TDD*

Alex Kavanagh
@ajkavanagh

*Test Driven Development

Why?

“Programming is like sex …
one mistake and you have to support it
for the rest of your life.”

–Michael Sinz

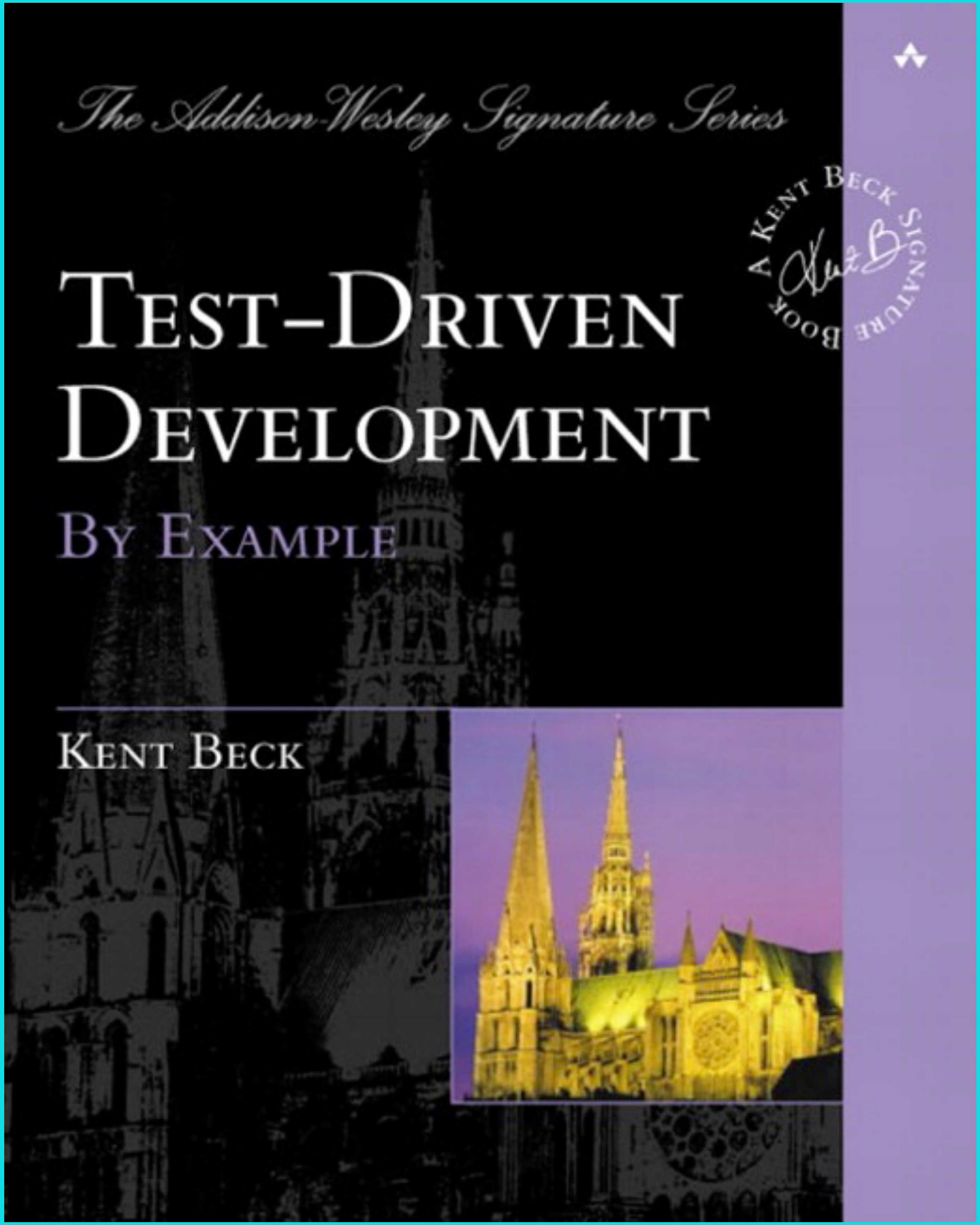
“The problem with quick and dirty
is that
Dirty
remains long
after the **quick** has been
forgotten.”

–Steve M. McConnell

“Always code as if the guy
who ends up maintaining your code
will be a
violent psychopath
who knows where you live.”

–John Woods

What *is* TDD?



The Addison-Wesley Signature Series

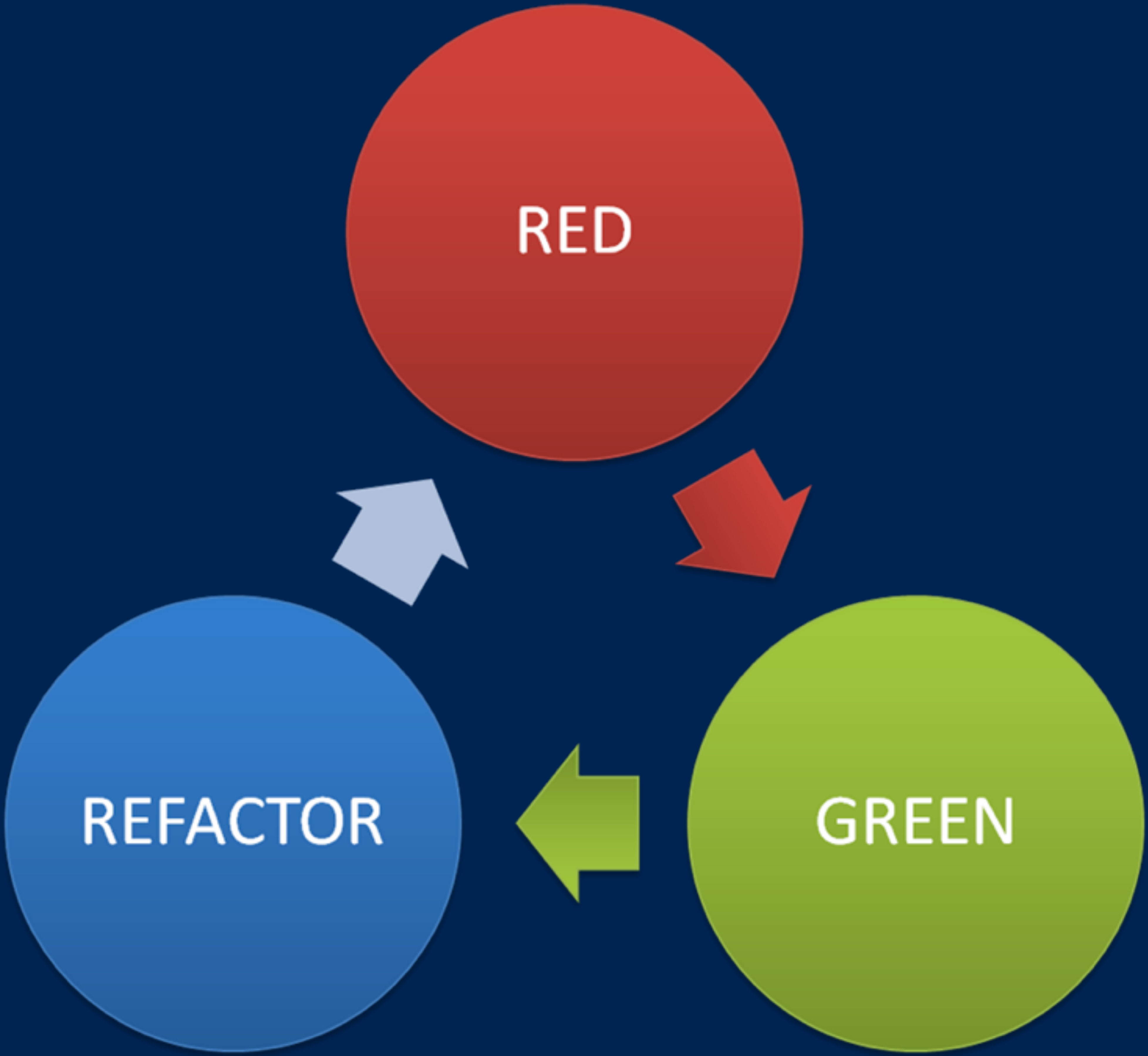
A Kent Beck Signature Book

Kent B

Test-Driven Development

By Example

Kent Beck

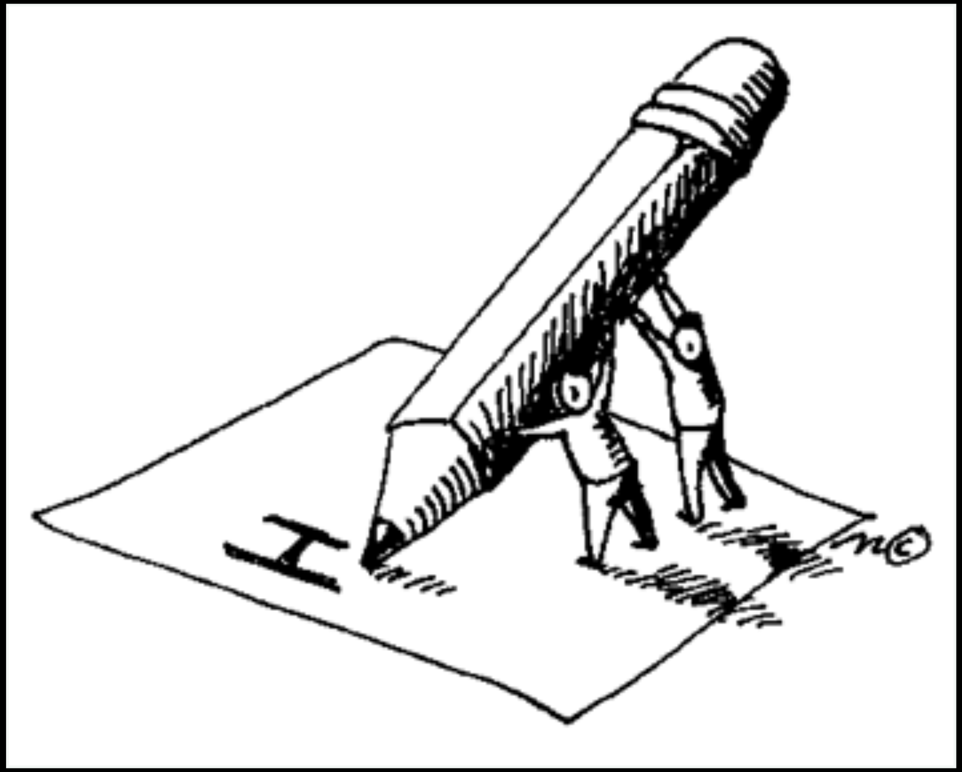


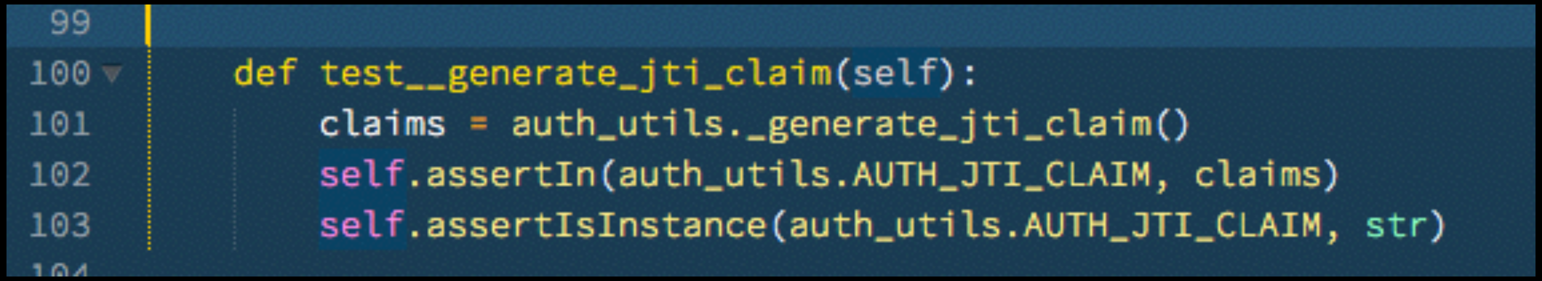
RED

GREEN

REFACTOR

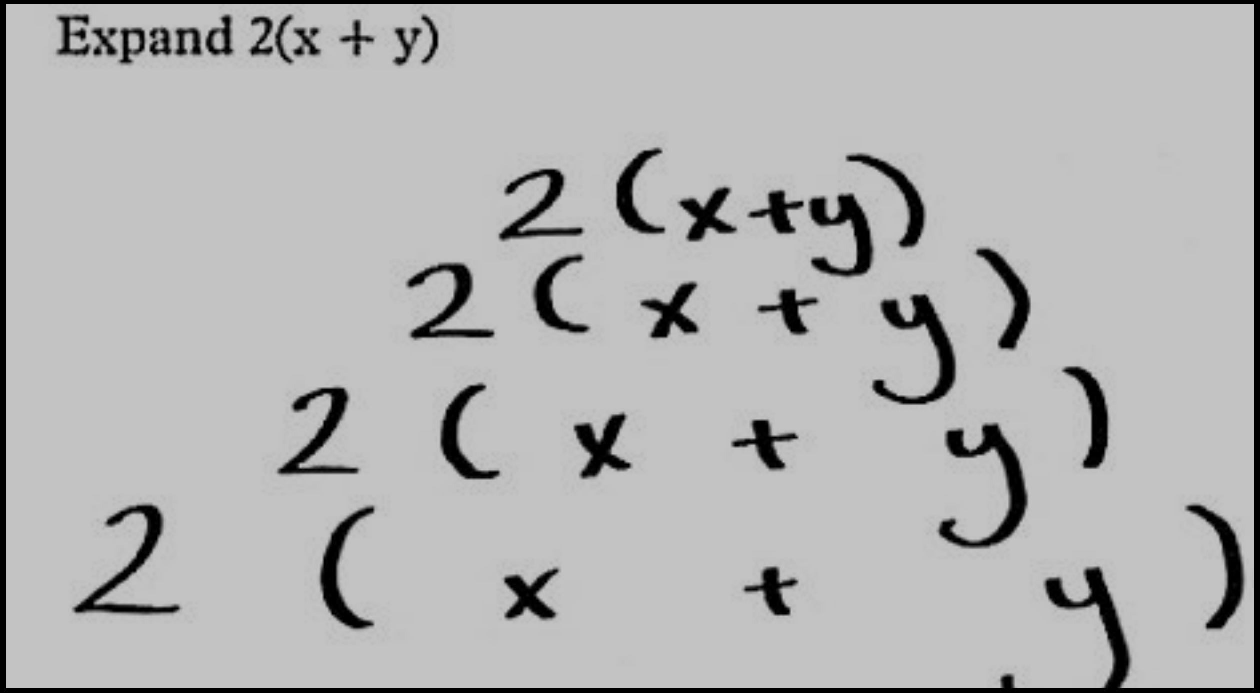
Write Test





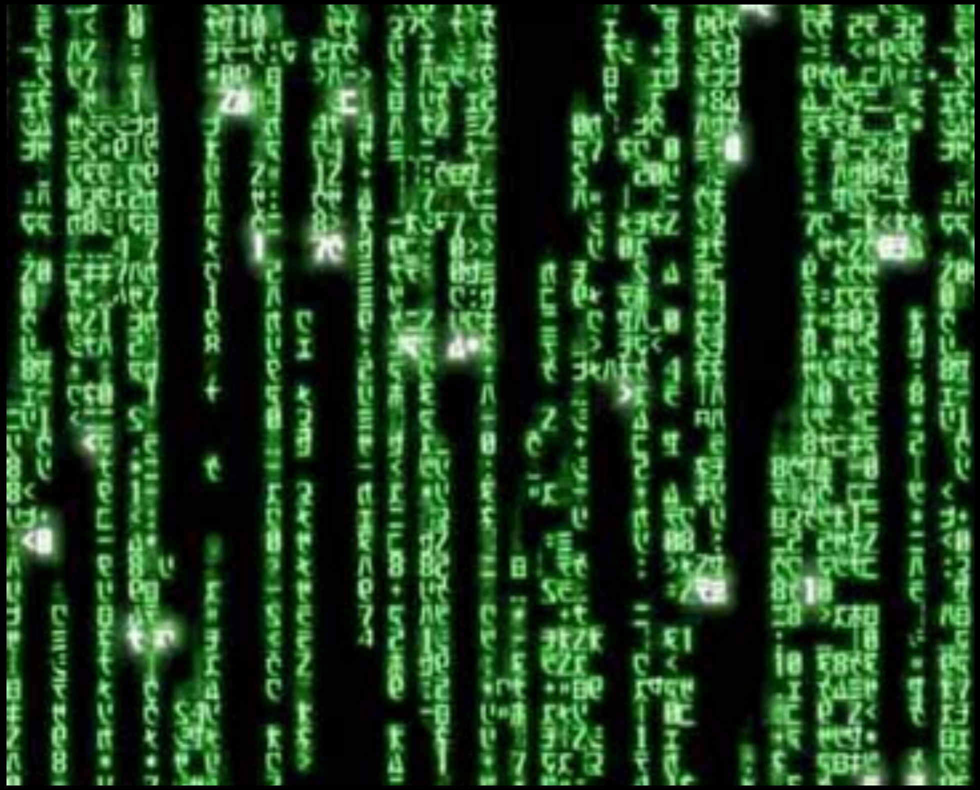
```
    def test__generate_jti_claim(self):
        claims = auth_utils._generate_jti_claim()
        self.assertIn(auth_utils.AUTH_JTI_CLAIM, claims)
        self.assertIsInstance(auth_utils.AUTH_JTI_CLAIM, str)
```

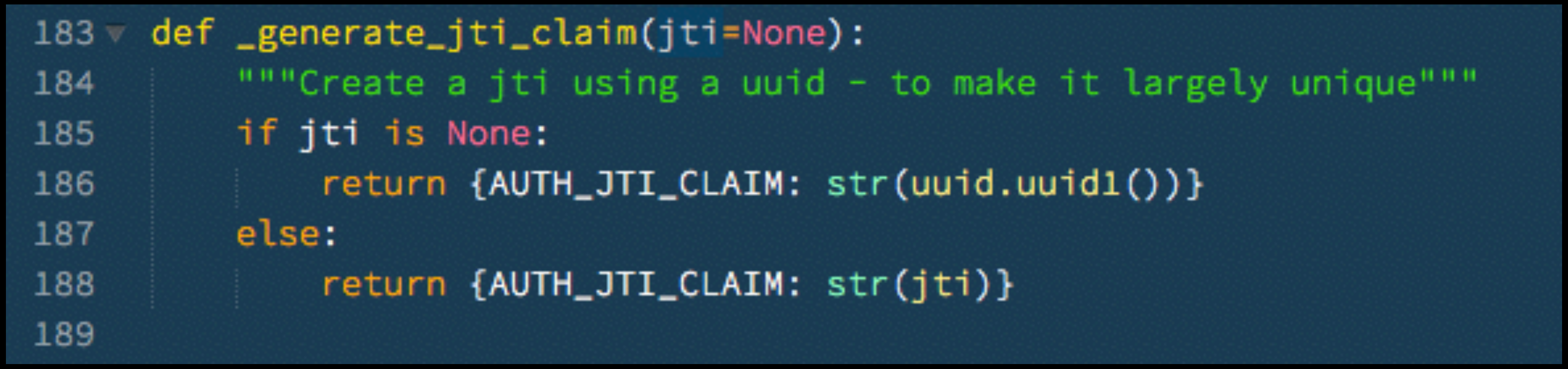
Run the test and see it fail



```
$ nosetests -v websandpym/tests/test_auth_utils.py:Utils_Tests.test__generate_jti_claim
test__generate_jti_claim (websandpym.tests.test_auth_utils.Utils_Tests) ... ERROR

======================================================================
ERROR: test__generate_jti_claim (websandpym.tests.test_auth_utils.Utils_Tests)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/alex/unison-root/Projects/Websand/instances/websand-pym/websand-pym/websandpym/tests/test_auth_utils.py",
line 101, in test__generate_jti_claim
    claims = auth_utils._generate_jti_claim()
nose.proxy.AttributeError: 'module' object has no attribute '_generate_jti_claim'
-------------------- >> begin captured logging << --------------------
passlib.utils.compat: DEBUG: loaded lazy attr 'SafeConfigParser': <class 'configparser.ConfigParser'>
passlib.utils.compat: DEBUG: loaded lazy attr 'NativeStringIO': <class '_io.StringIO'>
passlib.utils.compat: DEBUG: loaded lazy attr 'BytesIO': <class '_io.BytesIO'>
passlib.registry: DEBUG: registered 'bcrypt' handler: <class 'passlib.handlers.bcrypt.bcrypt'>
passlib.registry: DEBUG: registered 'pbkdf2_sha256' handler: <class 'passlib.handlers.pbkdf2.pbkdf2_sha256'>
--------------------- >> end captured logging << ---------------------

----------------------------------------------------------------------
Ran 1 test in 0.002s

FAILED (errors=1)
```

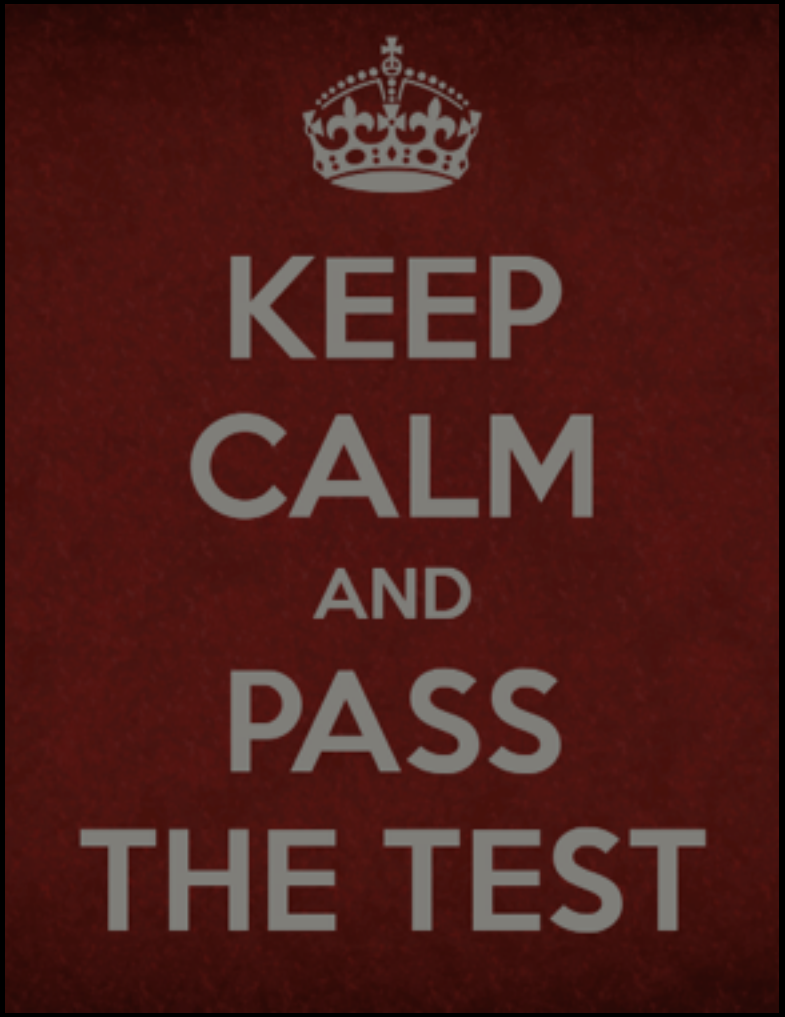
Write some code





```python
def _generate_jti_claim(jti=None):
    """Create a jti using a uuid - to make it largely unique"""
    if jti is None:
        return {AUTH_JTI_CLAIM: str(uuid.uuid1())}
    else:
        return {AUTH_JTI_CLAIM: str(jti)}

```

Run test again

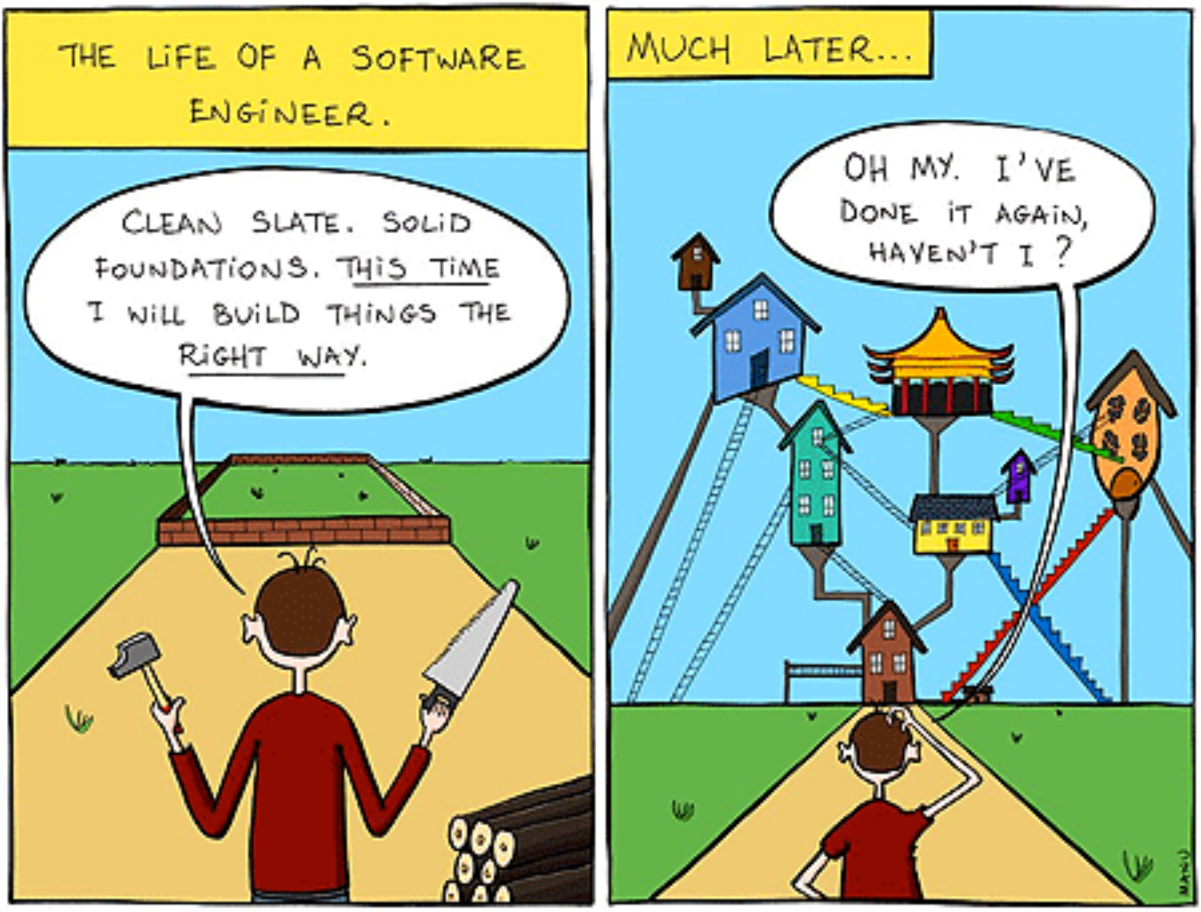


```
$ nosetests -v websandpym/tests/test_auth_utils.py:Utils_Tests.test__generate_jti_claim
test__generate_jti_claim (websandpym.tests.test_auth_utils.Utils_Tests) ... ok

----------------------------------------------------------------------
Ran 1 test in 0.014s

OK
```

…until the test passes

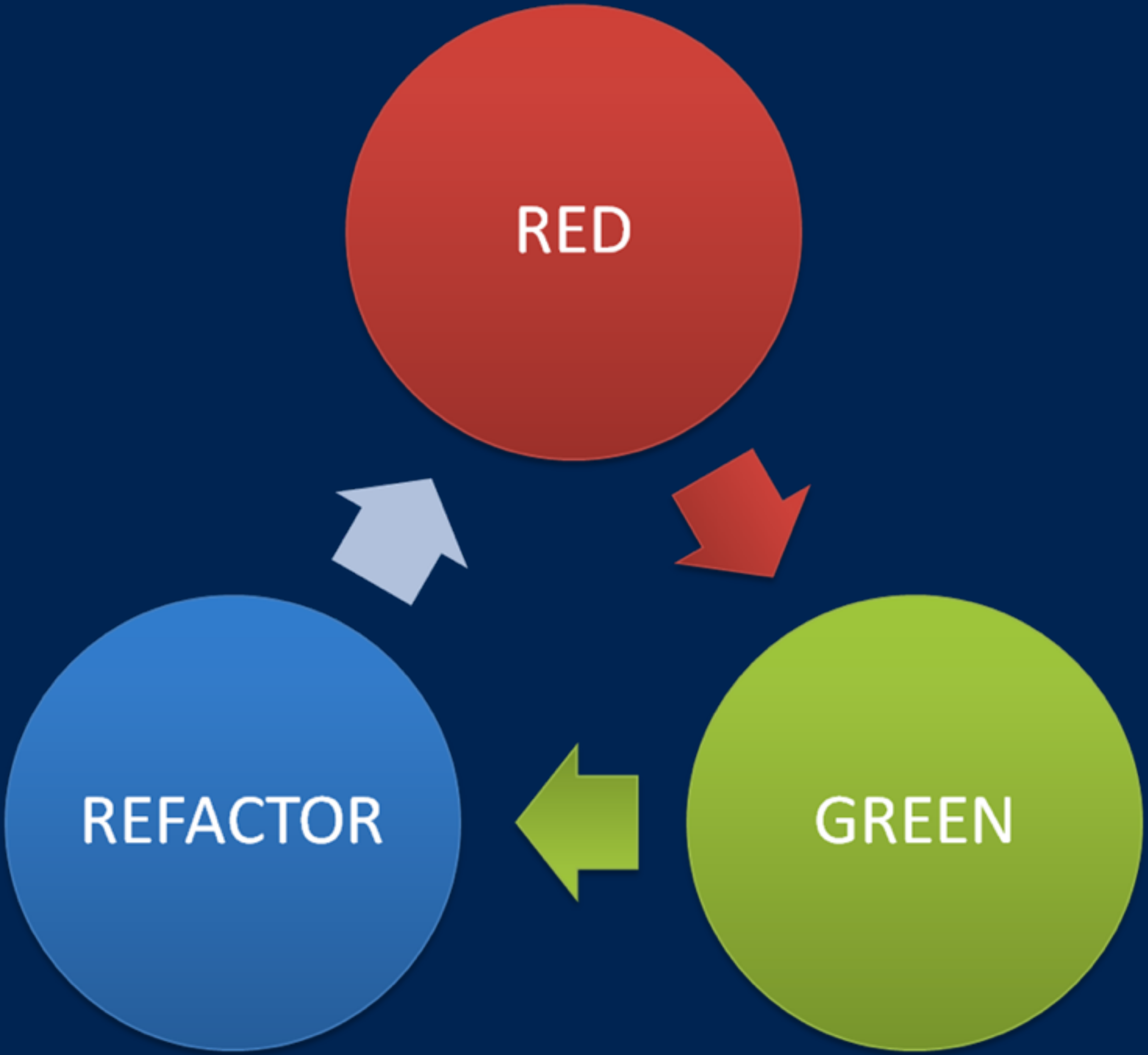
then …

Refactor


THE LIFE OF A SOFTWARE ENGINEER.
CLEAN SLATE. SOLID FOUNDATIONS. THIS TIME I WILL BUILD THINGS THE RIGHT WAY.
MUCH LATER...
OH MY. I'VE DONE IT AGAIN, HAVEN'T I ?
MANU

Today
Rinse
&
repeat

In summary


RED
GREEN
REFACTOR

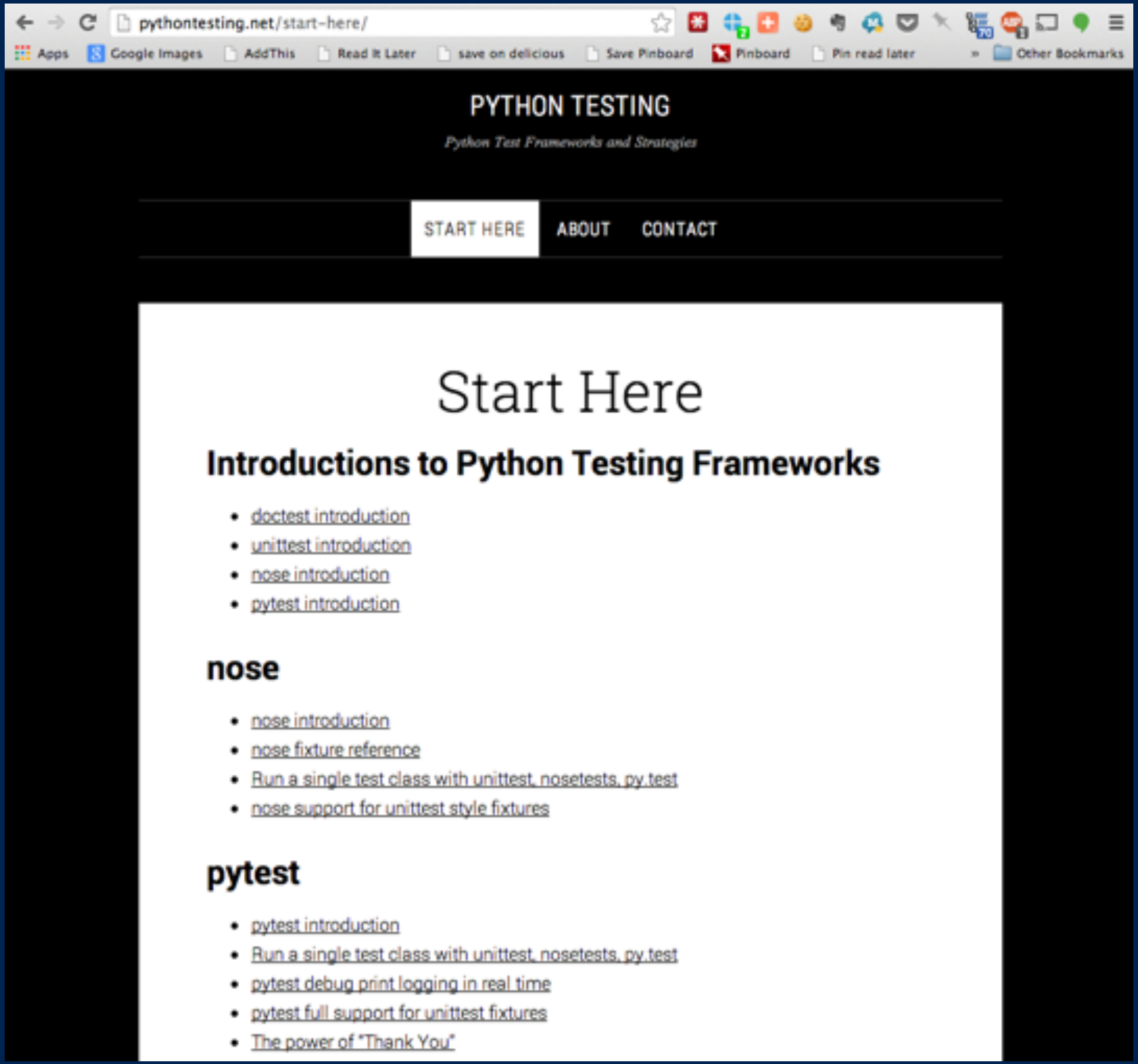
TDD and Python

Learning by doing … (scary live demo)

(to play along: https://github.com/ajkavanagh/PythonNE-TDD-Example)

Where to next?

http://pythontesting.net/start-here/



PYTHON TESTING

Python Test Frameworks and Strategies

Start Here

Introductions to Python Testing Frameworks

- doctest introduction
- unittest introduction
- nose introduction
- pytest introduction

nose

- nose introduction
- nose fixture reference
- Run a single test class with unittest, nosetests, py.test
- nose support for unittest style fixtures

pytest

- pytest introduction
- Run a single test class with unittest, nosetests, py.test
- pytest debug print logging in real time
- pytest full support for unittest fixtures
- The power of "Thank You"

Topics to look at

Mocking

End2End Testing (for web frameworks)

BDD

etc. (it’s a big field …)

"Unit Tests confirm we have written the code right [sic].

Acceptance Tests that we built the right software."

–codebetter.com

"TDD doesn't drive good design. TDD gives you immediate feedback about what is likely to be bad design."

–Kent Beck

Code Dojo Ideas

1. Write a module that ‘manages’ Polar coords. Allow transformations between cartesian and polar coords.
2. Write a module that counts frequencies of words passed to. Extra credit - remember previous counts and add to them.
3. Implement a really simple shopping cart. Extra credit - add a discounting system.
4. Implement a Bank Account class that can transfer money between accounts, have account limits.

Image Credits

http://www.cfn308bestnetwork.com/sites/default/files/writing.gif
http://2.bp.blogspot.com/_4OIF7pROmHM/SwV3hzmTaAI/AAAAAAAAAe8/wVyXyHhMrT0/s1600/54208__468x_failed-exam-answers-22.jpg
http://allantyoung.com/wp-content/uploads/2008/11/matrixcode.jpg
http://icons.iconarchive.com/icons/icons-land/vista-multimedia/256/Play-Mode-Repeat-All-Hot-icon.png
http://coltandsage.com/wp-content/uploads/2013/12/rinse-repeat.jpg
http://carlosble.com/downloads/tdd.png
http://www.ben-morris.com/wp-content/library/refactoring-vs-rewriting.gif