Python

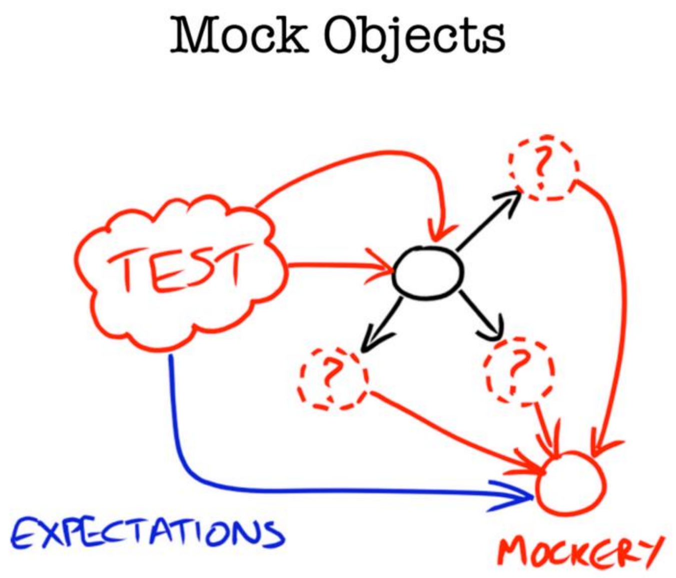
Full stack Skills Bootcamp

What are Mock Objects?

- Definition

  - Mock objects are simulated objects that mimic the behavior of real objects in controlled ways.

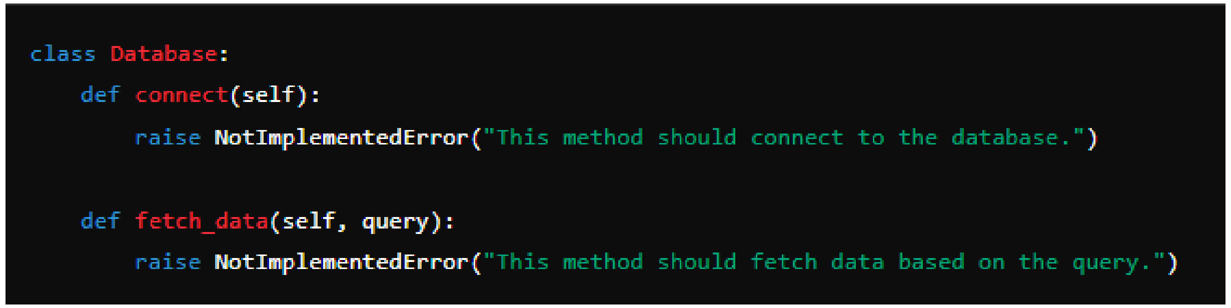
  Example: Replacing a database object with a mock to simulate fetching data, without actually connecting to a database.



Example Class: External Dependency

Class Database: Simulates a database that has two methods.

- connect(): For connecting to the database (raises an error because it's not implemented).
- fetch_data(query): For fetching data from the database (also raises an error)..



```python
class Database:
    def connect(self):
        raise NotImplementedError("This method should connect to the database.")

    def fetch_data(self, query):
        raise NotImplementedError("This method should fetch data based on the query.")
```

This class represents an external dependency that we don’t want to actually call in our tests, so we'll replace it with a mock object.

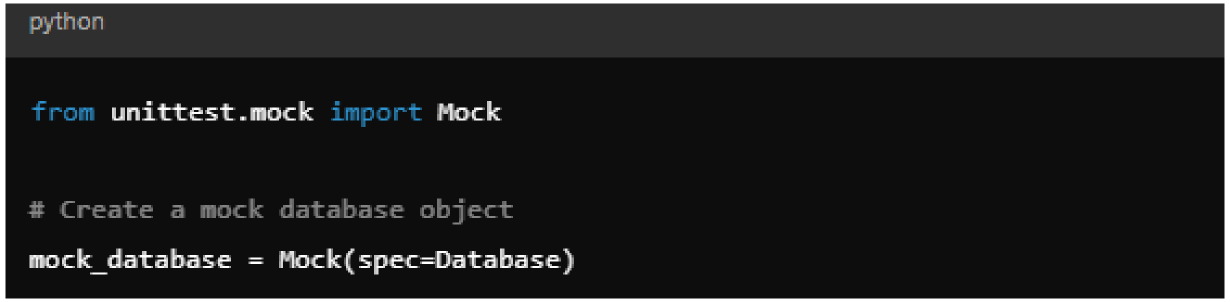
Using Mock Objects in Tests

Creating Mock Objects:

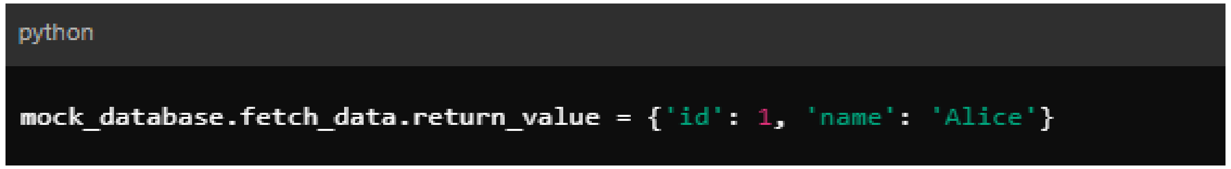
- Use Python's unittest.mock.Mock to create a mock object for Database.

Setting Mock Behaviour:

- Define what the mock should return when a method is called.



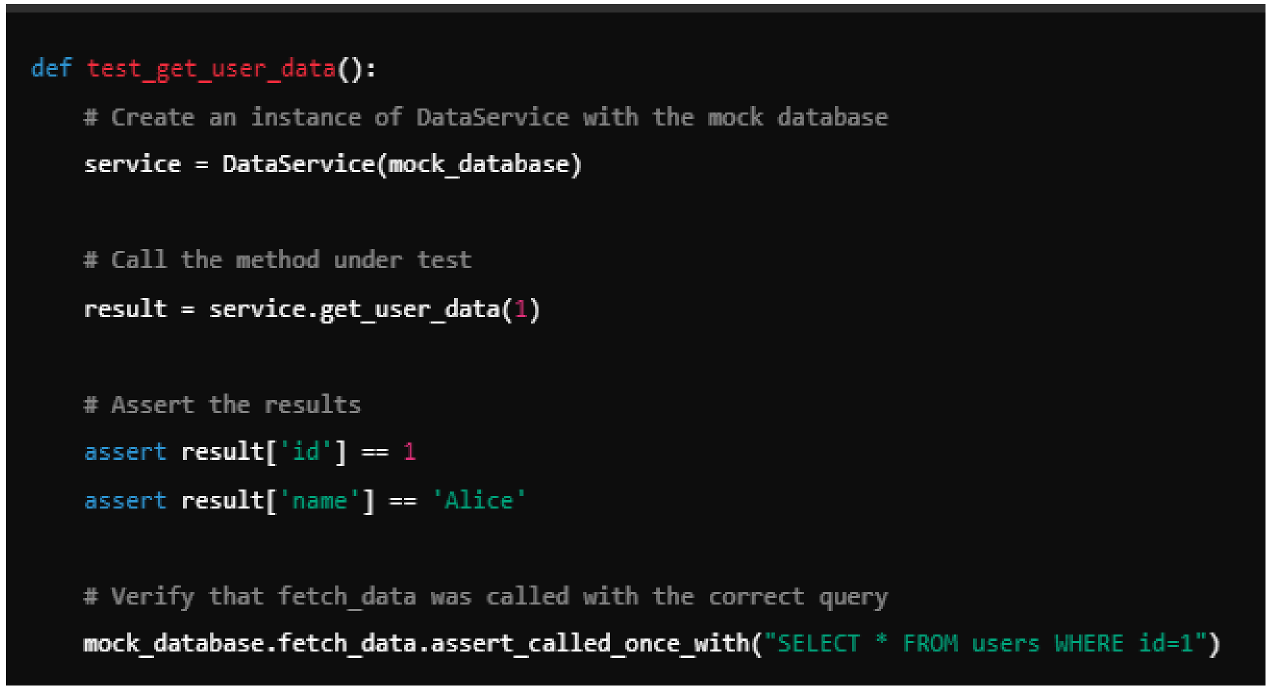
```python
from unittest.mock import Mock

# Create a mock database object
mock_database = Mock(spec=Database)
```



```python
mock_database.fetch_data.return_value = {'id': 1, 'name': 'Alice'}
```

Explanation: Here, we're telling the mock that whenever fetch_data() is called, it should return a predefined dictionary.

Writing a Test with Mock Objects



```python
def test_get_user_data():
    # Create an instance of DataService with the mock database
    service = DataService(mock_database)

    # Call the method under test
    result = service.get_user_data(1)

    # Assert the results
    assert result['id'] == 1
    assert result['name'] == 'Alice'

    # Verify that fetch_data was called with the correct query
    mock_database.fetch_data.assert_called_once_with("SELECT * FROM users WHERE id=1")
```

- We're testing the get_user_data() method of the DataService class by injecting a mock database instead of a real one.
- We also verify that the mock's fetch_data() method was called with the expected query.

Recap

Key Takeaways:

- Mocking allows you to simulate real objects in your tests, providing control and flexibility.
- Mock objects help you avoid external dependencies, keeping your tests fast and isolated.
- Use unittest.mock to easily create and customize mock objects in Python.
- Next Steps: Now that we understand how to mock objects, we'll see how to use these mock objects in various testing scenarios to handle external interactions effectively.