Unit Tests in Python

Automated tests

An *automated test* is a program that tests another program.

Some of you have some experience with this already.

There are different flavors, including:

- **Unit test** - Tests a specific component (function, class, etc.) in isolation. Checks for a specific response to a particular input.
- **Integration test** - Checks that several components or sub-systems work together correctly. The *interactions* are being tested, more than the individual parts.

And others too.

We focus on the ideas common to all automated test types.

A Simple Test

Let's write an automated test for this function:

```
# Split a number into portions, as evenly as possible. (But it has a bug.)
def split_amount(amount, n):
    portion, remain = amount // n, amount % n
    portions = []
    for i in range(n):
        portions.append(portion)
        if remain > 1:
            portions[-1] += 1
            remain -= 1
    return portions
```

How it ought to work:

```
>>> split_amount(4, 2)
[2, 2]
>>> split_amount(5, 3)
[2, 2, 1]
```

import unittest

Here's a basic unit test.

```
# test_splitting.py

from unittest import TestCase
from splitting import split_amount

class TestSplitting(TestCase):
    def test_split_amount(self):
        self.assertEqual([1], split_amount(1, 1))
        self.assertEqual([2, 2], split_amount(4, 2))
        self.assertEqual([2, 2, 1], split_amount(5, 3))
        self.assertEqual([3, 3, 2, 2, 2], split_amount(12, 5))
```

Running The Test

```
$ python3 -m unittest test_splitting.py
F
======================================================================
FAIL: test_split_amount (test_splitting.TestSplitting)
----------------------------------------
Traceback (most recent call last):
  File "test_splitting.py", line 8, in test_split_amount
    self.assertEqual([2, 2, 1], split_amount(5, 3))
AssertionError: Lists differ: [2, 2, 1] != [2, 1, 1]
First differing element 1:
2
1

- [2, 2, 1]
?     ^
+ [2, 1, 1]
?     ^
----------------------------------------
Ran 1 test in 0.001s
FAILED (failures=1)
```

Corrected Function

```
def split_amount(amount, n):
    'Split an integer amount into portions, as even as possible.'
    portion, remain = amount // n, amount % n
    portions = []
    for i in range(n):
        portions.append(portion)
        if remain > 0: # Was "remain > 1"
            portions[-1] += 1
            remain -= 1
    return portions
```

```
$ python3 -m unittest test_splitting.py
.
----------------------------------------
Ran 1 test in 0.000s

OK
```

Testing split_amount()

```python
# test_splitting.py

from unittest import TestCase
from splitting import split_amount

class TestSplitting(TestCase):
    def test_split_amount(self):
        self.assertEqual([1], split_amount(1, 1))
        self.assertEqual([2, 2], split_amount(4, 2))
        self.assertEqual([2, 2, 1], split_amount(5, 3))
        self.assertEqual([3, 3, 2, 2, 2], split_amount(12, 5))
```

```
$ python3 -m unittest test_splitting.py
.
----------------------------------------
Ran 1 test in 0.000s

OK
```

What's happening?

```
python3 -m unittest test_splitting.py
```

`unittest` is a standard library module. `test_splitting.py` is the file containing tests.

Inside is a class called `TestSplitting`. It subclasses `TestCase`.

(The name doesn't have to start with "Test", but often will.)

It has a method named `test_split_amount()`. That *test method* contains assertions.

Test methods **must** start with the string "test", or they won't get run.

Lab: Simple Unit Tests

Time to practice. You'll be challenged to develop code using test-driven development.

- In your `labs` folder
- Read and follow instructions: `tddlab.txt`