Pytest Mocking Handbook: Techniques and Examples.

Here's a quick reference guide on how to mock various objects and systems in Pytest using the pytest-mock library.

All the below code can be downloaded from here.

Mock A Constant Or Variable

Example: Mocking the value of **PI** in a module that calculates the area of a circle.

```
from mock_examples import area

def test_area_of_circle_with_mock(mocker):
    """
    Function to test area of circle with mocked PI value
    """
    mocker.patch("mock_examples.area.PI", 3.0) # Mock the
        value of PI, defined in `mock_examples/area.py`
    assert area.area_of_circle(5) == 75.0
```

Mock A Function

Example: Mocking the **area_of_circle** function.

```
from mock_examples import area

def test_area_of_circle_mocked_function(mocker):
    """
    Function to test area of circle with mocked function
    """
    mocker.patch("mock_examples.area.area_of_circle",
        return_value=100) # Mock the function
        `area_of_circle` defined in `mock_examples/area.py`
    assert area.area_of_circle(5) == 100
```

Mock A Class

Example: Mocking the **Person** class.

```python
import pytest
from mock_examples import person


@pytest.fixture
def mock_person_class(mocker):
    """
    Fixture to mock the entire Person class.
    """
    return mocker.patch(
        "mock_examples.person.Person", autospec=True
    )  # autospec=True ensures that the mock has the same
        interface as the class being mocked


def test_person_class_is_mocked(mock_person_class):
    """
    Test to verify that the Person class is mocked.
    """
    mock_person_instance = (
        mock_person_class.return_value
    )  # `mock_person_class.return_value` is the mock object
        that simulates any instance of 'Person'
    mock_person_instance.get_person_json.return_value = (
        {  # You set up what the `get_person_json` method
        should return when called
            "name": "FAKE_NAME",
            "age": "FAKE_AGE",
            "address": "FAKE_ADDRESS",
        }
    )

    # When you create a 'Person', you're using the mock, not
        the real Person class
    eric = person.Person("Eric", 25, "123 Farmville Rd")
    result = eric.get_person_json()  # # This calls the
        mocked method

    # Assert that the mocked method returns what you expect
    assert result == {
        "name": "FAKE_NAME",
        "age": "FAKE_AGE",
        "address": "FAKE_ADDRESS",
    }

    # Assert that the Person was instantiated correctly in
        the mock
```

```
    mock_person_class.assert_called_once_with("Eric", 25, "123
        Farmville Rd")

    # Assert that the `get_person_json` method was called on
        the mock instance
    mock_person_instance.get_person_json.assert_called_once()
```

Mock An External Rest API Call

Example: Mocking an external API call using **requests.get**. The **get_weather** function makes an API call to get the weather data for a city.

```
from mock_examples.api import get_weather


def test_get_weather_mocked(mocker):
    """
    Function to test get weather with mocked response
    """
    mock_data = {
        "temperature": "+7 °C",
        "wind": "13 km/h",
        "description": "Partly cloudy",
        "forecast": [
            {"day": "1", "temperature": "+10 °C", "wind":
        "13 km/h"},
            {"day": "2", "temperature": "+6 °C", "wind": "26
        km/h"},
            {"day": "3", "temperature": "+15 °C", "wind":
        "21 km/h"},
        ],
    }

    # Create a mock response object with a .json() method
        that returns the mock data
    mock_response = mocker.MagicMock()
    mock_response.json.return_value = mock_data
    mock_response.status_code = 200
        # Mocking status code as well

    # Patch 'requests.get' to return the mock response
    mocker.patch("requests.get", return_value=mock_response)

    # Call the function
    result = get_weather(city="London")

    # Assertions to check if the returned data is as expected
```

```
    assert result == mock_data
    assert mock_response.status_code == 200
    assert isinstance(result, dict)
    assert result["temperature"] == "+7 °C"
    assert result["wind"] == "13 km/h"
```

Mock System Variables / Environment Variables

Example: Mocking the os.getenv function to check if a feature is enabled. The is_feature_enabled function checks if a feature is enabled by looking up an environment variable.

```
import os


def is_feature_enabled():
    """
    Function that checks if a feature is enabled by looking
        up an environment variable.
    """
    return os.getenv("FEATURE_ENABLE", "false").lower() in
        ("true", "1", "yes")


def test_feature_enabled(mocker):
    """
    Test to ensure the function behaves correctly when the
        feature is enabled.
    """
    mocker.patch.dict(os.environ, {"FEATURE_ENABLE": "true"})
    assert is_feature_enabled()


def test_feature_disabled(mocker):
    """
    Test to ensure the function behaves correctly when the
        feature is disabled.
    """
    mocker.patch.dict(os.environ, {"FEATURE_ENABLE":
        "false"})
    assert not is_feature_enabled()
```

Mock A Database Connection (Postgres)

Example: Mocking a database connection using psycopg2. The fetch_all_users function fetches all users from the database using psycopg2 to establish a connection.

```
from mock_examples.pg_query import fetch_all_users


def test_fetch_all_users_mocked(mocker):
    # Mock psycopg2.connect to return a mock connection
        object
    mock_conn = mocker.MagicMock()
    mock_cur = mocker.MagicMock()

    # Configure the cursor to return mock data
    mock_cur.fetchall.return_value = [("Jane Doe", 28),
        ("John Smith", 30)]

    # Setting up the connection and cursor
    mock_conn.cursor.return_value = mock_cur
    mocker.patch("psycopg2.connect", return_value=mock_conn)

    # Call the function
    result = fetch_all_users()

    # Assertions to verify behavior and results
    assert result == [("Jane Doe", 28), ("John Smith", 30)]
    mock_conn.cursor.assert_called_once()
    mock_cur.execute.assert_called_once_with("SELECT * FROM
        users")
    mock_cur.fetchall.assert_called_once()
    mock_cur.close.assert_called_once()
    mock_conn.close.assert_called_once()
```

I hope you found this handbook helpful. If you have any questions, please feel free to reach out. Good luck!