pytest Markers

There are built-in and custom markers; important built-in markers to put before functions or classes:

```
@pytest.mark.skip(reason=
"Not yet implemented")

@pytest.mark.skipif(sys.plat
form == "win32", reason="does
not run on windows")

@pytest.mark.xfail(reason="
division by zero not handled
yet")
```

Handle Exceptions

Checks that a exception is raised:

```
import pytest

def square_root(value):
    if value < 0:
        raise ValueError(…)
    return value**0.5

def test_square_root():
    with pytest.raises           ↵
         (ValueError):
        square_root(-1)
```

pytest

pytest tests saved in **`test_*.py`** or **`*_test.py`** can be written as functions or grouped in classes:

```
import pytest

def test_addition():
    assert 1 + 1 == 2

class TestMathOperations:
    def test_addition(self):
        assert 1 + 1 == 2
```

pytest Fixtures

Provision of reusable default data to be used across multiple tests.

```
@pytest.fixture
def users():
    return [
        {"n": "A", "a": 30},
        {"n": "M", "a": 25}]

def test_user_exists(users):
    user = {"n": "A",
            "a": 30}
    assert user in users
```

Parametrization

Feature to run the same test with multiple data sets.

```
@pytest.mark.parametrize(
    "input_value,
    expected_output",
    [(2, 4), (-3, 9),
     (0, 0)])
def test_square(input_value,
           expected_output):
    assert fun(input_value)↵
    == expected_output
```

How to Run Tests

Running **`pytest`** executes all tests in your folder or select specific ones:

```
pytest test_example.py
pytest tests/
pytest -k "keyword"
pytest test_abc.py -k "key"
pytest test_abc.py:: test_ab
pytest test_abc.py::         ↵
       TestClass
pytest test_abc.py::         ↵
       TestClass::test_add
```