Textual RPG

Loops - Extra

1. Move Player

- **Task:** Write a function that moves a player in a given direction on a 2D grid.
- **Inputs:** A 2D grid representing the game world, the player's current position as an object `{x, y}`, and a direction string (e.g., 'north', 'south', 'east', 'west').
- **Example:**
  - Input: `[[0, 0, 0], [0, 1, 0], [0, 0, 0]]`, `{x: 1, y: 0}`, 'north'
  - Output: `{x: 1, y: 1}`
- **Tip:** Depending on the direction input, adjust the player's `x` or `y` position by 1. Ensure that the player doesn't move outside the boundaries of the game world grid.

2. Battle Simulation

- **Task:** Write a function that simulates a battle between the player and an enemy.
- **Inputs:** The player's current HP and attack power, and the enemy's HP and attack power. Assume that the player always attacks first, and the characters take turns to attack each other.
- **Example:**
  - Input: `playerHP: 100, playerAttack: 20, enemyHP: 50, enemyAttack: 15`
  - Output: `'Player wins'`
- **Tip:** Use a loop to simulate the battle. In each iteration, decrease the enemy's HP by the player's attack power, and then the player's HP by the enemy's attack power. The battle ends when the HP of one character reaches 0.

3. Find Enemies

- **Task:** Write a function that finds and returns all enemies in the game world.
- **Inputs:** A 2D grid representing the game world, where 0 is empty space, 1 is the player, and 2 is an enemy.
- **Example:**
  - Input: `[[0, 2, 0], [1, 0, 0], [0, 2, 0]]`
  - Output: `[{x: 0, y: 1}, {x: 2, y: 1}]`

- **Tip:** Use a nested loop to traverse the game world grid. Whenever a cell contains a '2', add the coordinates to your list of enemies.

4. Collect Item

- **Task:** Write a function that allows a player to collect an item at a given position.
- **Inputs:** The player's current position, and the item's position.
- **Example:**
  - Input: playerPos: `{x: 1, y: 1}`, itemPos: `{x: 0, y: 2}`
  - Output: `'Item collected'` or `'Item out of reach'`
- **Tip:** If the player's position matches the item's position, remove the item from the game world grid and return 'Item collected'. Otherwise, return 'Item out of reach'.

5. Find Path to Goal

- **Task:** Write a function that finds the shortest path from the player's current position to a goal. The function should simulate a simple search algorithm and return the sequence of moves that the player should make to reach the goal.
- **Inputs:** Two objects representing the player's current position and the goal's position. Each object contains `x` and `y` properties that denote the position on a 2D plane.
- **Example:**
  - Input: playerPosition: `{ x: 1, y: 1 }`, goalPosition: `{ x: 0, y: 2 }`
  - Output: `[ 'west', 'south' ]`
- **Tip:** Use while loops to iteratively move the player towards the goal, pushing the corresponding direction ('east', 'west', 'south', or 'north') to the path array. Repeat until the player's position matches the goal's position.