1. Why is high test coverage valuable for development? 1 / 1 point

- (●) Increases likelihood that the code works as expected
- ( ) Produces results that are cleaner and easier to read
- ( ) Reduces time that each test takes to complete
- ( ) Identifies code bloat that developers can remove

Correct

Correct. High test coverage gives you confidence that a large amount of code executed during the tests. In turn, the more lines of code executed through tests, the more confident you can be that the code works as expected.

2. Fake data is more practical to use than real data in which of the following testing situations? 1 / 1 point

- ( ) When testing requires only a small set of test data
- ( ) When testing requires patching one or more functions
- (●) When testing requires a large amount of test data
- ( ) When testing requires multiple levels of setups and teardowns

Correct

Correct. You can use sample data in testing. However, creating and maintaining sample data becomes tedious if you need hundreds, even thousands, of records of test data. Generating fake data is more practical in situations like this.

3. When generating fakes in Python, what is one purpose for using the Meta class inside the factory? 1 / 1 point

- ( ) Selects a random choice from a given fake iterable
- ( ) Generates a diverse sample of fake data for use in test assertions
- ( ) Provides information describing the purpose and number of fakes
- (●) Determines the attributes to use for generating data

Correct

Correct. One purpose of the using the Meta class is to indicate the model class for which the factory must create fakes.

4. How do developers test fakes generated by a factory? 1 / 1 point

- ( ) Developers set the data for all variables manually on use.
- ( ) Developers add additional test fixtures.
- ( ) Developers do not call real methods.
- (●) Developers test fakes just like real objects.

Correct

Correct. One purpose of the using the Meta class is to determine the attributes to use for generating data.

5. What is mocking? 1 / 1 point

- (●) A process for creating fake objects that mimic the behavior of real objects
- ( ) A process for creating a fake data set that acts like sample data
- ( ) A process for creating a fake test case that stress tests a test runner
- ( ) A process for creating duplicate data for testing purposes

Correct
Correct. Mocking is a process for creating objects that mimic the behavior of real objects.

6. Developers should use mocking for which of the following purposes? 1 / 1 point

- (●) Isolate their tests from external systems
- ( ) Include internal systems in their testing
- ( ) Avoid calls of untested code
- ( ) Ensure that database transactions actually work

Correct
Correct. Developers should use mocking to isolate their tests from external services that are not under test.

7. When patching, developers change the behavior of which of the following testing items? 1 / 1 point

- ( ) Entire object
- (●) Function call
- ( ) Fake data
- ( ) Testing factory

Correct
Correct. With patching, you can change the behavior of a function call.

8. For which of the following situations is it useful to patch return values with data? 1 point

- ○ Testing objects with method calls
- ◉ Testing objects that call other objects
- ○ Testing mocks
- ○ Testing error handlers

Incorrect

Incorrect. Refer to the Mocking with Patch video.

9. Mock objects are useful in testing for which of the following reasons? 1 / 1 point

- ○ Mock objects can create test fixtures.
- ○ Mock objects can improve code refactoring.
- ◉ Mock objects can behave like real objects.
- ○ Mock objects can generate unlimited test data.

Correct

Correct. Developers can control how a mock object behaves and what it returns. They can set multiple attributes on it and make it behave just like the real object that it's replacing.

10. In Python, how do you make a Mock object mimic a specific real object? 1 / 1 point

- ○ Subclass the real object.
- ○ Specify the real object as a return value.
- ◉ Specify the name of the real object in the spec parameter.
- ○ A mock is generic and cannot mimic a specific real object.

Correct

Correct. To make a Mock or MagicMock object mimic a specific real object, specify that real object's name in the spec parameter.