Schemathesis: Property-based Testing for Web APIs

Test more with less

$ whoami

- 13+ years Python / 7 years Rust
- Interpreters, fuzzing, high-performance tools
- Hypothesis / JSON Schema / pytest
- @Stranger6667

Schemathesis

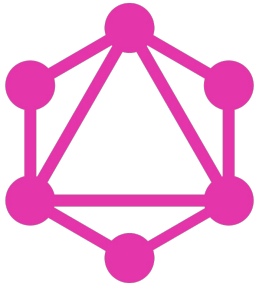
- Property-based Testing Framework for Web APIs
- Works with Open API & GraphQL
- Based on Hypothesis
- Written in Python & easily hackable

You can star it on GitHub

Property-based Testing for Web APIs?

- Random, structured inputs
- Finds edge cases, improves reliability
- Specification as the source of properties

Standards for Web APIs



Spec Compliance: The Eternal Challenge

Wasted time

Missing tests

Still buggy

Yes, your spec generator is buggy too

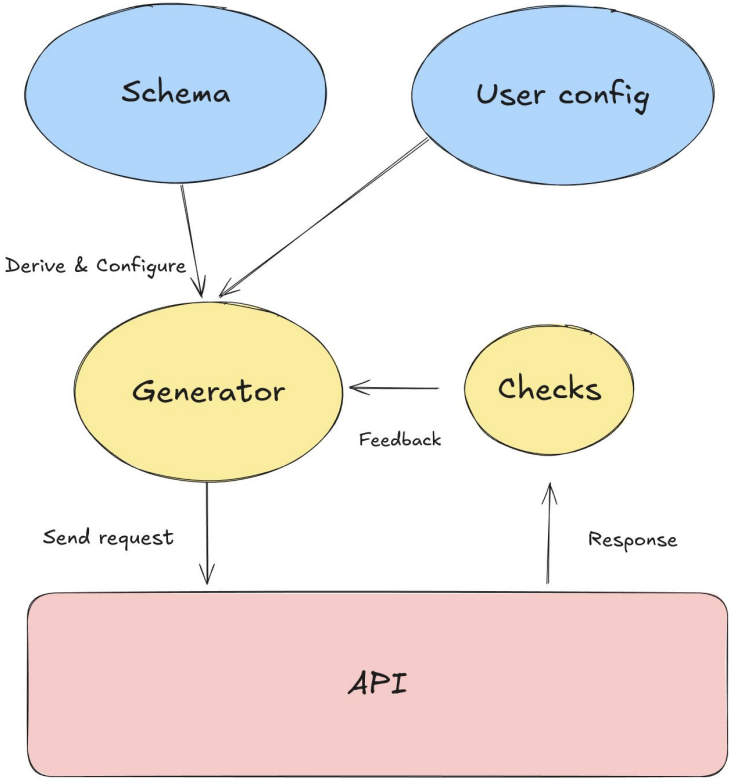
What properties?

Conformance

- Response
- Inputs
- Examples

- No Server Errors
- Response Time < N ms
- Access Control
- No Data leaks

How Schemathesis checks them?



- Positive & Negative
- Stateful & Stateless
- Performance Regressions
- Shrinking
- Reproduction snippet
- Coverage Reports

Case Studies

- Company X. 150 APIs & 8 ppl
  - -20% manual tests
  - 100x of conformance issues fixed
- Spotify Backstage
  - Plugins testing
- RedHat
  - Testing courses
- PayLead
- Princeton University, WordPress, and more

Research

- Deriving Semantics-Aware Fuzzers from Web API Schemas
- Testing RESTful APIs: A Survey
- Open Problems in Fuzzing RESTful APIs: A Comparison of Tools

> There are different levels of usability and documentation quality among these tools which have been reported, where Schemathesis clearly stands out among the most user-friendly and industry-strength tools.

Well Tested

- 4000+ schemas in Open API Directory
- Generated schema
- Gracefully handles schema errors

How does it work?

Open API

```yaml
openapi: 3.0.0
info:
  title: Book API
  version: 1.0.0
paths:
  /books:
    get:
      summary: List books
      parameters:
        - name: genre
          in: query
          required: false
          schema:
            type: string
            enum: [fiction, non-fiction, science, history]
      responses:
        '200':
          description: Successful response
          content:
            application/json:
              schema:
                type: array
```

Path

Method

Parameter

Response

Parameters

```yaml
/users:
  post:
    summary: Create a new user
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            properties:
              username:
                type: string
                minLength: 3
                maxLength: 20
    parameters:
      - in: query
        name: notify
        schema:
          type: boolean
        style: form
        explode: true
```

Schema

- type
- minLength
- maxLength
- …

Transformations

- Serialization (JSON / XML / etc)
- Query composition (style, etc)
- …

JSON Schema -> Hypothesis

```
{
  "minimum": 10
}
```

```
st.integers(min_value=10)
```

Optimizing Data Generation

```
{
  "allOf": [
    {
      "minimum": 10,
      "type": "integer"
    },
    {
      "type": "integer",
      "multipleOf": 3,
      "maximum": 42
    }
  ]
}
```

→

```
{
  "maximum": 42,
  "minimum": 12,
  "multipleOf": 3,
  "type": "integer"
}
```

Goal: Optimize generation

Optimizing Data Generation

```json
{
  "minimum": 10,
  "maximum": 50,
  "multipleOf": 10
}
```

```python
st.integers(
    min_value=10,
    max_value=50,
).filter(
    lambda x: x % 10 == 0
)
```

A lot of generated values are wasted

Optimizing Data Generation

```
st.sampled_from(
    [10, 20, 30, 40, 50]
)
```

Much more efficient

Optimizing Data Generation

```
{
  "allOf": [
    {
      "minimum": 10,
      "type": "number"
    },
    {
      "not": {
        "type": "integer",
        "multipleOf": 3
      }
    }
  ]
}
```

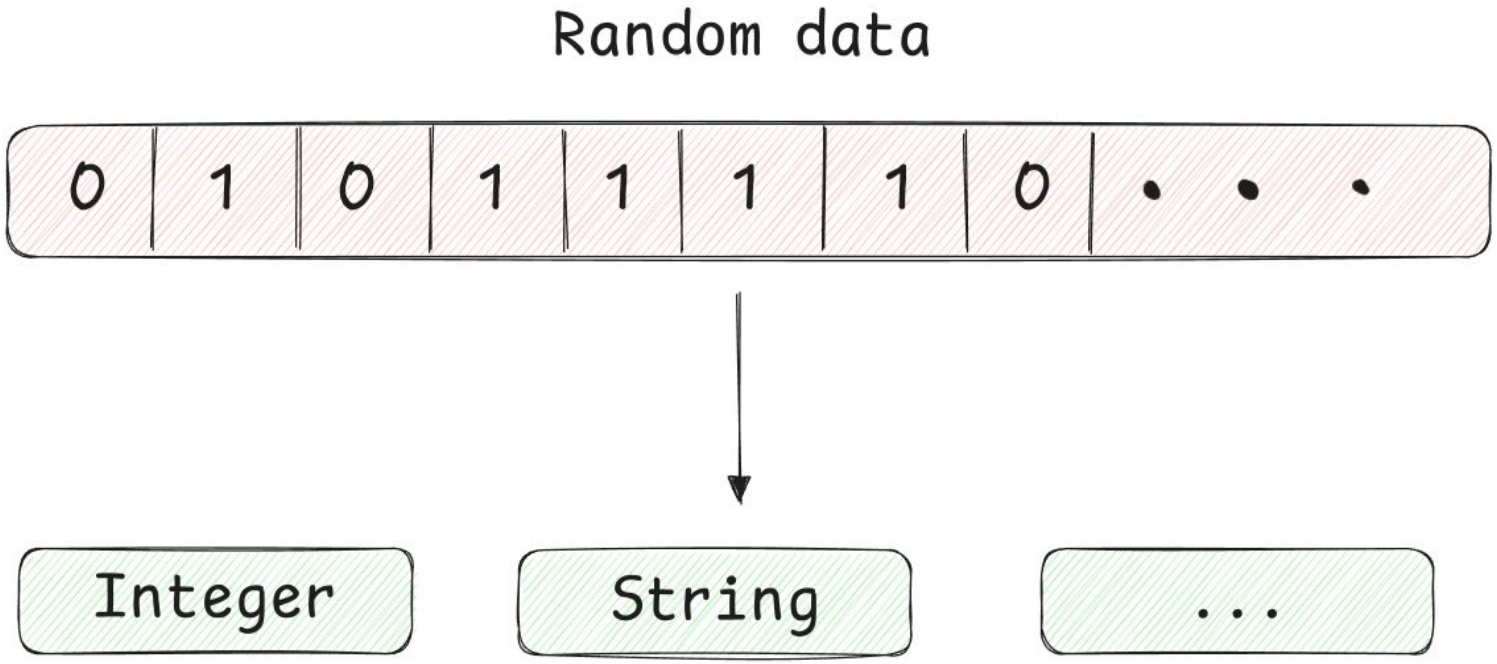
Canonicalisation is not always possible. But there is a [nice paper]() on schema equivalency

And [fast JSON Schema validators]()

What do we want to generate?

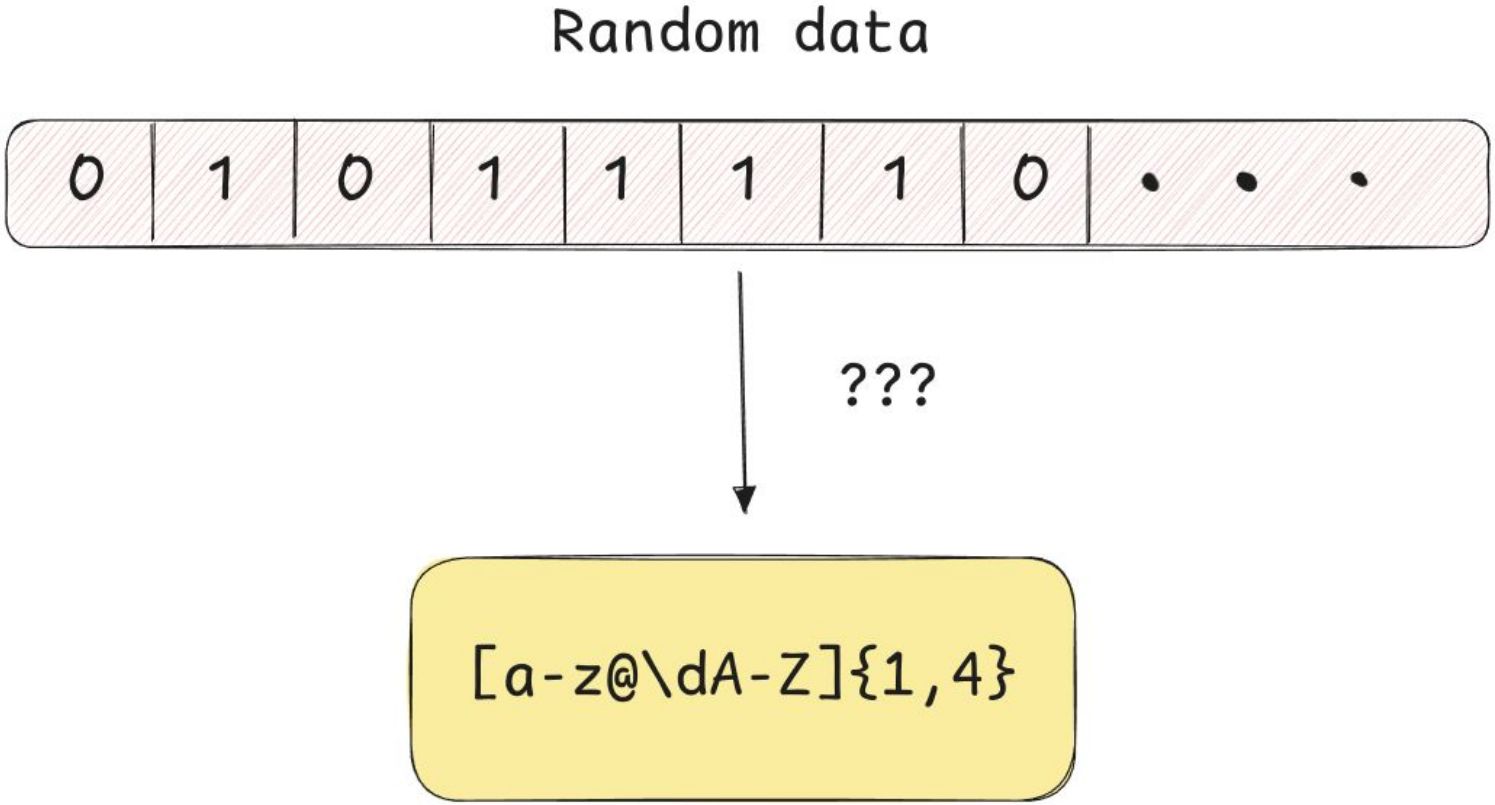
- Numbers
  - Float
  - Integers
- String
  - Known format (data-time, uuid, …)
  - Custom patterns
- Objects
- Arrays
- Boolean
- Null

How can we generate them?



And keep them in the desired form?

Example: Regular expression



Example: Regular expression

```
[a-z@\dA-Z]{1,4}
```

Example: Regular expression

```
[a-z@\dA-Z]{1,4}
```



```
a-z  ->  [0x0061, 0x007A]
@    ->  [0x0040, 0x0040]
\d   ->  [0x0030, 0x0039]
A-Z  ->  [0x0041, 0x005A]
```

Example: Regular expression

```
[a-z@\dA-Z]{1,4}
```

↓

```
a-z  ->  [0x0061, 0x007A]
@    ->  [0x0040, 0x0040]
\d   ->  [0x0030, 0x0039]
A-Z  ->  [0x0041, 0x005A]
```

↓

```
[0x0030, 0x0039]
[0x0040, 0x005A]
[0x0061, 0x007A]
```

Example: Regular expression

1. Choose N from [1,4]
2. Choose N times from IntervalSet
3. Concatenate

	10	27	26
Intervals	[0x0030, 0x0039],	[0x0040, 0x005A],	[0x0061, 0x007A]

Example: Regular expression

1. Choose N from [1,4]
2. Choose N times from IntervalSet
3. Concatenate

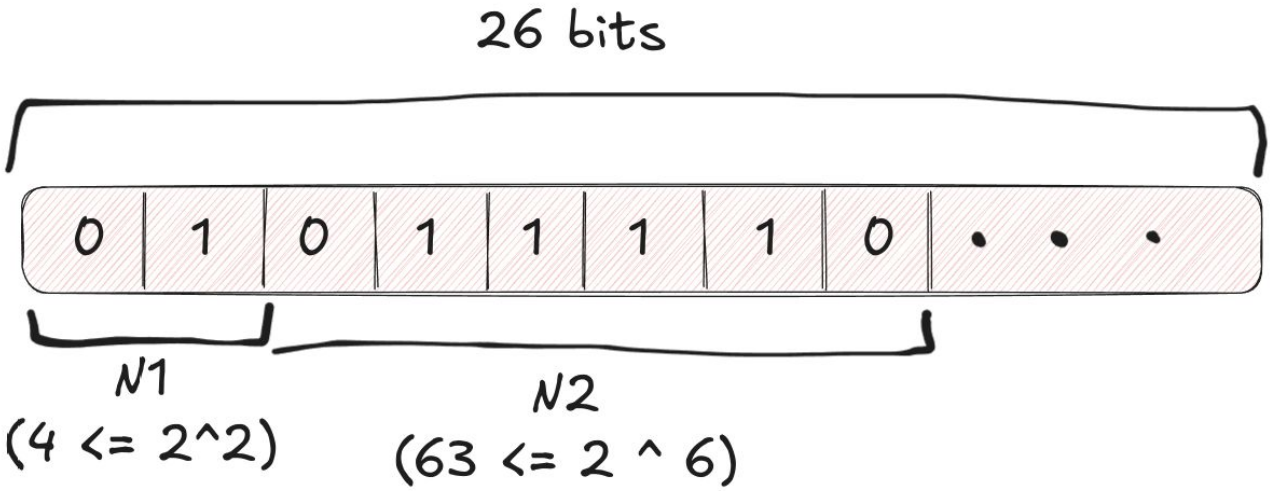
	10	27	26
Intervals	[0x0030, 0x0039],	[0x0040, 0x005A],	[0x0061, 0x007A]
Offsets	[0,	10,	37]

idx=10 -> 0x0040 (@)

IntervalSet: See [unicode-intervals]() or [Hypothesis]()

Example: Regular expression

```
[a-z@\dA-Z]{1,4}
```



Negative data

```
{
  "minimum": 10,
  "maximum": 50
}
```

```
{
  "not": {
    "minimum": 10,
    "maximum": 50
  }
}
```

```
{
  "anyOf": [
    {
      "maximum": 9
    }
    {
      "minimum": 51
    }
  ]
}
```

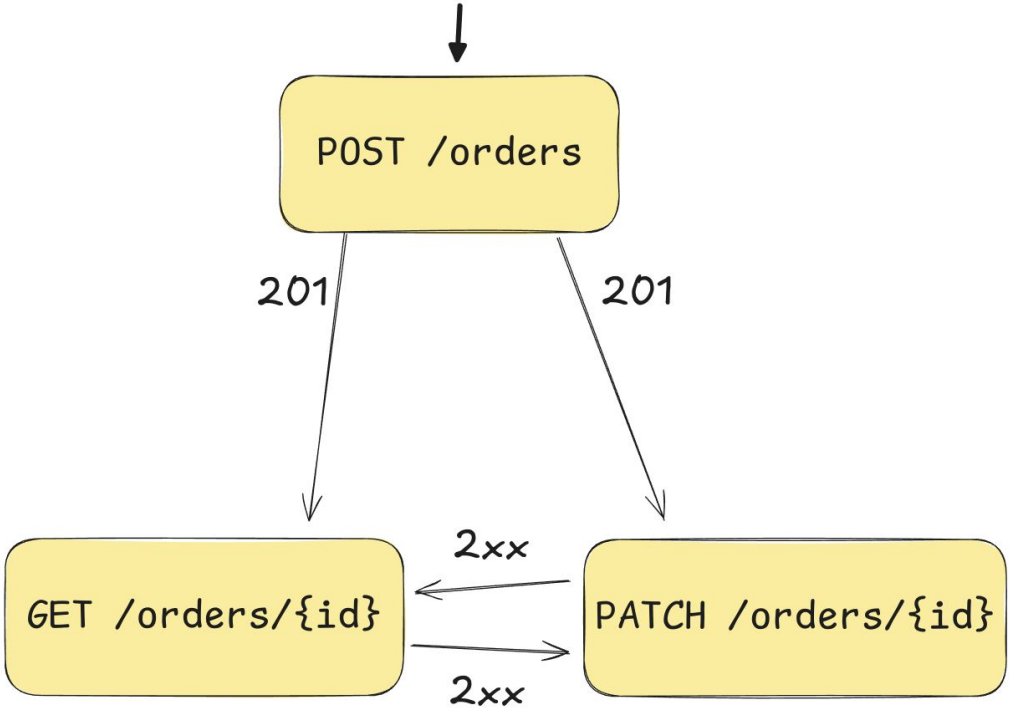
```
st.integers(max_value=9) | st.integers(min_value=51)
```

Stateful testing via Open API links

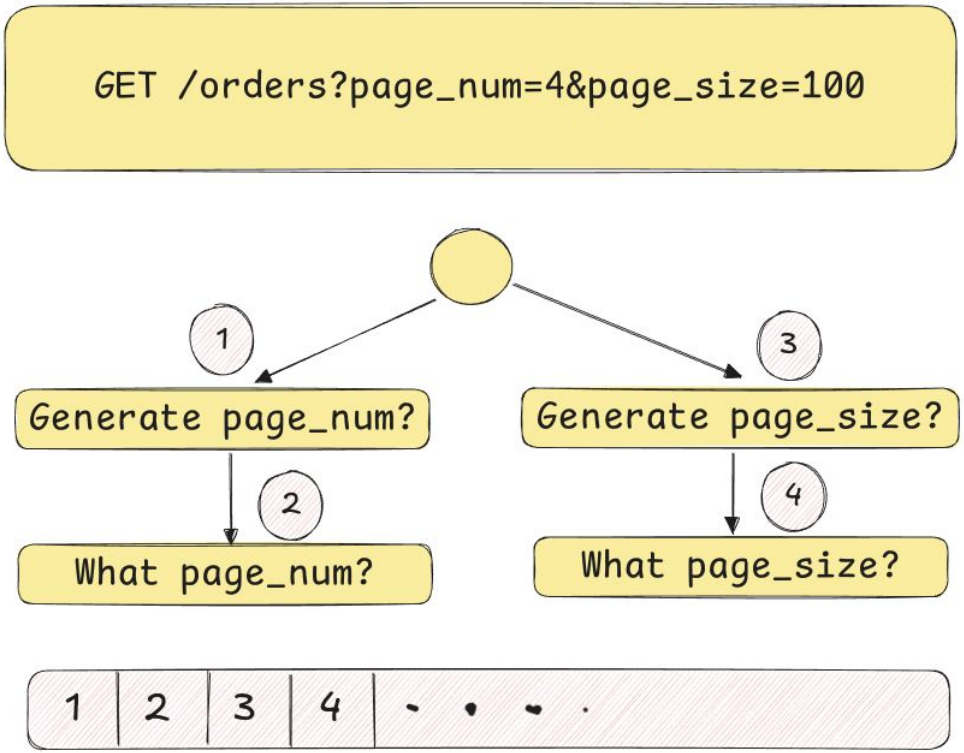
```yaml
paths:
  /orders:
    post:
      operationId: createOrder
      responses:
        '201':
          links:
            GetOrder:
              operationId: getOrder
              parameters:
                id: '$response.body#/id'
  /orders/{id}:
    get:
      operationId: getOrder
      parameters:
        - name: id
          in: path
          required: true
          schema:
            type: string
      responses:
        '2XX':
          description: Successful operation
          links:
            UpdateOrder:
              operationId: updateOrder
```

Stateful testing via Open API links

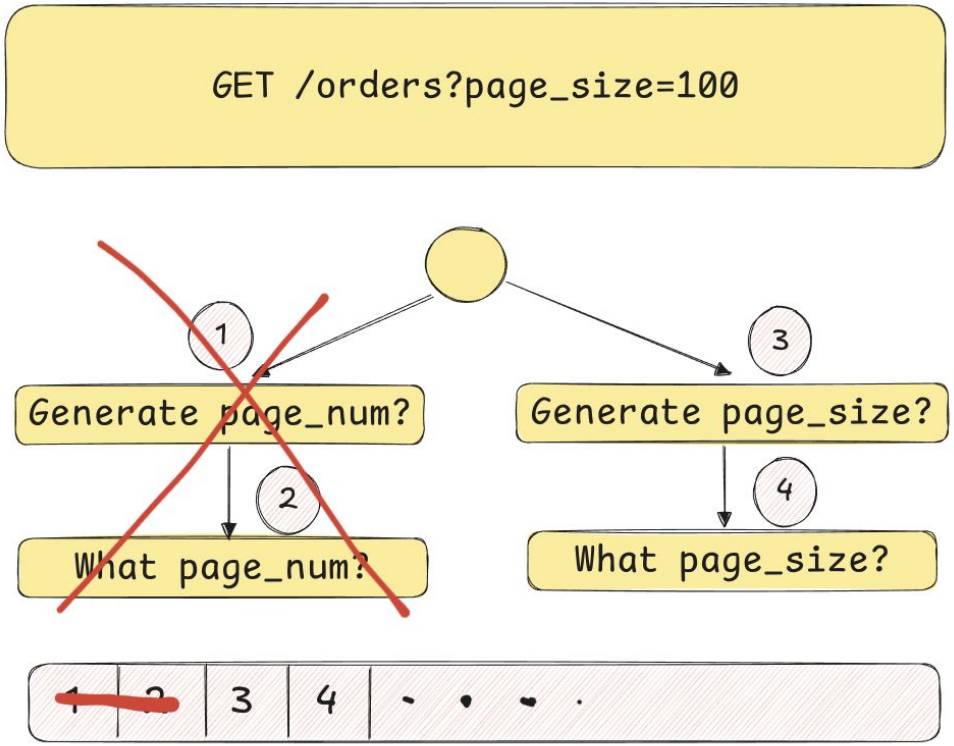
```yaml
paths:
  /orders:
    post:
      operationId: createOrder
      responses:
        '201':
          links:
            GetOrder:
              operationId: getOrder
              parameters:
                id: '$response.body#/id'
  /orders/{id}:
    get:
      operationId: getOrder
      parameters:
        - name: id
          in: path
          required: true
          schema:
            type: string
      responses:
        '2XX':
          description: Successful operation
          links:
            UpdateOrder:
              operationId: updateOrder
```



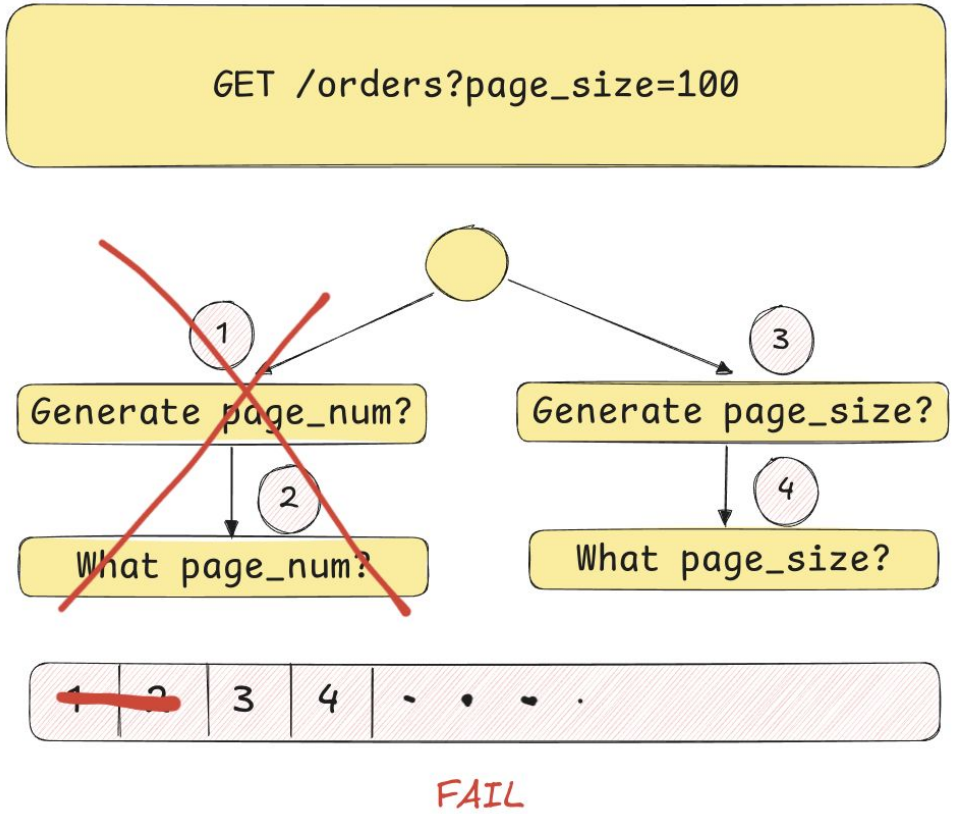
Shrinking


GET /orders?page_num=4&page_size=100
1
3
Generate page_num?
Generate page_size?
2
4
What page_num?
What page_size?
1
2
3
4

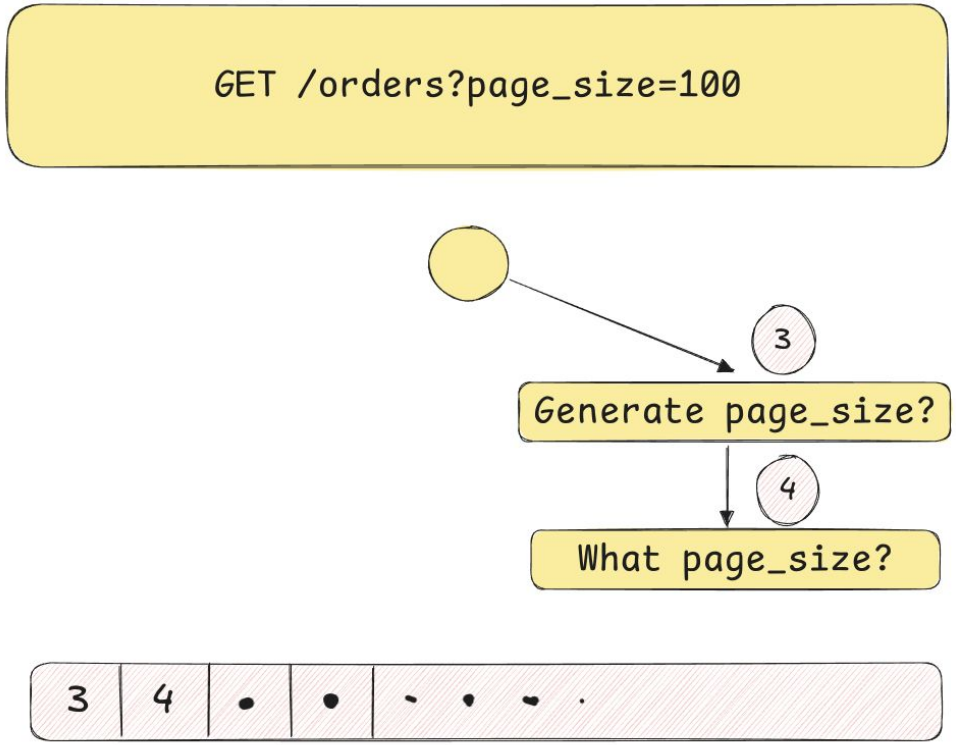
Shrinking


GET /orders?page_size=100
1
Generate page_num?
2
What page_num?
3
Generate page_size?
4
What page_size?
1
2
3
4

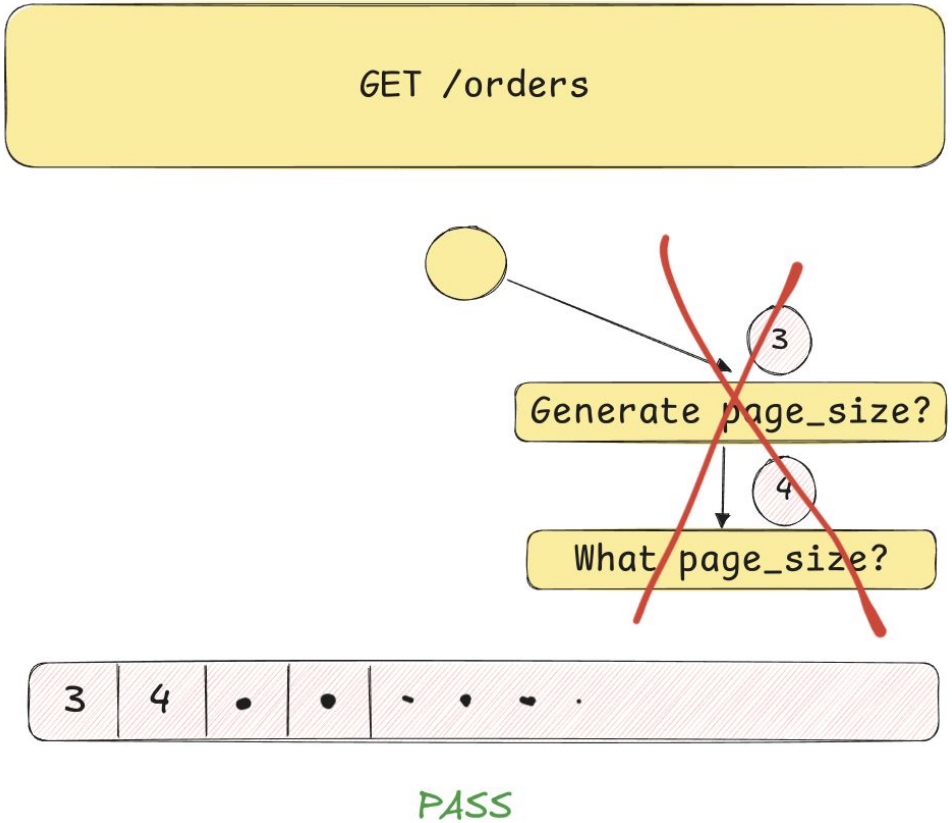
Shrinking


GET /orders?page_size=100
1
Generate page_num?
2
What page_num?
3
Generate page_size?
4
What page_size?
1
2
3
4
FAIL

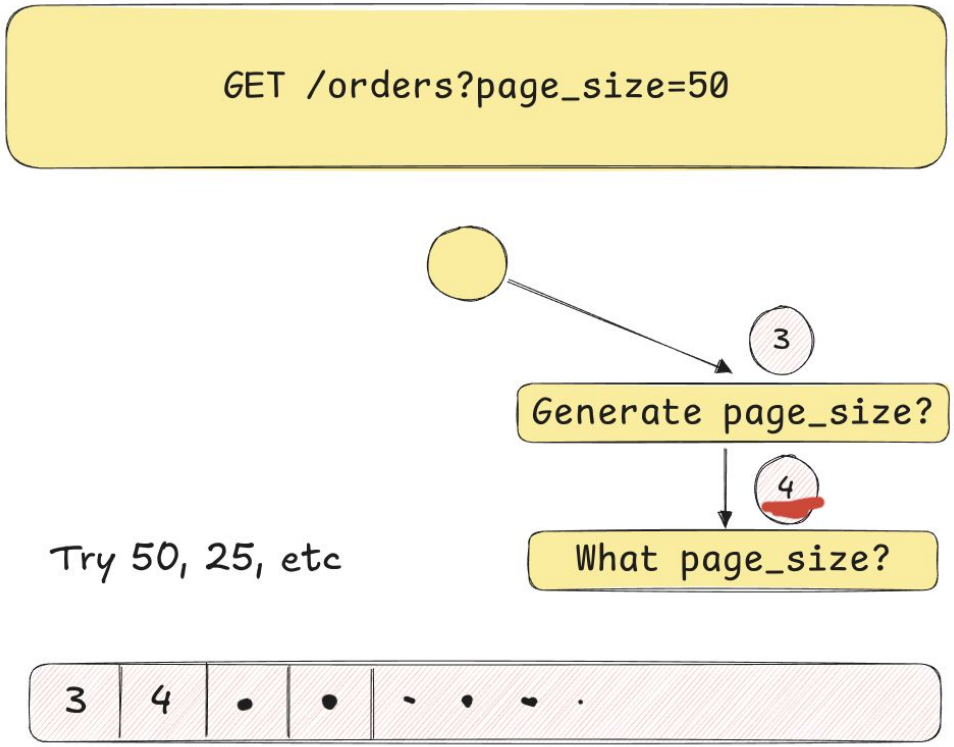
Shrinking


GET /orders?page_size=100
3
Generate page_size?
4
What page_size?
3
4

Shrinking



Shrinking


GET /orders?page_size=50
3
Generate page_size?
4
What page_size?
Try 50, 25, etc
3
4

Output example

```
> st run https://example.schemathesis.io/openapi.json
======================= Schemathesis test session starts =====================
Schema location: https://example.schemathesis.io/openapi.json
Base URL: https://example.schemathesis.io/
Specification version: Open API 3.0.2

POST /exceeding-column-size F                                          [100%]

================================== FAILURES ==================================
_________________________ POST /exceeding-column-size _________________________
1. Test Case ID: 1PzV3y

- Server error

[500] Internal Server Error:

    `{"error":"(sqlite3.IntegrityError) CHECK constraint failed: message\n[SQL:
INSERT INTO message (text) VALUES (?)]\n[parameters: ('0000000000000000',)]\n(Ba
ckground on this error at: https://sqlalche.me/e/14/gkpj)","success":false}`

Reproduce with:

    curl -X POST -H 'Content-Type: application/json' -d '{"text": "0000000000000
000"}' https://example.schemathesis.io/exceeding-column-size

================================== SUMMARY ===================================

Performed checks:
    not_a_server_error                    16 / 29 passed          FAILED

============================== 1 failed in 2.17s ==============================
```

Schema coverage report

```
167     editFolder:
168       title: Folder
169       required:
170       - id
171       - language
172       - lastModified
173       type: object
174       properties:
175         link:
176           type: array
177           items:
178             $ref: '#/components/schemas/Link'
179         name:
180           type: string
189
```

Getting Started

```python
import schemathesis

schema = schemathesis.from_uri("https://example.schemathesis.io/openapi.json")


@schema.parametrize()
def test_api(case):
    case.call_and_validate()
```

Getting Started

```yaml
api-tests:
  runs-on: ubuntu-latest
  steps:
    - uses: schemathesis/action@v1
      with:
        schema: "https://example.schemathesis.io/openapi.json"
        # OPTIONAL. Add Schemathesis.io token for pull request reports
        token: ${{ secrets.SCHEMATHESIS_TOKEN }}
```

Getting Started

schemathesis bot commented 1 minute ago

Schemathesis Report

> Of **10** total API operations, **10** failed in **1m 17s**.

Merging #1 will introduce `12` issues affecting `10` API operations:

- **Server Error**: `3`
- **Conformance**: `5`
- **Timeout**: `4`

	Operations	Details
NEW	`[500] POST /improper-unicode-encoding`	**Server Error**: HTTP 500 indicates a server-side error
NEW	`[500] POST /improper-input-type-handling`	**Server Error**: HTTP 500 indicates a server-side error
NEW	`[500] POST /exceeding-column-size`	**Server Error**: HTTP 500 indicates a server-side error
NEW	`[200] GET /incorrect-content-type`	**Conformance**: Response with `Content-Type: text/plain` is not declared in the schema
NEW	`[200] GET /missing-field`	**Conformance**: 'age' is a required property
	... and 7 more	

📢 Have feedback on this report? Share it with us

Adoption Challenges

- No schema
  - Use schema generators
  - Try to build from traffic
- “There is too much magic” - Guido van Rossum on Hypothesis

Summary

- Define properties, Schemathesis will do the rest
- Easy to use
- Spend less time debugging
- Complement, don’t replace

No Silver Bullet

```
curl -X GET 'https://example.schemathesis.io/orders?limit=986'

<!doctype html>
<html lang="en">
<head>
  <title>500 Internal Server Error</title>
</head>
<body>
  <h1>Internal Server Error</h1>
</body>
</html>
```

Thank you!

Github