LECTURE 0b100000

TESTING AND PYTEST

MCS 260 Fall 2021

Emily Dumas

REMINDERS

- Project 3 due at 6pm
- Project 4 description coming soon
- Homework 11 open, due Tuesday at 10am

ASSERT

Python's `assert` statement is simple: It is followed by an expression, usually a boolean expression. If that expression is truthy, nothing happens. If it is falsy, the AssertionError exception is raised.

One use for `assert` is during debugging of a program under development: It stops execution if expected conditions are not met.

For example,

```
assert fact(0)==1
```

is mostly equivalent to

```
if not fact(0)==1:
    raise AssertionError()
```

So a good use for `assert` is

```
assert thing_expected_to_always_be_truthy
```

TESTING

In software development, **testing** refers to the process of checking that a program does what it is supposed to do, i.e. that it meets the desired specifications.

Most often this takes the form of code that performs testing in an automated way.

Some software development approaches rely very heavily on testing.

SIMPLEST TESTING APPROACH

Make some scripts that test your modules and programs.

E.g. module `dynam.py` might tested by a script `test_dynam.py`.

In this approach, finding, running, and interpreting results of tests is all manual.

AUTOMATED TESTING IN PYTHON

Many options for orchestrating tests.

- unittest (part of the standard library)
- pytest (our focus today)
- nose
- *many others…*

PYTEST

Pytest is *not* typically included with Python. You need to install it before you can use it. This command should do that:

```
python3 -m pip install pytest
```

Then test to make sure it's working with

```
python3 -m pytest
```

PYTEST BASICS

Put tests in files named `test_????.py`.

Put each test in a function named `test_????()`.

Each test should return normally if it succeeds, or raise an exception if it fails.

In a terminal, the command

```
python3 -m pytest
```

will find all tests (in current dir), run them, and report.

EXAMPLE

Let's make tests for the module `factorial`.

RECOMMENDED PROCESS

Make tests that encode essential functions of the program, with most of them testing a single part (function, class, etc.) in isolation. These are *unit tests*.

Every time you make a change to the program, run the tests again.

- If all tests pass: Proceed.
- If a test fails: Correct the code or correct the test.

ASKING FOR RESOURCES

Most pytest test functions accept no arguments. But if you write one with an argument called `tmpdir`, pytest will create a temporary directory before the test runs, and delete it when the test ends.

The `tmpdir` argument will be an object that contains data about the created directory. Calling `str(tmpdir)` will give the full path as a string.

This feature is very helpful when a test involves writing and reading files.

REFERENCES

- pytest official documentation
- A pytest tutorial from RealPython.com

REVISION HISTORY

- 2021-11-05 Initial publication