

# Code Testing: Functions, Assertions, and Unit Testing

## 1. Functions

Functions play a central role in organizing and testing code. By breaking complex problems into smaller, reusable blocks, they simplify debugging and testing.

```
1 def average(L):
2     if not L:
3         return None
4
5     return sum(L) / len(L)
6
7 if __name__ == "__main__":
8     L = [1, 2, 3, 4, 5]
9     expected_avg = 3.0
10    result = average(L)
11
12    if expected_avg == result:
13        print("Test passed")
14
15    else:
16        print("Test failed")
17        print(f"Expected: {expected_avg}")
18        print(f"Got: {result}")
```

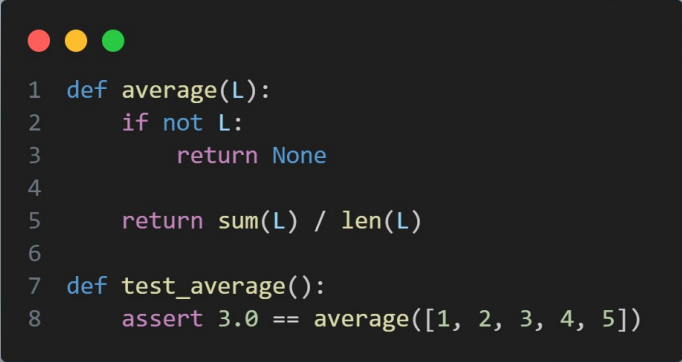
## 2. Assertions

Assertions are used to verify that a piece of code behaves as expected. An assertion is a statement that checks a condition, and if the condition evaluates to `False`, the program halts with an error. This helps catch bugs early.

```
1 # Function to calculate the average of a list
2 def average(L):
3     if not L:
4         return None
5
6     return sum(L) / len(L)
7
8 if __name__ == "__main__":
9     # Test case
10    L = [1, 2, 3, 4, 5]
11    expected_avg = 3.0
12
13    # Assert to check if the test passed or failed
14    assert expected_avg == average(L), "Test failed"
15    print("Test passed")
```

### 3. Unit Testing with Pytest

Pytest is a popular Python testing framework that makes it easy to write simple and scalable test cases. Its minimal syntax and powerful features make it a preferred choice for many developers.

A code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains Python code for a unit test using Pytest. The code defines a function 'average' and a test function 'test\_average'.

```
1 def average(L):  
2     if not L:  
3         return None  
4  
5     return sum(L) / len(L)  
6  
7 def test_average():  
8     assert 3.0 == average([1, 2, 3, 4, 5])
```

### 4. Table-Driven Testing

Table-driven testing is a method that uses tables (often as lists or arrays in code) to store test cases and their expected outcomes. This approach reduces repetition and ensures consistency, making it easier to add or modify test cases.