MISSION 8: Answer Bot	Time: 45-60 minutes

Overview:

This project builds on the concept of selecting from a **list** of items and adds **random** 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.

Cross Curricular:

- **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 reflection writing.

Materials Included in the learning portal Teacher Resources:

Mission 8 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 8 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 8 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 8 Lesson Plan

The lesson plan comes from the original CodeX Teacher Manual and is included here for easy reference.

Adding JPG Images (Optional Lesson, follows Mission 8)

This lesson is completely optional. The lesson shows students how to take their own images (or those from the Internet) and preps them as JPG files to use in a CodeX program. If you choose to do the lesson, it should be completed before the remix.

Mission 8 Remix

Following Mission 8 students should complete a remix of their code. Get supplemental materials from the folder.

Additional Resources:

- Mission 8 Solution (Answer Bot) – Answers section
- Mission 8 Review Kahoot

Formative Assessment Ideas:

- Exit ticket
- Mission log completion
- Completed program
- Mission 8 Review Kahoot
- Student Reflection

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)

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.
- Be familiar with the Mission Log (assignment) and the questions they will answer.
- Print the Mission Log for each student, or prepare it digitally.
- The mission program does not need to be portable. If you want students to use the CodeX without a cable, then have batteries available.

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.

There is one question for the pre-mission. There aren't any “right” answers here. The purpose is to get them thinking about the need for randomness. Also, there are real-world applications to what they are learning.

- In the last mission, the person running the program had control over what happened. Sometimes values need to be random. What are some examples of when you might need something random?

Possible answers: Video games, secure password encryption, real-world simulator trainers, scientific statistical sampling

💻 **Mission Activities:**

Most of this lesson is on the computer, writing code to make a variable speed heartbeat.

- 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 3-5, Pages 2-3

This objective is review. Students program use display.show() with a number. It will throw an error.

🗝️ **NOTE:** Students will get an error. This is to be expected. They will fix the error in the next objective.

💡 **Teaching tip: Objective #2** -- Slides 6-7, Page 4

Students fix the error from Objective #1 by using display.print() instead of display.show()

💡 **Teaching tip: Objective #3** -- Slides 8-11, Pages 5-7

Students learn about the random module, and one of its built-in functions: randrange(endValue). This function returns a random number between 0 and one less than the ending value. Examples are given in the instructions. For your information, you can supply a starting value as well. The default starting value is 0. The starting value is always included, and the ending value is not included.

Students then learn that the randrange() function is great for using with lists, since the index also starts at 0 and then goes to one less than the length. Example code is given.

Another concept is introduced: the ability to change the size of the text. The display.print() function has another argument for size. The values for scale given are 1, 2 and 3, but it can go much larger. If a scale is too large for the text to display on the screen, it comes out as shapes and gibberish. So if you see that happening, the scale needs to be smaller.

In the "Do This" activity, most of the code is given, but the student must fill in the end value. It should be 10.

💡 **Teaching tip: Objective #4** -- Slides 12-15, Pages 8-9
This objective starts with students writing in their mission log what they remember about CodeX buttons and loops. You might have students share what they remember, or look over the code for Heartbeat (mission 6) or Billboard (mission 7). Both programs use a while loop and if statements for button presses. The same code will be used for this program.

Students modify their code by adding a while True loop and an if statement for button A that will generate and display a random number.

In the "Do This" activity, most of the code is given, but the student must fill in the if statement for the button press. The code is `if buttons.was_pressed(BTN_A):`

💡 **Teaching tip: Objective #5** -- Slides 16-19, Pages 10-11
Students will create a list and use a random number for the index. They start by deciding on a topic for the question and generating a list of answers. Students will do this on their mission log.

The "To Do" activity is divided into two slides. The first part is creating the list in the code and creating a variable for the number of items in the list – len(answers). The second part is modifying the if statement to use the list and random number.

NOTE: the code in slidedeck and workbook is slightly different than the instructions in CodeSpace. This is to simplify the learning curve and keep the terms and variables consistent. This code will still pass off the goals and allow the students to continue through the mission. Either the code here or in CodeTrek can be used. Just be aware that there are slight differences.

💡 **Teaching tip: Quiz** -- Slide 20, Page 12

Students take a ❓ short quiz. The 2 Quiz questions are below. You can decide if you need to go over the question with your students.

💡 **Teaching tip: Objective #6** -- Slides 21-23, Pages 12-13
This objective shows students another application for random numbers – getting a random color and flashing the pixels during the program. The codex module has a built-in list of colors called COLOR_LIST. Students can access

and use the list, they DO NOT need to type it in their program. All list functions can be used with the list.

Sample code is given during the instructions, and the code for the activity is also given. If you want your students to be more independent thinkers, delete the code from the activity and let them develop it on their own, using the code snippets from the instructions.

NOTE: Once again, the code here is slightly different than CodeTrek, to keep the simplicity and avoid confusion. Either code will work, just be aware there are slight differences.

💡 **Teaching tip: Objective #7** -- Slides 24-28, Page 14-16
Another random module function is introduced. The random.choice() function combines the variable for a random number with assigning a value from a list into one short command. The list is the argument, and the function does the rest. This is pretty handy!

Students will use random.choice() throughout the code instead of random.randrange() and assign a value from the list. This means students will need to delete code, and then make modifications. All the code needed is given in the instructions, so it is not given in the “Do This'' activity. However, the last slide (and page) do show the completed code. Delete if you don’t think your students need it.

NOTE: Once again, the code here is slightly different than CodeTrek, to keep the simplicity and avoid confusion. Either code will work, just be aware there are slight differences.

💻 Mission Complete:

This mission ends with a completed, working program that will use a list for colors and a list for answers, with random selections for each. 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 works for you

👫 Post-Mission Reflection:

The post-mission reflection asks students to think about real-world applications for random numbers and items from a list. You can change the questions if there is something else you want to emphasize with your students.

- What are some projects you are interested in that might use random numbers or random items from a list?
- What is one error you made in coding today? How did you fix it?

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

💻 IMPORTANT Clearing the CodeX:

The students have already created a “Clear” program. Students should open and run “Clear” at the end of each class period.

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

❓ Quiz Questions

Why does `range(10)` only go up to 9?? +5 XP

- ☒ The variable itself consumes one integer so there are only 9 values in `range(10)`.
- ☑ Ranges start at 0 (zero), and there are 10 values from 0-9.
- ☒ The `range(10)` also includes -1 so the max is 9.

What is the `count` variable doing for you in this program? +5 XP

```
answers = ["0", "1", "2"]

while True:
    if buttons.was_pressed(BTN_A):
        count = len(answers)
        index = random.randrange(count)
```

- ☒ The `count` variable automatically scans the list and counts the number of items.
- ☑ The `count` variable stores len(answers) to give to the 'randrange' function.
- ☒ The `count` is a beloved character in educational television.