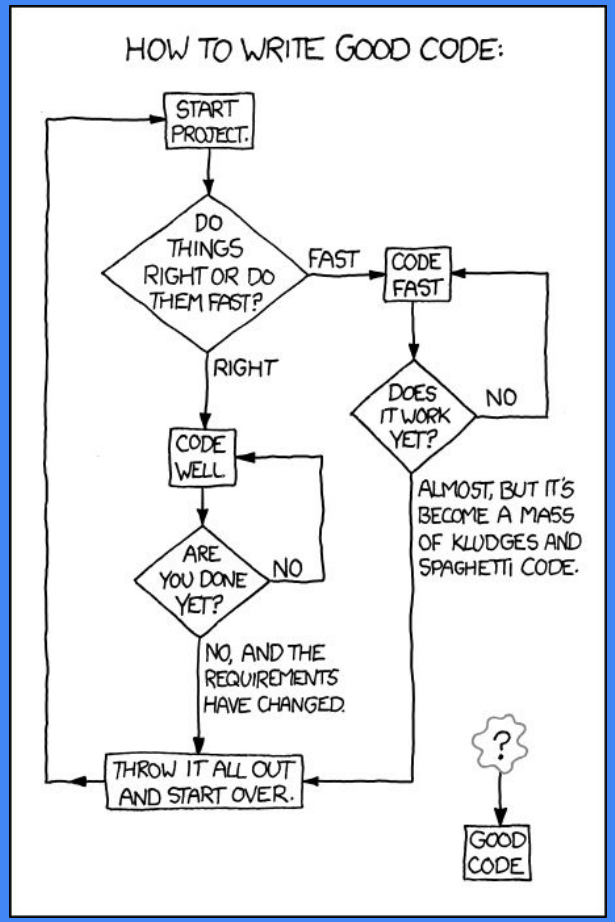
ITSE-1402 Intermediate Python

Class 7: Debugging, Tests, and Algorithm Analysis

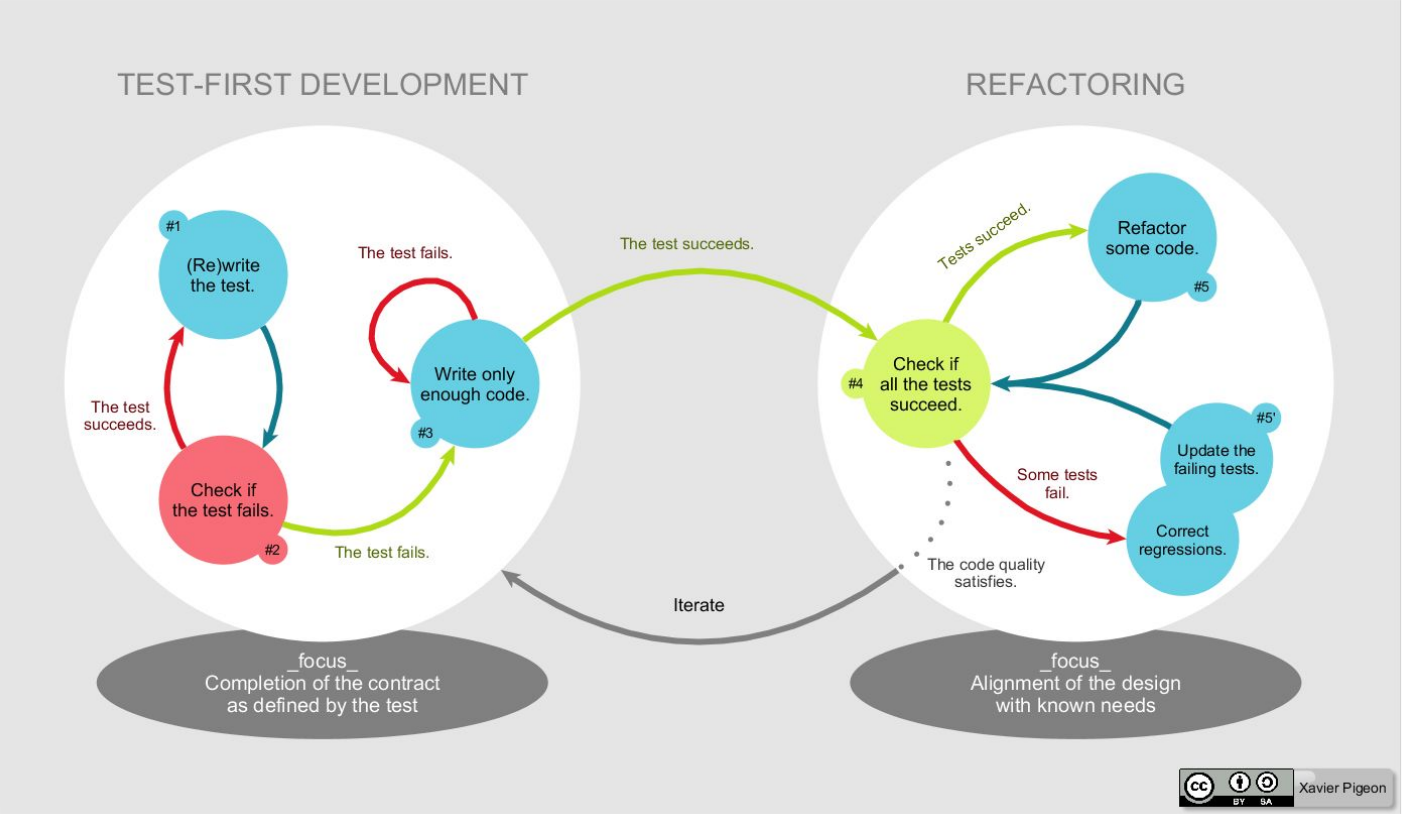
June 22, 2017



You can either hang out in the Android Loop or the HURD loop.

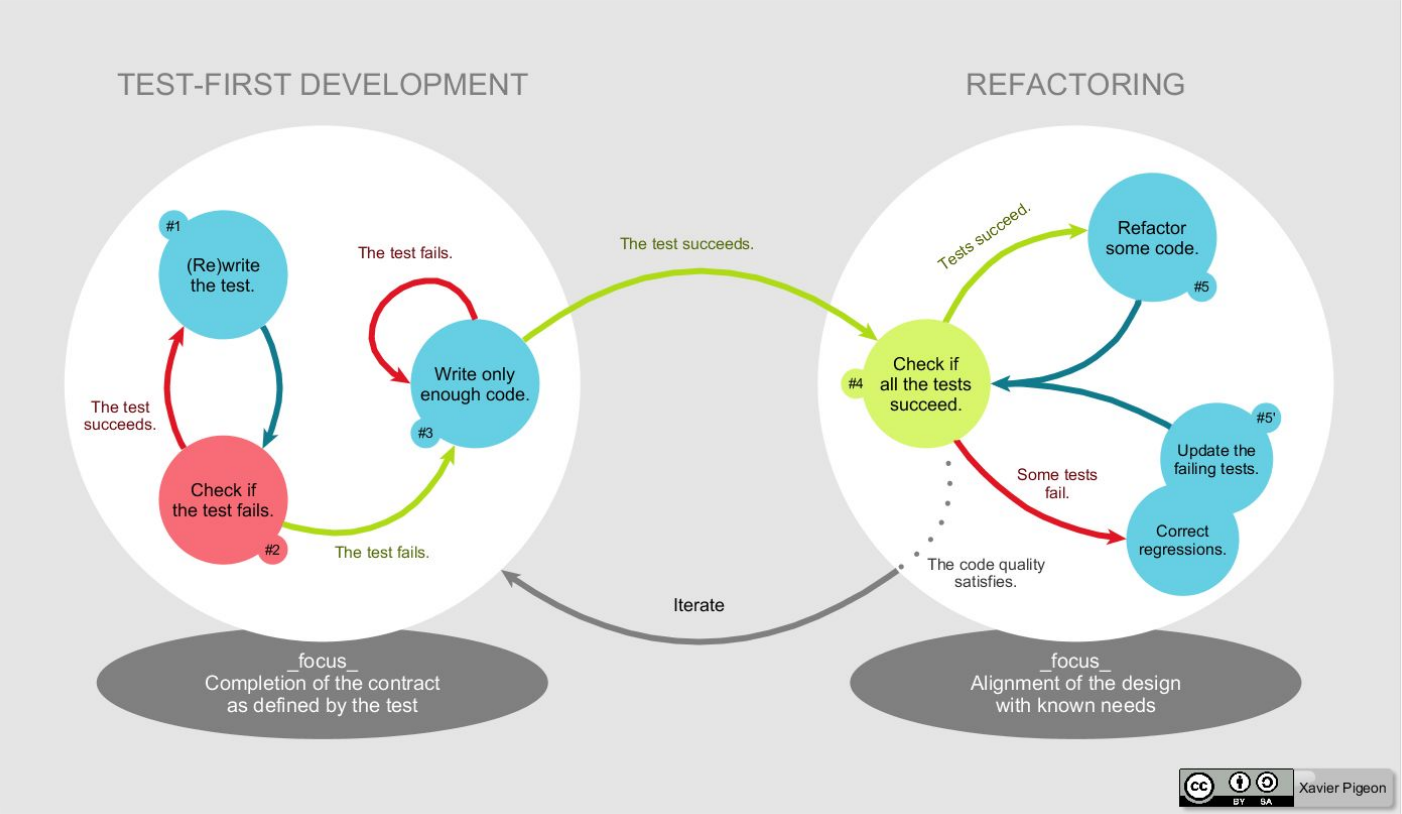
https://xkcd.com/844/

Test Driven Development



If we are going to talk about tests, why not talk about the type of development centered around them.

Test Driven Development



1. Write a test
2. Write code that passes test
3. Refactor
4. Repeat

Example Story

We need to check whether a given input string is a palindrome.

Example Story - Function

palindrome.py

```
def is_palindrome(strin):
    pass
```

Example Story - Test

test.py

```
from palindrome import is_palindrome

def test_palindrome_words():
        input = "noon"
        assert is_palindrome(input) == True
```

Example Story - Test Result

```
philipulrich:~/workspace/class7 (master) $ nosetests
F
======================================================================
FAIL: test.test_palindrome_words
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/usr/local/lib/python3.4/dist-packages/nose/case.py", line 198, in runTest
    self.test(*self.arg)
  File "/home/ubuntu/workspace/class7/test.py", line 5, in test_palindrome_words
    assert is_palindrome(input) == True
AssertionError
----------------------------------------------------------------------
Ran 1 test in 0.006s
FAILED (failures=1)
```

Example Story - Function

palindrome.py

```
def is_palindrome(strin):
    return strin == strin[::-1]
```

Example Story - Test Result

```
philipulrich:~/workspace/class7 (master) $ nosetests
.
----------------------------------------------------------------------
Ran 1 test in 0.005s
OK
```

Example Story - Test

test.py

```
def test_palindrome_case_independent():
        input = "Noon"
        assert is_palindrome(input) == True
```

Example Story - Test Result

```
philipulrich:~/workspace/class7 (master) $ nosetests
.F
======================================================================
FAIL: test.test_palindrome_case_independent
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/usr/local/lib/python3.4/dist-packages/nose/case.py", line 198, in runTest
    self.test(*self.arg)
  File "/home/ubuntu/workspace/class7/test.py", line 9, in test_palindrome_case_independent
    assert is_palindrome(input) == True
AssertionError
----------------------------------------------------------------------
Ran 2 tests in 0.005s
FAILED (failures=1)
```

Example Story - Function

palindrome.py

```
def is_palindrome(strin):
    lstrin = strin.lower()
    return lstrin == lstrin[::-1]
```

Example Story - Test Result

```
philipulrich:~/workspace/class7 (master) $ nosetests
..
----------------------------------------------------------------------
Ran 2 tests in 0.005s
OK
```

Example Story - Test

test.py

```
def test_palindrome_with_spaces():
        input = "     Noon         "
        assert is_palindrome(input) == True
```

Example Story - Test Result

```
philipulrich:~/workspace/class7 (master) $ nosetests
..F
======================================================================
FAIL: test.test_palindrome_with_spaces
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/usr/local/lib/python3.4/dist-packages/nose/case.py", line 198, in runTest
    self.test(*self.arg)
  File "/home/ubuntu/workspace/class7/test.py", line 13, in test_palindrome_with_spaces
    assert is_palindrome(input) == True
AssertionError
----------------------------------------------------------------------
Ran 3 tests in 0.005s
FAILED (failures=1)
```

Example Story - Function

palindrome.py

```
def is_palindrome(strin):
    cstrin = strin.strip().lower()
    return cstrin == cstrin[::-1]
```

Example Story - Test Result

```
philipulrich:~/workspace/class7 (master) $ nosetests
…
----------------------------------------------------------------------
Ran 3 tests in 0.005s
OK
```

Example Story - Test

test.py

```
def test_palindrome_with_spaces_between():
    input = "A man a plan ... a canal Panama"
    assert is_palindrome(input) == True
```

Example Story - Test Result

```
philipulrich:~/workspace/class7 (master) $ nosetests
..F
======================================================================
FAIL: test.test_palindrome_with_spaces
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/usr/local/lib/python3.4/dist-packages/nose/case.py", line 198, in runTest
    self.test(*self.arg)
  File "/home/ubuntu/workspace/class7/test.py", line 13, in test_palindrome_with_spaces
    assert is_palindrome(input) == True
AssertionError
----------------------------------------------------------------------
Ran 4 tests in 0.005s
FAILED (failures=1)
```

Example Story - Function

palindrome.py

```
def is_palindrome(strin):
    cstrin = strin.replace(" ", "").lower()
    return cstrin == cstrin[::-1]
```

Example Story - Test Result

```
philipulrich:~/workspace/class7 (master) $ nosetests
....
----------------------------------------------------------------------
Ran 4 tests in 0.005s
OK
```

Tests

Testing is important to ensure that your code is quality. As you can see though, it adds additional time to the development process. In this case, however, if we did not test… we would have missed quite a lot of potential problems.

Quality Tests

In order for tests to be good, they must be:

- Consistent / Reliable
- Clear
- Quick
- Offer Full Coverage

Testing Frameworks

We will be looking at three testing frameworks:

- nose
- unittest
- py.test

Testing Frameworks

Each Framework has its pros and cons, but “nose” is one of the most popular:

nose

Pros:
- works with unittest
- django (and others) support
- advanced features
- lots of plugins
- IDE support

Cons:
- not a standard

unittest

Pros:
- similar to other languages tests
- part of standard python library
- IDE support
- widely used

Cons:
- not pretty/pythonic
- lots of code
- no autodiscovery
- no advanced features

py.test

Pros:
- test autodiscovery
- easy to write/run
- advanced features
- lots of plugins

Cons:
- not standard
- little IDE support

Nose

```
def test_palindrome_words():
    input = "noon"
    assert is_palindrome(input) == True
```

py.test

```
def test_palindrome_words():
    input = "noon"
    assert is_palindrome(input) == True
```

unittest

```
Import unittest

class TestPalindrome(unittest.TestCase):
        def test_palindrome_words():
                input = "noon"
                self.assertTrue(is_palindrome(input))
```