Make your app work

Testing Python applications

Who am I

- CTO at Metwit
- Long time Python developer
- Contributor to the Python project

Your code

Your code

Stops working

when **you** don't expect it to

Your code

Your code

Does not do

what **you** intend it to do

Challenges

Challenges

- Code is complex

Challenges

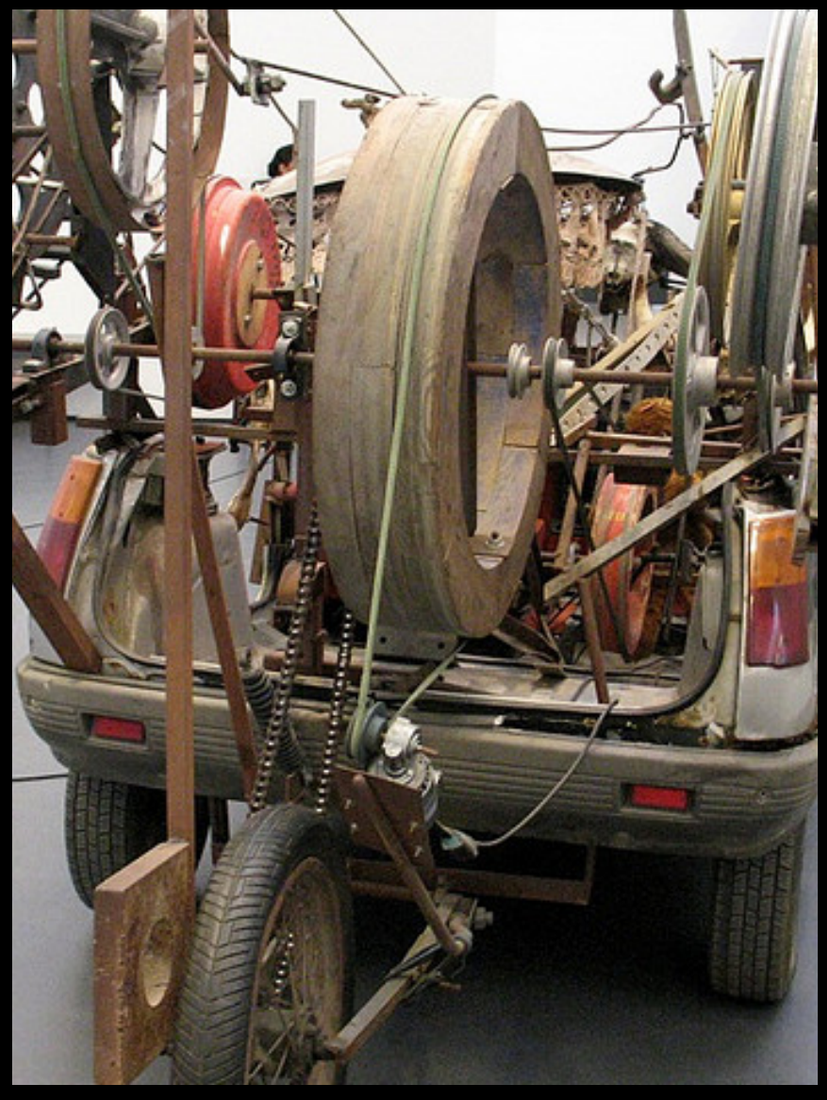
- Code is complex
- Interdependent parts

Challenges

- Code is complex
- Interdependent parts
- Requirements do change

Challenges

- Code is complex
- Interdependent parts
- Requirements do change



Added challenge

```
def area(w, h):
    return w * h


def main():
    w = 2.5
    h = input('height?')
    result = area(w, h)
    print('area:', result)
```

Added challenge

```
def area(w, h):
    return w * h


def main():
    w = 2.5
    h = input('height?')
    result = area(w, h)
    print('area:', result)
```

No compile time type checking!

Testing

- Run the code.
- Only way to ensure code behaves right.

If it's not tested, it's

broken

–Bruce Eckel

How to test

- Automated testing
- Write code that tests your code

```
>>> area(2, 5)
10
>>> area(0, 10)
0
>>> area(1.5, 0.25)
0.375
```

```
def area(w, h):
    """
    Return the area of a rectangle
    with the given width and height.

    >>> area(2, 5)
    10
    >>> area(0, 10)
    0
    >>> area(1.5, 0.25)
    0.375
    """
    return w * h
```

doctest

```
python -m doctest surface.py
```

- Simplest way to write tests in Python
- Encourages documentation

Test-driven

Test-driven

1. Write the tests

Test-driven

1. Write the tests

See the tests FAIL

Test-driven

1. Write the tests

See the tests FAIL

2. Write the code

Test-driven

1. Write the tests

   See the tests FAIL

2. Write the code

   Make the tests SUCCEED

test-driven example

Unit testing

- smallest meaningful piece of code
- success or failure depends only on itself

unittest

```
python -m unittest module
```

- Define `TestCases`
- Allows complex setups

```
class RectangleTest(TestCase):
    def setUp(self):
        self.rect = Rectangle(10, 20)

    def test_perimeter(self):
        self.assertEqual(
            self.rect.perimeter(),
            60)

def test_area(self):
    self.assertEqual(
        self.rect.area(),
        200)
```

Test fixtures

- Clean environment for each test
- `setUp()` and `tearDown()`

unittest example

Interdependence

- Units might be *tight-coupled*
- **B** depends on **A**'s result
- ...but **A**'s result cannot be predicted.

Mock

- Simulate **A**'s behaviour
- Test **B** with simulated **A**

```
class Clock(object):
    def time(self):
        return time.clock()

class MockClock(object):
    def __init__(self):
        self.count = 0

    def time(self):
        self.count += 1
        return self.count
```

Learn more...

- `unittest` documentation
- `mock` library (included in 3.3)
- `nose` test runner

More Python...

CPython

- Original Python
- Reference Python
- Python 2 -> 2.7, maintainance only
- Python 3 -> 3.2, 3.3 alpha

Other implementations

- Jython runs on Java runtime
- IronPython runs on .NET runtime

Other implementations

- PyPy is a Python... written in Python
- ...and faster than Python
- Next reference implementation?

Applications

- Django: most popular web framework
- Twisted: network framework
- NLTK: natural language toolkit

Applications

- NumPy and SciPy
- Efficient numerical computation libraries
- matplotlib to do graphical stuff