



PRT 582
SOFTWARE ENGINEERING PROCESS AND TOOLS

Submitted to
Charles Yeo

Submitted By
Jagruthi Nagamalla
S342597

Table of Contents

<u>Introduction:</u>	2
<u>Process:</u>	2
<u>Requirement 1:</u>	3
<u>Requirement 2:</u>	6
<u>Requirement 3:</u>	8
<u>Requirement 4:</u>	9
<u>Requirement 5:</u>	11
<u>Conclusion:</u>	12

Introduction:

The following report is focusing on creating a program to calculate scrabble score for a given word with fulfilling all the given requirements using test driven development (TDD) approach.

TDD approach creates a mindset where the functional code must be written based on the failed test case. This approach is highly recommended by many software developers and writers. This method was in existence from many years, but it has become more visible now a days with the use in agile methodologies.

Objectives:

1. The result of the program should meet all the user requirements.
2. Developing the efficient program using TDD approach.

Requirements:

1. The numbers are added up correctly for a given word
2. Upper- and lower-case letters should have the same value
3. Your program should prompt user with the right feedback if user does not enter an alphabet.
4. A 15-seconds timer is shown. User is asked to input a word of a certain length. The number of alphabets required in the word is randomly generated. The program will check to ensure that the right length of word is entered before generating the score. Score will be higher if less time is used to enter the right length of word.
5. Ensure that user enters a valid word from a dictionary. The program will not tabulate the score if the word is not a proper word from a dictionary. Prompts will be given asking the user to enter a valid word if the user does not enter a valid word.

I have chosen python as programming language which I am familiar with. There are some reasons to choose python as programming language and unit testing methodology. unittest is one of the testing framework available in python by default. Python can be used for building web, mobile and desktop applications. Python has rich library of useful packages for testing like pytest, unittest, doctest, selenium etc., I am using unittest package for testing my program.

Process:

Test Driven Development:

Test Driven Development is the approach where the code is developed by preparing test case. In simple terms- the test case will be written before writing the code which seemingly fail and developing the code and altering it accordingly to make the test case pass. The TDD approach helps in not falling into the trap where most developers fall into.

The TDD is implemented by writing a test by expecting an error and writing a code based on the test case and altering it accordingly. Writing the test before code will create a tendency of thinking about the possible errors or mistakes before starting the code so it makes easier to write the code as we have bunch of possible errors list in the mind.

Unittest Testing in python:

Python standard library comes with a unit testing framework called unittest. Unittest supports test automation, sharing of setup and shut down code for tests and aggregation of tests into collections.

In unittest, it was easy to put bunch of tests together that could go under one class

How to run Unittest:

1. Import unittest from the standard library.
2. Create a class called TestSum that inherits from the TestCase class.
3. Convert the test functions into methods by adding self as the first argument.
4. Change the assertions to use the self. ...
5. Change the command-line entry point to call unittest.

We are creating one python file with prefix test word for our program.

Test_scrabble_score.py

Requirement 1:

The numbers are added up correctly for a given word.

The screenshot displays a code editor with a file named `test_scrabble_score.py` open. The code defines a class `TestProject2` that inherits from `unittest.TestCase`. It includes a test method `test_scrabblescore` that checks if the score for the word "hall" is 6. The code is as follows:

```

1 #Importing unittest from python standard library
2 import unittest
3
4
5 # Creating TestProject2 class that inherits TestCase class from u
6 class TestProject2(unittest.TestCase):
7
8     #Defining testfunction as method with self as argument
9     #Test for checking scrabblescore of given word
10    def test_scrabblescore(self):
11
12        #Storing the result of scrabble_score method in variable
13        finalresult=scrabble_score("hall")
14        #Assertion for comparing two values
15        self.assertEqual(finalresult, 6)

```

The right side of the screen shows the terminal output of running the test. It indicates that the test failed due to a `NameError` because the variable `scrabble_score` is not defined.

```

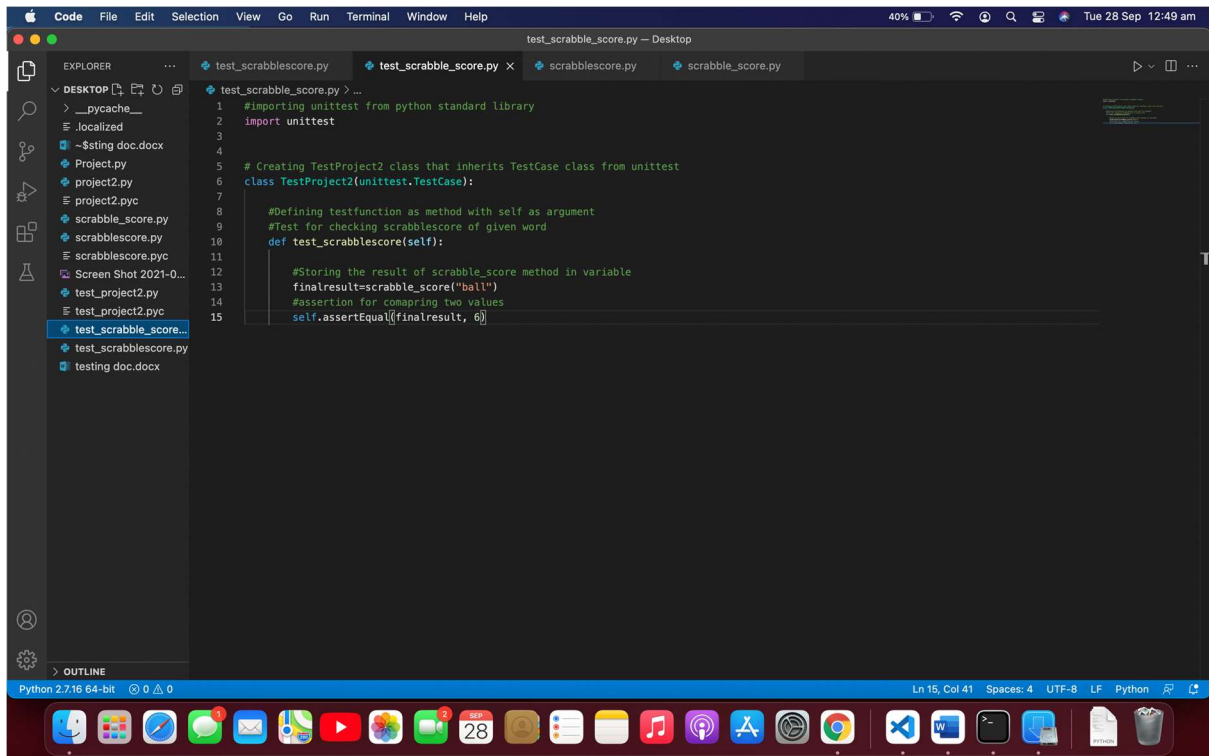
Last login: Tue Sep 28 00:11:15 on ttys000
maheshboorgu@Maheshs-MBP ~ % cd desktop
maheshboorgu@Maheshs-MBP desktop % python3 -m unittest test_scrabble_score.py
E
=====
ERROR: test_scrabblescore (test_scrabble_score.TestProject2)
-----
Traceback (most recent call last):
  File "/Users/maheshboorgu/Desktop/test_scrabble_score.py", line 13, in test_sca
    bblescore
    finalresult=scrabble_score("hall")
NameError: name 'scrabble_score' is not defined

Ran 1 test in 0.001s

FAILED (errors=1)
maheshboorgu@Maheshs-MBP desktop %

```

Left side screen showing python testcase for calculating score for a given word and right side of screen is the terminal execution showing the failing test case.



Executing through the terminal using
 Python3 -m unittest test_scrabble_score

```

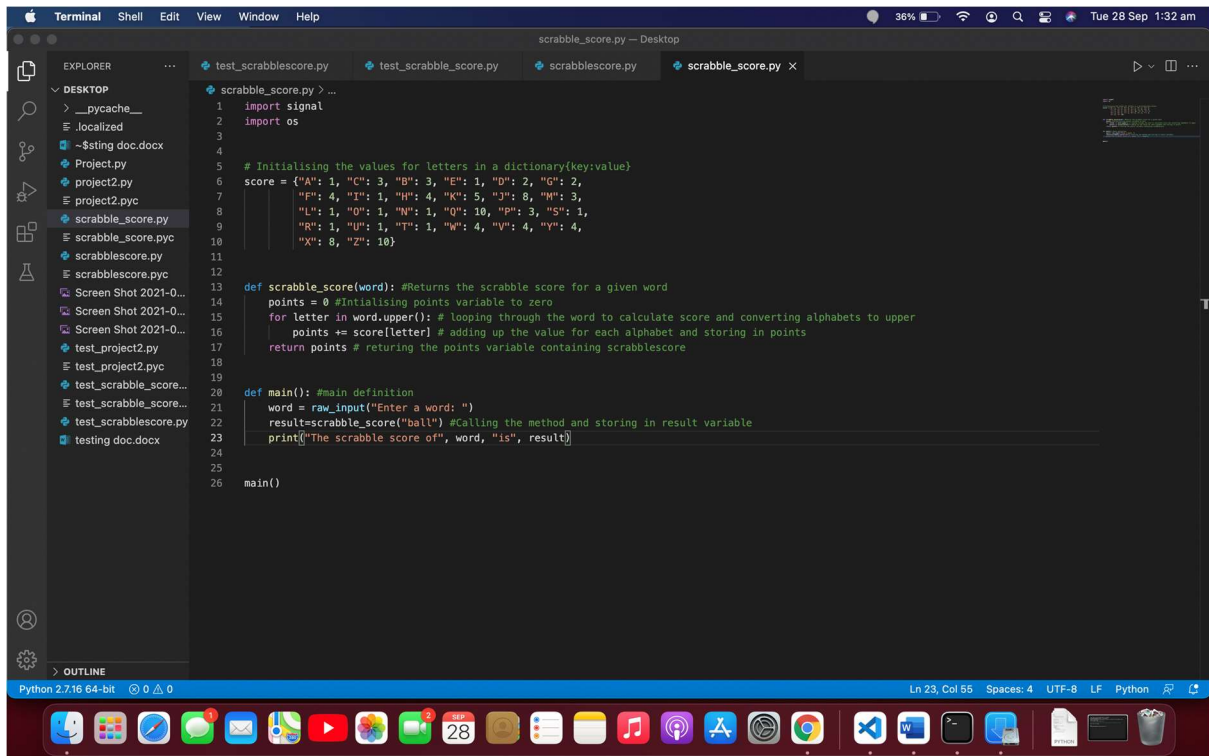
Last login: Tue Sep 28 00:11:15 on ttys000
maheshboorgu@Maheshs-MBP ~ % cd desktop
maheshboorgu@Maheshs-MBP desktop % python3 -m unittest test_scrabble_score.py
E
=====
ERROR: test_scrabblescore (test_scrabble_score.TestProject2)
-----
Traceback (most recent call last):
  File "/Users/maheshboorgu/Desktop/test_scrabble_score.py", line 13, in test_scrabblescore
    finalresult=scrabble_score("ball")
NameError: name 'scrabble_score' is not defined
-----
Ran 1 test in 0.001s

FAILED (errors=1)
maheshboorgu@Maheshs-MBP desktop %

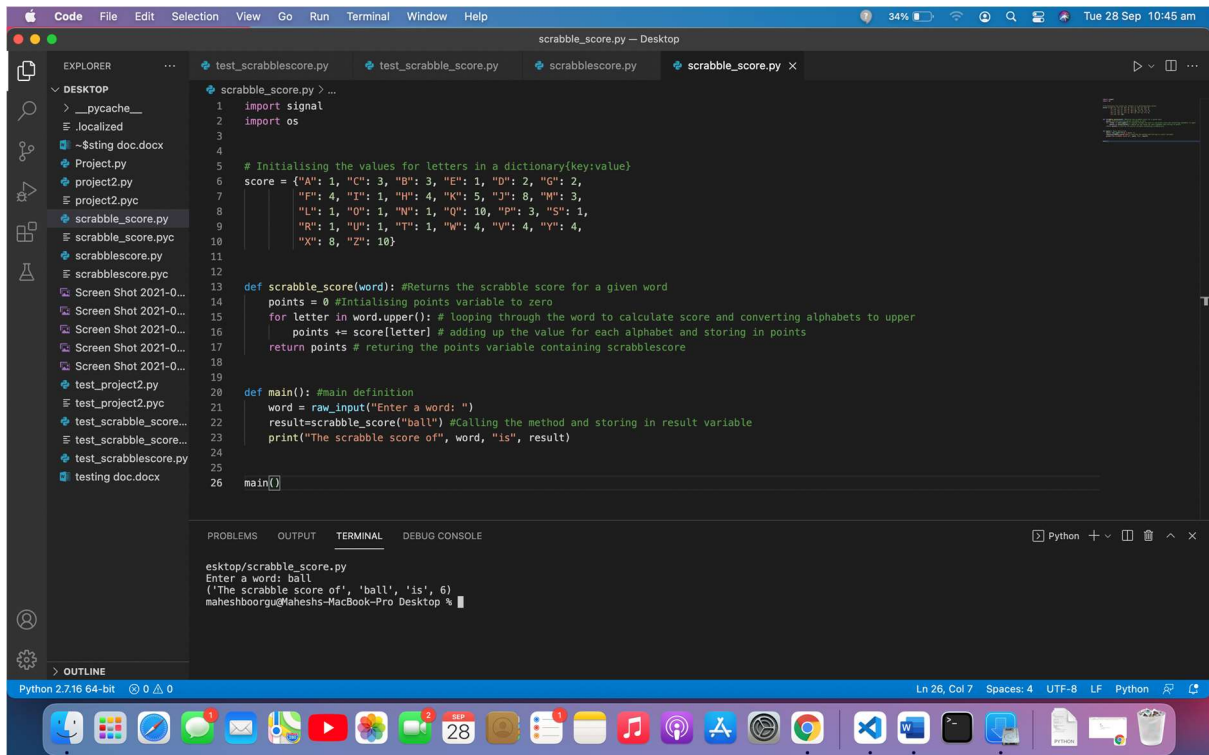
```

Now we need make modifications and make it pass.

Now we are writing the code to scrabble_score.py to make testcase pass. We are writing scrabble_score() method which takes word as argument for which we are calculating scrabble score which is in below screenshot.



```
1 import signal
2 import os
3
4 # Initialising the values for letters in a dictionary{key:value}
5 score = {"A": 1, "C": 3, "B": 3, "E": 1, "D": 2, "G": 2,
6          "F": 4, "I": 1, "H": 4, "K": 5, "J": 8, "M": 3,
7          "L": 1, "O": 1, "N": 1, "Q": 10, "P": 3, "S": 1,
8          "R": 1, "U": 1, "T": 1, "W": 4, "V": 4, "Y": 4,
9          "X": 8, "Z": 10}
10
11 def scrabble_score(word): #Returns the scrabble score for a given word
12     points = 0 #initialising points variable to zero
13     for letter in word.upper(): # looping through the word to calculate score and converting alphabets to upper
14         points += score[letter] # adding up the value for each alphabet and storing in points
15     return points # returning the points variable containing scrabble score
16
17 def main(): #main definition
18     word = raw_input("Enter a word: ")
19     result = scrabble_score("ball") #Calling the method and storing in result variable
20     print("The scrabble score of", word, "is", result)
21
22 main()
```



```
esktop/scrabble_score.py
Enter a word: ball
('The scrabble score of', 'ball', 'is', 6)
maheshboorgu@Maheshs-MacBook-Pro Desktop %
```

The above screenshot showing successful execution and displaying score.

The screenshot shows a VS Code editor with a file explorer on the left, a code editor in the center, and a terminal on the right. The code editor displays a Python script named `scrabble_score.py` with the following content:

```

1 import signal
2 import os
3
4 # Initialising the values for letters in a dictionary(key:value)
5 score = {'A': 1, 'C': 3, 'B': 3, 'E': 1, 'D': 2, 'G': 2,
6         'F': 4, 'I': 1, 'H': 4, 'K': 5, 'J': 8, 'M': 3,
7         'L': 1, 'O': 1, 'N': 1, 'Q': 10, 'P': 3, 'S': 1,
8         'R': 1, 'U': 1, 'T': 1, 'W': 4, 'V': 4, 'Y': 4,
9         'X': 8, 'Z': 10}
10
11 def scrabble_score(word): #Returns the scrabble score for a given word
12     points = 0 #Initialising points variable to zero
13     for letter in word.upper(): # looping through the word to calculate
14         points += score[letter] # adding up the value for each alphabetic
15     return points # returning the points variable containing scrabble score
16
17 def main(): #main definition
18     word = raw_input("Enter a word: ")
19     result=scrabble_score("ball") #Calling the method and storing in a variable
20     print("The scrabble score of", word, "is", result)
21
22 main()

```

The terminal on the right shows the execution of the script:

```

Last login: Tue Sep 28 01:10:19 on ttys000
maheshboorgu@Maheshs-MBP ~ % cd desktop
maheshboorgu@Maheshs-MBP desktop % python3 -m unittest test_scrabble_score
Enter a word: ball
The scrabble score of ball is 6
-
Ran 1 test in 0.001s
OK
maheshboorgu@Maheshs-MBP desktop %

```

Left side of screen showing implementation of code and right side is the terminal execution showing the pass of test case.

Requirement 2:

Upper and Lower case letters should have same value.

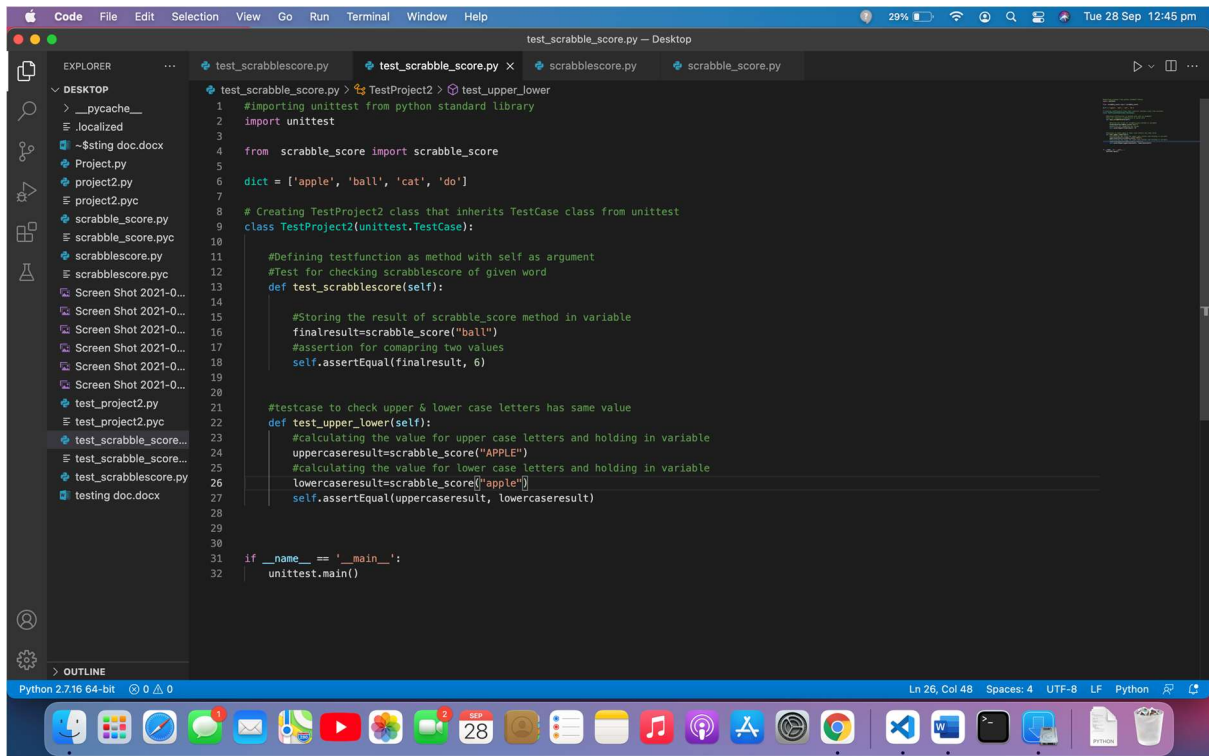
We are adding `test_upper_lower` to check upper and lower case letters has same value or not.

```
def test_upper_lower(self):
```

```

#testcase to check upper & lower case letters has same value
def test_upper_lower(self):
    #calculating the value for upper case letters and holding in variable
    uppercaseresult=scrabble_score("APPLE")
    #calculating the value for lower case letters and holding in variable
    lowercaseresult=scrabble_score("apple")
    #assertion for comparing upper and lowercase result
    self.assertEqual(uppercaseresult, lowercaseresult)

```

```

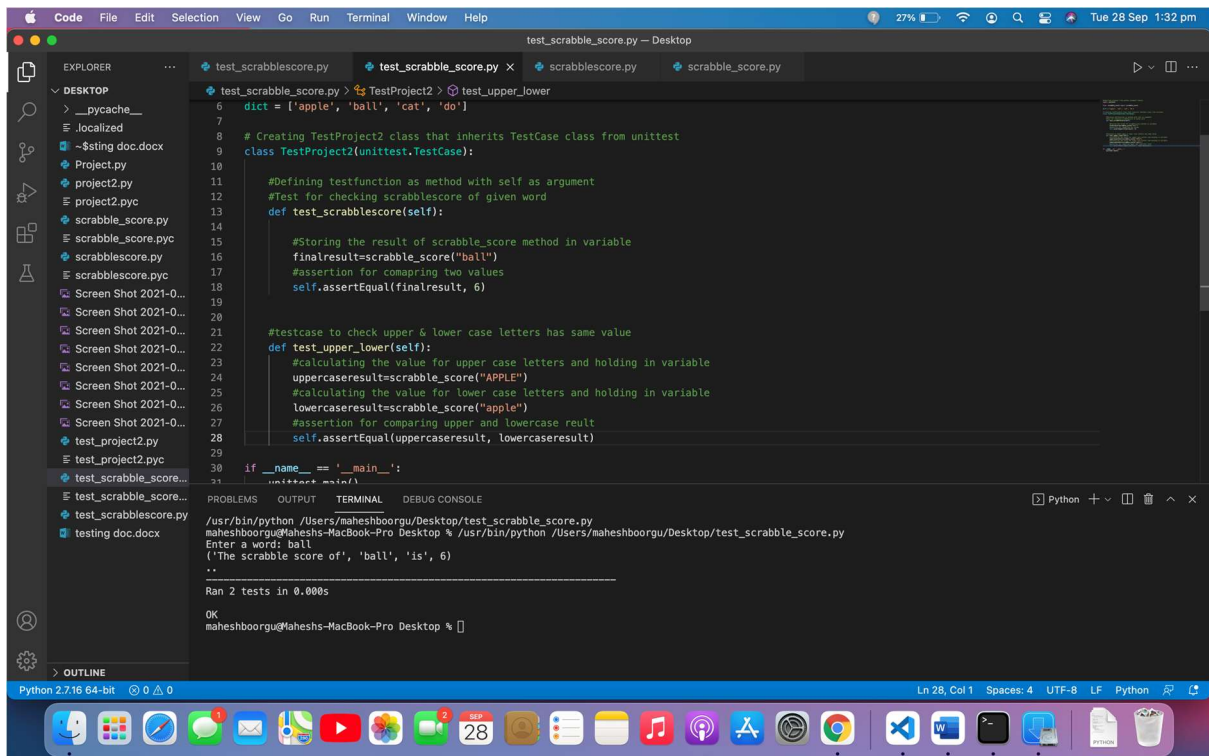
1  #Importing unittest from python standard library
2  import unittest
3
4  from scrabble_score import scrabble_score
5
6  dict = {'apple': 'ball', 'cat', 'do'}
7
8  # Creating TestProject2 class that inherits TestCase class from unittest
9  class TestProject2(unittest.TestCase):
10
11     #Defining testfunction as method with self as argument
12     #Test for checking scrabblescore of given word
13     def test_scrabblescore(self):
14
15         #Storing the result of scrabble_score method in variable
16         finalresult=scrabble_score("ball")
17         #assertion for comapring two values
18         self.assertEqual(finalresult, 6)
19
20     #testcase to check upper & lower case letters has same value
21     def test_upper_lower(self):
22         #calculating the value for upper case letters and holding in variable
23         uppercaseresult=scrabble_score("APPLE")
24         #calculating the value for lower case letters and holding in variable
25         lowercaseresult=scrabble_score("apple")
26         self.assertEqual(uppercaseresult, lowercaseresult)
27
28
29
30
31 if __name__ == '__main__':
32     unittest.main()

```

In the implementation part , in for loop we are looping through the word to calculate score.

`for letter in word.upper():`

we are using upper() method to convert lower case letters to upper case letters, so that either upper case or lower case word given, score will be same and we initialised all upper case letters values in dictionary.



```

1  #Importing unittest from python standard library
2  import unittest
3
4  from scrabble_score import scrabble_score
5
6  dict = {'apple': 'ball', 'cat', 'do'}
7
8  # Creating TestProject2 class that inherits TestCase class from unittest
9  class TestProject2(unittest.TestCase):
10
11     #Defining testfunction as method with self as argument
12     #Test for checking scrabblescore of given word
13     def test_scrabblescore(self):
14
15         #Storing the result of scrabble_score method in variable
16         finalresult=scrabble_score("ball")
17         #assertion for comapring two values
18         self.assertEqual(finalresult, 6)
19
20     #testcase to check upper & lower case letters has same value
21     def test_upper_lower(self):
22         #calculating the value for upper case letters and holding in variable
23         uppercaseresult=scrabble_score("APPLE")
24         #calculating the value for lower case letters and holding in variable
25         lowercaseresult=scrabble_score("apple")
26         self.assertEqual(uppercaseresult, lowercaseresult)
27
28
29
30
31 if __name__ == '__main__':
32     unittest.main()

```

```

/usr/bin/python /Users/maheshboorgu/Desktop/test_scrabble_score.py
maheshboorgu@Maheshs-MacBook-Pro Desktop % /usr/bin/python /Users/maheshboorgu/Desktop/test_scrabble_score.py
Enter a word: ball
('The scrabble score of', 'ball', 'is', 6)
..
Ran 2 tests in 0.000s

OK
maheshboorgu@Maheshs-MacBook-Pro Desktop %

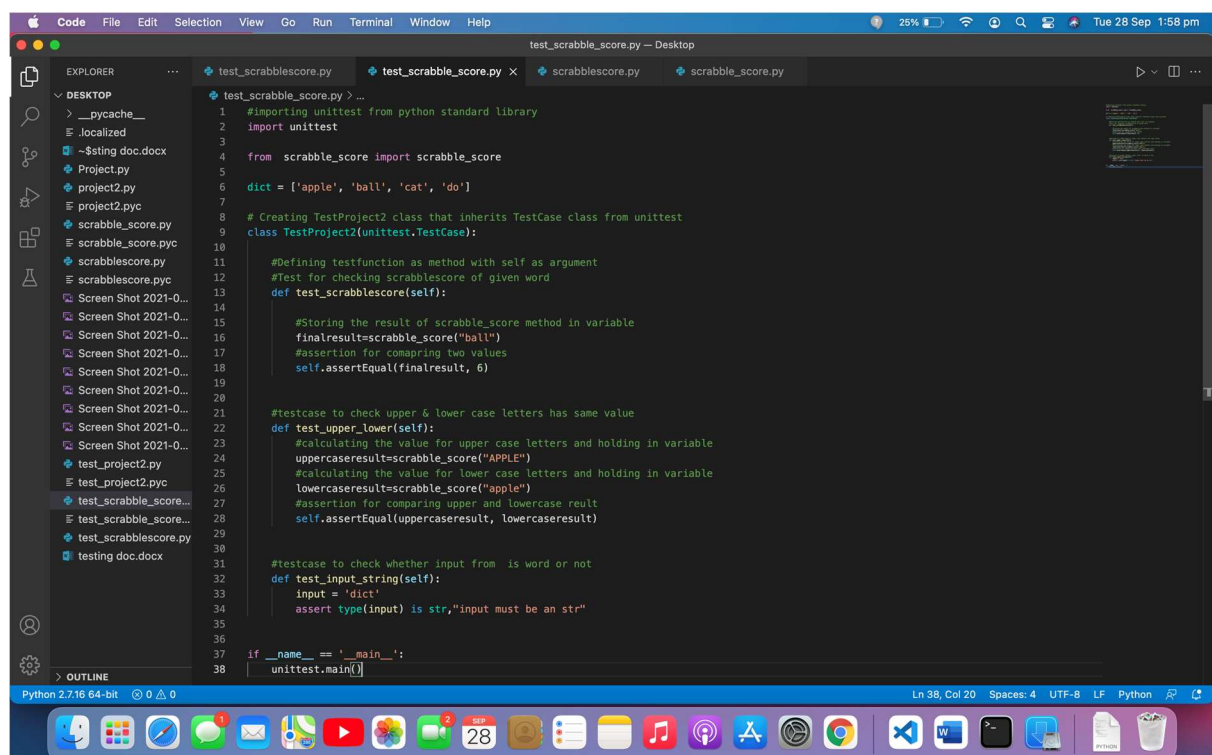
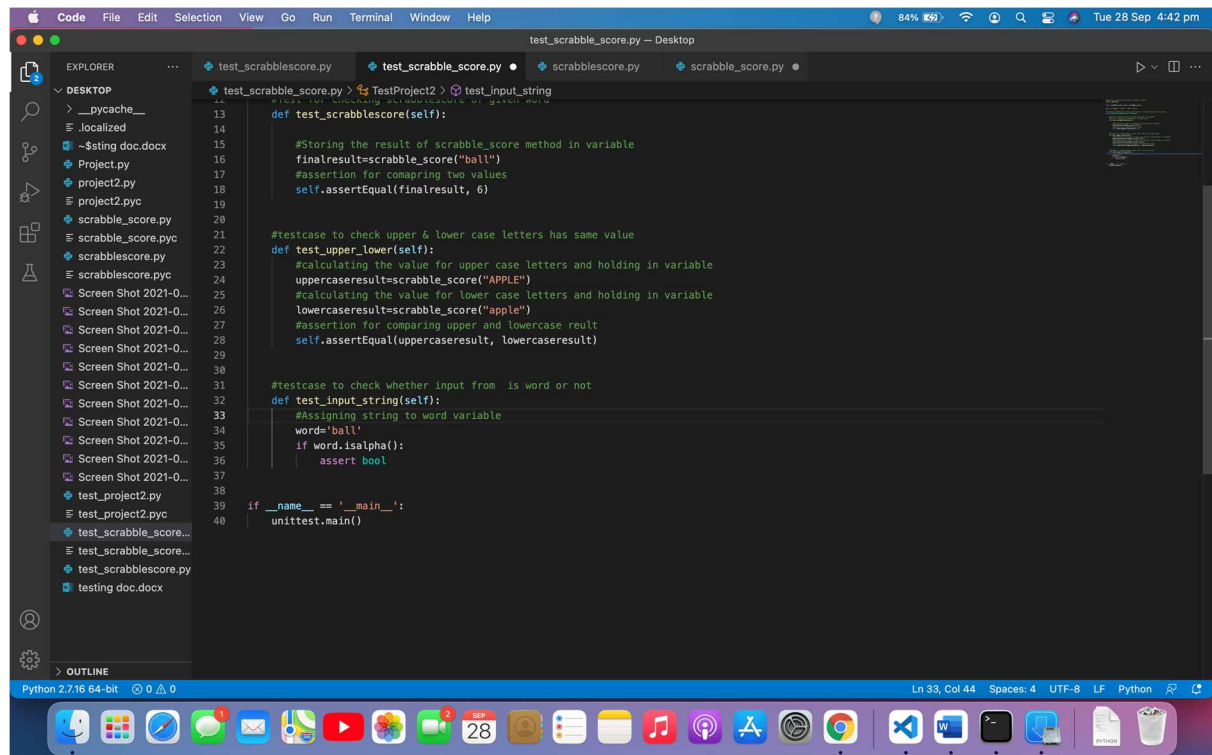
```


The above screenshot showing the upper case and lower case word having the same value and result of execution.

Requirement 3:

Your program should prompt user with the right feedback if user does not enter an alphabet.

```
def test_input_string(self):
```



In implementation part we are using `isalpha()` method to check on the input given by user.

```
if word.isalpha(): # Checking if the word entered is characters or not
    result=scrabble_score("ball") #Calling the method and storing in result variable
    print("The scrabble score of", word, "is", result) # Printing the scrabble score of word
else: #if condition fails then
    print("Please enter only alphabetical characters")
```

```
scrabble_score.py - Desktop
1 import signal
2 import os
3
4 # Initialising the values for letters in a dictionary(key:value)
5 score = {"A": 1, "C": 3, "B": 3, "E": 1, "D": 2, "G": 2,
6         "F": 4, "I": 1, "H": 4, "K": 5, "J": 8, "M": 3,
7         "L": 1, "O": 1, "N": 1, "Q": 10, "P": 3, "S": 1,
8         "R": 1, "U": 1, "T": 1, "W": 4, "V": 4, "Y": 4,
9         "X": 8, "Z": 10}
10
11 def scrabble_score(word): #Returns the scrabble score for a given word
12     points = 0 #initialising points variable to zero
13     for letter in word.upper(): # looping through the word to calculate score and converting alphabets to upper
14         points += score[letter] # adding up the value for each alphabet and storing in points
15     return points # returning the points variable containing scrabble score
16
17 def main(): #main definition
18     word = raw_input("Enter a word: ") # Prompting the user for word input
19     if word.isalpha(): # Checking if the word entered is characters or not
20         result=scrabble_score("ball") #Calling the method and storing in result variable
21         print("The scrabble score of", word, "is", result) # Printing the scrabble score of word
22     else: #if condition fails then
23         print("Please enter only alphabetical characters")
24
25 main()
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
/usr/bin/python /Users/maheshboorgu/Desktop/scrabble_score.py
maheshboorgu@Maheshs-MacBook-Pro Desktop % /usr/bin/python /Users/maheshboorgu/Desktop/scrabble_score.py
Enter a word: 234
Please enter only alphabetical characters
maheshboorgu@Maheshs-MacBook-Pro Desktop %
```

The above execution screenshot showing when user gives input numerics , it is giving message to enter alphabetical characters.

Requirement 4:

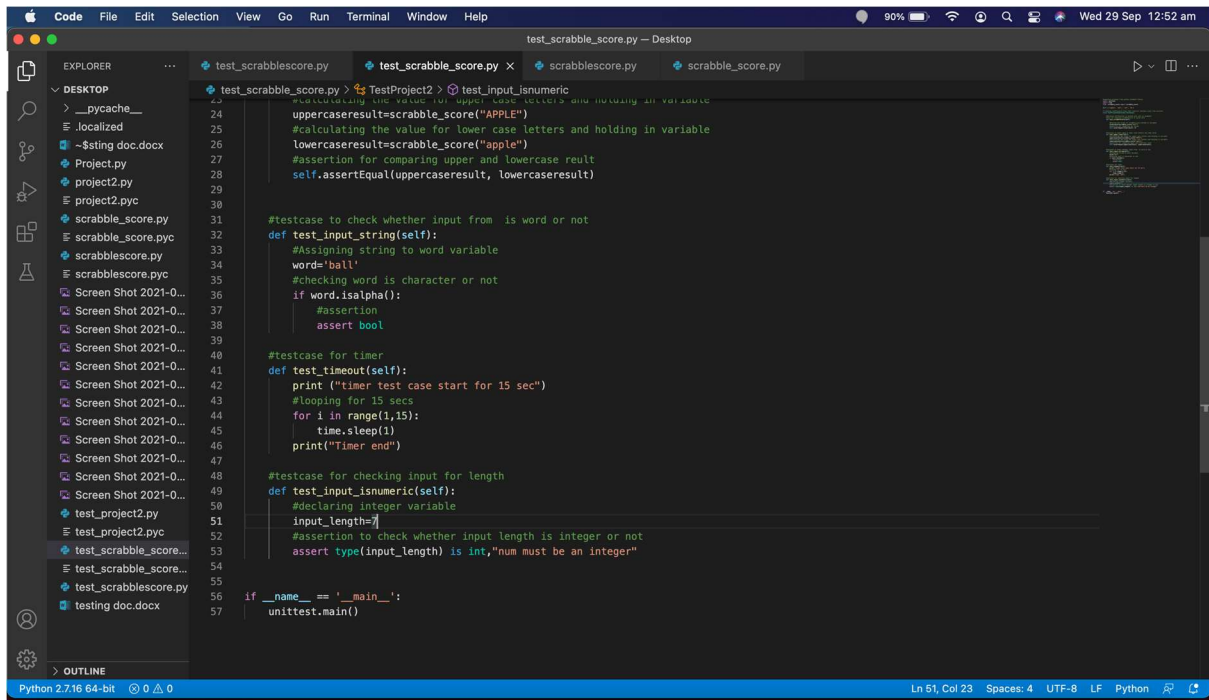
A 15-seconds timer is shown. User is asked to input a word of certain length. The number of alphabets required for a word is randomly generated. The program will check to ensure that right length of word is entered before generating the score. Score will be higher if less time is used to enter the right length of word.

Testcase for timer:

```
def test_timeout(self):
```

Test case for length of word:

```
def test_input_isnumeric(self):
```



In implementation part :

```
TIMEOUT = 15 # number of seconds your want for timeout
```

```
def interrupted(signum, frame): #called when read times out
    print ('Program timedout, please re-execute!!') #printing message for user
    os._exit(os.EX_OK) #To exit from console once timer reaches threshold 15 secs
```

```
#Method for timeout alarm
signal.signal(signal.SIGALRM, interrupted)
```

```
def main(): #main definition
    signal.alarm(TIMEOUT)
    print ('You have 15 seconds to type in your word..')#printing message on console to user
    length=int(raw_input("Please enter the length of word: ")) #prompting the user to enter length of word
    signal.alarm(0)
    signal.alarm(TIMEOUT)
    word = raw_input("Enter a word: ") # Prompting the user for word input
    signal.alarm(0)
    if word.isalpha():# checking if the characters are alphabets or not
        if length == len(word): #checking the length entered by user and length of word
            result=scrabble_score(word) # catching the result from scrabble_score method
            print("The scrabble score of", word, "is", result) # printing the message on console
        else: # else condition executed when length is not matching
            print("Please, enter correct string length word")
    else: #else condition executed when if condition checking for characters fails
        print("Please enter only alphabetical characters")
```

```

28
29
30 def main(): #main definition
31     signal.alarm(TIMEOUT)
32     print('You have 15 seconds to type in your word..') #printing message on console to user
33     length=int(raw_input("Please enter the length of word: ")) #prompting the user to enter length of word
34     signal.alarm(0)
35     signal.alarm(TIMEOUT)
36     word = raw_input("Enter a word: ") # Prompting the user for word input
37     signal.alarm(0)
38     if word.isalpha(): # checking if the characters are alphabets or not
39         if length == len(word): #checking the length entered by user and length of word
40             result=scrabble_score(word) # catching the result from scrabble_score method
41             print("The scrabble score of", word, "is", result) # printing the message on console
42         else: # else condition executed when length is not matching
43             print("Please, enter correct string length word")
44     else: #else condition executed when if condition checking for characters fails
45         print("Please enter only alphabetical characters")
46
47 #calling the main method
48 main()

```

```

/usr/bin/python /Users/mareshboorgu/Desktop/scrabble_score.py
mareshboorgu@maresh-MBP Desktop % /usr/bin/python /Users/mareshboorgu/Desktop/scrabble_score.py
You have 15 seconds to type in your word..
Please enter the length of word: Program timedout, please re-execute!!
mareshboorgu@maresh-MBP Desktop % /usr/bin/python /Users/mareshboorgu/Desktop/scrabble_score.py
You have 15 seconds to type in your word..
Please enter the length of word: 4
Enter a word: cat
Please, enter correct string length word
mareshboorgu@maresh-MBP Desktop % /usr/bin/python /Users/mareshboorgu/Desktop/scrabble_score.py
You have 15 seconds to type in your word..
Please enter the length of word: 3
Enter a word: cat
('The scrabble score of', 'cat', 'is', 5)
mareshboorgu@maresh-MBP Desktop %

```

Above screenshot showing the output after executing the code for timer and length of word.

Requirement 5:

Ensure that user enters a valid word from a dictionary. The program will not tabulate score if the word is not a proper word from a dictionary. Prompts will be given asking the user to enter a valid word if the user does not enter a valid word.

```
def test_dict_word(self):
```

```

42 #testcase for timer
43 def test_timeout(self):
44     print("Timer test case start for 15 sec")
45     #looping for 15 secs
46     for i in range(1,15):
47         time.sleep(1)
48     print("Timer end")
49
50 #testcase for checking input for length
51 def test_input_isnumeric(self):
52     #declaring integer variable
53     input_length=7
54     #assertion to check whether input length is integer or not
55     assert type(input_length) is int,"num must be an integer"
56
57
58 def test_dict_word(self):
59     for i in range(len(dict)):
60         self.assertEqual(dict[i], 'ball')
61
62
63 if __name__ == '__main__':
64     unittest.main()

```

```

Timer end
..
ERROR: test_dict_word (__main__.TestProject2)
Traceback (most recent call last):
  File "/Users/mareshboorgu/Desktop/test_scrabble_score.py", line 59, in test_dict_word
    for i in range(len(dict)):
TypeError: object of type 'type' has no len()

Ran 6 tests in 14.061s

FAILED (errors=1)
mareshboorgu@maresh-MBP Desktop %

```

Above screenshot showing the failing testcase .


```

def main():
    signal.alarm(TIMEOUT)
    print("You have 15 seconds to type in your word..")
    length = int(raw_input("Please enter the length of word: "))
    signal.alarm(0)
    word = raw_input("Enter a word: ")
    if word.isalpha():
        if length == len(word):
            result = scrabble_score(word)
            if word.lower() in dict:
                print("The scrabble score of", word, "is", result)
            else:
                print("Entered word should exist in dictionary")
        else:
            print("Please, enter correct string length word")
    else:
        print("Please enter only alphabetical characters")
    #calling the main method
    main()

```

```

/usr/bin/python /Users/mareshboorgu/Desktop/scrabble_score.py
mareshboorgu@maresh-MBP Desktop % /usr/bin/python /Users/mareshboorgu/Desktop/scrabble_score.py
You have 15 seconds to type in your word..
Please enter the length of word: 3
Enter a word: cat
Entered word should exist in dictionary
mareshboorgu@maresh-MBP Desktop % /usr/bin/python /Users/mareshboorgu/Desktop/scrabble_score.py
You have 15 seconds to type in your word..
Please enter the length of word: 3
Enter a word: cat
('The scrabble score of', 'cat', 'is', 5)
mareshboorgu@maresh-MBP Desktop %

```

Above screenshot showing success execution of requirement. Giving input word which is not in dictionary, gives message to user to enter word from dictionary. The input word cat is in dictionary, calculated score and displayed as shown in screenshot.

```

def test_timeout(self):
    print("timer test case start for 15 sec")
    #looping for 15 secs
    for i in range(1,15):
        time.sleep(1)
    print("Timer end")

#testcase for checking input for length
def test_input_isnumeric(self):
    #declaring integer variable
    input_length=7
    #assertion to check whether input length is integer or not
    assert type(input_length) is int,"num must be an integer"

def test_dict_word(self):
    for i in range(len(dict)):
        self.assertEqual(dict[i], 'ball')

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

```

```

/usr/bin/python /Users/mareshboorgu/Desktop/test_scrabble_score.py
mareshboorgu@maresh-MBP Desktop % /usr/bin/python /Users/mareshboorgu/Desktop/test_scrabble_score.py
You have 15 seconds to type in your word..
Please enter the length of word: 2
Enter a word: do
('The scrabble score of', 'do', 'is', 3)
....timer test case start for 15 sec
Timer end
..
Ran 6 tests in 14.065s
OK
mareshboorgu@maresh-MBP Desktop %

```

The above screenshot showing successful execution of all test cases.

```

26 @method for timeout alarm
27 signal.signal(signal.SIGALRM, interrupted)
28
29
30 def main(): #main definition
31     signal.alarm(TIMEOUT)
32     print('You have 15 seconds to type in your word..') #printing message on console to user
33     length=int(raw_input("Please enter the length of word: ")) #prompting the user to enter length of word
34     signal.alarm(0)
35     signal.alarm(TIMEOUT)
36     word = raw_input("Enter a word: ") # Prompting the user for word input
37     signal.alarm(0)
38     if word.isalpha():# checking if the characters are alphabets or not
39         if length == len(word): #checking the length entered by user and length of word
40             result=scrabble_score(word) # catching the result from scrabble_score method
41             if word.lower() in dict: # checking the word in dictionary
42                 print("The scrabble score of", word, "is", result) # printing the message on console
43             else: # else condition executed when if condition fails
44                 print("Entered word should exist in dictionary") #printing message on console
45             else: # else condition executed when length is not matching
46                 print("Please, enter correct string length word")
47         else: #Else condition executed when if condition checking for characters fails
48             print("Please enter only alphabetical characters")
49
50 #calling the main method
51 main()

```

desktop/scrabble_score.py
 You have 15 seconds to type in your word..
 Please enter the length of word: 7
 Enter a word: cabbage
 ('The scrabble score of', 'cabbage', 'is', 14)
 maheshboorgu@maheshs-MacBook-Pro Desktop %

The above screenshot showing successful execution of scrabble_score.py with the output for “cabbage” is 14.

Conclusion:

I Would like to conclude saying that the whole project was a new experience for me. Working on testing module in python is completely new and I struggled at the beginning but with the help of online sources I got to know how the python used for testing purpose. Functional development has gone through very well for me and learnt many new things in testing in the process.