MISSION 6: Heartbeat

Time: 30-45 minutes

Overview:

The end goal of this project is simple – a continuously flashing heartbeat. Along the way it reinforces understanding of coding concepts learned so far, and ushers in the crucial concept of **loops**.

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.)
- Supports **language arts** through reflection writing

Materials Included in the learning portal Teacher Resources:

Mission 6 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 6 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 6 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 6 Lesson Plan

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

Mission 6 Remix Folder

Following Mission 6, students should complete a remix of their code. Get supplemental materials from the folder.

Additional Resources:

- **CodeX mission reminders** – Unit 1 Section
- **Mission 6 Solution (Heart2)** – Answers section
- Kahoot (Mission 6)

Formative Assessment Ideas:

- Exit ticket
- Mission log completion
- Completed program
- Kahoot Mission 6 Review

Vocabulary:

- **Loop:** Repeats a block of code, subject to a given condition.
- **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 loop 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)

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. This is the 6th mission, so it may not be necessary.
- Be familiar with the Mission Log (assignment) and the questions they will answer.
- Print the Mission Log for each student.
- If you have a word wall, or another form of vocabulary presentation, prepare the new terms. Review a few terms from earlier missions.
- 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 animated output and user input. Also, there are real-world applications to what they are learning.

- Make a list of blinking indicators, like flashing road signs or traffic lights.
  - Possible answers: turn signal on car or bike, Christmas lights, Movie marquee sign, airport runway

💡 **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

💻 **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 2-4
This objective is all review. Students recreate Heart1, with the CodeX displaying a heart image.

💡 **Teaching tip: Objective #2** -- Slides 6-7, Page 5
Students add a second image to the program. You may want to review or remind students about the need for a sleep() in between the two images so they both display. This was first covered in Mission 3.

💡 **Teaching tip: Objective #3** -- Slides 8-9, Page 6
Students copy and paste the two heart images to create a heartbeat. Students will not keep this code, but it sets up the need for a loop to repeat code.

💡 **Teaching tip: Objective #4** -- Slides 10-12, Page 7-8
Students learn about loops and conditions. They will look up information in the toolbox and record their findings on their mission log. Take your time here. You may want to stop and go over this information, or have a student share-out.

Then students use a loop in their code to repeat the heartbeat. Their current code will need to be indented in the loop. They can use the TAB key for this. You can demonstrate this keyboard shortcut: to quickly indent code that is already typed, you can highlight the code and press TAB.

Remind students: You only need ONE heartbeat cycle in the loop, so remember to delete all the extra code they added for Objective #3.

Students will need to manually stop the code by pressing STOP.

💡 **Teaching tip: Objective #5** -- Slides 13-15, Page 9
The definition of an infinite loop is given. Students will write this in their mission log. Then they experiment with different numbers for delay. They can try any number. Only the number 2 is required to pass off the objective, but students should be free to experiment and try other numbers.

💡 **Teaching tip: Objective #6** -- Slides 16-18, Pages 10-11
Students learn how to break out of the infinite loop using an if statement, a button as input, and the break command. Students first review the buttons and the name of each button. This is recorded in their mission log. Then students add the code to their program.

💡 **Teaching tip: Objective #7** -- Slides 19-24, Pages 12-14
This objective has a lot of review. Students review branching (if statements) and buttons as input. If needed, you can take your time here and review the concepts with the students. With each review, examples are given to help students process the information. You can come up with more examples if needed. Or even give students sample code in small groups and have them label the parts and predict what will happen. After reviewing the concepts, students will go through their code using the debugger. You may need to do this with the students or review how to use the debugger. They will need to step over at least 8 times to see the if statement work. Make sure they step through several times before pressing the BTN_A.

💡 **Teaching tip: Quiz** -- Slide 25, Page 14

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 #8** -- Slides 26-29, Pages 15-16
Students learn about the float data type. These are real numbers. The decimal in a real number is referred to as a floating point. Thus the data type is called "float". Examples of float values are given. Then a review of all the data types so far is given. If needed, you can provide extra practice or review of all the data types. Then a float value is used in sleep().

🗝️ **NOTE:** Students have been using a variable for delay, and using delay in sleep() up to this point. To pass off the objective, they need to type the actual value in sleep() and not use the variable. If students are getting the objective passed off, check this.

💡 **Teaching tip: Objective #9** -- Slides 30-31, Page 17
This objective just has students use a delay variable in sleep(). They return their code to how they had it in earlier objectives. You shouldn't need to spend much time here.

💡 **Teaching tip: Objective #10** -- Slides 32-35, Pages 18-19
Students learn about incrementing a variable. This is when you add the same number to a value. It is like counting by a number. Incrementing by 1 is like counting. In this objective, students increment by 0.2 to slow down the heartbeat. A larger value for delay will cause a slower heartbeat. You can give students more practice with incrementing by trying different values. For example: count = count + 1 or twos = twos + 2, etc. They need to realize that every increment takes the current value and adds to it, so it is always changing by the same amount.

Then students change the if statement to increment delay in their program.

💡 **Teaching tip: Objective #11** -- Slides 36-41, Pages 20-22
Students learn about decrementing a variable, which is similar to incrementing but using subtraction. Again, you may want to go over additional examples with students. An additional slide is added to inform students about a potential runtime error with the code. A runtime error doesn't happen until the code is running, and something happens that causes the program to stop. In this case, if the delay used in sleep() is 0 or less, a runtime error will happen. The final check for this objective is to cause the error by decrementing delay too many times (pressing BTN_B).

Students will also give an example of increment and decrement in their mission log.

💡 **Teaching tip: Quiz** -- Slide 42, Page 23

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.

💻 Mission Complete:

This mission ends with a completed, working (short) program. However, the possibility of a runtime error still exists. That is okay. 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 using buttons as input. You can change the question if there is something else you want to emphasize with your students.

- What are some things with buttons you might want to program to control something?
  Possible answers: tv remote control, game controller, push button light dimmers

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

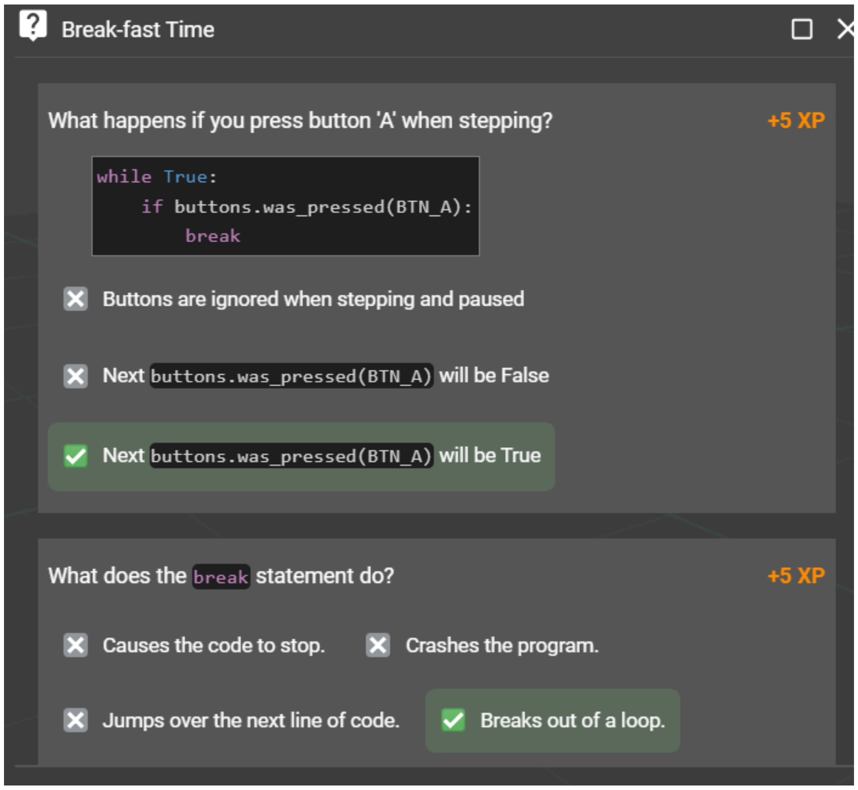
💻 IMPORTANT Clearing the CodeX:

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

SUCCESS CRITERIA:

❏ 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 any errors in the code
❏ Clear the CodeX of meaningful code

❓ Quiz #1 Questions



Break-fast Time

What happens if you press button 'A' when stepping? +5 XP

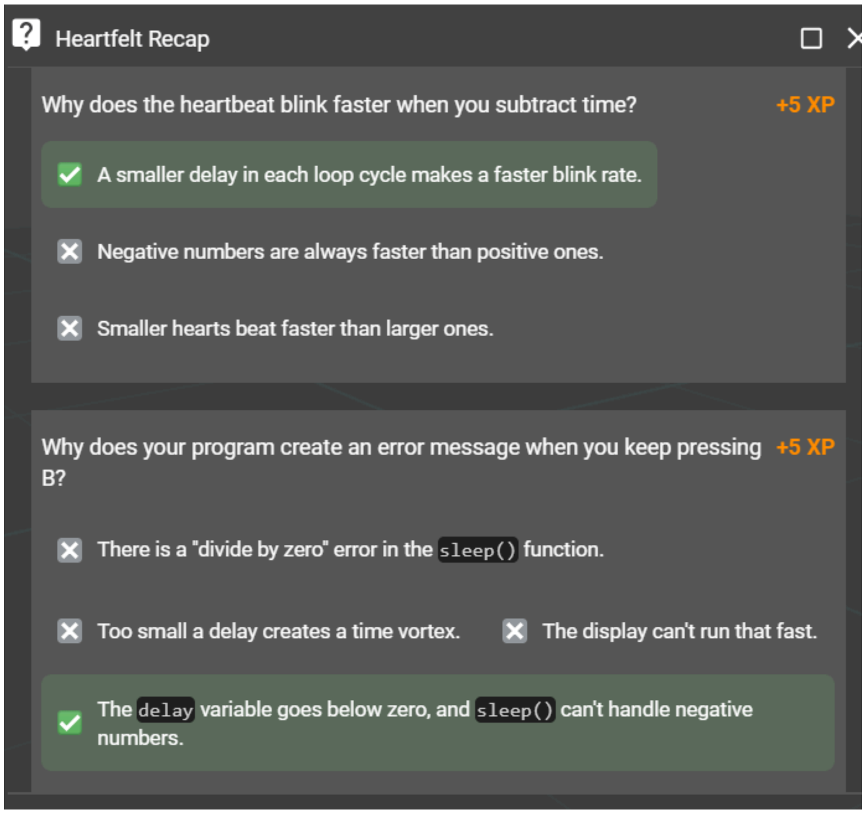
```
while True:
    if buttons.was_pressed(BTN_A):
        break
```

- ✗ Buttons are ignored when stepping and paused
- ✗ Next `buttons.was_pressed(BTN_A)` will be False
- ✓ Next `buttons.was_pressed(BTN_A)` will be True

What does the `break` statement do? +5 XP

- ✗ Causes the code to stop.
- ✗ Crashes the program.
- ✗ Jumps over the next line of code.
- ✓ Breaks out of a loop.

❓ Quiz #2 Questions



Heartfelt Recap

Why does the heartbeat blink faster when you subtract time? +5 XP

- ✓ A smaller delay in each loop cycle makes a faster blink rate.
- ✗ Negative numbers are always faster than positive ones.
- ✗ Smaller hearts beat faster than larger ones.

Why does your program create an error message when you keep pressing B? +5 XP

- ✗ There is a "divide by zero" error in the `sleep()` function.
- ✗ Too small a delay creates a time vortex.
- ✗ The display can't run that fast.
- ✓ The `delay` variable goes below zero, and `sleep()` can't handle negative numbers.