Hypothesis

Tests that write themselves

David Kua

@davidkua

Hypothesis

~~Tests that write themselves~~

Generating tests from properties of your code

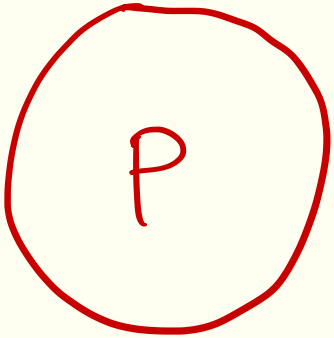
David Kua

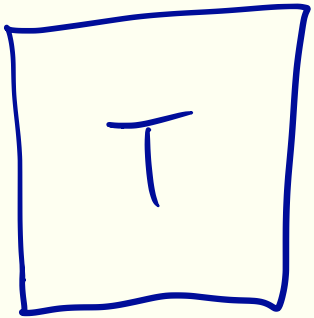
@davidkua

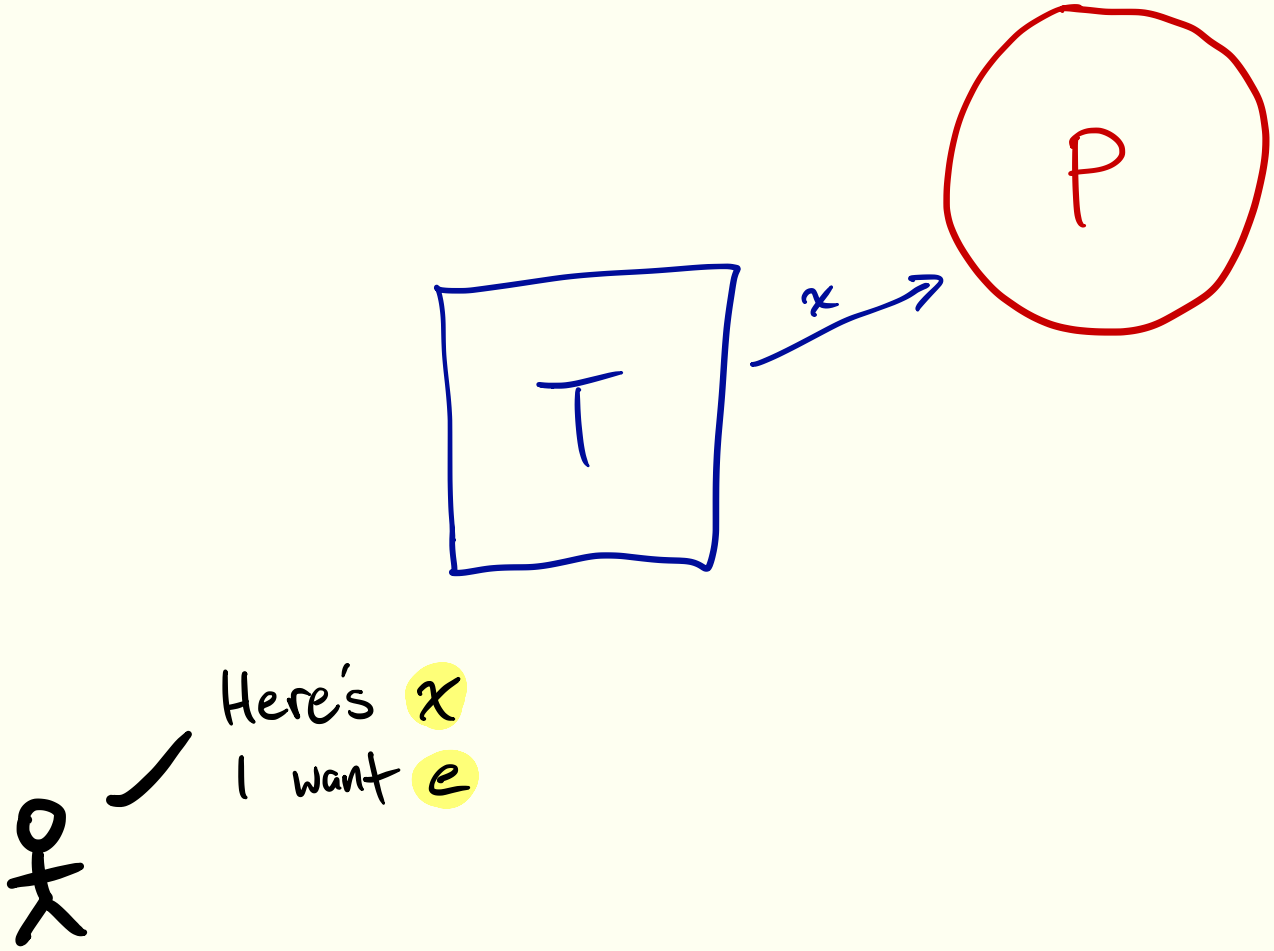
Agenda

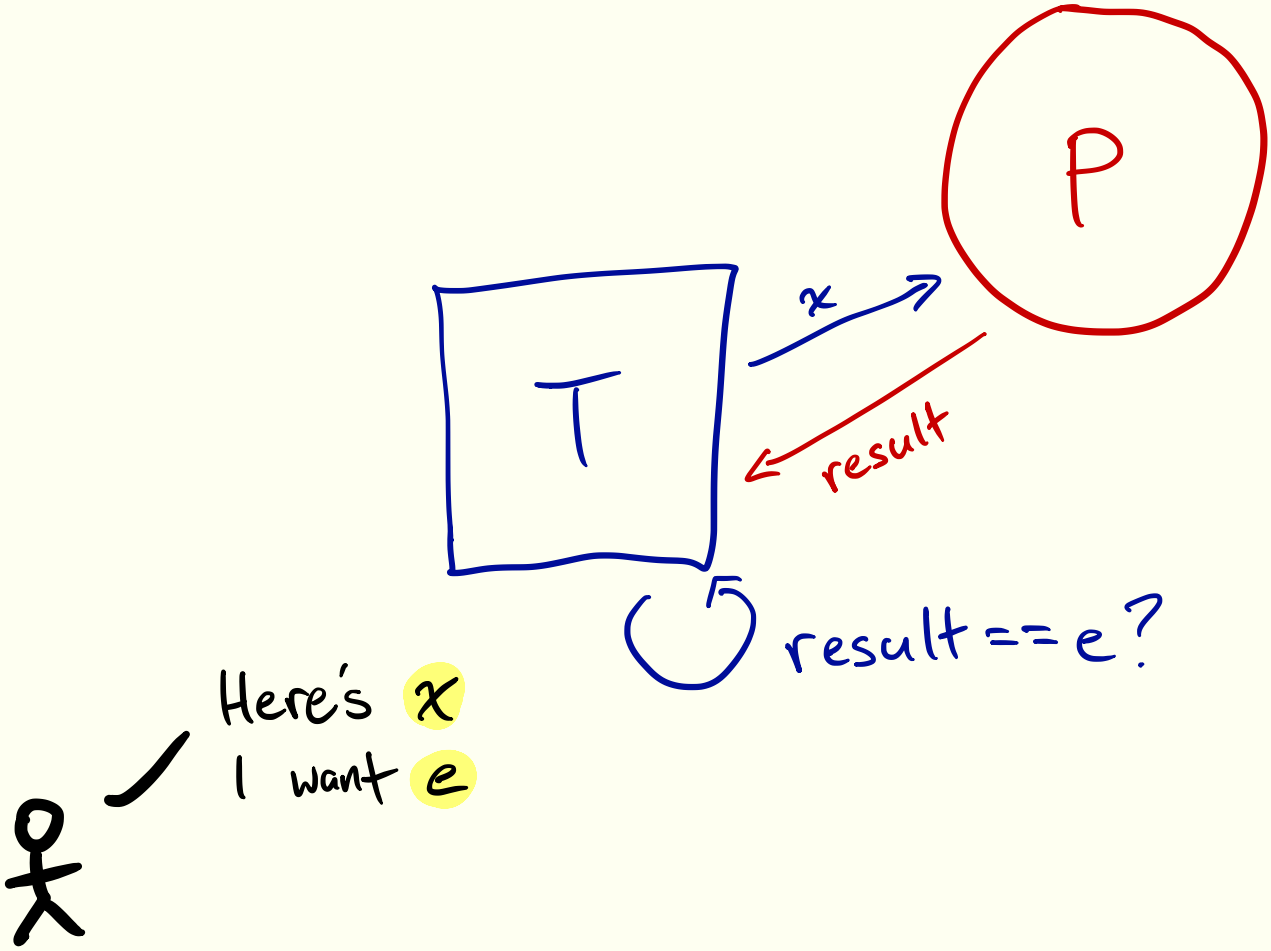
1. Overview of property-based testing (High level)
2. Hypothesis ☆

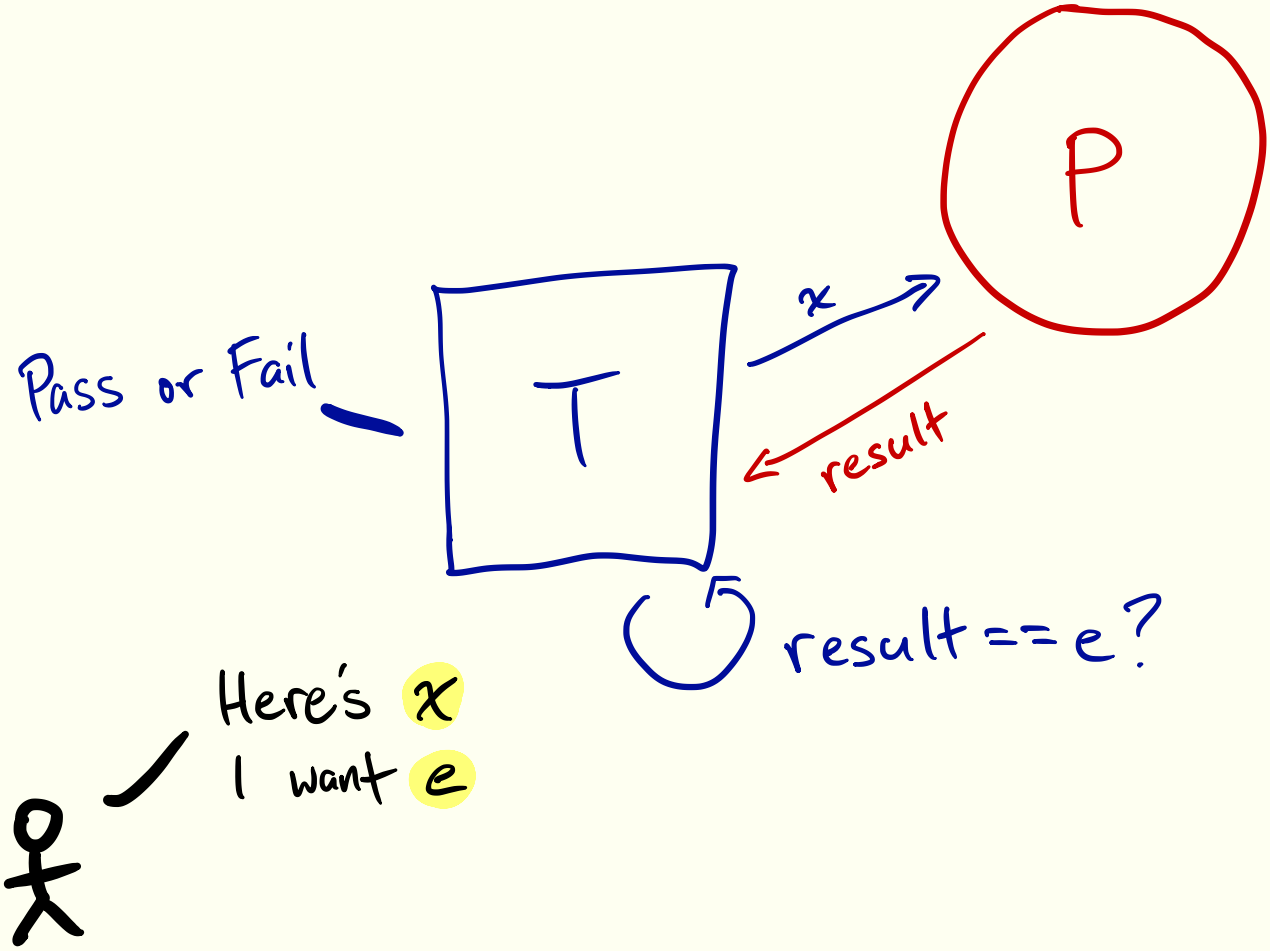
github.com/dkua/pyconca16-talk


P

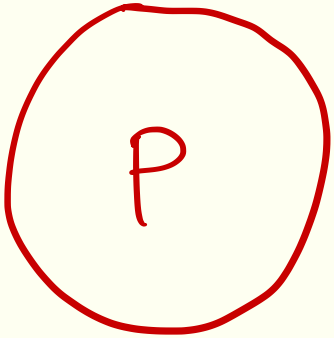

T

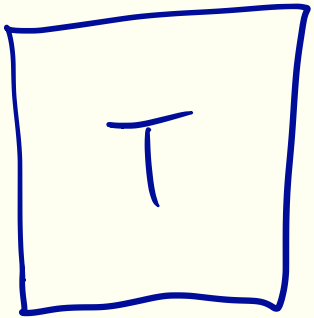

P
T
x
Here's x
I want e

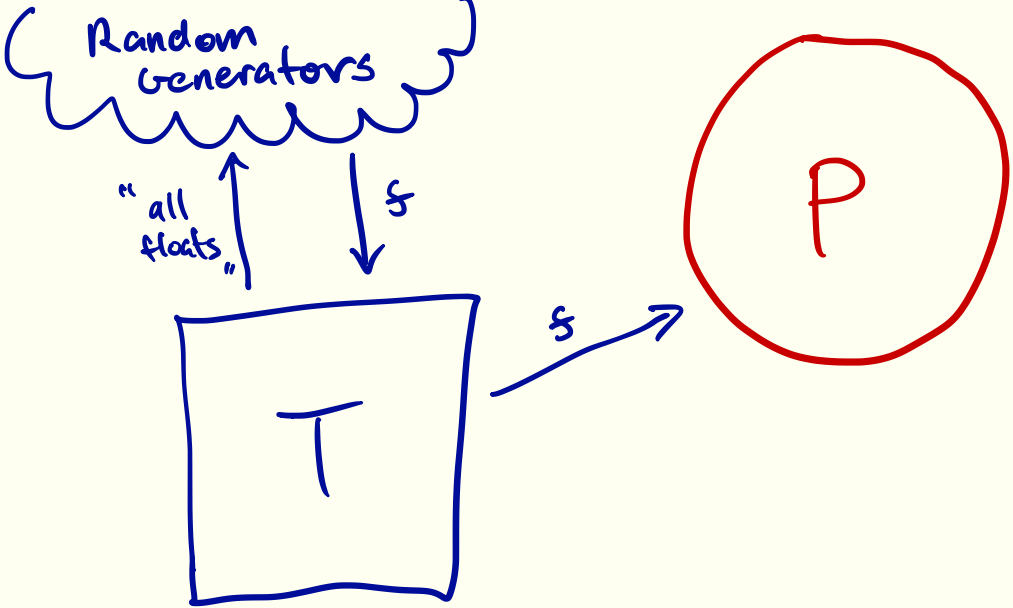

P
T
x
result
result == e?
Here's x
I want e

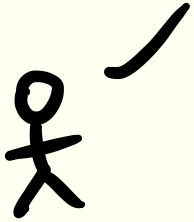

P
x
result
T
Pass or Fail
result == e?
Here's x
I want e


Random
Generators

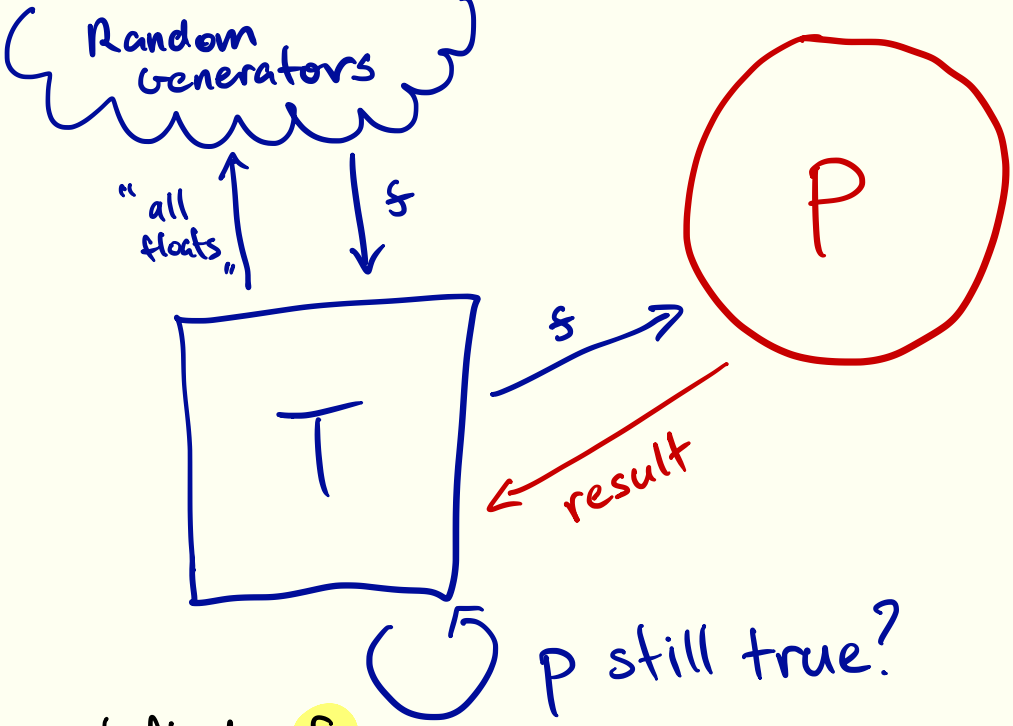

P


T



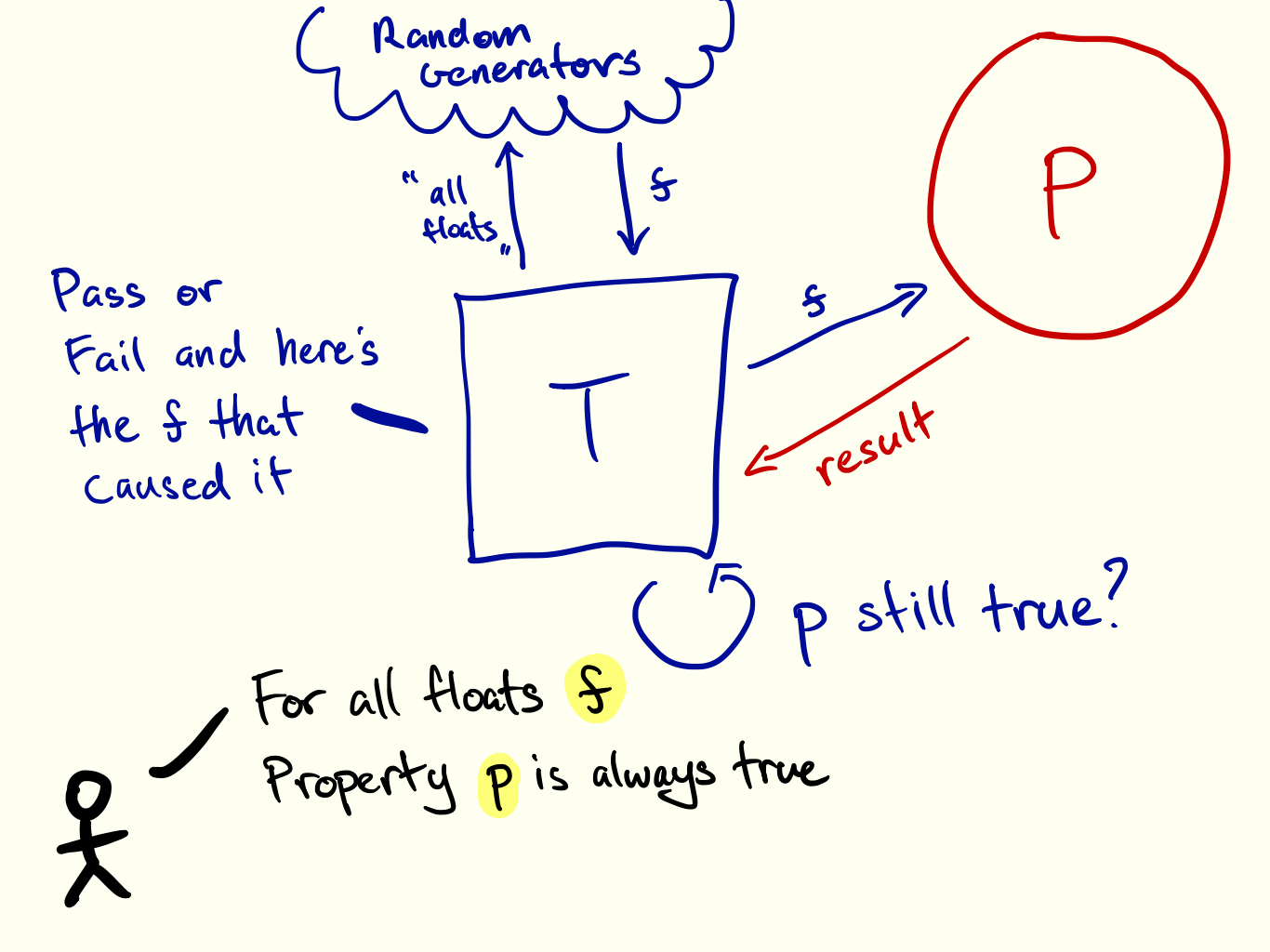


For all floats f
Property P is always true



For all floats f

Property p is always true


Random generators
"all floats"
f
P
Pass or Fail and here's the f that caused it
T
f
result
p still true?
For all floats f
Property p is always true

Unit Testing

Give an input. 2, 4, 56, 100, 888, ...

Give a result. 1, 2, 28, 50, 444, ...

Property-based Testing

Describe the inputs. "All ints"

Describe condition(s) that must hold.

"Remainder is 0"

or

"Result is an int"

Pros

- # of tests only limited by power & time
- code overhead is minimized
- random generation more likely to find edge case

Cons

- harder & longer to write
- finding properties can be difficult
- random generation might miss an edge case

An extra tool in your toolkit

QuickCheck:
A Lightweight Tool for Random Testing of Haskell Programs

Koen Claessen & John Hughes

Finding properties

"copy + paste"

"given"

"who cares"

"reversible"

The names are terrible I know. Sorry.

Frameworks

Haskell - QuickCheck

Erlang - Erlang QuickCheck, PropEr

Clojure - test.check

Scala - ScalaCheck

F# - FsCheck

PHP - Eris

C - QuickCheck for C

... and more

Hypothesis

- Python 2.7, 3.3+
- py.test, unittest, Nose (maybe)
- extra packages for: Django, datetime, NumPy
- super dope

hypothesis.readthedocs.io

1

Addition

$$x + y = y + x$$

w/ Unit Tests

```python
def test_add_zero():
    assert 0 + 1 == 1 + 0

def test_add_single_digits():
    assert 1 + 2 == 2 + 1

def test_add_double_digits():
    assert 10 + 12 == 12 + 10
```

"copy + paste"

w/ Hypothesis

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

@given(st.integers(), st.integers())
def test_add(x, y):
    assert x + y == y + x
```

Running Hypothesis

```
(pycon) ~/pyconca16-talk [master] »
(pycon) ~/pyconca16-talk [master] » pytest --hypothesis-show-statistics add_example.py
============================================================ test session starts =============
platform darwin -- Python 2.7.12, pytest-3.0.4, py-1.4.31, pluggy-0.4.0
rootdir: /Users/dkua/pyconca16-talk, inifile:
plugins: hypothesis-3.6.0
collected 4 items

add_example.py ....
============================================================ Hypothesis Statistics ===========

add_example.py::test_add:

  - 200 passing examples, 0 failing examples, 0 invalid examples
  - Typical runtimes: < 1ms
  - Stopped because settings.max_examples=200


========================================================== 4 passed in 0.11 seconds ===========
(pycon) ~/pyconca16-talk [master] »
```

hypothesis.strategies

none

booleans

floats

integers

complex numbers

lists

tuples

text

binary

dictionaries

sets

frozen sets

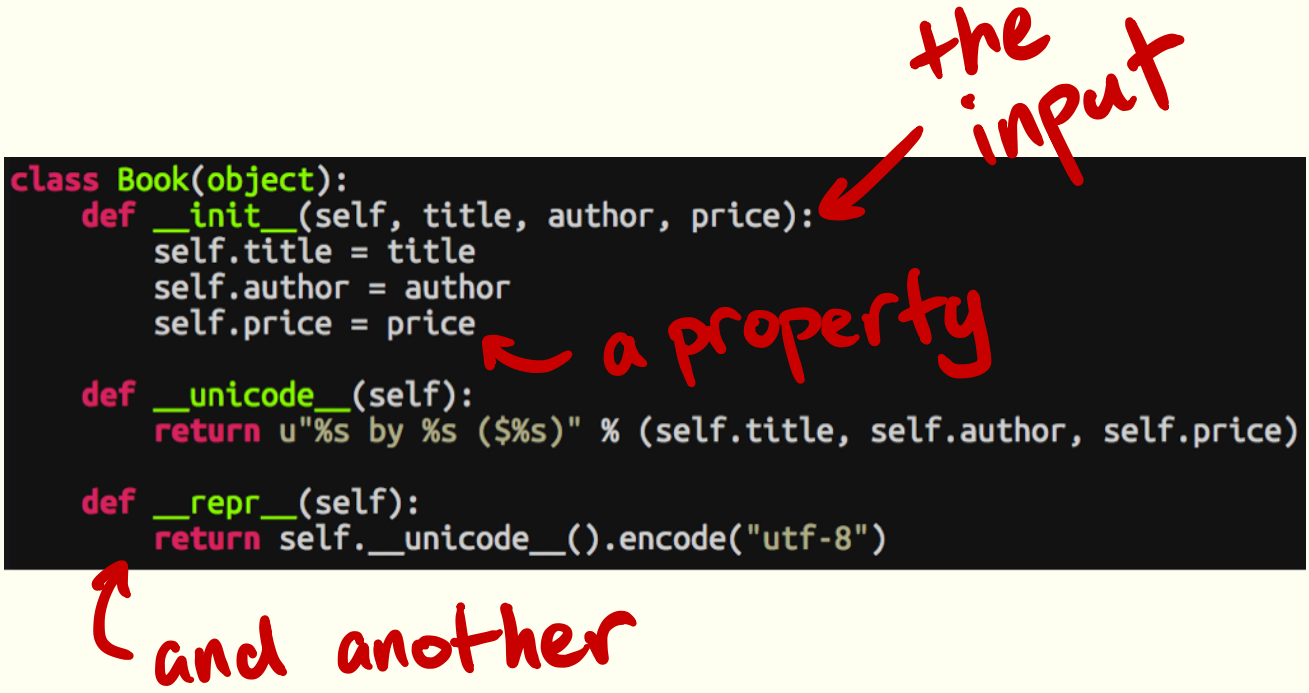
... and more!

2

Can I write my own strategies?

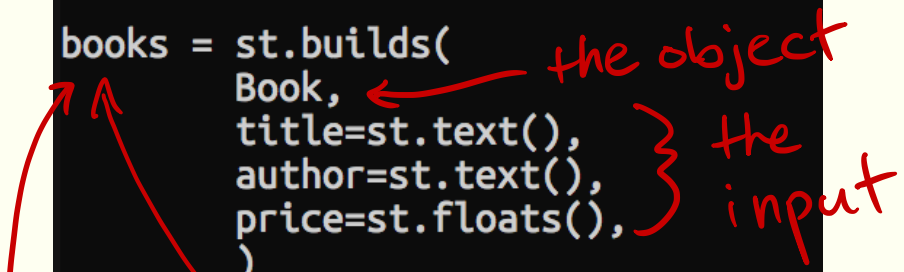
Make sure that all books support Unicode and never cost less than nothing!

"given"


the input
class Book(object):
def __init__(self, title, author, price):
self.title = title
self.author = author
self.price = price
a property
def __unicode__(self):
return u"%s by %s ($%s)" % (self.title, self.author, self.price)
def __repr__(self):
return self.__unicode__().encode("utf-8")
and another

Use builds()

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

books = st.builds(
        Book,
        title=st.text(),
        author=st.text(),
        price=st.floats(),
        )

@given(books)
def test_unicode(book):
    print(book)

@given(books)
def test_never_negative(book):
    assert book.price >= 0.00
```



A Hypothesis failure!

```
=================================================== FAILURES ===================================================
_____________________________________________ test_never_negative _____________________________________________

    @given(books)
>   def test_never_negative(book):

book_example.py:36:

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
../.virtualenvs/pycon/lib/python2.7/site-packages/hypothesis/core.py:524: in wrapped_test
    print_example=True, is_final=True
../.virtualenvs/pycon/lib/python2.7/site-packages/hypothesis/executors.py:58: in default_new_style_executor
    return function(data)
../.virtualenvs/pycon/lib/python2.7/site-packages/hypothesis/core.py:111: in run
    return test(*args, **kwargs)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

book =  by  ($nan)

    @given(books)
    def test_never_negative(book):
>       assert book.price >= 0.00
E       assert nan >= 0.0
E        +  where nan =  by  ($nan).price

book_example.py:37: AssertionError
---------------------------------------------------- Hypothesis ----------------------------------------------------
Falsifying example: test_never_negative(book= by  ($nan))
============================================ 1 failed, 1 passed in 1.16 seconds ============================================
```

the failed line

why it failed

the example that failed

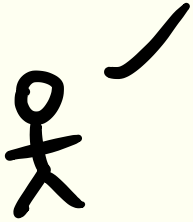
The fix

```
class Book(object):
    def __init__(self, title, author, price):
        self.title = title
        self.author = author
        if not price >= 0.00:
            price = 0.00
        self.price = price

    def __unicode__(self):
        return u"%s by %s ($%s)" % (self.title, self.author, self.price)

    def __repr__(self):
        return self.__unicode__().encode("utf-8")
```

3

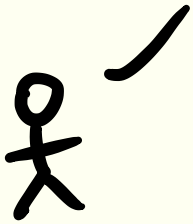
The future is in online shopping! We need a shopping cart.

My boss wants a shopping cart. What makes a bad shopping cart for my boss?

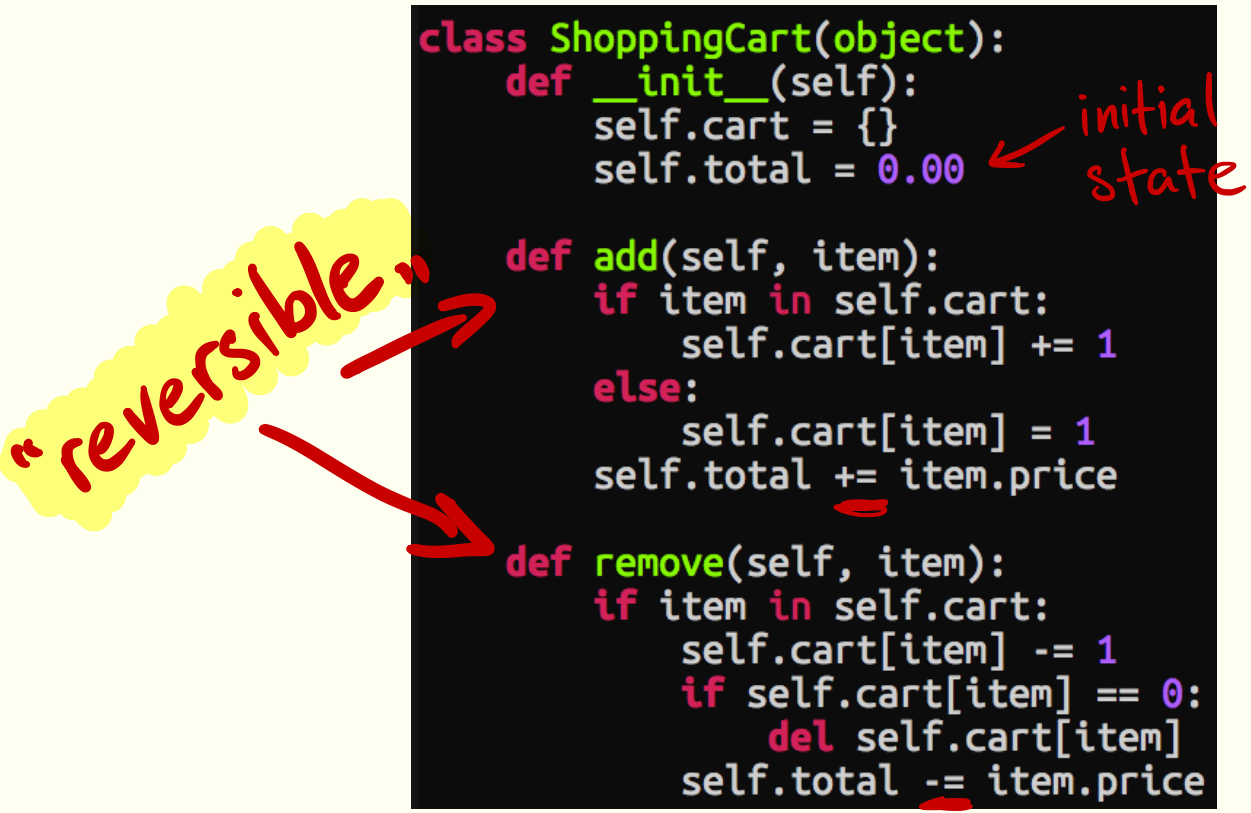


My boss wants a shopping cart. What makes a bad shopping cart for my boss?

"who cares"



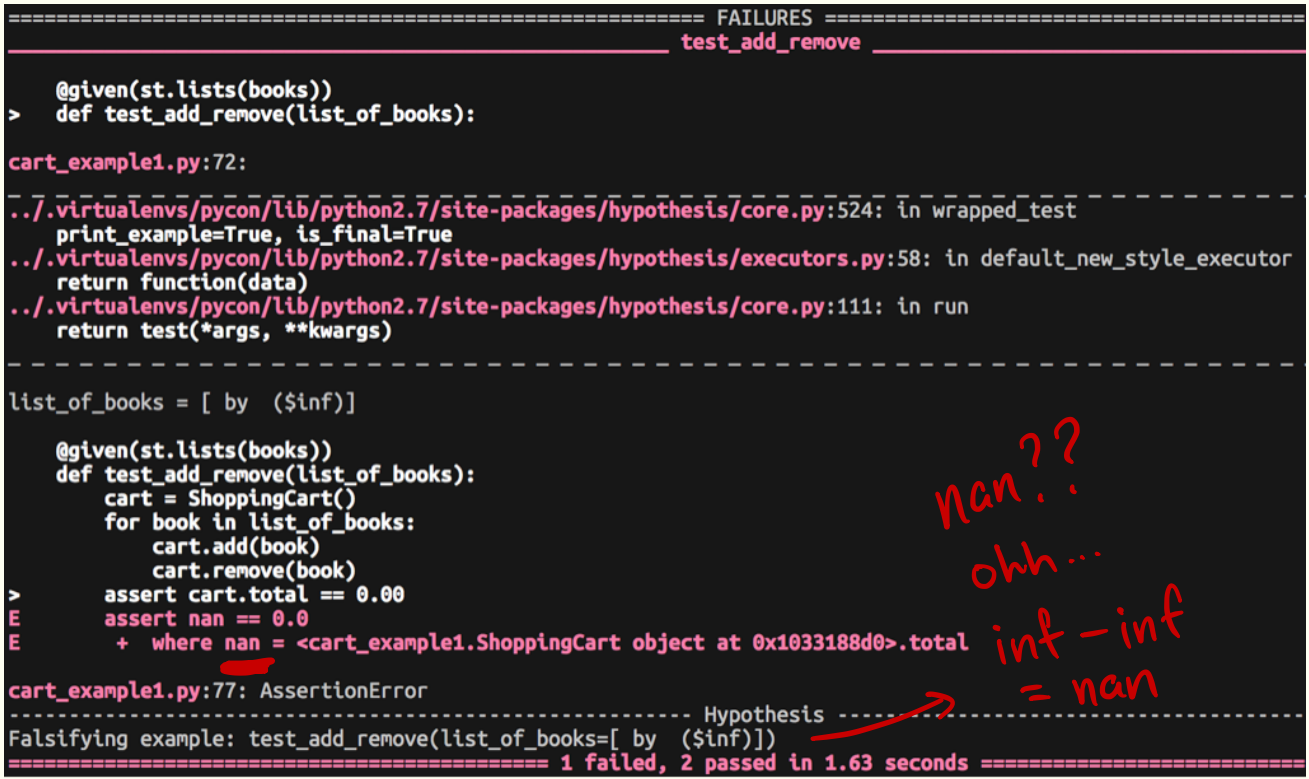
total < 0


class ShoppingCart(object):
def __init__(self):
self.cart = {}
self.total = 0.00
initial state
def add(self, item):
if item in self.cart:
self.cart[item] += 1
else:
self.cart[item] = 1
self.total += item.price
def remove(self, item):
if item in self.cart:
self.cart[item] -= 1
if self.cart[item] == 0:
del self.cart[item]
self.total -= item.price
"reversible"

composable!

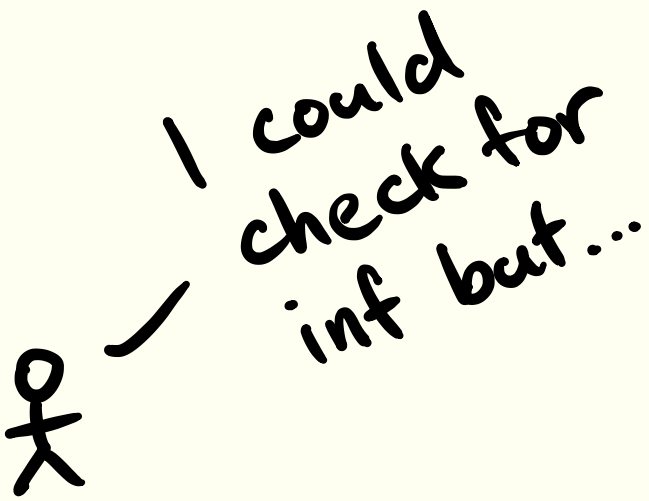
```python
@given(st.lists(books))
def test_add(list_of_books):
    cart = ShoppingCart()
    for book in list_of_books:
        cart.add(book)
    assert cart.total == sum(book.price for book in list_of_books)

@given(st.lists(books))
def test_remove(list_of_books):
    cart = ShoppingCart()
    for book in list_of_books:
        cart.remove(book)
    assert cart.total == 0.00

@given(st.lists(books))
def test_add_remove(list_of_books):
    cart = ShoppingCart()
    for book in list_of_books:
        cart.add(book)
        cart.remove(book)
    assert cart.total == 0.00
```

initial state

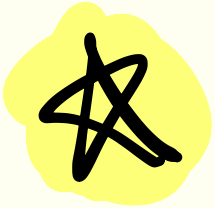

FAILURES
test_add_remove
@given(st.lists(books))
> def test_add_remove(list_of_books):
cart_example1.py:72:
../.virtualenvs/pycon/lib/python2.7/site-packages/hypothesis/core.py:524: in wrapped_test
print_example=True, is_final=True
../.virtualenvs/pycon/lib/python2.7/site-packages/hypothesis/executors.py:58: in default_new_style_executor
return function(data)
../.virtualenvs/pycon/lib/python2.7/site-packages/hypothesis/core.py:111: in run
return test(*args, **kwargs)
list_of_books = [by (ŝinf)]
@given(st.lists(books))
def test_add_remove(list_of_books):
cart = ShoppingCart()
for book in list_of_books:
cart.add(book)
cart.remove(book)
> assert cart.total == 0.00
E assert nan == 0.0
E + where nan = <cart_example1.ShoppingCart object at 0x1033188d0>.total
cart_example1.py:77: AssertionError
Hypothesis
Falsifying example: test_add_remove(list_of_books=[by ($inf)])
1 failed, 2 passed in 1.63 seconds
nan??
ohh...
inf - inf
= nan

```
books = st.builds(
        Book,
        title=st.text(),
        author=st.text(),
        price=st.floats(allow_infinity=False),
        )
```

customizable



Rule Based Stateful Testing

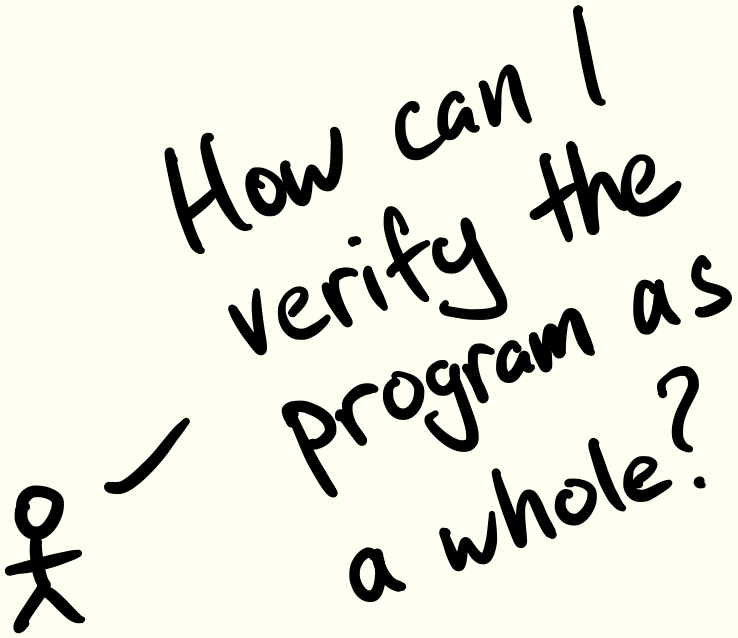


We need
to add coupons!

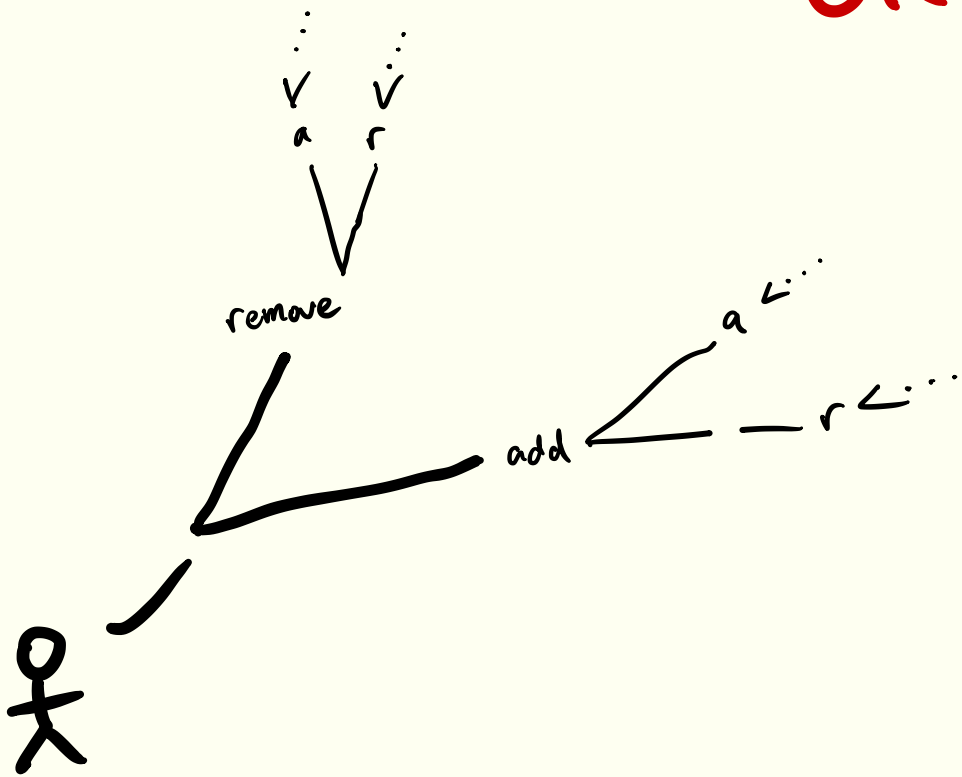
```
    def add_discount(self, coupon):
        if self.total > 0 and coupon not in self.discounts:
            self.discounts[coupon] = coupon.percent
            self.total *= self.discounts[coupon]

    def remove_discount(self, coupon):
        if coupon in self.discounts:
            self.total /= self.discounts[coupon]
            del self.discounts[coupon]
```

```
[
Coupon(name='10OFF', percent=0.9)
Coupon(name='20OFF', percent=0.8)
Coupon(name='30OFF', percent=0.7)
Coupon(name='40OFF', percent=0.6)
Coupon(name='50OFF', percent=0.5)
Coupon(name='60OFF', percent=0.4)
Coupon(name='70OFF', percent=0.3)
Coupon(name='80OFF', percent=0.2)
Coupon(name='90OFF', percent=0.1)
]
```

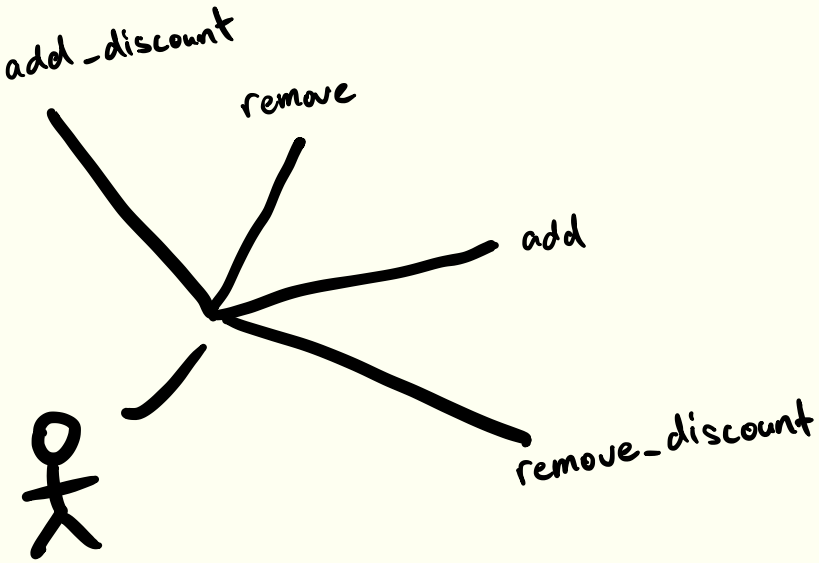
namedtuples

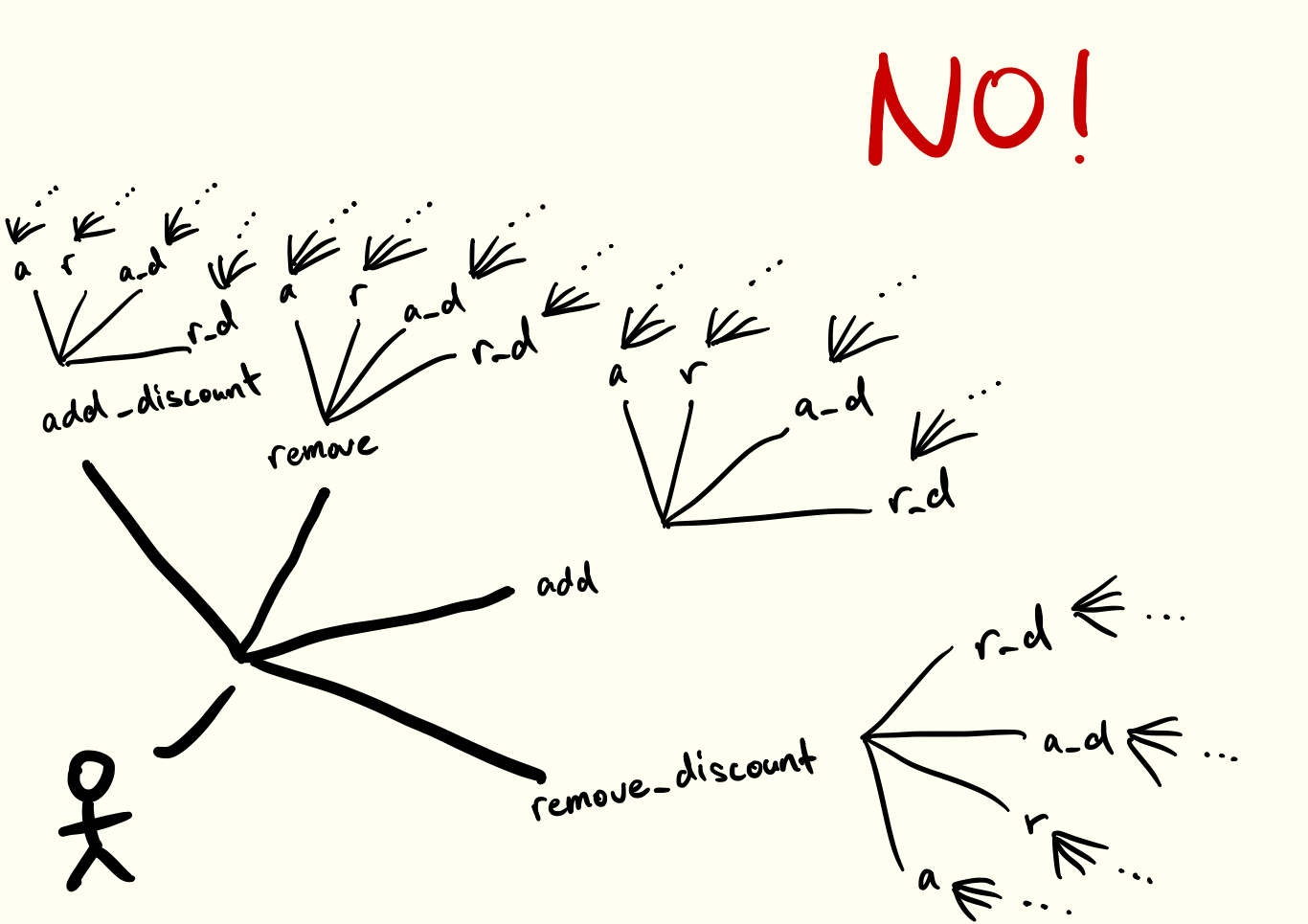

How can I verify the program as a whole?

OK...



4...




NO!
a
r
a_d
r_d
add_discount
remove
add
remove_discount

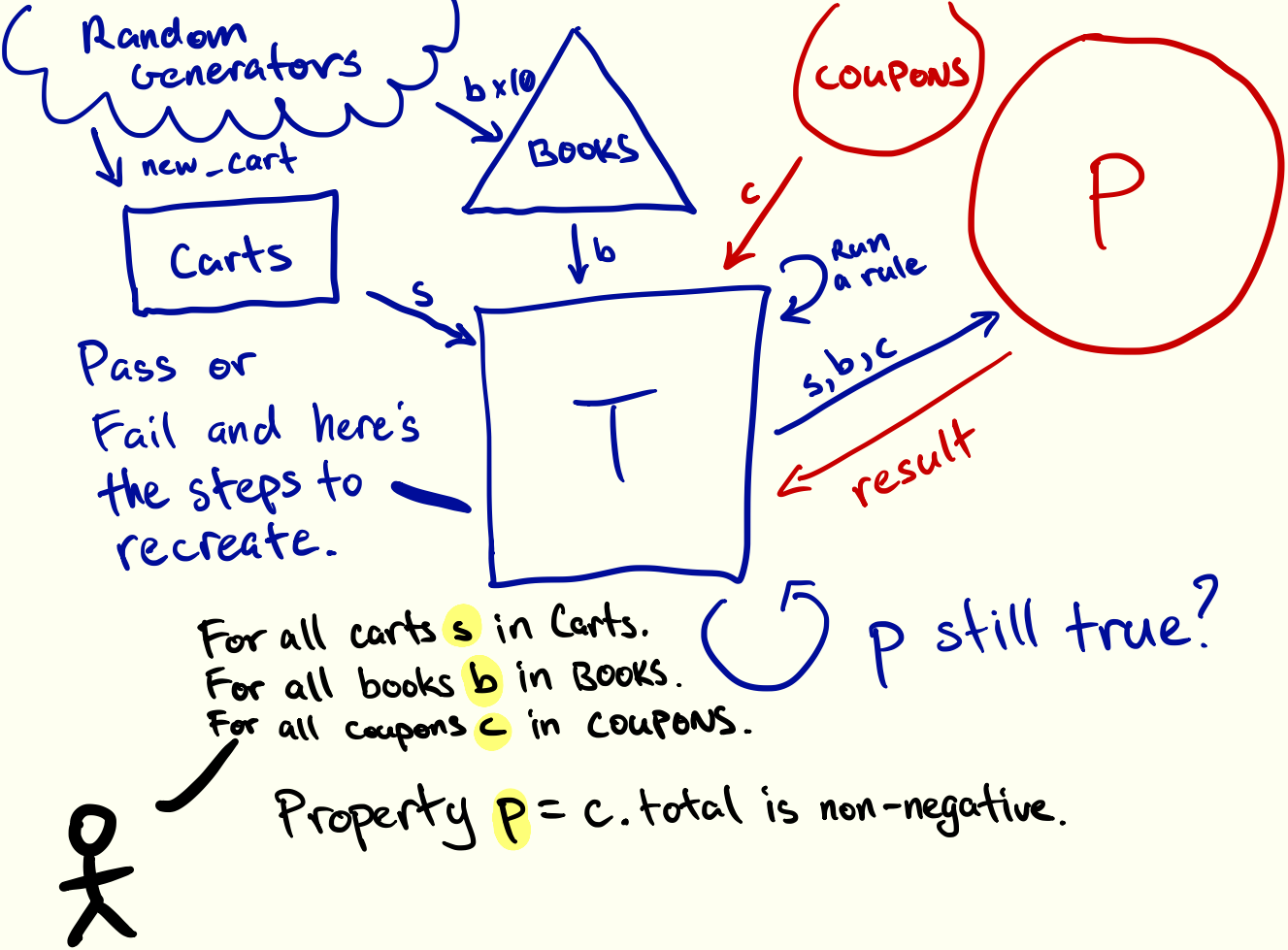
hypothesis.stateful

@rule

RuleBasedStateMachine

Bundle


class CartMachine(RuleBasedStateMachine):
Carts = Bundle("carts")
BOOKS = st.lists(books, min_size=10).example()
@rule(target=Carts)
def new_cart(self):
return ShoppingCart()
@rule(cart=Carts, item=st.sampled_from(BOOKS))
def add_item(self, cart, item):
cart.add(item)
assert cart.total >= 0.00
@rule(cart=Carts, item=st.sampled_from(BOOKS))
def remove_item(self, cart, item):
cart.remove(item)
assert cart.total >= 0.00
@rule(cart=Carts, coupon=st.sampled_from(COUPONS))
def add_coupon(self, cart, coupon):
cart.add_discount(coupon)
assert cart.total >= 0.00
@rule(cart=Carts, coupon=st.sampled_from(COUPONS))
def remove_discount(self, cart, coupon):
cart.remove_discount(coupon)
assert cart.total >= 0.00
TestCarts = CartMachine.TestCase
collection of state
similar to @given
checking property
so Pytest finds it


Random Generators
new_cart
b x10
BOOKS
COUPONS
Carts
b
c
Run a rule
P
s
T
Pass or Fail and here's the steps to recreate.
s,b,c
result
For all carts s in Carts.
For all books b in BOOKS.
For all coupons c in COUPONS.
p still true?
Property p = c.total is non-negative.

Running it...

```
self = CartMachine({'carts': [VarReference(name=u'v1')]}), cart = <cart_example2.ShoppingCart object at 0x10cffdbd0>
item = 婀0 ²
驥瘏끋?by 쿒 ($994.245374416)

    @rule(cart=Carts, item=st.sampled_from(BOOKS))
    def remove_item(self, cart, item):
        cart.remove(item)
>       assert cart.total >= 0.00
E       AssertionError: assert -99.42453744155341 >= 0.0
E        +  where -99.42453744155341 = <cart_example2.ShoppingCart object at 0x10cffdbd0>.total

cart_example2.py:91: AssertionError
---------------------------------------------------------- Hypothesis ----------------------------------------------------------
Step #1: v1 = new_cart()
Step #2: add_item(item=婀0 ²
驥瘏끋?by 쿒 ($994.245374416), cart=v1)
Step #3: add_coupon(coupon=Coupon(name='10OFF', percent=0.9), cart=v1)
Step #4: remove_item(item=婀0 ²
驥瘏끋?by 쿒 ($994.245374416), cart=v1)
================================================== 1 failed in 0.51 seconds ==================================================
```

} steps?!

1. new_cart — $t_1 == 0$
2. add_item — $t_2 > 0$
3. add coupon — $t_2 > t_3 > 0$
4. remove_item — $t_4 < 0$ ⇒ FAIL!

t_n = total at step n

The fix

```
class ShoppingCart(object):
    def __init__(self):
        self.cart = {}
        self.discounts = {}

    @property
    def total(self):
        subtotal = 0.00
        for item, num in self.cart.items():
            subtotal += item.price * num
        for discount in self.discounts.values():
            subtotal *= discount
        return subtotal
```

self.total ??

David MacIver

- Author & sole maintainer
- Hypothesis = hours of free work
- Looking for sponsorship(s)
- Teaches Hypothesis workshops
- Consults

hypothesis.works
@DRMacIver

~ lots of good posts on Hypothesis

Slides + Code

github.com/dkua/pyconca16-talk

Feedback or Questions

@davidkua

david@kua.io

Thank You!!