Top of Form



Bottom of Form

Top of Form



**Python 2: Getting More Out of Python  
Lesson 5, Project 1**

Handed in: 2 Mar 2015 06:36:21PM Graded: 3 Mar 2015 02:02:10PM

**Here are your instructions:**

Write a function (*not* a class) that takes two arguments, a string player name and an integer score, and keeps a "high score" table in a Python shelve. If the integer argument is higher than the given player's current high score (or if the player has no recorded high score), log the value as this player's new high score. The function should return the player's current high score. *Remember, a function is not the same thing as a class and it's a function that's needed.*

Again, write a separate test module that verifies the operation of the function.

**Your Comment:**

Back from the jungle, found some interesting things out there...

**Items Handed In**

* [Open Project Handed In](https://students.oreillyschool.com/student/project/?/.handin/160-7855-1/com.ost.jwoloson.160.7855.1.PersistentStorage_Homework.zip)

**Overall Comments:**

Hi Jason,

Welcome back from the jungle. I hope you brought back some real treasures.

This is a really solid first attempt. You've got some great tests and all the moving pieces are in place.

In your next draft of like you to take a look at a couple of things:

-The function is only supposed to take 2 arguments, the player name and the score.

-You want to avoid specifying pathnames because it virtually guarantees that your app will only run on your machine. Best practice here would be to use a temporary directory, switch into it, and stay on a "first name basis" with your filenames. All of the routines that deal with the file system, such as open (), assume the current directory if you do not otherwise specify another.

-You don't want do actual file operations in the test suite. Think of this as a circus with the main routine playing the juggler and the test routine serving as the stage crew. setUp() can sweep the stage, give it a fresh coat of paint, create directories, etc. Then, as the performer takes center ring, the tests can throw bowling pins to juggler. All of the performance (file opening, closing, etc.) should take place in the center ring, blissfully agnostic about whatever's happening back stage.

In fact, the test suite doesn't even have to know the name of the shelve file. Much better to keep it agnostic, or else you are setting yourself up for a maintenance nightmare in the future.

-The idea behind the tests is that they should beat the crap out of the main app. This nothing in the Project Description the prohibits multiple players, negative scores, etc., so you might as well throw some more junk at it.

- These tests look more difficult than necessary to write and easy to make mistakes with. If you cast them in this form:

def test\_somthing(self):

name, score, exp = ("Bree", 50, 50)

observed = high\_score(name, score)

self.assertEqual(observed, exp,

"I'm looking for: " + str(exp) +

" but got: " + str(observed))

... you're automatically populating the test and the error message with the data. Not so important for one test but what about the second?

In this format, it's easy to recycle code:

def test\_somthing\_else(self):

name, score, exp = ("Bree", 40, 50)

#recycled code

self.assertEqual(observed, exp,

"I'm looking for: " + str(exp) +

" but got: " + str(observed))

.... and it's not too tough to extend it to run any number of tests with very little typing/debugging:

def test\_a\_bunch(self):

name\_score\_exp = [('Bree', 50, 50), #new score

('Bree', 60, 60), #higher score

('Bree', -10, 60), #lower score

('Fred', 0, 0) #new score for new player

]

for name, score, exp in name\_score\_exp:

self.assertEqual(observed, exp,

"I'm looking for: " + str(exp) +

" but got: " + str(observed))

Overall, great work.

-Pat

**Grade:**

Try Again

Your instructor would like you to work on this project. Be sure to hand in your corrections!

[Take Me Back](ostreturn:)

© 2014, O'Reilly Media, Inc. All rights reserved.

Bottom of Form