REVIEW FEEDBACK

Ben Gittins 21/09

21 September 2020 / 03:00 PM / Reviewer: Katherine James

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

GENERAL FEEDBACK

Feedback: It was really great to see how conscious your process has become, it would say that it is solid throughout and will serve you well when tackling bigger tasks. It has been a pleasure reviewing your process and observing how you have actioned the feedback. You have good intuition now into how to approach tasks such as this and I hope that you will continue with katas regularly to keep this talent sharp. I wish you every success in the future.

I CAN TDD ANYTHING – Strong

Feedback: You have a solid grasp of how to approach tasks using TDD and also of consciously selecting an appropriate test progression. You approached the task by focusing on each type of input individually, before moving onto a test that tied them together, using gradually complexifying acceptance criteria for each type of behaviour. This was cleanly executed and resulted in the solution developing iteratively, with each behaviour tested methodically so that unexpected behaviour did not slip past untested.

I CAN PROGRAM FLUENTLY – Strong

Feedback: You are nicely fluent with programming in Python. You have a good grasp of list comprehension and also comfortably used for loops, if-else statements and string interpolation as part of your solution. You were familiar with primitive type conversion and were familiar with functions for conversion between string and array types (split, join) and needed only to confirm the syntax of the latter. This was a good solution to the task.

I CAN DEBUG ANYTHING – Steady

Feedback: Your debugging process was really steady and efficiently got your through a number of bugs in the session. You demonstrated proficiency with using the information in the stack trace to identify the error type and reason for it and were then able to take a logical action to resolve the error. You used prints to get visibility into the code, followed your logic through and used onliner resources when necessary.

I CAN MODEL ANYTHING – Steady

Feedback: You started by exploring the task using an input-output table and from this determined that a single method would be appropriate for this task. This was a nice and simple implementation and appropriate for this task. You named your method grade_checker. This was quite a good name, but it described what the method was (if it was an object), but not what it did. There is a subtle distinction here. Class names should describe what they are (eg. GradeChecker), method names should describe what they do (ie. check_grades). By using the verb form of the word, you can thus make the method name actionable.

I CAN REFACTOR ANYTHING –Steady

Feedback: Your refactoring process is solid. You looked for refactors at the right time in your red-green-refactor cycle and introduced changes which improved the code for readability (eg: extracting logic out from the string interpolation to its own variable and reducing the if-statements using list comprehension). You made sure to do one refactor at a time, which is a nice and methodical approach, ensuring that bugs are isolated easily.

I HAVE A METHODICAL APPROACH TO SOLVING PROBLEMS – Steady

Feedback: On the whole, your process is methodical. You use a regular red-green-refactor cycle, though should be careful to check that tests do fail as you expect. You were familiar with checking the inverse of this, ensuring that tests passed in the correct way. Your tests progressed in a logical and justified order and you prioritised simple tasks over complex tasks and core cases over edge cases to ensure value to the user along the shortest path possible. When you felt that you had taken to big a step in the implementation, you were able to step back and re-approach the step. You also regularly re-orientated against your notes.

I USE AN AGILE DEVELOPMENT PROCESS – Steady

Feedback: You efficiently gathered the requirements for this task, starting by confirming the high-level requirements and then asking for an example of the input and output. This allowed you to quickly identify the types and format of the input and output, and also gave you insight into the types of transformations that would need to take place. This also gave you a good basis for further questions, such as confirming the range of the numbers which mapped to the particular colours. You considered an edge case, where the format of the input might vary (asking about commas and spaces). You could also have considered variations in the capitalisation of grades here, and how to handle anything that was not of the allowed set of grades. You made use of an input-output (IO) table to record some examples, these served to confirm the output expectations and implicitly confirmed assumptions such as zero counts not being present in the output. These examples also formed a solid basis for your tests.

I WRITE CODE THAT IS EASY TO CHANGE – Steady

Feedback: Your code was written in a way that would make it relatively easy to change. You used Git as source control for your project and committed regularly. There is some scope for improving your commit messages, try to give a little more context in them, such that they give context to the change made.

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 chose variable names which reflected what they represented, thus making the code readable. Potentially, a further step you could take for the sake of changeability would be to turn your magic numbers into constants.

I CAN JUSTIFY THE WAY I WORK – Strong

Feedback: Your vocalisation was excellent and you provided reasons for each decision you made. This included how you would approach the task with your TDD, your test progression, your choice of a single method for the task and the steps you would have to take as a result of the increasing complexity of new tests. This underscores how conscious your process has become and also illustrates to a reviewer that you fully understand the implications of your choices. Your vocalisation also made it easy to follow your progress through the task, as you gave updates as to what the code did.