Testing by Developers: Why and When

Emily Bache

Technical Coach

@emilybache | coding-is-like-cooking.info

Overview

Why should you write Tests?

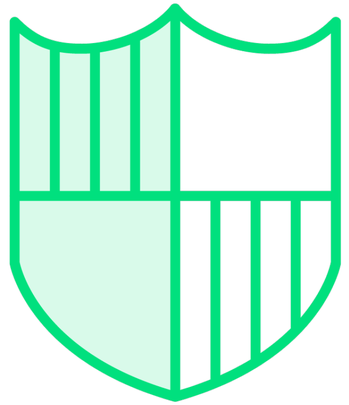
Test Driven Development

Continuous Integration

Why You Write Automated Tests

Two most important reasons:

Ensure the code does what *you* think it does



Protection from future regression

Demo
Feedback Loop

Developing a Yatzi Scorer

Ensure the Code Does What You Think It Does

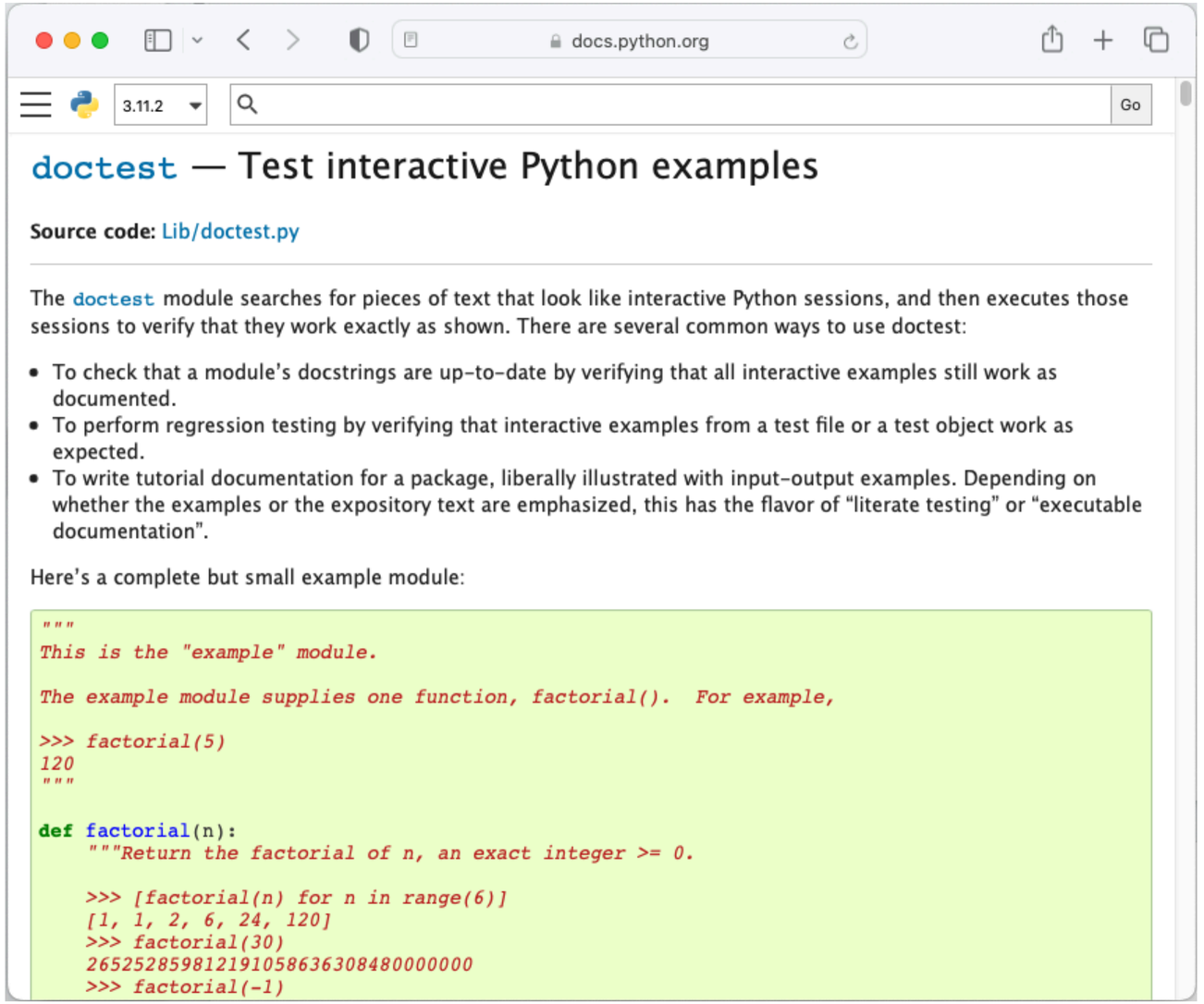
As a complement to:

Python Console (REPL)

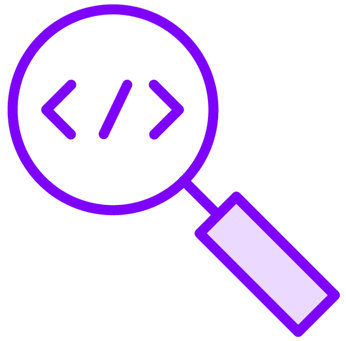
Step through it in a debugger

Run it as a user would

doctest: For Developers Who Like to Use the Console



Writing Unit Tests While You Develop



Helps you understand what your code does

Quickly pays back the time invested

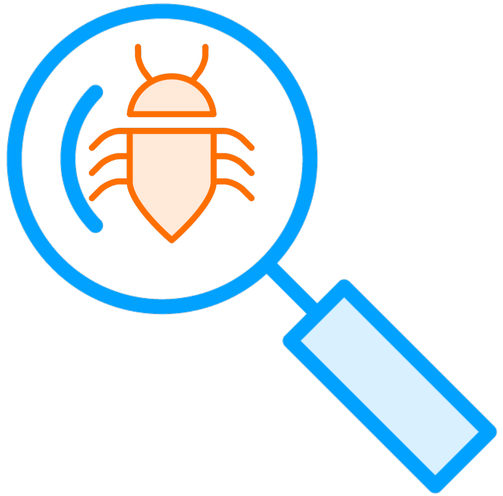
Regression Testing

Ensure previously developed and tested software still performs after a change

https://en.wikipedia.org/wiki/Regression_testing

The Two Most Important Jobs of a Regression Test

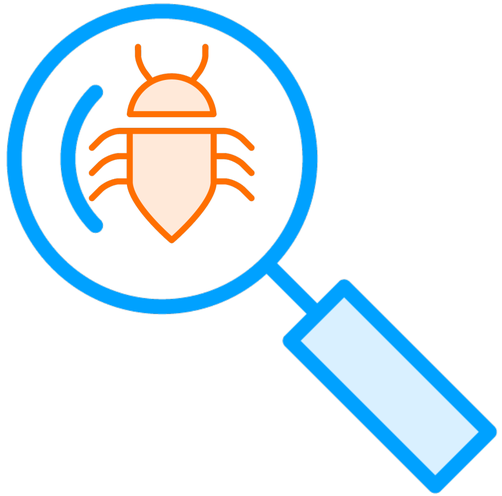
The Two Most Important Jobs of a Regression Test



Fail When There Is a Problem

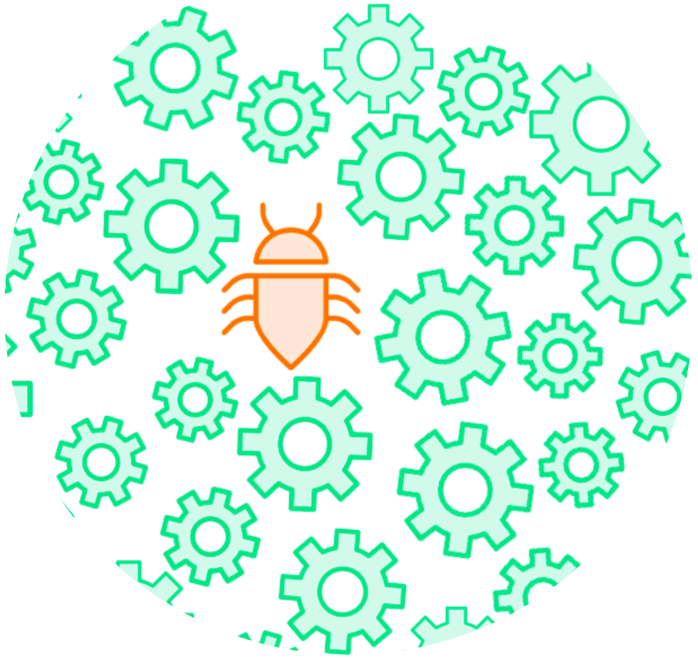
And not otherwise

The Two Most Important Jobs of a Regression Test



Fail When There Is a Problem

And not otherwise



Describe the Problem

And preferably where in the code to look

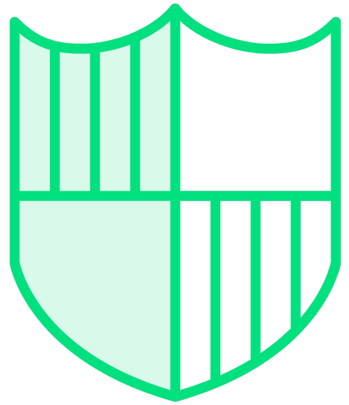
Demo
Protection from Regression

Pinpointing failure in a Yatzi Scorer

Why You Write Automated Tests

Two most important reasons:

Ensure the code does what *you* think it does



Protection from future regression

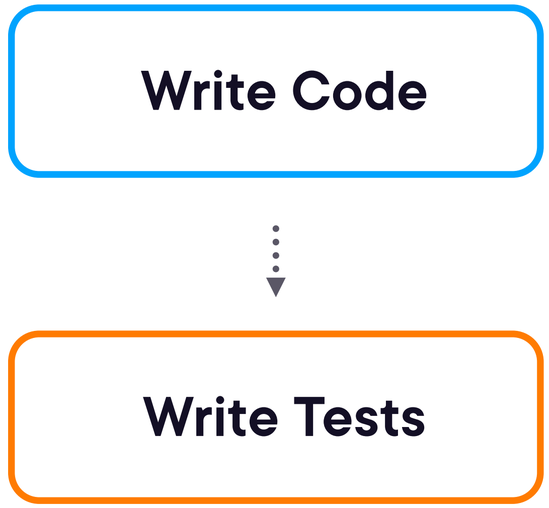
Unit Testing Is Part of Your Job

Units of code

Automated unit tests

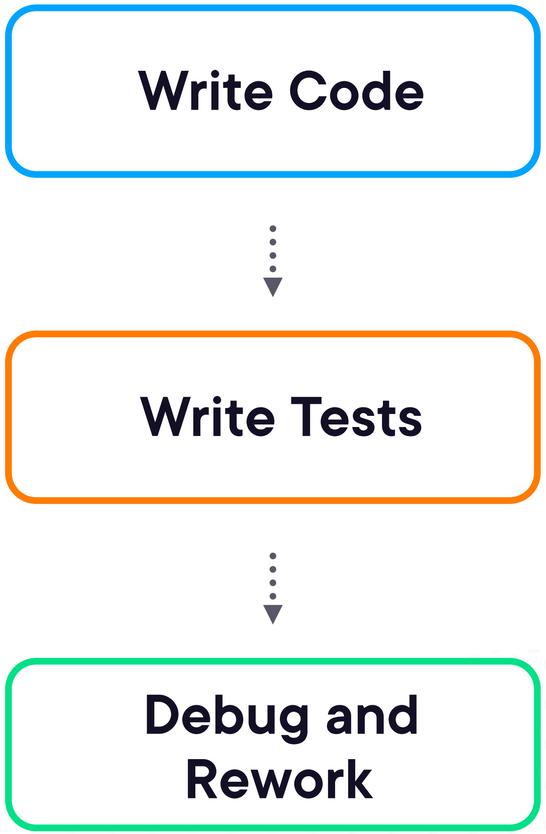
‘Test After’ Process

- Test comes after code
- Test when design is stable



‘Test After’ Process

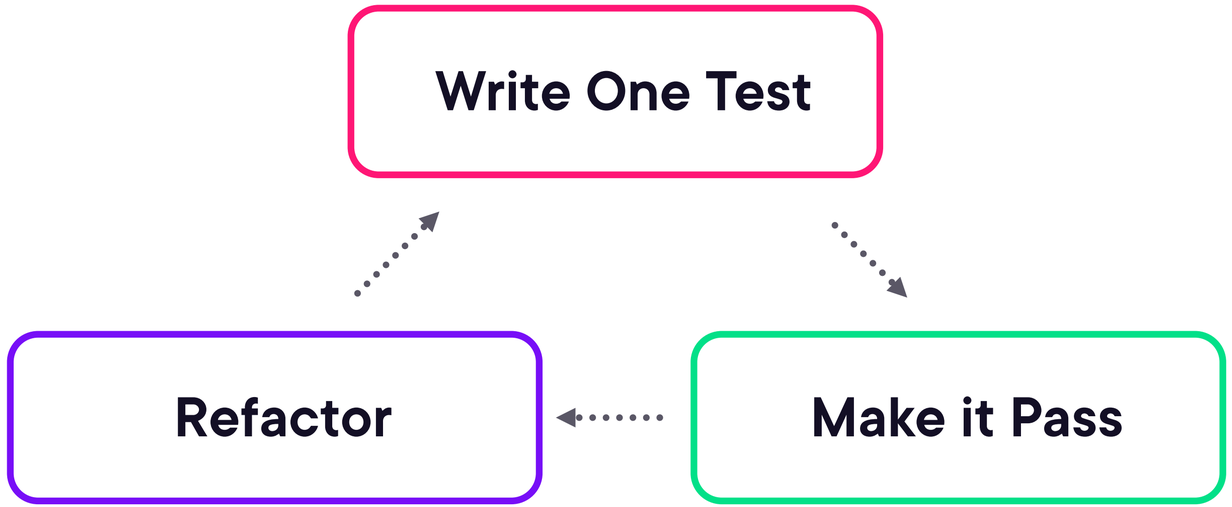
- Test comes after code
- Test when design is stable
- Discover bugs late
- Tests are rushed



Test Driven Development

Test Driven Development

- The tests drive the process
- Unfamiliar way of working
- Takes discipline and skill

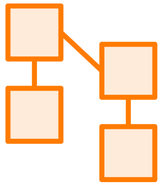


Demo
Test Driven Development

FizzBuzz

TDD Affects your Design

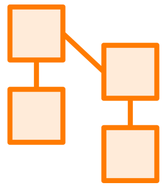
TDD Affects your Design



Decompose into Testable Units

Loose Coupling & High Cohesion more likely

TDD Affects your Design



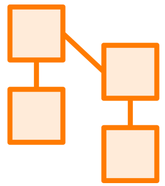
Decompose into Testable Units

Loose Coupling & High Cohesion more likely

Design Interface and Implementation Separately

By ‘interface’ I mean function name, parameter list, return type

TDD Affects your Design

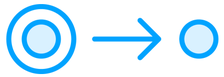


Decompose into Testable Units

Loose Coupling & High Cohesion more likely

Design Interface and Implementation Separately

By ‘interface’ I mean function name, parameter list, return type



Frequent Opportunities to Refactor

Improve function names, move code around, increase readability

TDD Is Not Widespread

2015 research study "When, how, and why developers (do not) test in their IDEs" by Beller et al

"The majority of developers in our study does not test; developers rarely run their tests in the IDE; Test-Driven Development (TDD) is not widely practiced"

How People Learn TDD

Practice on Code Katas

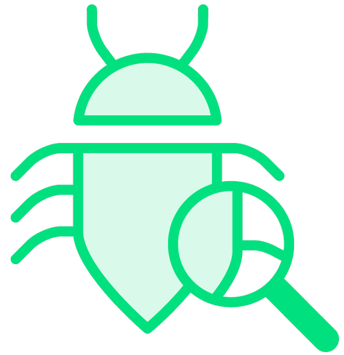
Pair program with someone who knows it

Get a technical coach for your team

Continuous Integration

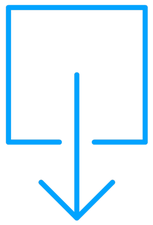
Share Your Tests with Other Developers

Document how to use code



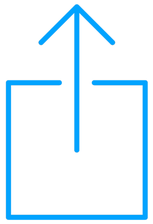
Protect from regression

Test Support Collaboration



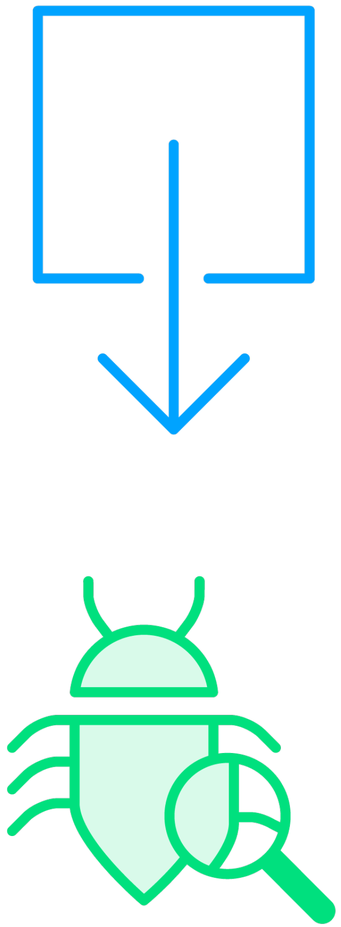
Pull changes from version control

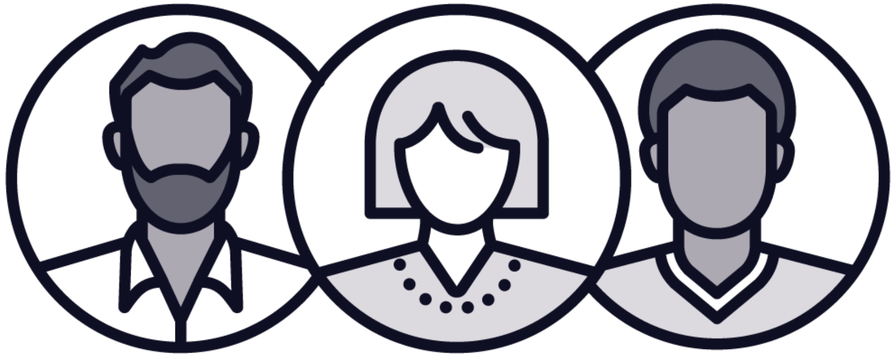
Run the tests & check they pass before changing code



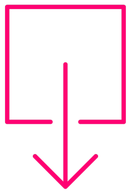
Run the tests again before sharing your changes

People Aren’t Perfect





Build Automation Server



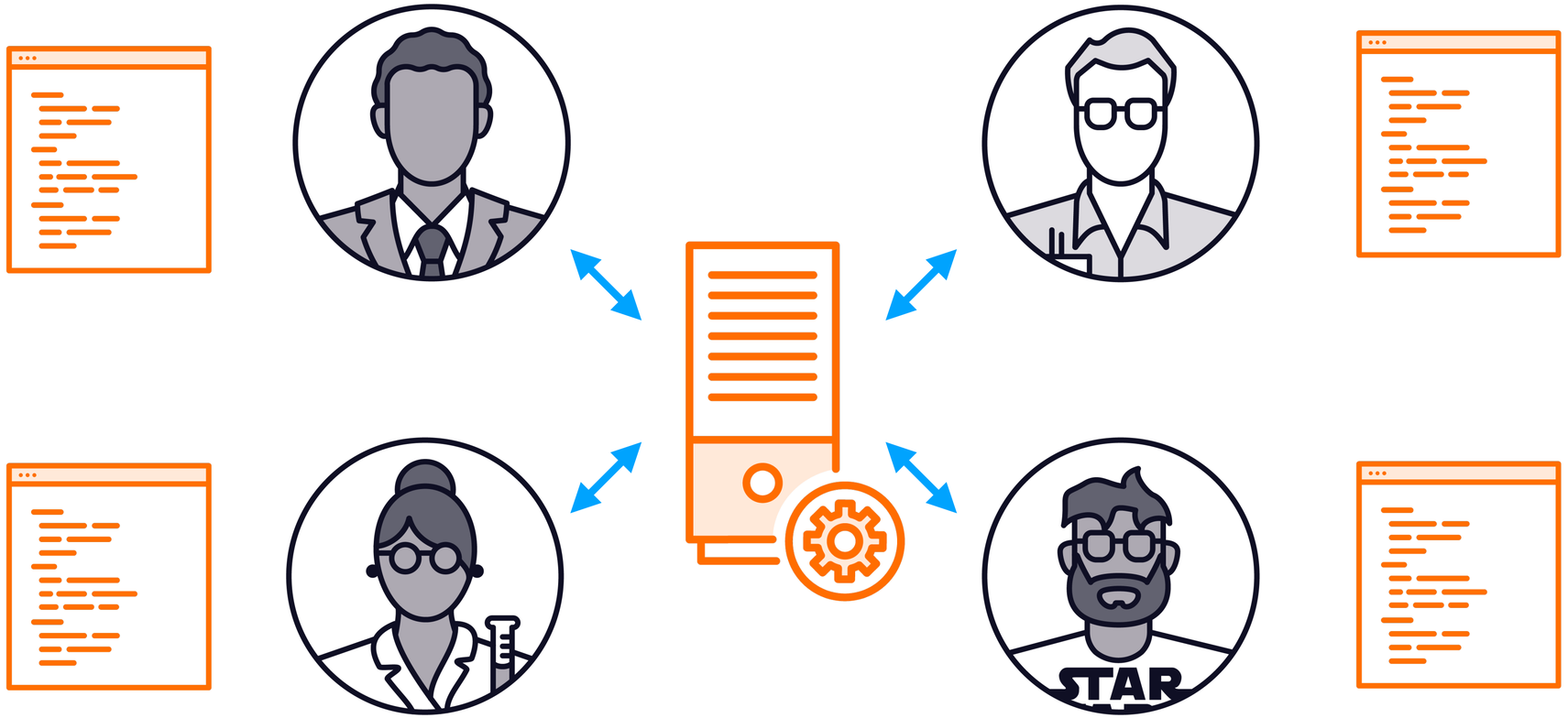
Detects changes from version control, fetches them

Builds the code & runs the tests

Communicates the result to developers

If passing tests - deploys to manual testing environment

Continuous Integration



http://www.martinfowler.com/articles/continuousIntegration.html

Summary

Why should you write Tests?

Test Driven Development

Continuous Integration