Goblin Battle Game Worksheet

Welcome to your coding challenge! In this worksheet, you'll create a Goblin Battle Game in Python. Follow the steps below to build your game incrementally. You'll use loops, conditionals, functions, classes, and randomization to create an interactive and fun experience.

Step 1: Introduction and Game Setup

Start by creating a simple welcome message and setting up initial health values for the player and goblin.

1. Create a print() statement to welcome the player.

2. Set up variables for player_health and goblin_health.

Here's the code:

```
import random


# Welcome message

print("Welcome to the Goblin Battle Game!")

print("Fight the goblin and survive to win the game!")


# Initial health values

player_health = 100

goblin_health = 50
```

Step 2: Create a Simple Attack Mechanism

Add a basic attack system where the player can attack the goblin, reducing its health.

1. Use a while loop to keep the game running.

2. Subtract damage from goblin_health using random attack power.

Here's the code:

```
while player_health > 0 and goblin_health > 0:
    print("\nYour Health:", player_health)
    print("Goblin's Health:", goblin_health)

    # Player's attack
    attack_power = random.randint(10, 20)  # Random attack power
    goblin_health -= attack_power
    print(f"You attacked the goblin for {attack_power} damage!")

    # Check if goblin is defeated
    if goblin_health <= 0:
        print("You defeated the goblin! You win!")
        break
```

Step 3: Add the Goblin's Turn

Make the goblin fight back! Alternate turns between the player and the goblin in the loop.

1. Add a goblin attack after the player's turn.

2. Use random numbers for the goblin's attack power.

Here's the code:

```
# Goblin's attack

    goblin_attack_power = random.randint(5, 15)

    player_health -= goblin_attack_power

    print(f"The goblin attacked you for {goblin_attack_power} damage!")


    # Check if player is defeated

    if player_health <= 0:

        print("You were defeated by the goblin! Game over.")

        break
```

Step 4: Add Healing for the Player

Introduce healing as an action the player can take. Use input() to let the player choose between attacking or healing.

1. Add a menu to choose actions.

2. Implement a healing mechanic to increase player_health.

Here's the code:

```
# Player's turn

    print("Choose your action:")

    print("1. Attack")

    print("2. Heal")

    action = input("> ")



    if action == "1":

        attack_power = random.randint(10, 20)

        goblin_health -= attack_power

        print(f"You attacked the goblin for {attack_power} damage!")

    elif action == "2":

        heal_amount = random.randint(10, 20)

        player_health += heal_amount

        print(f"You healed yourself for {heal_amount} HP!")

    else:

        print("Invalid action!")

        continue
```

Step 5: Use Classes for the Player and Goblin

Refactor the code to use classes for better organization. Define a Character class with methods for attack() and heal().

1. Create a Character class with attributes like name, health, and attack_power.
2. Add methods for attacking and healing.

Here's the code:

```
class Character:
    def __init__(self, name, health, attack_power):
        self.name = name
        self.health = health
        self.attack_power = attack_power

    def attack(self, opponent):
        damage = random.randint(1, self.attack_power)
        opponent.health -= damage
        return damage

    def heal(self):
        heal_amount = random.randint(5, 15)
        self.health += heal_amount
        return heal_amount

# Create player and goblin objects
player = Character("Player", 100, 20)
```

```
goblin = Character("Goblin", 50, 15)
```

Step 6: Integrate Classes into the Game Loop

Use your Character class to replace the old variables and functions. Update the game loop to call the methods for attacking and healing.

1. Create player and goblin objects.
2. Refactor the loop to use the new methods.

Here's the code:

```
while player.health > 0 and goblin.health > 0:
    print(f"\n{player.name}'s Health: {player.health}")
    print(f"{goblin.name}'s Health: {goblin.health}")

    # Player's turn
    print("Choose your action:")
    print("1. Attack")
    print("2. Heal")
    action = input("> ")

    if action == "1":
        damage = player.attack(goblin)
        print(f"You attacked the goblin for {damage} damage!")
    elif action == "2":
        heal = player.heal()
        print(f"You healed yourself for {heal} HP!")
    else:
        print("Invalid action!")
```

```
    continue


# Check if goblin is defeated
if goblin.health <= 0:
    print("You defeated the goblin! You win!")
    break


# Goblin's turn
goblin_action = random.choice(["attack", "heal"])
if goblin_action == "attack":
    damage = goblin.attack(player)
    print(f"The goblin attacked you for {damage} damage!")
elif goblin_action == "heal":
    heal = goblin.heal()
    print(f"The goblin healed itself for {heal} HP!")


# Check if player is defeated
if player.health <= 0:
    print("You were defeated by the goblin! Game over.")
    break
```