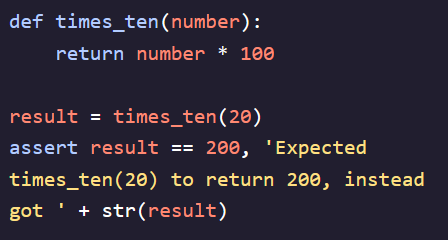
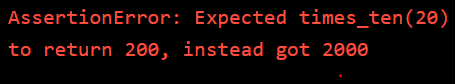
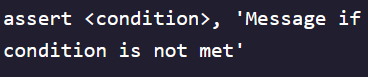
**Testing:**

- Crucial to creating quality software  
- Goal of testing isn’t just to find bugs but to find them quickly  
- Tests can be divided into two categories:  
- ***Manual Testing -***  A physical person interacts with software in the same way a user would  
­ ***Automated Testing*** – Tests that are performed with code, faster and less prone to human error

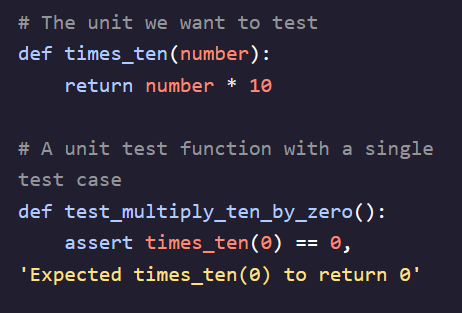
**Assert Statement:**

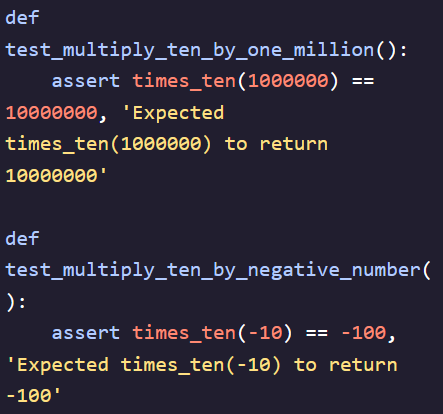
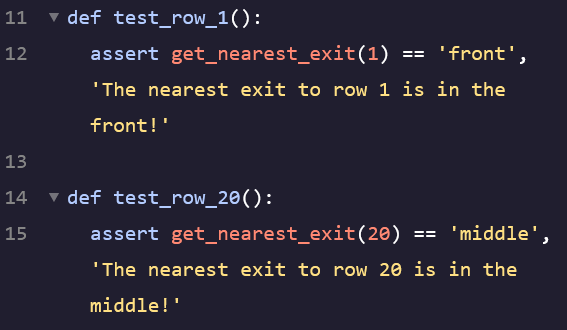
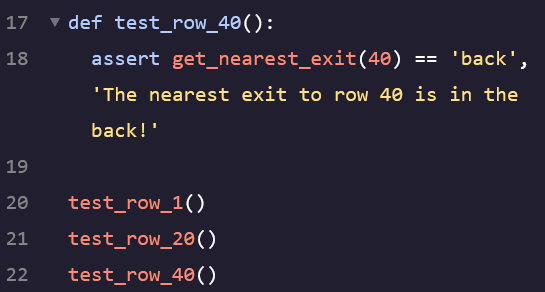
- Performing manual checks over large amounts of code is tedious and we should write automated testing to do it for us  
- An ***assert***  statement can be used to test that a condition is met. If a condition evaluates to *False* an ***AssertionError*** is raised with an optional error message  






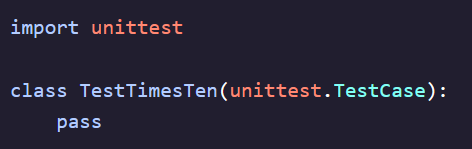
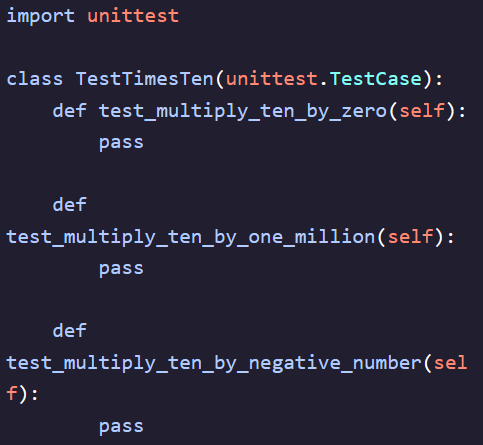
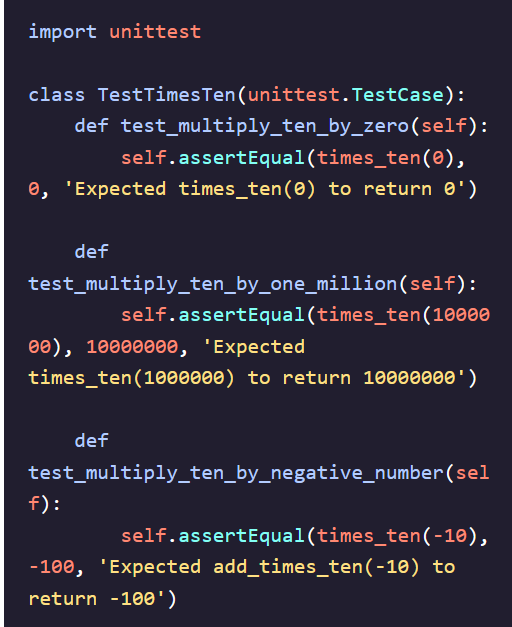
**Unit Testing:**

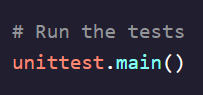
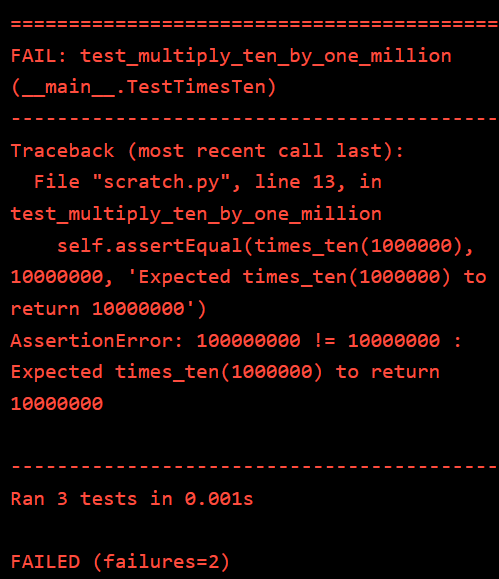
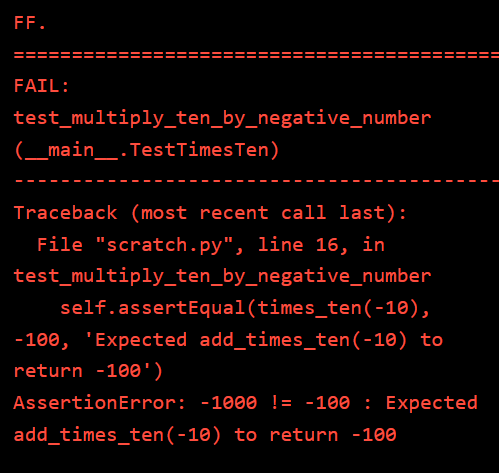
- Testing begins at the smallest unit of a program (function, loop, or variable)  
- A unit test validates a single behavior and will make sure all of the units of a program are functioning properly   
- ***Test Case*** – Validates that a specific set of inputs produces the expected output for the unit we are testing  


- Best practice is to create test cases for specific edge case inputs as well as reasonable ones (large number, negative number, and zero)  
  

**Unittest Framework:**

- Testing one by one is burdensome because:   
1). We have to call each function specifically when a new test is created  
2). No way of grouping tests together, necessary when number of tests increases  
3). If one test fails, *AssertionError* prevents any remaining tests from running  
  
- Python provides the ***unittest*** framework to solve these problem and provide other tools for writing unit tests  
- Need to import the module into code when using ***import unittest***- Provides us with a ***test runner*** – A component that collects and executes tests and then provides results to the user. Also provides other tools for test grouping, setup, teardown, and skipping  
**Creating a Unittest:**

- First, we need to create a class which inherits from *unittest.TestCase* to server as main storage of all unit testing functions  
- Then, we need to create functions within the class. These are required to begin with the word *test*- Lastly, we need to use *assertEqual* statements to delineate tests  


- Run code by calling *unittest.main()* and it will detect any tests in existing module, run them, and provide results  
 

**Assert Methods I: Equality and Membership:**

-