

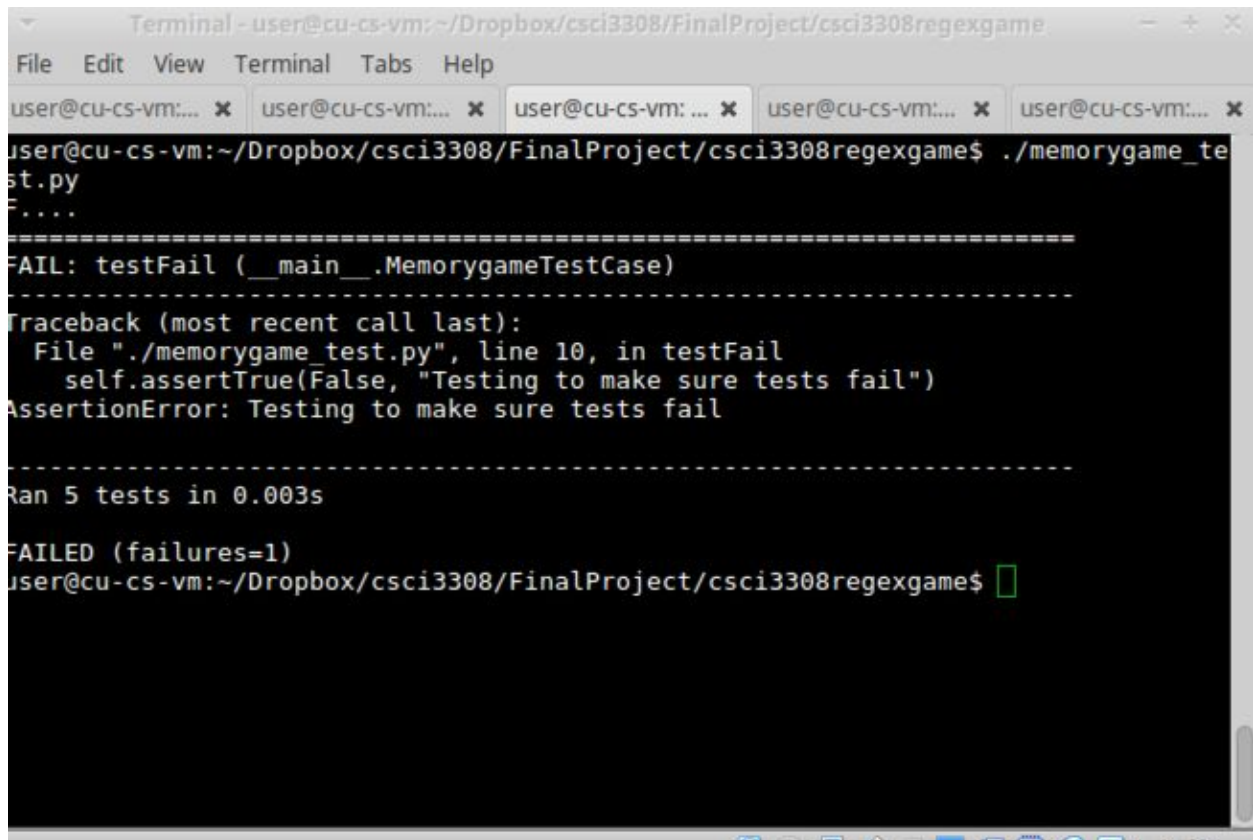
William Diment
Kathryn Gray
Eddie Stoian
Oscar Grandara

Regex Memory Game

Our vision is to create a game that will help students learn Regex in a fun and interactive manner.

We used python unit tests to test both our pygame and our terminal game.

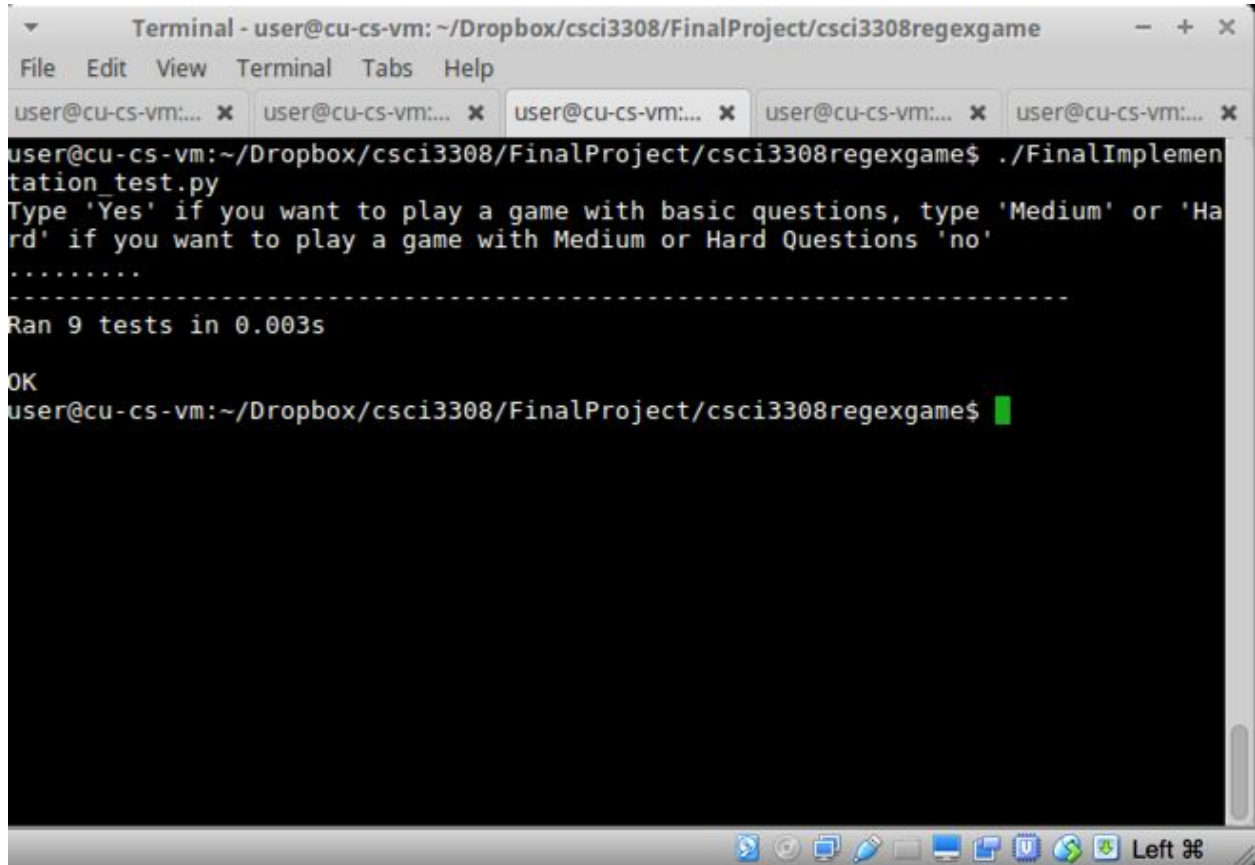
Here is the test of the pygame. The failure is intentional, to make sure the tests were working properly.



```
Terminal - user@cu-cs-vm: ~/Dropbox/csci3308/FinalProject/csci3308regexgame
File Edit View Terminal Tabs Help
user@cu-cs-vm:~/Dropbox/csci3308/FinalProject/csci3308regexgame$ ./memorygame_test.py
F....
=====
FAIL: testFail (__main__.MemorygameTestCase)
=====
Traceback (most recent call last):
  File "./memorygame_test.py", line 10, in testFail
    self.assertTrue(False, "Testing to make sure tests fail")
AssertionError: Testing to make sure tests fail
=====
Ran 5 tests in 0.003s

FAILED (failures=1)
user@cu-cs-vm:~/Dropbox/csci3308/FinalProject/csci3308regexgame$
```

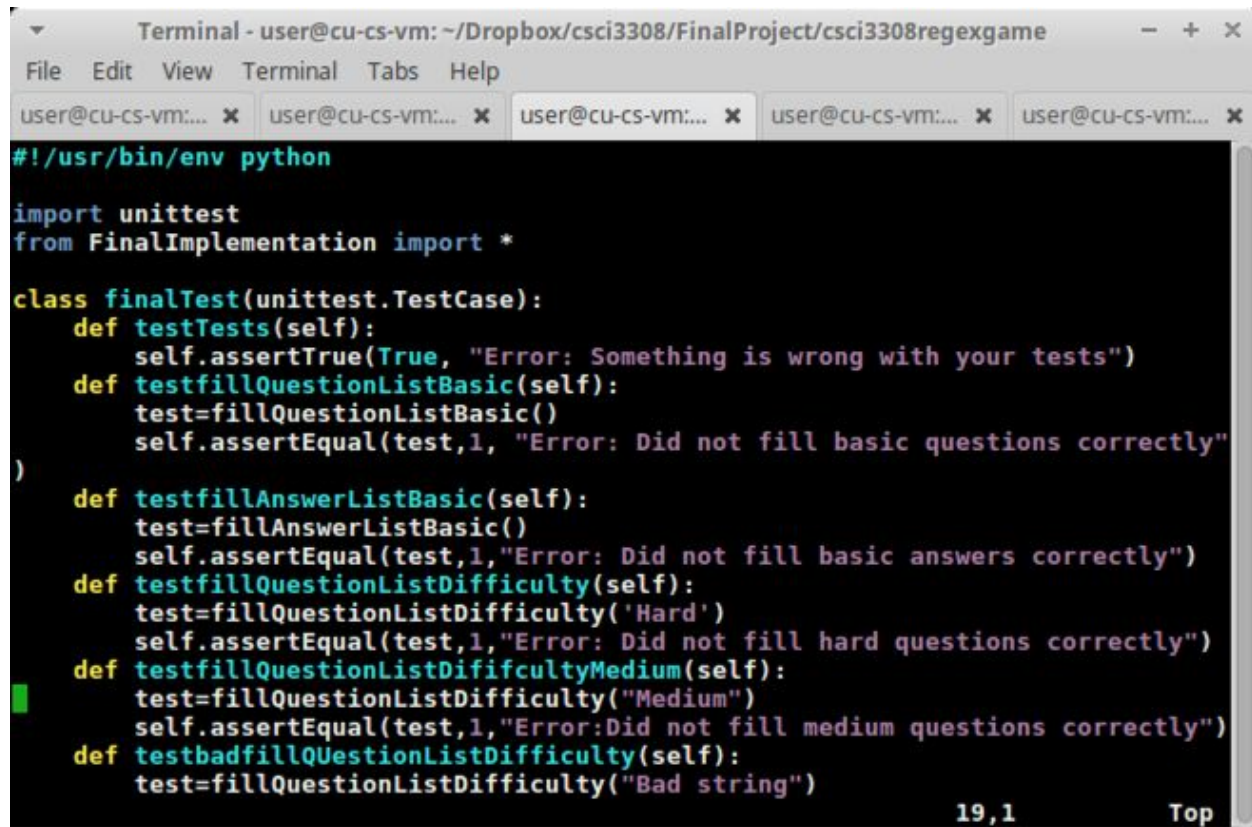
Here is the test of the terminal version. All the tests we ran passed.



```
Terminal - user@cu-cs-vm: ~/Dropbox/csci3308/FinalProject/csci3308regexgame
File Edit View Terminal Tabs Help
user@cu-cs-vm:~/Dropbox/csci3308/FinalProject/csci3308regexgame$ ./FinalImplementation_test.py
Type 'Yes' if you want to play a game with basic questions, type 'Medium' or 'Hard' if you want to play a game with Medium or Hard Questions 'no'
.....
-----
Ran 9 tests in 0.003s

OK
user@cu-cs-vm:~/Dropbox/csci3308/FinalProject/csci3308regexgame$
```

Here is an example of the tests we wrote and ran to determine whether the questions and answers were pulled from the database correctly.

A terminal window titled "Terminal - user@cu-cs-vm: ~/Dropbox/csci3308/FinalProject/csci3308regexgame" with a menu bar (File, Edit, View, Terminal, Tabs, Help) and five tabs. The code is a Python unittest class named finalTest. It includes imports for unittest and FinalImplementation. The class has several test methods: testTests (asserts True), testfillQuestionListBasic (calls fillQuestionListBasic and asserts 1), testfillAnswerListBasic (calls fillAnswerListBasic and asserts 1), testfillQuestionListDifficulty (calls fillQuestionListDifficulty with 'Hard' and asserts 1), testfillQuestionListDifficultyMedium (calls fillQuestionListDifficulty with 'Medium' and asserts 1), and testbadfillQuestionListDifficulty (calls fillQuestionListDifficulty with 'Bad string'). The bottom right of the terminal shows "19,1" and a "Top" link.

```
#!/usr/bin/env python

import unittest
from FinalImplementation import *

class finalTest(unittest.TestCase):
    def testTests(self):
        self.assertTrue(True, "Error: Something is wrong with your tests")
    def testfillQuestionListBasic(self):
        test=fillQuestionListBasic()
        self.assertEqual(test,1, "Error: Did not fill basic questions correctly")
    )
    def testfillAnswerListBasic(self):
        test=fillAnswerListBasic()
        self.assertEqual(test,1,"Error: Did not fill basic answers correctly")
    def testfillQuestionListDifficulty(self):
        test=fillQuestionListDifficulty('Hard')
        self.assertEqual(test,1,"Error: Did not fill hard questions correctly")
    def testfillQuestionListDifficultyMedium(self):
        test=fillQuestionListDifficulty("Medium")
        self.assertEqual(test,1,"Error:Did not fill medium questions correctly")
    def testbadfillQuestionListDifficulty(self):
        test=fillQuestionListDifficulty("Bad string")
```

19,1 Top