REVIEW FEEDBACK

Harry Turnbull 06/05

06 May 2020 / 12:00 PM / Reviewer:

Steady - You credibly demonstrated this in the session.
Improving - You did not credibly demonstrate this yet.

GENERAL FEEDBACK

Feedback: This was a great review - especially well done since this was your first review and you are already steady across most of the course goals. I would say that the most important aspect to work on in your process is to continuously evolve your code, rather than removing it all and introducing new code each time that makes your tests pass.

I CAN TDD ANYTHING – Improving

Feedback: For the most part, your TDD process was really good. You based your tests on overall input-output behaviour and stuck to the excellent test progression you had laid out in your readme. There were a few tweaks that would make your process even better here:

1. To get your third test to pass, you refactored your code to simply append ": 1" to the input string. If you look at about 32 mins into the video, you will see that the if-statements you have there are very similar to the ones you were developing in response to your first tests, before you refactored them away for the third test. I would encourage you to address all the simplest examples first through if-statements (if there are a reasonable number of 'simplest' cases) and then only refactor these away later when you are sure you won't need them. Recall that on the green step of a red-green-refactor cycle, the rule for green is to do the simplest thing possible to get the test to pass.

2. After both your 3rd and 4th tests, you completely removed all the code you had written before, and effectively started a new solution from scratch. This

means that the code you put effort into developing before is wasted - you may as well have written all the tests up to this point all at once. To improve here, try to focus on evolving or upgrading the code you already have in an iterative process from test to test. This means that your tests are always driving your code, and you are likely to be able to use the key structures (such as if-statements) which your earlier tests have developed, rather than having to rethink through the logic for the entire program each time.

3. This third point is a symptom of the previous one. To get your 4th test to pass (starting with no code) you had to introduce approximately 20 lines of code, implementing a full counting solution for multiple greens, but also ambers and reds. Adding this much code at once, a near-complete solution is a symptom of something going wrong with your red-green-refactor cycle (in this case, removing all your code each time/refactoring away if statements too soon) and as a result, you end up with a really long green step, with many opportunities for bugs to come along. On a green step, you should only be adding code to upgrade your existing code to cope with the new test case, eg adding a loop and introducing count variables and incrementation. Taking the fact you started from scratch at the beginning of this test, if you try to follow the rule for the green part of the cycle here, you would probably do something like hard-coding with some if-statements (ie, reverting back to earlier tests code) or splitting the array and checking its length...however this wouldn't be very productive (and I am actually glad you didn't do this) because doing the simplest thing on the green step only makes sense if you've been evolving your existing code iteratively between tests. I hope I have explained this here better than I did in the session!

I CAN PROGRAM FLUENTLY – Strong

Feedback: You seem to have good programming skills. You were very familiar with Ruby and used Rubocop as a guide to adhere to best practices. You were familiar with the syntax for if-else statements and loops, array manipulation, hashes and some of Ruby's in-built functions, such as split. You developed a nice and clean solution and through several refactors were working towards making the code as succinct as possible.

I CAN DEBUG ANYTHING – Steady

Feedback:

I CAN MODEL ANYTHING – Steady

Feedback: You modelled your solution in a single method which I felt was a nice and simple implementation and provided a good place to start.

I CAN REFACTOR ANYTHING –Steady

Feedback: You looked for some refactors after getting the 4th test to pass. I really liked that you chose to create a helper method to clean up the input string, refactoring out the trimming from your loop. You also refactored the count incrementation to make use of a hash, and remove the if-statements. These were good refactors.

I HAVE A METHODICAL APPROACH TO SOLVING PROBLEMS – Steady

Feedback: You were very thorough in your information gathering and laid out really solid test progression. You also methodically checked off tests in your readme when the relevant test passed. You red-green-refactor cycle was punctuated with commits to git, along with a highly methodical commit message, indicating whether it was a red or green test as well as a useful message denoting the feature.

I USE AN AGILE DEVELOPMENT PROCESS – Strong

Feedback: You conducted a really excellent information gathering session, which was extremely thorough. You started by asking questions about the input format and asked some clarifying questions about what was expected as the output. It was great that you asked the client about how they wished to interact with the code. It was also really good that you considered that the input might not be perfect, so there could be edge cases to address. We

discussed several aspects of format and strange input and how they should be addressed.

Furthermore, your use of an input-output table was excellent and allowed you to clarify a finer point of operation as to the output format. You used this table to great effect to plan your tests.

I WRITE CODE THAT IS EASY TO CHANGE – Strong

Feedback: I was very happy with your Git usage. You initialised Git early on, committed on both the red and green steps of the red-green-refactor cycle and supplied useful commit messages, making it easy for future developers to come in and change a feature.

I was pleased that you had your test suite properly decoupled from your implementation by making sure the tests were based solely on acceptance criteria, and not reliant on the current implementation. This makes changes to the code much easier.

I CAN JUSTIFY THE WAY I WORK – Strong

Feedback: It was easy to follow your process. You highlighted each step you took, what you were testing, let me know why errors were occurring, when you were expecting them and how you addressed them. You also noted when you were looking for refactors, which was great, as it highlighted that you were following a red-green-refactor cycle. You made good justifications as to your decisions.