MISSION 3: Light Show	Time: 45 minutes

Overview:	Cross Curricular:
This mission will teach students to turn on (and off) the four pixel LEDs on the CodeX. They can be set to any color using RGB triplets, called tuples. The basic lesson will use built-in colors.	● **ART:** discuss colors used in paints, and mixing colors; compare to mixing light ● **SCIENCE:** This lesson introduces RGB lights and how they can make all colors. Have a lesson that discusses this. ● Supports **language arts** through reflection writing

Materials Included in the learning portal [Teacher Resources](#):

Mission 3 Slidedeck
> The slide deck is for teacher-led instructions that let you guide students through the material using the slides. It is an alternative to the students reading a lot of instructions in CodeSpace. The slides mirror the instructions, with simplified language that is chunked into smaller sections at a time. The information is shown on slides with "Objective". The tasks to complete are on slides with "Mission Activity".

Mission 3 Workbook
> The workbook can be used instead of slides for student-led or independent work. It is an alternative to students reading a lot of instructions in CodeSpace. It mirrors the instructions (and the slide deck), with simplified language that is chunked into smaller sections at a time. Each objective is on its own page. The tasks to complete are labeled "DO THIS" and have a robot icon next to it.

Mission 3 Log (and answers)
> This mission log is the worksheet for students to complete as they work through the mission. It should be printed and given to each student before the mission starts. They write on the mission log during the assignment and turn it in at the completion of the mission (assignment).

Mission 3 Lesson Plan
> The lesson plan comes from the CodeX Teacher Manual and is included here for easy reference.

[Mission 3 Remix Folder](#)
> Following Mission 3, students should complete a remix of their code. Get supplemental materials from the folder.

Additional Resources:	Formative Assessment Ideas:
● **CodeX mission reminders** – in Unit 1 ● **[Pixels video](#) (on youtube)** ● **Mission 3 Solution (Pixel1)** – in Answers section ● **[Kahoot](#)** (Missions 1-3)	● Exit ticket ● Mission log completion ● Completed program ● [Kahoot Mission 3 Review](#)

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)

Preparing for the lesson:

Students will use the Codex throughout the lesson. Decide if they will work in pairs or individually.

- Look through the slide deck and workbook. Decide what materials you want to use for presenting the lesson. The slide deck can be projected on a large screen. The workbook (if used) can be printed or remain digital through your LMS.
- Decide if you want to discuss CodeX mission reminders. Since this is the first real program, they may not need the reminders yet. However, you may feel like they are good reminders to go over right from the start.
- Be familiar with the Mission Log (assignment) and the questions they will answer.
- Print the Mission Log for each student.
- This lesson uses a youtube video on pixels. Can you access youtube, or do you need an alternative?
- If you have a word wall, or another form of vocabulary presentation, prepare the new terms.
- Do you want to have a lesson on color? If so, prepare it and decide when you will teach it.

Lesson Tips and Tricks:

💡 **Teaching tip:**
You can use a variety of discussion strategies to get the most engagement from your students. For example, you can have students write their answers before asking anyone for an answer. You can use one of many think-pair-share methods. You can have students write their answer and share with someone, and then have other students share answers they heard from their peers. You can randomly select students to answer.

👫 **Pre-Mission Discussion:** Slide 2, page 1

Students can write in their log first and then share, or discuss first and then write in their log.

- What are primary colors?
  You can bring up here the difference between pigment and light, or let students discover this organically.

💻 **Pre-Mission Activity:** Slides 3-4, page 2

These slides discuss how the LEDs use three lights to display all colors. It doesn't go into any detail, just states that it happens. The slide deck includes a youtube video that briefly (under 2 minutes) talks about pixels. The Mission 3 Remix goes into more detail and has another video clip you can show that talks about RGB.

💡 **Teaching tip – Reminders for the beginning of mission: (go over these if you think it will help your students)**
These reminders are organized on a short slide deck that can be shown to students at the beginning of class
- 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!
- You are making a project – not just working random problems. Focus on the *project-based* objectives and avoid rushing through the material too quickly.
- Test your understanding along the way by "coloring outside the lines". Try stuff!
- Collect all the Tools you find! (They're indicated with a **wrench** icon)
- Read carefully – usually the answer is right there in front of you!

💻 **Mission Activities:**

Most of this lesson is on the computer, writing code to turn the LED lights different colors.

- Each student will complete a Mission Log.
- Students could work in pairs through the lesson, or can work individually.
- Students will need the CodeX and USB cable.

💡 **Teaching tip: Objective #1** -- slides 5-8, pages 3-4
Students use the 3D simulator to identify one of the LED pixels (NeoPixels). They also create a new file for the Mission program.

💡 **Teaching tip: Objective #2** -- Slides 9-10, page 5
Students write 2 lines of code to turn on a NeoPixel. The programming term "argument" is emphasized here. Arguments are the values in the parenthesis when you use a function like pixels.set(). The argument is the value the function needs to complete its task. In this case, the function pixels.set() needs to know which light and what color (2 arguments).

💡 **Teaching tip: Objective #3** -- Slides 11-12, page 6
Students add a line of code to also turn the NeoPixel green. They should have three lines of code -- don't delete the line that turns the NeoPixel red. But when they run the code, they will only see the green. CodeTrek mentions this, but it is easy to overlook. This is what the code is supposed to do.

💡 **Teaching tip: Objective #4** -- Slides 13-14, page 7
Without fixing the speed problem yet, students will add two more colors to their code, for a total of 5 lines. Students are asked to ***make a prediction*** of what will happen in their mission log before running the code. When the code runs, they will only see the last color.

💡 **Teaching tip: Quiz** -- Slide 15, page 7
students take a ❓ short quiz making a prediction. Quiz question below. You can decide if you need to go over the question with your students.

💡 **Teaching tip: Objective #5** -- Slides 16-17, page 8
This objective introduces the debugger. This is a powerful tool embedded in the text editor that lets you see your code execute the statements one line at a time. There is a ***short video*** that shows how to use the debugger that is included in the instruction panel. You can decide if you want to go over this video together. You may even want to ***model using the debugger*** a couple of times before students try it on their own.
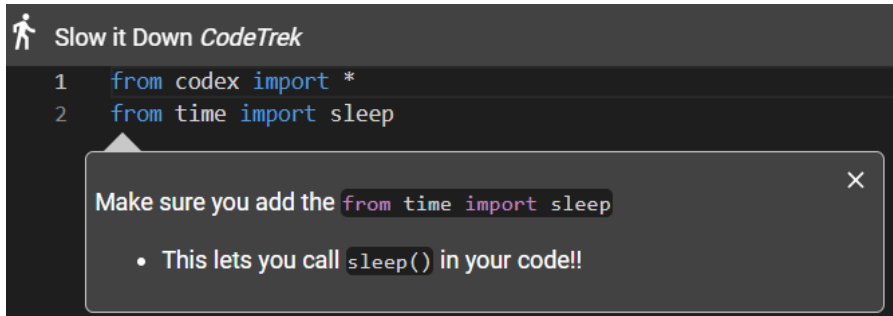
💡 **Teaching tip: Objective #6** -- Slides 18-19, page 9
Students use the debugger in this objective. By slowing down the code, they see all four colors. The thing to get comfortable with is that you see the execution of the code AFTER it is highlighted. We expect to see it happen as the code is highlighted, but it happens AFTER. You will want to point this out several times, particularly when you are demonstrating using the debugger.

💡 **Teaching tip: Objective #7** -- Slides 20-21, page 10

A new function is introduced: sleep(). It requires 1 argument -- the amount of time the code "sleeps", or the execution of commands is delayed.

IMPORTANT: Students must import the time module or they will get an error.

💡 **Teaching tip: Objective #8** -- Slides 22-26, page 11-12

Two important vocabulary words are introduced here: literal and variable. These words are used a lot in programming. Some time is spent particularly on variables, and examples are given. Spend more time here if you think your students will need it.

A particular tricky concept can be the difference between the function **sleep()** and the variable **delay**. Students may mistakenly think that **delay** causes the program to wait. But actually, the **sleep()** function does that. The variable **delay** simply tells the **sleep()** function how long to wait. Their understanding of this concept will be tested in the quiz following this objective. Many students will get the quiz question wrong. You may want to go over this frequently during other missions to help the concept really sink in.

💡 **Teaching tip: Quiz** -- Slide 27, page 13

Students take a second ❓ short quiz about variables. Quiz questions below. You can decide if you need to go over the question with your students. The concept of import isn't covered in great detail during these lessons for elementary school students. You can go into detail if you want to, or simply help students know that they need to use the import statement to access some functions.

💡 **Teaching tip: Objective #9** -- Slide 28-30, page 13-14

Students use a second variable for color and use the variable in their code to change the NeoPixels more than one color. They build on their code for the final objective of the mission. They can be creative here. They need to meet the objective by using a color variable set to RED. Anything else they want to add, using their own creativity, is an added bonus.

The solution code given in the folder shows the option with flashing red and yellow lights, but it isn't required.

💻 **Mission Complete:**

This mission ends with a completed, working program. You need to decide how you will use the program for assessment. You could:

- Go to each student and check-off their code
- Have the students download their code to a text file and turn it in using your LMS
- Have students print their code (either download and then print the text file, or print a screenshot)
- Have students switch computers and run each other's code. Fill out a simple rubric and turn in to teacher
- Any other way that words for you

👫 **Post-Mission Reflection:** Slide 31, page 15

The post-mission reflection asks students to think about color LED lights and how they could be used. You can change the question if there is something else you want to emphasize with your students.

- What are some ways you see Smart LEDs (programmable lights) being used?
  Answers could include stadium signs, traffic signs, wearable clothing, etc.

End by collecting the Mission Log and any formative assessment you want to include.

💻 **IMPORTANT Clearing the CodeX:** Slides 32-34, page 16

The last code that is run on the CodeX STAYS on the CodeX until a new program is run. If the CodeX is shared with other students, their last program will be on the CodeX. Even if not sharing, it is good practice to clear the CodeX of any meaningful code at the end of each class period.

The last three slides in the slide deck walk the students through creating a short **Clear** file that can be run at the end of each class period and basically erase any meaningful code. This information is also included in the workbook and as stand-alone slides.

Please help your students go through this process the first time, and guide them at the end of each day to open the **Clear** file and run it before returning the CodeX.

SUCCESS CRITERIA:
- ❏ Define RGB, literal, and variable
- ❏ Define and use a variable delay in sleep()
- ❏ Define and use a variable for color that is changed and used multiple times
- ❏ Debug any errors in the code
- ❏ Write a program, run it, and save it to the CodeX
- ❏ Clear the CodeX of meaningful code

? Quiz #1 Questions

Two Images

What do you expect the following code to do? +5 XP

```
from codex import *
display.show(pics.HEART)
display.show(pics.HAPPY)
```

✅ Display each image quickly and end showing the last one

❌ Show the images for about 1 second each

❌ Display only the first image

NEXT

? Quiz #2 Questions

Variable Questions

What does `from codex import *` do? +5 XP

❌ Turns on the codex LEDs

❌ Imports asterisks from the land of codex

✅ Provides access to built-in codex code

What does `delay = 1` do? +5 XP

❌ Delays execution for 1 second

❌ Puts the CPU into sleep mode for 1 second

✅ Assigns the value 1 to a variable named 'delay'

NEXT