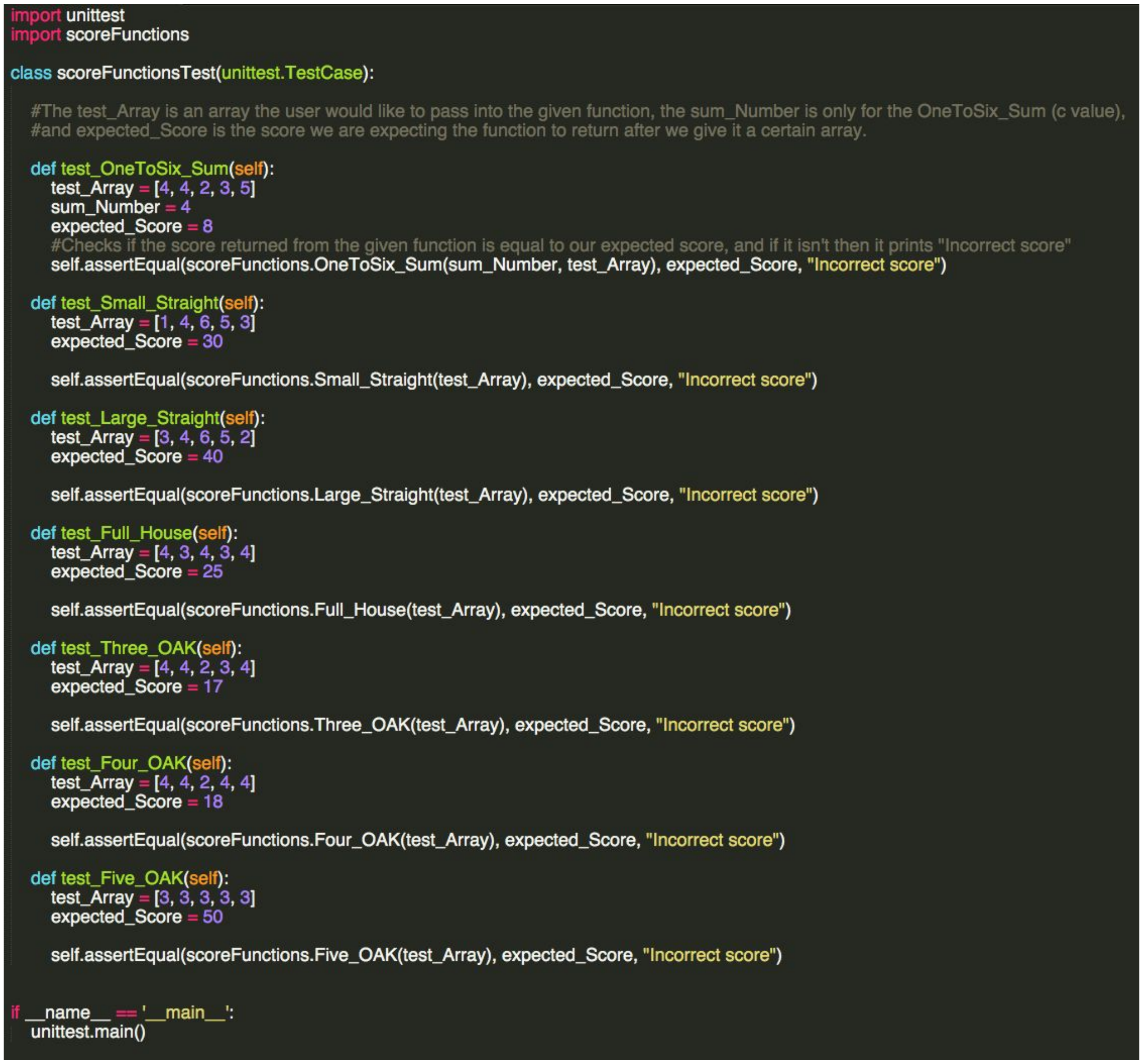
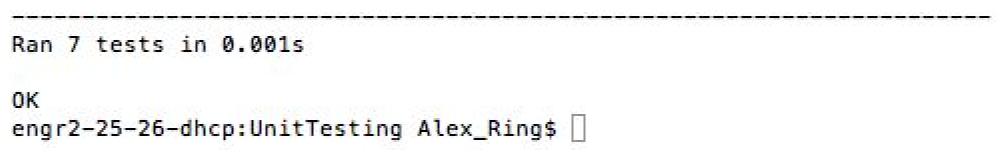
Who: Robert Allen, Alex Ring, and David Zhuzhunashvili
Title: Yahtzee
Vision: To create a functional, easy to use, Yahtzee emulator that can be played by one to four people.
Automated Tests: We wrote test cases using Python's "unittest," similarly to Lab 8. We wrote test cases for all the score calculation functions, which simply calculate the total score of any 5 dice given using a specific scoring method (five of a kind, full house, small straight…). Each one of the tests has a "test_Array" (of size 5) which can be changed around to be any possible combinations of 5 dice, and there is an "expected_Score" value which stores the value that the user expects to be returned from a certain score calculation function. The one to six sum function also has a variable called "sum_Number" which defines the dice value that the user wants to pick in order to find the one to six sum of the dice. Following these variables, each test case compares if the expected score is equal to the score returned from the given function, and if it is equal it passes the test. This is what the code looks like:



```python
import unittest
import scoreFunctions

class scoreFunctionsTest(unittest.TestCase):

    #The test_Array is an array the user would like to pass into the given function, the sum_Number is only for the OneToSix_Sum (c value),
    #and expected_Score is the score we are expecting the function to return after we give it a certain array.

    def test_OneToSix_Sum(self):
        test_Array = [4, 4, 2, 3, 5]
        sum_Number = 4
        expected_Score = 8
        #Checks if the score returned from the given function is equal to our expected score, and if it isn't then it prints "Incorrect score"
        self.assertEqual(scoreFunctions.OneToSix_Sum(sum_Number, test_Array), expected_Score, "Incorrect score")

    def test_Small_Straight(self):
        test_Array = [1, 4, 6, 5, 3]
        expected_Score = 30

        self.assertEqual(scoreFunctions.Small_Straight(test_Array), expected_Score, "Incorrect score")

    def test_Large_Straight(self):
        test_Array = [3, 4, 6, 5, 2]
        expected_Score = 40

        self.assertEqual(scoreFunctions.Large_Straight(test_Array), expected_Score, "Incorrect score")

    def test_Full_House(self):
        test_Array = [4, 3, 4, 3, 4]
        expected_Score = 25

        self.assertEqual(scoreFunctions.Full_House(test_Array), expected_Score, "Incorrect score")

    def test_Three_OAK(self):
        test_Array = [4, 4, 2, 3, 4]
        expected_Score = 17

        self.assertEqual(scoreFunctions.Three_OAK(test_Array), expected_Score, "Incorrect score")

    def test_Four_OAK(self):
        test_Array = [4, 4, 2, 4, 4]
        expected_Score = 18

        self.assertEqual(scoreFunctions.Four_OAK(test_Array), expected_Score, "Incorrect score")

    def test_Five_OAK(self):
        test_Array = [3, 3, 3, 3, 3]
        expected_Score = 50

        self.assertEqual(scoreFunctions.Five_OAK(test_Array), expected_Score, "Incorrect score")


if __name__ == '__main__':
    unittest.main()
```

The user simply needs to change around the “test_Array” and “expected_Score” values to test if the score calculation functions work properly or not.

Screenshot of the test results



```
----------------------------------------------------------------------
Ran 7 tests in 0.001s

OK
engr2-25-26-dhcp:UnitTesting Alex_Ring$ 
```

User Acceptance Tests:

Project Name			Yahtzee		
Test Case Template					
Test ID: OneToSix_Sum			Test Designed by: Robert, Alex, David		
Test Priority: High			Test Designed date: 11/9/2015		
Module Name: Score Functions			Test Executed by: Robert, Alex, David		
Test Title: Verify OneToSix_Sum works correctly not			Test Execution date: 11/9/2015		
Description: Checks if the one to six sum score is calculated correctly					
Pre-condition: The test_Array is a valid list and expected_Score and sum_Number are valid integers. Also the expected_Score should be correctly calculated by the user based on the test_Array.					
Step	Test Steps	Test Data	Expected Score	Actual Score	Pass/Fail
1	Turn on the unit testing python file	scoreFunctions_test.py	-	-	-
2	Assign test_Array in test_OneToSix_Sum an array	test_Array is [4,4,2,3,5]	-	-	-
3	Assign a value to sum_Number	sum_Number is 4	8	8	Pass

Project Name			Yahtzee		
Test Case Template					
Test ID: Full_House			Test Designed by: Robert, Alex, David		
Test Priority: High			Test Designed date: 11/9/2015		
Module Name: Score Functions			Test Executed by: Robert, Alex, David		
Test Title: Verify Full_House works correctly or not			Test Execution date: 11/9/2015		
Description: Checks if the full house score is calculated correctly					
Pre-condition: The test_Array is a valid list and expected_Score is a valid integer. Also the expected_Score should be correctly calculated by the user based on the test_Array.					
Step	Test Steps	Test Data	Expected Score	Actual Score	Pass/Fail
1	Turn on the unit testing python file	scoreFunctions_test.py	-	-	-
2	Assign test_Array in test_Full_House an array	test_Array is [4,3,4,3,4]	25	25	Pass

Project Name			Yahtzee		
Test Case Template					
Test ID: Five_OAK			Test Designed by: Robert, Alex, David		
Test Priority: High			Test Designed date: 11/9/2015		
Module Name: Score Functions			Test Executed by: Robert, Alex, David		
Test Title: Verify Five_OAK works correctly or not			Test Execution date: 11/9/2015		
Description: Checks if the five of a kind score is calculated correctly					
Pre-condition: The test_Array is a valid list and expected_Score is a valid integer. Also the expected_Score should be correctly calculated by the user based on the test_Array.					
Step	Test Steps	Test Data	Expected Score	Actual Score	Pass/Fail
1	Turn on the unit testing python file	scoreFunctions_test.py	-	-	-
2	Assign test_Array in test_Five_OAK an array	test_Array is [3,3,3,3,3]	50	50	Pass

We ran these test with multiple different test_Array's, also we ran these tests for all the other scoring functions, also with multiple different test_Array's, and they all passed the tests.

VCS: https://github.com/Djion/djion.github.io