# PRT 582
# SOFTWARE ENGINEERING PROCESS AND TOOLS

Submitted to
Charles Yeo

Submitted By
Jagruthi Nagamalla
S342597

## Table of Contents

## 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 ùnittest. 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

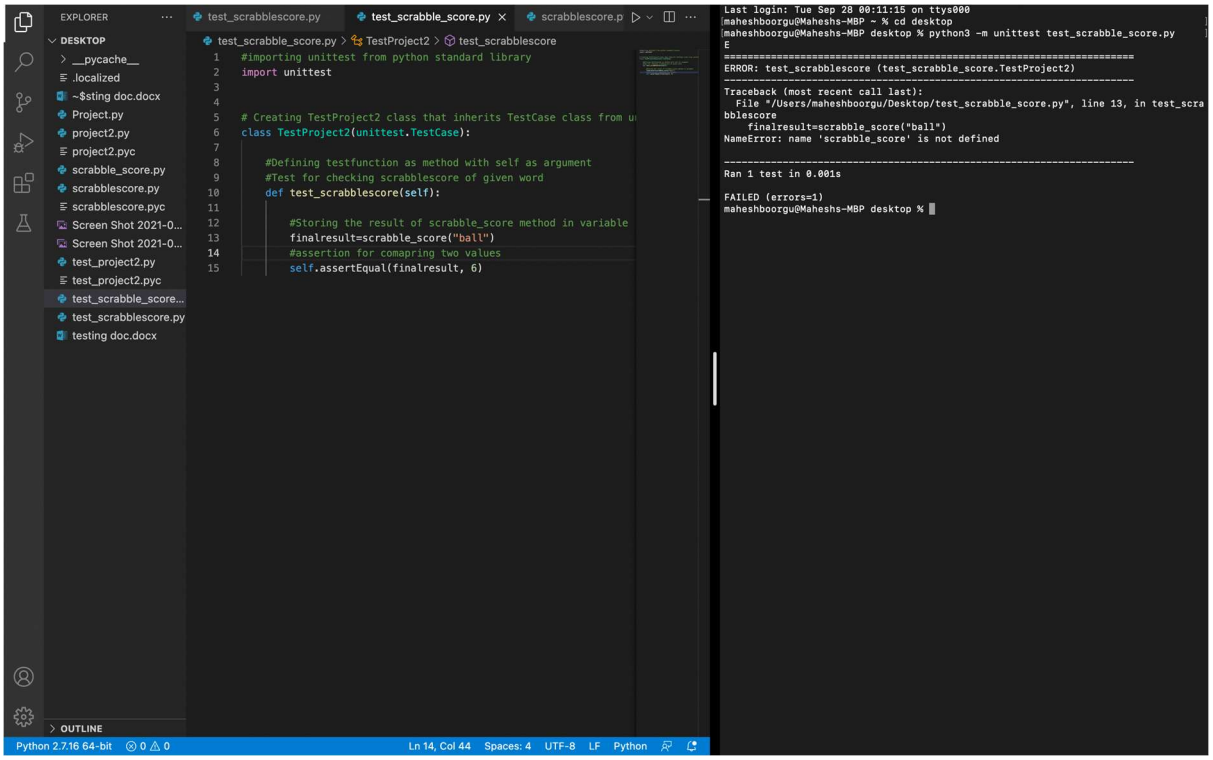
How to run Unittest:

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.

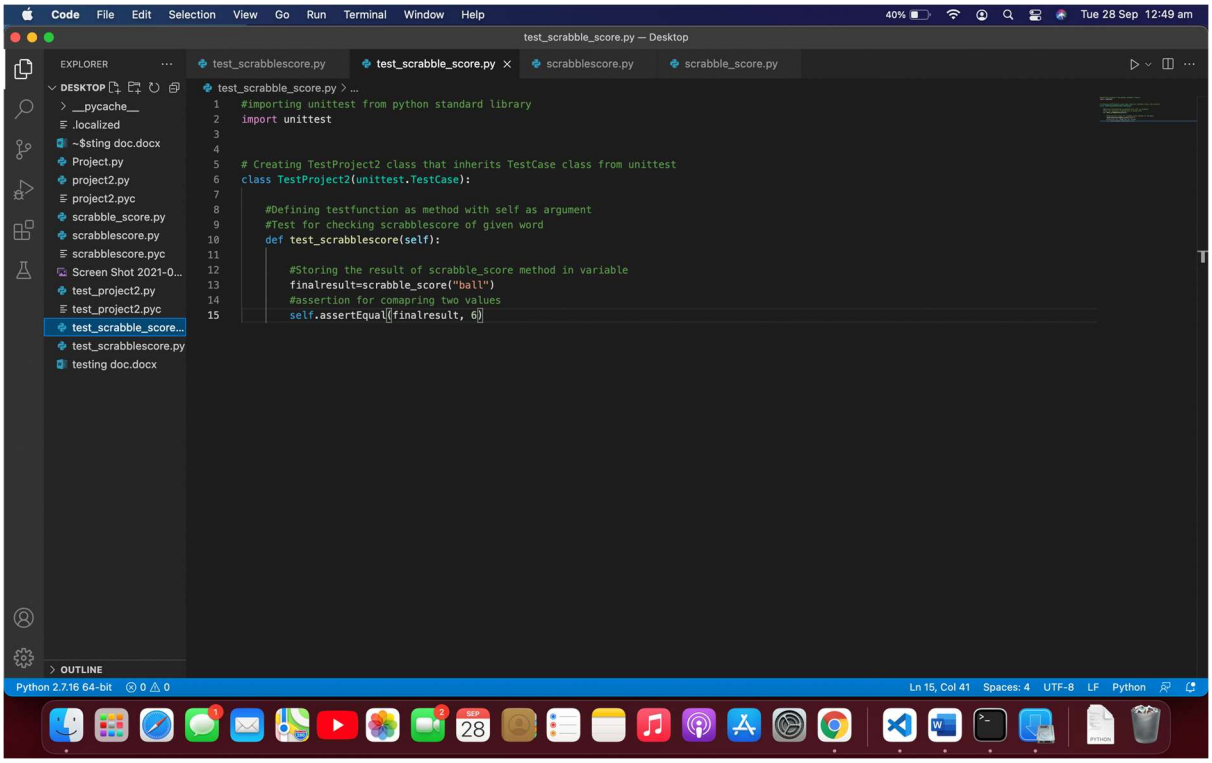
We are creating one python file with prefix test word for our program.
Test_scrabble_score.py

## Requirement 1:

The numbers are added up correctly for a given word.




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.

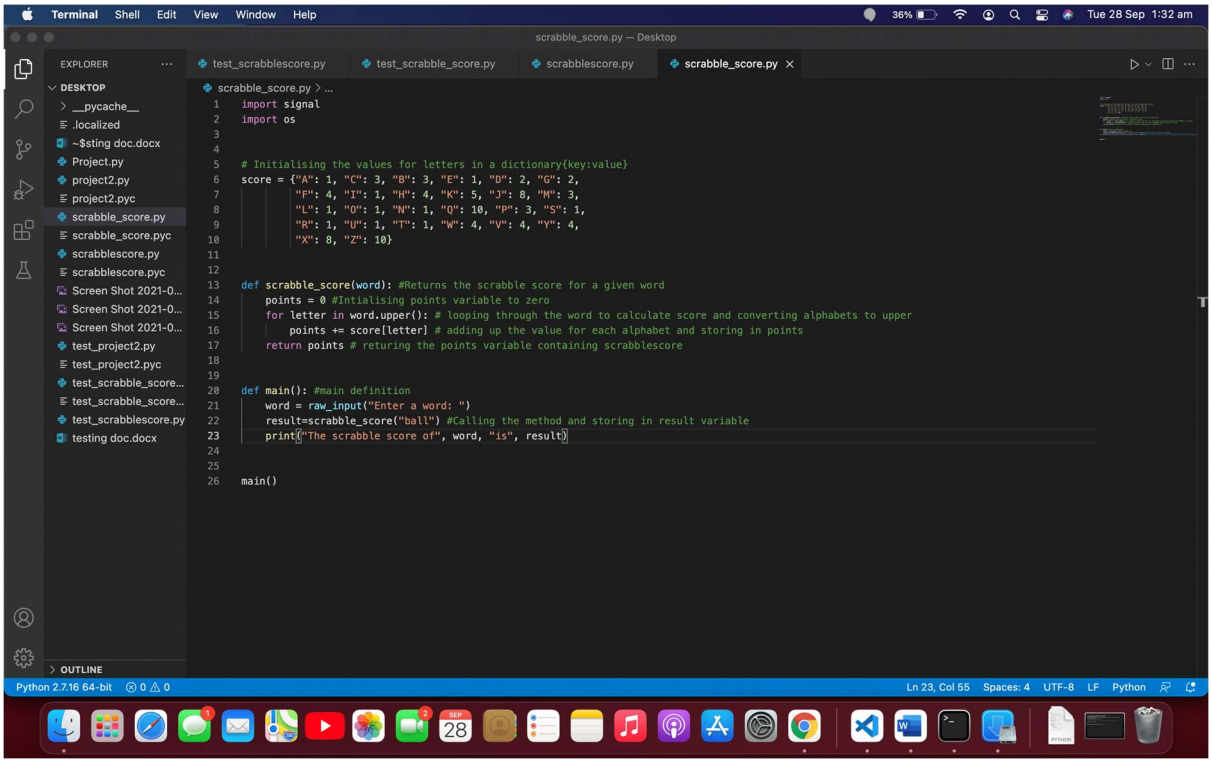


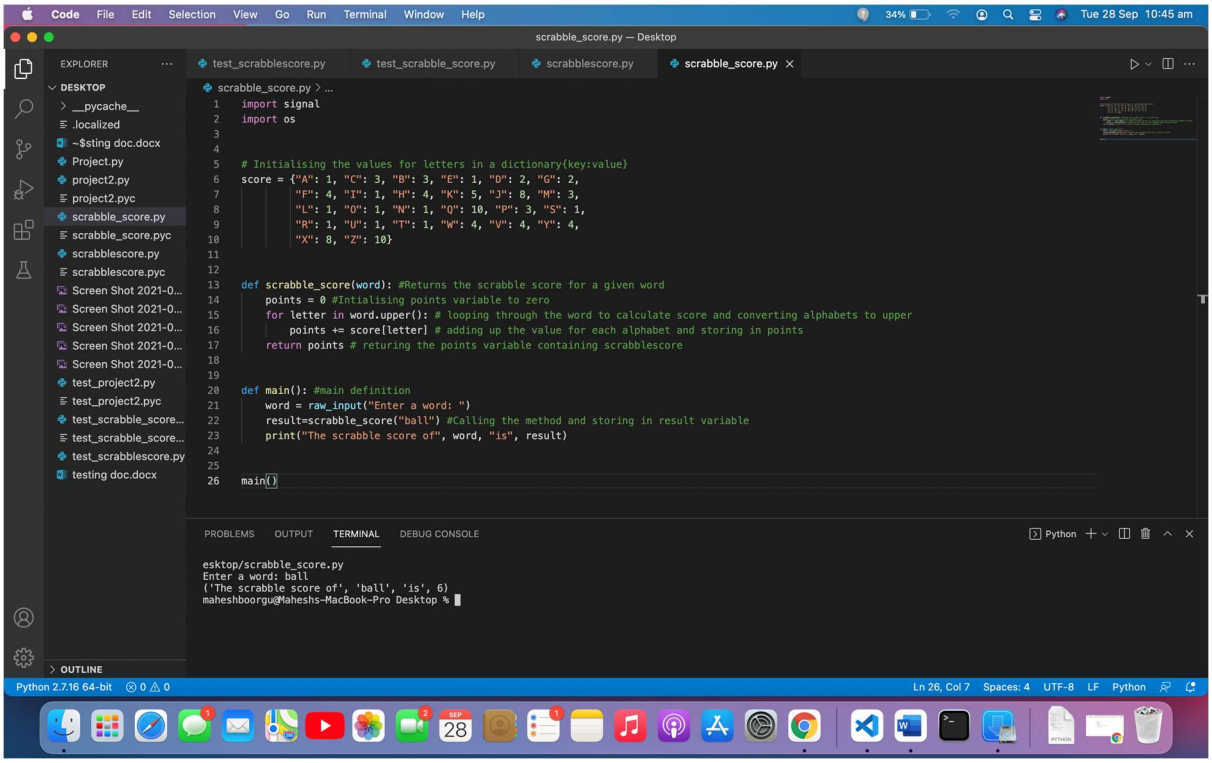

Executing through the terminal using
Python3 -m unittest test_scrabble_score

```
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 %
```

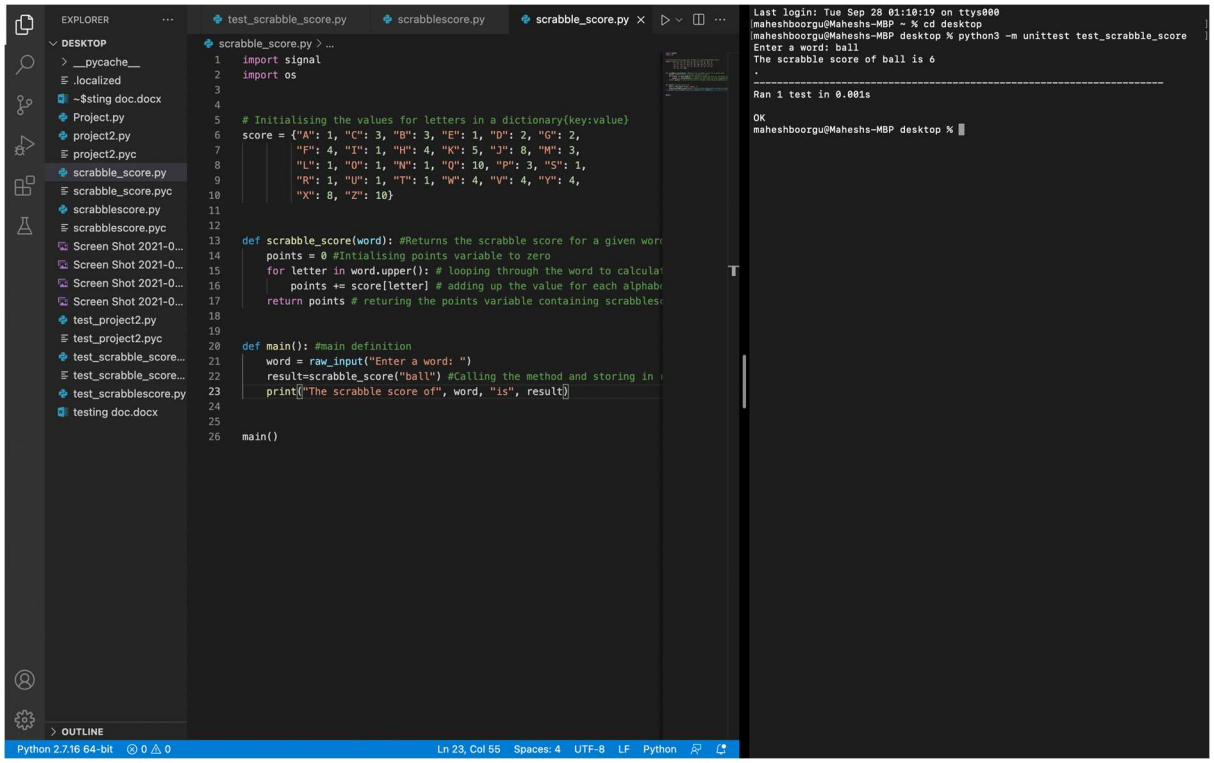
Now we need make modifications and make it pass.

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.




The above screenshot showing successful execution and displaying score.



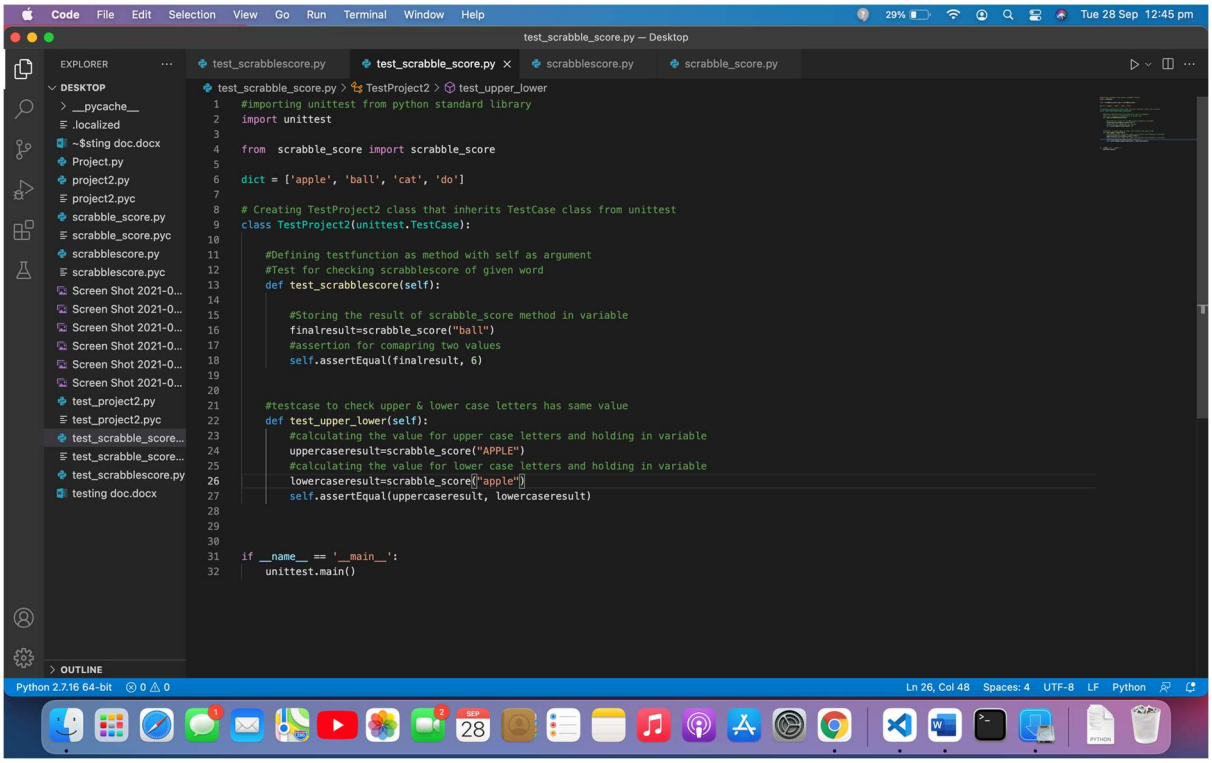

Left side of screen showing implementation of code and right side is the terminal execution showing the pass of testcase.

## 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):
```

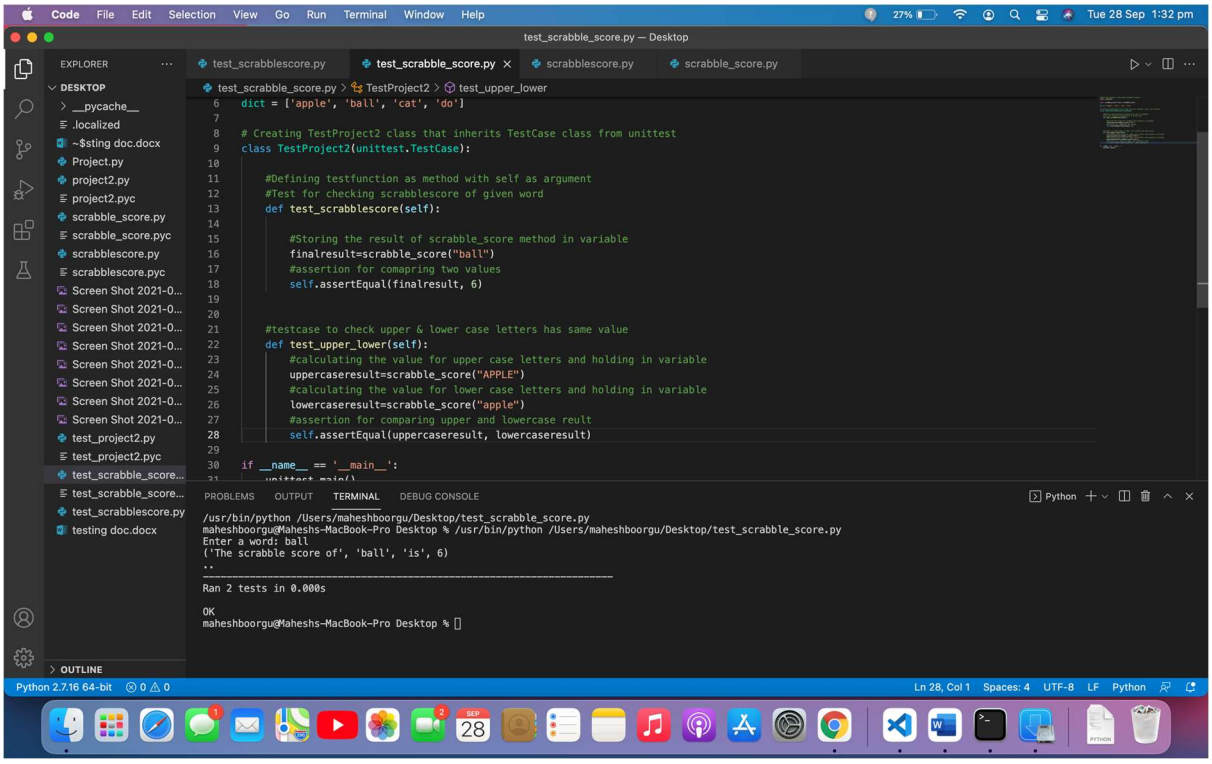
```
#testcase to check upper & lower case letters has same value
    def test_upper_lower(self):
        #calculating the value for upper case letters and holding in variable
        uppercaseresult=scrabble_score("APPLE")
        #calculating the value for lower case letters and holding in variable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




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.

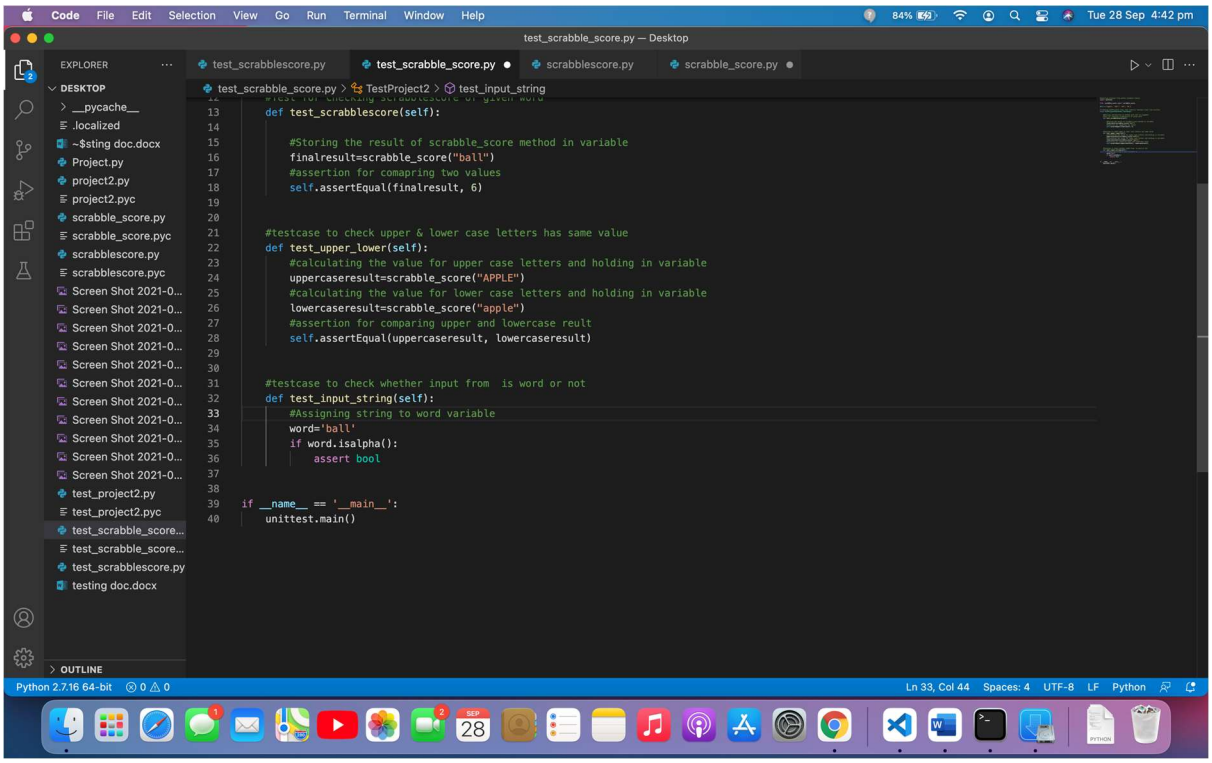


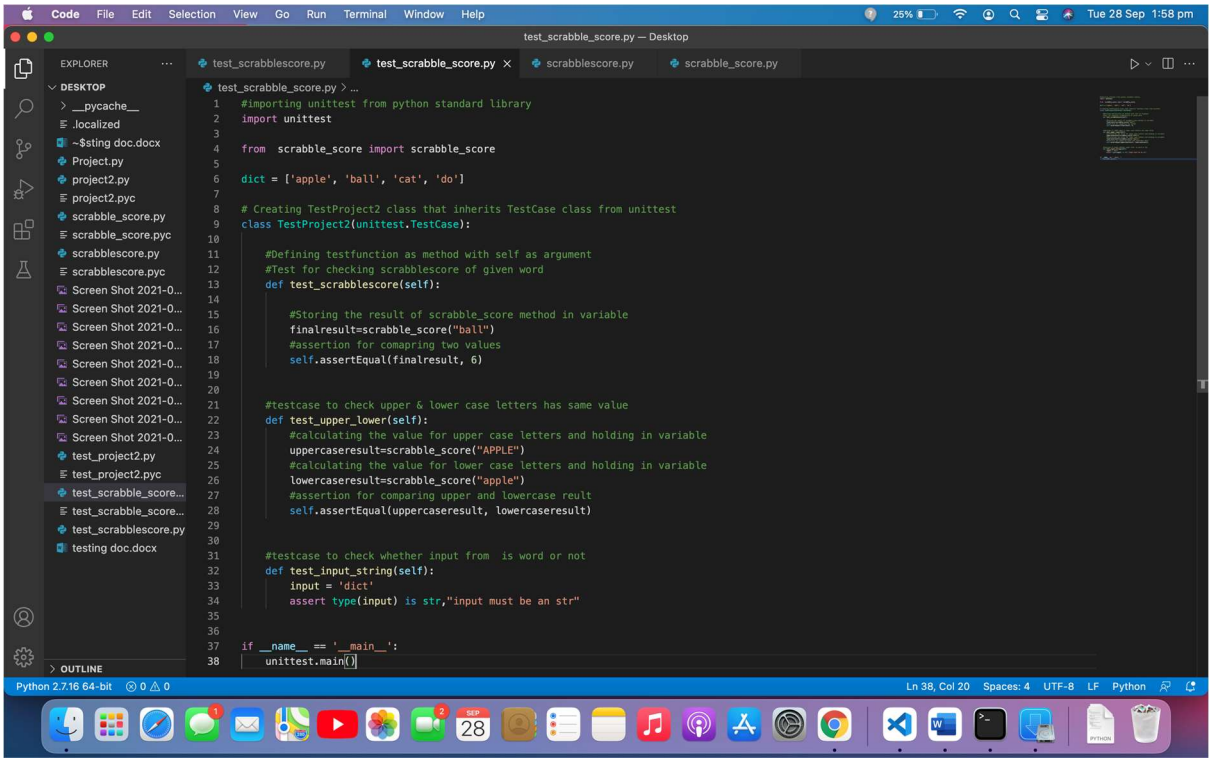
The above screenshot showing the upper case and lower case word having the same value and result of execution.

## Requirement 3:

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

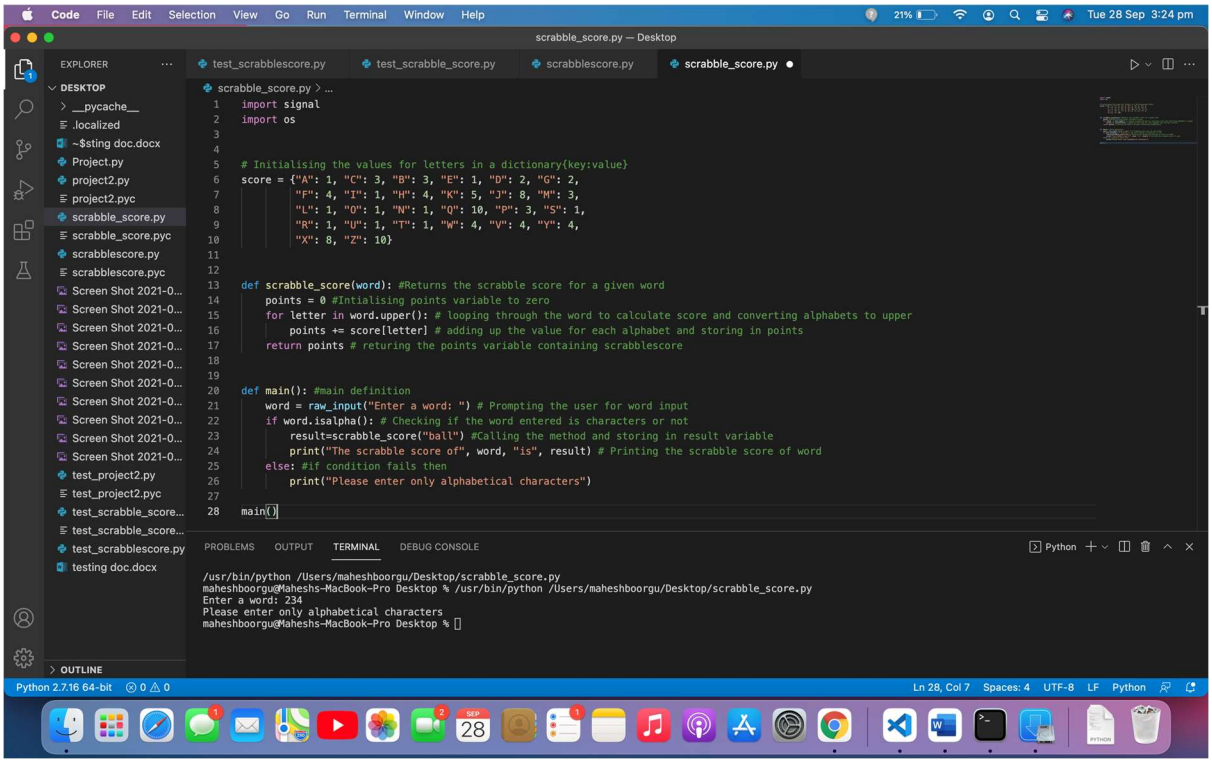
```
def test_input_string(self):
```



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")
```




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

## Requirement 4:

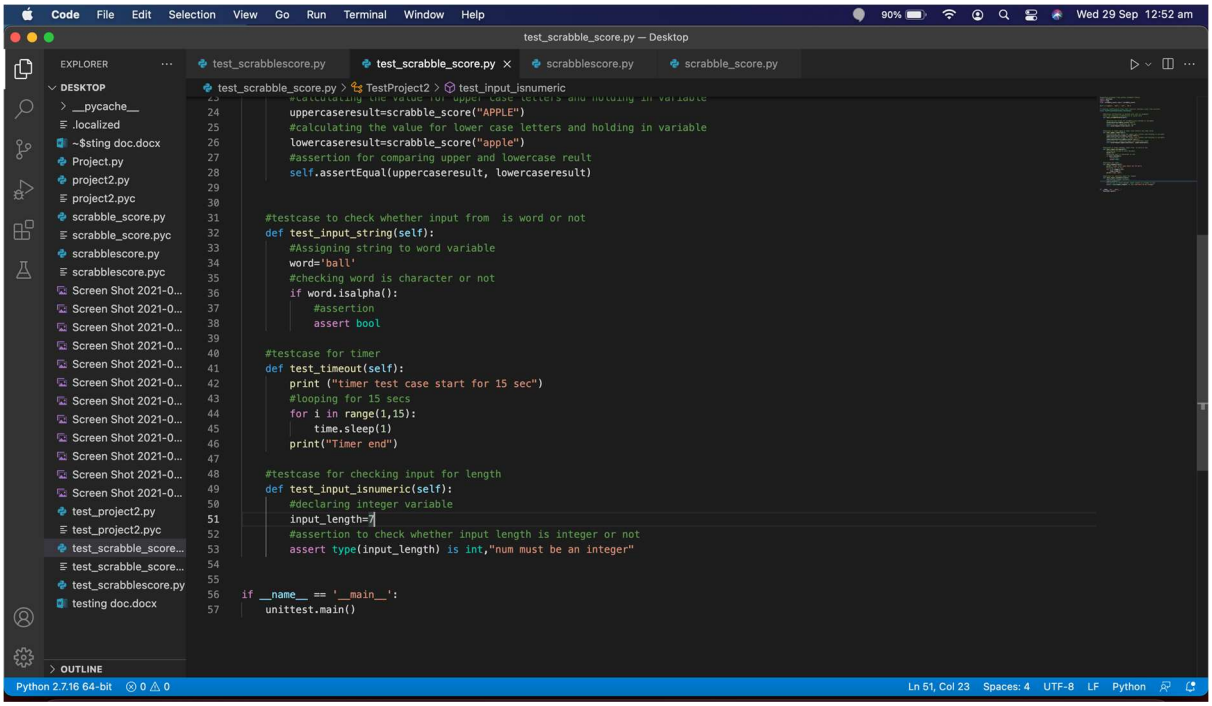
A 15-seconds timer is shown. User is asked to input a word of certain length. The number of alphabets required for a word is randomly generated. The program will check to ensure that right length of word is entered before generating the score. Score will be higher if less time is used to enter the right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

Testcase for timer:

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

Test case for length of word:

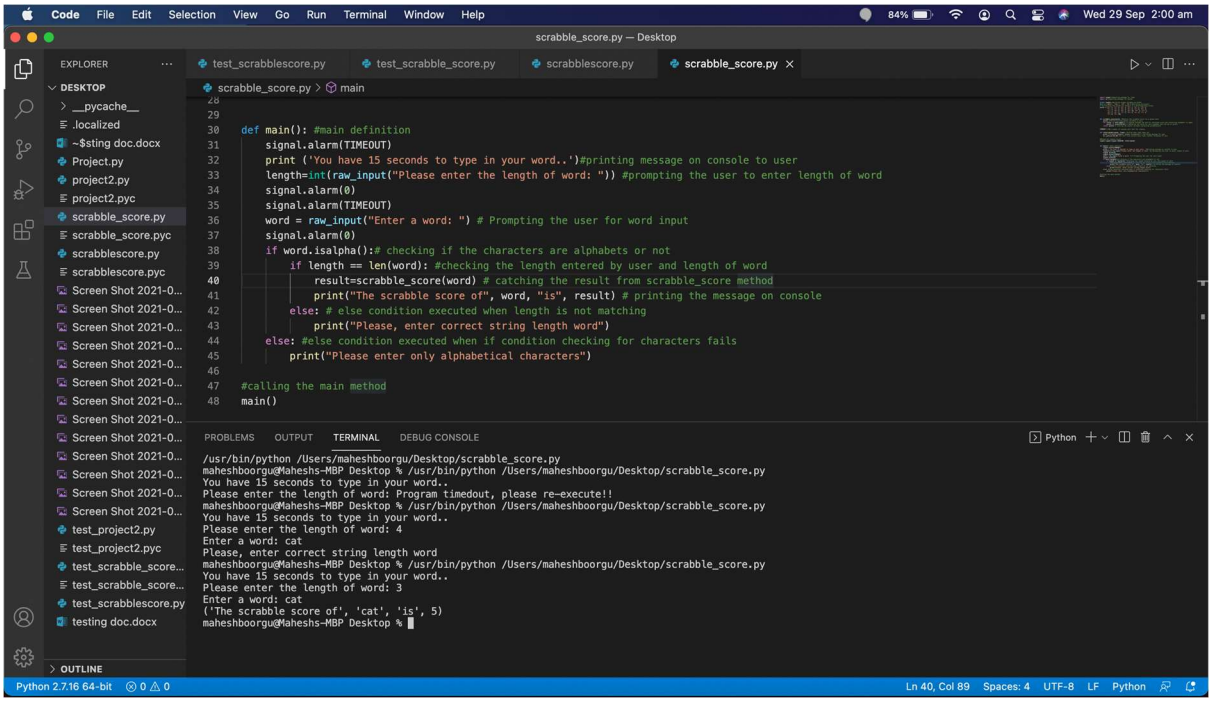
```
def test_input_isnumeric(self):
```




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")
```

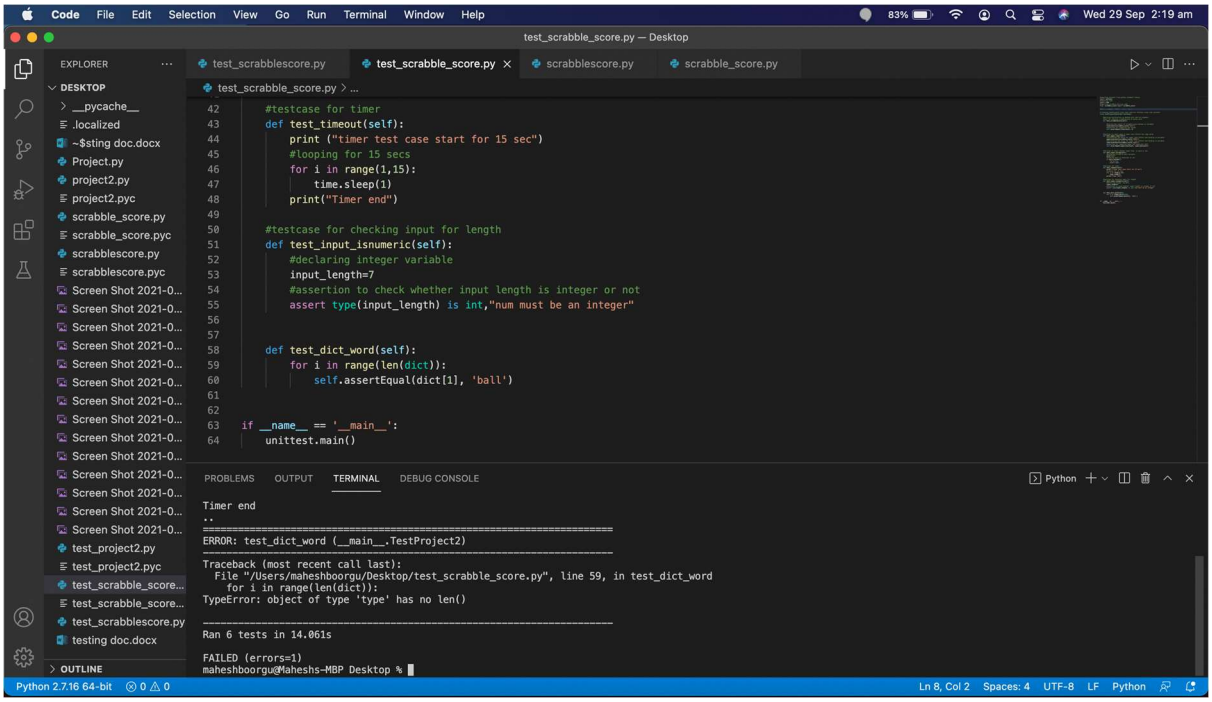



Above screenshot showing the output after executing the code for timer and length of word.

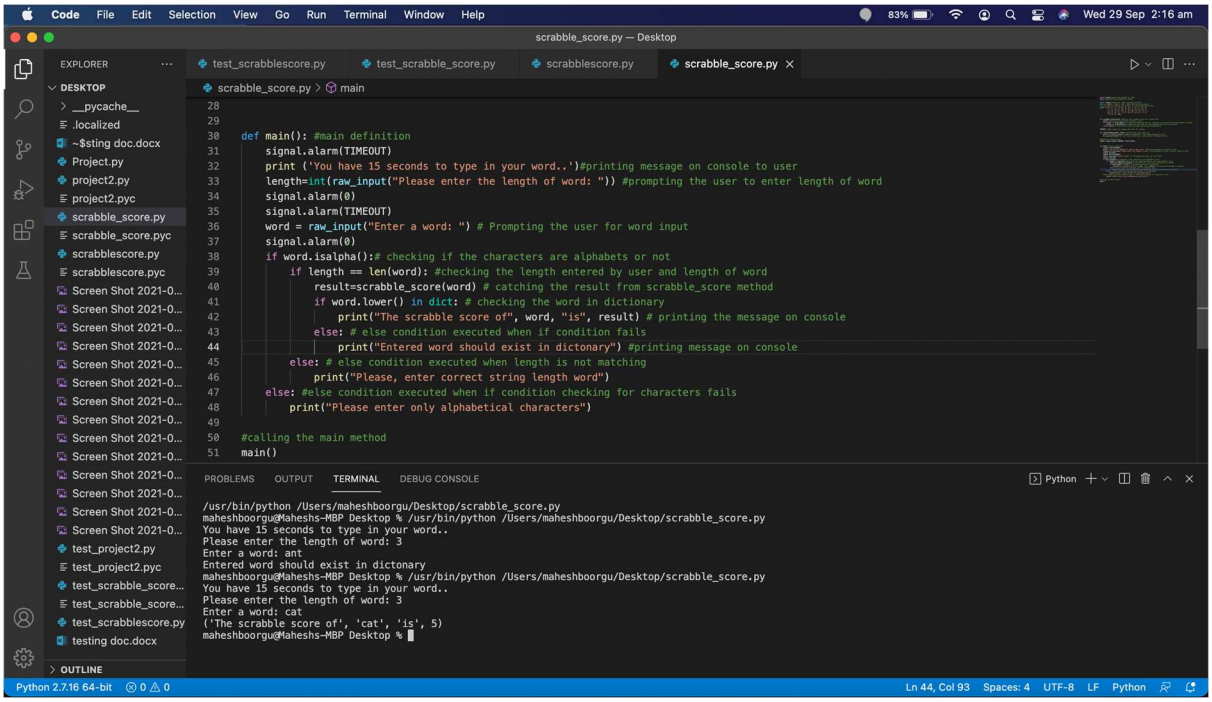
## Requirement 5:

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.

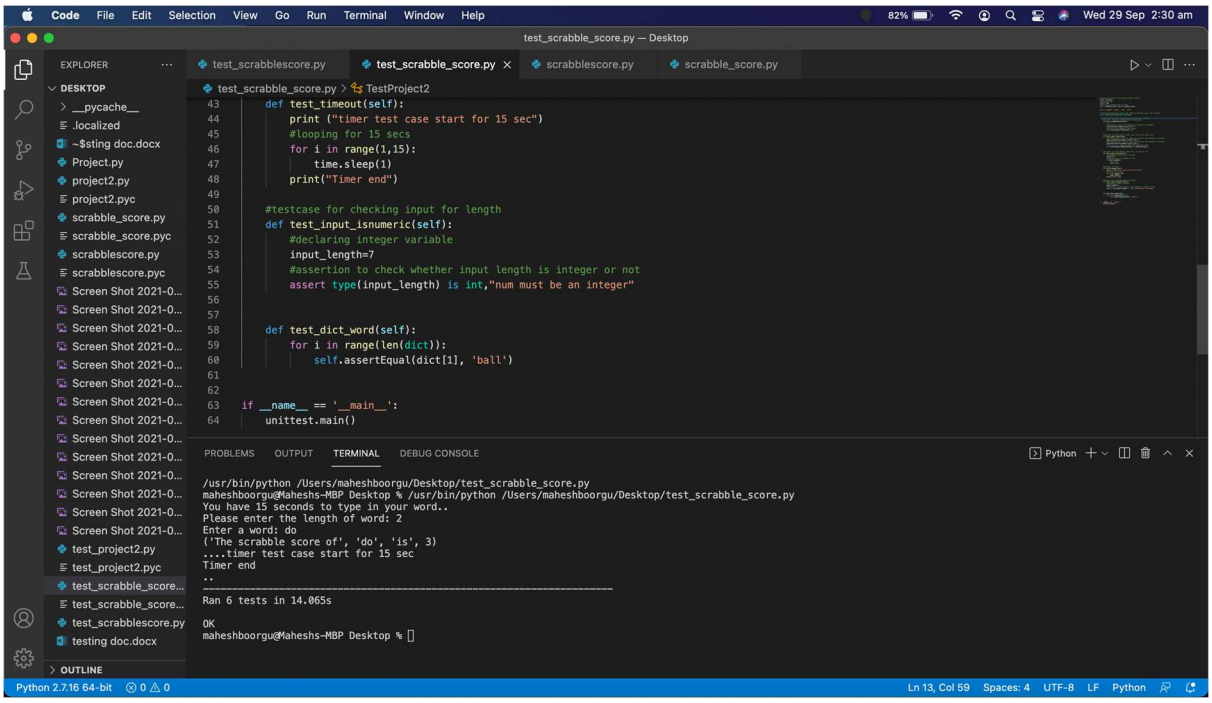
```
def test_dict_word(self):
```



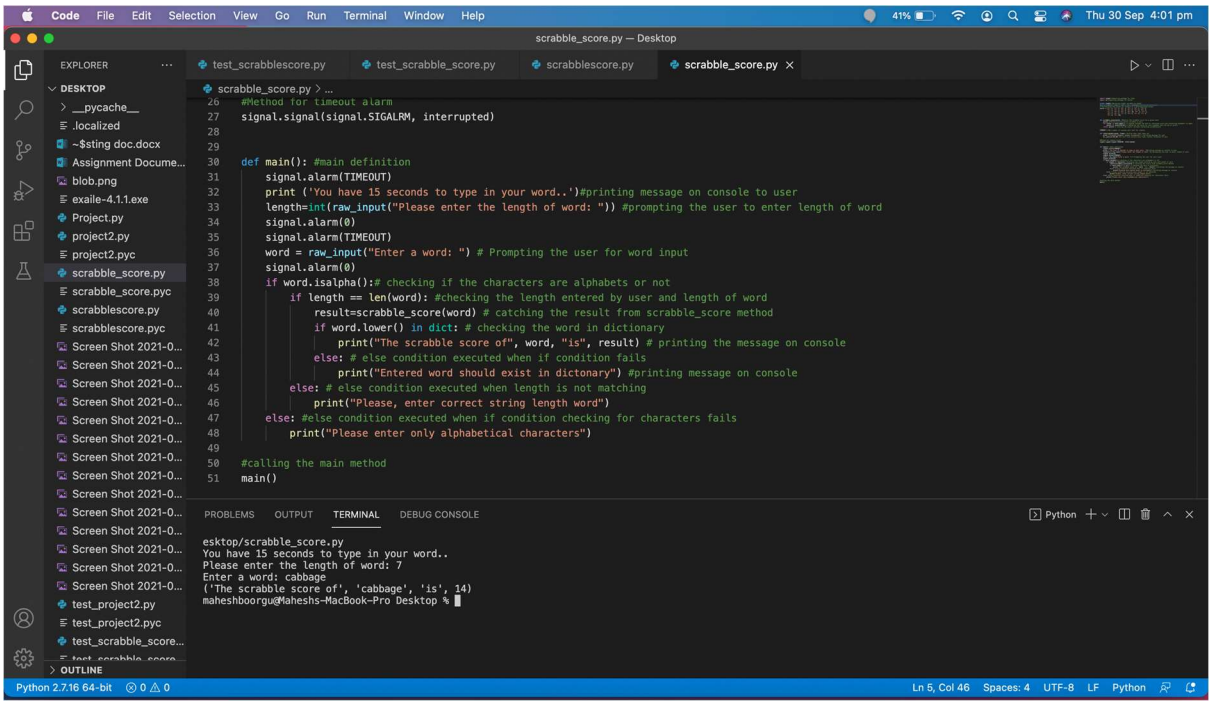

Above screenshot showing the failing testcase .




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.




The above screenshot showing successful execution of all test cases.




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.