Introduction to Pytest

Automated testing

Exploratory testing – manually run app, see what works & doesn't

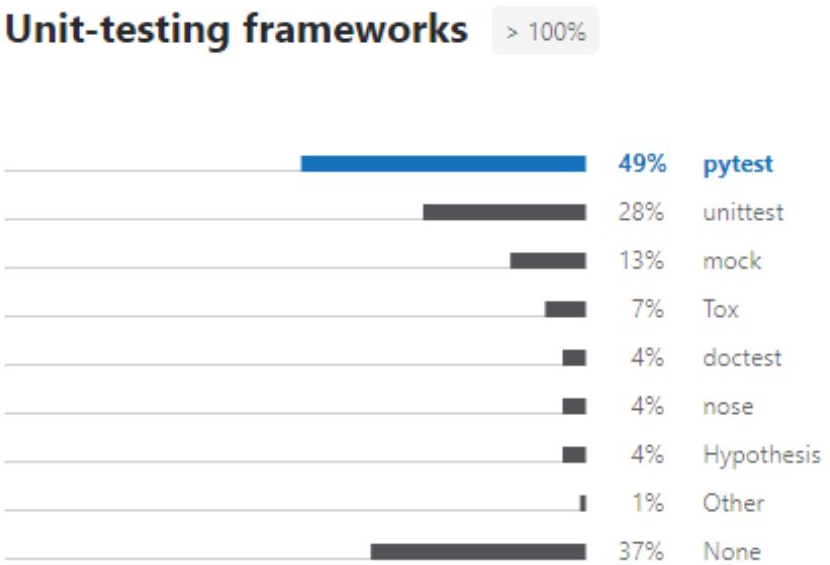
Automated testing – programmatic, planned in advance

Automated testing saves time, more consistent results

Automated testing uses testing frameworks

Python testing frameworks

JetBrains Python Developer Survey 2020



[Source: https://www.jetbrains.com/lp/python-developers-survey-2020/]

pytest: helps you write better programs

The `pytest` framework makes it easy to write small, readable tests, and can scale to support complex functional testing for applications and libraries.

[Source: https://docs.pytest.org]

pytest

Third party package, installed separately from Python

800+ external plugins to extend functionality,
integrate with other tools & frameworks

Tests can be written as Python functions or class methods

Tests can be run from command line with pytest CLI tool, or from Python code

The Python assert statement

Pytest uses built-in Python **assert** statement to pass or fail tests

Used with a Boolean conditional expression

Typically a comparison, such as an equality check

Examples

```
assert 2 + 2 == 4  # True
assert 2 + 2 == "4"  # False
```

If condition evaluates as False, **AssertionError** is raised
AssertionError causes Pytest to report test failure

Other pytest features

Auto-discovery – automatically finds tests based on test name

Modular fixtures – used to manage test resources

Run multiple tests in parallel

& more

pytest vs unittest

pytest	unittest
Third party package	Part of Python's standard library
Tests can be functions or class methods	Tests must be class methods only
Uses Python's built-in **assert** statement	Uses set of inherited class methods such as **.assertEqual()** and **.assertTrue()**
Many features such as fixtures, running tests in parallel	Some similar features but not as many as pytest, e.g. cannot run tests in parallel
pytest can run unittest tests	unittest cannot run pytest tests