**Even More Tests**

**Introduction**

In this lesson we are going to build a couple more tests. If we check our code coverage with forge coverage, the terminal will show that we are only at around 53% coverage for the Raffle.sol contract. Code coverage refers to the percentage of lines of code that have been tested.

💡 **TIP**  
Achieving 100% coverage isn't always required, but it is a recommended target.

**checkUpkeep tests**

To improve our coverage, we need to write additional tests. For example we can address the checkUpkeep function, to ensure it really executes as intended under various circumstances.

1. Let’s start by ensuring that checkUpkeep returns false when there is no balance. We’ll do this by setting up our test environment similarly to previous tests but without entering the raffle. Here’s the code:

function testCheckUpkeepReturnsFalseIfItHasNoBalance() public {

// Arrange

vm.warp(block.timestamp + automationUpdateInterval + 1);

vm.roll(block.number + 1);

// Act

(bool upkeepNeeded,) = raffle.checkUpkeep("");

// Assert

assert(!upkeepNeeded);

}

1. Next, we want to assert that checkUpkeep returns false when the raffle is in a *not open* state. To do this, we can use a setup similar to our previous test:

function testCheckUpkeepReturnsFalseIfRaffleIsntOpen() public {

// Arrange

vm.prank(PLAYER);

raffle.enterRaffle{value: raffleEntranceFee}();

vm.warp(block.timestamp + automationUpdateInterval + 1);

vm.roll(block.number + 1);

raffle.performUpkeep("");

Raffle.RaffleState raffleState = raffle.getRaffleState();

// Act

(bool upkeepNeeded,) = raffle.checkUpkeep("");

// Assert

assert(raffleState == Raffle.RaffleState.CALCULATING);

assert(upkeepNeeded == false);

}

**Conclusion**

By writing these additional tests, we enhance our test coverage rate, improve the reliability of our Raffle.sol contract, and check that checkUpkeep behaves correctly under various conditions.