Project 2: "Hero vs. Monsters" Game

Objective:
Design a text-based game where a hero challenges various monsters, each having a unique vulnerability. This project aims to hone your understanding of Python basics, functions, lists, dictionaries, control structures, and randomization techniques.

Overview:
In the dynamic world of "Hero vs. Monsters", players assume the role of a valiant hero faced with the challenge of battling a series of menacing monsters. Each monster possesses a distinct vulnerability, demanding players to strategically select from their randomized arsenal of weapons and resources. As the journey unfolds, players must grapple with the unpredictability of their encounters, with the sequence of monsters encountered being entirely random. Success hinges on choosing the right weapon amongst the hero's dwindling resources. With an engaging text-based interface, this game offers an enthralling blend of strategy, unpredictability, and the classic tale of heroism against overwhelming odds.

Mechanics Overview:
The mechanics of "Hero vs. Monsters" are built upon strategy and resource management. At the start, players are provided with a randomized arsenal, each item having a specific quantity. As the hero encounters monsters in a shuffled order, players must decipher the monster's weakness and select the appropriate item from their arsenal to ensure victory. With each encounter, the chosen item's quantity diminishes, introducing the challenge of resource conservation. An incorrect choice spells defeat for our hero. The game's flow, governed by these mechanics, creates a balance between forethought, adaptability, and swift decision-making, pushing players to be both tactical and resourceful.

Game Structure:

Introduction:
- Display a brief intro message.

Initialize Arsenal:
- Randomly assign starting quantities for each resource in the format: [int_sword, int_arrow, int_magic, int_hammer]. (E.g., Using Python's random.randint()).
- Show the player their starting arsenal.

Random Monster Encounters:

- Monsters and their weaknesses are provided in the following dictionary:

```
monsters = {
    "Dragon": "int_arrow",
    "Zombie": "int_sword",
    "Ghost": "int_magic",
    "Minotaur": "int_hammer"
}
```

- Shuffle the order of monsters from the monsters dictionary to ensure random encounters.
- For each shuffled monster:
    - Display its name.
    - Offer a choice of weapons or actions.
    - React according to the hero's choice:
    - If correct, provide a victory message and proceed to the next monster.
    - If wrong, display a defeat message and conclude the game.

Resource Management:

- After each encounter, update and showcase the hero's remaining resources.
- If a resource is exhausted, notify the player. A game loss occurs if the hero attacks with the wrong attack.

Conclusion:

- The game runs continuously until the hero loses to a monster.
- Display an appropriate game over massage and the number of monsters defeated.

Requirements:

User Interface:

- Design a simple text-based interface to present the hero's available resources, the approaching monster, and options to select an action.

- Offer clear messages when the hero defeats a monster, faces defeat, or depletes a resource.

Game Mechanics:

- As the game progresses, the hero confronts monsters one by one. For each monster, the hero must decide on an action, drawing from available resources.

- The game concludes either when the hero is defeated (wrong action against a monster.

Possible Bonus Options:

- Add monsters and Attacks.
- Add health, and the hero takes damage, possibly a random amount.
- Add a way to gain attack quantity, maybe a treasure, or shop or money system.
- Anything else you can think of.

Submission:
Submit the Python code file via canvas
No Late submissions

"Hero vs. Monsters" Game Rubric (100 points)

Game Functionality and User Experience (50 points)

Monster Encounters & Arsenal Management (30 points)

- Correct implementation of randomized monster encounters.
- Proper deduction of resources after each encounter.
- Game handles outcomes based on the hero's choices accurately.

User Interface (20 points)

- An intuitive and easy-to-navigate text-based interface.
- Clear feedback provided to the player regarding victories, defeats, and resource statuses.

Code Organization, Readability, and Comments (40 points)

Organization (15 points)

- Logical code structure.
- Appropriate segregation between game logic and user interface.

Readability (15 points)

- Consistent naming conventions.
- Proper whitespace and indentation use.

Comments (10 points)

- Essential code sections are commented on.
- Comments are succinct and informative.
- Defend any techniques that are used outside the scope of this class

Testing (10 points)

- Well-defined test cases that consider both common gameplay and edge cases.
- Test cases validate game functionality.
- Annotations explaining each test case's objective and anticipated outcome.

Bonus Features (if implemented) (10 points)

Additional Game Mechanics (5 points)

- Integration of features beyond the standard requirements, such as new monsters, weapons, or health and money.

Innovative Approach (5 points)

- Creative solutions or unique enhancements that enrich gameplay or user interaction.