

Why are we talking about testing?

On board to DevOps and automation

- Faster feedback cycle
- Reusability
- Accelerate time to market

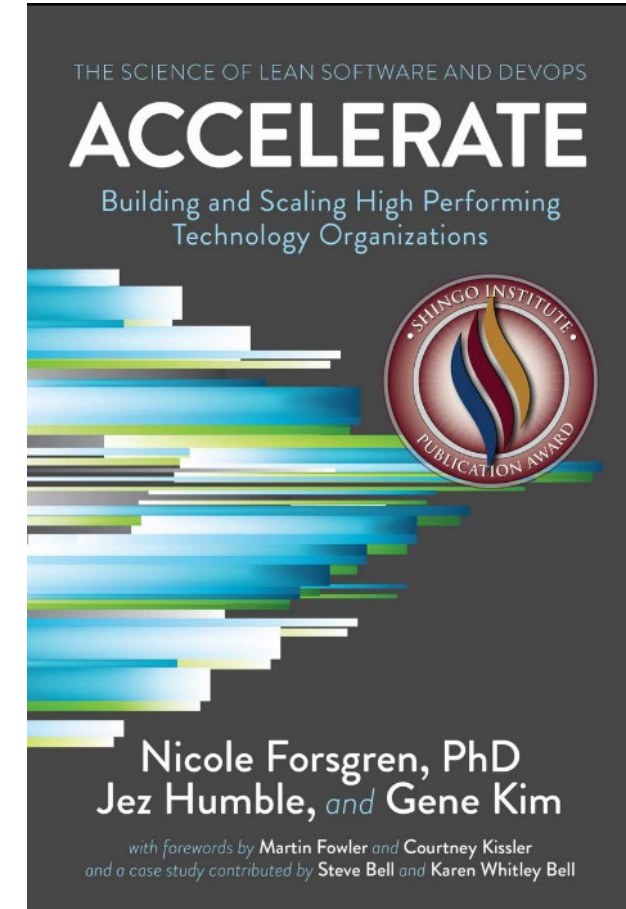
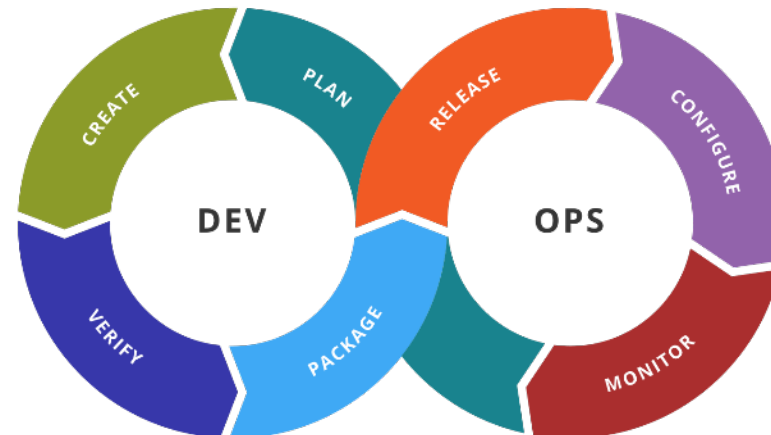
Immediate payback

- Enhanced accuracy
- Earlier bug detection

Begin capturing business logic

Need evidence?

- Read the book! (9781942788331)



Tribal Knowledge is walking out the door

Developers

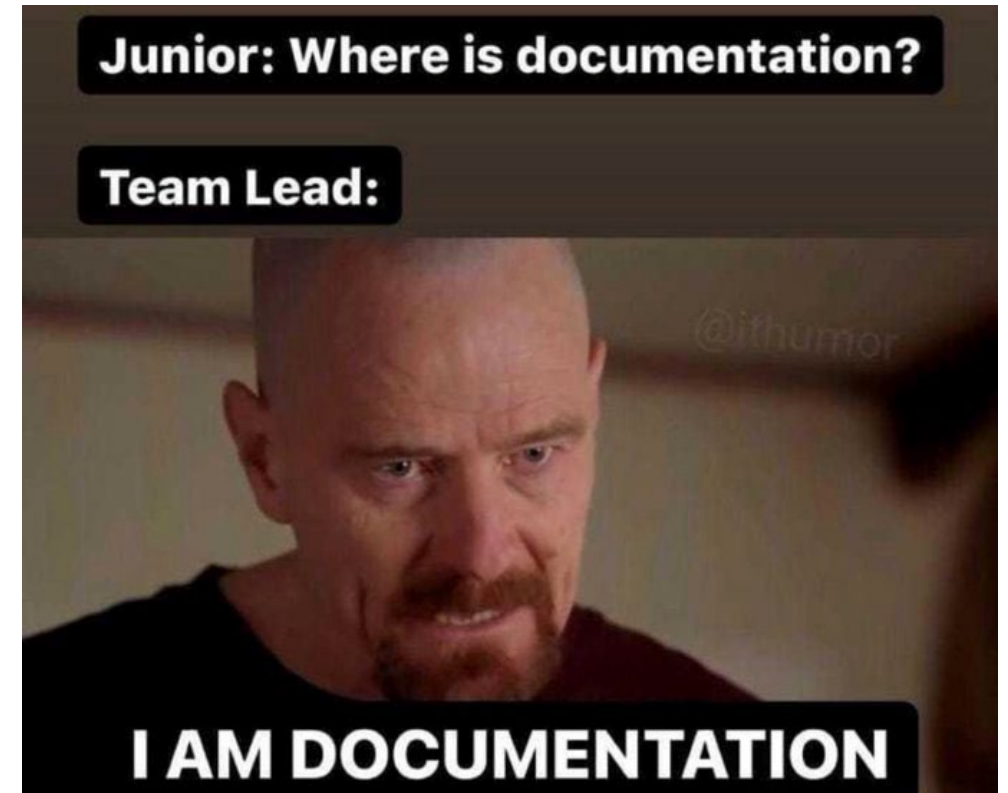
- Retiring
- Moving to new tech / platforms

Users

- Application testers
- Thought leaders

No one is safe!

- Great resignation
- Next layoff



Agenda of Test-Driven Development: TDD

Discuss linting

Fundamentals

Explore the sample requirement

Write a test

Write some code

Rinse & repeat

- Conditionals
- Value error

Wrap-up



What is Linting?

Linting is the automated checking of

- Syntax
- Style

Most of this is done in a quality IDE

- IDLE is NOT a quality IDE
- PyCharm, Sublime, VS Code are all quality IDEs

We will explore PyLint

See PEP8 for rules...

Why?

- Interoperability!

How do you know if you are OK?

PyLint module (third party module)

pip install pylint

```
Windows PowerShell
PS C:\Users\MikeP> pip install pylint
Collecting pylint
  Downloading pylint-3.1.0-py3-none-any.whl (515 kB)
----- 515.6/515.6 kB 4.6 MB/s eta 0:00:00
Collecting platformdirs>=2.2.0
  Downloading platformdirs-4.2.0-py3-none-any.whl (17 kB)
Collecting astroid<=3.2.0-dev0,>=3.1.0
  Downloading astroid-3.1.0-py3-none-any.whl (275 kB)
----- 275.6/275.6 kB 17.7 MB/s eta 0:00:00
Collecting isort!=5.13.0,<6,>=4.2.5
  Downloading isort-5.13.2-py3-none-any.whl (92 kB)
----- 92.3/92.3 kB 5.5 MB/s eta 0:00:00
Collecting mccabe<0.8,>=0.6
  Downloading mccabe-0.7.0-py2.py3-none-any.whl (7.3 kB)
Collecting tomkit>=0.10.1
  Downloading tomkit-0.12.4-py3-none-any.whl (37 kB)
Collecting dill>=0.3.6
  Downloading dill-0.3.8-py3-none-any.whl (116 kB)
----- 116.3/116.3 kB ? eta 0:00:00
Collecting colorama>=0.4.5
  Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Installing collected packages: tomkit, platformdirs, mccabe, isort, dill, colorama, astroid, pylint
Successfully installed astroid-3.1.0 colorama-0.4.6 dill-0.3.8 isort-5.13.2 mccabe-0.7.0 platformdirs-4.2.0 pylint-3.1.0 tomkit-0.12.4

[notice] A new release of pip available: 22.3.1 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip
PS C:\Users\MikeP>
```

```
PS C:\Users\MikeP> pylint --version
pylint 3.1.0
astroid 3.1.0
Python 3.11.3 (tags/v3.11.3:f3909b8, Apr 4 2023, 23:49:59) [MSC v.1934 64 bit (AMD64)]
PS C:\Users\MikeP>
```

Short program

Snippet of code to exploring the linter

Use command line to “lint” the file

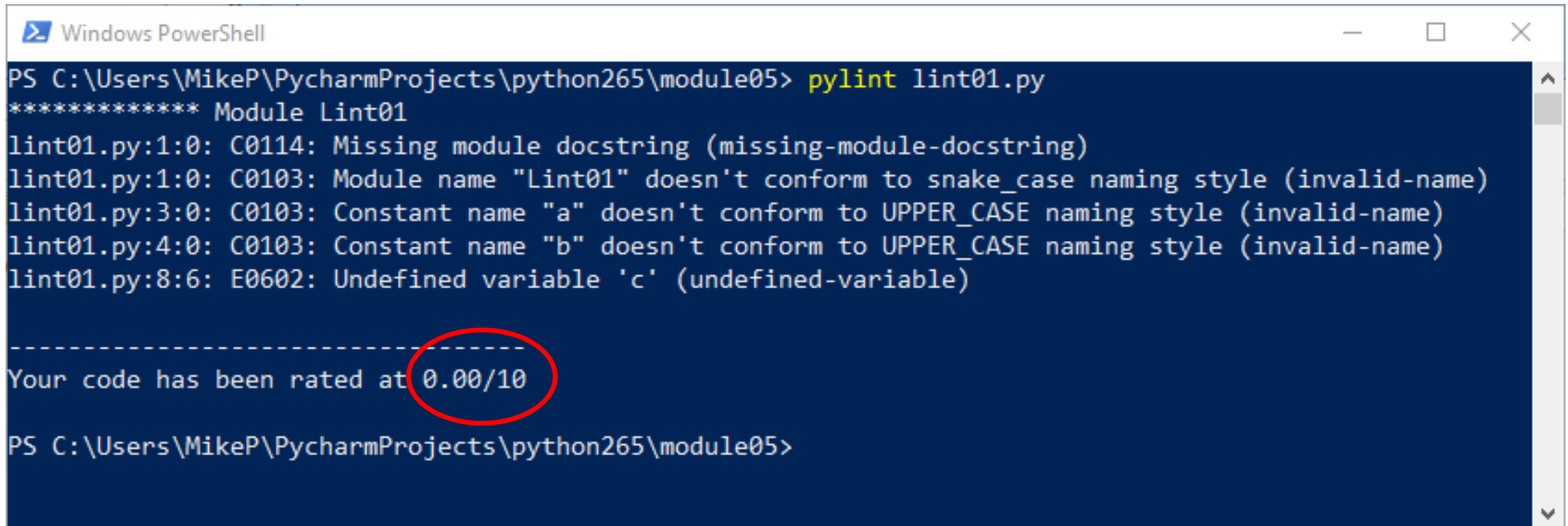
```
a = 1
```

```
b = 2
```

```
print(a)
```

```
print(b)
```

```
print(c)
```



```
Windows PowerShell
PS C:\Users\MikeP\PycharmProjects\python265\module05> pylint lint01.py
***** Module Lint01
lint01.py:1:0: C0114: Missing module docstring (missing-module-docstring)
lint01.py:1:0: C0103: Module name "Lint01" doesn't conform to snake_case naming style (invalid-name)
lint01.py:3:0: C0103: Constant name "a" doesn't conform to UPPER_CASE naming style (invalid-name)
lint01.py:4:0: C0103: Constant name "b" doesn't conform to UPPER_CASE naming style (invalid-name)
lint01.py:8:6: E0602: Undefined variable 'c' (undefined-variable)

-----
Your code has been rated at 0.00/10

PS C:\Users\MikeP\PycharmProjects\python265\module05>
```

And all better

Upper case the constants...

```
""" This is a simple program to illustrate linting """
```

```
A = 1
```

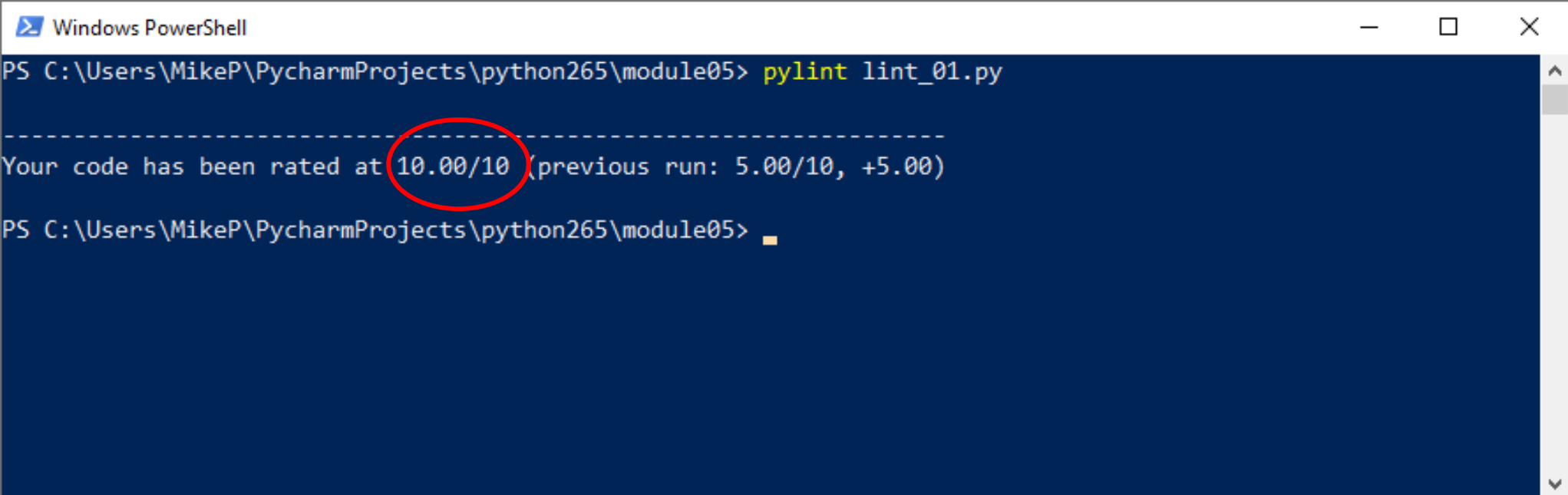
```
B = 2
```

```
C = 3
```

```
print(A)
```

```
print(B)
```

```
print(C)
```



```
Windows PowerShell
PS C:\Users\MikeP\PycharmProjects\python265\module05> pylint lint_01.py
-----
Your code has been rated at 10.00/10 (previous run: 5.00/10, +5.00)
PS C:\Users\MikeP\PycharmProjects\python265\module05>
```

Fundamentals



TDD: Test Driven Development

Different approach to developing require some basic assumptions:

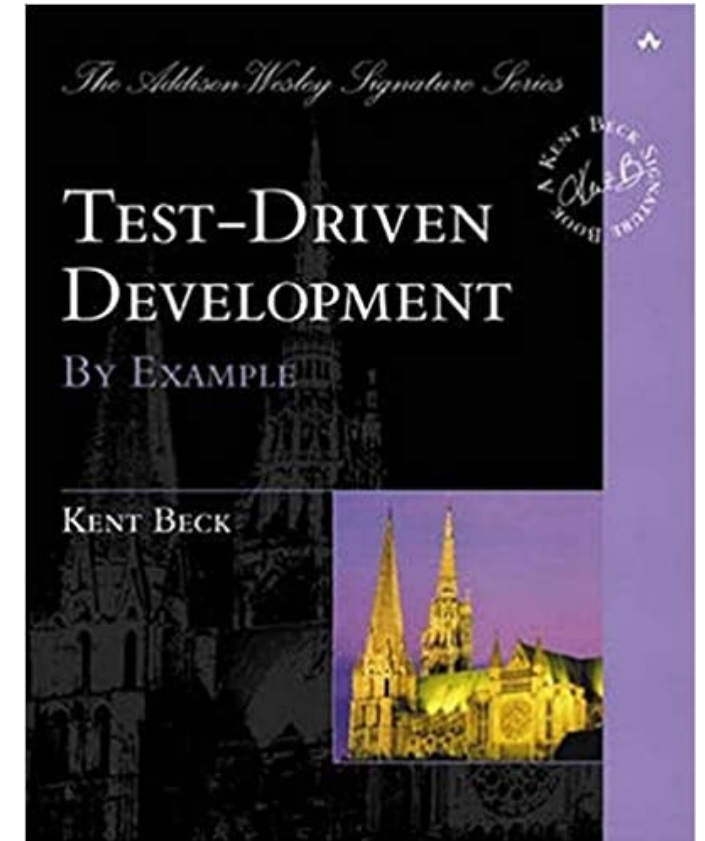
- Code is loosely coupled
- Part of a bigger system
- Tests are for components, not necessarily programs
 - Functions and methods
- Iterative development model
- Tests are written before code....(Wahhhh?)

Conceptualized by Kent Beck in his book

- Test Driven Development By Example

Does not replace UAT

- Reduces it
- Automates unit testing



TDD: Test Driven Development (Why?)

DevOps is the way forward

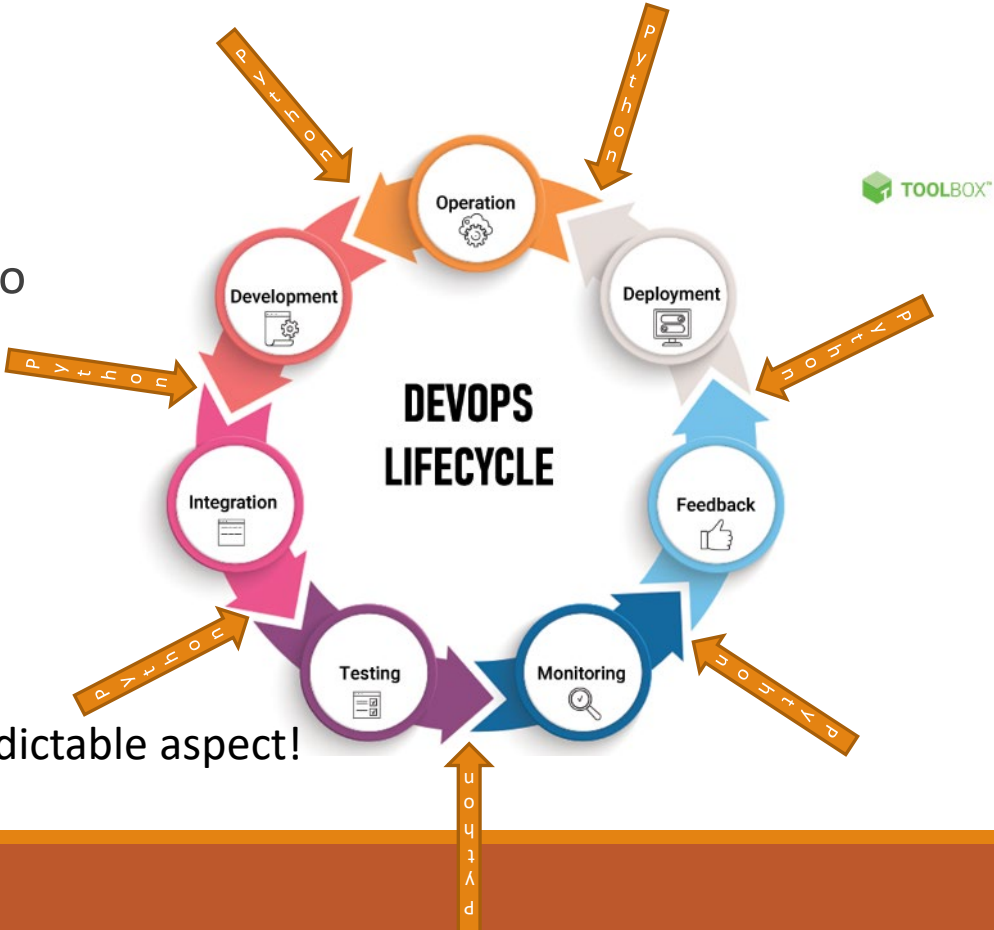
- Automating the Application Development Life Cycle

Manual tests are difficult to automate

APIs are difficult to test until the UI is complete

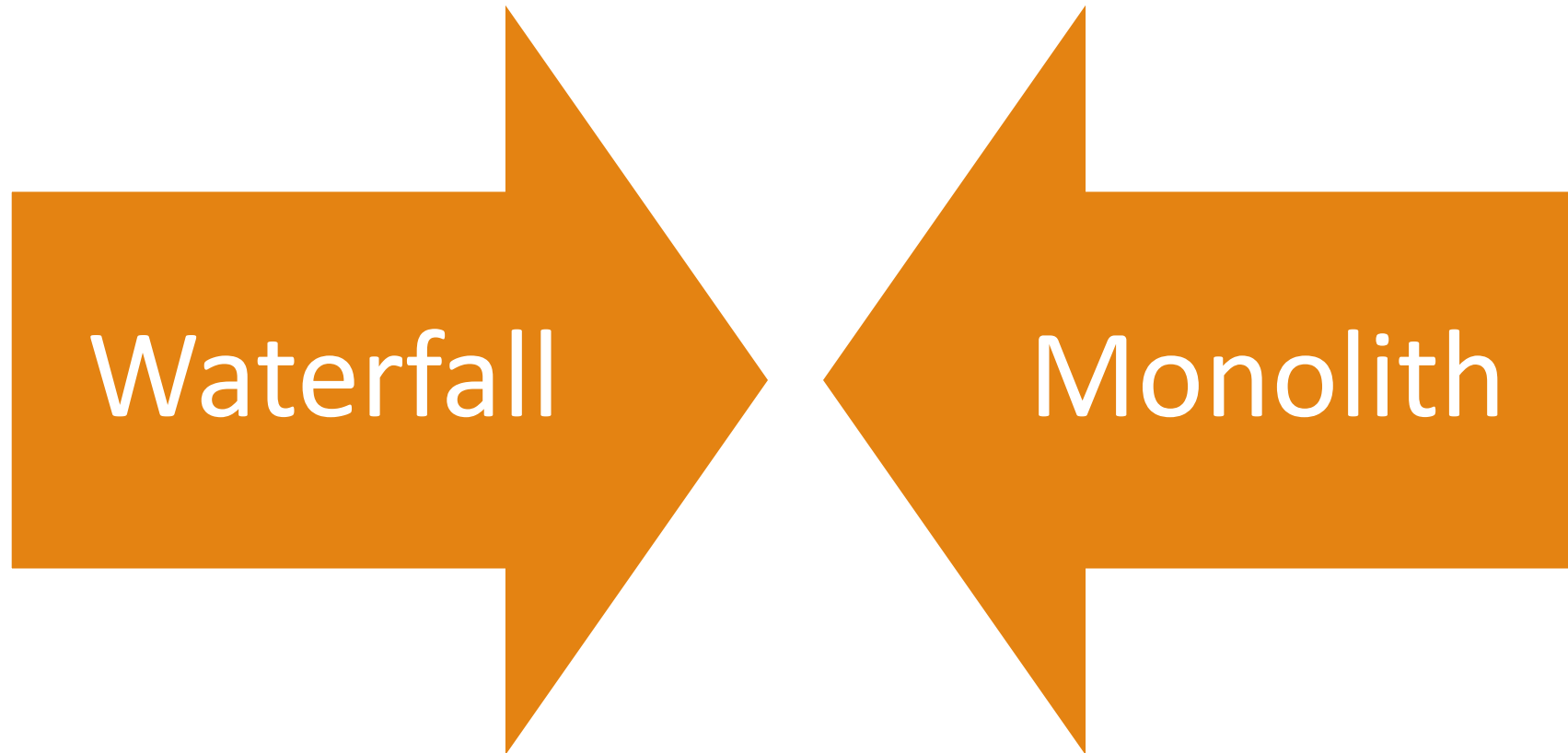
Decouple back end and front end, better testing scenario

- Expected Benefits**
- many teams report significant reductions in defect rates, at the cost of a moderate increase in initial development effort
 - the same teams tend to report that these overheads are more than offset by a reduction in effort in projects' final phases
 - although empirical research has so far failed to confirm this, veteran practitioners report that TDD leads to improved design qualities in the code, and more generally a higher degree of "internal" or technical quality, for instance improving the metrics of cohesion and coupling

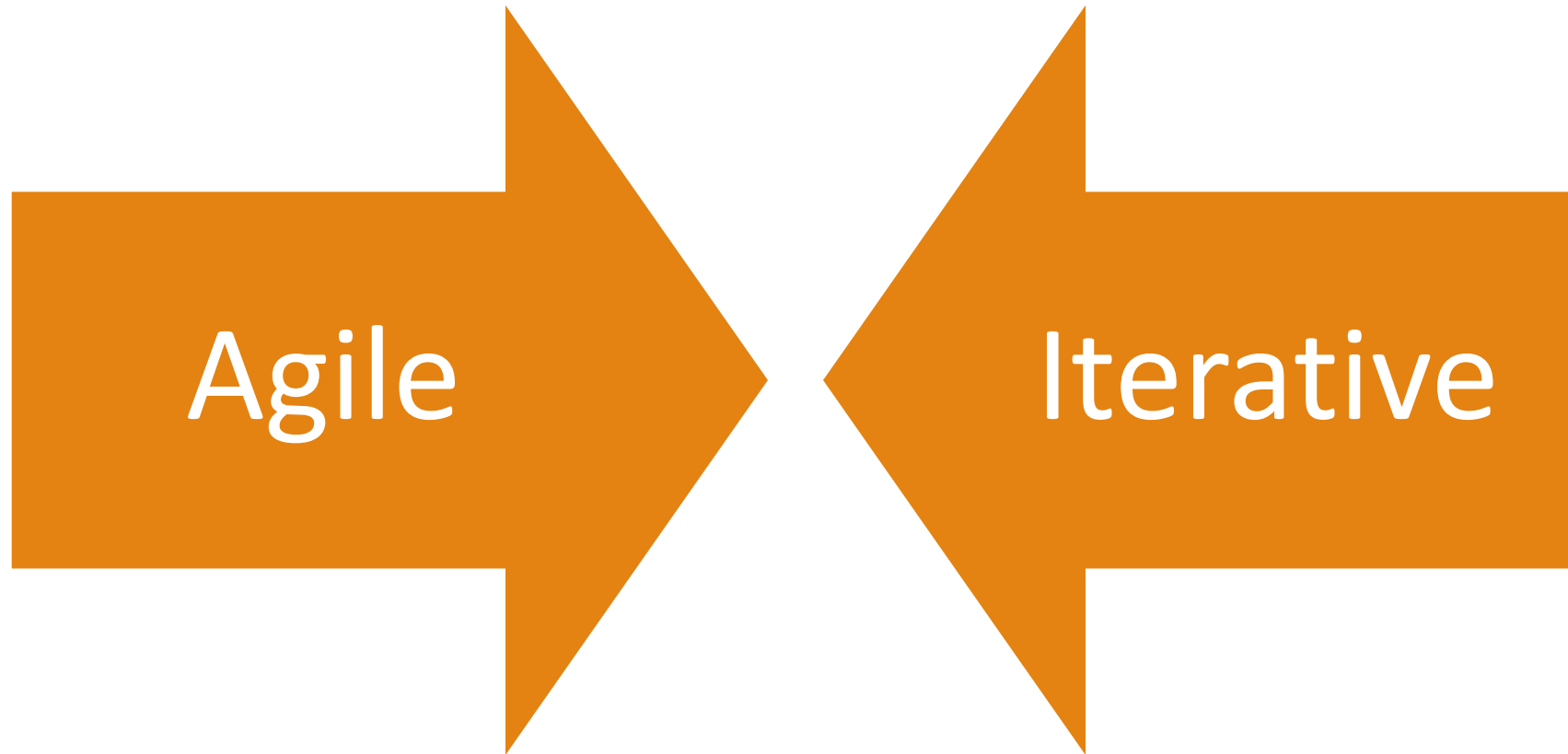


Testing is the most unpredictable aspect!

Simple Context Part 1: Legacy

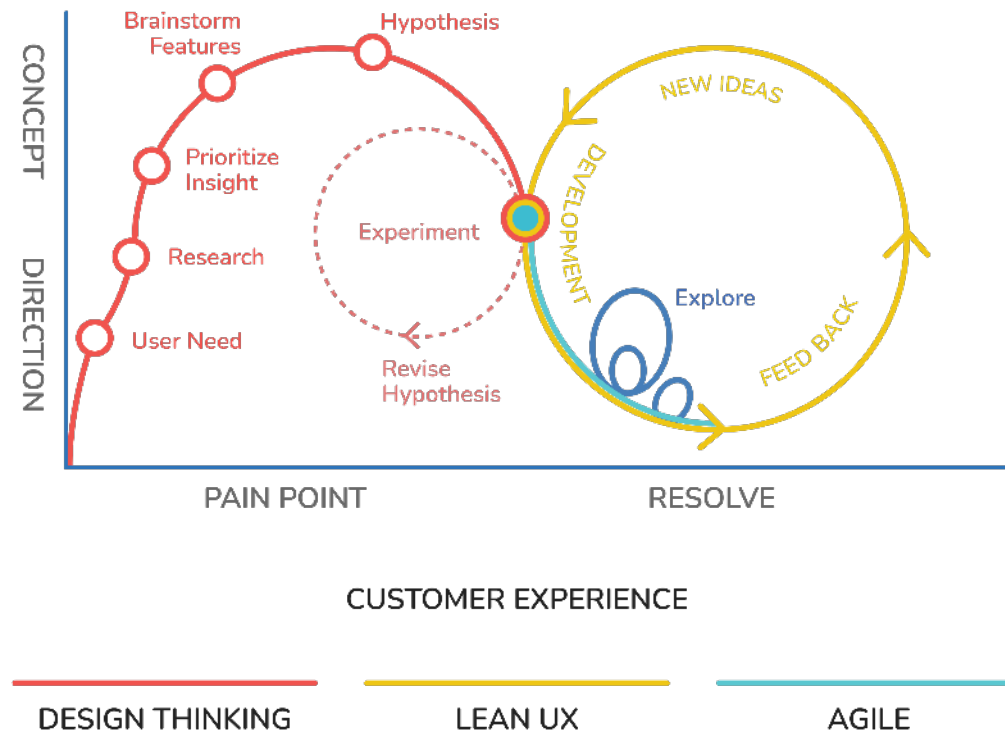


Simple Context Part 2: Modern



Getting started with Agile

MVP Process



A Python Example of TDD

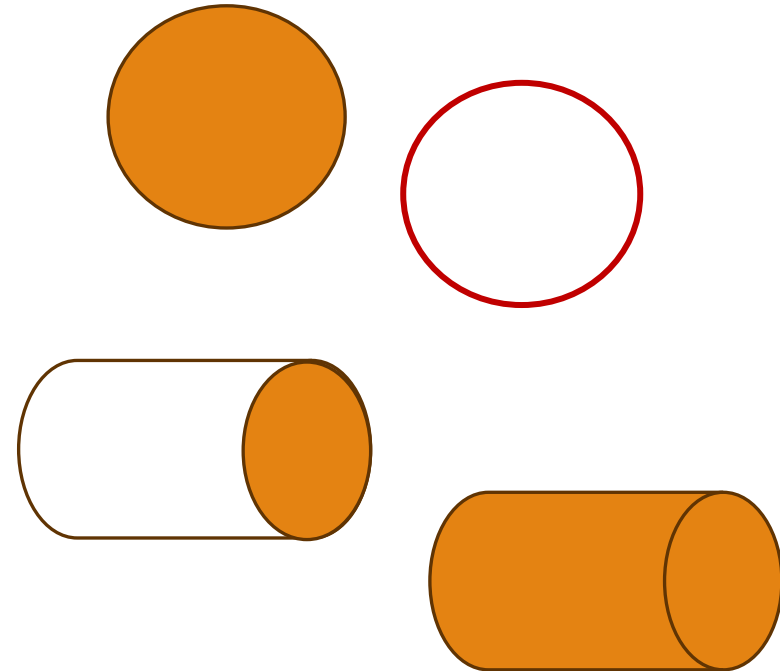
Start with basic circle functions

Area of a circle – πr^2

Circumference of a circle – $2\pi r$

Volume of a cylinder – $\pi r^2 h$

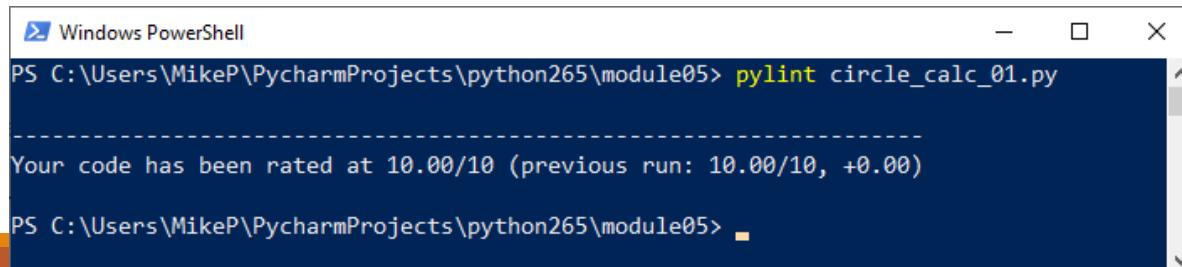
Surface area of a cylinder - $2\pi r h + 2\pi r^2$



Function file

Import pi from the math module

Define the function and save in a file



```
Windows PowerShell
PS C:\Users\MikeP\PycharmProjects\python265\module05> pylint circle_calc_01.py
-----
Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)
PS C:\Users\MikeP\PycharmProjects\python265\module05>
```

```
from math import pi
```

```
def square_area(l):
```

```
    """ calculate the area of c circle """
```

```
    area = l**2
```

```
    return area
```

```
def circle_area(r):
```

```
    """ calculate the area of c circle """
```

```
    area = pi * r**2
```

```
    return area
```

```
def circle_circum(r):
```

```
    """ calculate the circumference of a circle """
```

```
    circum = 2 * pi * r
```

```
    return circum
```

```
def cyl_volume(r,h):
```

```
    """ Calculate the volume of a cylinder """
```

```
    volume = pi * r ** 2 * h
```

```
    return volume
```


Need some examples

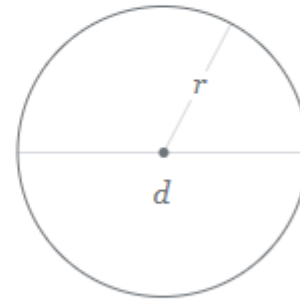
Hey Google!

Calculators >

Solve for
area ▼

$$A \approx 78.54$$

r Radius



Solution

$$A = \pi r^2 = \pi \cdot 5^2 \approx 78.53982$$

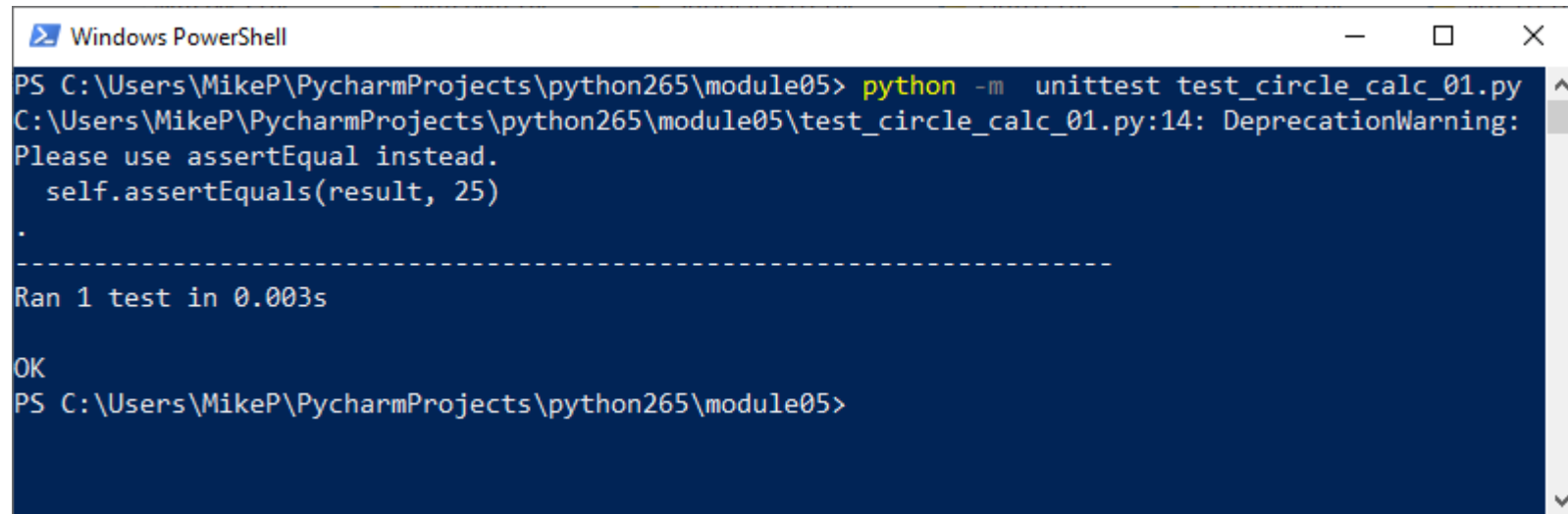
Start with the square

Using assertEquals works (notice the warning...we'll get to that)

```
import unittest
import circle_calc_01 as circ
from math import pi

class TestCircles(unittest.TestCase):
    """ Class for testing circles """

    def test_circle_area(self):
        result = circ.square_area(5)
        # self.assertEqual(result, 78.54)
        self.assertEqual(result, 25)
```



```
Windows PowerShell
PS C:\Users\MikeP\PycharmProjects\python265\module05> python -m unittest test_circle_calc_01.py
C:\Users\MikeP\PycharmProjects\python265\module05\test_circle_calc_01.py:14: DeprecationWarning:
Please use assertEquals instead.
  self.assertEqual(result, 25)
.
-----
Ran 1 test in 0.003s

OK
PS C:\Users\MikeP\PycharmProjects\python265\module05>
```

Now let's do a circle

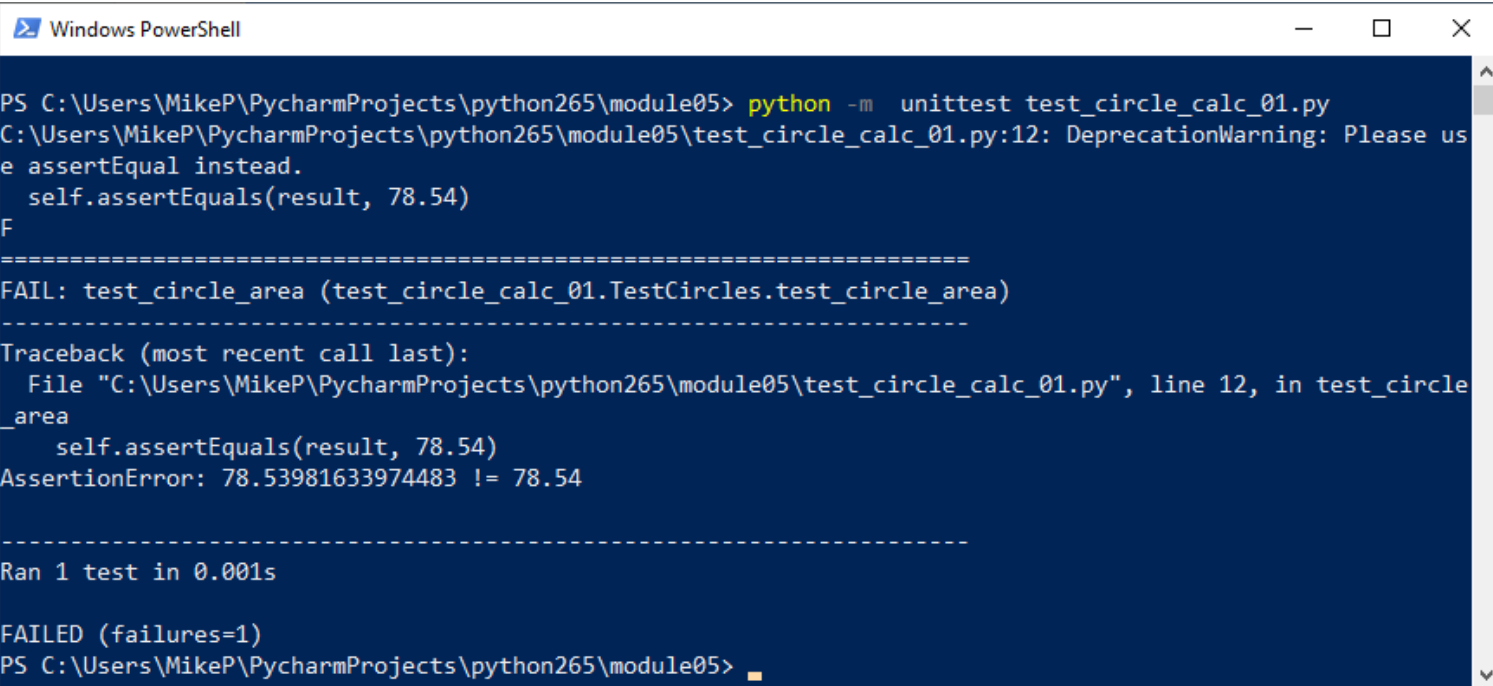
test_nnnnn

Hmmmm, it seems close enough to me?

```
import unittest
import circle_calc_01 as circ
from math import pi

class TestCircles(unittest.TestCase):
    """ Class for testing circles """

    def test_circle_area(self):
        result = circ.circle_area(5)
        self.assertEqual(result, 78.54)
```



```
Windows PowerShell

PS C:\Users\MikeP\PycharmProjects\python265\module05> python -m unittest test_circle_calc_01.py
C:\Users\MikeP\PycharmProjects\python265\module05\test_circle_calc_01.py:12: DeprecationWarning: Please use
self.assertEqual instead.
  self.assertEqual(result, 78.54)
F
=====
FAIL: test_circle_area (test_circle_calc_01.TestCircles.test_circle_area)
-----
Traceback (most recent call last):
  File "C:\Users\MikeP\PycharmProjects\python265\module05\test_circle_calc_01.py", line 12, in test_circle
_area
    self.assertEqual(result, 78.54)
AssertionError: 78.53981633974483 != 78.54
-----
Ran 1 test in 0.001s

FAILED (failures=1)
PS C:\Users\MikeP\PycharmProjects\python265\module05>
```

Lots of asserts...

Linters in PyCharm

```
self.assert_
assertEquals
assert_
assertIn(self, member, container, msg)
assertIs(self, expr1, expr2, msg)
assertTrue(self, expr, msg)
assertEqual(self, first, second, msg)
assertAlmostEqual(self, first, second, places, msg, del
assertAlmostEquals
assertCountEqual(self, first, second, msg)
assertDictContainsSubset(self, subset, dictionary, msg)
assertDictEqual(self, d1, d2, msg)
assertFalse(self, expr, msg)
assertGreater(self, a, b, msg)
assertGreaterEqual(self, a, b, msg)
assertIsInstance(self, obj, cls, msg)
assertIsNone(self, obj, msg)
assertIsNot(self, expr1, expr2, msg)
assertIsNotNone(self, obj, msg)
assertLess(self, a, b, msg)
assertLessEqual(self, a, b, msg)
assertListEqual(self, list1, list2, msg)
assertLogs(self, logger, level)
assertMultiLineEqual(self, first, second, msg)
assertNoLogs(self, logger, level)
assertNotAlmostEqual(self, first, second, places, msg,
assertNotAlmostEquals
assertNotEqual(self, first, second, msg)
```

Method	Checks that	New in
assertEqual(a, b)	<code>a == b</code>	
assertNotEqual(a, b)	<code>a != b</code>	
assertTrue(x)	<code>bool(x)</code> is True	
assertFalse(x)	<code>bool(x)</code> is False	
assertIs(a, b)	<code>a is b</code>	3.1
assertIsNot(a, b)	<code>a is not b</code>	3.1
assertIsNone(x)	<code>x is None</code>	3.1
assertIsNotNone(x)	<code>x is not None</code>	3.1
assertIn(a, b)	<code>a in b</code>	3.1
assertNotIn(a, b)	<code>a not in b</code>	3.1
assertIsInstance(a, b)	<code>isinstance(a, b)</code>	3.2
assertNotIsInstance(a, b)	<code>not isinstance(a, b)</code>	3.2

Method	Checks that	New in
assertRaises(exc, fun, *args, **kwargs)	<code>fun(*args, **kwargs)</code> raises <code>exc</code>	
assertRaisesRegex(exc, r, fun, *args, **kwargs)	<code>fun(*args, **kwargs)</code> raises <code>exc</code> and the message matches regex <code>r</code>	3.1
assertWarns(warn, fun, *args, **kwargs)	<code>fun(*args, **kwargs)</code> raises <code>warn</code>	3.2
assertWarnsRegex(warn, r, fun, *args, **kwargs)	<code>fun(*args, **kwargs)</code> raises <code>warn</code> and the message matches regex <code>r</code>	3.2
assertLogs(logger, level)	The <code>with</code> block logs on <code>logger</code> with minimum <code>level</code>	3.4
assertNoLogs(logger, level)	The <code>with</code> block does not log on <code>logger</code> with minimum <code>level</code>	3.10

Method	Checks that	New in
assertAlmostEqual(a, b)	<code>round(a-b, 7) == 0</code>	
assertNotAlmostEqual(a, b)	<code>round(a-b, 7) != 0</code>	
assertGreater(a, b)	<code>a > b</code>	3.1
assertGreaterEqual(a, b)	<code>a >= b</code>	3.1
assertLess(a, b)	<code>a < b</code>	3.1
assertLessEqual(a, b)	<code>a <= b</code>	3.1
assertRegex(s, r)	<code>r.search(s)</code>	3.1
assertNotRegex(s, r)	<code>not r.search(s)</code>	3.2
assertCountEqual(a, b)	<code>a</code> and <code>b</code> have the same elements in the same number, regardless of their order.	3.2

What's the deal?

First, plural vs. singular

- `assertEquals` and `assertEqual` are the same thing
- Python 2.6 introduced the aliased singular to help with clarity and deprecated the plural
- No one in the python community wants to enforce this until the next major release or something

Second, equal vs. almost equal

- `assertEqual` is precise while `assertAlmostEqual` allows for decimal precision
- You can override the default 7 decimals of precision

```
assertAlmostEqual(first, second, places=7, msg=None, delta=None)
```

Success!

Wait, that looks like 2 tests! Huh?

That's better

```
import unittest
import circle_calc_01 as circ
from math import pi

class TestCircles(unittest.TestCase):
    """ Class for testing circles """

    def test_circle_area(self):
        result = circ.circle_area(5)
        #self.assertEqual(result, 78.54)
        self.assertAlmostEqual(result, 78.5398163)
        self.assertAlmostEqual(result, 78.54, places=2)
```



```
Windows PowerShell
PS C:\Users\MikeP\PycharmProjects\python265\module05> python -m unittest test_circle_calc_01.py
.
-----
Ran 1 test in 0.000s

OK
PS C:\Users\MikeP\PycharmProjects\python265\module05>
```

Are we done? Maybe not

Some possible scenarios are

Radius = -5 or True or "Hello"

Let's add some more tests

ValueError?

If the radius passed in is negative, function should raise a `ValueError`

Go back to code...

```
def test_circle_area(self):  
    result = circ.circle_area(5)  
    # self.assertEqual(result, 78.54)  
    self.assertAlmostEqual(result, 78.5398163)  
    self.assertAlmostEqual(result, 78.54, places=2)  
    self.assertRaises(ValueError, circ.circle_area, -5)
```



```
Run Python tests in test_circle_calc_01.py x  
Tests failed: 1, passed: 1 of 2 tests - 15 ms  
Testing started at 12:00 AM ...  
Launching unittests with arguments python -m unittest C:\Users\MikeP\PycharmProjects\python265\Module05\test_circle_calc_01.py  
Ran 2 tests in 0.017s  
FAILED (failures=1)  
Failure  
Traceback (most recent call last):  
  File "C:\Users\MikeP\PycharmProjects\python265\Module05\test_circle_calc_01.py", line 21, in test_circle_area  
    self.assertRaises(ValueError, circ.circle_area, -5)  
AssertionError: ValueError not raised by circle_area
```


Iterative development

Tests Pass! Woohoo!

```
def circle_area(r):  
    """ calculate the area of a circle """  
  
    if r < 0:  
        raise ValueError("No negative numbers, dude!")  
    area = pi * r**2  
    return area
```

```
✓ Tests passed: 2 of 2 tests - 6 ms  
C:\Users\MikeP\PycharmProjects\python265\venv\Scripts  
Testing started at 11:38 AM ...  
Launching unittests with arguments python -m unittest  
  
Ran 2 tests in 0.008s  
  
OK  
  
Process finished with exit code 0
```

What if Boolean or string passed?

Raise type error

```
def test_circle_area(self):  
    result = circ.circle_area(5)  
    # self.assertEqual(result, 78.54)  
    self.assertAlmostEqual(result, 78.5398163)  
    self.assertAlmostEqual(result, 78.54, places=2)  
    self.assertRaises(ValueError, circ.circle_area, -5)  
    self.assertRaises(TypeError, circ.circle_area, True)  
    self.assertRaises(TypeError, circ.circle_area, "Hello")
```



Run Python tests in test_circle_calc_01.py x

Test Results 10 ms

- test_circle_calc_01 10 ms
 - TestCircles 10 ms
 - test_circle_area 10 ms

Tests failed: 1, passed: 1 of 2 tests - 10 ms

Testing started at 11:48 AM ...

Launching unittests with arguments python -m unittest C:\Users\MikeP\PycharmProjects\python265\Module05\test_cir

Failure

Traceback (most recent call last):

File "C:\Users\MikeP\PycharmProjects\python265\Module05\test_circle_calc_01.py", line 22, in test_circle_area

```
self.assertRaises(TypeError, circ.circle_area, True)
```

AssertionError: TypeError not raised by circle_area

Ran 2 tests in 0.012s

FAILED (failures=1)

Fix code

Add type check and rerun the test

```
def circle_area(r):  
    """ calculate the area of c circle """  
  
    if r < 0:  
        raise ValueError("No negative numbers, dude!")  
    if type(r) == bool or type(r) == str:  
        raise TypeError("Invalid type, dude!")  
    area = pi * r**2  
    return area
```

✓ Tests passed: 2 of 2 tests – 8 ms

Launching unittests with arguments python -m unittest

Ran 2 tests in 0.009s

OK

See a pattern occurring?

- 1) Create a function/method (stub)
- 2) Write a failing test
- 3) Prove the test fails
- 4) Correct the code
- 5) Prove the test succeeds
- 6) Go back to step 2 until all possibilities are exhausted.



SW Dev == Guided Evolution

Classes?

Let's convert the first two methods to a class

- Added constructor to hydrate the object
- Functions migrated to properties

```
class CircleCalcs:
    """ Class for Circle Calculations """

    def __init__(self, r, l=0.0):
        self.r = r
        self.l = l

    @property
    def square_area(self):
        """ calculate the area of a square """

        area = self.l ** 2
        return area

    @property
    def circle_area(self):
        """ calculate the area of c circle """

        if self.r < 0:
            raise ValueError("No negative numbers, dude!")
        if type(self.r) == bool or type(self.r) == str:
            raise TypeError("Invalid type, dude!")
        area = pi * self.r ** 2
        return area
```

Once for all?

Add class methods for setup and tearDown

```
✓ Tests passed: 2 of 2 tests - 7ms
```

```
C:\Users\MikeP\PycharmProjects\py  
Testing started at 3:22 PM ...  
Launching unittests with argument:
```

```
class setUp called
```

```
Ran 2 tests in 0.010s
```

```
OK
```

```
setUp called  
test_circle called  
tearDown called  
setUp called  
test_square called  
tearDown called  
class tearDown called
```

```
class TestCircleCalcs(unittest.TestCase):
```

```
@classmethod  
def setUpClass(cls):  
    print('class setUp called')
```

```
@classmethod  
def tearDownClass(cls):  
    print('class tearDown called')
```

```
def setUp(self):  
    self.circle1 = CircleCalcs(5,10)  
    self.circle2 = CircleCalcs(5,13)  
    print("setUp called")
```

```
def test_square(self):
```

```
    self.assertEqual(self.circle1.square_area, 100)  
    self.assertEqual(self.circle2.square_area, 169)  
    print("test_square called")
```

```
def test_circle_area(self):
```

```
    self.assertAlmostEqual(self.circle1.circle_area, 78.5398163)  
    self.assertAlmostEqual(self.circle2.circle_area, 78.54, places=2)  
    print("test_circle called")
```

```
    circle3=CircleCalcs(-5,0)
```

```
    circle4 = CircleCalcs(True, 0)
```

```
    circle5 = CircleCalcs("Hello", 0)
```

```
    self.assertRaises(ValueError, lambda: circle3.circle_area)
```

```
    self.assertRaises(TypeError, lambda: circle4.circle_area)
```

```
    self.assertRaises(TypeError, lambda: circle5.circle_area)
```

```
def tearDown(self):  
    print("tearDown called")
```

This is not just for OSS languages

RPG has Unit Testing capabilities

Cannot unit test a monolith

Sub-procedures!!!

Marina Schwenk for more details

Product Team Enterprise Explore Marketplace Pricing Search

MarinaSchwenk / IBMiUnit Public

<> Code Issues 8 Pull requests 2 Actions Projects Wiki Security Insights

master 1 branch 0 tags Go to file Code

StevensJE v1.1 04d2dea on Nov 15, 2019 9 commits

src	v1.1	2 years ago
.gitattributes	v1.1	2 years ago
.gitignore	v1.1	2 years ago
Fall19 Intro to Unit testing using IB...	v1.1	2 years ago
IBMIUNIT.SAVF	v1.1	2 years ago
IBMiUnit Developer Documentatio...	v1.1	2 years ago
LICENSE	Create LICENSE	3 years ago
README.md	v1.1	2 years ago
RELEASE NOTES.md	v1.1	2 years ago
build	v1.1	2 years ago

README.md

IBMiUnit

An RPGLE unit testing framework

Dependencies

Where to from here?

Try this in WHATEVER languages you use.

Language	Testing Framework	URL
JavaScript	Jest	https://jestjs.io/
Python	unittest	https://docs.python.org/3/library/unittest.html
RPG	IBMiUnit	https://github.com/MarinaSchwenk/IBMiUnit
COBOL	Cobol-check	https://github.com/openmainframeproject/cobol-check
Java	JUnit	https://junit.org/junit4/
PHP	PHPUnit	https://phpunit.de/