

SOFTWARE ENGINEERING: PROCESS AND TOOLS



October 1, 2021

sagar manandhar

S348743

# **1.Introduction: -**

# Code needs to generate the number that would be used to fix length of word. User interface is required where user can enter any word as per required limit of length that is random generator for word. Then dictionary for the words is available to match with user input.

Run the program in timer environment to get value of word. User would accept only letters.

# Users need to interact with Scrabble word using programming language like Python to create dictionary with key and value as pair. check the word including all letters to count the value.

# To interact with word, User need to assign value to each letter first using dictionary. Then check entered word is only containing letters. In case any symbol, user has to create an error message. Create testing environment for all type of cases. Timer is needed to set timer value. Timer would allow the output of testing input after timed out only.

User input is required to enter value while running the program in form of a line or sentence. System would interpret the line or sentence as in the form of list of words or in form of array words. Testing of word would be on random based function. Randomly system would select any word and test the word and calculate the value.

# **Required Technology used: -**

PyCharm Community Edition 2020.3.3 x64 is implemented for Python coding. As Python supports dictionary and unit testing also. According to the problem, Python is easily implemented to solve the problem.

1. **Process: -**

Processes are required to manage the system as follows: -

1. Process for dictionary to contain all letters and assigned value.
2. Process for timer is required to set time.
3. Process for user input required at run time.
4. Process to select word randomly.
5. Process to calculate the value for scrabble word is mandatory.
6. Testing tool is required to test the values while comparing expected value with exact value calculated from result**.**

TestScrabble class to design interface to run the code and imported module to test different cases at unit level-based testing.

In which each module or unit are tested individually by comparing excepted value with resulted one.

**Use of Inbuilt class using import module like:**

1. Unit test

**Modules used are:**

import unittest  
import new\_word\_list

**Code:**

**Program file1: -**

new\_word\_list.py

#A file to design module for Scrabble score

import time  
import random  
  
def scrabble\_score\_COUNT():  
 # DICTIONARY DESIGNED TO ASSIGN VALUES  
  
 score\_1 = {"a": 1, "b": 3, "c": 3, "e": 1, "g": 2,  
 "i": 1, "k": 5, "j": 8, "m": 3,  
 "l": 1, "o": 1, "n": 1, "p": 3, "s": 1,  
 "r": 1, "u": 1, "t": 1,  
 "x": 8, "q": 10, "z": 10, "d": 2, "f": 4, "h": 4, "v": 4, "y": 4, "w": 4, }  
 rand\_num1 = random.randint(5, 9)# random value generator  
 print("Word should be limit", rand\_num1)  
 user\_inp = input("enter now")# word from user  
 flag = 1# to check status for length of word  
 if len(user\_inp) != rand\_num1:  
 print("Enter only specified length for word")  
 exit()  
 else:  
 flag=1  
  
  
 # to length of word entered by user input  
 flag1=0  
  
 words = ['CABBAGE', 'Cabbage', 'cabbage', 'programs',  
 'python', 'player', 'condition',  
 'reversed', 'water', 'board', 'geeks']  
 if(flag==1):  
 for i in words:  
 if (user\_inp == i):  
 flag1=1  
 else:  
 print("Enter only dictionary word")  
 exit()  
 total\_1 = 0  
 time\_sec=5  
 if(flag1==1):  
 if(len(user\_inp)<6):  
 #user\_inp="Cabbage CABBAGE cabbage"  
 while time\_sec:  
 mins\_1, secs\_1 = divmod(time\_sec, 60)  
 timeformat\_1 = '{:02d}:{:02d}'.format(mins\_1, secs\_1)  
 print(timeformat\_1)  
 time.sleep(1)  
 time\_sec -= 1  
  
 print("Timed out ")  
 try:  
 if user\_inp.isalpha():  
 for X in user\_inp:  
 total\_1 = total\_1+score\_1[X.lower()]  
 else:  
 raise ValueError("Invalid Input .....Accept only Characters")  
 #return total\_1  
 except ValueError as e:  
 print("Information for User",e)  
 total\_1="No result"  
 return total\_1  
 else:  
 return total\_1  
  
def main():  
 *"""Print the latest tutorial from Real Python"""* tic = time.perf\_counter()  
 print("value is::::",scrabble\_score\_COUNT())  
 toc = time.perf\_counter()  
 print(f"Code is working in {toc - tic:0.4f} seconds")  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Program file 2: -**

Test Scrabble\_test.py

import time #import for timer  
import random #import for random selection

#IMPORT FOR GET AND SET INPUT AND OUTPUT IN bASE1.PY CLASS  
from base1 import get\_display\_output,set\_keyboard\_input

#IMPORTING NEW\_WORD\_LIST . PY TO GET IMPLEMENTED FOR TESTING UNDER TESTING ENVIRONMENT.

.  
import new\_word\_list as test1  
#testing module 1 for code  
def test\_scrabble\_score\_COUNT():  
  
#SETTING KEYBOARD WITH USER INPUTS  
 set\_keyboard\_input(["Cabbage","CABBAGE","CAbbage","cabbage"])  
 y=14

#calling module that needs to test  
 result\_1=test1.scrabble\_score\_COUNT()  
# saving output in to the RESULT2

RESULT2=get\_display\_output()  
 assert RESULT2  
 #,"Test case function for test\_scrabble\_score\_COUNT ... Passes  
  
def test\_scrabble\_score\_COUNT2():  
  
  
  
 y=14  
#calling module that needs to test

result\_1=test1.scrabble\_score\_COUNT()  
 assert result\_1 == y  
 #,"Test case function for test\_scrabble\_score\_COUNT ... Passes"  
def test\_scrabble\_score\_COUNT3():  
  
  
 expected=14  
#calling module that needs to test  
 result\_1=test1.scrabble\_score\_COUNT()  
 # assert would trace error or pass / failure status

assert result\_1  
 #,"Test case function for test\_scrabble\_score\_COUNT ... Passes"  
#test case 4 for module testing

def test\_scrabble\_score\_COUNT4():  
  
  
  
 y=14  
 result\_1=test1.scrabble\_score\_COUNT()  
 assert result\_1

**File for User Input and output: -**

**Base1.py**

import builtins  
  
input\_values\_1 = []  
print\_values\_1 = []  
  
  
def mock\_input\_1(s):  
 print\_values\_1.append(s)  
 return input\_values\_1.pop()  
  
#input type   
def mock\_input\_output\_start\_1():  
 global input\_values\_1, print\_values\_1  
  
 input\_values\_1 = []  
 print\_values\_1 = []  
#assigning the values  
 builtins.input = mock\_input\_1  
 builtins.print = lambda s: print\_values\_1.append(s)  
  
# display method  
def get\_display\_output():  
 global print\_values\_1  
 return print\_values\_1  
  
#setting user inputs  
def set\_keyboard\_input(mocked\_inputs\_1):  
 global input\_values\_1  
  
 mock\_input\_output\_start\_1()  
 input\_values\_1 = mocked\_inputs\_1

**Testing: -**

**TDD: -**TDD stands for Test Driven development in python. It is behalf on three basic concepts: -

compiling failure case, designing passing case

Refactoring cases are required mainly. It is mainly a test or module designed for specific module or programs to get test under testing environment either using unit testing tool or by pytest tool.

Installation of package is required to get implemented in testing environment.

**Automatic type testing: -**Automatic type means manual testing done by user itself by comparing different types of inputs and match the result with expected output itself. User would enter differ inputs to test each case. In each case, user would enter word for testing. Then match each one.

**Pytest module** is one of testing under TDD, which is supported by PyCharm community in two ways: -

1. Unit testing by using import unittest is one method.
2. Pytest testing by using import pytest is other method. pytest works for module that is included for testing.

**Implementation of TDD**: - Testing is implemented by pytest in the code by adding installation of pytest and pytest-xdist-2.4.0.

In the terminal, use pytest modulefile.py

For example: -file name is Scrabble\_test.py

So, Command would be: -

Pytest Scrabble\_test.py

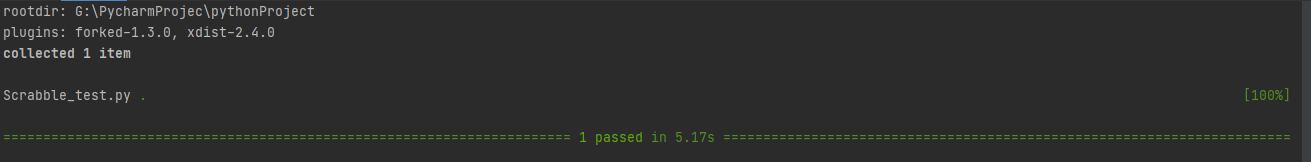
**Implementing Pytest module: -**

**Pytest Scrabble\_test.py**

When program is run on PyCharm Community Edition 2020.3.3 x64. Firstly, running program with show Test case as unit testing. Install Pytest package and pyest-xdist with versions from file option.When user clicks on file option,then go to settings and dialogue for setting would be opened.User needs to select Project interpreter option and then double click on pip .

Pip would take to new window with list of all packages or library available for python. Click on any library double times would start the installation of library or package with specified version under pycharm environment.

Running first case



(python) G:\PycharmProjec\pythonProject>pytest Scrabble\_test.py

====================================================================== test session starts ======================================================================

platform win32 -- Python 3.8.6, pytest-6.2.5, py-1.10.0, pluggy-1.0.0

rootdir: G:\PycharmProjec\pythonProject

plugins: forked-1.3.0, xdist-2.4.0

collected 4 items

Scrabble\_test.py ..FF [100%]

=========================================================================== FAILURES ============================================================================

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ test\_scrabble\_score\_COUNT3 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

def test\_scrabble\_score\_COUNT3():

expected=14

> result\_1=test1.scrabble\_score\_COUNT()

Scrabble\_test.py:29:

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

new\_word\_list.py:13: in scrabble\_score\_COUNT

user\_inp = input("Input: ") # prompt for input

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

s = 'Input: '

def mock\_input\_1(s):

print\_values\_1.append(s)

> return input\_values\_1.pop()

E IndexError: pop from empty list

base1.py:9: IndexError

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ test\_scrabble\_score\_COUNT4 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

def test\_scrabble\_score\_COUNT4():

y=14

> result\_1=test1.scrabble\_score\_COUNT()

Scrabble\_test.py:37:

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

new\_word\_list.py:13: in scrabble\_score\_COUNT

user\_inp = input("Input: ") # prompt for input

(python) G:\PycharmProjec\pythonProject>pytest Scrabble\_test.py

====================================================================== test session starts ======================================================================

platform win32 -- Python 3.8.6, pytest-6.2.5, py-1.10.0, pluggy-1.0.0

rootdir: G:\PycharmProjec\pythonProject

plugins: forked-1.3.0, xdist-2.4.0

collected 4 items

Scrabble\_test.py .... [100%]

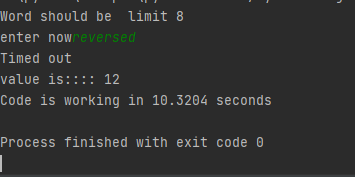
======================================================================= 4 passed in 4.17s ==============

**Implementing Automated/Manual type testing:**

Test case1:- Input as reversed

When “CABB3GE” is entered by user.

It will show test Failed with value of Failed.

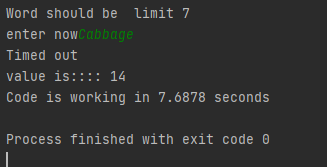


Test case1 :- Input as cabbage

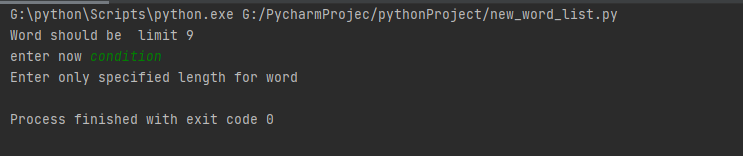
When “CABBAGE” is entered by user.

It will show test Failed with value of Failed.

**Output: -**



Test case1:- Input as “condition”

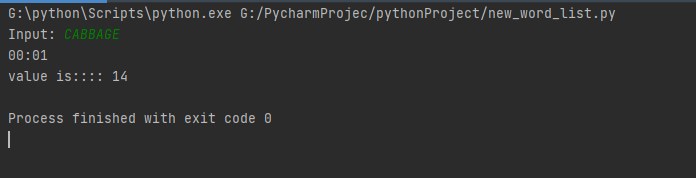


Test case2: - Input as “CABBAGE

When “CABBAGE” is entered by user.

It will show test successful with value of 14.

**Output**: -

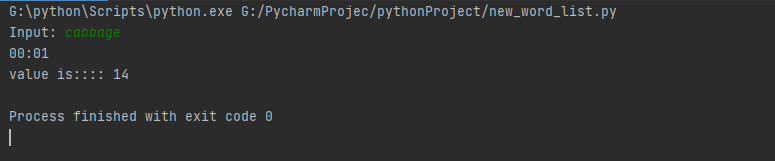


Test case3: - Input as “cabbage”

When “cabbage” is entered by user.

It will show test successful with value of 14.

**Output**: -



1. **Conclusion: -**

The project is a successful executed, it would complete all the mandatory activities and the output have been tested in proper way.

Github Link: <https://github.com/sagarmanandhar/Scrabble-Program>