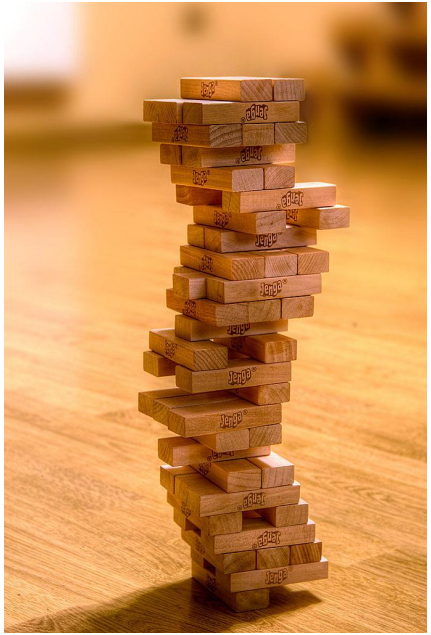
Property-Based Testing

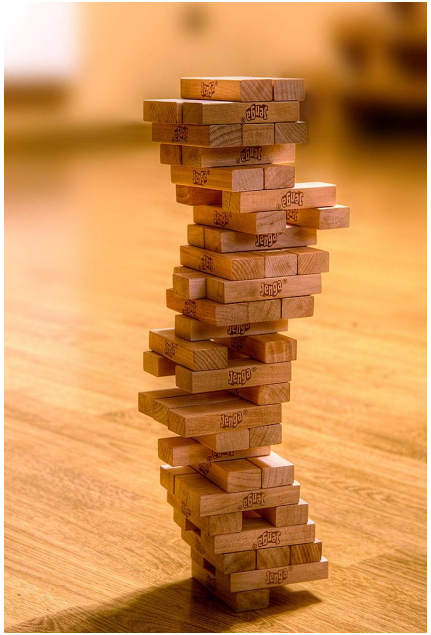
Matt Bachmann @mattbachmann

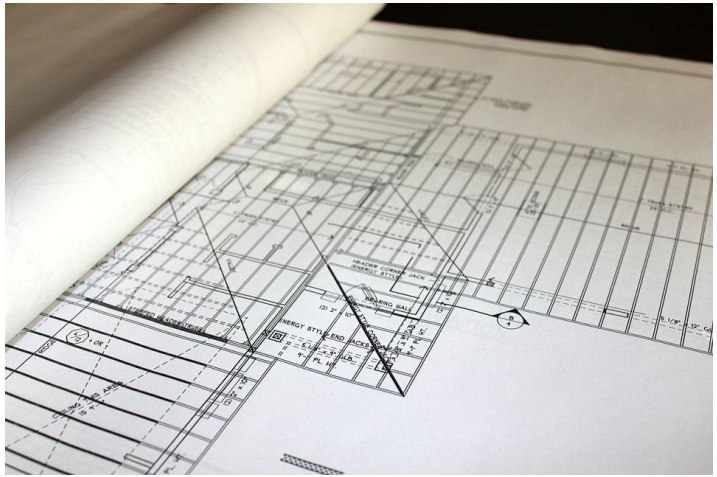
fitbit®

Testing is Important

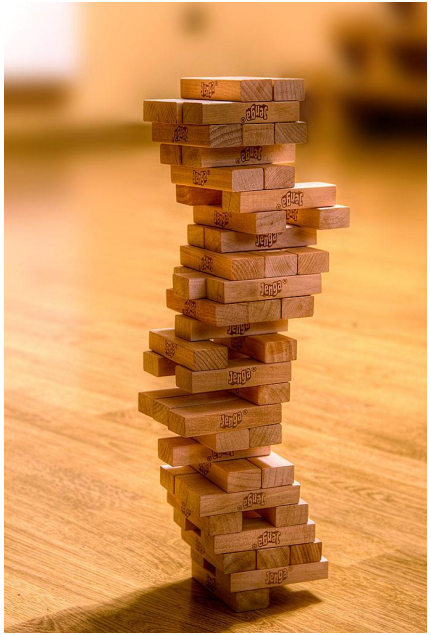


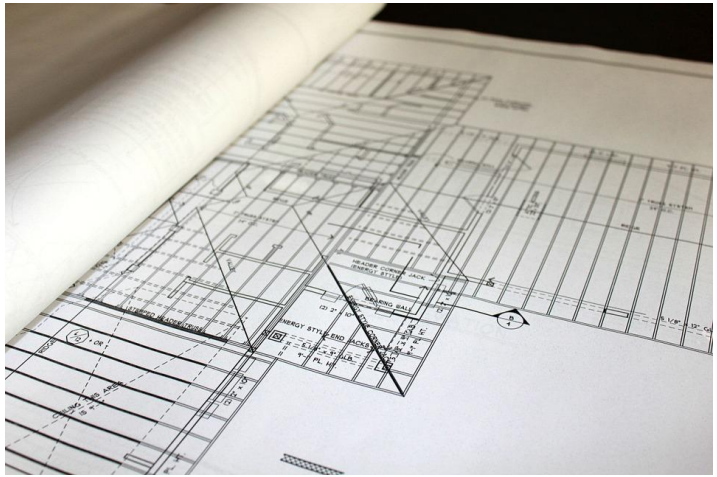
Testing is Important

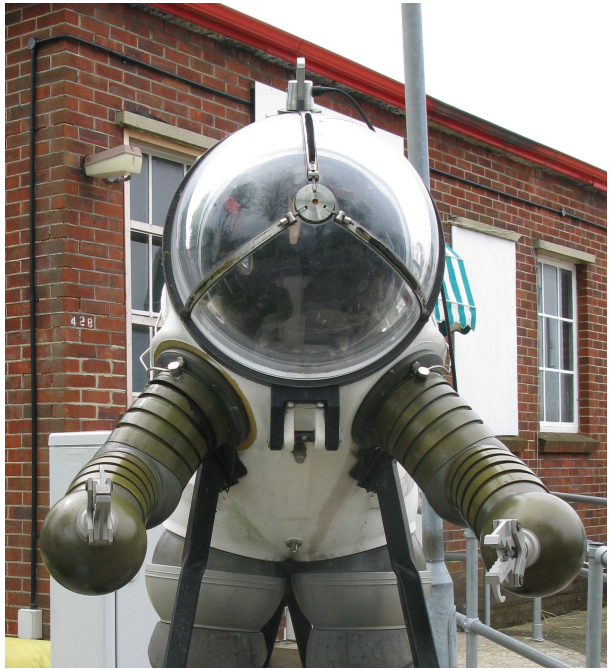




Testing is Important

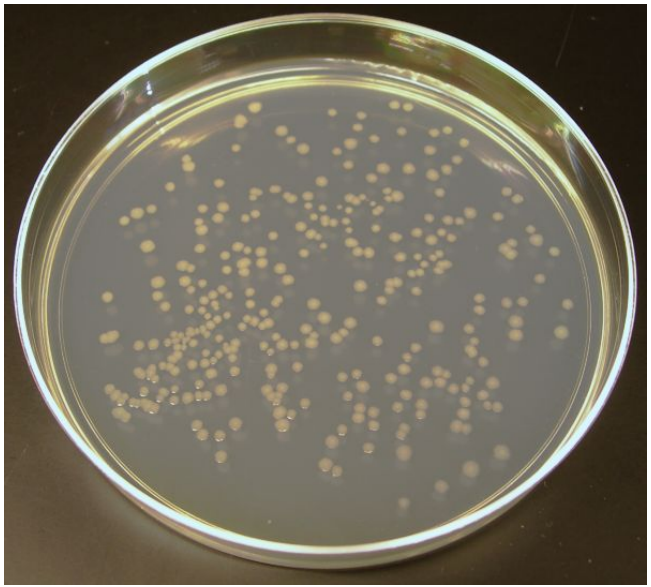




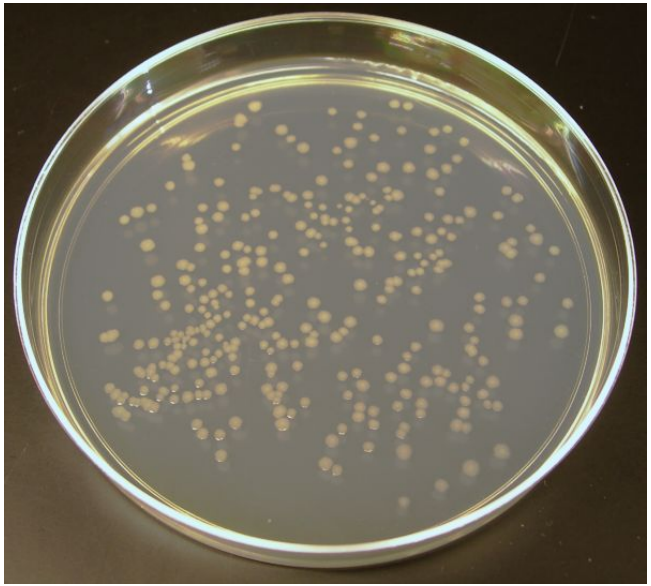


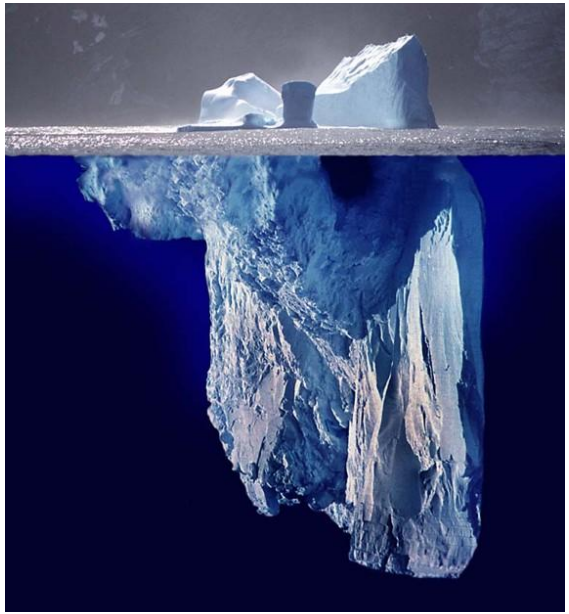
Testing is *Hard*

Testing is *Hard*



Testing is *Hard*





Capture the Important Cases

Minimize The Coding Overhead

Sorting a list of integers

```
def test_sort_empty(xs):
    assert quicksort([]) == []

def test_sorted(xs):
    assert quicksort([1, 2, 3]) == [1, 2, 3]

def test_sort_unsorted(xs):
    assert quicksort([5, 4]) == [4, 5]
```

Property Based Testing

- Describe the arguments
- Describe the result
- Have the computer try to prove your code wrong

- **The Arguments**
  - List of Integers
- **Properties of the Result**
  - List
  - All elements preserved
  - Results are in ascending order

Hypothesis http://hypothesis.works/

Inspired by Haskell's QuickCheck

Fantastic Docs

Offers training/contracting

David R. MacIver
DRMacIver

```
@given(st.lists(st.integers()))
def test_sort(xs):
    sorted_xs = quicksort(xs)
    assert isinstance(sorted_xs, list)
    assert Counter(xs) == Counter(sorted_xs)
    assert all(
        x <= y for x, y in
        zip(sorted_xs, sorted_xs[1:])
    )
```

```
@given(st.lists(st.integers()))
def test_sort(xs):
    sorted_xs = quicksort(xs)
    assert isinstance(sorted_xs, list)
    assert Counter(xs) == Counter(sorted_xs)
    assert all(
        x <= y for x, y in
        zip(sorted_xs, sorted_xs[1:])
    )
```

```
@given(st.lists(st.integers()))
def test_sort(xs):
    sorted_xs = quicksort(xs)
    assert isinstance(sorted_xs, list)
    assert Counter(xs) == Counter(sorted_xs)
    assert all(
        x <= y for x, y in
        zip(sorted_xs, sorted_xs[1:])
    )
```

```
@given(st.lists(st.integers()))
def test_sort(xs):
    sorted_xs = quicksort(xs)
    assert isinstance(sorted_xs, list)
    assert Counter(xs) == Counter(sorted_xs)
    assert all(
        x <= y for x, y in
        zip(sorted_xs, sorted_xs[1:])
    )
```

```
@given(st.lists(st.integers()))
def test_sort(xs):
    sorted_xs = quicksort(xs)
    assert isinstance(sorted_xs, list)
    assert Counter(xs) == Counter(sorted_xs)
    assert all(
        x <= y for x, y in
        zip(sorted_xs, sorted_xs[1:])
    )
```

```
@given(st.lists(st.integers()))
def test_sort(xs):
    sorted_xs = quicksort(xs)
    assert isinstance(sorted_xs, list)
    assert Counter(xs) == Counter(sorted_xs)
    assert all(
        x <= y for x, y in
        zip(sorted_xs, sorted_xs[1:])
    )
```

```
@given(st.lists(st.integers()))
def test_sort(xs):
    sorted_xs = quicksort(xs)
    assert isinstance(sorted_xs, list)
    assert Counter(xs) == Counter(sorted_xs)
    assert all(
        x <= y for x, y in
        zip(sorted_xs, sorted_xs[1:])
    )
```

```
@given(st.lists(st.integers()))

test_sort([])

test_sort([0])

test_sort([93932932923, 82883982983838])

test_sort([9999,77,2,3,3,100,3,39,3993])
```

```
========================= test session starts ==========================
platform darwin -- Python 2.7.11, pytest-2.9.1, py-1.4.31, pluggy
-0.3.1
rootdir: /Users/bachmann/Desktop/test, inifile:
plugins: hypothesis-3.1.0
collected 1 items

sort.py .

====================== 1 passed in 0.07 seconds ========================
```

Check Out Other Languages

Haskell - QuickCheck

Scala - ScalaCheck

Java - QuickTheories

[-3223, -9999999999, 32]

[-3223, -99999999999, 32]

[-3223, -99999999999]

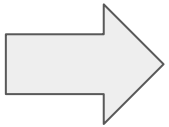
[-3223, -99999999999, 32]

[-3223, -99999999999]

[1, 0]

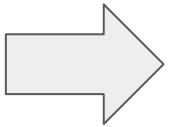
@given

```
@given(text())
def demonstrate(s):
    print s
```



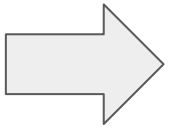
“10111□2”
“1120蔀â”
“Ksdjkjadasdamaslk889
1h2ne d-1dni2hd”

```
@given(integers())
def demonstrate(s):
    print s
```



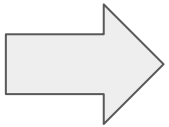
```
12312312
0
-99423
```

```
@given(lists(integers()))
def demonstrate(s):
    print s
```



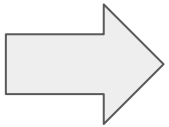
```
[]
[1,3,23,42]
[-1]
```

```
@given(
    dictionaries(
        integers(), list(text())
    )
)
def demonstrate(s):
    print s
```



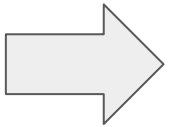
{323 : [“321312321”]}
{8382: [“”, “!1”, “asdas"}
{111: [“saodjad”]}

```
@given(
    builds(
        func,
        integers()
    )
)
def demonstrate(s):
    print s
```



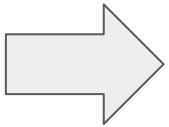
```
func(0)
func(31321)
func(-21)
```

```
@given(
    sampled_from(
        [‘A’, ‘B’, ‘C’]
    )
)
def demonstrate(s):
    print s
```



‘A’
‘A’
‘C’

```
@given(
    recursive(
        booleans()
        lists
    )
)
def demonstrate(s):
    print s
```



```
True
[True]
[True, [[False, True]]]
```

```
@given(
    st.builds(
        Dog,
        breed=st.text(),
        name=st.text(),
        height=st.floats(),
        weight=st.floats()
    )
)
```

```
@given(
    st.builds(
        Dog,
        breed=st.sampled_from(KNOWN_BREEDS),
        name=st.text(),
        height=st.floats(),
        weight=st.floats()
    )
)
```

```
@given(
    st.builds(
        Dog,
        breed=st.sampled_from(KNOWN_BREEDS),
        name=st.text(min_size=5),
        height=st.floats(),
        weight=st.floats()
    )
)
```

```
@given(
  st.builds(
      Dog,
      breed=st.sampled_from(KNOWN_BREEDS),
      name=st.text(min_size=5),
      height=st.floats(
          min_value=1, max_value=6, allow_nan=False, allow_infinity=False
      ),
      weight=st.floats(
          min_value=1, max_value=300, allow_nan=False, allow_infinity=False
      ),
  )
)
```

```
@given(
  st.builds(
      Dog,
      breed=st.sampled_from(KNOWN_BREEDS),
      name=st.text(min_size=5),
      height=st.floats(
          min_value=1, max_value=6, allow_nan=False, allow_infinity=False
      ),
      weight=st.floats(
          min_value=1, max_value=300, allow_nan=False, allow_infinity=False
      ),
  )
)
@example(Dog(breed="Labrador", name="Spot", height=2.1, weight=70))
```

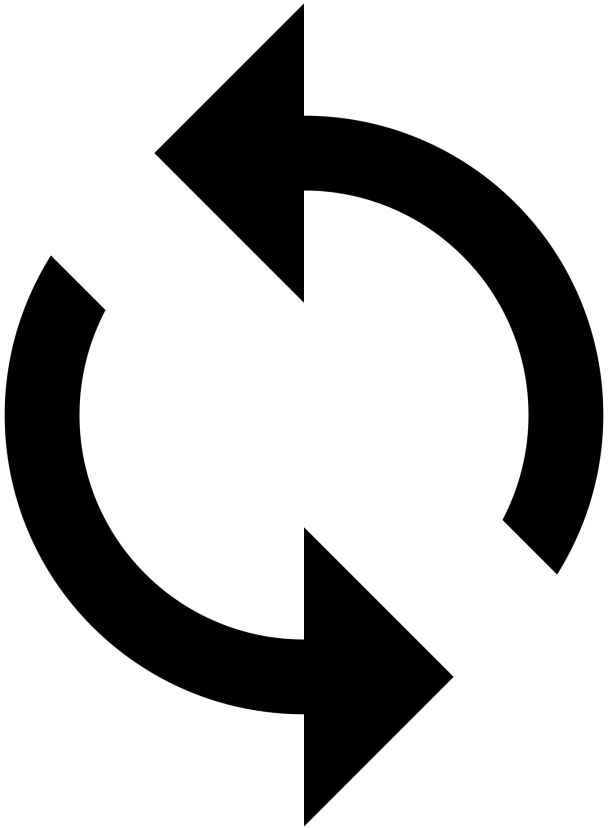
Potentially infinite cases

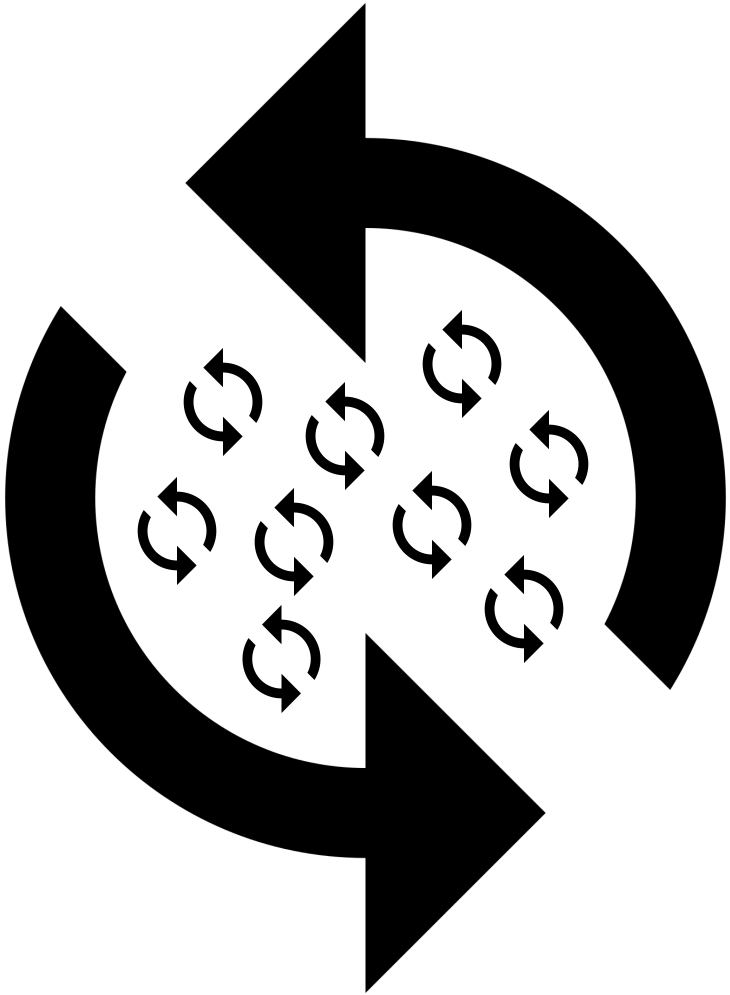
Nothing hardcoded

Very little code

Slow Tests Make Their Voices Heard

Not Super Friendly to TDD





Properties Require More Thought

Pattern 1:

The code should not explode

/batputer/criminals/aliases/**{id}**?
sort=**{sort}**&max_results=**{max}**

- JSON response
- Expected response codes
  - 200 - OK
  - 401 - Forbidden
  - 400 - Invalid data
  - 404 - Not Found

```
@given(st.integers(),
       st.text(),
       st.integers()))
def test_no_explosion(id, sort, max):
    response = requests.get(
        BATPUTER_URL.format(id, sort, max)
    )
    assert response and response.json()
    assert (response.status_code in
            [200, 401, 400, 404])
```

```
@given(st.integers(),
       st.text(),
       st.integers()))
def test_no_explosion(id, sort, max):
    response = requests.get(
        BATPUTER_URL.format(id, sort, max)
    )
    assert response and response.json()
    assert (response.status_code in
            [200, 401, 400, 404])
```

```
@given(st.integers(),
       st.text(),
       st.integers()))
def test_no_explosion(id, sort, max):
    response = requests.get(
        BATPUTER_URL.format(id, sort, max)
    )
    assert response and response.json()
    assert (response.status_code in
            [200, 401, 400, 404])
```

```
@given(st.integers(),
       st.text(),
       st.integers()))
def test_no_explosion(id, sort, max):
    response = requests.get(
        BATPUTER_URL.format(id, sort, max)
    )
    assert response and response.json()
    assert (response.status_code in
            [200, 401, 400, 404])
```

```
@given(st.integers(),
       st.text(),
       st.integers()))
def test_no_explosion(id, sort, max):
    my_cool_function(id, sort max)
```

```
@given(st.integers(),
       st.text(),
       st.integers()))
def test_no_explosion(id, sort, max):
    try:
        my_cool_function(id, sort max)
    except ValueError:
        pass
```

Pattern 2:

Reversible Operations

Encoding

Undo Operations

Serialization

Encoding

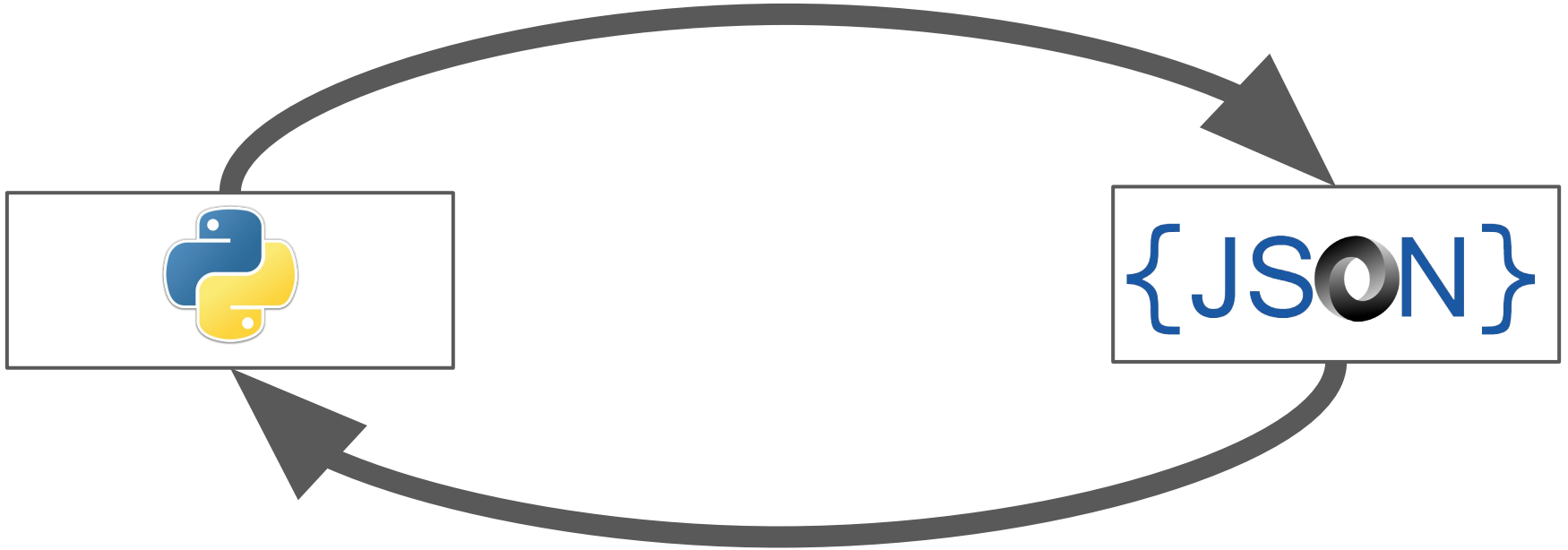
Undo Operations

Serialization

```
class HistoricalEvent(object):
    def __init__(self, id, desc, time):
        self.id = id
        self.desc = desc
        self.time = time
```

```
class EventEncoder(json.JSONEncoder):
    def default(self, obj):
        if isinstance(obj, HistoricalEvent):
            return {
              "id": obj.id
              "description": obj.description
              "event_time":obj.event_time.isoformat()
              }
        return json.JSONEncoder.default(self, obj)
```

```
def fromJson(json_str):
    dct = json.loads(json_str)
    return HistoricalEvent(
      dct['id'],
      dct['description'],
      dateutil.parser.parse(
          dct['event_time']
       )
   )
```


{JSON}

```
@given(st.integers(), st.text(), datetimes(timezones=['UTC']))
def test_to_from_json(id, desc, date):
    original_event = HistoricalEvent(id, desc, date)
    encoded_event = json.dumps(event, cls=EventEncoder)
    decoded_event = json.loads(
        encoded_event,
        object_hook=HistoricalEvent.fromJson
    )
    assert decoded_event == original_event
```

```
@given(st.integers(), st.text(), datetimes(timezones=['UTC']))
def test_to_from_json(id, desc, date):
    original_event = HistoricalEvent(id, desc, date)
    encoded_event = json.dumps(event, cls=EventEncoder)
    decoded_event = json.loads(
        encoded_event,
        object_hook=HistoricalEvent.fromJson
    )
    assert decoded_event == original_event
```

```
@given(st.integers(), st.text(), datetimes(timezones=['UTC']))
def test_to_from_json(id, desc, date):
    original_event = HistoricalEvent(id, desc, date)
    encoded_event = json.dumps(event, cls=EventEncoder)
    decoded_event = json.loads(
        encoded_event,
        object_hook=HistoricalEvent.fromJson
    )
    assert decoded_event == original_event
```

```
@given(st.integers(), st.text(), datetimes(timezones=['UTC']))
def test_to_from_json(id, desc, date):
    original_event = HistoricalEvent(id, desc, date)
    encoded_event = json.dumps(event, cls=EventEncoder)
    decoded_event = json.loads(
        encoded_event,
        object_hook=HistoricalEvent.fromJson
    )
    assert decoded_event == original_event
```

```
@given(st.integers(), st.text(), datetimes(timezones=['UTC']))
def test_to_from_json(id, desc, date):
    original_event = HistoricalEvent(id, desc, date)
    encoded_event = json.dumps(event, cls=EventEncoder)
    decoded_event = json.loads(
        encoded_event,
        object_hook=HistoricalEvent.fromJson
    )
    assert decoded_event == original_event
```

When possible dont convert years when the year has been specified past two digits #96

Edit

Open **Bachmann1234** wants to merge 3 commits into `dateutil:master` from `Bachmann1234:handle-pre-100-dates-when-possible`

Conversation 1 | Commits 3 | Files changed 3 | +56 −10

Bachmann1234 commented 21 hours ago

Howdy!

While playing around with dateutil parsing I found what I think is a bug

```
datetime(99, 1, 1, 0, 0).isoformat()
'0099-01-01T00:00:00'

dateutil.parse('0099-01-01T00:00:00')
datetime.datetime(1999, 1, 1, 0, 0)
```

This PR tries to address this. When the library can identify a single possible year token, if that year is greater than 3 digits it will assume the year is correct rather than coercing it.

I also noticed problems with years like 0039. So I tried to address that as well

A better solution would be to detect padding zeros and handle those tokens differently... but I was struggling with that so I took the post processing approach instead.

When possible dont convert years when the year has been specified pas… ··· ✓ 9cfbe67

Labels

None yet

Milestone

No milestone

Assignee

No one assigned

Notifications

Unsubscribe

You're receiving notifications because you authored the thread.

1 participant

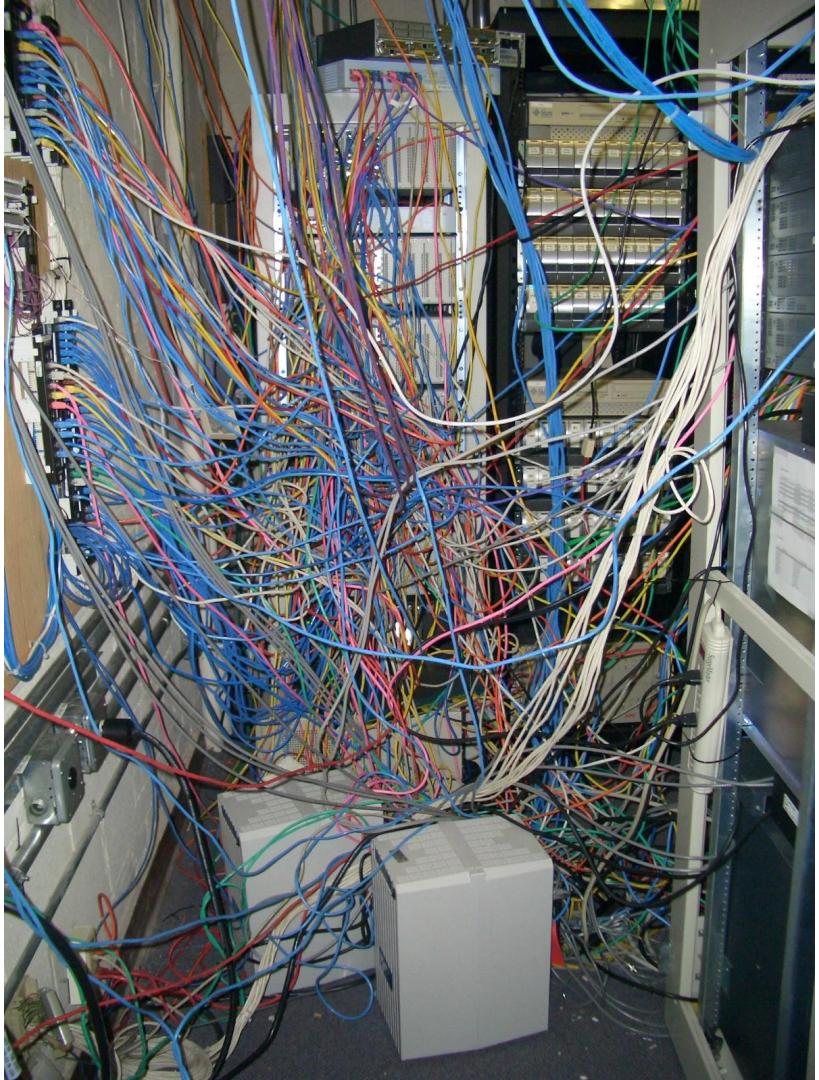
Pattern 3:

Testing Oracle

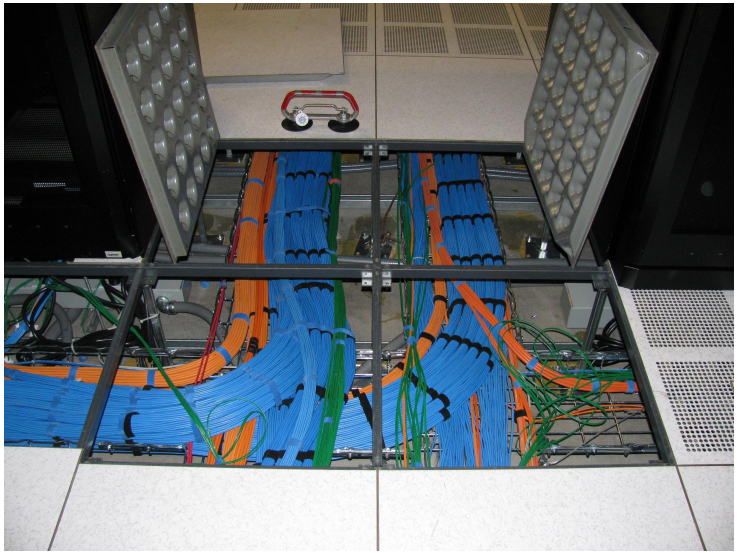
Optimizing

Refactoring

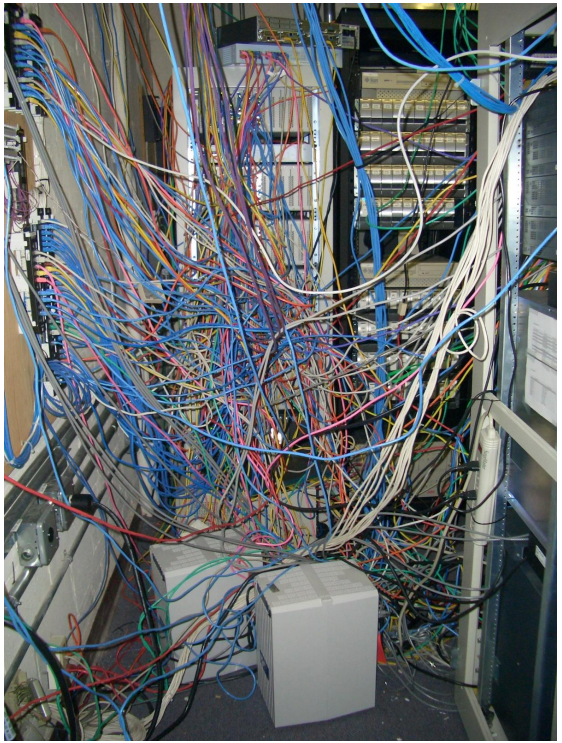
Emulating



Leave It Alone



=



```
@given(st.integers(), st.text())
def test_against_legacy(arg1, arg2):
  assert (
    new_hotness(arg1, arg2)
    ==
    legacy_system(arg1, arg2)
  )
```

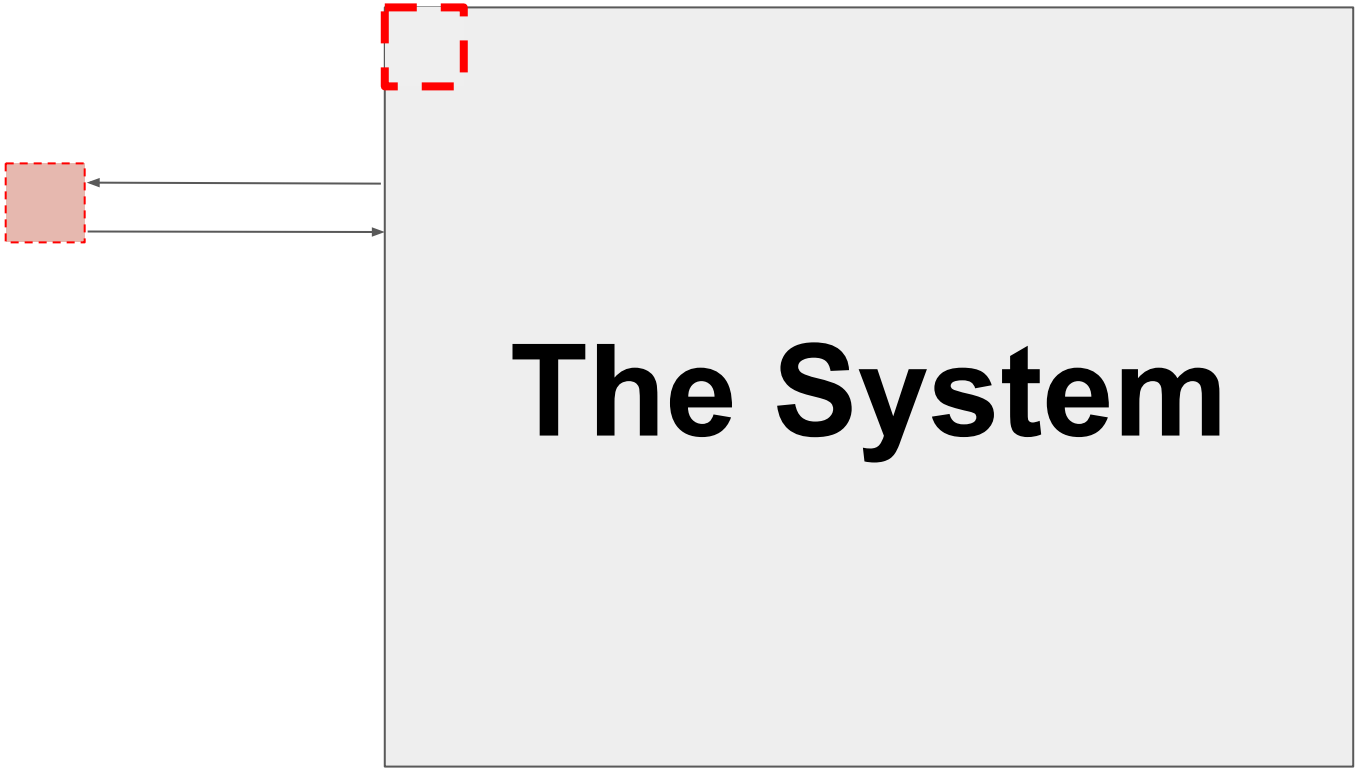
Compare Against Brute Force

```
@given(st.lists(st.integers()))
def test_against_brute_force(input):
    assert (
        easy_but_inefficent(input)
        ==
        optimized(input)
    )
```

The System

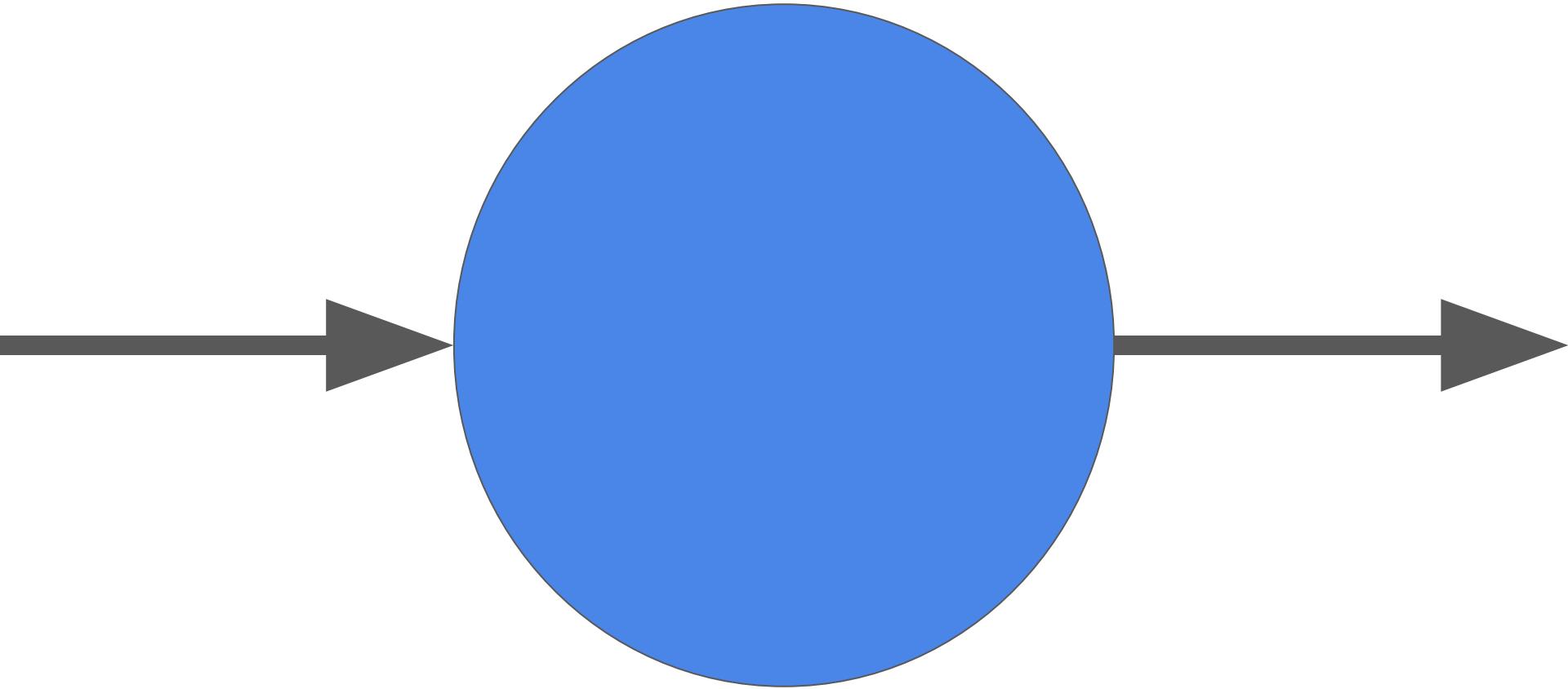

The System

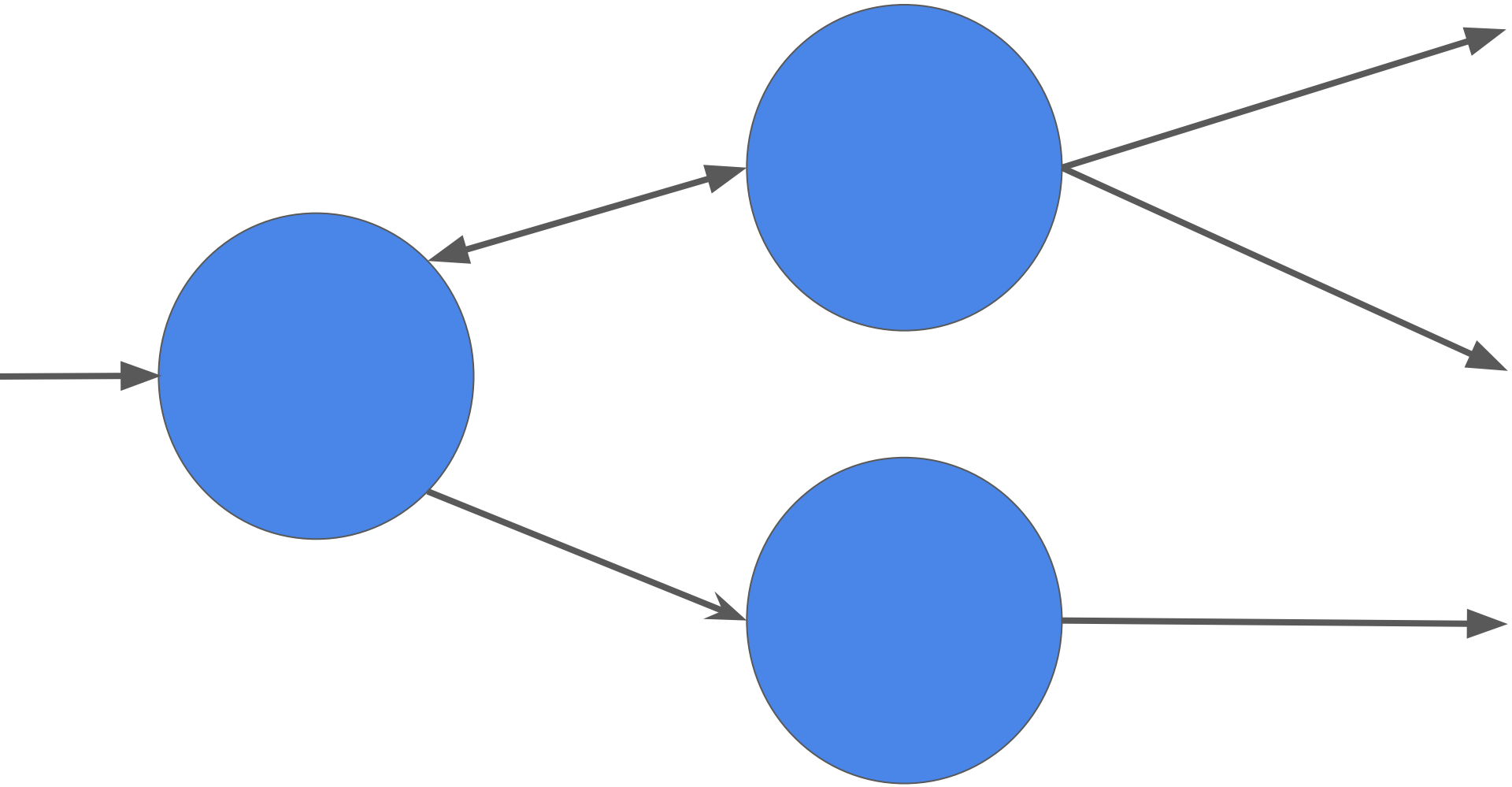

The System

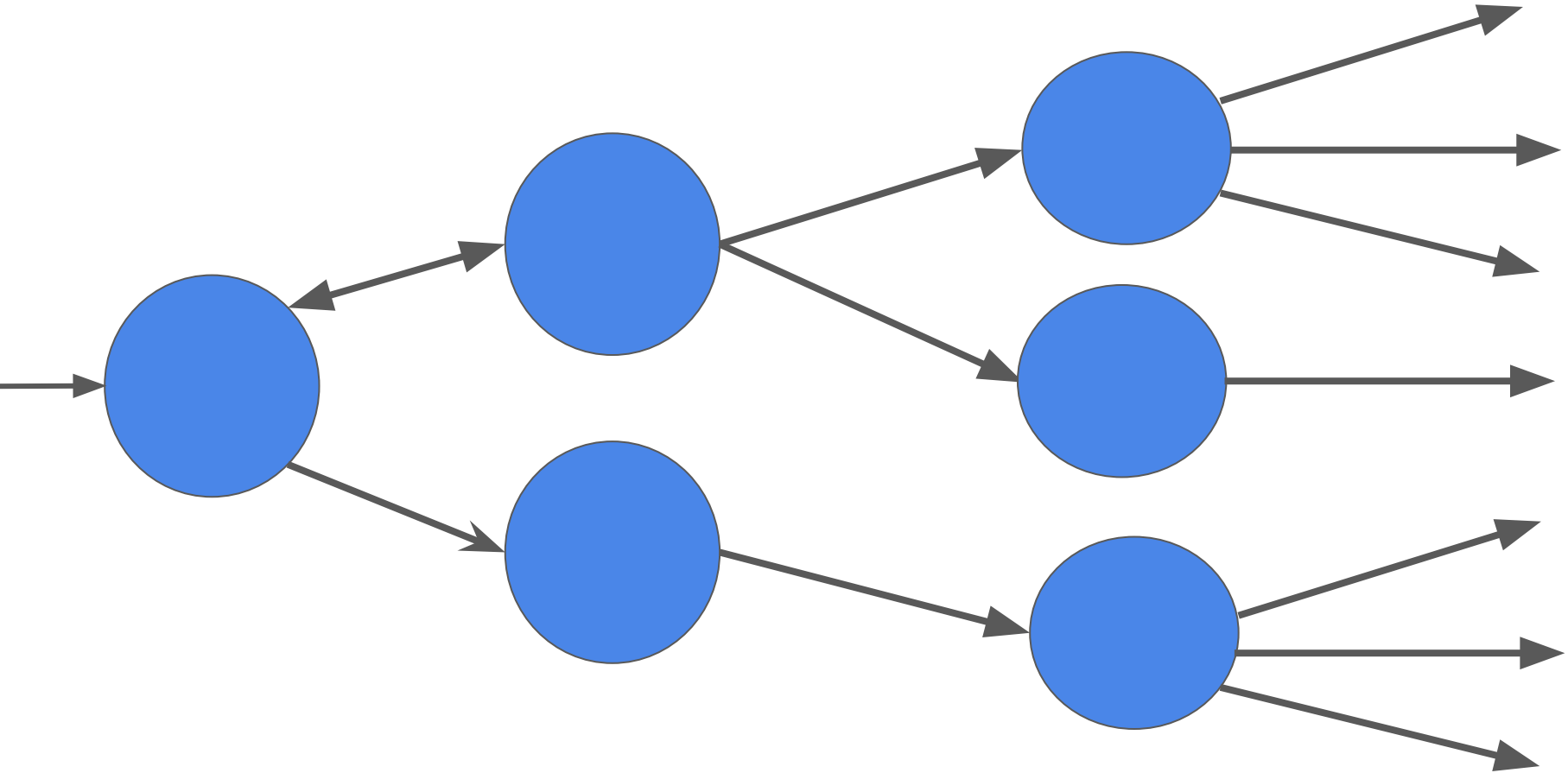

The System

Pattern 4:

Invariants

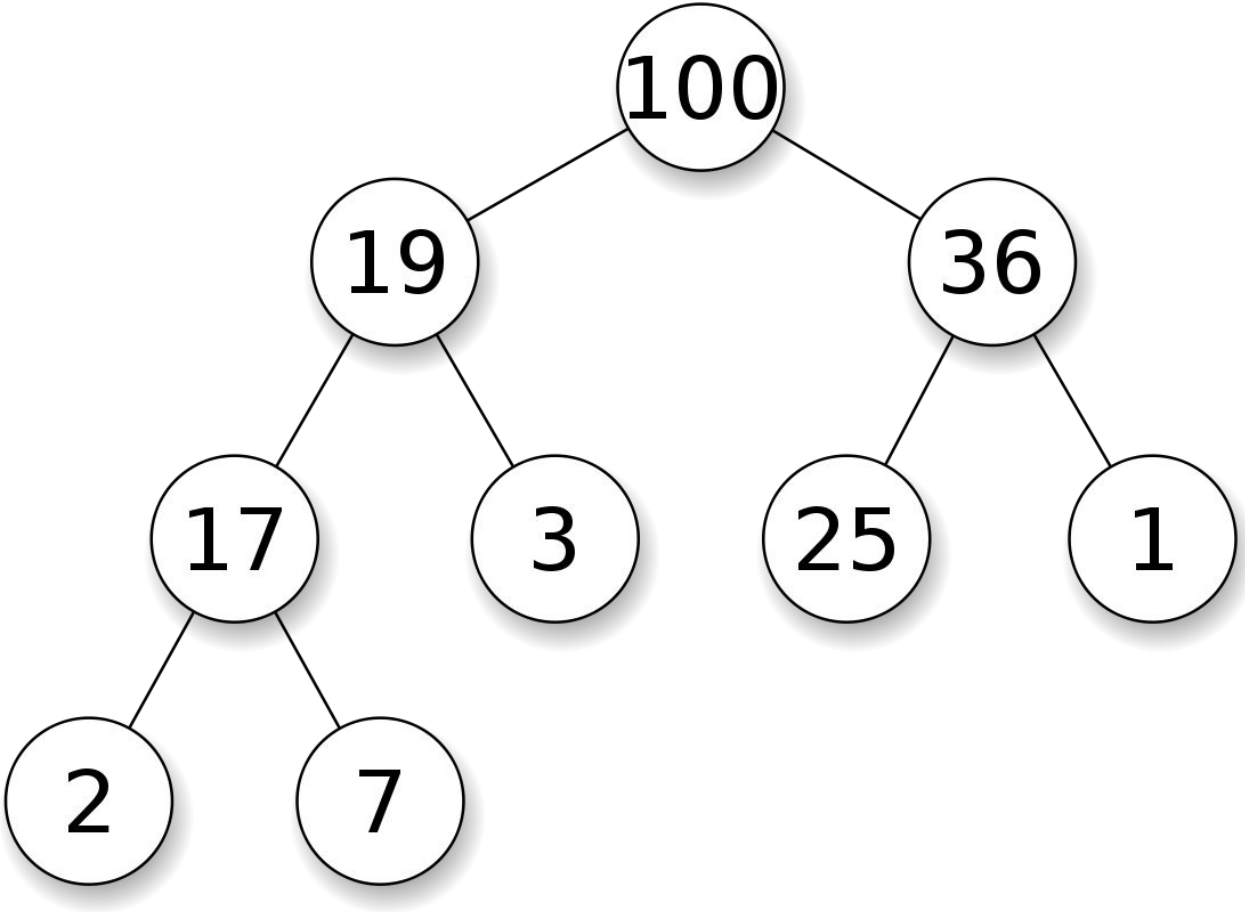






Stateful Tests

- Define a state
- What operations can happen in what conditions?
- How do operations affect the state?
- What must be true for each step?



https://commons.wikimedia.org/wiki/File:Max-Heap.svg

http://hypothesis.works/articles/rule-based-stateful-testing/

Invariant

- No matter what series of operations is performed the head of the tree must be the max element

__init__

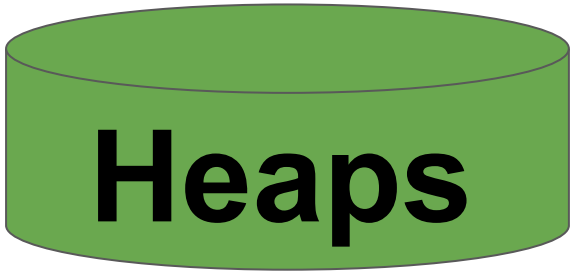
push

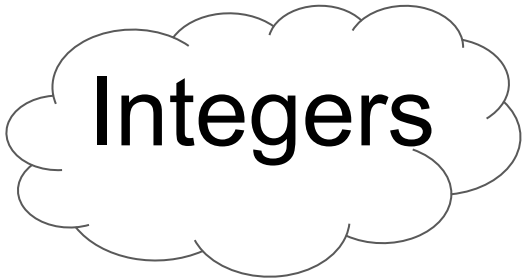
pop

merge

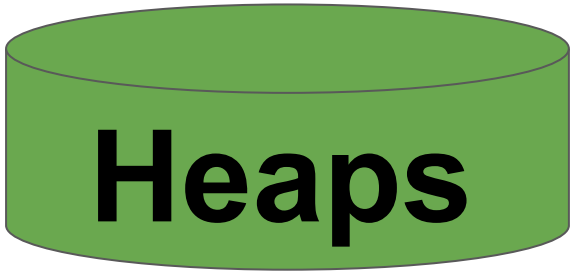
http://hypothesis.works/articles/rule-based-stateful-testing/

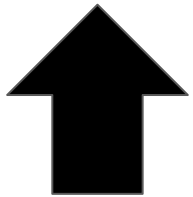

FIND. ME. BUGS.



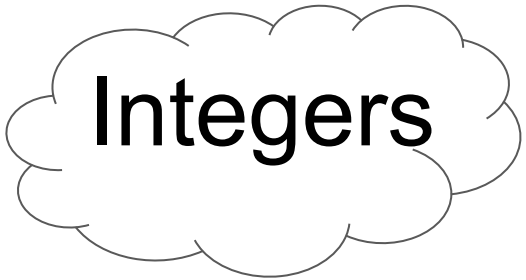


http://hypothesis.works/articles/rule-based-stateful-testing/

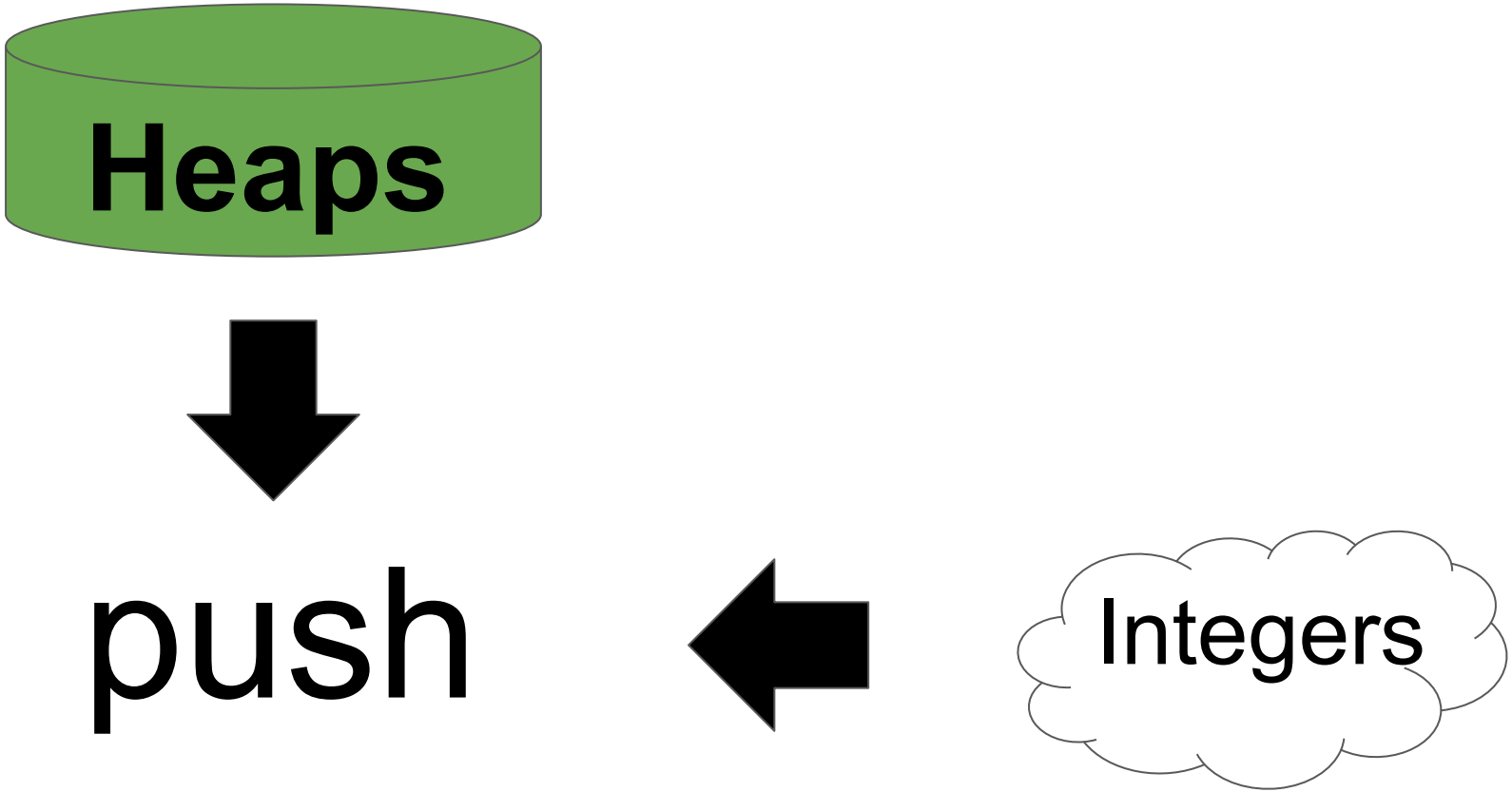




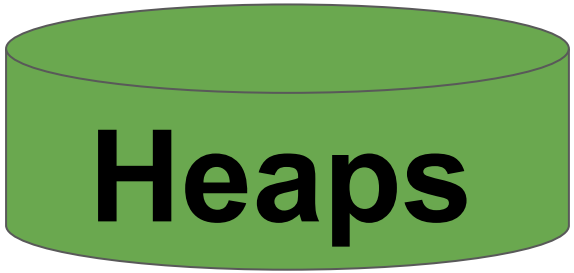
__init__

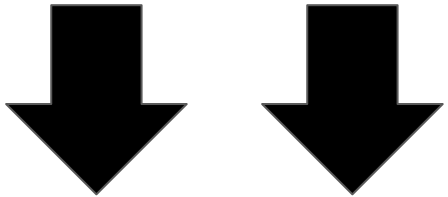


http://hypothesis.works/articles/rule-based-stateful-testing/

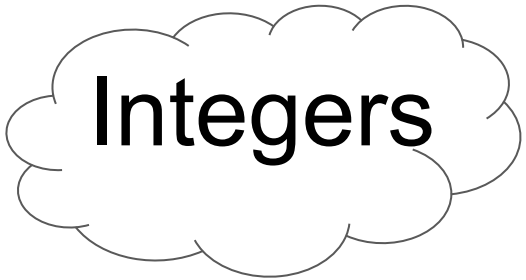


http://hypothesis.works/articles/rule-based-stateful-testing/

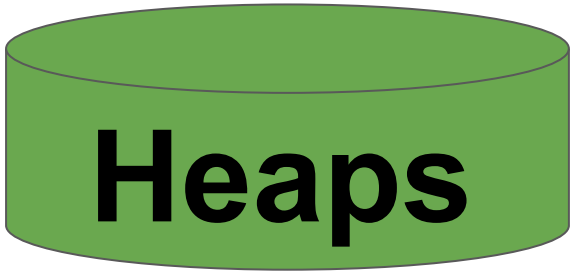


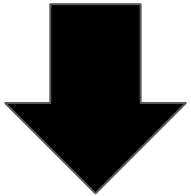


merge

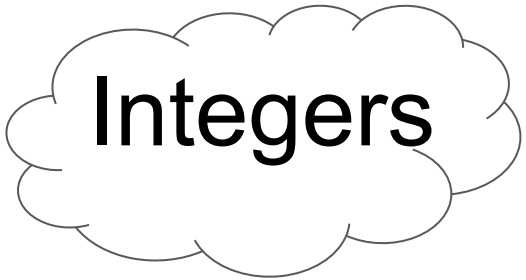


http://hypothesis.works/articles/rule-based-stateful-testing/





pop



assert result actually max

http://hypothesis.works/articles/rule-based-stateful-testing/

```
class HeapMachine(RuleBasedStateMachine):
    Heaps = Bundle('heaps')

    @rule(target=Heaps)
    def new_heap(self):
        return Heap()

    @rule(heap=Heaps, value=integers())
    def heap_push(self, heap, value):
        push(heap, value)
```

http://hypothesis.works/articles/rule-based-stateful-testing/

```
class HeapMachine(RuleBasedStateMachine):
    Heaps = Bundle('heaps')

    @rule(target=Heaps)
    def new_heap(self):
        return Heap()

    @rule(heap=Heaps, value=integers())
    def heap_push(self, heap, value):
        push(heap, value)
```

http://hypothesis.works/articles/rule-based-stateful-testing/

```
class HeapMachine(RuleBasedStateMachine):
    Heaps = Bundle('heaps')

    @rule(target=Heaps)
    def new_heap(self):
        return Heap()

    @rule(heap=Heaps, value=integers())
    def heap_push(self, heap, value):
        push(heap, value)
```

http://hypothesis.works/articles/rule-based-stateful-testing/

```
class HeapMachine(RuleBasedStateMachine):
    Heaps = Bundle('heaps')

    @rule(target=Heaps)
    def new_heap(self):
        return Heap()

    @rule(heap=Heaps, value=integers())
    def heap_push(self, heap, value):
        push(heap, value)
```

http://hypothesis.works/articles/rule-based-stateful-testing/

```
@rule(target=Heaps, heap1=Heaps, heap2=Heaps)
def merge(self, heap1, heap2):
    return heap_merge(heap1, heap2)
```

```
@rule(heap=Heaps.filter(bool))
def pop(self, heap):
    correct = max(list(heap))
    result = heap_pop(heap)
    assert correct == result
```

http://hypothesis.works/articles/rule-based-stateful-testing/

```
@rule(target=Heaps, heap1=Heaps, heap2=Heaps)
def merge(self, heap1, heap2):
    return heap_merge(heap1, heap2)
```

```
@rule(heap=Heaps.filter(bool))
def pop(self, heap):
    correct = max(list(heap))
    result = heap_pop(heap)
    assert correct == result
```

http://hypothesis.works/articles/rule-based-stateful-testing/


FIND. ME. BUGS.

```
        @rule(heap=Heaps.filter(bool))
        def pop(self, heap):
            correct = max(list(heap))
            result = heap_pop(heap)
>           assert correct == result
E AssertionError: assert 1 == 0
```

http://hypothesis.works/articles/rule-based-stateful-testing/

```
v1 = new_heap()
push(heap=v1, value=0)
push(heap=v1, value=1)
push(heap=v1, value=1)
v2 = merge(heap2=v1, heap1=v1)
pop(heap=v2)
pop(heap=v2)
```

http://hypothesis.works/articles/rule-based-stateful-testing/

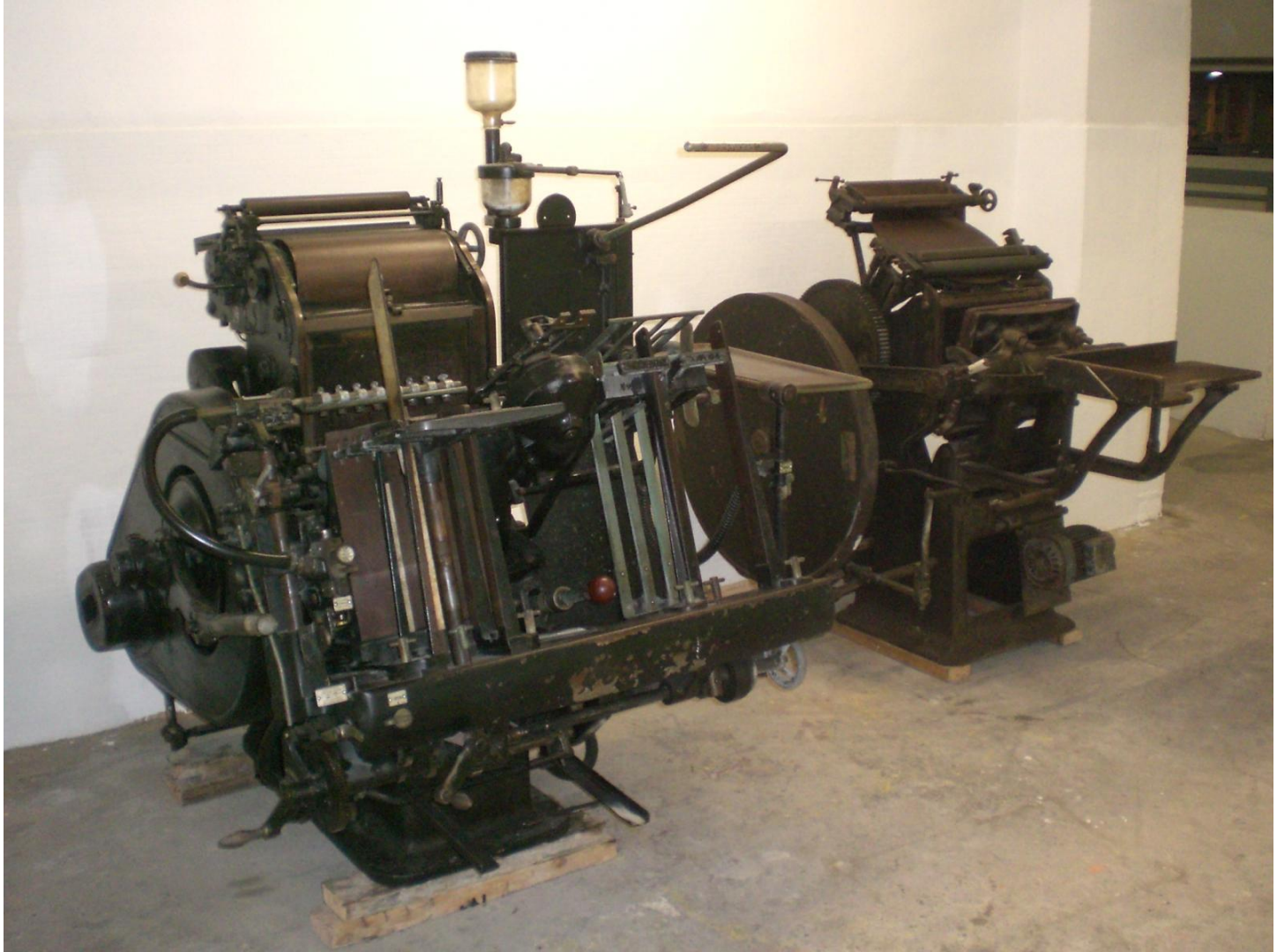
```
self = HeapMachine({'heaps': [VarReference(name=u'v1')]}), heap = [0]

    @rule(heap=Heaps.filter(bool))
    def pop(self, heap):
        correct = min(heap)
        result = heappop(heap)
        if correct != result:
            pass
>       assert correct == result
E       AssertionError: assert 0 == 1

tests/test_code.py:62: AssertionError
---------------------------- Captured stdout call -----------------
-
Step #1: v1 = newheap()
Step #2: push(heap=v1, value=1)
Step #3: push(heap=v1, value=0)
Step #4: push(heap=v1, value=0)
Step #5: pop(heap=v1)
Step #6: pop(heap=v1)
========================== 1 failed in 0.64 seconds ================
```

Property Based Testing

- Describe the arguments
- Describe the result
- Have the computer try to prove your code wrong



Thanks!

Matt Bachmann
Bachmann1234

Matt Bachmann
https://github.com/Bachmann1234

Twitter: @mattbachmann

Slides: https://goo.gl/LbeIRE

More about Hypothesis

- http://hypothesis.works/

More On Property Based Testing In General

- http://www.quviq.com/
- https://fsharpforfunandprofit.com/posts/property-based-testing-2/

Testing Eventual Consistency with RIAK

- https://www.youtube.com/watch?v=x9mW54GJpG0

Images

https://www.flickr.com/photos/t0fugurl/2507049701
Picture (without crossout) by Leanne Poon
https://creativecommons.org/licenses/by-nc-nd/2.0/

https://upload.wikimedia.org/wikipedia/commons/3/36/Under_Floor_Cable_Runs_Tee.jpg

https://commons.wikimedia.org/wiki/File:HK_Jockey_Club_Creative_Arts_Centre_JCCAC_Old_Machine_Printers.JPG

https://upload.wikimedia.org/wikipedia/commons/4/4e/The_Thinker_by_Rodin_at_the_Cantor_Arts_Center_of_Stanford_University.JPG

https://upload.wikimedia.org/wikipedia/commons/3/39/Chip-pan-fire.jpg

https://www.flickr.com/photos/wonderlane/27784474073

https://www.flickr.com/photos/versageek/493800514
versageek
https://creativecommons.org/licenses/by-sa/2.0/

https://upload.wikimedia.org/wikipedia/commons/8/83/Jan_Fabre's_%22Searching_for_Ut
opia%22.jpg

https://upload.wikimedia.org/wikipedia/commons/thumb/1/19/Dtjohnnymonkey-Stopwatch-no-shading.svg/2000px-Dtjohnnymonkey-Stopwatch-no-shadi
ng.svg.png

https://upload.wikimedia.org/wikipedia/commons/thumb/1/1d/Speed_bump,_Segev%C3%A5ng,_Malm%C3%B6.jpg/640px-Speed_bump,_Segev%C3%A5ng,_Malm%C3%B6.jpg

Images

https://upload.wikimedia.org/wikipedia/commons/6/62/Bird%27s_Rat%27s_nest.jpg

https://upload.wikimedia.org/wikipedia/commons/a/ac/Iceberg.jpg

https://upload.wikimedia.org/wikipedia/commons/4/49/Sleepy_Kit_(6969930840).jpg

http://imgs.xkcd.com/xk3d/303/compiling_4.png

https://pixabay.com/static/uploads/photo/2015/08/18/20/33/blueprints-894779_960_720.jpg

https://upload.wikimedia.org/wikipedia/commons/thumb/6/6b/Jenga_distorted.jpg/686px-Jenga_distorted.jpg

https://upload.wikimedia.org/wikipedia/commons/5/5c/MSC_Tomoko_in_the_Santa_Barbara_Channel_-_IMO_9309461_(3898801499).jpg

https://upload.wikimedia.org/wikipedia/commons/thumb/e/e0/Git-logo.svg/2000px-Git-logo.svg.png

https://upload.wikimedia.org/wikipedia/commons/c/c3/Podlaskie_-_Czarna_Bia%C5%82ostocka_-_Puszcza_Knyszy%C5%84ska_-_G%C3%B3ry_Czuma%C5%BCowskie_-_E_-_v-NW.JPG

https://www.flickr.com/photos/petsadviser-pix/8126550573