## Potential Classes

Details to be implemented:

* Rooms don’t have tiles, but the outside of rooms do, however we can still represent it as tiles, and randomly assign a tile within a room to a piece, or overlay each piece and show the appropriate piece for the turn ( e.g. to mitigate someone making a room which is a 1x1 or if a room isn’t large enough to house each character in the game) but **only** happens within a room. Not all described is necessary for the board, more so for the class Person, but feels appropriate to describe here
* Weapons assigned to a room will be taken if

**Potential classes so far :**

* **Board**
  + Generates board from file
    - stores it in a matrix with symbols for associated things (room, token (includes player and weapon token)
    - Start spaces are in board file
    - Secret passages are in the four corners of the board assigned to the opposite position. E.g. top left moves you to bottom right
  + Weapons assigned to rooms
  + Has a matrix which holds info where the different rooms, tunnels, general tiles and holds positions of tokens.
  + Tracks users turn
* **Solution**
  + Generates solution, used to compare to when an accusation is made
    - Fields
      * w → Weapon
      * r → Room
      * p → Player
    - Methods
      * generateSolution() [Need to somehow have the objects selected from] → need to remove cards from deck
      * checkSolution(Weapon, Room, Player)
* **Card**
  + E.g. name, type and associated map symbol
  + I haven’t removed the copied methods from each one just in case we still need them
* **Weapon**
  + Weapon, type (e.g. the type of weapon), associated map symbol (?)
* **Room**
  + Name, type (e.g. the decor type for the room), associated map symbol (?)
* **Player**
  + Stores hand of card, can accuse (need somewhere to store every card for each card type), name, move, dice roll, all user actions, associated map symbol, type (for what icon to use in the token) etc
* **AI**
  + Inherits **Player**
  + **Player** but automated decisions
* **Human**
  + Inherits **Player**
  + **Player** but need a way to input for each method
* **Token**
  + Holds position on board, icon
* **PlayerToken**
  + Inherits **Token**
  + Stores relevant **player** object
  + Can move position
* **WeaponToken**
  + Inherits **Token**
  + Stores relevant **weapon** object
  + Static position
* **CardDeck**
  + All the cards not given to players or in the solution
  + Store in Array

**-------------------------------------- Is Old --------------------------------------**

**Most likely in order of creation** (if above classes are the ones we implement) **:**

* **Board**
  + Will have to figure out a few problems
* **Card** (I think, depends if we use it, **weapons**, **room**, **people** (they are all cards irl) could inherit it and others depending on the rules of the game, need to play the game to see how I could represent objects)
* **Room** and / or **Rooms**
* **Weapons**
* **People**
* **Solution**
* **Turn**, **Dice**
* **Accusation**
* **Player**
* (Any other class if any are needed)
* **CluedoAI**

## Problems

We would need one script to have the main procedure in to initialise all the objects etc, probably in the same class with the GUI, like what we done with minesweeper

With the GUI, it could be developed afterwards or while we are programming the other aspects

I need to go over all the aspects of the game again to see if this is all covered

Problems to try and figure out:

* Multiple people playing
* How to store the map and how to edit it
  + If no map file exists create a default one or somehow generate one
* Cluedo AI in general

Potential way of structuring the files of our project:

project\_name/

project\_name/

\_\_init\_\_.py

...source code files...

test/

\_\_init\_\_.py

...unit test files...

docs/

...documentation files...

project\_name.py

README.md

LICENSE.md

requirements.txt

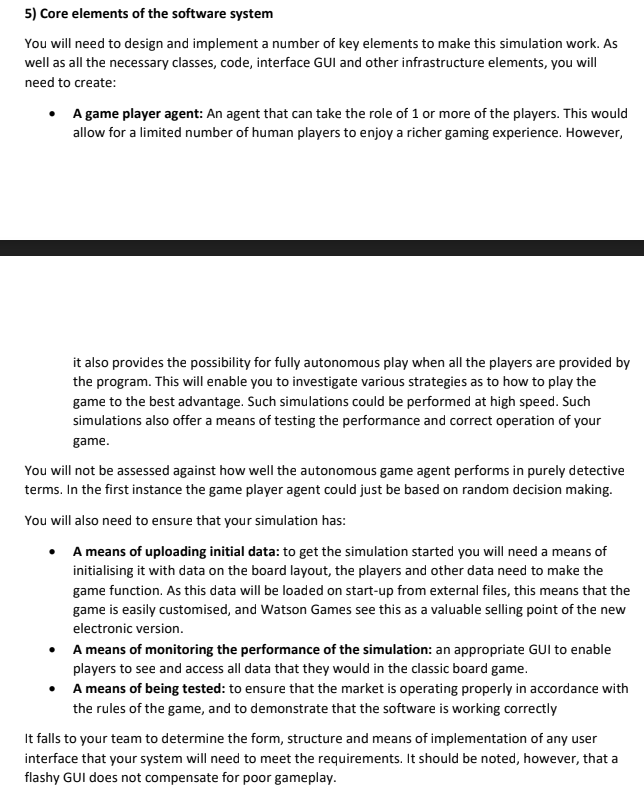
.gitignore

( <https://www.patricksoftwareblog.com/structure-of-a-python-project/> )

As it needs to run on Mac and PC, we need to figure out a way to easily start up the program without using the console for both, we can use pyinstaller but will need a Mac VM or if someone has a Mac to use pyinstaller on it with our project to get a distributable for that OS

## TODO

* **Play the game**
* **Look through the spec again**



## Old

Pink - Sam, Adam

Green - Will, Umar

Yellow - Sarah, Ethan

Reddy orange boi - Sarah, Ethan

* **Player -** a class that determines each human player in the game
  + Fields
    - Position
* **Person - A class for the similarities between Player and Character, e.g. name**
* **Turn -** an object for each turn that occurs
* **Solution -** what the solution is for the current game
  + Fields
    - Weapon
    - Room
    - Person
  + Methods
    - getWeapon() → Weapon
    - getRoom() → Room
    - getPerson() → Person
    - getSolution() → Tuple(Weapon, Room, Person)
    - setSolution(Weapon, Room, Person)
* **Board -** the board, its design and locations
  + Rooms don’t have tiles, but the outside of rooms do, however we can still represent it as tiles, and randomly assign a tile within a room to a piece, or overlay each piece and show the appropriate piece for the turn ( e.g. to mitigate someone making a room which is a 1x1 or if a room isn’t large enough to house each character in the game) but **only** happens within a room. Not all described is necessary for the board, more so for the class Person, but feels appropriate to describe here
  + Need to load the board from a file, has to be easily edited.
    - Could just be a text file with different symbols for the different associated rooms
* (Possibly) Tile
  + An alternative is to have a matrix of the board size stored on the board class itself with dictionaries for each position for each piece in those locations. Dictionaries because we could store different things such as find if something has a weapon by accessing the key ‘weapon’ and see if we get a value returned. Check if a tile is a door to be placed into the room in a random tile if one is available and if not be placed on top of another (**Need to figure out this problem**)
  + Fields
    - isRoomTile Boolean
      * If it is it can have multiple pieces in that position
    - Pieces (Person or Card (If similar attributes can make them inherit a class) )
* **Room -** object for every room
  + Include secret passage
  + Room name
* Rooms - Stores multiple **Room** classes
  + Inherits Cards
  + Have predefined rooms within functions and call them during creation of the game
* **Weapons -** Stores multiple **Weapon** classes
  + Inherits Cards
* **Dice -** dice…
  + It should be integrated with **Turn** as a method instead of being another class, unless customisable dice?
  + roll()
* **Character**
* **Characters -** the characters, selected by players
  + Inherits Cards
* **Accusation -** a suggestion of room, player and weapon
  + Fields
    - Accuser as type Person or Player
    - Weapon
    - Room
    - Person
  + Methods
    - isCorrect(Weapon, Room, Person) → Boolean
* **CluedoAI -** AI objects for the AI players should we choose to includethem
  + Inherits from **Player**
* **Card** 
  + Also possibly **Cards**
  + Fields
    - Name
    - Position (?)
    - Owner (?)
* **Cards**
  + Stores multiple Card classes, all categories of card is split into several decks of Cards
  + Private vars
    - cards (Array)
  + Methods
    - add(card)
    - random() → Select random element from card