



<b>MISSION 6: Heartbeat</b>		<b>Time: 45 minutes (mission 6) 90 minutes with extensions</b>
<b>Overview:</b> <p>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 <b>loops</b>. The mission also introduces float values and changing the value of a variable using the counter pattern (incrementing and decrementing).</p> 		<b>Objectives:</b> <ul style="list-style-type: none"> <li>• I can create an infinite loop to make my code more efficient.</li> <li>• I can apply variables to a new program.</li> <li>• I can change the value of a variable using math</li> <li>• I can program using <i>float values</i>.</li> <li>• I can code a “kill switch”</li> </ul>
<b>Standards:</b> <b>2-AP-11, 2-AP-16, 2-AP-19</b> <b>2-CS-03</b> Systematically identify and fix problems with computing devices and their components  <b>2-AP-16</b> Incorporate existing code, media, and libraries into original programs and give attribution.  <b>2-AP-19</b> Document programs in order to make them easier to follow, test, and debug.	<b>CSP Framework:</b> <b>Big Idea 2.1 – DAT-1.A</b> Explain how data can be represented using bits.  <b>Computational Thinking Practices:</b> 4.C Identify and correct errors in algorithms and programs, including error discovery through testing.  6.A Collaborate in the development of solutions.	<b>Key Concepts:</b> <ul style="list-style-type: none"> <li>• Infinite loops – when a <i>condition</i> is always <b>True</b>.</li> <li>• Readability is very important in coding. Whitespace can help, and comments are essential.</li> <li>• Using a <b>variable</b> to hold the <i>current state</i> of your program. In this case it’s the speed of the heartbeat, or to be specific the <i>delay</i> between beats.</li> </ul>
<b>Preparation:</b> <p><b>Make a copy</b> of the assignment or put it in the LMS.  <b>Prepare</b> meaningful notes examples  <b>Decide</b> if you are going to have the students keep a programming journal and how it will be kept (Google doc, assignment, etc.)</p>	<b>Links:</b> <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="#">Meaningful notes slide deck</a></li> <li>• <a href="#">Programming journal</a></li> <li>• Daily reflection form</li> </ul>	<b>Agenda:</b> <ul style="list-style-type: none"> <li>• Warm-up (5 minutes)</li> <li>• Mission 6 (35 minutes)</li> <li>• Extensions (25 minutes)</li> <li>• Meaningful notes (20 minutes)</li> <li>• Wrap-up (5 minutes)</li> </ul>
<b>Vocabulary:</b> <ul style="list-style-type: none"> <li>• <b>Loop:</b> Repeats a block of code, subject to a given condition.</li> <li>• <b>While loop:</b> Repeats a block of indented code as long as the condition is true.</li> <li>• <b>Infinite loop:</b> A loop that never ends because the loop is always true.</li> </ul> <p><i>Not explicitly in the mission instructions, but can be introduced in the wrap-up</i></p> <ul style="list-style-type: none"> <li>• <b>Iteration (AP CSP vocab):</b> The repeating portion of an algorithm; code that repeats until a given condition is met or a specified number of times.</li> <li>• <b>Increment</b> (a counter): Increasing a variable by a specific amount. Often counters are incremented by one: <b>count = count + 1</b> , like a counter, but the value can be any literal number (or constant).</li> <li>• <b>Decrement</b> (a counter): Decreasing a variable by a specific amount. Often counters are decremented by one, like a countdown: <b>count = count - 1</b> , but it can be any literal number or constant.</li> </ul>		
<b>Assessment:</b> <ul style="list-style-type: none"> <li>• Daily reflection journal</li> </ul>		

- [Meaningful notes](#) (or notes to your future forgetful self)
- Programming journal

## Teaching Guide


### Warm-up (5 minutes)

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


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

- Topic: Ask students if they’ve seen flashing road signs. Can they think of other continuously blinking or repeating indicators? How would you make something repeat forever with code?

### Activity – Mission #6 (35 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:**

Review the [Mission Reminders slides](#).

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

Students go to [sims.firialabs.com](https://sims.firialabs.com) and should be at the beginning of Mission 6

 **Teaching tip – Objective 2:**

CodeTrek will remind students about adding the sleep command, but they still need to replace the #TODO with the actual code for displaying a small heart. It doesn’t show the code for that.

 **Teaching tip – Objective 3:**

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/WlTk\\_kpkGiU](https://youtu.be/WlTk_kpkGiU).

 **Teaching tip – Objective 4:**

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.

 **Teaching tip – Objective 7:**

This objective has students use the debugger. You may need to demonstrate this or remind them how to use it.

 **Teaching tip – Extension:**

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

**Teaching tip – Extension #1:**

Earlier in the mission (Objective #6) students learned about a “kill switch” and used Button A. For this extension, they add a “kill switch” using a different button. They can use U, D, L or R.

The code could look like this, but doesn’t have to be the first if statement. Indenting is important! The kill switch must be indented inside the while loop:

```
while True:
    display.show(pics.HEART)
    sleep(delay)
    display.show(pics.HEART_SMALL)
    sleep(delay)

    if buttons.was_pressed(BTN_U):
        break

    if buttons.was_pressed(BTN_A):
        delay = delay + 0.2
```

**Teaching tip – Extension #2:**

Students need to come up with a way to keep the program from crashing. It crashes when Button B is pressed too many times and delay becomes less than 0. There are many ways they can keep the program from crashing. Here are three ways:

```
if buttons.was_pressed(BTN_B):
    if delay > 0.2:
        delay = delay - 0.2
```

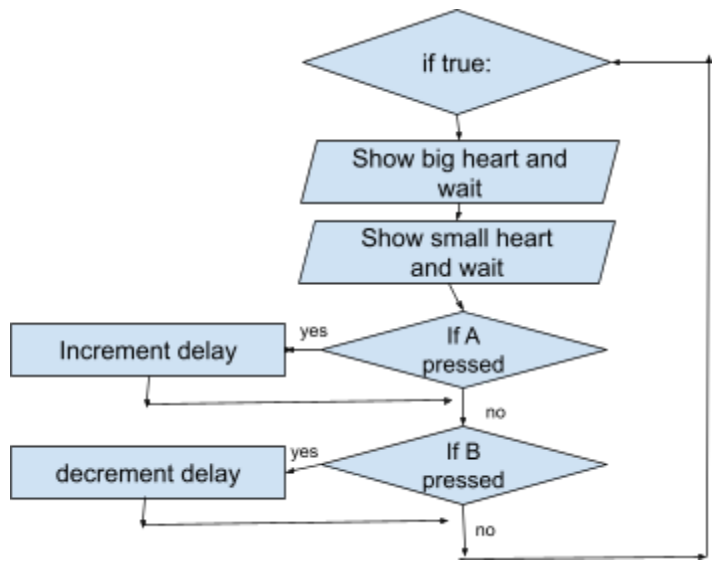
```
if buttons.was_pressed(BTN_B):
    delay = delay - 0.2
    if delay < 0.1:
        delay = 0.2
```

```
if buttons.was_pressed(BTN_B):
    delay = delay - 0.2
    if delay < 0.2:
        delay = 0.2
```

**Teaching tip – Extension #3:**

Students create a flowchart of their final project. You can have the students write the flowchart on a piece of paper, on a white board, or add room on the assignment document and have them complete it digitally.

Flowcharting a loop hasn’t been shown yet. They may be able to figure it out. Or you can specifically teach it or demonstrate to the students that a loop has a condition (starts with a diamond) and uses an arrow to indicate all the code in the loop.




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

✓ To turn in the program, students should download their code (FILE-DOWNLOAD), which will be a text file. Then they should submit their file through Google Classroom or your LMS.

*You may need to demonstrate this to your students*


## Wrap-Up (5 minutes)


 **Vocabulary** – Give students the definitions to these terms. They can add the definitions to their vocabulary canvas, or add them to a programming journal and/or vocabulary document. The first three words are included in the mission instructions:

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

The next two terms are not explicitly in the mission instructions. Students changed the value of a variable (delay) by adding or subtracting a specific number. In coding, this is often referred to as incrementing and decrementing. Students should be familiar with these pseudocode terms.

- **Increment** (a counter): Increasing a variable by a specific amount. Often counters are incremented by one:  $\text{count} = \text{count} + 1$ , like a counter, but the value can be any literal number (or constant).
- **Decrement** (a counter): Decreasing a variable by a specific amount. Often counters are decremented by one, like a countdown:  $\text{count} = \text{count} - 1$ , but it can be any literal number or constant.

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

 **SAY:** You've 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. Ask students to think of real-world applications to what they learned. Some examples include:

- TV Remote Controls (volume + / -)
- Game controllers
- Push button light dimmers

## ✓ IMPORTANT!!

Students should clear their CodeX by running their ClearCodeX program.

If students haven't created a "clear" program yet, they can follow the steps in this [slide deck](#).

## MEANINGFUL NOTES.

If time permits, this is an excellent time to start (or add to) [meaningful notes](#) – or notes to their future forgetful selves (note making). This can be done in groups of three at white boards, or however you choose for the most student engagement. I don't recommend leaving this as homework (lower engagement and less participation).

If you have students work collaboratively, standing at white boards, you can take pictures of their notes and assemble them into a shared Google Doc for the students to use and reference as needed.

Also, if you have time, students can return to their seats and write their own notes. I suggest having them do



only Quadrant C and D (their own example and things to remember). All groups are doing Quadrant A and B, and it should be the same for everyone. If you are taking pictures of their notes, you can pick the best A and B to post, and then everyone else's C and D.

You can refer to the slide deck. This information should be given to the students the first time you do meaningful notes. You can use the slide deck, or just present the information in the [slide deck](#) to the group.

#### Formative Assessment:

- Daily reflection journal or [Google form](#)
- Completion of assignment and/or mission
- Exit ticket on vocabulary (many to choose from)
- Group review on vocabulary (many to choose from)
- Students create a vocabulary canvas with vocabulary words (pick just a few).

This concept is adapted from Code.org AP CSP curriculum. The [vocabulary canvas slide deck](#) is created by Code.org.

#### SUCCESS CRITERIA:

- Create a debugging table that documents your debugging process
- 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
- Include a "kill switch" to stop the program
- Add code so the program doesn't crash
- Create a flowchart for the program