Testing for Data Scientists

EmboldenHer X AI Club

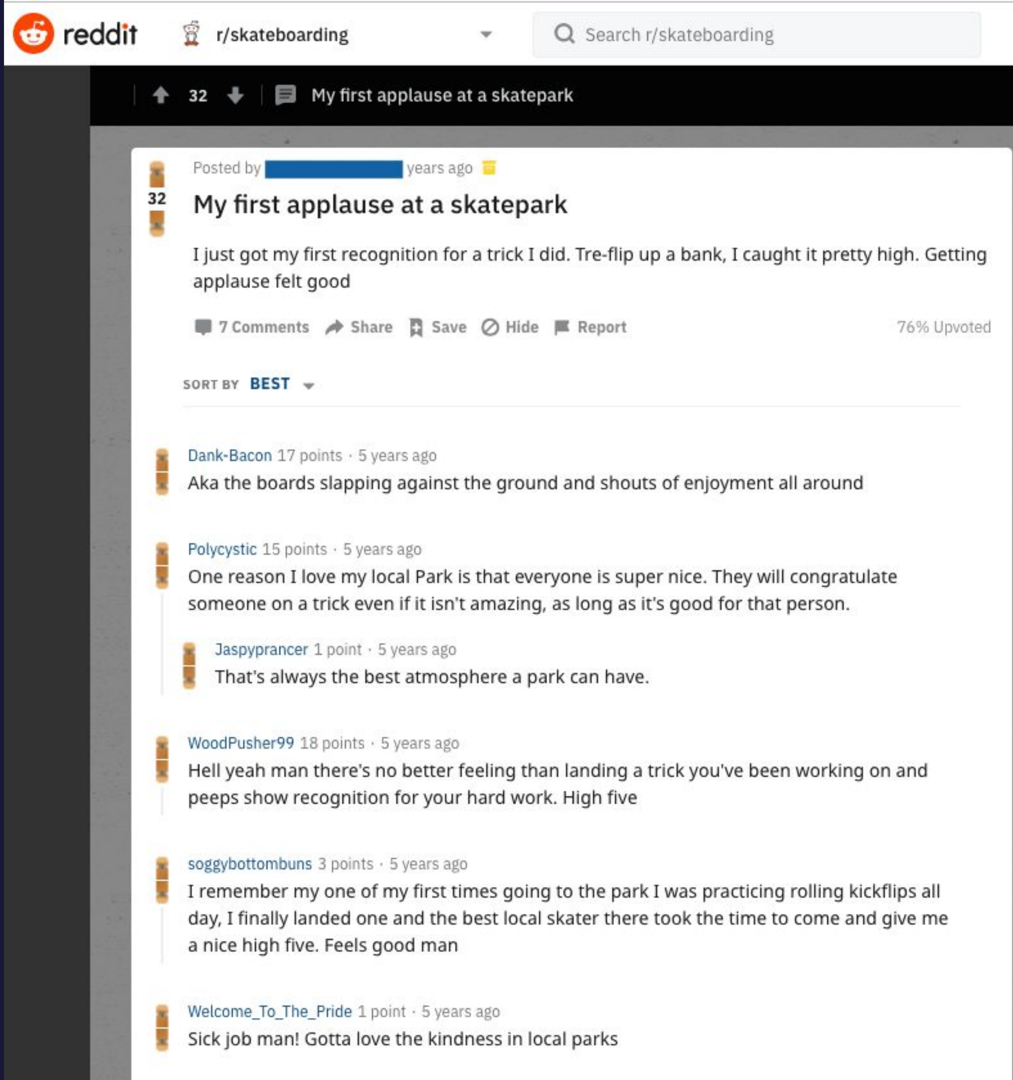
Tips from Software Engineering for Data Science!

Fei Phoon, Data Engineer at Kobalt Music

AI Club for Gender Minorities

Skateboarding's best kept secret

- Most of us don't know each other
- We may not have met before
- We don't have to know each other
- We are sharing the same space for the evening
- We're at all levels working on different things



reddit r/skateboarding Search r/skateboarding

32 My first applause at a skatepark

Posted by [redacted] years ago

32

My first applause at a skatepark

I just got my first recognition for a trick I did. Tre-flip up a bank, I caught it pretty high. Getting applause felt good

7 Comments Share Save Hide Report 76% Upvoted

SORT BY BEST

Dank-Bacon 17 points · 5 years ago
Aka the boards slapping against the ground and shouts of enjoyment all around

Polycystic 15 points · 5 years ago
One reason I love my local Park is that everyone is super nice. They will congratulate someone on a trick even if it isn't amazing, as long as it's good for that person.

Jaspyprancer 1 point · 5 years ago
That's always the best atmosphere a park can have.

WoodPusher99 18 points · 5 years ago
Hell yeah man there's no better feeling than landing a trick you've been working on and peeps show recognition for your hard work. High five

soggybottombuns 3 points · 5 years ago
I remember my one of my first times going to the park I was practicing rolling kickflips all day, I finally landed one and the best local skater there took the time to come and give me a nice high five. Feels good man

Welcome_To_The_Pride 1 point · 5 years ago
Sick job man! Gotta love the kindness in local parks

(We're actually starting the talk for real now)

- Data engineer of 9 months at Kobalt Music
- Working on a team of data engineers & data scientists
- Extremely supportive working and learning environment
- We're getting things done & we love what we do

What is data engineering?

- In charge of keeping data flowing into and making it available to the organisation
- Still a young field with a very broad range of definitions & skill sets, depending on what the data is, who the data needs to serve and how
- We architect, build and maintain the infrastructure that allows the organisation to access the data we receive
- Data science work is important, but we need to deliver it reliably at scale

(A decent explanation: https://www.digitalvidya.com/blog/data-engineering/)

Fact: Data engineers LOVE testing

Because:

- Our responsibility is to guarantee the quality & timeliness of data to everyone else
- A small, undetected issue with data at an early stage can quickly snowball into a huge problem downstream

Some idea of how paranoid we are

- Testing the applications we write to grab data from sources
- Testing that we know when unexpected data enters the pipeline
- Testing that we know when not enough data shows up
- Testing the code we write to process and join data for colleagues and customers to use downstream
- Testing that our pipelines (constructed of Amazon AWS microservices and code) work as we expect
- **Testing that all of the above things (and more) break exactly the way we need them to.**

The data engineer's wishlist (of their data scientist colleagues)

“Our data scientists are great,

but I wish they would write tests.”

(Item 1 of 1)

The data scientist's process

1. Discovery & proof of concept stage in a Jupyter notebook - data cleaning, feature extraction, etc
2. Throw notebooks over the wall, so an engineer can figure out how to productionise them.

   OR

2. Rewrite notebook work to the best of your abilities and pass it on to a data engineer.

Functions? Maybe, maybe not, haha!

Why write tests yourself?

- Catch bugs early
- Clearly communicate the expectations you have of your code, to your downstream colleagues
- The above, to your future self
- Less stress trying (and failing) to remember what scenarios you designed your code for. Gives you lasting confidence in your code reliability

A useful *useful test* checklist

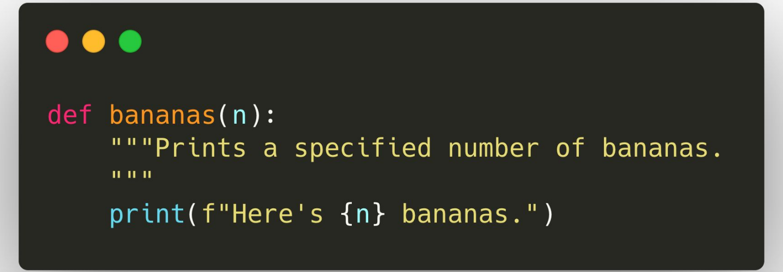
- Know your input & output: demands deterministic code
- Use a Given-When-Then pattern
- Assert that a result matches expectations

doctest

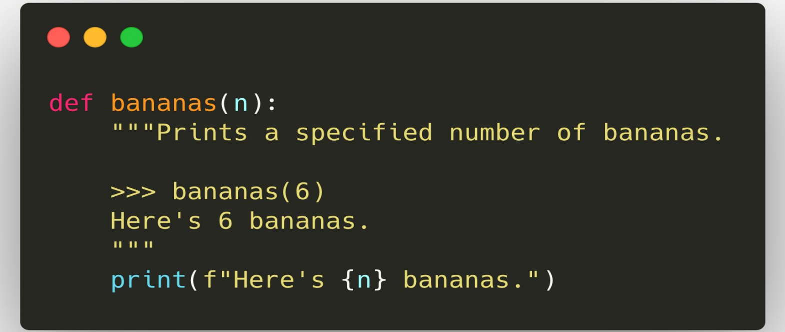
A module that runs input/output tests from your docstrings.

It looks in your docstrings for text resembling interactive Python sessions, and executes them to check if they work as shown.

https://docs.python.org/3.8/library/doctest.html



```
def bananas(n):
    """Prints a specified number of bananas.
    """
    print(f"Here's {n} bananas.")
```



```
def bananas(n):
    """Prints a specified number of bananas.

    >>> bananas(6)
    Here's 6 bananas.
    """
    print(f"Here's {n} bananas.")
```

doctest in Jupyter

Let's look at doctest-notebook.ipynb

http://localhost:8888/notebooks/doctest-notebook.ipynb

A quick demo of doctest

Let's look at math_functions.py

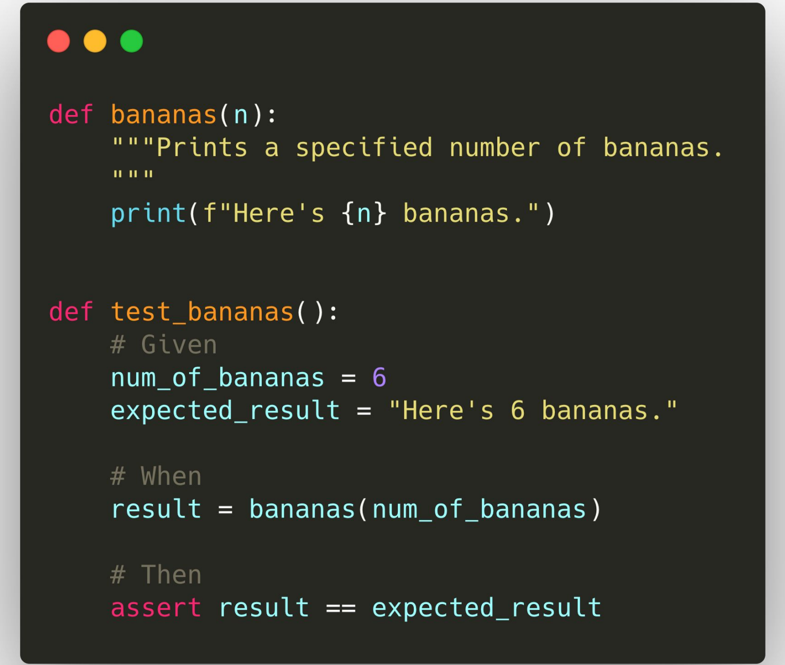
“Testing that all of the above things (and more) break exactly the way we want.”

pytest

A mature testing framework that supports small to complex tests.

Let's look at the first item in test_math_functions.py

https://docs.pytest.org/en/latest/



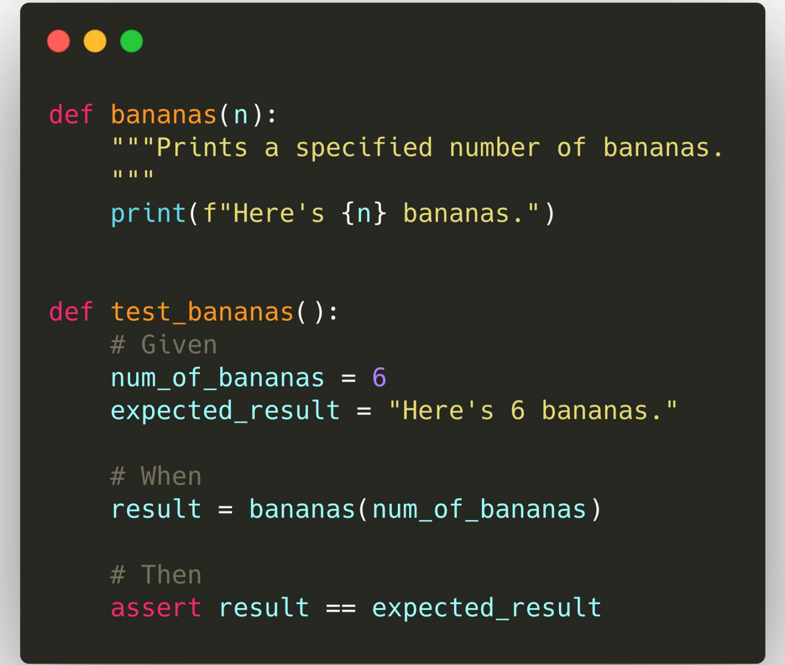
```python
def bananas(n):
    """Prints a specified number of bananas.
    """
    print(f"Here's {n} bananas.")


def test_bananas():
    # Given
    num_of_bananas = 6
    expected_result = "Here's 6 bananas."

    # When
    result = bananas(num_of_bananas)

    # Then
    assert result == expected_result
```

A useful *useful test* checklist

- ~~Know your input & output: demands deterministic code~~
- Use a Given-When-Then pattern
- Assert that a result matches expectations

The Given-When-Then pattern

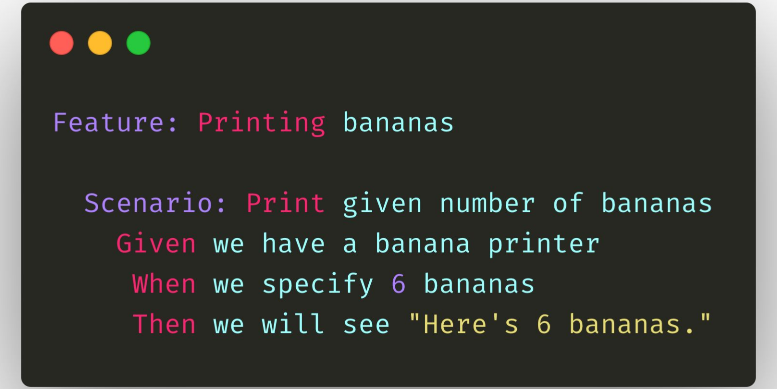
- **Given:** conditions for your test
- **When:** the action that you want to test
- **Then:** check if it meets expectations



```python
def bananas(n):
    """Prints a specified number of bananas.
    """
    print(f"Here's {n} bananas.")


def test_bananas():
    # Given
    num_of_bananas = 6
    expected_result = "Here's 6 bananas."

    # When
    result = bananas(num_of_bananas)

    # Then
    assert result == expected_result
```

The Given-When-Then pattern is the basis of behaviour tests

- https://github.com/behave/behave
- https://behave.readthedocs.io/en/latest/



```
Feature: Printing bananas

  Scenario: Print given number of bananas
    Given we have a banana printer
     When we specify 6 bananas
     Then we will see "Here's 6 bananas."
```

Given-When-Then with pytest

Let's look at the second item in test_math_functions.py

A useful *useful test* checklist

- ~~Know your input & output: demands deterministic code~~
- ~~Use a Given-When-Then pattern~~
- ~~Assert that a result matches expectations~~

Getting fancy with parametrization in pytest

Guest-starring pytest.raises

Parametrization: lets you define multiple sets of inputs and outputs to be run by a test function or class.

Let's look at the rest of test_math_functions.py

https://docs.pytest.org/en/latest/parametrize.html

Getting fancier with fixtures in pytest

Fixtures: a test representation of your input and output, which can be easily run in any test you like. It can also be used as scaffolding for test functions.

Let's look at dictionary_functions.py and test_dictionary_functions.py

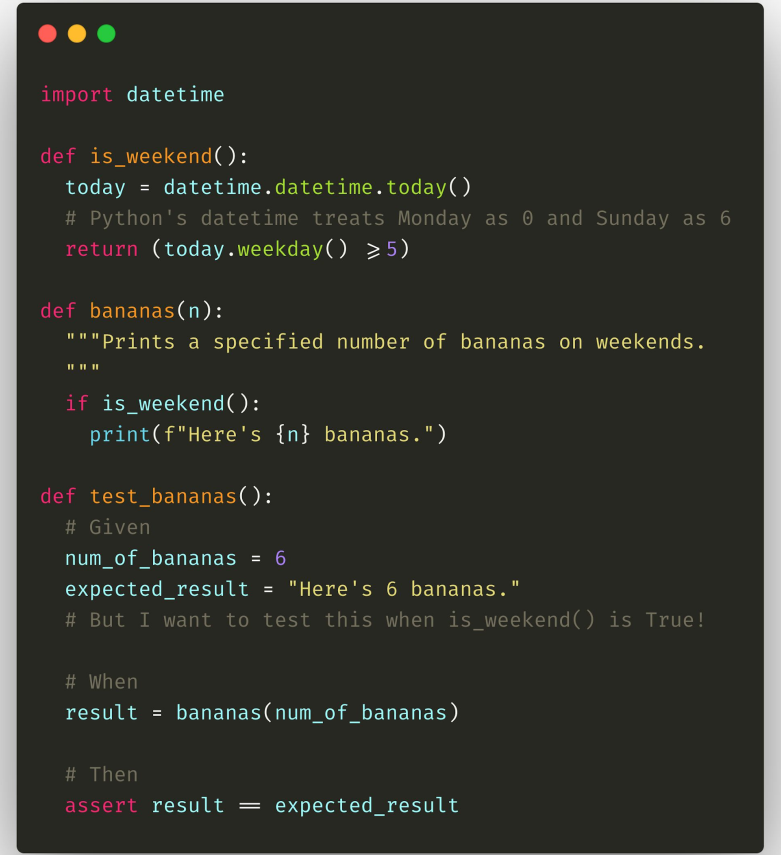
https://docs.pytest.org/en/latest/fixture.html

Mock, MagicMock & patch from unittest

What happens when a function you want to test, has to evaluate another function inside it?

If you're curious, here are some test examples: https://github.com/evandekieft/python-mock-examples/blob/master/tests.py

❤️ My colleague **Ester Ramos Carmona** gave a great talk on Python Mocks at PyConES 2019 Alicante



```python
import datetime

def is_weekend():
  today = datetime.datetime.today()
  # Python's datetime treats Monday as 0 and Sunday as 6
  return (today.weekday() ≥ 5)

def bananas(n):
  """Prints a specified number of bananas on weekends.
  """
  if is_weekend():
    print(f"Here's {n} bananas.")

def test_bananas():
  # Given
  num_of_bananas = 6
  expected_result = "Here's 6 bananas."
  # But I want to test this when is_weekend() is True!

  # When
  result = bananas(num_of_bananas)

  # Then
  assert result == expected_result
```

Where to from here?

1. Discovery & proof of concept stage in a Jupyter notebook - data cleaning, feature extraction, etc
2. Export .py from Jupyter notebook
3. Turn your code into functions
4. Write unit tests
5. Work with your data engineers to turn it into a reusable module

Good habits for effective testing

Have a solid environment creation process. A reliable routine to create an easily reproducible environment that can be standardised, replicated and used anywhere. Remove as many unknowns as possible.

Don't rely heavily on writing documentation to record code logic. Documentation rot is a big problem, especially if it isn't a strict part of your process, and worse yet if your documentation lives apart from your code.

Focus on structuring your use cases as input/output.

Good habits for effective testing (cont’d)

Give each function a single responsibility. Giant functions with several complexities within them are harder and more messy to test. Refactor each of these complexities out into separate functions.

Bonus: Don't use assert statements in your non-test code, e.g. assert num == 10. You can override these too easily by running your file with the -O option.

The most useful tests...

... describe your expectations of your code.

Tips for test cases:

- **size** (I should be able to print 10,000 bananas)
- **type** (I should only accept numbers as an input)
- **limit/boundary** (if I can only print < 50 bananas, I should check 49 & 50)
- **order** (if I print a list of banana messages, they should be in the order I want)
- **dichotomy** (if I can only print bananas on a weekend, I should check both weekend and weekdays)

Unit tests in machine learning and NNs

- https://medium.com/@keeper6928/how-to-unit-test-machine-learning-code-57cf6fd81765
- https://medium.com/@keeper6928/mltest-automatically-test-neural-network-models-in-one-function-call-eb6f1fa5019d
- https://www.oreilly.com/library/view/thoughtful-machine-learning/9781449374075/ch01.html

Questions

Material & feedback

https://github.com/feiphoon/testing-for-data-scientists-talk

fei.phoon@gmail.com

In this presentation

doctest

pytest

assert

raise

context manager

parametrize

test classes

fixtures

fixture scope

test-driven development (TDD)

behaviour-driven development (BDD)

unittest

Mock

MagicMock

patch