Project 3

Overview: For this project, you will be adding additional functions to your simple battle game. The points breakdown is as follows:

- (15 pts) Create 20 characters with at least 4 different "classes" (classes can be whatever you want, but an example of classes would be warrior, mage, healer, rogue) and add them to your player dictionary. *You should do this by reading the data in from a .csv file, which we covered in class. **Note** that you can't easily read in a list for your attack range (for example, [0,6]). Instead, have one column in your csv for the range min, one for the range max, and then put them together when creating a dictionary.*
- (3 pts) Add to your player dictionaries an entry called "weapon".
- (2 pts) Add to your player dictionaries an entry called "armour"
- (5 pts) Create a dictionary of 10 weapons that each have their own ideal class (i.e. a wand that might be best used by a mage, a sword by a warrior...etc). Each weapon, should have a dictionary that has at least the **name of the weapon, damage range**, and **ideal class** of the weapon. E.g. {"name": "blah", "damage": [10,12], "ideal": "mage"}
- (5 pts) Create a dictionary of 10 pieces of armor. Each piece of armor should have a dictionary that has at least the **name, blocking range** (how many damage points it will block)
- (10 pts) After a user has selected their character, **ask them to choose a weapon**. You must print out a string that describes each weapon. Be creative with this! Make sure you tell them what class is best with the weapon. **Store the weapon dict in your player dictionary.** Do NOT just print out the dictionary, you need to create and format a string by pulling information from the dictionary. DO NOT hardcode description strings.
- (5 pts) Also **ask them to choose a piece of armor**, also printing out a string (NOT the dict) to describe each piece of armor. **Store armor in your player dictionary**.
- (25 pts) When in "battle" **the basic attack no longer does just [0,6] damage, instead we get the damage range from the damage range of the weapon they choose**. If they choose a weapon that is not made for their class, **the weapon does 1 point less damage per hit** (still generate a random number for damage, but if their class does not match the weapon's, subtract 1 before inflicting damage. REMEMBER that you don't want to do -1 damage, so be careful of your numbers…) *Special powers are NOT affected by the weapon at all, they do the regular amount of damage, do not subtract 1.*
- (15 pts) Speaking of special powers, let's make ours more fun. Each special power is now a little riskier. When you use your special power, this function will choose a random integer between 0 and 10. If the integer is less than or equal to 2, heal the attackee 2 points instead of using the special attack. This STILL counts as using your special power. **Print** a message to tell them if they healed their opponent or damaged them.
- (15 pts) Alter your current game so that each time someone attacks, the attackee **has a 10% chance of their armor blocking some of the damage** (this means about 10%

of the time, it will block some of the damage, the other 90% of the time it will block 0 damage). The amount of damage blocked should be randomly selected from the range of the armour. Create a function called **blocking()** that calculates and returns how much damage is blocked. You can do this by randomly choosing from the list [0,1, 1, 1, 1, 1, 1, 1, 1, 1] and doing one thing if you pick 0, and the other if you pick a 1.

- (5 pts) Extra Credit: Use Classes to implement things in this game. You should have a class **for each of the** classes of character you have. Instead of storing the character choice of the player and opponent as a dictionary, use these values to set class attributes.
    - The classes should have at least an **__init__()** method that takes the values from a character dictionary and assigns them as attributes, a **set_weapon()** method and **set_armor()** method that adds armor/weapons to their attributes, an **attack()** method that takes the other player as an argument and works similarly to your player turn function. If you do the extra credit, you may have more questions, come ask me or shoot me an email.
        - https://www.learnpython.org/en/Classes_and_Objects
        - https://jeffknupp.com/blog/2014/06/18/improve-your-python-python-classes-and-object-oriented-programming/
        - https://www.codecademy.com/courses/learn-python/lessons/introduction-to-classes/exercises/why-use-classes?action=lesson_resume
        - https://www.programiz.com/python-programming/class