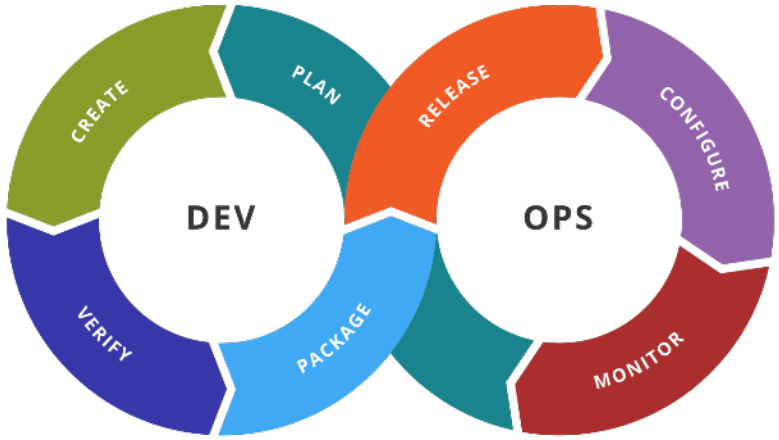
Why are we talking about testing?

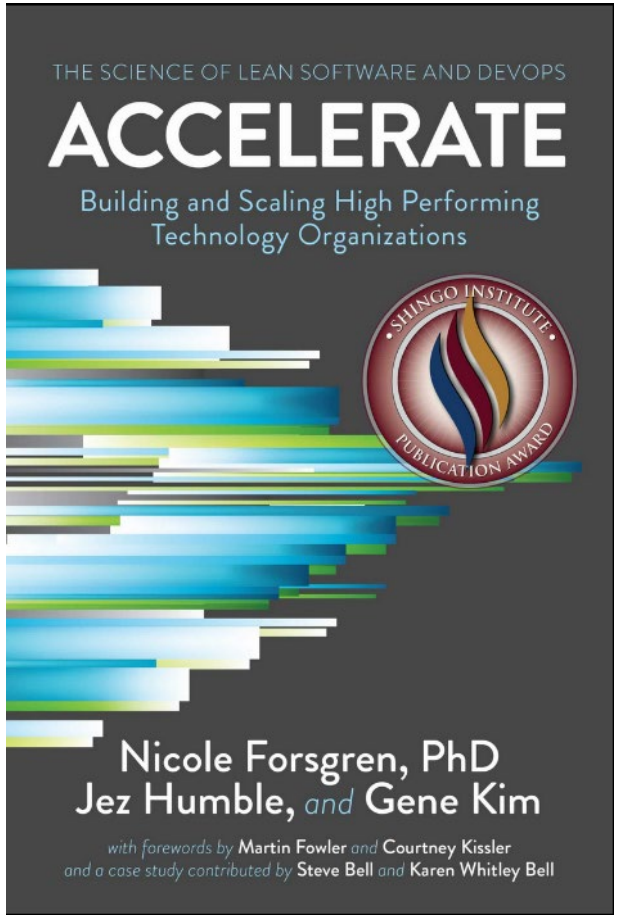
On board to DevOps and automation
- Faster feedback cycle
- Reusability
- Accelerate time to market

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

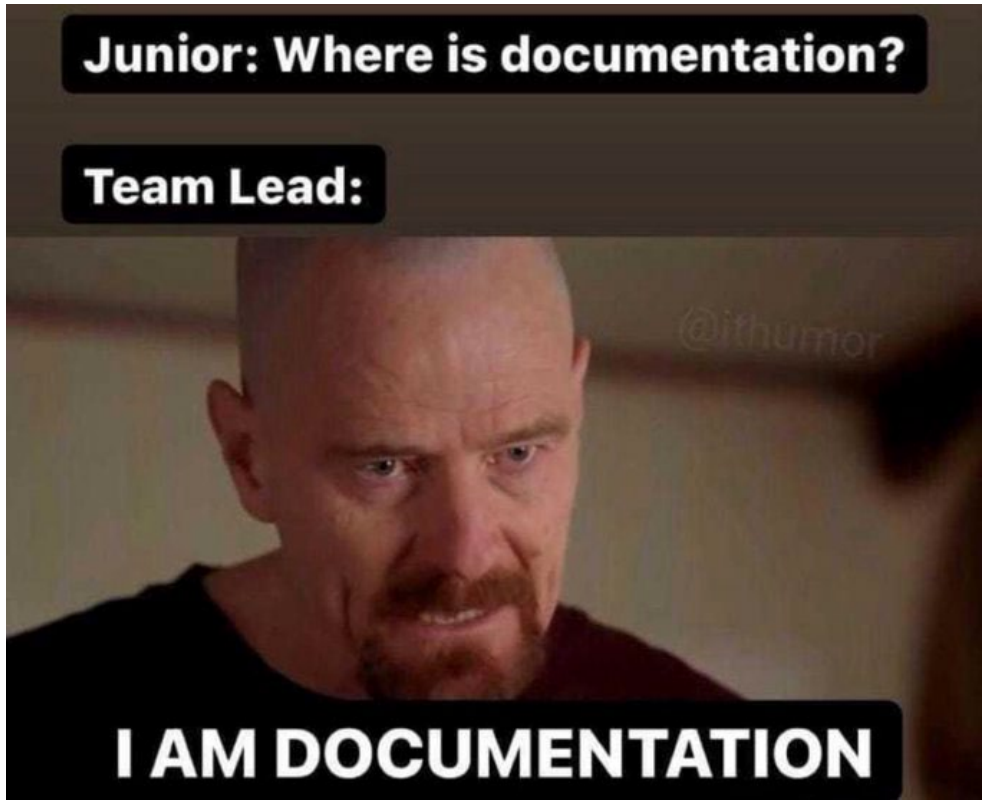
- Retiring
- Moving to new tech / platforms

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

https://peps.python.org/pep-0008/

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?

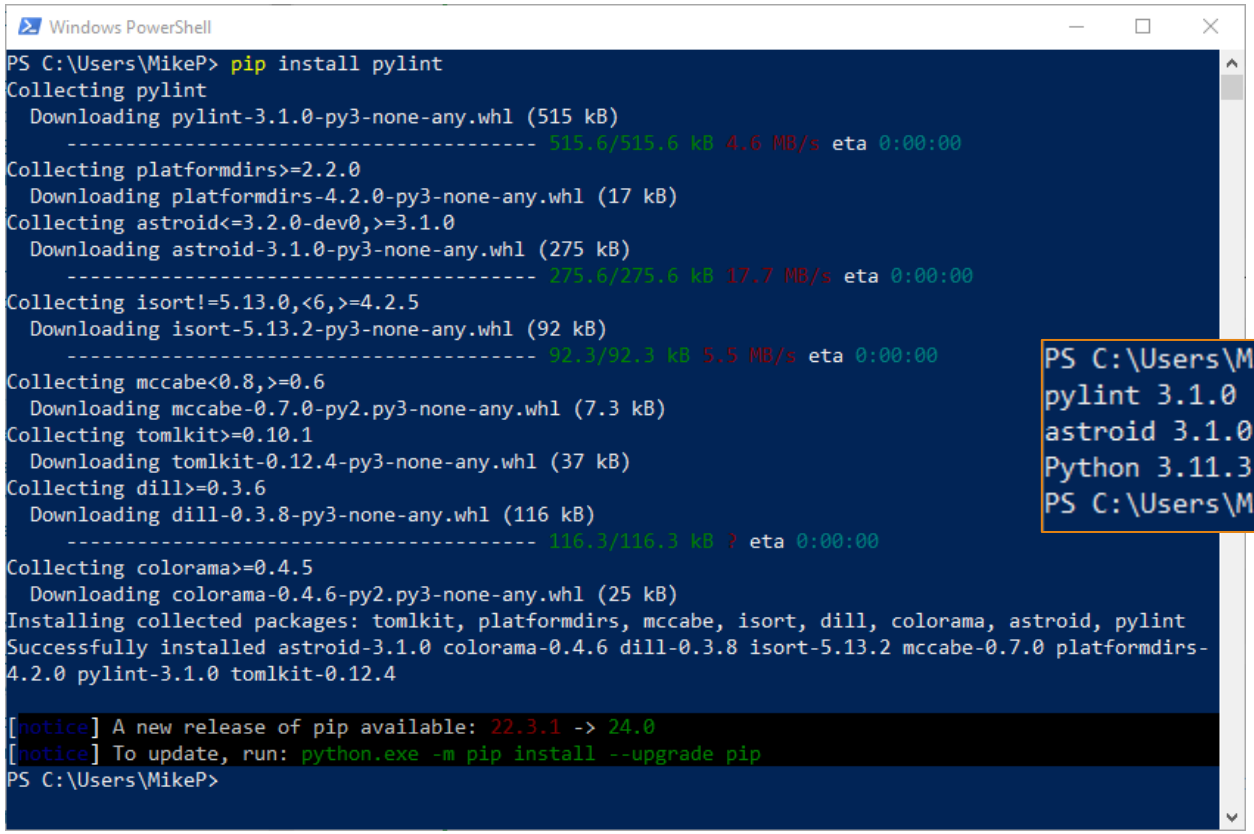
- Interoperability!

https://pylint.readthedocs.io/en/stable/

How do you know if you are OK?

PyLint module (third party module)

```
pip install pylint
```




```
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>
```

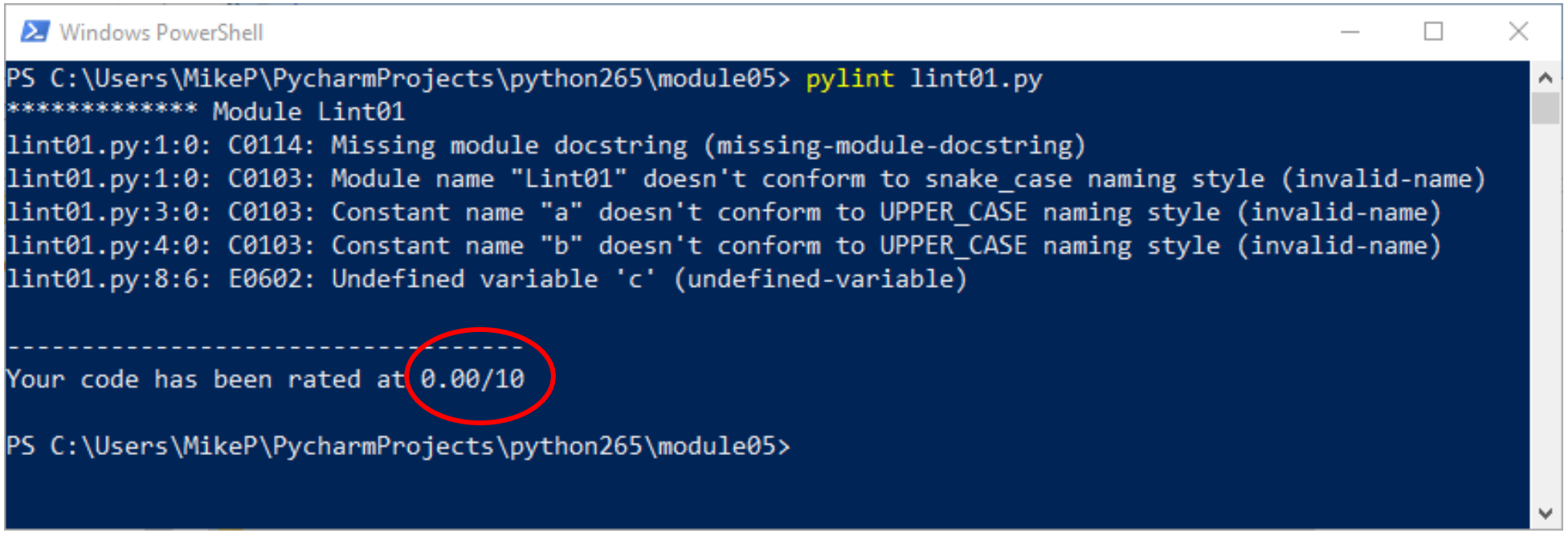
Short program

Snippet of code to exploring the linter

Use command line to “lint” the file

```
a = 1
b = 2

print(a)
print(b)
print(c)
```



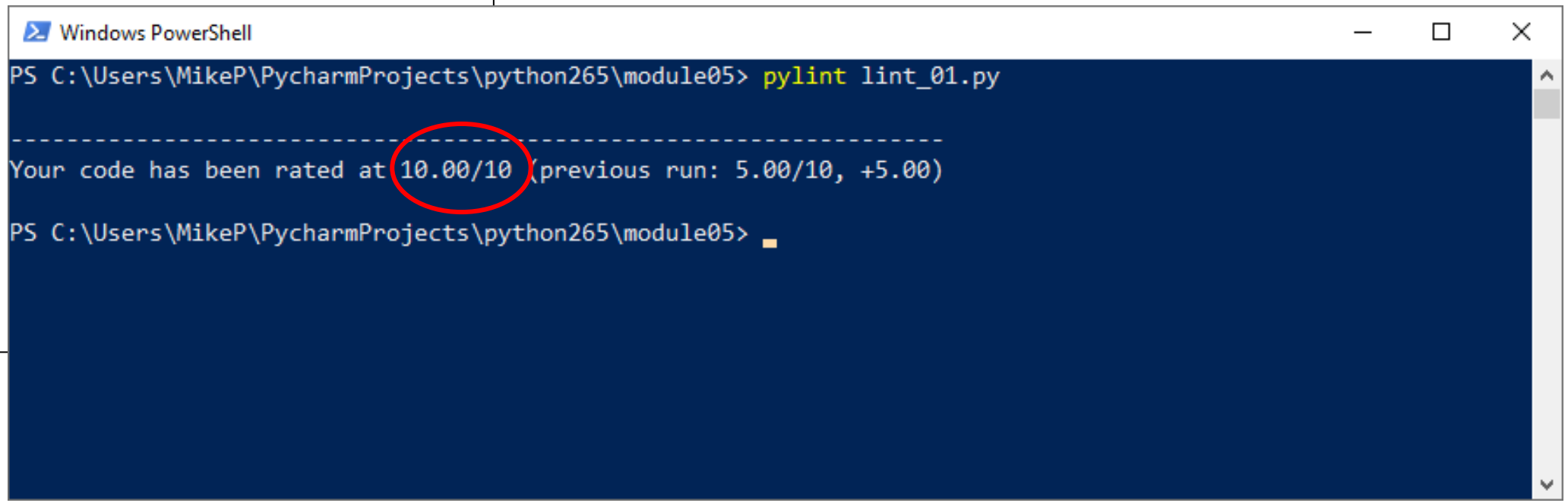
And all better

Upper case the constants…

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


A = 1
B = 2
C = 3

print(A)
print(B)
print(C)
```



Fundamentals

TDD: Test Driven Development

Different approach to developing require some basic assumptions:

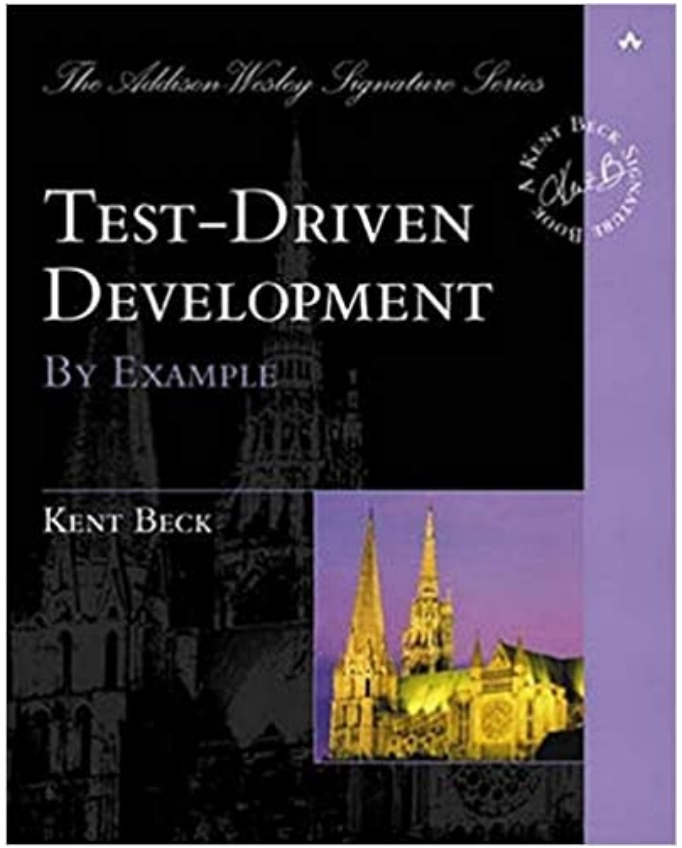
- Code is loosely coupled
- Part of a bigger system
- Tests are for components, not necessarily programs
  - Functions and methods
- Iterative development model
- Tests are written before code….(Wahhhh?)

Conceptualized by Kent Beck in his book

- Test Driven Development By Example

Does not replace UAT

- Reduces it
- Automates unit testing



https://www.agilealliance.org/glossary/tdd/#

TDD: Test Driven Development (Why?)

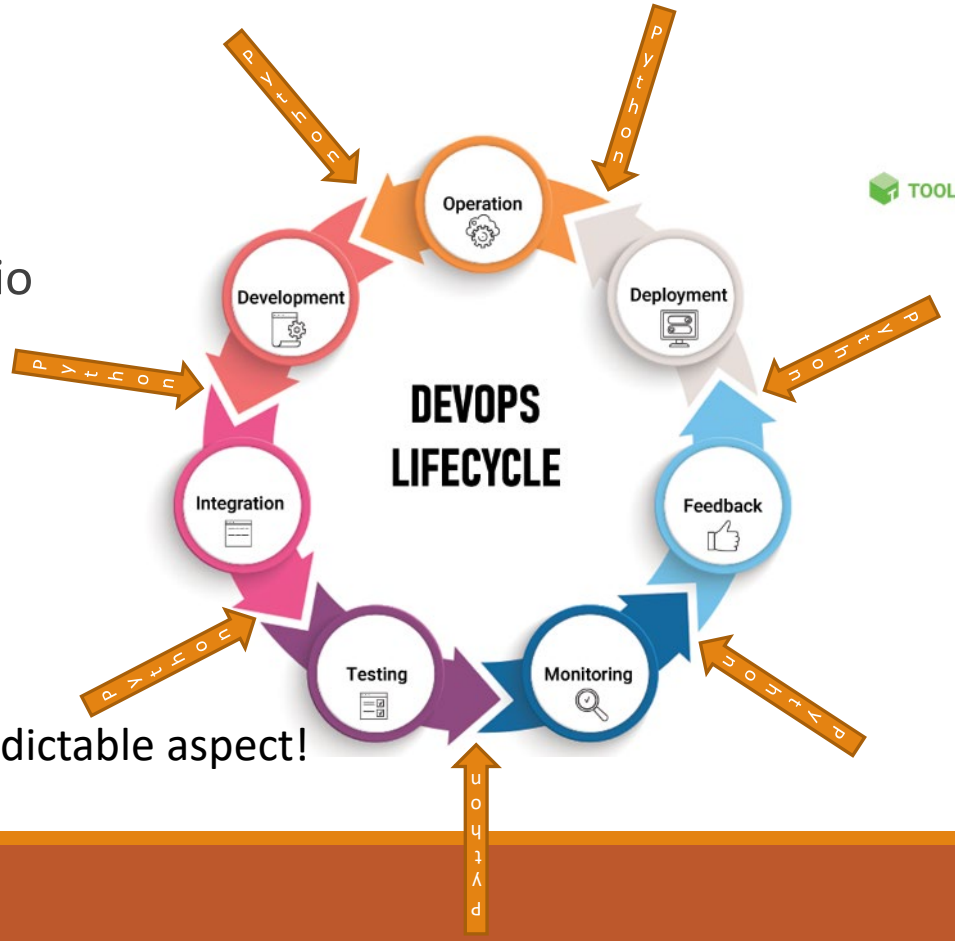
DevOps is the way forward

- 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



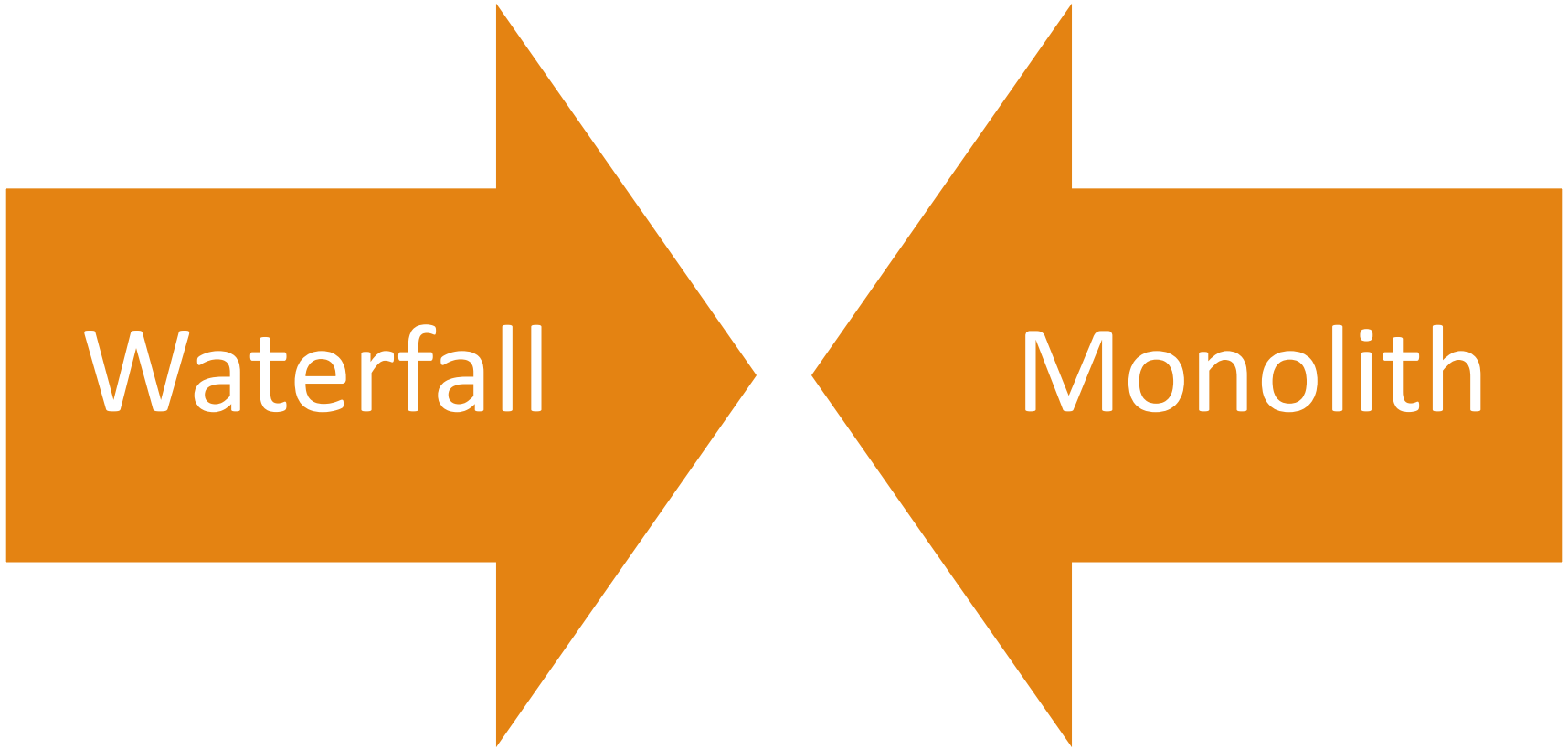
Expected Benefits

- 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

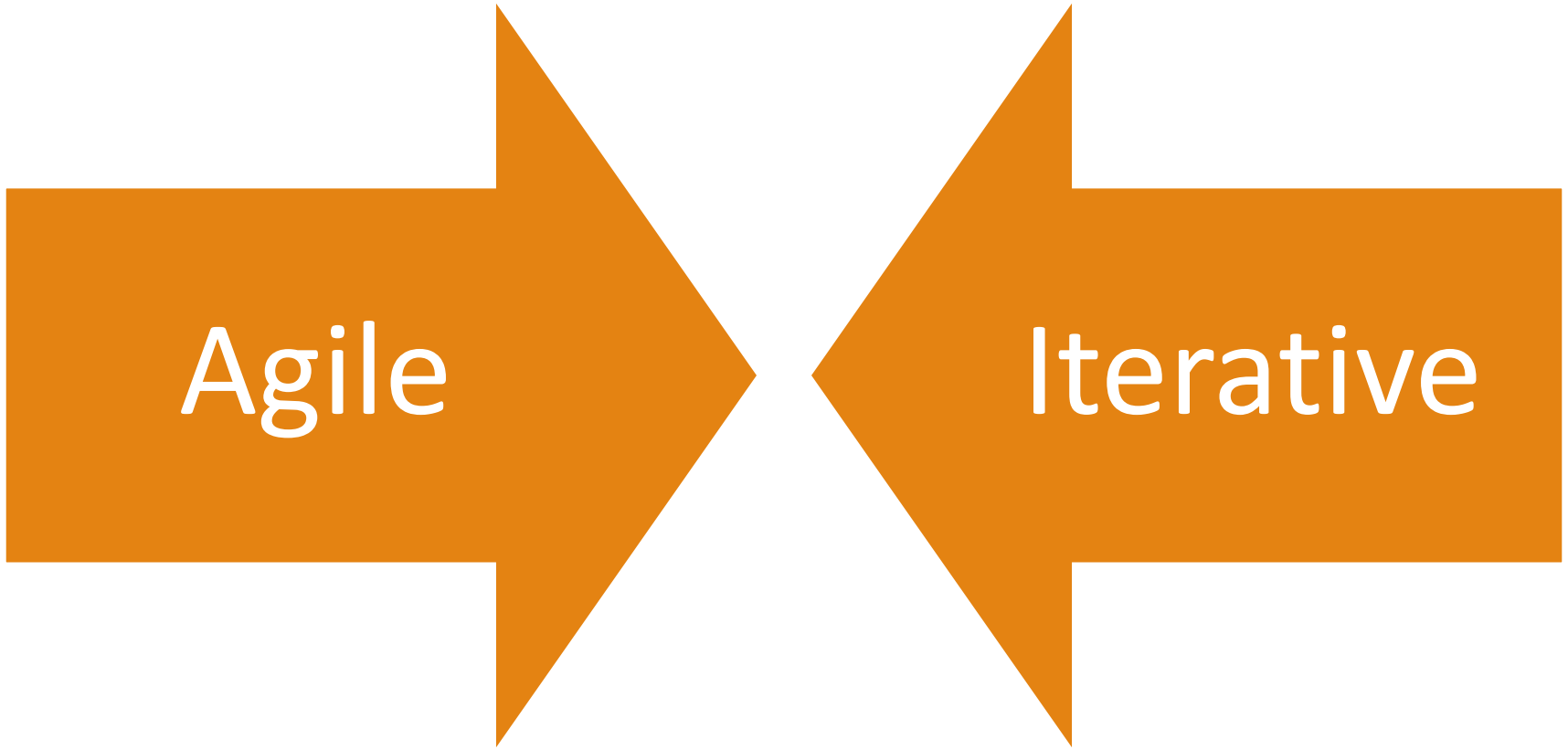
Agile Alliance

Testing is the most unpredictable aspect!

Simple Context Part 1: Legacy

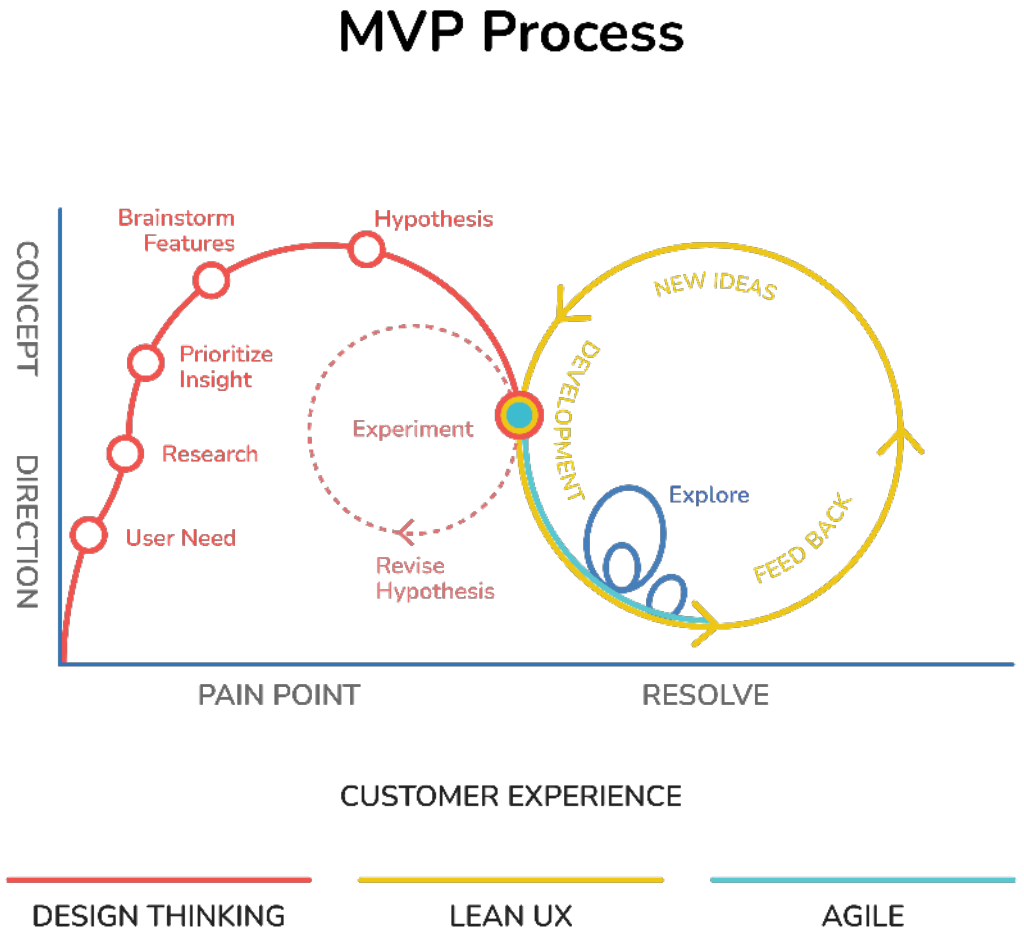


Simple Context Part 2: Modern



https://www.datasciencecentral.com/how-to-build-a-minimum-viable-product-mvp-a-detailed-guide/

Getting started with Agile



A Python Example of TDD

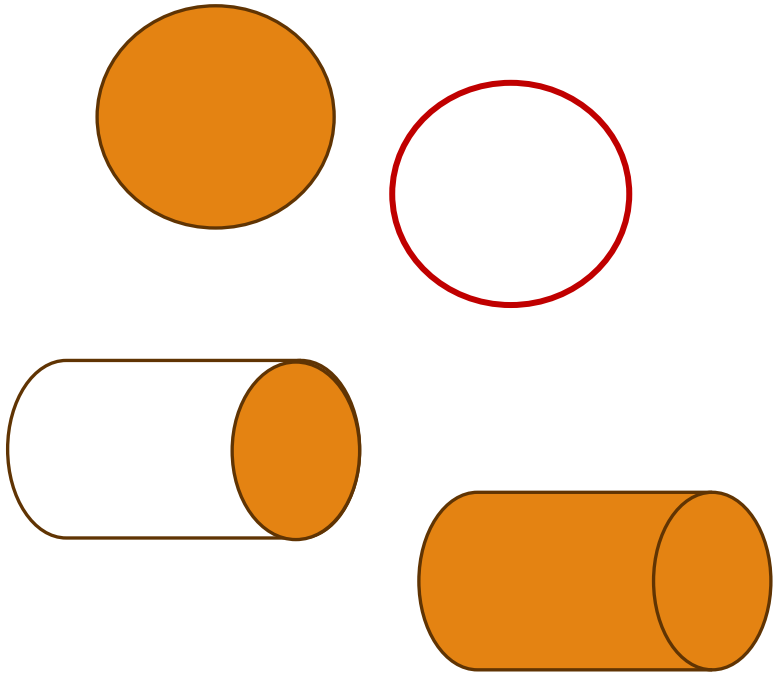
Start with basic circle functions

Area of a circle – πr^2

Circumference of a circle – $2\pi r$

Volume of a cylinder – $\pi r^2 h$

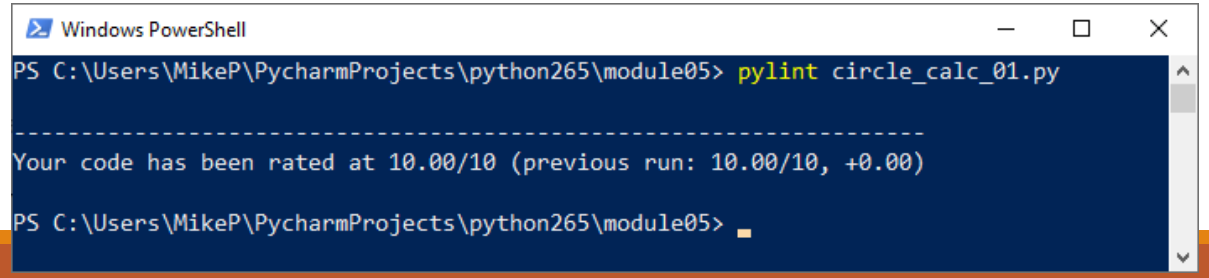
Surface area of a cylinder - $2\pi rh + 2\pi r^2$



Function file

Import pi from the math module

Define the function and save in a file




```
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

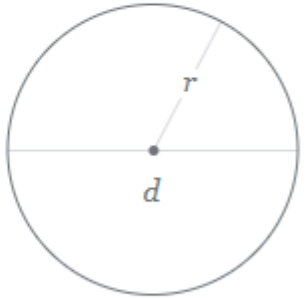
Hey Google!

Calculators ›

Solve for
area

$A \approx 78.54$

r Radius 5



Solution

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

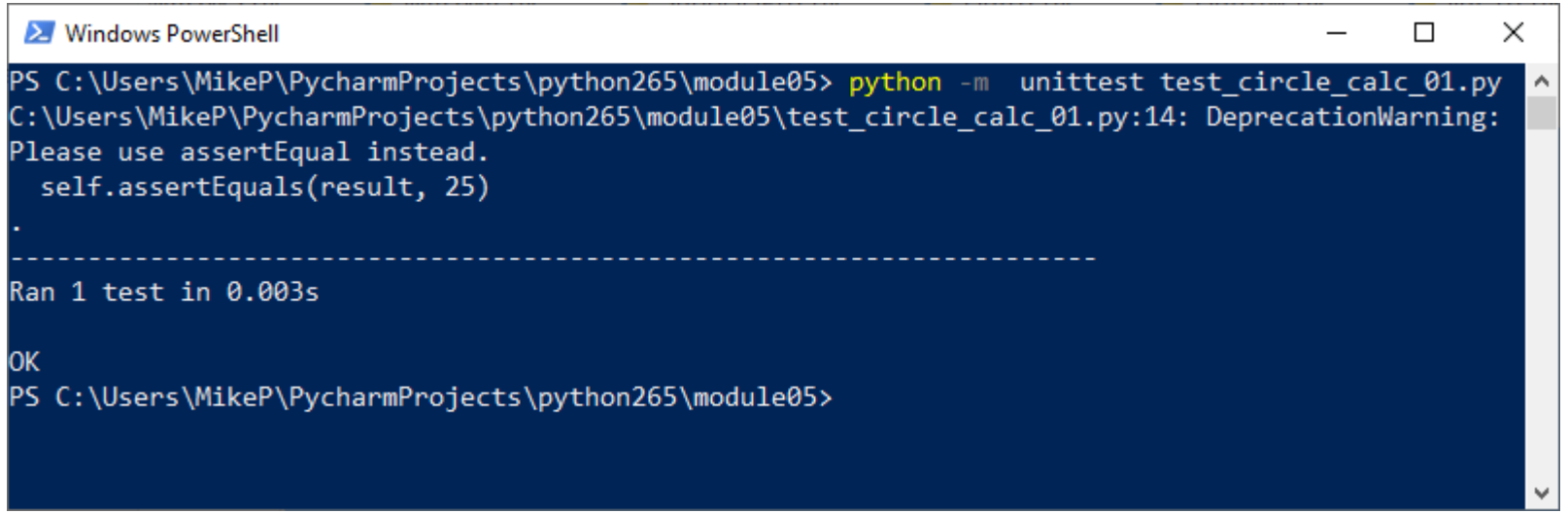
Start with the square

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.assertEquals(result, 78.54)
        self.assertEquals(result, 25)
```



Now let's do a circle

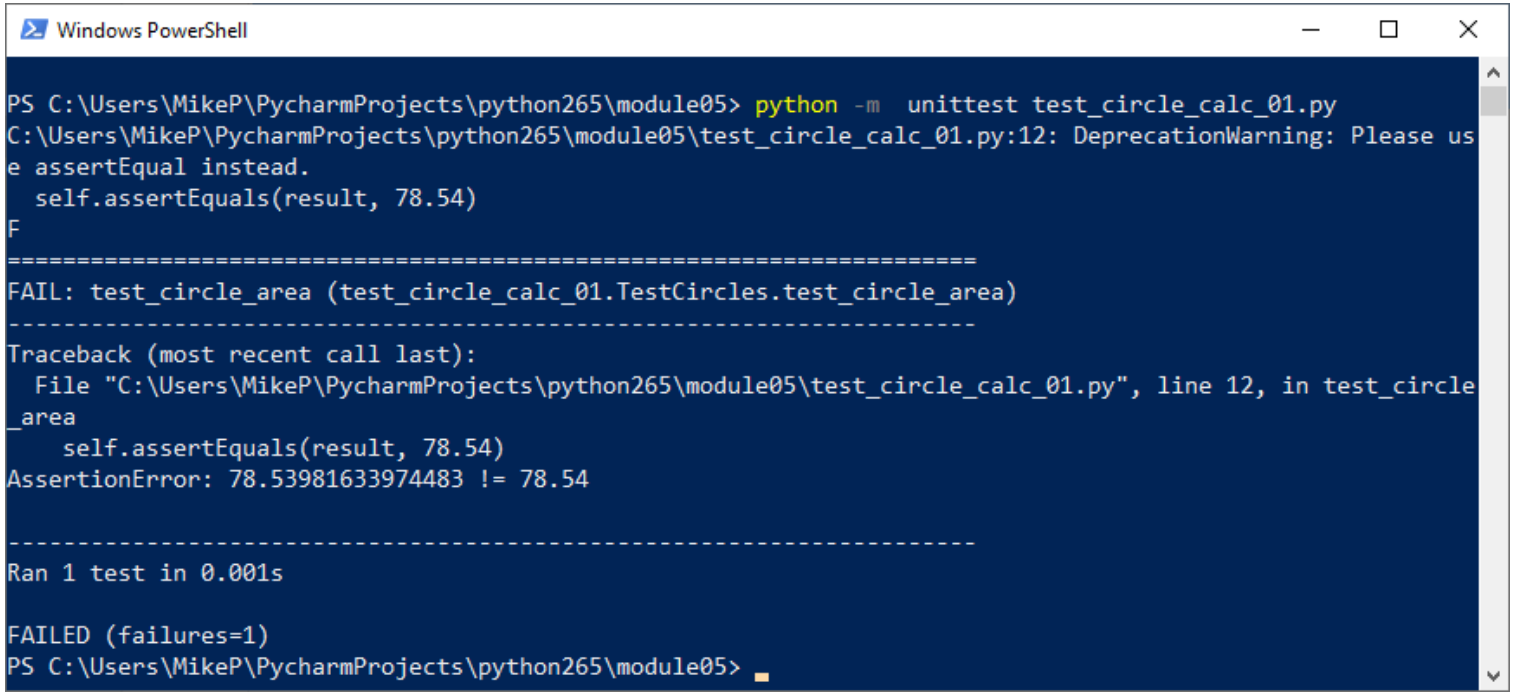
test_nnnnn

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.assertEquals(result, 78.54)
```



https://docs.python.org/3/library/unittest.html

Lots of asserts…

Linter in PyCharm

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

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

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

https://docs.python.org/3/library/unittest.html#unittest.TestCase.assertAlmostEqual

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

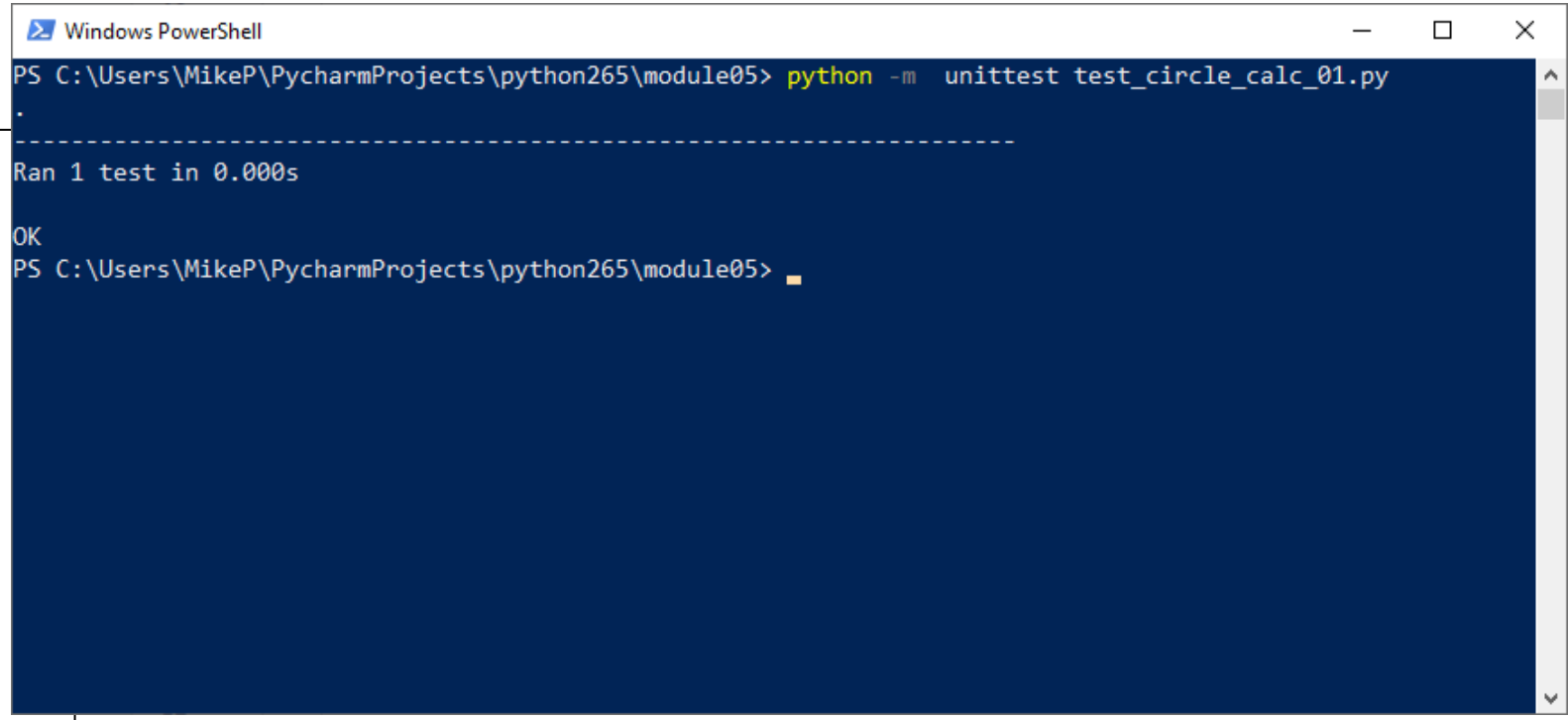
```
import unittest
import circle_calc_01 as circ
from math import pi


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

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

Test

Assertions



Are we done? Maybe not

Some possible scenarios are

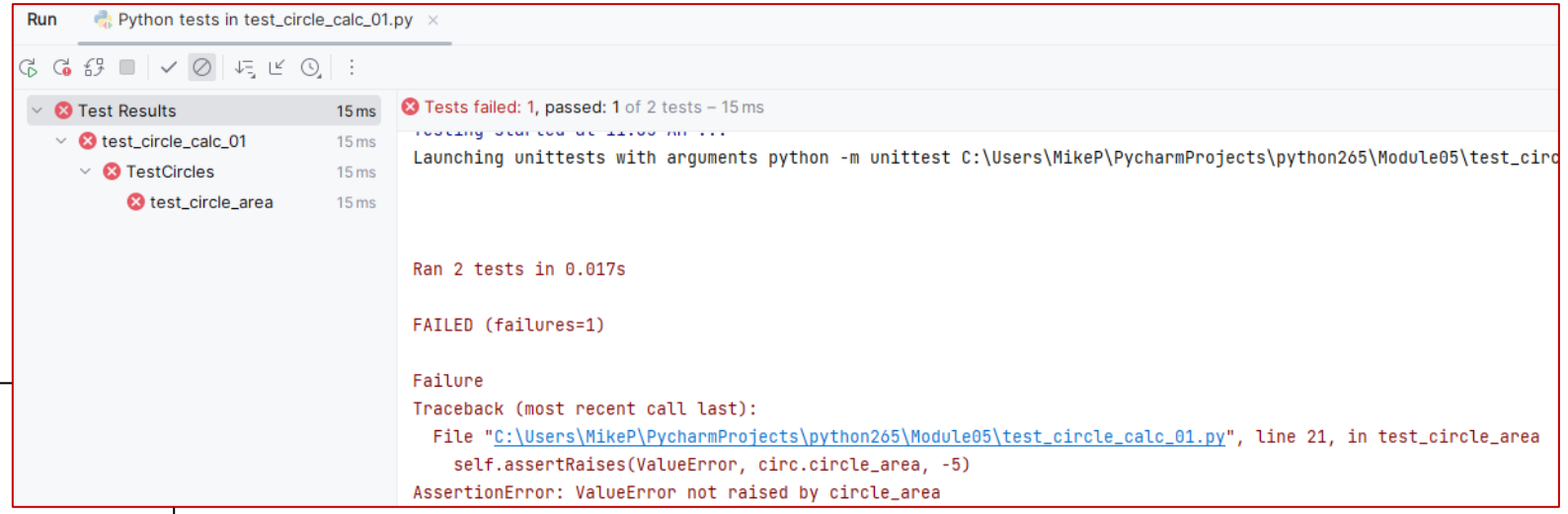
Radius = -5 or True or “Hello”

Let’s add some more tests

ValueError?

If the radius passed in is negative, function should raise aValueError

Go back to code…



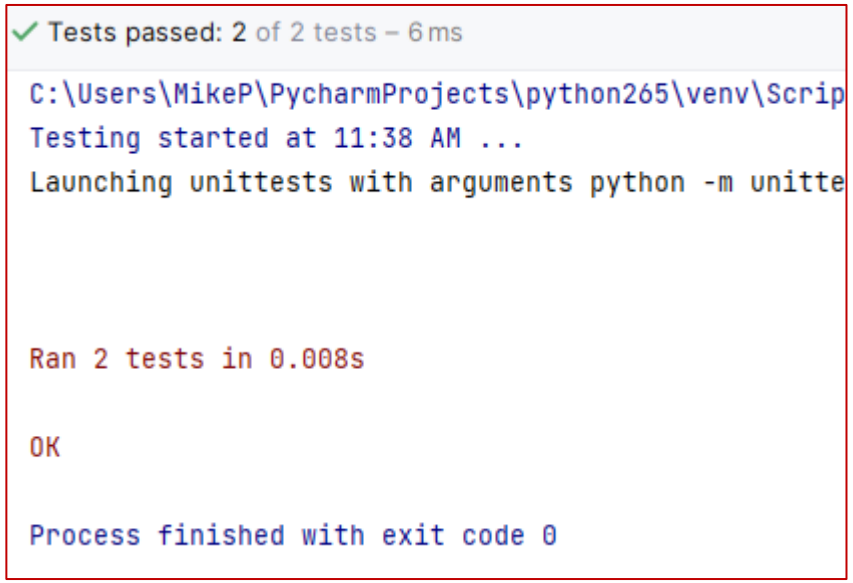

```
def test_circle_area(self):
    result = circ.circle_area(5)
    # self.assertEquals(result, 78.54)
    self.assertAlmostEqual(result, 78.5398163)
    self.assertAlmostEqual(result, 78.54, places=2)
    self.assertRaises(ValueError, circ.circle_area, -5)
```

Iterative development

Tests Pass! Woohoo!

```
def circle_area(r):
    """ calculate the area of c circle """

    if r < 0:
        raise ValueError("No negative numbers, dude!")
    area = pi * r**2
    return area
```



What if Boolean or string passed?

Raise type error

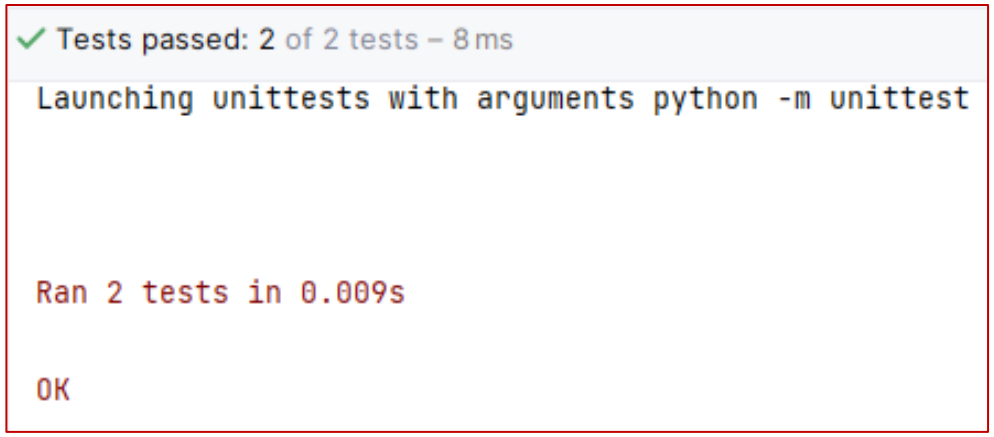

```
def test_circle_area(self):
    result = circ.circle_area(5)
    # self.assertEquals(result, 78.54)
    self.assertAlmostEqual(result, 78.5398163)
    self.assertAlmostEqual(result, 78.54, places=2)
    self.assertRaises(ValueError, circ.circle_area, -5)
    self.assertRaises(TypeError, circ.circle_area,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

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
```



https://youtu.be/-f_HgWbomCI

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 – 7 ms
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")
```

https://github.com/MarinaSchwenk/IBMiUnit

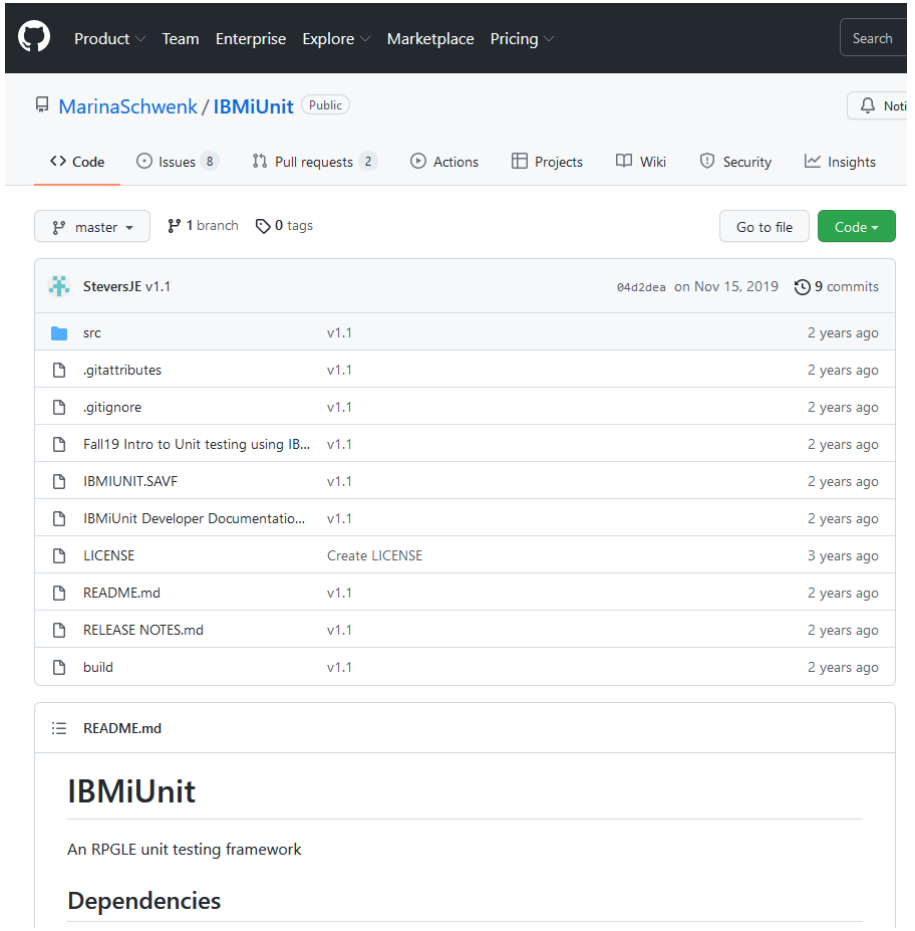
This is not just for OSS languages

RPG has Unit Testing capabilities

Cannot unit test a monolith

Sub-procedures!!!

Marina Schwenk for more details



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/