Python Epic Battle Simulator: Welcome to Thunderdome

Lab/Project for 5001

I’d like you to finish up what we started in our class demonstration. When we left off you had two characters and you could make one character fight another. Let’s take it to the next level and add weapons.

**1. Goals**

* Explore class creation
* Practice existing programming skills by making a more advanced application
* Have fun

**2. In Recitation**

In recitation your most important goal is to make sure you have, and understand, the demo code presented in class. It’s the base for your lab assignment.

Part 1 : Group session work in groups to prepare and understand the base code provided in class.

Part 2 : Review the test class provided in this lab and discuss how you are going to make sure the tests pass.

Part 3 : Separate and begin independent work on the lab assignment.

Remember: Don’t share code after establishing the starting code presented in class.

**3. Instructions**

Instructions here are minimal as code was provided in class and I want you to reverse engineer the test class to establish some of the program details. You might have to change the code presented in class to have the tests pass. Don’t change the test class. If it is altered from what is provided below, an announcement will be made.

Weapon class:

Create a weapon class. A weapon's attributes should be name, strength, and durability. A weapon should have methods for attack and \_\_str\_\_ method that will allow for printing. In the end this class should exist in its own file. Create a nice constructor with default values.

Weapon’s attack method:

Attack should accept nothing but self and return a random number from 1 - the weapon’s strength unless durability is 0.

Each time an attack is used, durability is reduced by some amount. If durability is 0 when an attack is called, it should return 0 instead of an attack value. You can adjust strength and durability for each weapon.

Weapon Constructors:

Include a constructor that will set default values. Make sure your constructor will also allow you to create a weapon given all three attributes. For example this should work just fine:

*weapon1 = Weapon(“Sword of awesomeness”, 100, 2)*

Character adaptations:

Give characters the ability to have a weapon object. When a character attacks it calls the weapon’s attack and adds that value to its attack value before it attacks the opponent.

If the character has a weapon, make that part of the print statement as well as the character's attributes.

Tip: create a setWeapon method.

Driver:

Create a driver to run your epic battle simulator. Create 4 unique weapons and 4 characters. Put all of the weapons in a list of weapons and all of the characters in a list. (Alternatively, you could place the weapons in a dictionary and use the weapon name as a key.)

Print all the weapons and all of the characters in each list. For each character, ask your user to assign a weapon. Once a weapon has been assigned, remove it from the list of available weapons so you don’t have two characters with the same weapon.

Begin a game loop that will continue as long as there are more than 1 character in your character list.

Game loop:

Ask your user to select a challenger and a character to be challenged. Have them fight and then show the results. Remove the character from the list of available characters.

Tip: work smarter not harder. I would use the index of each character to set this up. For example,

*list\_characters[selection1].fight(list\_characters[selection2])*

Tip: for weapon and character selection just use numbered menus

Testing:

Use this test file and see how you did. Notice that a big part of your grade will come from being able to pass these tests. You’ll have to reverse engineer the test file to discover some requirements I left out above.

class CharacterTest(unittest.TestCase):

def test\_init\_char1(self):

c1 = Character()

self.assertEqual(c1.name, "Ward")

self.assertEqual(c1.hitPoints,5)

self.assertEqual(c1.strength,5)

def test\_init\_char2(self):

c1 = Character("Jane", 1, 2)

self.assertEqual(c1.name, "Jane")

self.assertEqual(c1.hitPoints,1)

self.assertEqual(c1.strength,2)

def test\_take\_damage(self):

c1 = Character()

c1.take\_damage(1)

self.assertEqual(c1.hitPoints,4)

def test\_take\_damage\_zero(self):

c1 = Character()

c1.take\_damage(25)

self.assertEqual(c1.hitPoints,0)

self.assertEqual(c1.alive, False)

def test\_init\_weapon(self):

w1 = Weapon()

self.assertEqual(w1.name, "generic dagger")

self.assertEqual(w1.strength, 1)

self.assertEqual(w1.durability,2)

def test\_give\_weapon(self):

c1 = Character()

w1 = Weapon()

c1.giveWeapon(w1)

self.assertEqual(c1.weapon.name, "generic dagger")

def test\_weapon\_durability(self):

w1 = Weapon("pool noodle", 1, 1)

w1.attack()

self.assertEqual(w1.attack(), 0)

Getting creative:

I love it when you get creative and enjoy your lab assignments. Feel free to embellish, alter, or add additional functionality to make your game more fun. Just take a quick look at the rubric beforehand to make sure you still satisfy the grading requirements. When in doubt, ask the TA in charge of grading your assignment.

Other requirements:

Separate all your code into separate files

Make sure each method is commented correctly (see video)

Make sure all code is neat and organized

**4. Extensions**

Remember you can get up to 26/30 points for finishing the lab, but for 30 points you have to go above and beyond. You don’t have to use any of the below extensions. They are just examples. Explore on your own and come up with something fun.

Extension ideas:

1. Add logical exception handling where appropriate.
2. Add other random effects with your weapon other than damage. For example, create a gun that has a 10% chance of backfiring and damaging the player. (You could let the user know this happened by returning a negative value.)

Ok

dagger 50% chance of critical attack

elven 50% chance to use fire magic

1. Add the ability for a character to randomly choose to heal instead of attack.

Ok

1. Redesign it to have a hero fight a set of monsters and let your user choose what the hero does. You could have options like: heal, regular attack, attack without weapon, super attack and skip a turn.
2. Add ASCII art or music.
3. Add additional creative elements not suggested here.

Type writer effect

Tip: if you want to add random events as suggested in the above extension ideas, just choose a random number from 1 - 100. Each number has an equal chance of occurring, so if you wanted a 10 % chance of something happening you could have that happen if it returns 1 - 10. If you wanted a 50% chance of something happening you could check to see if the number selected was 1 - 50 and so on.

**5. Report:**

Reflection:

What was the easiest and hardest part of this assignment?

What did you learn?

What grade would you give yourself?

Extension:

What extension did you add to the assignment?

**6. Submission:**

You should be submitting these at the least, but you may submit more files if you separated out your assignment for organizational purposes:

* Report09.pdf
* driver.py
* weapon.py
* character.py

Submit your project code on canvas as Lab9\_”Your\_name”.zip

There’s no need to submit the test file. We will use the one from within the assignment.

When you submit, double-check the following.

* Is your name and an appropriate header at the top of each Python file?
* Does every function have a comment or docstring specifying what it does?
* Does your report have all sections completed?
* Is your report a pdf document?

**7. Rubric:**

|  |  |  |
| --- | --- | --- |
|  | **Possible** | **Given** |
| Tests | | |
| test\_init\_char1 | 2 | 0 |
| test\_init\_char2 | 2 | 0 |
| test\_take\_damage | 2 | 0 |
| test\_take\_damage\_zero | 2 | 0 |
| test\_init\_weapon | 2 | 0 |
| def test\_give\_weapon | 2 | 0 |
| test\_weapon\_durability | 2 | 0 |
| Driver | | |
| Create 4 chars and 4 weapons | 1 | 0 |
| Assign weapons to chars | 1 | 0 |
| Weapons assignment done by user | 1 | 0 |
| Weapons removed once assigned | 1 | 0 |
| Battle commences | 1 | 0 |
| Misc | | |
| Code Quality | 4 | 0 |
| Report | 3 | 0 |
| Not included in total possible: | | |
| Extensions (Not calculated without report) | 4 | 0 |
| Creative or went above and beyond | 4 | 0 |
| Code does not compile | -30 | 0 |
| Late penalty | -6 | 0 |
| Not implemented as requested | -30 | 0 |
|  | |  |
| TOTAL POINTS POSSIBLE out of 30 | 26 | 0 |