1. What is your favourite SOLID principle, and why?
2. What are the pros and cons of test-first vs test-second development?
3. For your favourite language, tell me about a new (or upcoming) language feature that has you excited. Why is it exciting to you?
4. What do you hate to see when you're reviewing code?
5. Tell me about a time you fixed a performance issue.
6. Last question: Can you please provide me with an implementation to a palindrome-checker, as described here: [http://pastebin.com/ehSuc1f3](http://workable.com/nr?l=http%3A%2F%2Fpastebin.com%2FehSuc1f3)
7. Single Responsibility. It ensures functions or classes are focused on just doing one thing. It makes the code predictable as you wont have any surprises. If a method says GetCustomers, Single responsibility principle dictates that it does just that and not update states of the customer.

|  |  |
| --- | --- |
| Test-First | CON: Has high learning curve |
| Test-First | CON Needs more time to build than test-second |
| Test-First | PRO: Lowers Maintenance cost |
| Test-First | PRO: Elevates productivity |
| Test-Second | PRO: Code size is relatively smaller |
| Test-Second | PRO: Simplicity |
| Test-Second | CON: Less coverage than Test-First |

1. C# - No particular new feature but I really liked how it evolved over the years.
2. Functions that have multiple responsibilities and poorly named variables.
3. It was when we had to switch a highly synchronized process into async. This immediately has a real effect on memory as resources are being released on the web server while awaiting result.
4. You can find my answers at: <https://github.com/cythecy1/pushpaytest>