Project 2

Using the python methods we've learned so far, create a game that can be played by the user. Choose one option below, or email me if you have another idea you want to do. Make sure to utilize things we've learned in class like **functions**, **lists**, and **dictionaries** as needed to make your code simple and elegant.

Simple battle game:

- (13 points) Create a dict of 4 characters that a user can choose to play. These character dictionaries should have *at least* a **name, age, health, class** (as in mage, wizard, rogue...whatever), and a **"special power" dictionary** that has *at least* the **name** of the power, the **range** of attack points it can have, and how many **turns** the player has to wait to use it again (aka, cooldown). The dictionary object should look like this, but with your values: **{'name': 'Chelsea, 'age': 10, 'range': [6,12], 'class': 'mage', 'health': 100, 'SP': {name: 'smash', 'range': [5,10], 'cooldown': 4}**
- (5 points) Let the user **choose** which character to play by typing in the name of the character they'd like to play. Print a message telling them their name, age, class, and a STRING that describes their special power. DO NOT just print out the dictionary object. You need to create a string by pulling information from each dictionary, do NOT hardcode the strings.
- (5 points) Randomly choose one of the *remaining* characters using the random package from python. Tell them who their opponent is
- (2 points) For each turn, Ask the user whether they want to do a basic attack or use their special power (use something simple to type, like using 1 and 0, or 'ba' and 'sp')
- (18 points) Create a function called **player_turn()** that takes an attacker dictionary (the one attacking), and the attackee dictionary (the one being attacked), and an argument **attack_special** that is either *True* or *False* as arguments. *This function should run exactly ONE turn (either yours or the opponents depending on the arguments)*
- (7 points) Inside this function, if the player uses their special attack, check that it is not on "cooldown" (aka they haven't used it in the past *n* turns where *n* is the cooldown value from their special character dictionary)
- (5 points) If they want to use their special power and they're allowed to, use the python *random* package to choose the number of damage points this attack does using the range of damage from their special power dict.
- (3 points) If they don't want to or cannot use their special attack, then have them do a regular attack that has a range of [0,6]
- (2 points) Print a message that says they've done *x* amount of damage this round, and subtract that number of health points from the attackee and print the current health of the player to the screen.
- (10 points) Repeat this process (the above 4 points) for the player they're playing against, but randomly choose whether to do special or regular attacks using python's random package (hint: remember 0 = False, 1=True)

- (10 points) If at **any** point either player's health is < 0, that person loses, and the other wins! Create a function called **check_health()** that takes a players dictionary as an argument and checks if their health is too low, if it is, choose a random string message (something like “oh no, you’re dead” from a list you previously defined that has 6 options of death messages as strings, and print that string to the screen. (If your player wins, print “you win”) Then end the game.
- **EXTRA CREDIT (3 points):** Create a function called **petAttack()** that has a pet character that helps the character your player chooses by attacking the enemy with a regular attack with range[0,2] every other turn (a turn is player A goes and then player B goes). Print a message that your pet has gotten an energy boost and does x points damage to the opponent (x is the number of damage it does).

Note: the python package *random* might help: https://docs.python.org/2/library/random.html