Python Epic Battle Simulator: Welcome to Thunderdome

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.

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.

Weapon's attack:

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.

Note: if you want to get more creative with what the weapon returns as an attack value or how durability is handled, then I'm fine with that. The important thing is that the weapon uses durability and returns 0 if durability is 0.

Constructors:

Include a constructor that will set default values of: "Generic dagger", 1 strength, and 1 durability. 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)

Weapon test class:

The TAs should have helped you create a test file for your character class. Make sure to also create one for your weapon class. Test your constructors and make sure when durability goes to 0 your weapon's attack method will return 0.

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.

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])

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.

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.)
3. Add the ability for a character to randomly choose to heal instead of attack.
4. 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.
5. Add ASCII art or music.
6. Add additional creative elements not suggested here.

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.

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?

Rubric:

Total Points: 30

Weapon (~~8~~ 4 pts)
- Weapon constructors created
- Weapon constructor has default values
- Weapon attack function is as requested
- Weapon returns 0 if durability is 0

Weapon test class (2 pts)
- Constructors tested
- Attack method tested

Character adaptations(4 pts) Character can hold a weapon
- Character's attack value is adjusted based on equipped weapon

Driver (6 pts)
- User allowed to set weapons to characters
- User allowed to choose combatants
- Battle simulation works (3 pts)

Extensions - ~~4~~ 8 points

Code Quality - 4 points

Report - 2 points

Character class base functionality is there as setup in class: -100% without
All files exist in their own files: -4 code quality deduction without