CODING NOMADS

11) Introduction To Testing Your App Lesson

Quick Recap of Python Unit Testing

7 min to complete · By Brandon Gigous

Contents

Info: This section will briefly introduce you to the fundamentals of unit testing and how to apply it to Flask.

What do you think when you see the word "testing" in terms of software testing? Of course you test your app as you build it, but how about making functions and scripting tests *intentionally*? Testing your app might not seem so fun at first, but they can help ease a lot of pain. That's because, with tests, you can catch bugs *before* they become potentially *even bigger* problems. Let's dive in to what testing looks like.

What is Unit Testing?

The material here and in the next few lessons will focus on **unit tests**. What is unit testing? Here's a succinct definition from Wikipedia:

> *Unit testing is a software testing method by which individual units of source code... are tested to determine whether they are fit for use.*

Fit for use depends on who is judging, but here are two big ones:

1. Users who may utilize a piece of software
2. *You!* And if you're on a team, your fellow developers

That's right, not only can tests make your users happier with better functioning software, but it can also prevent you and other developers from running around in circles because feature X doesn't work or because you're stuck for hours fixing a software bug. Unit tests help you test small pieces of code like functions, which help you test chunks of code like modules, which help you test bundles of code like libraries. If your code works properly, and you know it does with tests, you can sleep a little easier at night.

Pytest and Unittest

"Words words words, where's some test code?" Fair enough, you and I are programmers, so let's communicate like programmers.

There are many testing frameworks written in Python to test Python or other code. Two of the most popular ones are the `pytest` framework and Python's built-in `unittest` framework. Either one can help you do some quick tests, but pytest is a more mature and capable framework. It's also easy to get started with, and you'll more about the basics later. For now, below is what a simple test might look like:

```
# contents of test_file.py

# the function to test
def add(a, b):
    return a + b

# a test function to test add()
def test_add():
    assert add(3, 5) == 8
```

If you haven't already installed pytest, go ahead and run:

```
(env) $ pip install pytest
```

Then, assuming pytest is installed, running the `pytest` command in your terminal in the same directory as `test_file.py` will discover this file, discover the `test_add()` function, and finally will run the test function and give the test results.

```
(env) $ pytest
```

Output:

```
====================== test session starts ====================
platform linux -- Python 3.x.y, pytest-6.x.y, py-1.x.y, pluggy-0.x.y
cachedir: $PYTHON_PREFIX/.pytest_cache
rootdir: $REGENDOC_TMPDIR
collected 1 item

test_file .

================== 1 passed in 0.01 seconds ==================
```

What if the `add()` function was slightly different, like this?

```
def add(a, b):
    return a + b - 1
```

Then the output would show that the test failed:

```
# ...
test_file.py F                                                  [

======================= FAILURES =======================
_____________________________ test_add _____________________________

    def test_add():
```

```
>       assert add(3, 5) == 8
E       assert 7 == 8
E        +  where 7 = add(3, 5)

test_file.py:6: AssertionError
=============== short test summary info ===============
FAILED test_file.py::test_add - assert 7 == 8
================== 1 failed in 0.12s ==================
```

While this was an easy example, your code and certainly your Flask app code could be anything but. There could be bugs hiding within your app that you don't know about, and creating tests similar to the above can help you uncover those bugs.

And that's unit testing in a nutshell. Not so bad, huh? Your tests can do a lot more, though, so in the next lesson you'll learn how to use pytest to prepare for testing your

`

Summary: Python Flask Unit Testing

- Testing your app might not seem so fun at first, but they can help ease a lot of pain. That's because, with tests, you can catch bugs *before* they become potentially *even bigger* problems.

- Unit Tests test individual units of source code for bugs and errors