Project 4:

Text-Based Role-Playing Game

Goal:

Create a turn-based RPG battle game where:

- The player and the enemy take turns attacking each other.
- Each attack reduces health, and the game ends when one character's health reaches zero.
- The game uses functions to organize the code, making it easier to manage and reuse.

Game Requirements:

1. The game should start with a welcome message and instructions.
2. The player and the enemy each start with 100 health points.
3. The game should include at least the following functions:
    - display_status() – displays the current health of both the player and the enemy.
    - player_attack() – calculates and applies damage to the enemy.
    - enemy_attack() – calculates and applies damage to the player.
    - check_victory() – checks if either the player or the enemy has won and ends the game if health reaches zero.
4. Use random damage values for attacks to make each round unpredictable.
5. The game loop should continue until the player or the enemy's health is zero.

Instructions:

Create a New Python File:

- In your "week5" folder, create a new folder named "project4".
- Open your "project4" folder and create a Python file named "rpg.py".

Add an Intro Message:

- Let's make our game a bit more fun by starting with a welcome message for the player.
- Use the print( ) function to create a message at the top of your file.

Set Up the Player and Enemy Health:

- Create variables to store the player's and enemy's health each starting at 100.
  Example:

```
# Health values
player_health = 100
enemy_health = 100
```

Import the Random Module:

- Import the random module to generate random damage values (Look at project 2 for help).

Define your Functions():

- Define the display_status() Function
  - Create a function to display the current health of both the player and the enemy.
    Example:

```
# Display health status
def display_status():
    # Print the health values for the player and the enemy
```

- Define the player_attack() Function
  - Create a function that calculates and applies a random amount of damage to the enemy's health.
  - Using "global enemy_health" allows the function to modify the enemy_health variable outside the function.
  - Use random.randint(10, 20) to determine the damage dealt in each attack.
    Example:

```
# Player's attack function
def player_attack():
    global enemy_health
    damage = random.randint(10, 20)
```

- Define the enemy_attack() Function
  - Create a function similar to player_attack() but for the enemy's attack on the player.

- This function works like player_attack() but reduces the player's health instead of the enemy's.

- Define the check_victory() Function
  - Create a function to check if the game is over and display the winner.
  - This function checks if either the player or the enemy's health is zero or below. If so, it prints a victory or defeat message and returns True, signaling the game should end.

  Example:

```
# Check if the game is over
def check_victory():
    # if player health is 0 or less, player loses
    # else if enemy health is 0 or less, player wins
    # else game continues
```

Create the Game Loop:

Build the Loop with a game_run Boolean:

- Set up a game_run boolean variable to control the game loop. Inside the loop, display the status, call the attack functions, and check for victory after each turn.
- This loop continues as long as game_run is **True**. On each iteration:
  - The player's and enemy's health are displayed.
  - The player is prompted to attack or quit.
  - After each attack, check_victory() is called to see if the game should end.

Display Status:

- Inside the loop, the first thing we want to do is print the health for the player and enemy each iteration of the loop.
- Call the display_status() function in your loop.

Ask for Player Input:

- After printing the display status, prompt the player to enter 'a' or 'q' to either attack or to quit the game.
- Use the .lower() function to convert the user input to a lowercase string.

```
action = input("Press 'a' to attack or 'q' to quit: ").lower()
```

Check for Victory after Player's Attack:

- If the check_victory() function returns true end the game.
- Create a conditional statement that checks if check_victory() is true.
    - If true set game_run to false to end the game.
    - If false continue with loop.

Enemy Turn:

- Do a function call for the enemy_attack() function.

Check for Victory after Enemy's Attack:

- If the check_victory() function returns true end the game.
- Create a conditional statement that checks if check_victory() is true.
    - If true set game_run to false to end the game.
    - If false continue with loop.

Run Your Game:

- Save your file and run it from the terminal. Remember to check if you're in the right folder, you should be in your project4 folder.
- Run your game by typing "python rpg.py".
- Test your game and see how many times you can win.

Bonus:

- **Special Moves**: Add a special move option for the player that deals extra damage but can only be used once.
- **Healing Potions**: Add a healing function that allows the player to regain health by a random amount (e.g., between 5 and 15).
- **Difficulty Levels**: Add difficulty settings that adjust the damage range for the player and enemy.