**Code Of Duty Documentaion**

**Player Class**

**The Player class represents a player in the game. Each player has a name, health, attack power, defense power, and the type of weapon they use.**

**Methods**

* **\_\_init\_\_(self, name, health, attack, defense, weapon\_type): Initializes a player with their name, health, attack power, defense power, and weapon type.**
* **take\_damage(self, damage): Reduces the player's health by a certain amount when they get hit by an attack.**
* **is\_alive(self): Checks if the player is still alive by checking if their health is greater than 0. If they are alive, it returns their current health.**

**Weapon Class**

**The Weapon class represents a weapon that can be used by players. Each weapon has a name, damage power, and the type of weapon it is.**

**Methods**

* **\_\_init\_\_(self, name, damage, weapon\_type): Initializes a weapon with its name, damage power, and weapon type.**

**Game Class**

**The Game class represents the game itself. It handles the gameplay by setting up the players, calculating the distance between them, and running the game loop.**

**Methods**

* **set\_players(self, player1, player2): Sets the two players for the game.**
* **calculate\_distance(self): Calculates a random distance between the players.**
* **play(self): Runs the main game loop. It displays player information, asks the current player for an action, and determines the outcome of each turn.**

**Global Variables**

**The code includes two global variables: melee\_weapons and ranged\_weapons. These are lists that contain the available melee and ranged weapons, respectively.**

**Execution Flow**

1. **The code asks for the names of Player 1 and Player 2 and creates instances of the Player class for them.**
2. **An instance of the Game class is created.**
3. **The set\_players method of the Game class is called to set the player instances.**
4. **The play method of the Game class is called to start the game.**
5. **During each round of the game, the code displays the players' information, calculates the distance between them, and determines the current player.**
6. **The current player is asked to choose an action: either attack or defend.**
7. **If the player chooses to attack, they are asked to choose a weapon based on its type.**
8. **The damage dealt is calculated based on the action and weapon chosen.**
9. **The defender takes the damage, and their health is updated.**
10. **If the defender's health drops to 0 or below, the game ends, and the code displays the winner.**
11. **Once the game loop ends, the code displays the "Game Over!" message.**