

Hypothesis

September

December 17, 2019

1 Team Members

Omnia Zakaria	omniazakaria3112@yahoo.com
Reham Hamdy	rihamhamdy6@gmail.com
Zeinab Rabie	zeinabrabie37@gmail.com
Hagar Haytham	hagarhaytham597@gmail.com

2 Hypothesis

Hypothesis is a Python library for creating unit tests which are simpler to write and more powerful when run, finding edge cases in your code you wouldn't have thought to look for. Hypothesis lets you write tests which instead look like this:

1. For all data matching some specification.
2. Perform some operations on the data.
3. Assert something about the result.

It generates arbitrary data matching some input specification and checks that some guarantee holds on the output for the generated input, whatever it is. It also saves examples where problems were found to use them for testing again in the future.

3 Getting Started with Hypothesis

The main important feature of hypothesis is using strategies for generating different inputs meeting some input specification.

```
#applying the concept of property based testing.
def add(x,y):
    return x+y

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

Figure 1: Property-based testing.

In the previous example, both inputs to the tested functions were generated using hypothesis strategies. Also, there was no assertion on the output value, it only cared about the Add operation having the commutative property.

```
(-1684219848, 4)
(-31, -109)
(-79, -2128)
(30, -117)
(61, -16927)
(-47, -26550)
(-27568, -26550)
(38, 7213242882631685493)
(-16927, -27568)
(-27568, 21022)
(31245, -60)
(-26391, 2116558928)
(-7114, -20814)
(-23699, -60)
(31245, -30059)
(-18573, -27)
(-18331, 116)
(9661, -5723732689512576271)
(-30436, -10082)
(-3046, -4665635225988175244)
(107, 1441)
(23863, -16782)
(-105, -27098)
(8206182688708375197, -25)
(291, 19702)
(43, 110)
(-120, -19663)
(430784773, 22477)
(29971, 22097)
(10488, 11377)
(-19488, -18573)
(1190688556, -13556)
(-84, -17026)
(-35, -31007)
(-49, 3666)
(7894, -126)
(23550, -608530345)
(-69, 5)
(-10447, -5111)
(20101, -1684739011)
(32020, 13562213543060483)
(-12692812735858489618698311750730152814L, 298131671)
```

Figure 2: A sample of the input generated by strategies.

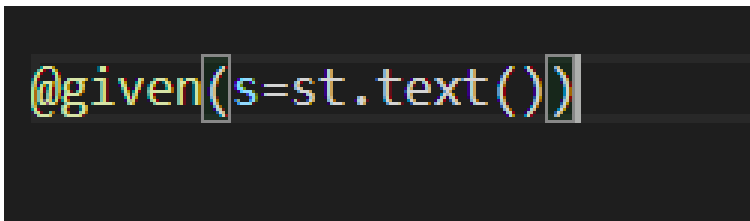
And That’s basically it to get started with hypothesis, just knowing the correct strategy and the property that’s guaranteed in the output of the tested function.

4 Details and Advanced Features

This section lists features that are less common in Hypothesis but make life much easier. Each Feature is followed by a use case.

4.1 Input Generation: @given

@given is the entry point of hypothesis to a test function. The @given decorator may be used to specify which arguments of a function should be parameterized over.



```
@given(s=st.text())
```

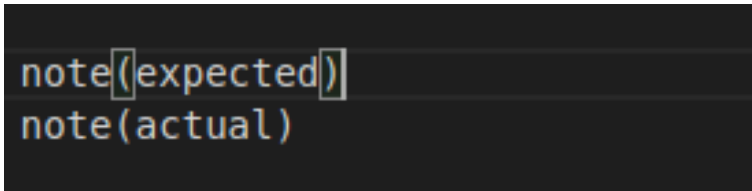
Figure 3: Using @given to generate text input.

4.2 Search Strategies

The object that is used to explore the examples given to your test function is called a Search Strategy. These are created using the functions exposed in the hypothesis.strategies module. Many of these strategies expose a variety of arguments you can use to customize generation. For example for integers you can specify min and max values of integers to be given to the tested function.

4.3 Additional test output

Hypothesis supporting printing out intermediate values during the execution of the test.



```
note(expected)
note(actual)
```

Figure 4: Using `note()` to print out intermediate values.

4.4 Making Assumptions : `assume()`

Sometimes Hypothesis doesn't give you exactly the right sort of data for the test. You can just ignore these by aborting the test early, but this runs the risk of accidentally testing a lot less than you think you are. Also it would be nice to spend less time on bad examples - if you're running 100 examples per test (the default) and it turns out 70 of those examples don't match your needs, that's a lot of wasted time.

4.5 Targeted Example Generation : `@example`

In addition to generating input, Hypothesis also supports specifying some input example that you don't want the test to miss out.

4.6 Dependent Strategies : `@composite`

One of the important features with strategies is having the ability to generate some strategy based on another. For example: Consider generating some integer then depending on it generate multiple arrays with that integer as their length.

5 Hypothesis for the Scientific Stack

- Hypothesis offers a number of strategies for NumPy testing, available in the `hypothesis[numpy]` extra. It lives in the `hypothesis.extra.numpy` package. The centerpiece is the `arrays()` strategy, which generates arrays with any dtype, shape, and contents you can specify or give a strategy for. To make this as useful as possible, strategies are provided to generate array shapes and generate all kinds of fixed-size or compound dtypes.
- Hypothesis also provides strategies for several of the core pandas data types: `pandas.Index`, `pandas.Series` and `pandas.DataFrame`.

6 Limitations

- Hypothesis mainly focuses on applying the concept of property-based testing. This makes it not able to test different types of software where such concept doesn't apply.
- The strategies are not fully customizable. If the software under test required specific inputs, filtering out generated data is not acceptable all time.

- To run hypothesis, a strategy have to be specified even if it won't be used. Consider the case when you need a specific input and its corresponding output only. You would still have to specify a strategy even if it's a dummy one and for each input in the strategy, the test will run even if it's doing the same thing.

7 Case Study

We used Hypothesis to test A collection of Numerical Analysis functions. For doing that, we used the numpy extension of hypothesis.

7.1 The Features covered

1. Search strategies : Different search strategies were used, adding tweaks to the generated data to match the input specifications of the tested functions.
2. Targeted Example : Since the tested functions were made to be custom to some problems, we needed to set some specific input example and make sure that the test passed for these inputs.
3. Making Assumptions : We made use of this one when manipulating the search strategy wasn't enough. For example: While testing a function that solves a linear system of equations, the coefficients matrix sent to the function should be a non-singular matrix.
4. Dependent Strategies : Dependent strategies were useful in the case where we needed to test functions on multiple sizes of arrays/matrices. One strategy generated the size and depending on that one, another generated the matrices/arrays.
5. Disabling health checks : Hypothesis forces a number of examples to be generated in order to guarantee the correctness of the function. in case of limiting the inputs(using assume),the test

7.2 Strategies used

We used strategies for generating arrays, integers, floats, booleans, and also functions based on regular expressions.

7.3 Functionalities tested

Integration
Solving Linear systems of Equations
Interpolation
Ordinary Differential Equations
Partial Differential Equations
Eigen Value Problem
Curve fitting
Extrapolation

7.4 Limitations

1. Having no ground truth to test the functions on, because the function were implementing approximate methods to solve numerical problems.
2. Not being able to exploit the property-based testing offered by hypothesis because the outputs of the functions didn't have a specific property.

3. Not being able to fully customize the input to meet the input specifications using strategy filtering because it resulted in test health check violations. We had to filter the input ourselves in the test functions.

8 Work Load Divison

For dividing the work load, each one of us tested some of the Software under test functionalities.

Omnia Zakaria	Solving Linear Systems, RungeODE's, InterpolationDiff, Integration
Reham Hamdy	Eigen Value Problem, PDE's, Tool Research.
Zeinab Rabie	Integration, Interpolation, Tool Research.
Hagar Haytham	Curve Fitting,Extrapolation, Tool Research, SUT.

9 References

- Hypothesis Documentation : hypothesis.readthedocs.io.