HyperionDev

Mentor Call Training Reflections Session 1

Name:

Henri Branken

Session 1

Notes

The Steps for Debugging

[1] Describe the Problem:

- What am I **expecting** the code to do?
  VS
  What is the code **actually** doing?
- **Narrow down** what we are trying to fix.

[2] Bug Hunt:

- See how the code is **behaving**.
- Create a **hypothesis** about what is **causing the issues**.

[3] Test the Hypothesis:

- Make **targeted changes** to the code → **Observe any changes** in the behaviour.
- If there isn't a desirable change, **move on** to a different hypothesis.

[4] Document what you have learned:

- **Why** did the code behave the way it did?
- **How** was the bug resolved?
- **Refer** to this documentation when faced with a similar situation.

How to get Visibility into How the Code is Behaving

[1] Divide and Conquer:

- Run code in **isolation**.
- **Reduce the amount of complexity** you need to focus on.
- **Comment out code** that is not relevant to the error you are facing.

[2] Use outputs to the Terminal:

- **`print`**, **`console.log`** statements
- Do the sections of code return values that are expected?
- Do variables hold values you expect them to hold?

[3] Use a REPL: Read-Eval-Print-Loop:

- **Test** your assumption about how the code actually behaves.
- **Python**: Python Shell
- **Ruby**: IRB (Interactive Ruby)
- **JavaScript**: Node.js

[4] Use a Debugger in your IDE:

- Very useful tool for **complex** debugging
- Introduce this to students at **~ Level 2**.

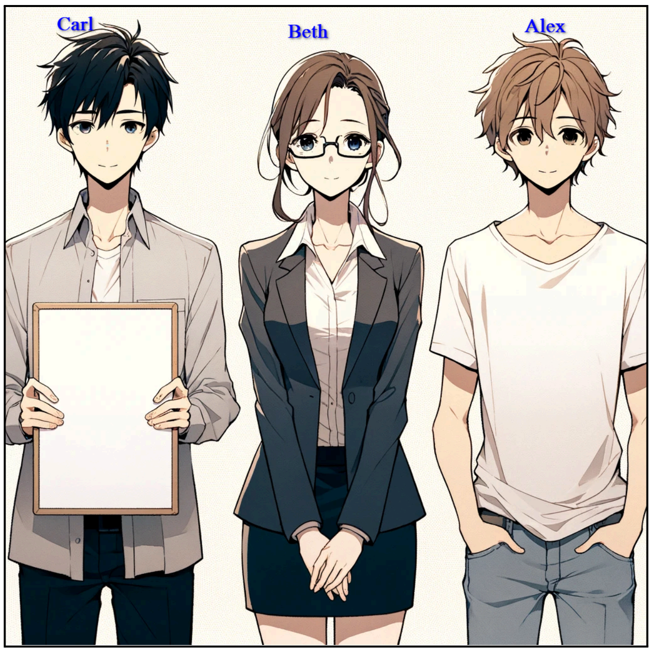
Reflect and Apply

*A student is struggling with understanding **how loops work** in the context of first taking an input that tells the program **how many names are going to be entered** (**x number of names**), then **taking the name inputs** x number of times as stipulated and then **printing out all of the names in the console**. Given what you have read in the paper provided on Basecamp about applying the **Socratic method** to teaching code, **create a scenario** of how you could approach assisting the student to get a **better understanding of how to use loops**. Feel free to be as **creative** as possible and **add justifications** as to **why you are taking the approach that you are taking**.*

Answer

The English Version:

There are 3 people in this play (as visualised in the picture below): Carl, Beth, and Alex.



Imagine the following script:

1. Carl asks Alex (by means of a White Board) how many names he has in mind.
2. Alex reads Carl's question, and tells Beth he has 4 names in mind. (We are simply using "4" for illustration purposes.)
    a. Beth keeps the counter/number 4 in her head.
3. Repeat from 1 to 4 (i.e. repeat 4 times):
    a. Beth asks Alex to give her a name.
    b. Alex gives a name to Beth.
    c. Beth asks Carl to:
        i. Clear his white board if something was written on it in the first place.
        ii. Write the name Alex gave her onto the White Board and display it to the rest of the class.

In this **Analogy**, we can establish the following associations:

- "Alex" is the user providing input to the computer.
- "Beth" is the computer with the ability to store values in memory locations.
    - She is also the middleman/interface between the user (Alex) and screen (Carl).
- "Carl" is a peripheral screen (or console) that provides information to the outside world based on Beth's instructions.

The facilitator asks for 3 students to audition for this script. He then hands them the script and gives them some time to come up with a play to enact in front of the entire class.

The facilitator asks the actors to perform their play.

After finishing their skit, the facilitator provides additional information to the class about what each person in the play represents, i.e.: Console/Screen, Computer/Memory, User.

Finally, the facilitator asks the actors to do their skit again from start to finish, this time also prompting the students in the audience to try and link up the actions in the skit with the coding challenge they are trying to solve.

<u>Motivation</u>:

This type of delivery is feasible because there are no complex problems involved in this loop challenge. Furthermore, this skit develops a mental model of for-loops in the minds of students learning this programming concept.

The Pseudocode Version:

1. Ask the user how many names are going to be entered.
2. Declare an integer variable `` `n` ``.
   Store the number of repetitions (or names) in the variable `` `n` ``.
3. Repeat the bullets below `` `n` `` amount of times:
   a. Ask the user to enter a name.
   b. Store the name value in the `` `name` `` variable.
   c. Print out the value of `` `name` `` to the console.

<u>Side-Note</u>:

We know beforehand how many times we must iterate - in this case `` `n` ``. Therefore, based on theory the student should know at this point, we must use a **`for`**-loop.
