

Hunting Bugs While Sleeping

Property-Based Testing with Continuous Integration



Paul Amazona

Developer

@whatevergeek





DataKind





Django Against the Dark Arts

@attacus_au

Container image security - Notary

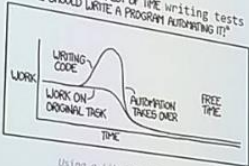
- Image repository.
- Digitally signs images to be published.
- The Update Framework (TUF) underlays the tool to maintain up-to-date software.
- Allows user to verify content that they download.
- Honourable mentions go to Clair

Polylingualism

boy3742

Generative Tests

"I SPEND A LOT OF TIME writing tests
I SHOULD WRITE A PROGRAM AUTOMATING IT!"



Using a library lets you
skip the 'ongoing development'
panel from the original comic

3Ai

Basically automating automated testing

- Hypothesis can generate...
- arguments to a test function
 - entire test programs!



pyconau

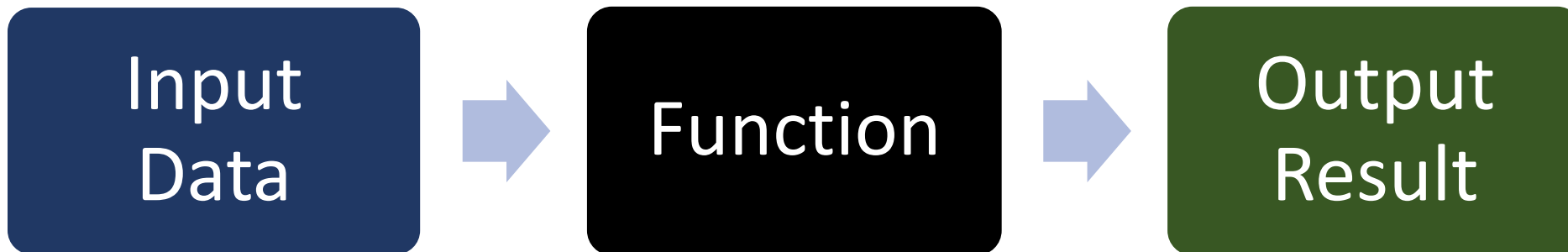
Property-Based Testing with Hypothesis



<https://tinyurl.com/hypothesis-prezo>

Property-Based Testing

A type of testing that asserts based on properties that describe the **relationship** between the **input** and **output** of the **function** being tested.



Testing the Multiply Function

```
def test_multiply():  
    actual_output = multiply(2,3)  
    expected_output = 2*3  
    assert actual_output == expected_output
```

Reimplementing

Testing the Multiply Function

```
def test_multiply():  
    actual_output = multiply(2,3)  
    expected_output = 2*3  
    assert actual_output == expected_output
```



Testing using Example Outputs

```
def test_example1():  
    actual_output = multiply(2,3)  
    expected_output = 6  
    assert actual_output == expected_output
```

```
def test_example2():  
    actual_output = multiply(4,5)  
    expected_output = 20  
    assert actual_output == expected_output
```


Parameterized Tests

```
@pytest.mark.parametrize("factor1, factor2, expected_output", [
    (2, 3, 6),
    (4, 5, 20),
])
def test_example2and3(factor1, factor2, expected_output):
    actual_output = multiply(factor1, factor2)
    assert actual_output == expected_output
```

Multiplication Properties

- **Commutative property**

When two numbers are multiplied together, the product is the same regardless of the order of the multiplicands.

For example $4 * 2 = 2 * 4$

- **Associative Property**

When three or more numbers are multiplied, the product is the same regardless of the grouping of the factors.

For example $(2 * 3) * 4 = 2 * (3 * 4)$

- **Multiplicative Identity Property**

The product of any number and one is that number.

For example $5 * 1 = 5$.

- **Distributive property**

The sum of two numbers times a third number is equal to the sum of each addend times the third number.

For example $4 * (6 + 3) = 4*6 + 4*3$

Property-Based Tests (PBT)

with Predetermined Inputs

```
@pytest.mark.parametrize("factor1, factor2", [
    (2, 3),
    (4, 5),
])
def test_commutative_property(factor1, factor2):
    assert multiply(factor1, factor2) == multiply(factor2, factor1)
```

Property-Based Tests (PBT)

with Predetermined Inputs

```
@pytest.mark.parametrize("factor1", [2,4])
def test_identity_property(factor1):
    assert multiply(factor1,1) == factor1

@pytest.mark.parametrize("num1, num2, num3", [
    (2, 3, 4),
    (4, 5, 6),
])
def test_distributive_property(num1, num2, num3):
    assert multiply(num1,(num2 + num3)) == multiply(num1,num2) + multiply(num1,num3)
```

Property-Based Tests (PBT)

with Randomized Inputs

```
from hypothesis import given
import hypothesis.strategies as st

@given(st.integers(), st.integers())
def test_commutative_property(factor1, factor2):
    assert multiply(factor1, factor2) == multiply(factor2, factor1)
```


Property-Based Tests Libraries...

Language	PBT Libraries
python	hypothesis
.NET (C#, F#, etc)	FsCheck
haskell	quickcheck
java	junit-quickchek
javascript	fast-check
swift	SwiftCheck
scala	ScalaCheck

Some PBT library features...

More elaborate input criteria

```
@given(st.integers().filter(lambda x: x > 0), st.integers())
```

```
@example(factor1=2, factor2=3)
```

Ability to use predetermined input

```
@settings(max_examples=200)
```

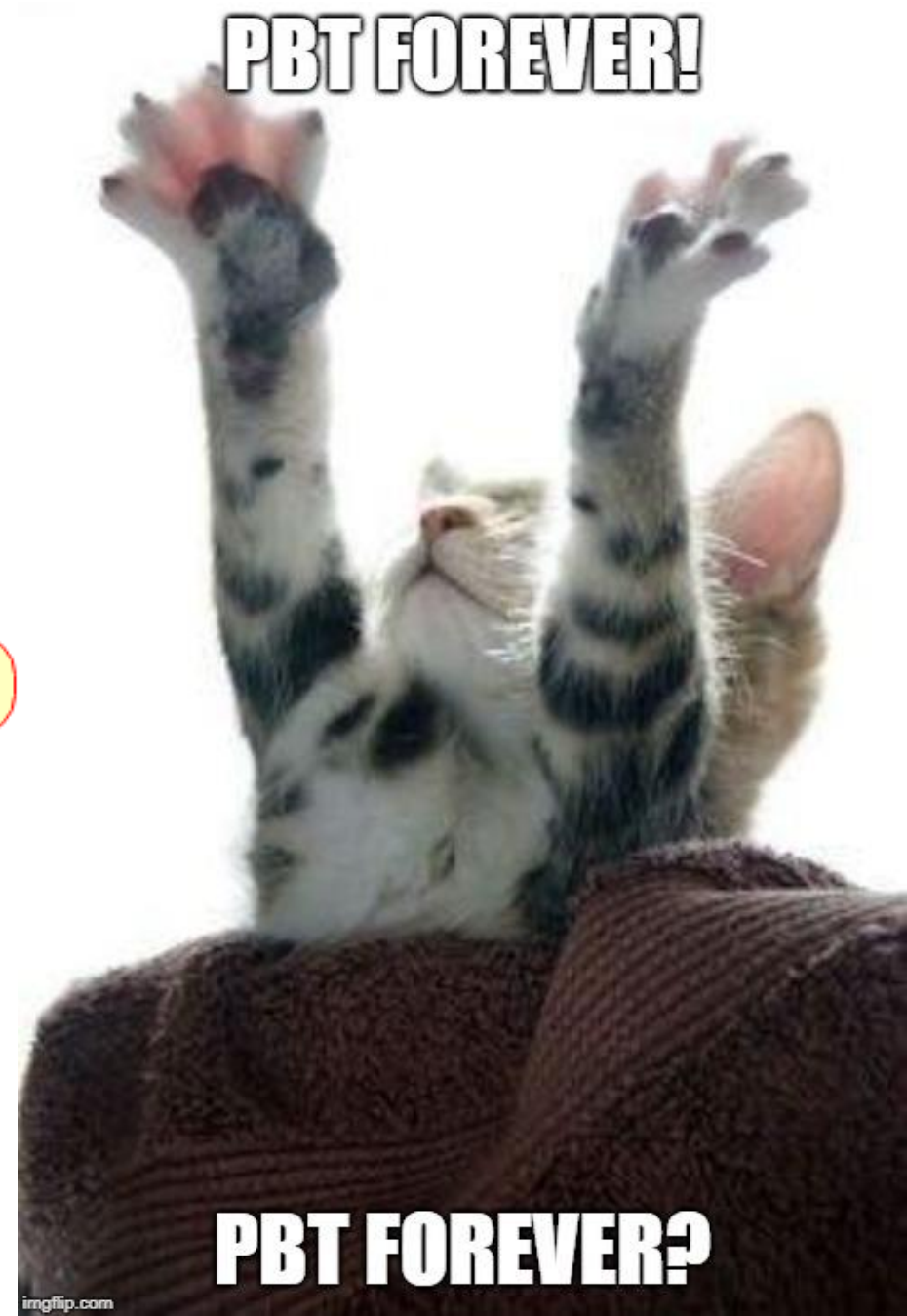
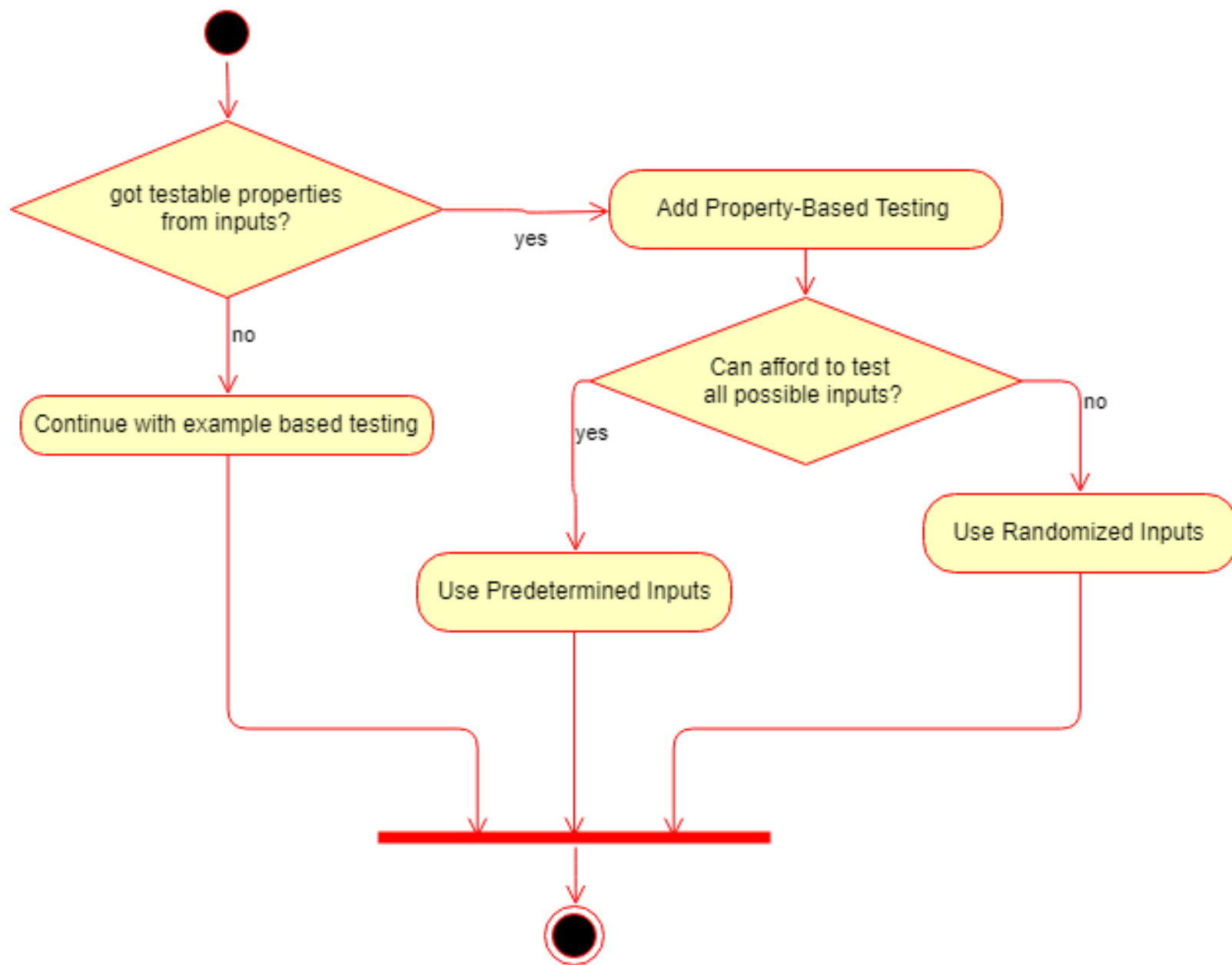
```
def test_commutative_property(factor1, factor2):
```

```
    assert multiply(factor1, factor2) == multiply(factor2, factor1)
```

Adjust sample size

Diamond Kata


INPUT	OUTPUT
A	A
B	A B B A
C	A B B C C B B A
D	A B B C C D D C C B B A







Hunting Bugs with CI











Pipelines	Process	Characteristics
Normal CI	Build-> Tests with Predetermined Inputs	<ul style="list-style-type: none">• Trigger: merge/PR/etc• More predictable duration• Purpose: Detect problems early
Bug Hunting CI	Build->Test with Randomized Inputs	<ul style="list-style-type: none">• Trigger: Scheduled build (every hour, every day, etc).• Can take time depending on sample size of randomized inputs• Purpose: Hunt Bugs

DEMO: Bug Hunting CI Pipeline

 geeklogic / multipycation / Pipelines

 Search






✓ Connect

✓ Select


✓ Configure

Create pipeline

azure-pipelines.yml 

DocsRun

```
12  · pool:
13  ·   vmImage: 'Ubuntu-16.04'
14  · strategy:
15  ·   matrix:
16  ·     Python37:
17  ·       python.version: '3.7'
18  ·   maxParallel: 4
19
20  · steps:
21  · - task: UsePythonVersion@0
22  ·   inputs:
23  ·     versionSpec: '$(python.version)'
24  ·     architecture: 'x64'
25
26  · - script: python -m pip install --upgrade pip && pip install -r requirements.txt
27  ·   displayName: 'Install dependencies'
28
29  · - script: |
30  ·     pip install pytest
31  ·     pytest --doctest-modules --junitxml=junit/test-results.xml
32  ·   displayName: 'pytest'
33
```



DEMO: Bug Hunting CI Pipeline

[Logs](#) [Summary](#) [Tests](#)

Test Python37

Started: 16/03/2019, 01:01:53

Pool: [Hosted Ubuntu 1604](#) · Agent: Hosted Agent

... 33s





✓	Initialize Agent · succeeded	<1s
✓	Prepare job · succeeded	<1s
✓	Initialize job · succeeded	1s
✓	Checkout · succeeded	8s
✓	UsePythonVersion · succeeded	1s
✓	Install dependencies · succeeded	11s
✓	pytest · succeeded	5s
✓	PublishTestResults · succeeded	3s
✓	Post-job: Checkout · succeeded	<1s
✓	Finalize Job · succeeded	<1s

DEMO: Bug Hunting CI Pipeline

```
✓ pytest ↑ Previous task ↓ Next task ×
```

```
37 ===== Hypothesis Statistics =====
38 test_arithmetic_properties_with_randomized_inputs.py::test_commutative_property:
39
40   - 100 passing examples, 0 failing examples, 0 invalid examples
41   - Typical runtimes: < 1ms
42   - Fraction of time spent in data generation: ~ 54%
43   - Stopped because settings.max_examples=100
44
45 test_arithmetic_properties_with_randomized_inputs.py::test_associative_property:
46
47   - 100 passing examples, 0 failing examples, 30 invalid examples
48   - Typical runtimes: < 1ms
49   - Fraction of time spent in data generation: ~ 69%
50   - Stopped because settings.max_examples=100
51   - Events:
52     * 64.62%, Retried draw from integers().filter(lambda x: x > 0) to satisfy filter
53     * 23.08%, Aborted test because unable to satisfy integers().filter(lambda x: x > 0)
54
55 test_arithmetic_properties_with_randomized_inputs.py::test_identity_property:
56
57   - 100 passing examples, 0 failing examples, 0 invalid examples
58   - Typical runtimes: < 1ms
59   - Fraction of time spent in data generation: ~ 44%
60   - Stopped because settings.max_examples=100
61
62 test_arithmetic_properties_with_randomized_inputs.py::test_distributive_property:
63
64   - 100 passing examples, 0 failing examples, 0 invalid examples
65   - Typical runtimes: < 1ms
66   - Fraction of time spent in data generation: ~ 59%
67   - Stopped because settings.max_examples=100
68
69 ===== 4 passed in 0.46 seconds =====
```

DEMO: Bug Hunting CI Pipeline

YAML Variables **Triggers** History |  Save & queue  Discard  Summary  Queue ...

Continuous integration



whatevergeek/multipycation
Enabled

Pull request validation



whatevergeek/multipycation
Enabled



Scheduled Add



1:30
Mon through Sun

Build completion Add


Build when another build completes

 Mon through Sun at 1:30  Delete

When to build

☒ Mon ☒ Tue ☒ Wed ☒ Thu ☒ Fri ☒ Sat ☒ Sun

01h 

30m 

(UTC+08:00) Kuala Lumpur, Singapore 


☐ Only schedule builds if the source or pipeline has changed

Branch filters

Type

Include 

Branch specification

master 



 Add

<https://github.com/whatevergeek/multipycation>

Summary and Links

- Testing using Example Outputs
- Parameterized Tests
- Property-based Tests with Predetermined Inputs
- Property-based Tests with Randomized Inputs
- Bug Hunting CI Pipeline

Paul Amazona

@whatevergeek



<https://tinyurl.com/hypothesis-prezo>

<https://github.com/whatevergeek/multipycation>

<https://gitlab.com/whatevergeek/multipycation-gl-demo>

