REVIEW FEEDBACK

Oliver Smith 12/10

12 October 2020 / 12:00 PM / Reviewer: Katherine James

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

GENERAL FEEDBACK

Feedback: This was a great review and you developed a solid solution to the task, justifying your decisions extremely well at each step. You demonstrated notable improvement in your test progression, which was nice and iterative and introduced the perfect amount of complexity to the code in response to each test.

To refine further I would suggest looking out for refactors where you generalise against a particular input case (eg. after your second test for in-range). Also, try to really punctuate your red-green-refactor cycle with commits when tests pass.

Overall, really nice work - keep it up!

I CAN TDD ANYTHING – Steady

Feedback: You demonstrated an excellent TDD process today. You were nicely familiar with the testing syntax and were able to verify syntax for error raising using good online resources. You tested at the interface of the program, which means that your tests were nicely decoupled from your code, which leaves your implementation nice and flexible.

You identified the simplest case to start with (in range, so no modification needed) and the simplest example of this (a single element in the array). From here you consistently tested the next simplest example which broke the assumptions of your current code, progressing through a different example for

in-range, through a value above the max, below the min and then onto multiple frequencies and edge cases. This is exactly the right approach to take.

You also ensured that you introduced an appropriate level of complexity to the code in response to each test, starting with simple if-statements and working within the assumptions made by your current test suite. This was great to see.

I CAN PROGRAM FLUENTLY – Steady

Feedback: You conceptualised the steps you would need to take early on, but allowed the tests to drive iterative changes in the code. You seemed to be nicely familiar with programming in Ruby and smoothly produced code to meet the goal set by the current test. You were familiar with the syntax necessary to complete this task and demonstrated familiarity with array manipulation. You also demonstrated familiarity with iteration options and refactored to map, which was an excellent selection for this task. You also demonstrated familiarity with default values.

I CAN DEBUG ANYTHING – Steady

Feedback: You demonstrated familiarity with the common error types and took care to understand why you received the error messages you did on failing tests. Your debugging process was good and you were able to interpret the error messages you received in the stack trace on unexpected errors, used the output of failing tests for guidance on how your program needed to change and were able to step logically through your code to identify errors.

I CAN MODEL ANYTHING – Steady

Feedback: You considered the high-level behaviour of what your program should do and from this concluded that a single method would be suitable. This was a nice and simple implementation and provided a good place to start as greater complexity was not needed. You named your method restrict_frequencies which was a suitable name and conformed to case conventions for Ruby. You developed

a logical algorithm to solve this task and took care to consider where your guard clause should be placed to ensure that the implementation was efficient.

I CAN REFACTOR ANYTHING –Steady

Feedback: You looked for refactors in the correct point of your cycle and justified doing/not doing reactors. You executed a refactor after completing the core logic to reduce the number of variables, switching to map instead of each. This was a good refactor.

Just note that you could have refactored to just return the array for the in-range case, after you second test. This would have prevented the confusion you ran into with the hard-coding. You had to do this refactor on your 4th test on a green step as a result of not doing it at this point.

I HAVE A METHODICAL APPROACH TO SOLVING PROBLEMS – Steady

Feedback: Overall your process was methodical. In the session today you made use of a regular RGR cycle, running RSpec regularly to see error messages changing and also introducing the appropriate level of complexity into the code for each test. You caught yourself when you added code before tests and stepped back. Your tests progressed in a logical order which resulted in iterative development of the solution. You prioritised core cases over edge cases to provide immediate value to the user, although I might have encouraged you to switch the order of the defaults and invalid input cases, as the default case is a little more on the ‘happy path’ than the invalid input case.

I USE AN AGILE DEVELOPMENT PROCESS – Steady

Feedback: This was a well-conducted information gathering session. Your questions progressed in a logical order and you immediately seemed to have an idea of the various components that would be supplied as input, but took care to verify these assumptions. You described the output to verify that you were on the same page as the client, and this led to a clarifying discussion about the expected behaviour. You considered edge cases by asking about invalid

input and took care to determine how the client would like you to handle this. You also asked for example to verify the behaviour of the program.

I WRITE CODE THAT IS EASY TO CHANGE – Steady

Feedback: Your code was written so that it would be easy to change and maintain. 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. You also utilised Git as part of your process and committed at some key points during the review. Ideally, you should try to really punctuate your red-green-refactor cycle with commits after the green and refactor steps. This ensures that the latest working version of your code is always backed up. Also note that commit messages should start with a capital letter at the beginning, like a sentence.

I CAN JUSTIFY THE WAY I WORK – Strong

Feedback: Your vocalisation was excellent, each decision was carefully substantiated and your process was very easy to follow as a result. You justified your selection of a method over a class nicely. You highlighted your selection of example to test next, indicated options you had in terms of this and let me know why tests failed given your existing implementation. You indicated what you would need to add to your code to make tests pass and highlighted the achievements of the current code at key points. Throughout the review, you gave insights into your decisions regarding the implementation.