



<b>MISSION 8: Answer Bot</b>		<b>Time: 45 minutes (mission 8) (45 minutes challenges)</b>
<p><b>Overview:</b></p> <p>This project builds on the concept of selecting from a <b>list</b> of items and adds <b>random</b> number generation to the mix. Up to this point the CodeX has been pretty predictable – as you’d expect a computer to be! But some applications need randomness, or unpredictable results. 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.</p> 		<p><b>Objectives:</b></p> <ul style="list-style-type: none"> <li>• I can apply properties of lists to a new program.</li> <li>• I can use randomization with lists</li> <li>• I can create and call a function</li> </ul>
<p><b>Standards:</b></p> <p><b>2-AP-12</b> Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.</p> <p><b>3A-AP-14</b> Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.</p>	<p><b>CSP Framework:</b></p> <p><b>Computational Thinking Practices:</b></p> <p>2.B Implement and apply an algorithm</p> <p>3.C Explain how abstraction manages complexity</p> <p>4.C Identify and correct errors in algorithms and programs, including error discovery through testing.</p> <p>6.A Collaborate in the development of solutions.</p>	<p><b>Key Concepts:</b></p> <ul style="list-style-type: none"> <li>• Random number generators are crucial for many computer applications.</li> <li>• Lists can be used for a variety of purposes and uses in a program</li> </ul>
<p><b>Preparation:</b></p> <p><b>Make a copy</b> of the assignment or put it in the LMS.</p> <p><b>Prepare</b> formative assessments</p> <p>Review using Photopea so you can help students prepare their own images</p>	<p><b>Links:</b></p> <ul style="list-style-type: none"> <li>• Codespace: <a href="https://sim.firialabs.com/">https://sim.firialabs.com/</a></li> <li>• <a href="#">Assignment</a></li> <li>• <a href="#">Adding JPG images</a> (slides)</li> <li>• <a href="#">Adding JPG images</a> (doc)</li> <li>• Daily reflection form</li> <li>• <a href="#">Mission 8 code solution</a></li> <li>• <a href="#">Mission 8 after challenges</a></li> </ul>	<p><b>Agenda:</b></p> <ul style="list-style-type: none"> <li>• Warm-up (5 minutes)</li> <li>• Mission 7 (40 minutes)</li> <li>• Extensions (15 minutes)</li> <li>• Challenges (30 minutes)</li> <li>• Wrap-up (5 minutes)</li> </ul>
<p><b>Vocabulary:</b></p> <ul style="list-style-type: none"> <li>• <b>Range:</b> A sequence of numbers you can iterate over. You must provide at least the stop (or last) number in the sequence. Optional: you can provide the start (or first) number in the sequence and also the step, if other than increasing by 1.</li> <li>• <b>Constant:</b> A named value that doesn’t change during the run of the code. By convention, constants are represented with ALL CAPS</li> </ul> <p>Review <b>list</b> (mission 7) and <b>conversion function</b> (mission 4)</p>		
<p><b>Assessment:</b></p> <ul style="list-style-type: none"> <li>• Daily reflection form</li> <li>• Meaningful notes (or notes to your future forgetful self)</li> <li>• More suggestions listed below in the Walk-Through Wrap-up</li> </ul>		

## Teaching Guide

### Warm-up (5 minutes)

The actual coding part of this Mission is about one normal class period. The extensions and challenges extend the learning and also incorporate thinking, flowcharts and functions.


 **Discuss** – Use a discussion strategy, like journaling, working at boards, selecting random students, or a form of think-pair-share.

- **Topic:** This is a project with *practical* applications. In our previous program, the programmer or the person running the program had control over when something happened or what image appeared. However, in real life and computer applications, we might have a lot of randomness
- What are some examples of when we need something randomized?


**DISCUSS REAL WORLD APPLICATIONS:**

- Video games
- Secure password encryption
- Real-world simulator trainers
- Scientific statistical sampling

### Activity – Mission #8 (40 minutes)

 Randomly group students into pairs for pair programming.

Students log in to one computer. Two computers can be used if they want to see instructions on one computer and work on the other computer. However, the assignment document requires snippets, so it will need to be open on the same computer as CodeSpace.

 **Teaching tip – Before they start:**

Optional: Review the [Mission Reminders slides](#).

**Important!** Remind students that they need to document their errors and how they fixed them. There is a table at the end of the assignment document for this.

Students go to [sims.firialabs.com](https://sims.firialabs.com) and should be at the beginning of Mission 8

 **Teaching tip – Objective 1:**

The code in this objective will cause an error. The instructions clearly state this, but students aren't always good at reading the instructions. The assignment document asks them if they know what the error is and how to fix it. They will do this on the next objective, but hopefully they remember before the instructions help them.

 **Teaching tip – Objective 3:**

This objective shows students an additional parameter to `display.print`. I recommend using the keyword "parameter" so they become familiar with it. The instructions have them use `scale=3`. They can try different numbers for the scale. When the scale makes the text too big for the screen, it will turn into strange lines and not be readable. When this happens, they need to adjust the scale smaller.

 **Teaching tip – Objective 5:**

Students create a list. The list can contain any text that is school appropriate. The example is for food, but it can be anything they are interested in (sports, fashion, food, games, etc.)

**Completed code for Mission 8 is available in the folder.**

### Teaching tip – Objective 6:

Students will use another list, but they DO NOT need to create it. The instructions show the list, but they do not need to type it. The list used has all the built-in colors for the neoPixels, and the list is already created. Students simply need to access it: `COLOR_LIST`.

### Teaching tip – Extensions:

There are two extensions included in the lesson and should be completed by the students to solidify their learning and practice flowcharts and functions.

### Teaching tip – Extension #1:

Add a kill switch. Students should have done this several times by now and shouldn't need assistance.

### Teaching tip – Extension #2:

Students create a function for the section of code that turns the neoPixels random colors.

### Teaching tip – Challenges:

There are three challenges included in the lesson. The challenges can be done in any order, and they don't all have to be done. I recommend the order they are presented, but it is not necessary. An explanation of each challenge is included below. Also, completed code with all the challenges is available in the folder.

### Teaching tip – Challenge #1:


Modify the function created in Extension #2 so that it uses a loop. The function uses the same two steps four times, so a loop can be used. As a hint, a flowchart showing the loop is included at the end of the assignment.


### Teaching tip – Challenge #2:

Add some of the built-in images to the list. Students use `display.print` for the text, but will need `display.show` for the images. Therefore, they need to include an if statement and check the type of the item from the list (x). If the type is `str`, use `display.print`. Otherwise use `display.show`. If you need help with this, check out [the code](#).


### Teaching tip – Challenge #3:

This challenge shows students how to prepare their own images to use on the CodeX. They prepare and then upload a few JPG images. Then students need to create a second list for these images. Program the B button to display an image from the new list and keep button A as it is. This part is optional, so if students struggle or don't get to it, it is okay.


 Assignment is complete and ready to turn in. Both students should include their names on the document.


 Determine how you want to check-off the student program (turn in text file, submit through LMS, observe on student computer, etc.)

## Wrap-Up (5 minutes)

 **Vocabulary** – Review the vocabulary for today’s lesson:

- **Index:** A number that keeps track of what choice should be displayed.
- **Nested Condition:** Another if statement that is part of (embedded in) the block of code in an if statement (an if statement within an if statement).
- **List:** A sequence of items you can access with an index.

 **Discuss** – Use a discussion strategy, like journaling, working at boards, selecting random students, or a form of think-pair-share.

 **REVIEW the TOPIC:** You may want to review how random numbers, or random selection from a list, can be used in other practical applications.

### **IMPORTANT!!**

Students should clear their CodeX by running their ClearCodeX program.

Formative Assessment:

- Daily reflection form
- Completion of assignment and/or mission
- Exit ticket on vocabulary
- Group review on vocabulary
- Programming journal (add notes and vocabulary)
- Students create a vocabulary canvas with vocabulary words.

### **SUCCESS CRITERIA:**

- Program the buttons to print a random number.
- Modify the code to print a random message from a list of possible answers.
- Create a function that randomly assigns a color to each of the four neoPixels.