dungeaondudes

Column	
Grade	
Language	python

Project Requirements

Overview

Objective is to create a simple role-playing game that includes `Heroes`, `Monsters`, `Treasures`, and a `Quest or Adventure`

Monsters with more dice are stronger

Requirements Checklist

Project Requirements

- [x] ~~Put test cases in /test~~
- [x] ~~Write up in `/doc`~~
- [x] ~~Write a test plan in `/doc`~~
- [x] ~~Program must run on the class VM~~
- [x] ~~Design plan should be placed in `/doc`~~
- [ ]

Bonus

- [x] ~~[7 pts] Read from `data/.dd_monsters` to create monsters~~
- [x] ~~[4 pts] Unittests~~
- [x] ~~[4 pts] Run away~~

Code Requirements

- [x] ~~Project is named `dungeon_dudes`~~
- [x] ~~The name of main should be `dungeon_dudes.py`~~
- [ ] Does the program compile with `python3 compileall .`
- [x] ~~Does invalid input or choices make the program crash?~~
- [x] ~~Every battle round, the program must display:~~
  - [x] ~~Monster's Name~~
  - [x] ~~Hero's Health~~
  - [x] ~~Monster's Health~~
  - [x] ~~Menu of possible actions (See battle)~~
- [x] ~~Program must display the following every non-battle round:~~
  - [x] ~~Hero's Health~~
  - [x] ~~Loot the Floor (if possible)~~
  - [x] ~~Menu of possible actions~~
  - [x] ~~Quit~~
- [x] ~~Program should support `--dice` which prints out the dice rolls~~

- [x] ~~When hero dies, printout the list of hero loot and exit~~
- [x] ~~Loot should be given based on `(# of monsters + monster's health points) * 10)`~~
- [x] ~~At least 5 different monsters must be implemented~~
- [x] ~~At least 5 different kinds of loot must be implemented~~
- [x] ~~Hero starts the game with 10 health~~
- [x] ~~NEED TO HAVE A SHBANG~~

Brainstorm and Research

- We need to create `Heroes` `Monsters Treatures, and a Quest/Adventure`
- As the hero walks through the adventure, they will go through "rooms" which can be a `room` `cave` or `glen`
- Every time the hero goes into a new room, the room must be described
- In each location the Hero is met with monster(s) that they need to fight
- The entity with the highest initiative
    - The decision to go first is based on a `d20` roll
- Combat: The hero will roll 3 6d, same with the monsters if they have that many
- If the attackers highest die exceeds the defender's highest die then the attack is a hit
- Monsters may take from 1 to 3 hits to be killed
- Loot may be dropped by monsters when they are killed and the user had the option of looting it

loot:

consumable:

attack_potion

increase number of combat dice

best_of_odds

increase the number of faces the combat dice have

health_potion

increase health by x amount

heavy_hand

increases damage by 2

pierce_shot

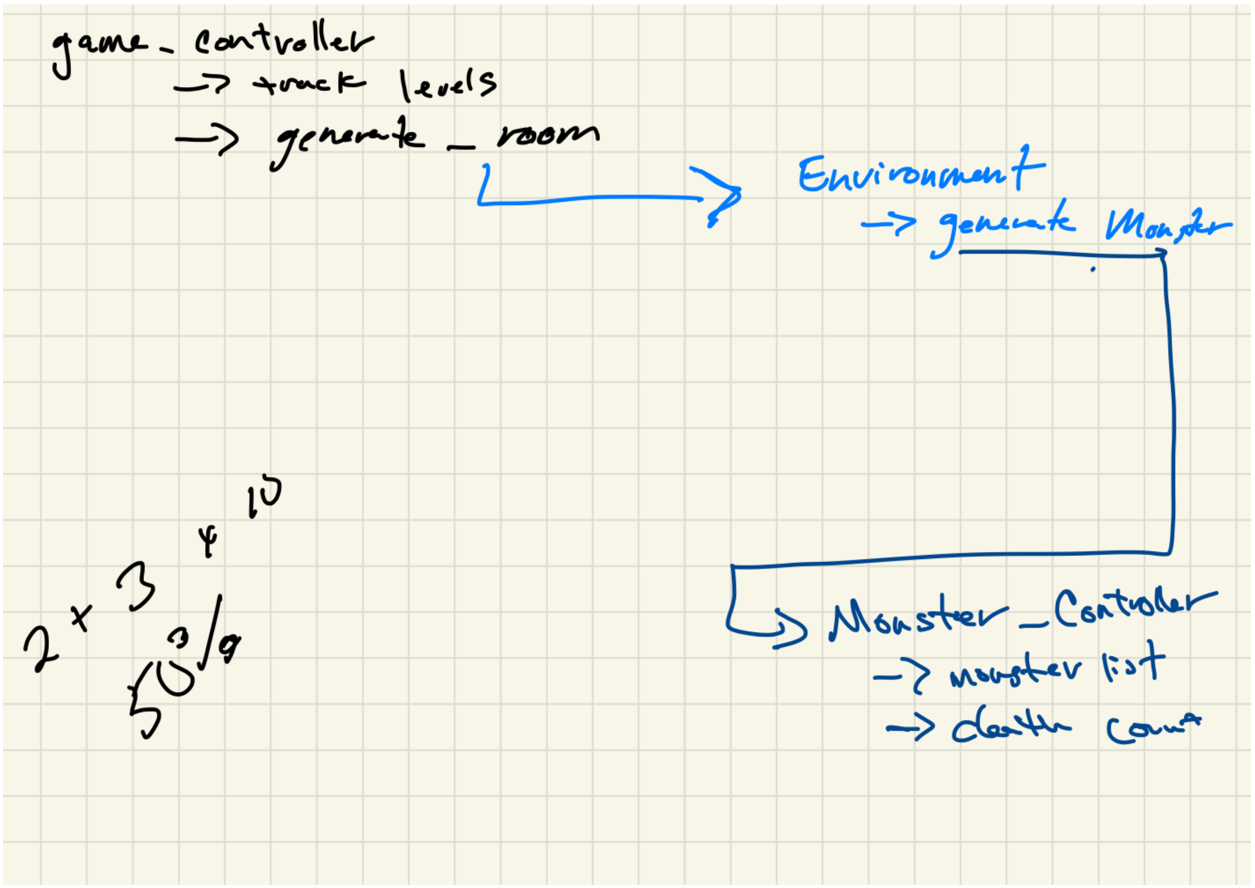
gurantee a hit

Terminology:

A battle is a confrontation between hero and monsters from start to finish.

A round is a complete phase where hero does a action and all monsters have completed one action

A duel is the individual combat between a hero and one monster. If there is two monsters in a round then each one gets a matchup with the hero



Adventure

The adventure will be an over arching concept containing all the things needed to run the game. With in it will be a list of `environments` that the hero will walk through.

The adventure will keep track of how many rooms have been completed and increase the difficulty as it goes. It will also contain the actual while loop until the character exits

Environments

Each environment will be the actual room that the hero is going to walk through and will contain conduct the actual fighting

```
-> env_descrioption
-> monsters: List[Monster]
-> loot_list: appended when a monster dies
-> initiative roll
-> hero_var
-> monters_var
```

Context Menus

Main Menu

The main menu will be handled by the `main()` function it will be a mechanism for entering their username or quitting

```
    ___                                          ___           _
   /   \_   _ _ __   __ _  ___  ___  _ __      /   \_   _  __| | ___  ___
  / /\ / | | | '_ \ / _` |/ _ \/ _ \| '_ \    / /\ / | | |/ _` |/ _ \/ __|
 / /_//| |_| | | | | (_| |  __/ (_) | | | |  / /_//| |_| | (_| |  __/\__ \
/___,'  \__,_|_| |_|\__, |\___|\___/|_| |_| /___,'  \__,_|\__,_|\___||___/
                    |___/
1) Start an Adventure
q) Quit
```

Executed at "start" or "next room"

```
Loading Room...
---------------------------------------
$hero.name walks to the next room and...
<desc>

$env.monster_count monstesr apear!
What will you do?
---------------------------------------
```

Adventure Menu — No Combat

In here the game will initialize with game data and put the user in their first dungeon that is generated based on the level they are at.

```
# * means optional display on condition
---------------------------------------
Your stats:
Hero:          $hero.name
Health:        $hero.health
```

```
Bag Items:    $hero.item_count
*Loot Nearby: $env.loot_list_count

Action:
1) Go to next room
2) Show Bag
3) *Pickup loot
q) quit
---------------------------------------
```

Adventure Menu — Battle/Round

Battle has three portions to it

→ Battle: The overall battle between hero and monster(s)

→ Round: A single round is when each character has performed an action

→ Duel: A duel is the individual action taken between 1 hero and 1 monster

```
# This menu gets repeated until death or run away
# * means optional display on condition

Battle in $env.name!
$env.attacker has the initiative!
--------------------------------------
Round $env.round:

$hero.name stats:
----------------
  Health:    $hero.health
  Bag Items: $hero.item_count

vs.

Monster(s) stats:
-----------------
  *Monster $#:
  ------------
  Monster:    $monster.name
  Health:     $monster.health
  Dice Count: $mosnter.dice_count

---------------------------------------
Actions:
1) Battle!
2) Show Bag
2) Run away (Chance of Success: $($hero.health * .1))
```

Adventure Menu — Duel

This menu will display what actually happened between attacker and defender

```
# * means optional display on condition

# When hero is defender
Duel: $env.attacker vs $env.defender
------------------------------------
$env.attacker.name rolls: [8, 3, 1]
$env.defender.name rolls: [3, 1]

Attack $attack_success!

* Defender takes $attack.damage.
* Defender dies!
------------------------------------

###################################################

# When hero is attacker
Duel: $env.attacker vs $env.defender
------------------------------------
* Monster 1 of x

$hero.name stats:
----------------
  Health:    $hero.health
  Bag Items: $hero.item_count

$monster.name stats:
-----------------
  Monster:    $monster.name
  Health:     $monster.health
  Dice Count: $mosnter.dice_count
--------------------------------------
Actions:
  1) Fight!
  2) Show Bag
  3) Run Awayeoka
```

Loot Bag

There are two types of items, consumables or trinkets

```
$hero.name inventory:
-----------------------------------------------
QTY           Name         Affect
$loot.count   $loot.name   $loot.descrioption

Action:
* 1) Consume $loot.name
q) Leave bag
------------------------------------------------
```

Design and Implementation

Duel

The duel will ideally take in two generic objects that are guarantee to have methods to assist with this function. This means that both the `hero` and `monsters` need to be inherited from the same `abstract class` with these methods

```
def duel(attacker: PlayerCharacter, defender: PlayerCharacter): -> None
  """Only resonsible for an actual fight between attacker and defender regardless
      if monster or not"""

  # roll for combat
  attacker.combat_roll()
  defender.combat_roll()

  # check if attack succeeded
  if attack_successful(attacker, defender):
    defender.takes_hit()
```

Death and Loot

Called after each duel

```
def death_check(hero: HeroCharacater, monster: MonsterCharacter) -> None:
    if hero.is_dead():
      # exit game status
    if monster.is_dead():
      loot(hero, monster)

def loot(hero: HeroCharacter, monster: MonsterCharacter) -> None:
    if monster.has_loot():
      # provide the option to loot it
```

Character Abstract Class

```
class Character(ABC):
  @abstractmethod
  def takes_hit(self) -> None:
      """Defines a way to decrement the character"""

  @abstractmethod
  def combat_roll(self) -> None:
      """Rolls the x number of dice and sorts them"""

  @property
  @abstractmethod
  def get_roles(self, index) -> Optional[int]:
    """Returns what was rolled on the n'th die if any"""

  @property
  @abstractmethod
  def is_dead(self) -> bool:
    """set during 'takes hit'"""

  @property
  @abstractmethod
  def is_monster(self) -> bool:
```

MonsterCharacter

This class inherits from the Character abstract. From here the individual monsters are inherited

Character → MonsterCharacter → ImpMonster

```
class MonsterCharacter(Character):
  def __init__(self):
    self._loot = self.get_loot()
```

KanBan

KanBan

Name	Tags	Desc
Create base classes	Complete	Create the base classes needed for the game to work
Create Loot Mechanism	Complete	
Create Combat functions	Complete	
Create menus	Complete	
Monster_Controller	Complete	
Game_Controller	Complete	
writeup	Complete	
desgin plan	Complete	

Resources