Python

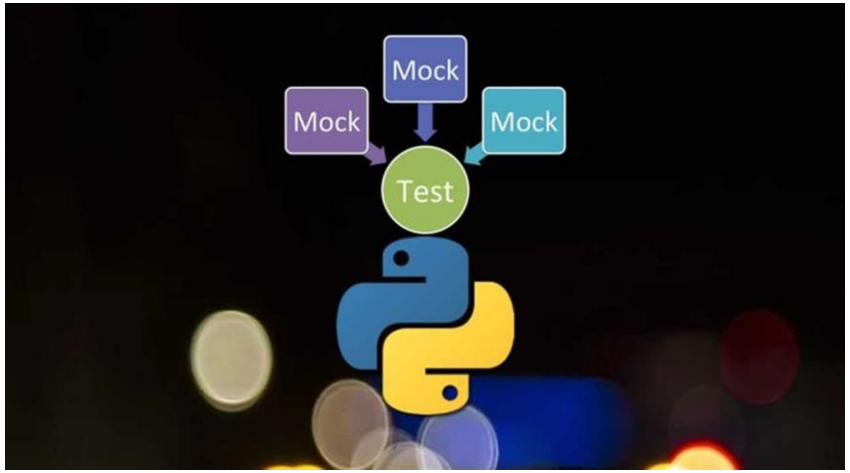
Full stack Skills Bootcamp

What is Mocking?

- Definition

  - Mocking is a technique used in testing to replace real objects with mock objects that simulate the behaviour of the real objects.

  Mocking an API call to return a predefined response instead of making an actual network request.



Why Use Mocking?

- Benefits:

  - Isolation: Tests can focus solely on the unit of code, without interference from other components or services.
  - Speed: Mocking can significantly speed up tests by eliminating network calls or database access.
  - Control: Allows you to simulate different scenarios and edge cases without needing the actual external services to behave that way.

  When to Mock: Use mocking when your code interacts with external systems (like databases, APIs, or file systems).

Mocking in Python

- Python Libraries for Mocking:
  - unittest.mock: A built-in library in Python for creating mock objects..
- Key Features:
  - Create mock objects with Mock().
  - Configure return values and side effects.
  - Assert calls and check how they were used.

Example: Using unittest.mock to replace a real function call with a mock that returns a specified value.

Example

Mock Objects:

- A mock object can mimic the behaviour of real objects and can be customized to return specific values.

```python
from unittest.mock import Mock

# Create a mock object
mock_api = Mock()

# Define what the mock should return when called
mock_api.get_data.return_value = {'key': 'value'}

# Call the mock
response = mock_api.get_data()

# Check the response
assert response == {'key': 'value'}
```

mock_api simulates an API object, allowing you to test code that interacts with it without actually making API calls.

Recap

Key Takeaways:

- Mocking allows you to isolate tests, control test environments, and speed up testing.
- Use unittest.mock to create and manage mock objects in Python.
- Understand the difference between mocking and stubbing to choose the right approach for your tests..

In the upcoming slides, we'll dive deeper into creating mock objects and using them effectively in our tests.