kinds deal with lists or collections of different types of data. The code you've learned to write in this project has thousands of real world applications.
    - What if you made it so instead of button presses, the selections just advanced on their own? The list could be animation sequences, or the set list for a band. You decide!

Mission 7 Remix	**Time Frame:** 1-3 hours

Project Goal: Students will use the skills and concepts they learned in the first seven missions to create their own project.

Project Outline: Follow the five-steps of the design process to design a remix project.

Assessment Opportunities

- Peer reviews / Gallery walk
- Remix Planning Guide (5-step process)
- Remix Rubric Checklist
- Submit Remix Program
- Mission 7 Review Kahoot
- Student Reflection

Teacher Resources

- Use the Mission 7 Remix Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, Workbook, suggestions and solutions

New Python Code

`display.print("Hello\nthere")`
`-will print as-`
`Hello`
`there`
	Use the escape character **\n** in a print() statement to go to a new line.

`leds.set(LED_A, True)`
`leds.set(LED_B, False)`
	Turn on or off the red LED above BTN_A and BTN_B.

Remix Ideas:

- **MILD**: Select a button other than A or B and add an if statement that will break out of the loop to stop the program. Add a text message that lets the user know the program has ended.
- **MILD**: Create a list of 6 items (images, text or colors). Assign a value to choice when the button is pressed. Then display the image, text or color from the list – no scrolling needed.
- **MEDIUM**: Create a list of 6 items, like Mild #2. At the beginning, and after each item, clear the screen to black and print "Press a button."
- **MEDIUM**: Add print statements that introduce the program. Add a loop that will "wait" until a button is pressed to begin. Also, include a button that will break out of the loop and stop the program. Display an ending message.
- **MEDIUM**: Create two lists: one for images and one for sounds. Using the same "choice" variable, display an image AND play a sound. You can use the audio sound files or play tones.
- **MEDIUM**: Create a list of text strings with facts from math, science, history, etc. Scroll through and display the list of facts. You will need to sleep(), clear the display, and print a scroll message. HINT: Use the \n escape character in a string to print a new line.
- **SPICY**: Create 2 lists and use BTN_A and BTN_B to determine which list to display. Then use BTN_L and BTN_R to scroll through the lists. The lists can be anything that interests you, or facts from two different subjects or units.
- **CHALLENGE**: Complete the spicy program. Add another list with colors (define your own!). Use the list to light the pixels a different color for each corresponding item in the list. OPTIONAL: Light up the red LED above BTN_A or BTN_B when the button is pressed so you know what list is being displayed.

Cross-Curricular

- **MATH:** Students learn the six different comparison operators. An extension lesson can have students use comparison operators with numbers and in various math scenarios.
- **CROSS-CURRICULAR:** Students learn about lists. Take time in one of your core subjects to brainstorm how you can use lists. Or utilize lists in some way with another subject.
- **COMPUTATIONAL THINKING:** The lesson requires students to think algorithmically and step-by-step to plan their project. They will follow the design process.
- **LANGUAGE ARTS:** This mission is about personal expression. Write an “all about me” essay.
- Supports **language arts** through peer review and reflection writing.

Rubric Checklist:

□ Write an original program, run it, and save it to the CodeX
□ Follow the design process and document your work in the log assignment
□ Add readability to your program by adding blank lines and comments
□ Use at least one variable that represents an index
□ Use at least one list
□ Use at least two buttons as input to affect the code
□ Debug any errors in the code so that it runs correctly
□ Complete a review or reflection of their original remix program

Mission 8: Answer Bot

Time Frame: 45-90 minutes

Project Goal: Students will use a list and main while loop to create a playlist or a bank of random answers.

Learning Targets

- I can create a list to use abstraction in my code.
- I can apply properties of lists in my program.
- I can utilize multiple variables in a program and describe their purpose.
- I can apply I/O (input and output) to make my code more efficient..

Key Concepts

- Random number generators are crucial for many computer applications.
- The CodeX uses a pseudo random number generator, which means the "random" numbers it provides are really just a fixed sequence that's meant to have an unpredictable pattern.

Assessment Opportunities

- Quiz after Objective 5
- Daily exit ticket
- Mission 8 Review Kahoot
- Submit final **Answer_Bot** program
- Student Reflection

Success Criteria

☐ Program button A to print a random number.
☐ Add a list of answers to the program.
☐ Modify the code to print a random message from a list of possible answers.
☐ Change the size of the text when printed on the screen
☐ Randomly assign a color to each pixel

Teacher Resources

- Use the Mission 8 Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, Workbook, and solution

Vocabulary

- **Item:** An individual element or value in a list
- **Index:** A number that keeps track of what choice should be displayed in a list. (review from Mission 7)
- **List:** A sequence of items you can access with an index. (review from Mission 7)

New Python Code

`import random`	Import the random module (library)

`number = random.randrange(10)`
`number = random.randrange(1, 6)`
	Generate a random integer (gives an integer between 0 and 9)
	(Gives a number between 1 and 5)
	** the default starting value is 0 unless specifically stated.
	** random numbers will go from the starting value to one less than the ending value

`display.print(number, scale=3)`	Change the size of the text. Scale adjust the text size. If the scale is too big, the text will appear as gibberish or shapes on the display. Default scale=1.

`color = random.choice(COLOR_LIST)`
`my_choice = random.choice(answers)`
	Select a random item from a list (this example uses a built-in list)
	(this example uses a programmer-defined list)

Real World Applications

- Up to this point CodeX has been pretty predictable. But some applications need randomness.
  - Games, where there shouldn't be an obvious pattern for the human player to learn.
  - Cryptography, where randomness helps secure stored passwords and messages.
  - Scientific studies, where statistical sampling requires random selection.

Mission 8 Remix	Time Frame: 1-3 hours
Project Goal: Students will use the skills and concepts they learned in the first eight missions to create their own project. **Project Outline:** Follow the five-steps of the design process to design a remix project.	**Assessment Opportunities** ● Peer reviews / Gallery walk ● Remix Planning Guide (5-step process) ● Remix Rubric Checklist ● Submit Remix Program ● Mission 8 Review Kahoot ● Student Reflection

Teacher Resources

- Use the Mission 8 Remix Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, Workbook, suggestions and solutions

New Python Code (from “Adding JPG Images” optional lesson)

```
display.draw_jpg(“teacherBear.jpg”)
x = “pics/teacherBear.jpg”
display.draw_jpg(x)
my_images = [“pics/teacherBear.jpg”,
             “pics/doggie.jpg”,
             “pics/goldfish.jpg”]
display.draw_jpg(random.choice(my_images)
```

Display a jpg image. Three different examples are given:

Draw an image with the file name in “ “
Assign a variable the file name and use in code
Choose a random file from a list of jpg files

Remix Ideas:

- **MILD**: Create your own list of colors. When a button is pressed, pick a random color and turn on all four pixels. Or use two buttons – one for pixels 0 & 1 and a different button for pixels 2 & 3.
- **MILD**: Create a list of images and/or colors. When a button is pressed, pick a random image/color to display. Also program a button to break out of the loop and end the program.
- **MILD**: Create a list of sounds. When a button is pressed, pick a random sound to play. Also program a button to break out of the loo and end the program.
- **MEDIUM**: Create two lists. The items can be colors, text, images or sound. The lists do not need to be similar items or the same number of items. Use BTN_A to select one of the lists and BTN_B to select the other list and display a random item from the list. Program a button to break out of the loop and end the program. Include instructions and a “wait” button.
- **SPICY**: Create two different lists, selected with BTN_A and BTN_B buttons. Program BTN_L to scroll backwards, BTN_R to scroll forwards and BTN_U to select a random item from the chosen list. Include an introduction, “wait” button and exit button.
- **SPICY**: Create a dice roller, showing the number of a 6-sided die. Change the scale of the text so the number is large. Show a message like “rolling, rolling, rolling” before displaying the number.

Cross-Curricular

- **MATH:** Students learn the difference between integer and real (float) data types. An extension lesson can have students identify number types and discuss the similarities and differences.
- **MATH:** The program introduces incrementing and decrementing values. This is like counting by a specific number, or multiples. An extension lesson can be built around multiples of numbers, or counting by a specific number (count by 3s, etc.)
- **COMPUTATIONAL THINKING:** The lesson requires students to program multiple steps and coordinate pixels with images. They must think algorithmically and step-by-step to plan their project.
- **LANGUAGE ARTS:** Have students compare and contrast Mission 7 to Mission 8.
- **SEL:** Students will decide on a question and create different answers for the question. This individuality supports social-emotional learning and engagement. A specific topic in social and emotional learning could be used as a question, and this could follow an SEL lesson.

- Supports **language arts** through peer review and reflection writing.

Rubric Checklist:

☐ Write an original program, run it, and save it to the CodeX
☐ Follow the design process and document their work in the log assignment
☐ Add readability to your program by adding blank lines and comments
☐ Use at least one variable that represents an index
☐ Use at least one list
☐ Use at least two buttons as input to affect the code
☐ Use a random function at least once
☐ Debug any errors in the code, so the code runs correctly
☐ Complete a review or reflection of the final remix project

Unit 2 Assessments

Time Frame: 2-5 hours

Project Goal: Assess students for mastering Unit 2 Vocabulary, Coding and Concepts.

Learning Targets

- I can create my own original program using concepts from Missions 6, 7 and 8.
- I can define vocabulary words from Missions 6-8.
- I can answer multiple choice questions that test coding and concepts from Missions 6, 7 and 8.

Assessment Opportunities

- Remix Planning Guide (5-step process)
- Remix Rubric Checklist
- Remix Project Rubric
- Submit Remix Program
- Student Reflection
- Unit 2 Vocabulary Exam
- Unit 2 Coding and Concepts Exam

Teacher Resources

- Kahoot Reviews: Unit 2 Vocabulary / Unit 2 Coding and Concepts
- For a Unit 2 Remix, use the Mission 8 Remix Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- **NOTE:** *If students have already created remix programs for the missions in this unit, you can skip the remix here. If they haven't done any remixes so far, they should do at least one before moving to the next unit. Depending on the time you have to spend in the curriculum, you might find value in having students do a Unit 2 Remix, even if they have done remixes for other missions.*
- You can find two rubrics to use for the Unit 2 Remix: CodeX Project Rubric and/or CodeX Sliding Rubric
- Use the planning guide from one of the Unit Remix assignments.
- End the unit with the Unit 2 Vocabulary Exam and Unit 2 Coding and Concepts Exam

Remix Ideas:

- Pick a remix idea from Mission 6, Mission 7 or Mission 8. Try one you haven't done yet.
- Combine remix ideas from the different missions into your own original program.
- Create a 2-person Personal Billboard. Select person A with a BTN_A and person B with BTN_B. Then use the other buttons to scroll forward and backward or select a random item from the chosen list. Add pixel lights in specific colors for each person. Turn on the LED for the button when the person is selected. Add sounds for more of a challenge.
- Create a "high-low" game. Button B is used to start or restart the game. At the start of the game a random number is displayed. Then the player must press UP or DOWN to guess whether the next number will be higher or lower than the number shown. If the guess is correct, pixel LED 0 is lit GREEN, and another number is selected. This proceeds until all 4 pixels are GREEN or the player guesses wrong and the game ends. Create a nice "winning" or "losing" sequence with the LEDs and display.
- Think of your own creative project. Maybe it will be a game. Maybe it will solve a problem. Maybe it will incorporate science, math or language arts. It is up to you!

Rubric Checklist:

☐ Use a list
☐ Use buttons for input
☐ Use a random function
☐ Use a conditional if statement
☐ Use a loop
☐ Program uses a concept from Mission 6
☐ Program uses a concept from Mission 7
☐ Program uses a concept from Mission 8
☐ Add readability to your program by adding blank lines and comments
☐ Use strategies to debug the code
☐ Program should work correctly and without any errors

Unit 3: Functions and Sensors (9-20 hours)

Students learn about and use functions, an added level of abstraction. The programs also use a variety of inputs, like an internal clock, the accelerometer and the light sensor, and different ways of showing output.

Mission 9 Game Spinner:

Students will use a list and several functions to create a game spinner arrow. Parameters are introduced in the context of programmer-defined functions.

Mission 10 Reaction Tester:

Students learn about and use CodeX's internal clock. More functions from the time module are introduced. Students create a reaction-style game.

Mission 11 Spirit Level:

Students learn about and use the CodeX's accelerometer. They will use a tuple to store data from the accelerometer, and then use it to determine if the CodeX is level or tilted.

Mission 12 Night Light

Students use the built-in light sensor to detect the amount of ambient light. They use the data to make the CodeX into a night light, turning itself on when the environment is dark.

Preparation and Materials:

- Students need a computer / laptop with the Chrome web browser.
- Make sure the students can successfully login to http:/make.firialabs.com
- Students login with their account (possibly using their gmail account)
- Each student (or pair) needs a CodeX and connecting cable.
- Optional: a light source, like a flashlight, for Mission 12.

Assessment:

Mission 9 Review Kahoot	Mission 10 Review Kahoot	Mission 11 Review Kahoot	Mission 12 Review Kahoot
U3 Vocab Review Kahoot	U3 Coding Review Kahoot	U3 Vocab Test (MS Form)	U3 Coding Test (MS Form)

Standards addressed in this unit:

CSTA Standards Grades 6-8	CSTA Standards Grades 9-10	CSTA Standards Grades 11-12
• 2-CS-02 • 2-CS-03 • 2-DA-07 • 2-DA-08 • 2-DA-09 • 2-AP-10 • 2-AP-11 • 2-AP-12 • 2-AP-13 • 2-AP-14 • 2-AP-15 • 2-AP-16 • 2-AP-17 • 2-AP-19	• 3A-CS-01 • 3A-CS-02 • 3A-CS-03 • 3A-DA-09 • 3A-DA-11 • 3A-DA-12 • 3A-AP-13 • 3A-AP-14 • 3A-AP-16 • 3A-AP-17 • 3A-AP-18 • 3A-AP-19 • 3A-AP-21 • 3A-IC-26	• 3B-CS-02 • 3B-DA-05 • 3B-DA-06 • 3B-DA-07 • 3B-AP-10 • 3B-AP-12 • 3B-AP-14 • 3B-AP-15 • 3B-AP-16 • 3B-AP-17 • 3B-AP-21 • 3B-AP-22 • 3B-AP-23

Mission 9: Game Spinner

Time Frame: 60-120 minutes

Project Goal: Students will use a list, functions and a while loop to create an arrow spinning, like for a game.

Learning Targets

- I can apply properties and uses on an index.
- I can define and call a function.
- I can utilize multiple variables.
- I can utilize loops to accomplish a task.
- I can create a simulation using hardware and software.

Key Concepts

- Logical operators “and”, “or” and “not” allow your code to test for multiple conditions.
- Define functions to break up complex code into smaller, easy to use and reuse pieces.
- You can detect instantaneous button presses using *buttons.is_pressed()* inside an infinite loop.
- Loop for a determined number of iterations using a condition (while index < count),
- Add simple physics effects for a simulation, like slowing down a spin with gradual friction.
- Overflow your first array and learn what to do.

Assessment Opportunities

- Quiz after Objective 1
- Quiz after Objective 4
- Daily exit ticket
- Mission 9 Review Kahoot
- Submit final **Game_Spinner** program
- Student Reflection

Success Criteria

- □ Display an Arrow in a random direction
- □ Detect an input- button A or B - to trigger the Arrow spin
- □ Use a loop to animate an arrow spinning around
- □ Define a function for spinning arrows
- □ Use a parameter in the function and an argument in the function call
- □ Define a function that displays one random arrow
- □ Use three variables:
    - □ Variable for the index
    - □ Variable to control the loop
    - □ Variable to vary the speed of the animation

Teacher Resources

- Use the Mission 9 Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, Workbook, and solution

Vocabulary

- **Logical Operator:** Operators that handle combinations of Boolean results; not, and, or
- **Function:** A named chunk of code you can run anytime just by calling its name; also called a procedure
- **Parameter**: A local variable in a function that receives a value passed into the function when it is called; information the function needs to complete its task
- **Argument**: The value passed into a function – information the function needs to complete its task. An argument can be a literal value, a variable, or an expression.
- **Control Variable**: A variable used in a condition that determines when a loop will end; must be incremented or changed inside the loop.

New Python Code

```
if buttons.is_pressed(BTN_A) or
   buttons.is_pressded(BTN_B)
```

Use the logical operator “or”
Either or both can be true for the condition to be true

```
def show_random_arrow():
    num = random.randrange(8)
    display.show(pics.ALL_ARROWS[num])
```

Define a function (no parameter)

`while index < 8:`
`   my_arrow = pics.ALL_ARROWS[index]`
`   display.show(my_arrow)`
`   sleep(0.1)`
`   index = index + 1`
	Finite loop with condition
	(increment the control variable <index>)

`while loops < count:`
`   my_arrow = pics.ALL_ARROWS[index]`
`   display.show(my_arrow)`
`   sleep(delay)`
`   delay = delay + 0.005`
`   loops = loops + 1`
`   index = index + 1`
`   if index == 8:`
`      index = 0`
	Finite loop with condition and list wrapping
	(avoid ‘index out of range’ error)

Real World Applications

- Use the game spinner in a real-world application. After the project, have a group of students gather in a circle to play with what they’ve programmed. Disconnect CodeX and run it on batteries. Put it on the floor or table in the middle of the group, so the random-arrow selects a student each time the button is pressed. Maybe the chosen one gets to name something different they could build using CodeX and the code they’ve learned!
- Fast button inputs, animation and simulation! Those are essential ingredients for lots of interesting applications:
    - Video games
    - Flight simulators
    - Virtual reality

Mission 9 Remix

Time Frame: 1-3 hours

Project Goal: Students will use the skills and concepts they learned through mission 9 to create their own project.

Project Outline: Follow the five-steps of the design process to design a remix project.

Assessment Opportunities

- Peer reviews / Gallery walk
- Remix Planning Guide (5-step process)
- Remix Rubric Checklist
- Submit Remix Program
- Mission 9 Review Kahoot
- Student Reflection

Teacher Resources

- Use the Mission 9 Remix Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, Workbook, suggestions and solutions

Remix Ideas:

- **MILD**: Create a function that will give an introduction and wait to start the game spinner. Then add a button to break out of the infinite loop and end the program with a message.
- **MILD**: Add a beep for each arrow spin.
- **MILD**: Make the animation spin counterclockwise instead of clockwise.
- **MEDIUM**: Use BTN_A for spinning clockwise and BTN_B for spinning counterclockwise.
- **MEDIUM**: Many games use colors, numbers or images. Show a random color (or a large number) on the screen. Get a random number for count and whatever is showing at the end is the selection.
- **SPICY**: Use both BTN_A and BTN_B for two different things. For example, use BTN_A for the arrow spin and BTN_B for the dice roll. Or BTN_A for arrow spin and BTN_B for random color.
- **SPICY**: Create another list of pitches and use a different beep for each arrow.
- **SPICY**: Create a 2-player game. Each player presses their own button. Use the dice roll or assign points to the arrows or colors. The first player to a point value wins.
- **CHALLENGE**: Many games use two dice. Create a program that rolls two dice at the same time.
- **CHALLENGE**: Create a color match game. Display two rectangles, each with a random color.

Cross-Curricular

- **CROSS CURRICULAR:** The remix can be like a Jeopardy game, with categories on the wheel. Use it for review or discussion in any subject.
- **CROSS CURRICULAR:** Spin a wheel of student names. The selected student answers the question, is team captain, etc.
- **MATH:** Spin a wheel of numbers 1 through 10. The number it stops is the answer, and students come up with an expression that equals the answer.
- **MATH:** Use the dice roll or game spinner for a lesson on probability. Then see how partial or impartial the wheel is.
- **LANGUAGE ARTS:** Spin a wheel of words – types of speech, sentence starters, parts of a story, etc. Use the wheel spin for discussion, review, etc.
- **VISUAL ARTS:** Spin a wheel of colors. Use the wheel to discuss color, or use the designated color in a project, etc.
- Supports **language arts** through peer review and reflection writing.

Rubric Checklist:

□ Write an original program, run it, and save it to the CodeX
□ Follow the design process and document their work in the log assignment
□ Add readability to your program by adding blank lines and comments
□ Use at least one list
□ Use at least one button as input to affect the code
□ Use a random function at least once
□ Debug any errors in the code so that the program runs correctly.
□ Complete a review or reflection of their original remix program

Mission 10: Reaction Tester

Time Frame: 45-90 minutes

Project Goal: Students will use CodeX's internal clock to create a reaction tester game.

Learning Targets

- I can define and use a function.
- I can utilize multiple variables.
- I can utilize loops to iterate code.

Key Concepts

- Computers are driven by internal clocks.
- Functions can have named parameters (optional parameters) like *loop=True* and *wait=False*.
- The DRY concept is to never write the same code twice (don't repeat yourself!).

Assessment Opportunities

- Quiz after Objective 6
- Daily exit ticket
- Mission 10 Review Kahoot
- Submit final **Reaction_Time** program
- Student Reflection

Success Criteria

□ Give the player a 3-2-1 countdown.

□ Program a random delay so the player can't "guess" the timing.

□ Measure the time until a button press occurs.

□ Display the reaction time.

□ Wait for a button press, then restart the game.

□ Use functions to reduce repetition and to organize your code.

Teacher Resources

- Use the Mission 10 Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, Workbook, and solution

Vocabulary

- **Computer Clock:** Electronic clock circuits; the heartbeat of the computer. The tick of the clock moves through the code one line at a time. It is also used in the sleep function, scheduled activities within the CPU, and everything timing related on the computer.

New Python Code

`pixels.set([BLACK, BLACK, BLACK, BLACK])`	Turn off all pixels using a list
`pixels.set([RED, RED, GREEN, GREEN])`	Turn on all pixels a color using a list
`display.clear()`	Clear the display
`start_time = time.ticks_ms()`	Get the current clock time
`time.ticks_diff(end_time, start_time)`	Find the difference between two clock times
`buttons.was_pressed(BTN_A)`	Reset the button state

Real World Applications

- Computers measure time in all types of applications.
  - Football play clocks, and stop watches for other sports
  - Electronic drum machines
  - Microwave oven timers
  - Alarm clocks
  - Etc. – how many more can you name?

Mission 10 Remix

Time Frame: 1-3 hours

Project Goal: Students will use the skills and concepts they learned in through Mission 10 to create their own project.

Project Outline: Follow the five-steps of the design process to design a remix project.

Assessment Opportunities

- Peer reviews / Gallery walk
- Remix Planning Guide (5-step process)
- Remix Rubric Checklist
- Submit Remix Program
- Mission 10 Review Kahoot
- Student Reflection

Teacher Resources

- Use the Mission 10 Remix Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, Workbook, suggestions and solutions

Remix Ideas:

- **MILD**: The code inside the loop is very long. Create a function for the countdown code.
- **MILD**: Create a function that will give an introduction and wait to start the game (use the wait() function).
- **MEDIUM**: Create a list of colors and select a random color for the display screen. Use the screen as the indicator instead of pixels.
- **MEDIUM**: Create a list of images and select a random image for the display screen. Use the image as the indicator instead of pixels.
- **SPICY**: Make the reaction tester into a game. Use the average time of your reactions as the deciding value. If your time is better than average, add a point. As an option, also subtract a point for slower times.
- **SPICY**: Use two images: one to indicate BTN_A and a different one to indicate BTN_B. Select a random image and see if the player presses the correct button. If so, add a point to the score.
- **CHALLENGE**: Add functionality to Spicy #1. Choose a way to win or lose the game. For example, run the test 20 times. If the player scores at least 15 points, they win. Another example: the player wins if they get five points, or lose if they get -5 points.
- **CHALLENGE**: Use BTN_L, BTN_R and BTN_U to determine which indicator to use. For example, BTN_L for pixels, BTN_R for images and BTN_U for sounds. BTN_D can be a random selection.

Cross-Curricular

- **SCIENCE:** The mission creates a program that measures reaction time. This could be made into a lab, following the scientific method. Students try different indicators (pixels, colors, images, etc.) and keep track of and plot their times, compare to other students, experiment with ages, etc.
  See the cross-curricular project **Mental Chronometry**
- **MATH:** Students can make a list of their times and the data can be used to calculate average or median. Students can create various charts or graphs with their data.
- **MATH:** Change the delay times in increments, plot the points and then develop the algebraic equation to reaction times (y = mx + b).
- **GAME DESIGN:** Students can take the basic reaction timer and make it into a game. Students must decide how to award points and a way to win or lose.
- Supports **language arts** through peer review and reflection writing.

Rubric Checklist:

- □ Write an original program, run it, and save it to the CodeX
- □ Follow the design process and document your work in the log assignment
- □ Add readability to your program by adding blank lines and comments
- □ Use at least one function
- □ Use at least one button as input to affect the code
- □ Use the computer clock functions: ticks_ms, sleep and ticks_diff
- □ Use a random function at least once
- □ Debug any errors in the code so the program runs correctly
- □ Complete a review or reflection of their original remix program

Mission 11: Spirit Level

Time Frame: 45-90 minutes

Project Goal: Students will use CodeX's accelerometer and a graphical interface to show its tilt.

Learning Targets

- I can use comments to explain and document the purpose of each line of code.
- I can use variables to calculate and convert measurements.

Key Concepts

- Accelerometers are in many digital devices, including your cell phone. Students learn what acceleration is and what it has to do with "horizontal." Read the toolbox entry!
- The sensors on CodeX will not give you data you can use "as is" – it must be converted to something useful. In this program, you will convert the data to degrees.
- You can use a bit of math for scaling the degrees to a range suitable for the graphical 'bubble'.

Assessment Opportunities

- Quiz after Objective 5
- Daily exit ticket
- Mission 11 Review Kahoot
- Submit final **Spirit_Level** program
- Student Reflection

Success Criteria

☐ Display a numeric "tilt" value from the accelerometer.

☐ Scale the raw tilt value to show 0-9, indicating 0° to 90° incline.

☐ Replace the number display with a graphical bubble simulation!

☐ Erase the graphical bubble before drawing a new bubble.

Teacher Resources

- Use the Mission 11 Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, Workbook, and solution

Vocabulary

- **Accelerometer:** A sensor chip that detects motion, impacts, and orientation; a device that measures proper acceleration.
- **Tuple:** A read-only version of a list, indicated with parenthesis, and has items you can access with an *index*.

New Python Code

`import math`	Import the math module for math operations like math.pi
`val = accel.read()`	Get the values from the accelerometer (is a tuple)

`val = accel.read()`
`tilt_x = val[0]`
	Get a single value from the accelerometer

`display.fill(WHITE)`	Change the display color

`display.draw_line(x1, y1, x2, y2, color)`
`display.draw_line(CENTER, 0, CENTER, 105, BLACK)`
	Draw a line

Real World Applications

- Let students have time to play with the spirit level. If they use batteries, they can measure the levelness of various items in the room. Discuss how a physical level works. Accelerometers are used as tilt sensors for:
  - Controlling your phone screen (landscape or portrait)
  - Building a house
  - Keeping solar panels pointed at the sun, and much more!

Mission 11 Remix	Time Frame: 1-3 hours
Project Goal: Students will use the skills and concepts they learned through Mission 11 to create their own project. **Project Outline:** Follow the five-steps of the design process to design a remix project.	**Assessment Opportunities** ● Peer reviews / Gallery walk ● Remix Planning Guide (5-step process) ● Remix Rubric Checklist ● Submit Remix Program ● Mission 11 Review Kahoot ● Student Reflection

Teacher Resources

- Use the Mission 11 Remix Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, Workbook, suggestions and solutions

New Python Code

`display.fill_circle(CENTER, CENTER, 15, RED)`	Filled in circle
`display.draw_text(str(score),x=20,y=20,scale=3,color=BLACK)`	Display text at a specific place

Remix Ideas:

- **MILD**: Start with the original code. Create at least one function and call it in the program. The function could set up the initial screen and lines, or it could scale the raw data to degrees.
- **MILD**: Add an introduction and wait to start the program. Add a way to exit the program and include an ending message. These could be functions!
- **MILD**: Change the horizontal bubble to a vertical bubble (use y instead of x).
- **MEDIUM**: Change the circle to a filled-in circle when CodeX is level. Include an intro and ending.
- **MEDIUM**: Change the spirit level to check both horizontal and vertical positions (use x and y).
- **SPICY**: Make the spirit level into a game. Award a point for going close to the edge but not touching. Get four points to win the game. Touch an edge and the game ends.
- **SPICY**: Change the spirit level to a spirograph. Instead of erasing the circle, let it continue to draw on the screen. Choose a random color after every 20 circles.

Cross-Curricular

- **MATH:** Use the level to measure angles.
- **SCIENCE:** Physical science -- discuss how a real mechanical spirit level works. The 'spirit' is a liquid with space for a bubble, which will be in the center of the tube when it is in its horizontal position.
- **SCIENCE:** Physical science -- discuss gravity and how it is measured. Use the information from z to measure and study gravity. See the cross-curricular project **Gravity**
- **GAME DESIGN:** Students can take the basic spirit level and make it into a game. Students must decide how to award points and a way to win or lose.
- **FINE ARTS:** Students can make the spirit level into a spirograph and use it to "draw" on the screen. Use of color and color theory, as well as other topics, can be included.
- **LANGUAGE ARTS:** Have students write a short description of the spirit level for an online shopping store.
- Supports **language arts** through peer review and reflection writing.

Rubric Checklist:

□ Follow the design process and document your work in the log assignment
□ Use readings from the accelerometer
□ Use the graphics function of drawing shapes, like a circle and lines
□ Add functionality or something different to the original spirit level mission
□ Debug any errors in the code so that the program works correctly
□ Complete a review or reflection of their original remix program

Mission 12: Night Light

Time Frame: 45-90 minutes

Project Goal: Students will use CodeX's built-in light sensor to become a night light and turn on when the environment is dark.

Learning Targets

- I can apply I/O values to complete a task.
- I can explain the difference between analog and digital I/O.
- I can explain the difference between reading and writing input and output.
- I can use variables to calculate and convert measurements.

Key Concepts

- The photocell helps convert a light level into an electrical voltage level.
- Analog means infinite variation from dark to light, cold to hot, and so on. The CodeX's analog to digital converter (ADC) gives a digital approximation of the photocell's analog reading.
- You can create your own 'Image' object, using the fill() function, to set all pixels to a given brightness level.

Assessment Opportunities

- Quiz after Objective 4
- Daily exit ticket
- Mission 12 Review Kahoot
- Submit final **NightLight** program
- Student Reflection

Success Criteria

- □ The CodeX determines when it is dark and turns on the pixels.
- □ The brightness adjusts with the amount of darkness.

Teacher Resources

- Use the Mission 12 Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, Workbook, and solution

Vocabulary

- **Light sensor:** A sensitive electronic device that measures the amount of light falling on it

Review from Mission 5:

- **Analog:** Infinite variation in something, like hot to cold or light to dark; smooth and continuous signals that represent a quantity, like sound waves
- **Digital:** A numerical representation of an analog signal, represented in increments
- **ADC**: analog to digital conversion

New Python Code

`value = light.read()`	Read data from the light sensor

`pixels.fill(WHITE) – on!`
`pixels.fill(BLACK) – off!`
	Set all pixels the same color

`pixels.fill(WHITE, brightness=20)`
`pixels.fill(RED,brightness=level)`
	Adjust the brightness of pixels

Real World Applications

- This project introduces an area with lots of potential for improving the world! Light sensors and LCD lights controlled with code can reduce energy consumed and make lighting more awesome! Some real-world applications are:
  - Outdoor lighting – street lights, parking lots, home lighting, etc.
  - Stadium lights – even controlling the light color so it looks better on camera
  - Indoor lighting – sensing daylight from windows and skylights is daylight harvesting, saving energy

Mission 12 Remix

Time Frame: 1-3 hours

Project Goal: Students will use the skills and concepts they learned through Mission 12 to create their own project.

Project Outline: Follow the five-steps of the design process to design a remix project.

Assessment Opportunities

- Peer reviews / Gallery walk
- Remix Planning Guide (5-step process)
- Remix Rubric Checklist
- Submit Remix Program
- Mission 12 Review Kahoot
- Student Reflection

Teacher Resources

- Use the Mission 12 Remix Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, Workbook, suggestions and solutions

Remix Ideas:

- **MILD**: Add another level variable of light intensity to the program – it could be for bright light or very dark light. This will give CodeX three options for the light sensor reading.
- **MEDIUM**: Change CodeX to a people counter. When the light sensor changes from room light to dark, increment a counter. Wait a little before checking again, so the person isn't counted twice.
- **MEDIUM**: Change CodeX to an alarm. When room light is detected instead of dark, sound the alarm! Otherwise, display a peaceful image.
- **SPICY**: Add more level variables for CodeX, with each variation of light intensity displaying a different picture and pixel color. Try five levels.
- **SPICY**: Use CodeX's internal clock as a timer. Use the timer to turn on the light, and turn it off.

Cross-Curricular

- **MATH:** Review the math for scaling the digital reading to a brightness, and the need to reverse the number.
- **SCIENCE:** Use the light sensor in an experiment, following the scientific method. An example could be light feedback. Place the photocell near the LCD so that when the display is ON, it shines right at the photocell. Make sure the room is relatively dark. Do you get a flicker? This is the optical version of audio feedback (when speakers squeal because the mic is too close). This also shows how fast the CodeX is sampling the ADC and controlling the LCDs.
- **LANGUAGE ARTS:** Use CodeX as a selection tool. The amount of light can display different poems, or words as sentence starters, colors for mood, etc.
- **LANGUAGE ARTS:** Have students write a lab report about this mission.
- **FINE ARTS:** Students display different colors or images depending on the amount of light it receives.
- Supports **language arts** through peer review and reflection writing.

Rubric Checklist:

□ Write an original program, run it, and save it to the CodeX
□ Follow the design process and document your work in the log assignment
□ Add readability to your program by adding blank lines and comments
□ Use readings from the light sensor
□ Use the readings as input and if statements in the code to determine what CodeX does
□ Add functionality or something different to the original night light mission
□ Debug any errors in the code so that the program works correctly
□ Complete a review or reflection of their original remix program

Unit 3 Assessments

Time Frame: 2-5 hours

Project Goal: Assess students for mastering Unit 3 Vocabulary, Coding and Concepts.

Learning Targets

- I can create my own original program using concepts from Missions 9-12.
- I can define vocabulary words from Missions 9-12.
- I can answer multiple choice questions that test coding and concepts from Missions 9-12.

Assessment Opportunities

- Remix Planning Guide (5-step process)
- Remix Rubric Checklist
- Remix Project Rubric
- Submit Remix Program
- Student Reflection
- Unit 3 Vocabulary Exam
- Unit 3 Coding and Concepts Exam

Teacher Resources

- Kahoot Reviews: Unit 3 Vocabulary / Unit 3 Coding and Concepts
- For a Unit 3 Remix, use the Mission 12 Remix Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- **NOTE:** *If students have already created remix programs for the missions in this unit, you can skip the remix here. If they haven't done any remixes so far, they should do at least one before moving to the next unit. Depending on the time you have to spend in the curriculum, you might find value in having students do a Unit 3 Remix, even if they have done remixes for other missions.*
- You can find two rubrics to use for the Unit 3 Remix: CodeX Project Rubric and/or CodeX Sliding Rubric
- Use the planning guide from one of the Unit Remix assignments.
- End the unit with the Unit 3 Vocabulary Exam and Unit 3 Coding and Concepts Exam

Remix Ideas:

- Pick a remix idea from Mission 9, Mission 10, Mission 11 or Mission 12. Try one you haven't done yet.
- Combine remix ideas from the different missions into your own original program.
- Create a dice roller, with a graphical representation for the pips using circles. Add a more realistic roll by adding "rolling" three times, and maybe even a sound effect before the die roll appears. After a delay, clear the screen so it is ready for the next roll.
- Think of your own creative project. Maybe it will be a game. Maybe it will solve a problem. Maybe it will incorporate science, math or language arts. Use the sensors and create functions throughout. What you do is up to you!

Rubric Checklist:

□ Use a list
□ Use buttons for input
□ Use a random function
□ Use a conditional if statement
□ Use a loop
□ Use the CodeX internal clock or a sensor
□ Add readability to your program by adding blank lines and comments
□ Use strategies to debug the code
□ Program should work correctly and without any errors

Unit 4: Graphics and Sounds (6-15 hours)

Students learn about creating graphics on CodeX. First, students learn more about the draw features and create a graphical user interface. Then they use the draw features to create line art. These projects use a bit of math, but do not let that keep you from completing these engaging programs.

Mission 13 Sounds Fun:

In this mission students will create a user-friendly graphical interface and explore the 'soundlib' library for CodeX sound effects. This library will allow the code to continue executing while sound is being played. This means sounds and music can play in the background while the code is running. The CodeX will become a controller for a race.

Mission 14 Line Art:

Students will create beautiful visual string art and learn about computer graphics while writing only a few lines of code and the power of the pixel.

Preparation and Materials:

- Students need a computer / laptop with the Chrome web browser.
- Make sure the students can successfully login to http://make.firialabs.com
- Students login with their account (possibly using their gmail account)
- Each student (or pair) needs a CodeX and connecting cable.

Assessment:

Mission 13 Obj 1-6 Review	Mission 13 Obj 7-11 Review	Mission 14 Obj 1-5 Review	Mission 14 Obj 6-9 Review
U4 Vocab Review Kahoot	U4 Coding Review Kahoot	U4 Vocab Test (MS Form)	U4 Coding Test (MS Form)

Standards addressed in this unit:

CSTA Standards Grades 6-8	CSTA Standards Grades 9-10	CSTA Standards Grades 11-12
• 2-CS-01 • 2-CS-03 • 2-DA-08 • 2-AP-10 • 2-AP-11 • 2-AP-12 • 2-AP-13 • 2-AP-14 • 2-AP-15 • 2-AP-16 • 2-AP-17 • 2-AP-19	• 3A-CS-03 • 3A-DA-09 • 3A-DA-11 • 3A-DA-12 • 3A-AP-13 • 3A-AP-14 • 3A-AP-15 • 3A-AP-16 • 3A-AP-17 • 3A-AP-18 • 3A-AP-19 • 3A-AP-21 • 3A-AP-23 • 3A-IC-26	• 3B-CS-02 • 3B-AP-10 • 3B-AP-11 • 3B-AP-12 • 3B-AP-14 • 3B-AP-15 • 3B-AP-16 • 3B-AP-17 • 3B-AP-21 • 3B-AP-22 • 3B-AP-23

Mission 13: Sounds Fun

Time Frame: 60-120 minutes

Project Goal: Students will create a user-friendly graphical interface and use audio functions from the '**soundlib**' library to create a controller for a race that can play music in the background.

Learning Targets

- I can explain what a GUI is used for in a computer application.
- I can explain the difference between a local and global variable.
- I can use a '**for**' loop to iterate over a list.
- I can distinguish between blocking and non-blocking functions.
- I can use the '**not**' operator to solve a problem.

Key Concepts

- Variables defined inside a function are local variables; variables defined outside a function are global variables.
- The '**soundlib**' Python module has functions to create different types of tones, as well as playing recorded samples and mp3s.
- The '**for**' loop is made for iterating across a range of numbers, or iterating over lists.
- Blocking functions stop your code from continuing until they finish.
- Non-blocking functions allow code to continue to run without waiting for the sound to finish.
- Use the '**not**' logical operator to flip the state of a global Boolean variable.

Assessment Opportunities

- Quiz after Objective 3
- Quiz after Objective 8
- Quiz after Objective 9
- Many opportunities throughout the lesson for practice and to demonstrate learning: global and local variables, 'for' loop, 'not' operator, etc.
- Daily exit ticket
- Mission 13 Obj. 1-6 Review Kahoot
- Mission 13 Obj. 7-11 Review Kahoot
- Submit final **Race_Control** program
- Student Reflection

Success Criteria

- □ Display a GUI with four menu items
- □ Use a rectangle to show which menu item is selected
- □ Scroll the selection rectangle up and down to the different menu items
- □ Select a menu item using BTN A
- □ Play music in the background
- □ Play music for starting the race
- □ Play music for finishing the race
- □ Play music as a warning sound

Teacher Resources

- Use the Mission 13 Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, Workbook, and solution

Vocabulary

- **User Interface:** The area where a person interacts with a physical device, often through a screen.
- **Bitmap:** Graphics bits – drawing images and text. A bitmap is an object that can hold a 2D image of a given width and height; a list of pixel RGB values.
- **Local Variable**: A variable that is "private" to a function. It only exists while the function is running, and is separate from any other variable outside the function.
- **Global Variable**: Variables defined outside of a function. They exist the entire life of the program and can be accessed and used inside a function.
- **Initialization**: Set the initial or first value of a global variable when the program starts. Also, set the screen to its beginning look.
- **Soundlib module**: Functions for creating music and sound effects, including different types of tones.
- **For Loop**: Looping across a range of numbers, or iterating over a list.
- **Blocking Function**: Functions that block your code from continuing until they are finished. The code has to wait while a song plays, for example.
- **Non-blocking Function**: A function that doesn't make your code wait for the function to finish. For example, other lines of code can execute while a song is playing.
- **Toggle**: Flip the state of a variable (True to False or False to True) that is used to either do or not do something.
- **Nested For Loop**: A for loop with a for loop inside, or nested.

New Python Code

display.draw_rect(0, 80, 240, 40, GRAY)
display.draw_rect(0, menu_y[prev_sel], 240, 40, BLACK)
	Draw a rectangle – first one is an outline
	Second example is a filled-in rectangle

display.draw_text(‘MUSIC’,x=20,y=90,color=WHITE,scale=3)	Draw text (different from display.print()) Parameters are optional and can be listed in any order

max(menu_index-1, 0)
menu_index = min(menu_index+1, 3)
	Max and min functions – returns the largest or smallest item from the choices in the parenthesis. Usually part of an assignment statement.

from soundlib import *	Import the soundlib module
trumpet = soundmaker.get_tone(‘trumpet’)	Get a tone from the soundmaker library.

siren.set_pitch(440)
siren.play()
	Set the pitch of a tone and “turn on” the note playing.

sleep(1.5)
siren.stop()
	Stop playing the note or music file.

trumpet.set_pitch(440)
for i in range(4):
   trumpet.play()
   sleep(0.1)
   trumpet.stop()
   sleep(0.1)
	Example of a ‘for’ loop.
	The ‘i’ is the loop control variable, incrementing from 0 to 3.

global is_playing is_playing = not is_playing	‘Not’ operator – used to toggle. Flips the state of a variable, like from True to False

race_music=soundmaker.get_mp3(‘sounds/funk.mp3’)
race_music=soundmaker.get_mp3(‘sounds/funk.mp3’, play=False)
	Non-blocking function for playing an mp3. Add a parameter so the music does not start automatically.

trumpet.play()
for freq1 in range(500, 1500, 100):
   for freq2 in range(freq1, freq1+1000, 100):
      trumpet.set_pitch(freq2)
      sleep(0.023)
	Use a ‘for’ loop while playing a sound. This example uses a ‘nested’ for loop. The inner loop takes the current frequency and increases it for a sweeping sound. This is repeated 10 times (outer loop), changing the first frequency by 100 each time.

siren.set_pitch(440)
siren.play()
siren.glide(880, 1.5)
	Glide from ‘soundlib’. Glide takes two arguments (new, or ending, pitch and duration). It is non-blocking.

Real World Applications

- You can now make a GUI on CodeX! Using a GUI enables a person to communicate with a computer through a simple interface. Think about all the ways CodeX can be used as a controller.
- A GUI is an example of abstraction. The details of drawing the GUI are hidden from the user and can be changed without the user. A simplified interface is presented to the user to control the device.

Mission 13 Remix	**Time Frame:** 1-3 hours
Project Goal: Students will use the skills and concepts they learned through Mission 13 to create their own project. **Project Outline:** Follow the five-steps of the design process to design a remix project.	**Assessment Opportunities** ● Peer reviews / Gallery walk ● Remix Planning Guide (5-step process) ● Remix Rubric Checklist ● Submit Remix Program ● Mission 13 Obj 1-6 Review Kahoot ● Mission 13 Obj 7-11 Review Kahoot ● Student Reflection

Teacher Resources

- Use the Mission 13 Remix Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, Workbook, and suggestions.

Remix Ideas:

- **MILD**: Add an introduction and/or ending message, including a border, to an earlier program.
- **MILD**: Add non-blocking music and sounds to a program from an earlier mission.
- **MEDIUM**: Create your own musical instruments. Use a GUI to represent different instruments or controls. Use the buttons to play music or sound effects.
- **MEDIUM**: Using a program from an earlier mission, add a toggle for a two-player experience. For example, two different people could have their own billboard. Or make the reaction tester into a 2-player game.
- **SPICY**: Revisit the dice roller remix program. Instead of numbers or images, use circles to draw pips for each die number. Add a GUI to let the user choose between a die rolling or an arrow spinning.
- **SPICY**: Using a program from an earlier mission, create a graphical user interface for the program.

Cross-Curricular

- **MATH:** A for loop can increment the loop control variable by a value other than 1. This is like skip-counting. Have students count by different amounts, like 5s, 10s, etc. and then write a for loop that does the same thing.
- **MATH:** Multiplication, or finding area, is similar to nested for loops with addition. Show the math behind multiplication by using nested for loops.
- **SCIENCE:** This mission discusses sounds by using a frequency. Have a lesson about sound and pitch or frequency. Try out the concepts on the CodeX in a program.
- **FINE ARTS:** Have students use a grid to design a user-friendly graphical interface for an app prototype.
- **LANGUAGE ARTS:** Have students write about how a for loop (and nested for loops) work. Or students can write a compare and contrast essay on blocking and non-blocking functions.
- Supports **language arts** through peer review and reflection writing.

Rubric Checklist:

- □ Write an original program, run it, and save it to the CodeX
- □ Follow the design process and document your work in the log assignment
- □ Add readability to your program by adding blank lines and comments
- □ Use at least one concept from Mission 13:
    - □ Non-blocking sound
    - □ Rectangles, circles, lines or draw_text
    - □ For loop
    - □ Nested for loop
- □ Debug any errors in the code so that the program works correctly
- □ Complete a review or reflection of their original remix program

Mission 14: Line Art	**Time Frame:** 60-120 minutes

Project Goal: Students will use computer graphics to create beautiful string art.

Learning Targets

- I can use pixels to draw a line on my LCD screen.
- I can use the bitmap function to get display width and height.
- I can explain how to convert a number from a float to an integer.
- I can use a loop to create a pixel grid on the LCD.
- I can draw an envelope (webbing) using the **display.draw_line()** function.

Key Concepts

- Display is a bitmap, and every bitmap has a width and height.
- Using built-in line drawing functions are faster, simpler and support drawing diagonal lines.
- One way to optimize code is to replace many lines of code that draw lines with a function instead.
- Creating string art is easy by using an envelope, a loop and a constant.
- Defining a custom function with line points that spin counterclockwise makes a web pattern.

Assessment Opportunities

- Quiz after Objective 1
- Quiz after Objective 5
- Daily exit ticket
- Mission 14 Obj. 1-5 Review Kahoot
- Mission 14 Obj. 6-9 Review Kahoot
- Submit final **LineArt** program
- Student Reflection

Success Criteria

☐ Draw an x- and y- axis on the display screen
☐ Draw a white grid of dots on the display screen
☐ Draw a blue square as a border around the screen
☐ Use built-in screen information for drawing lines (display.width and display.height)
☐ Create a function for drawing a web
☐ Call the web function at least 6 time

Teacher Resources

- Use the Mission 14 Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, and Workbook.

Vocabulary

- **Bitmap:** Graphics bits – drawing images and text. A bitmap is an object that can hold a 2D image of a given width and height; a list of pixel RGB values.
- **Pixel**: Elements of a picture, short for "picture element." They are the tiny dots that make up larger images.
- **Magic Number**: Numbers that just appear in code with no explanation. When something changes in the future, the number doesn't work anymore and you have to change it.
- **Literal**: A specific value, like 120 or 239; also known as a magic number.
- **Envelope:** In geometry, a curve created by straight lines moving down and across a grid.

New Python Code

`display.set_pixel(50, 120, WHITE)`	Turn on a single pixel in color
`display.get_pixel(120, 120)`	Returns a tuple for the color at the given pixel location.
`display.width / display.height`	Bitmap functions that return the display width and height.

`x_center=int(display.width/2)`
`y_center=int(display.height/2)`
	Convert a float value to an integer
	Use the int() function

`for x in range(display.width):`
`   display.set_pixel(x, y_center, RED)`
`for y in range(display.height):`
`   display.set_pixel(x_center, y, RED)`
	'For' loop that draws a straight line of pixels.
	First example draws a horizontal line
	Second example draws a vertical line

`y=20`
`for x in range(0, display.width, 10):`
`   display.set_pixel(x_center, y, RED)`
	Use a 'step' parameter in a 'for' loop. The step is like skip-counting. The loop control variable increments by the step amount each time it loops.

`# Draw a grid of white pixels`
`GRID = 10`
`for y in range(0, display.height, GRID):`
`   for x in range(0, display.width, GRID):`
`      display.set_pixel(x, y, WHITE)`
	Use a nested 'for' loop to draw a grid.
	GRID is the constant for the 'step' of the 'for' loop.

Real World Applications

- This lesson has skills that are important for graphic designers. Learning bitmap functions and mastering how to draw an envelope provide valuable skills for professionals in such fields as graphic design, game development, computer vision, and more! These skills are foundational for creating visually appealing and interactive digital content.
  - Bitmap functions are essential for applications like Adobe Photoshop or GIMP.
  - In game development, bitmap functions are crucial for rendering sprites, background, and visual elements.
  - In UI, drawing envelopes can be used for creating custom shapes, text effects, or intricate patterns.

Mission 14 Remix	**Time Frame:** 1-3 hours
Project Goal: Students will use the skills and concepts they learned through Mission 13 to create their own project. **Project Outline:** Follow the five-steps of the design process to design a remix project.	**Assessment Opportunities** ● Peer reviews / Gallery walk ● Remix Planning Guide (5-step process) ● Remix Rubric Checklist ● Submit Remix Program ● Mission 14 Obj. 1-5 Review Kahoot ● Mission 14 Obj. 6-9 Review Kahoot ● Student Reflection

Teacher Resources

- Use the Mission 14 Remix Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include student assignment Log, Slides, Workbook, and suggestions.

Remix Ideas:

- **MILD**: Use functions that create different webs using the draw_web() function you created. Then press buttons to display different webs.
- **MILD**: Use the CodeX graph paper to design an original small picture using circles, rectangles and lines. Use the “CodeX and Drawing Images” slides for additional help.
- **MEDIUM**: Create your own ASCII art (bitmap image). Use the “CodeX and ASCII Art” slides or “CodeX and ASCII Art” PDF instructions for help. Use buttons to display a bitmap image or web design.
- **MEDIUM**: Design your own image using circles, rectangles and lines (see Mild #2). Then add x and y variables so the image can be displayed in a random position on the screen. Use the “Random Original Graphic” slides for help.
- **SPICY**: Create a set of predefined envelopes like a wave, a bulge or a twist that a user can choose from. Show a menu of options and use button presses (or a GUI) to display the chosen envelope.
- **SPICY**: Use your original graphic (See Mild #2 and Medium #2) and add ‘for’ loops to draw rows, columns and a grid of the original graphic. Use the “Loops with Original Graphic” slides for help.

Cross-Curricular

- **MATH:** This mission uses some very advanced math. If your students are in an upper math class, you can do more with the math in the lesson, specifically deltas.
- **ART:** Have students design and create their own ASCII Art (Medium 2A). (**Digital Art with Codex**)
- **ART:** Have students create their own image (Mild 1B, Medium 2B, Spicy 3B). (**Digital Art with Codex**)
- **ART:** Have students design and create their own physical string art. It doesn’t have to follow the same rules, but it could.
- **LANGUAGE ARTS:** Have students write step by step instructions on how to create computer string art.
- **LANGUAGE ARTS:** Have students write a review of another student’s project.
- Supports **language arts** through peer review and reflection writing.

Rubric Checklist:

- □ Create an original work of art using the CodeX
- □ Follow the design process and document your work in the log assignment
- □ Add readability to your program by adding blank lines and comments
- □ Use at least one concept from Mission 14:
    - □ Setting pixels on the LCD
    - □ Using ‘display.width’ and ‘display.height’ functions
    - □ Convert a number to an integer
    - □ Use a ‘for’ loop, change the step of a ‘for’ loop control variable
    - □ Use a nested ‘for’ loop
- □ Debug any errors in the code so that the program works correctly
- □ Complete a review or reflection of their original remix program

Unit 4 Assessments

Time Frame: 2-5 hours

Project Goal: Assess students for mastering Unit 4 Vocabulary, Coding and Concepts.

Learning Targets

- I can create my own original program using concepts from Missions 13-14.
- I can define vocab words from Missions 13-14.
- I can answer multiple choice questions that test coding and concepts from Missions 13-14.

Assessment Opportunities

- Remix Planning Guide (5-step process)
- Remix Rubric Checklist
- Remix Project Rubric
- Submit Remix Program
- Student Reflection
- Unit 4 Vocabulary Exam
- Unit 4 Coding and Concepts Exam

Teacher Resources

- Kahoot Reviews: Unit 4 Vocabulary / Unit 4 Coding and Concepts
- For a Unit 4 Remix, use the Mission 13 Remix Lesson Prep or Mission 14 Remix Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- **NOTE:** *If students have already created remix programs for the missions in this unit, you can skip the remix here. If they haven't done any remixes so far, they should do at least one before moving to the next unit. Depending on the time you have to spend in the curriculum, you might find value in having students do a Unit 4 Remix, even if they have done remixes for other missions.*
- You can find two rubrics to use for the Unit 4 Remix: CodeX Project Rubric and/or CodeX Sliding Rubric
- Use the planning guide from one of the Unit Remix assignments.
- End the unit with the Unit 4 Vocabulary Exam and Unit 4 Coding and Concepts Exam

Remix Ideas:

- Pick a remix idea from Mission 13 or Mission 14. Try one you haven't done yet.
- Combine remix ideas from the different missions into your own original program.
- Create your own art (ASCII Art – bitmaps – or line drawings) using the teacher resources in the Cross Curricular options for Art. Digital Art with CodeX
- Use a GUI to give options of different original art. Add the ability to play music in the background.
- Think of your own creative project. Maybe it will be a game. Maybe it will solve a problem. Maybe it will incorporate science, math or language arts. What you do is up to you!

Rubric Checklist:

□ Use a list
□ Use buttons for input
□ Use a random function
□ Use a conditional if statement
□ Use a for loop
□ Create an original graphic (any type)
□ Add non-blocking music or sound effects
□ Add readability to your program by adding blank lines and comments
□ Use strategies to debug the code
□ Program should work correctly and without any errors

Unit 5: Python Applications (7-14 hours)

Students put all their learning and skills together to make a vintage arcade game.

Mission 15 Handball:

In this mission students use math and physics to move and bounce a ball. Gaming concepts of keeping score and tracking lives are utilized. Students define and use several functions.

Mission 16 Breakout:

Students continue the game project from Mission 15 (they do not start a new file). They use computer graphics to add bricks and lists to keep track of what brick they are hitting and if it has already been hit. Students add to the score and lives abilities for a completing functioning game.

Final Project:

Students create their own original program that incorporates their skills and applies programming concepts. This is an opportunity for a group project and discussion on the impact of computer technology.

Preparation and Materials:

- Students need a computer / laptop with the Chrome web browser.
- Make sure the students can successfully login to http://make.firialabs.com
- Students login with their account (possibly using their gmail account)
- Each student (or pair) needs a CodeX and connecting cable.

Assessment:

U5 Vocab Review Kahoot	Daily Reflection for Final Project	
U5 Coding Review Kahoot	U5 Vocab Test (MS Form)	U5 Coding Test (MS Form)

Standards addressed in this unit:

CSTA Standards Grades 6-8	CSTA Standards Grades 9-10	CSTA Standards Grades 11-12
2-CS-01	3A-CS-03	3B-CS-02
3-CS-02	3A-DA-09	3B-AP-10
2-CS-03	3A-AP-13	3B-AP-11
2-DA-09	3A-AP-14	3B-AP-12
2-AP-10	3A-AP-15	3B-AP-14
2-AP-11	3A-AP-16	3B-AP-15
2-AP-12	3A-AP-17	3B-AP-16
2-AP-13	3A-AP-18	3B-AP-17
2-AP-14	3A-AP-19	3B-AP-20
2-AP-15	3A-AP-21	3B-AP-21
2-AP-16	3A-AP-22	3B-AP-22
2-AP-17	3A-AP-23	3B-AP-23
2-AP-18	3A-IC-24	3B-AP-25
2-AP-19	3A-IC-25	3B-AP-26
2-IC-20	3A-IC-26	3B-AP-27
2-IC-21	3A-IC-27	

Mission 15: Handball

Time Frame: 60-120 minutes

Project Goal: Students will apply Python programming concepts and their skills to create a single player game of handball.

Learning Targets

- I can program a basic physics game engine.
- I can use multiple functions to draw the ball, update position, clear the screen, etc.
- I can use angles to control the ball rebounds.
- I can use ***delta time*** instead of *sleep()* to manage a fast-paced game.
- I can manage the game state by initializing the game, updating the game state, and handling game over.

Key Concepts

- Use functions to organize your code into reusable blocks: paddle position, collision check, updating score, etc.
- Use constants for variables that don't change.
- Define and initialize variables as local or global.
- Focus and refine your game's user experience.
- Implement collision detection algorithms to detect collisions between the ball and other game objects, such as the paddle or the walls.

Assessment Opportunities

- Quiz after Objective 1
- Quiz after Objective 3
- Quiz after Objective 7
- Quiz after Objective 11
- Daily exit tickets
- Submit final **Handball** program
- Student Reflections

Success Criteria

☐ Define and call 12 functions

☐ Draw a screen layout for the game

☐ Use CodeX buttons as event handlers for the game

☐ Keep track of score and lives

☐ Create 2 animated objects in the game: paddle and ball

☐ Game works correctly, showing a ball bounce, paddle movement, and score

Teacher Resources

- Use the Mission 15 Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include a student assignment, lesson plan and code solutions.

Vocabulary

- **Physics engine:** A device that uses the mechanics of velocity, distance and time
- **Initialization (review)**: Set the initial or first value of a global variable when the program starts
- **Delta time**: Elapsed time in milliseconds (or change in time)
- **UX**: User experience

New Python Code

`else:`
`    continue`
 	
`if n_lives == 0:`
`    continue`
	Continue – jumps back to the top of the while loop instead of going to the next sequential step.
	 – it can only be used inside a loop
	 – the second example skips the rest of the game loop if no lives are left

`tone=soundmaker.get_tone(‘trumpet’)`
`tone.set_leve(15)`
	Adjust the volume of a sound effect

Real World Applications

- You are creating a very basic physics engine, modeling the mechanics of velocity, distance and time.
- You are using trigonometric functions sine, cosine and tangent to calculate ball angles.
- This game provides hands-on experience in creating a simple yet engaging game, allowing learners to understand game mechanics, collision detection, user input handling and basic game design.

Mission 16: Breakout

Time Frame: 60-120 minutes

Project Goal: Students will continue the Handball program from Mission 15 by adding graphical features and lists to create a “Breakout” video game.

Learning Targets

- I can define prototype functions.
- I can detect a collision on a brick using the previous position in the matrix.
- I can create a tuple to hold the official score value for each row of bricks.
- I can review all the code in the program and locate the “secret word.”
- I can add a “mute” feature to toggle the sound on and off.

Key Concepts

- Define a function to draw bricks. Use a list for the brick positions.
- Create a matrix using a list of lists with 8 rows and 10 columns of True to indicate if a brick has been collided with.
- Identify which side of the brick was hit to initiate the correct rebound.
- Use variables to hold the previous position and check them against the new position to make the ball rebound.

Assessment Opportunities

- Quiz after Objective 2
- Quiz after Objective 4
- Daily exit tickets
- Submit final **Breakout** program
- Student Reflections
- Optional: Kahoot unit reviews

Success Criteria

□ Add 8 rows of bricks to the handball program using a matrix
□ Detect if the ball collides with a brick
□ Delete a brick when the ball collides with it
□ Bounce the ball when it collides with a brick
□ Add a score and lives to the game
□ Add a mute button
□ Add a “level-up” feature
□ Game works correctly and without errors

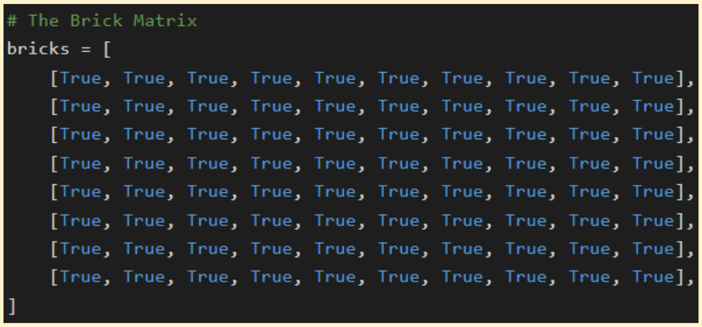
Teacher Resources

- Use the Mission 16 Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Teacher resources include a student assignment, lesson plan and code solutions.
-

Vocabulary

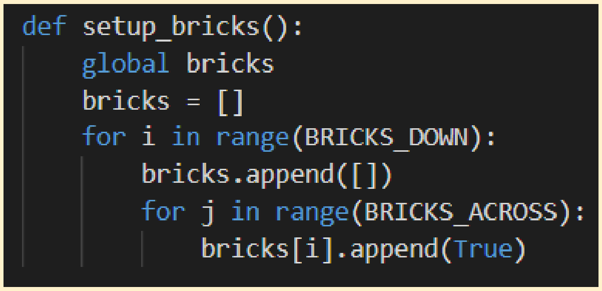
- **Prototype:** A model of something, or an early sample created to test a concept
- **Matrix**: a structure with rows and columns – a 2D array

New Python Code



```
# The Brick Matrix
bricks = [
    [True, True, True, True, True, True, True, True, True, True],
    [True, True, True, True, True, True, True, True, True, True],
    [True, True, True, True, True, True, True, True, True, True],
    [True, True, True, True, True, True, True, True, True, True],
    [True, True, True, True, True, True, True, True, True, True],
    [True, True, True, True, True, True, True, True, True, True],
    [True, True, True, True, True, True, True, True, True, True],
    [True, True, True, True, True, True, True, True, True, True],
]
```

Matrix – a list of lists

<table>
<tr>
<td>

<pre>
def setup_bricks():
    global bricks
    bricks = []
    for i in range(BRICKS_DOWN):
        bricks.append([])
        for j in range(BRICKS_ACROSS):
            bricks[i].append(True)
</pre>
</td>
<td>Code for creating a matrix.
‘i’ is the rows, ‘j’ is the columns</td>
</tr>
<tr>
<td><code>mute = not mute</code></td>
<td>‘Not’ operator (review from Mission 13)
Toggle the Boolean variable</td>
</tr>
<tr>
<td><code>leds.set(LED_A, mute)</code></td>
<td>Turn on the red LED above the BTN_A (review from mission 7).
‘mute’ is a Boolean (True or False)</td>
</tr>
</table>

Real World Applications

- Developing the Breakout game requires students to break down complex problems into smaller, manageable tasks, design algorithms to solve those tasks efficiently, and debug their code to fix errors. These problem-solving skills are applicable to various real-world scenarios, such as software engineering, scientific research, and business analysis.
- Coding the Breakout game provides students with hands-on experience in programming fundamentals such as variables, loops, conditionals, functions and event handling. These skills form the building blocks of software development and can be applied to create a wide range of applications beyond games, including web development, mobile apps, and data analysis tools.

Cross Curricular Applications (for both Mission 15 and 16)

- **MATH:** The missions include a lot of advanced math, especially trigonometry. Review angles or how to convert from radians to degrees.
- **MATH:** The mission introduces and uses the concept of a matrix. If your students are able, have a lesson or practice creating and using a matrix.
- **SCIENCE:** The animation of the ball and the paddle uses physics, specifically velocity and distance traveled. Use the game as the introduction or ending to a physics unit.
- **SCIENCE:** Mission 16 goes into more detail about collisions, but the result of the collision is simple. Discuss what happens in a real collision, like with cars or atoms, etc.
- Supports **language arts** through reflection writing.

Final Project	Time Frame: 1-2 weeks
Project Goal: Students will use the skills and concepts they learned during the mission pack "Python with CodeX" to create their own project. **Project Outline:** Follow the five-steps of the design process to design a final project.	**Assessment Opportunities** ● Final project planning document ● Peer reviews / Gallery walk ● Final Project CSTA Standards Rubric ● Final Project Sliding Scale Rubric ● Submit Final Program ● Student Reflection ● Project presentation or report

Teacher Resources

- Use the Final Project Lesson Prep for a planning guide, more ideas, suggestions, and teacher notes.
- Final Project Lesson Plan (for students)
- Final Project Planning Guide (for students)
- Final Project Rubrics – CSTA Standards and Sliding Scale

Remix Ideas:

- Create a new element to a game, like using data from sensors or the accelerometer to control an element on the screen or affect the score.
- Create a new game, like "Simon Says" or "Concentration" or "Family Feud".
- Make up your own game and program it – be original!
- Create an interactive story, with pictures and sound and several possible choices.
- Think of your own creative project that combines concepts and code from several of the missions into something unique and that you find interesting.

Teacher Notes:

- Two rubrics are provided. Choose one or use both. Adjust them to the needs and special interests of your students, or use your own.
- Most state and national computer science standards include **teamwork and time management**. Use the final project as an opportunity for teamwork, leadership roles, electronic communication and managing a project. Review the computer science standards for your state and grade band, and incorporate them into this project. CSTA Standards are listed below.
  - ***Grades 6-8***
  - 2-AP-18: Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.
  - 2-IC-22: Collaborate with many contributors through strategies such as crowdsourcing or surveys when creating a computational artifact.
  - ***Grades 9-10***
  - 3A-AP-22: Design and develop computational artifacts working in team roles using collaborative tools.
  - 3A-AP-27: Use tools and methods for collaboration on a project to increase connectivity of people in different cultures and career fields.
  - ***Grades 11-12***
  - 3B-AP-20: Use version control systems, integrated development environments (IDEs) and collaborative tools and practices (code documentation) in a group software project.
- Most state and national computer science standards also include **evaluating computational artifacts**. Use the final project as a point of discussion for the global impact of computers. The standards are listed below to help guide class discussions, written prompts, project requirements, etc.
  - 2-IC-20: Compare tradeoffs associated with computing technologies that affect people's everyday activities and career options.
  - 2-IC-21: Discuss issues of bias and accessibility in the design of existing technologies.
  - 3A-IC-24: Evaluate the ways computing impacts personal, ethical, social, economic and cultural practices.
  - 3A-IC-25: Test and refine computational artifacts to reduce bias and equity deficits.
  - 3B-IC-25: Evaluate computational artifacts to maximize their beneficial effects and minimize harmful effects on society.
  - 3B-IC-26: Evaluate the impact of equity, access, and influence on the distribution of computing resources in a global society.
  - 3B-IC-27: Predict how computational innovations that have revolutionized aspects of our culture might evolve.

Unit 5 Assessments

Project Goal: Assess students for mastering Unit 5 Vocabulary, Coding and Concepts.

Learning Targets

- I can create my own original program using concepts from Missions 15-16.
- I can define vocab words from Missions 15-16.
- I can answer multiple choice questions that test coding and concepts from Missions 15-16.

Time Frame: 1 hour

Assessment Opportunities

- Final Project
  - Planning guide
  - Final project rubric
  - Final program
  - Presentation
- Student Reflection
- Unit 5 Vocabulary Exam
- Unit 5 Coding and Concepts Exam

Teacher Resources

- Kahoot Reviews: Unit 5 Vocabulary / Unit 5 Coding and Concepts
- For the coding assessment, use the Final Project (Page 51)
  - Two rubrics are provided: CodeX Project Rubric / CodeX Sliding Rubric
- End the unit with the Unit 5 Vocabulary Exam and Unit 5 Coding and Concepts Exam

Final Project Ideas

- See Page 51

Final Exam Rubric

- Final Project CSTA Standards Rubric
- Final Project Sliding Scale Rubric

Possible Pacing Guides

Once-a-week or after-school pacing guide *If your students are in elementary school, you may want to slow down and do more remix projects. The missions after 12 require more time and advanced math and do not have to be completed to get a fun and exciting introduction to programming.*	
Week 1	Mission 1 & Mission 2
Week 2	Mission 3
Week 3	Mission 4
Week 4	Mission 5
Week 5	Remix Project
Week 6	Mission 6
Week 7	Mission 7
Week 8	Mission 8
Week 9	Remix Project
Week 10	Mission 9
Week 11	Mission 10
Week 12	Mission 11
Week 13	Mission 12
Week 14	Remix Project
Week 15	Mission 13
Week 16	Mission 14
Week 17	Mission 15
Week 18	Mission 16

One-semester pacing guide *This pacing guide is for a one-semester class that meets approximately 225 to 300 minutes a week. The guide can be adjusted to meet your class needs. If you want to spend more time on remix projects and review, you can eliminate the last few missions.*	
Week 1	Mission 1, Mission 2, Mission 3
Week 2	Mission 4 and Mission 5
Week 3	Unit 1 Assessment and Remix Project
Week 4	Mission 6 & Remix Project
Week 5	Mission 7 & Remix Project
Week 6	Mission 8 & Remix Project
Week 7	Unit 2 Assessment
Week 8	Mission 9 & Remix Project
Week 9	Mission 10 & Remix Project
Week 10	Mission 11 & Remix Project
Week 11	Mission 12 & Remix Project
Week 12	Unit 3 Assessment
Week 13	Mission 13 & Remix Project
Week 14	Mission 14 & Remix Project
Week 15	Unit 4 Assessment and Remix Project
Week 16	Mission 15 & Mission 16
Week 17	Final Project
Week 18	Final Project

If you are using Python with CodeX as a single unit in an elective course, with only a few weeks, do not try to complete all the missions. Completing Missions 1-8 with remixes is an excellent introduction to Python programming.

If you are using Python with CodeX in elementary school, the math may get complicated in later missions. Completing Missions 1-8, or even Mission 9, with remixes is an excellent introduction for younger students.

If you are using Python with CodeX in a two-semester course, your students will have more time for remixes and Python programming exploration. Also, you can include extra computer science topics, like cybersecurity, the Internet, digital information, and machine learning.

If you are using Python with CodeX for the programming portion of AP CSP, a specific pacing guide is included with the AP CSP materials.

Appendix A: Required Resources

Computer Resources

Each student will need:

- A computer with the Chrome web browser.
- Chromebooks work great – just make sure they are up to date.
- Current versions of Windows OS will work with no additional drivers needed.
- A current Mac OS will also work with no additional drivers needed.
- A USB port is used to connect and program the CodeX. The CodeX comes with a USB to USB-C cable. If your laptop or computer has any other configuration, you will need a cable that has USB-C on one end.

Software Resources

- The interactive textbook and text editor is web-based. Make sure the website is not blocked.
- An email is required for signing in and saving work. It can be a gmail account, but any email will work.
- A per CodeX device (not per student) license is needed to access the curriculum.

Physical Resources

The missions can be completed by individual students or student pairs utilizing pair programming. It is possible to share a CodeX with more than one student or student pair in the same class, but that is not recommended. Each student or student pair will still need a CodeX license for the curriculum.

All CodeX and curriculum licenses can be used throughout the day with different classes and groups of students.

The CodeX comes with a connecting USB cable and curriculum license. Other materials that will be needed throughout the missions are:

- 4 AAA batteries for each CodeX.
- Optional: a flashlight for Mission 12, which uses a light sensor

Notes

- When the CodeX is plugged into a computer, it will appear as a USB mass storage device, similar to a flash drive. This is not required for normal classroom use. So don't worry if your school has a policy preventing flash drives. You just close the pop-up window and continue.

- Occasionally Firia Labs will provide a software update that requires updating the core software on the CodeX. At those times you will need the flash drive feature to update the software, so you will need to use a computer with USB drive access. Often a teacher's computer is used to update all the CodeX.

- If you want to add images or sound files to the CodeX, you will need the flash drive feature enabled.

Appendix B: Our Approach

Physical Computing and CodeSpace: a web-based professional-learning platform

Hardware brings code to life! Our versatile physical computing devices and peripherals get students excited about code. Our CodeSpace learning environment enables them to step up to computer science with real-world text-based Python coding. We include ready-to-teach standards-aligned curriculum with hands-on projects that motivate students.

While there are some great online coding educational programs, we think our approach helps reach a broader range of students. Our approach:

- Gets students focused “off-screen,” programming with physical hardware that connects and interacts independently of their computers.
- Teaches a real, professional programming language. Even younger students appreciate that you can make real money with these exact skills. If they can read, and they can type, they can code in text-based Python.
- Gives students the tools to create *anything* they can imagine. Beyond projects and curriculum, we give students a full-fledged software development environment. These are professional-strength tools for writing code. Instead of a game-playing environment, students can “win with code” through engaging hands-on projects and their own creativity.

Project Based Motivation

Students may wonder why they are learning to code. We all find that knowledge tastes so much better when you’re hungry for it! Our goal is to **motivate** students with tangible, challenging and practical **projects**...that just so happen to require coding to build. We want students to think about how they might code a given project using what they already know. Only then do we teach *just enough* coding concepts to help them get the job done. This approach gives reason and meaning to each concept, as well as relevant problem context, which helps them retain it.

Type it In

Students are often tempted to just copy and paste from lesson examples. Prior to our extensive testing of the curriculum on groups of 4th through 12th grade students, we were concerned that the typing burden might be a problem. But we were willing to risk it.

- Typing in the code forces focus, dramatically improving retention.
- Keyboarding proficiency is key to expressiveness in using a programming language.
- Mistakes in structure, grammar, punctuation, capitalization, etc. are priceless learning opportunities.

Students learn an incredible amount from their mistakes. Our goal is to provide awesome safety-nets for them, guiding them to iterate quickly through successive failed attempts to arrive at a working solution. Extensive classroom observation has convinced us that the typing burden is not a problem. Students dive right in, and they don’t have to be speed typists to make great progress in coding.

Exploration and Creativity

One of the great things about coding is the expressiveness it affords. Coding is a craft that takes time to master, but with only a few basic tools you can start crafting some pretty amazing things! Before they even complete the first project, some of your students will probably be experimenting “off-script” with some ideas of their own. That’s a good thing! We even offer a sandbox for code experimentation In every lesson we list some ideas for re-mixing each project’s concepts. Remember that students are learning programming skills they can use to build *any* application – from controlling a rocketship to choreographing dance moves. Nurture creativity, explore, and instill the joy of coding!

Appendix C: Teacher Resources

If you and your students are still fairly new to text-based coding, don't worry! Like our other physical devices and their curriculum, we've designed the Python with CodeX Mission Pack and this curriculum guide to gently guide you from absolute beginner to a very comfortable level of proficiency. Remember this – Don't Panic 🙂

We understand that tackling a subject like Computer Coding can be pretty intimidating. Fear not, we've built some amazing tools to help you! As you begin this journey, know that the team at Firia Labs is here to help, too. If you run into any problems, just let us know and we'll get you back on track.

Classroom Preparation

Writing code can be like literary writing. Like developing writing skills requires individual practice, learning to code requires students to compose and test their work individually. They need to make their own mistakes and struggle through correcting them.

There is also a place for pair programming and collaboration in the coding classroom. Such practices foster knowledge sharing, collective code ownership and code review "on the go". It also gives students a chance to communicate about what they are learning and reflect on their practices. It builds confidence and keeps students focused on the task. Pair programming can result in better quality work with less errors, and keeps teams "in the flow".

You may need to think about a balance between independent work and pair programming to give your students the best opportunities to succeed and truly engage in and enjoy programming.

Daily Routine

We recommend students work for at least 30 minutes each programming session. Adjust accordingly to your day. Because of the time it takes to set up equipment, log in to computers, and then collect equipment at the end of the learning period, it may take more time than you anticipate. Each lesson has a suggested time frame. This range accounts for completing the basics to continuing with cross-curricular lessons or extensions. Some missions may go even longer, depending on the time you have to spend in coding, the length of time for each mission, the abilities of your students, etc.

This mission pack has a lot of flexibility built-in. You should complete each mission in order, but the amount of time spent on each mission is up to you. A basic pacing calendar is provided, but there is so much flexibility and options for the mission pack. Also, a suggested timeframe for each lesson is given to help you decide how best to plan for the course in your class period with your seat minutes, ability level, other constraints.

For pacing considerations, the mission pack can be:

- A once-a-week activity for an elective class or after school club
- A drop-in unit in a required or elective course
- Extended to a 9-week or 18-week course or 2-semester course

Extensions

Naturally students will progress at different speeds. The material is set up for independent study. You can allow students to work ahead at their own pace, or slow down as needed.

As an alternative to independent work, you can keep the class together and have "high flyers" work on extensions to the missions. Several suggestions are given for each mission. You can also pace the class and engage in rich discussions and extra computer science activities with your students.

Extensions give students a chance to review their learning and add to their program in ways that interest them. Many students will want to experiment with what they've learned, and we offer suggestions along the way to spur this creative tinkering. Extensions are also an excellent opportunity for students to synthesize their learning and create their own projects. We highly recommend including extensions into your pacing calendar. Students can use the sandbox inside CodeSpace to experiment and write their own programs.

Cross Curricular

Another natural extension to each mission is to tie-in the project to other subject areas. Suggestions are given for each mission to extend student learning through cross-curricular project, such as language arts, math and science. These extensions can be separate lessons that build from the mission, lessons given before the mission, or just other ways to look at the project. If you teach multiple subjects or work with teachers in other disciplines, you may want to consider adding in cross curricular extensions as well.

Managing a Class

Our CodeSpace learning platform makes it easy for you to create a class for your students to join, and enables you to monitor their progress.

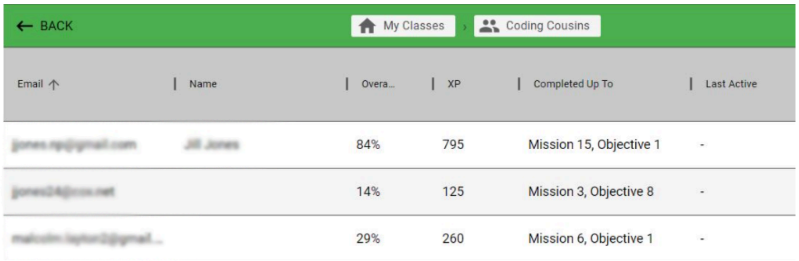
For help and step-by-step instructions, go to "Teacher Dashboard" in the Teacher Resources learning portal. If you are a **Google Classroom** teacher, you can import assignments from CodeSpace into your classes. For instructions, go to "Google Classroom" in the Teacher Resources for Python with CodeX.

If you need assistance for anything, please send an email to: support@firialabs.com

Here are the basics of the CodeSpace Teacher Dashboard

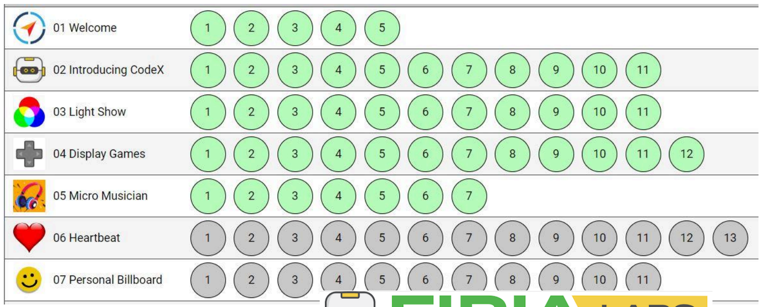
- Log in to CodeSpace and from HELP, select CLASS DASHBOARD
- Once you are in the dashboard, click + in the green bar, top right corner, to add a class.
- Assign each class a name, and allow members to join with a join code.
- You can assign Google Classroom as your LMS.
- After the class is created, you can edit the class, get a join code, disable joining, etc.
- You can delete a student using the "remove" function.
- Students go to CodeSpace and click the SELECT CLASS button.
- They can click the JOIN CLASS button and enter their join code for your class.
- The class will be activated and they are ready to start working!
- In the dashboard, you can see student progress, as a whole class and individually.

Class dashboard



Email ↑	Name	Overa...	XP	Completed Up To	Last Active
[redacted]	[redacted]	84%	795	Mission 15, Objective 1	-
[redacted]		14%	125	Mission 3, Objective 8	-
[redacted]		29%	260	Mission 6, Objective 1	-

Individual progress



01 Welcome	1	2	3	4	5								
02 Introducing CodeX	1	2	3	4	5	6	7	8	9	10	11		
03 Light Show	1	2	3	4	5	6	7	8	9	10	11		
04 Display Games	1	2	3	4	5	6	7	8	9	10	11	12	
05 Micro Musician	1	2	3	4	5	6	7						
06 Heartbeat	1	2	3	4	5	6	7	8	9	10	11	12	13
07 Personal Billboard	1	2	3	4	5	6	7	8	9	10	11		

Appendix D: Assessing Student Projects

The lessons give many opportunities for formative assessment. Any formative assessments you already use in your classroom can be used with programming assignments. Each lesson has suggestions for assessment, including the quizzes embedded in the interactive textbook, turning in completed programs, and Kahoot! Reviews.
Each unit has a vocabulary test and a concepts and coding test. Review Kahoot!s are available for each. Also, the remix project for each unit can be used for assessment. A rubric checklist is included for each remix project.

Remix Projects

Two different project rubrics are provided – one that maps to CSTA standards and another that is a sliding scale for mastery. They can be used as a written form, or made into a digital form. The rubrics can be modified and edited as needed. You can also customize the rubric by adding custom requirements or assigning point values before students begin.

Students should be given a copy of the rubric before beginning the project. Discuss the criteria and what it means to earn mastery. It is beneficial to give students time to revise and improve their projects, as time permits. Students who approach mastery may be motivated to improve, so decide what your classroom policy and expectations will be and explain them to students early on. You may need to revise policies as you get to know your students and observe how CodeSpace works for them. Flexibility is important!

Student-Teacher Conferencing

Student-teacher conferencing is integral to the learning process. This takes more time in class, but this is not wasted time! Students will work harder and be more willing to do revisions, which is truly a workplace life skill we'd like to instill in our students. To manage the process, it helps to have a submission window, rather than one set due date. Once a student submits their work, call him/her up for a conference. Begin with an open-ended question, like "Tell me about your project." Then move on to the rubric. This may give you insight into who did what, if working in pairs, and what challenges they encountered. As you conference about the rubric, ask them what level of mastery they think they achieved, and why. Students are often more critical of their work than they need to be. It's a good time to emphasize that challenges and mistakes are learning opportunities rather than just being "wrong." If time allows, students should be allowed to debug and improve before a final submission of their work.

Peer Feedback

Before students submit a remix project, they should complete a peer review. This may take modeling a few times before students do it correctly. Remind students that revising is just as important here as it is in English class. These revisions can lead to great conversations during the conferencing process. They should go through the checklist or rubric and test the program just as you would. This will give them the chance to find and correct mistakes before doing a student-teacher conference. A peer review form is included in the remix log for every remix.

Early Finishers

Students who finish earlier than the submission deadline may enjoy having time to work on other unscripted projects, and just trying things out. This is not wasted time! Learning through trial and error is time well-spent, and we want to encourage curiosity for their motivation.

Appendix E: Links to teacher materials

General Resources for Teachers	Includes Curriculum Guide, Vocabulary by Mission, Python by Mission, the Python with CodeX Appendix (text for all missions), Unit Review and Test Links, Toolbox Appendix, and Project Rubrics
Unit 1 Resources	Includes Missions 1-5 Adding Audio Files, Mission Reminders, Analog and Digital Sound
Unit 2 Resources	Includes Missions 6-8 and Adding Image Files
Unit 3 Resources	Includes Missions 9-12 and Gravity Project
Unit 4 Resources	Includes Missions 13-14
Unit 5 Resources	Includes Mission 15-16 and Final Project
Answer Keys by Mission	In alphabetical order, by objective
CodeX Remix Projects	Includes general-use remix projects for Missions 3-12 and solutions
Elementary Resources	Includes Elementary Guide, Getting Started, Resources for Missions 1-12, and Remix Projects. *These resources can also be used with other special needs students, or any students.*
AP CSP Resources	Includes a pacing guide and several supplemental lessons to prepare students for the Create Performance Task
Cross Curricular Projects	Includes projects in Language Arts, Math and Science, and Art; with solutions
Standards Alignment	CSTA, Common Core, ISTE, and individual states
Teacher Resources for Python with CodeX	A general starting-off point to find anything you want or need
Firia Labs Documentation	Python modules, Primer, Working the Files, Hardware Reference
License and Dashboard	Help with licensing or using the teacher dashboard
Video Library	Videos that showcase remixes and cross curricular projects
Project Showcase	More videos that showcase cool CodeX projects
Teacher Community	A professional learning network and teacher discussion board