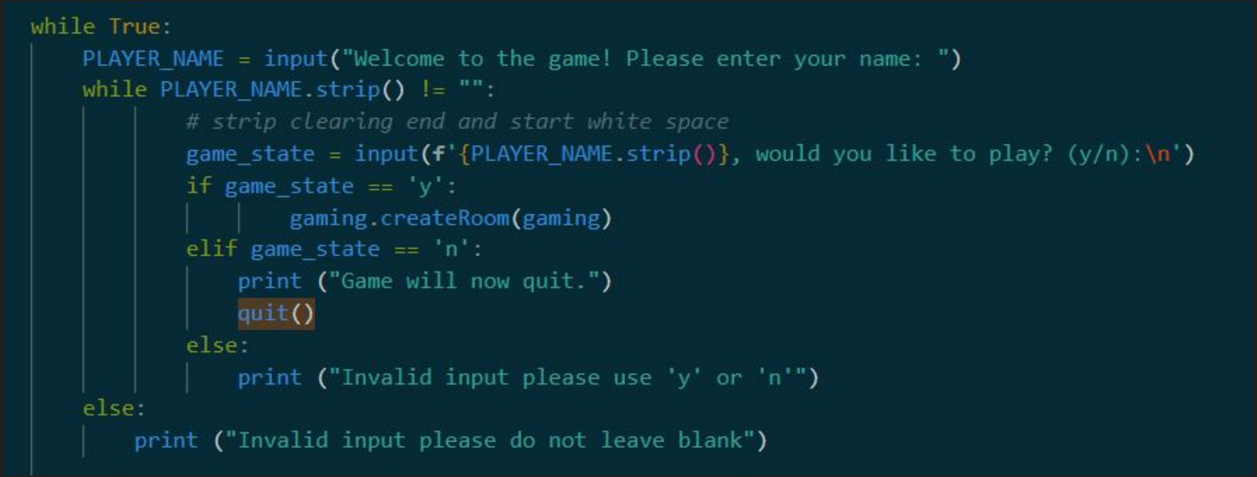
T1A3

Features

- User Input
- Weapon Check
- Random Password 8 Digit Code

1. User Input

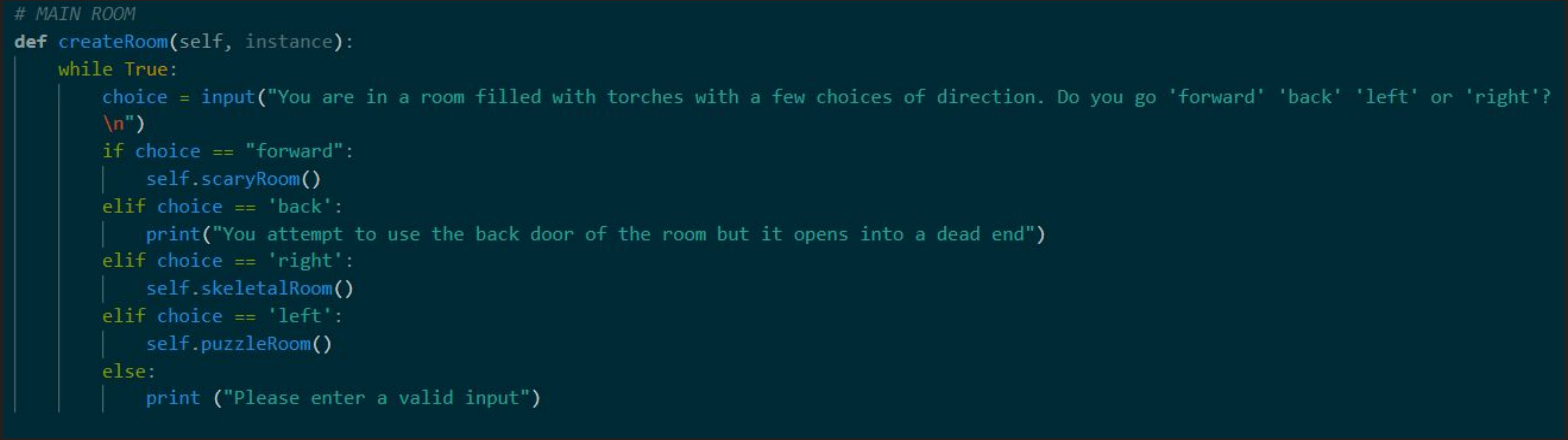
- Input name into game
- Strip() function ensuring not an empty space
- While loop until valid input entered



```python
while True:
    PLAYER_NAME = input("Welcome to the game! Please enter your name: ")
    while PLAYER_NAME.strip() != "":
            # strip clearing end and start white space
            game_state = input(f'{PLAYER_NAME.strip()}, would you like to play? (y/n):\n')
            if game_state == 'y':
                    gaming.createRoom(gaming)
            elif game_state == 'n':
                print ("Game will now quit.")
                quit()
            else:
                print ("Invalid input please use 'y' or 'n'")
    else:
        print ("Invalid input please do not leave blank")
```

1. User Input

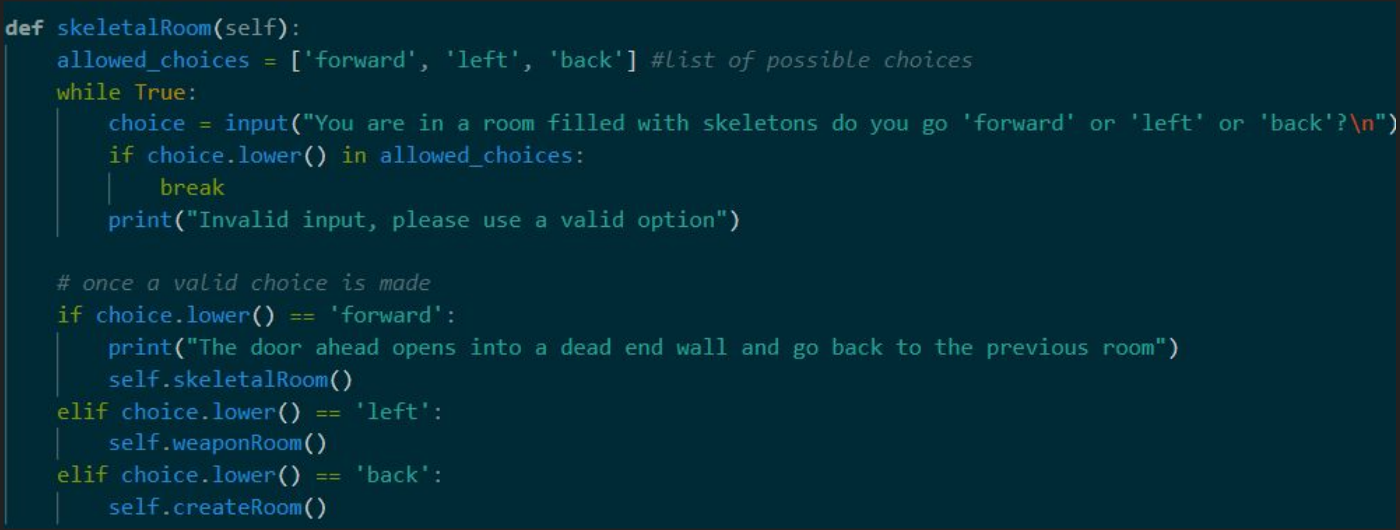
- Allow user to move through rooms
- While loop until valid input entered
- Elifs to call next function in code.



```python
# MAIN ROOM
def createRoom(self, instance):
    while True:
        choice = input("You are in a room filled with torches with a few choices of direction. Do you go 'forward' 'back' 'left' or 'right'?
        \n")
        if choice == "forward":
            self.scaryRoom()
        elif choice == 'back':
            print("You attempt to use the back door of the room but it opens into a dead end")
        elif choice == 'right':
            self.skeletalRoom()
        elif choice == 'left':
            self.puzzleRoom()
        else:
            print ("Please enter a valid input")
```

1. User Input

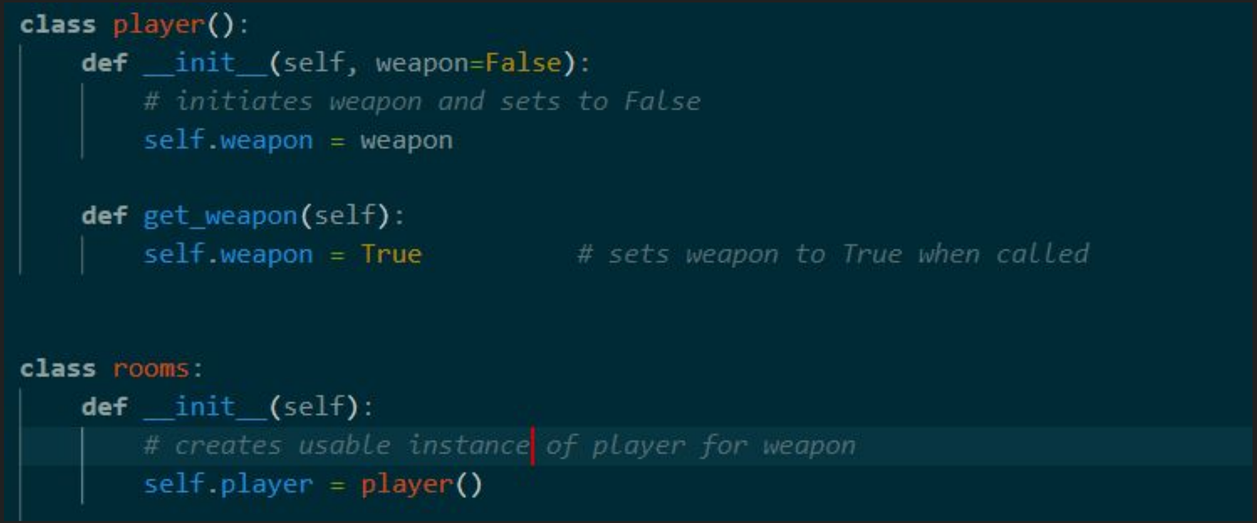
- Different style of coding using a list and looping before going to if statement



```python
def skeletalRoom(self):
    allowed_choices = ['forward', 'left', 'back'] #list of possible choices
    while True:
        choice = input("You are in a room filled with skeletons do you go 'forward' or 'left' or 'back'?\n")
        if choice.lower() in allowed_choices:
            break
        print("Invalid input, please use a valid option")

    # once a valid choice is made
    if choice.lower() == 'forward':
        print("The door ahead opens into a dead end wall and go back to the previous room")
        self.skeletalRoom()
    elif choice.lower() == 'left':
        self.weaponRoom()
    elif choice.lower() == 'back':
        self.createRoom()
```

2. Weapon Check

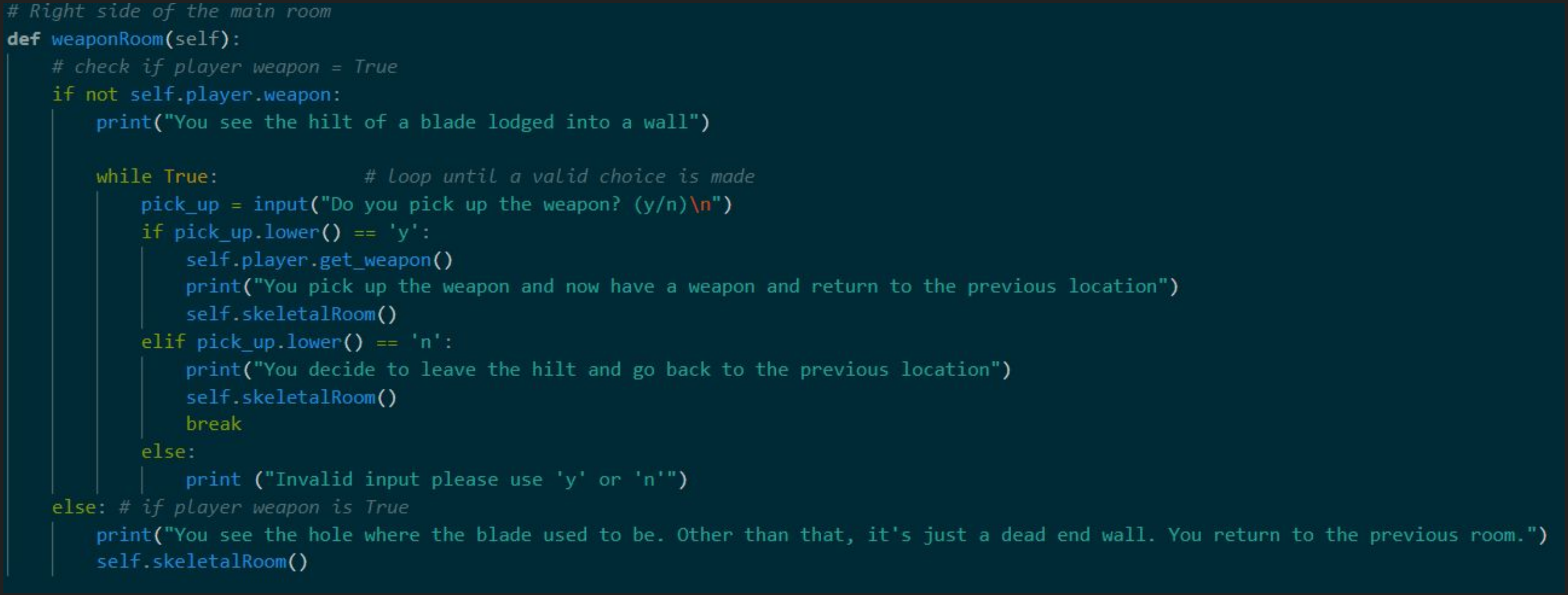
- Initialise Weapon set as False
- Function to set Weapon to True
- Place Weapon into init of rooms class to be usable throughout



```python
class player():
    def __init__(self, weapon=False):
        # initiates weapon and sets to False
        self.weapon = weapon

    def get_weapon(self):
        self.weapon = True          # sets weapon to True when called


class rooms:
    def __init__(self):
        # creates usable instance of player for weapon
        self.player = player()
```

2. Weapon Check

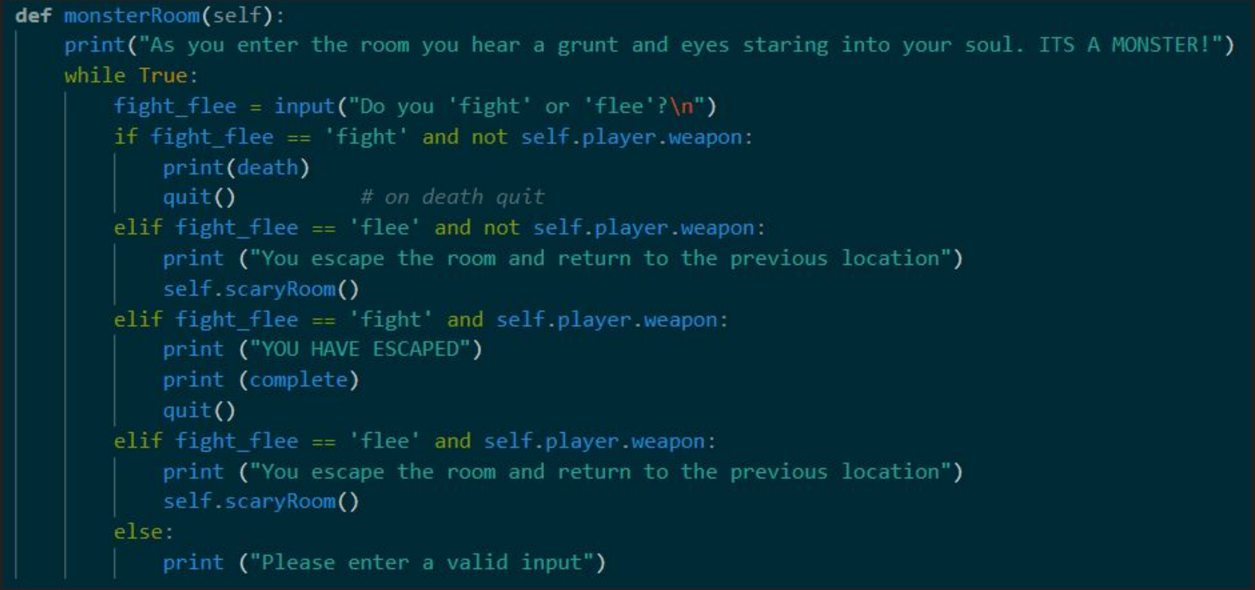
- Once weapon picked up changes text due to checking if True



```
# Right side of the main room
def weaponRoom(self):
    # check if player weapon = True
    if not self.player.weapon:
        print("You see the hilt of a blade lodged into a wall")

        while True:             # loop until a valid choice is made
            pick_up = input("Do you pick up the weapon? (y/n)\n")
            if pick_up.lower() == 'y':
                self.player.get_weapon()
                print("You pick up the weapon and now have a weapon and return to the previous location")
                self.skeletalRoom()
            elif pick_up.lower() == 'n':
                print("You decide to leave the hilt and go back to the previous location")
                self.skeletalRoom()
                break
            else:
                print ("Invalid input please use 'y' or 'n'")
    else: # if player weapon is True
        print("You see the hole where the blade used to be. Other than that, it's just a dead end wall. You return to the previous room.")
        self.skeletalRoom()
```

2. Weapon Check

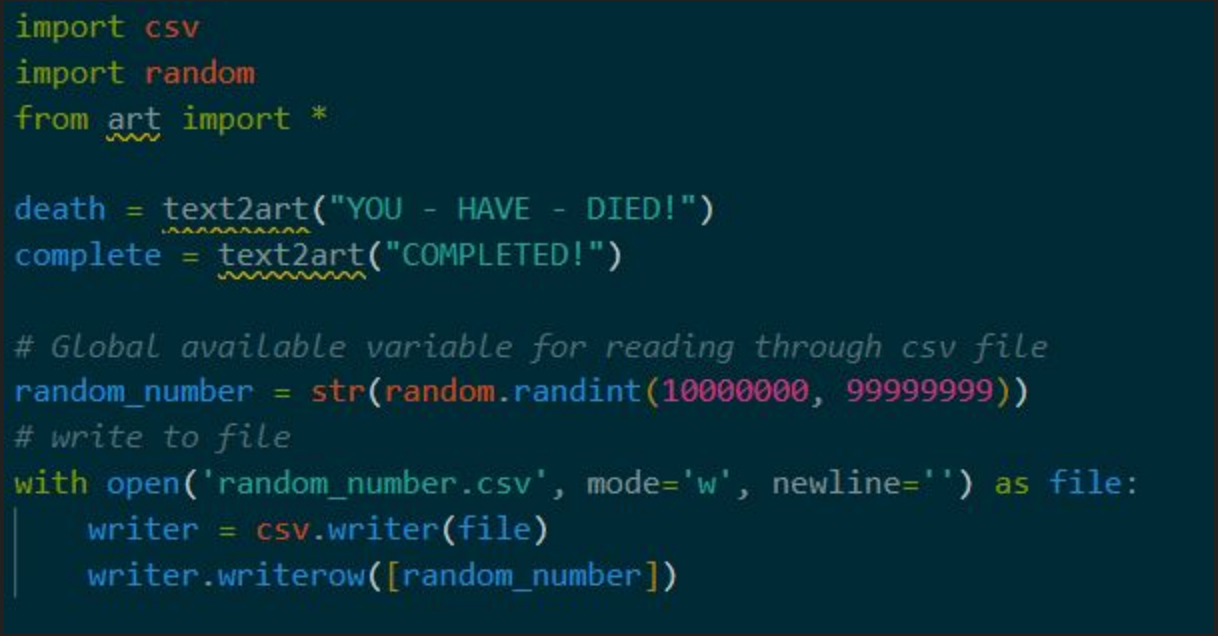
- Weapon is referenced in monsterRoom affecting the outcome of choices.



```python
def monsterRoom(self):
    print("As you enter the room you hear a grunt and eyes staring into your soul. ITS A MONSTER!")
    while True:
        fight_flee = input("Do you 'fight' or 'flee'?\n")
        if fight_flee == 'fight' and not self.player.weapon:
            print(death)
            quit()          # on death quit
        elif fight_flee == 'flee' and not self.player.weapon:
            print ("You escape the room and return to the previous location")
            self.scaryRoom()
        elif fight_flee == 'fight' and self.player.weapon:
            print ("YOU HAVE ESCAPED")
            print (complete)
            quit()
        elif fight_flee == 'flee' and self.player.weapon:
            print ("You escape the room and return to the previous location")
            self.scaryRoom()
        else:
            print ("Please enter a valid input")
```

3. Random Number

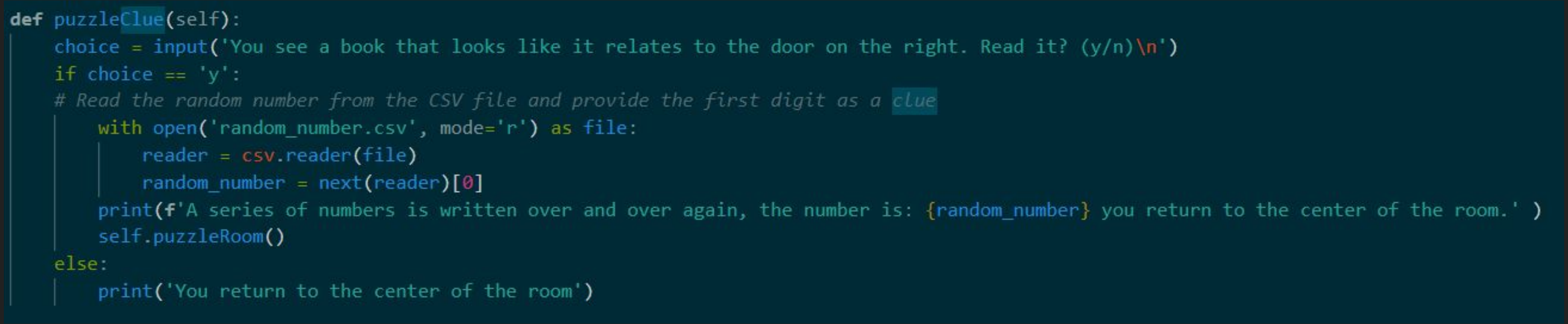
- Random number generated and printed to CSV file



```python
import csv
import random
from art import *

death = text2art("YOU - HAVE - DIED!")
complete = text2art("COMPLETED!")

# Global available variable for reading through csv file
random_number = str(random.randint(10000000, 99999999))
# write to file
with open('random_number.csv', mode='w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow([random_number])
```

3. Random Number

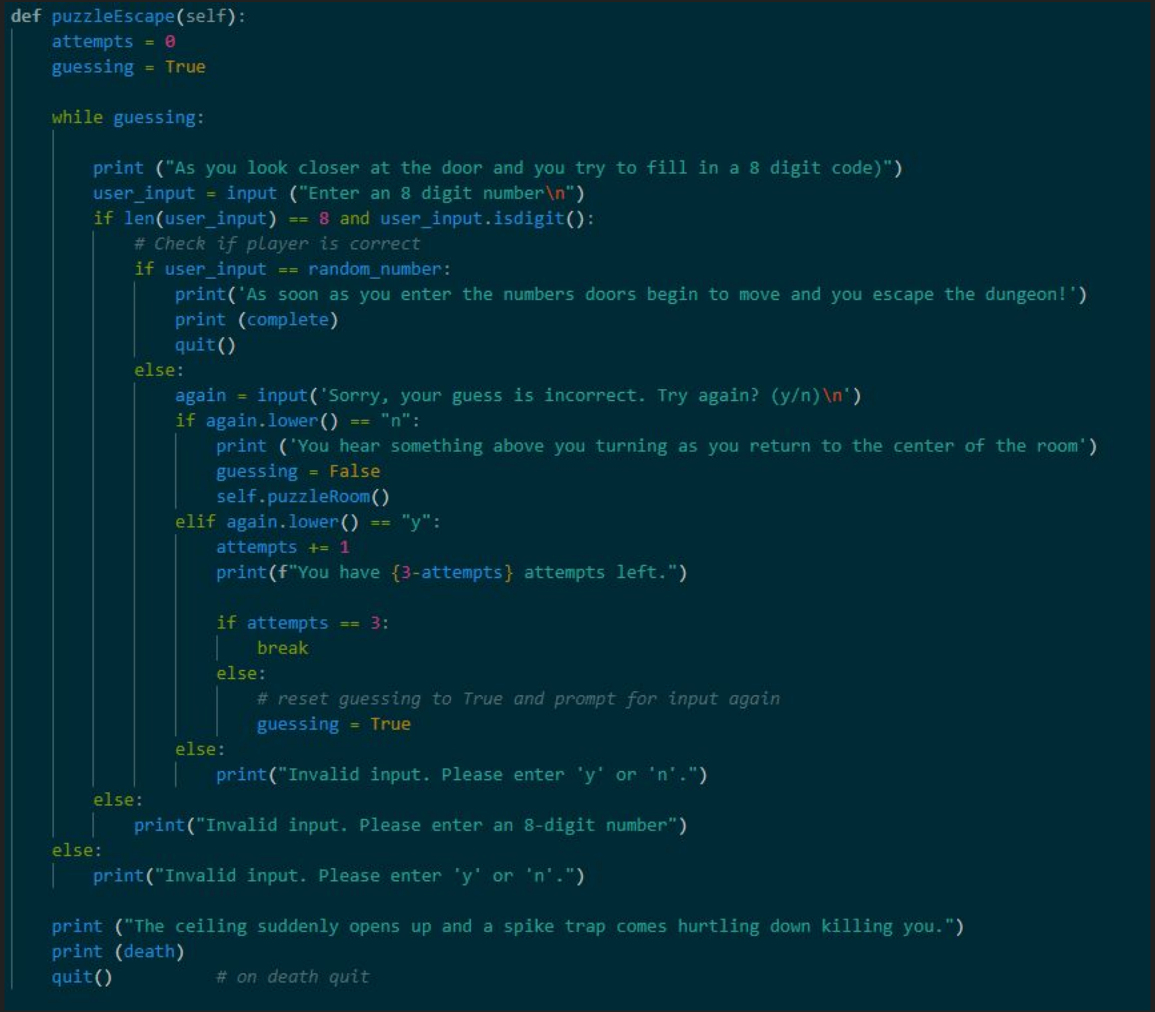
- Player interaction to read



```
def puzzleClue(self):
    choice = input('You see a book that looks like it relates to the door on the right. Read it? (y/n)\n')
    if choice == 'y':
    # Read the random number from the CSV file and provide the first digit as a clue
        with open('random_number.csv', mode='r') as file:
            reader = csv.reader(file)
            random_number = next(reader)[0]
        print(f'A series of numbers is written over and over again, the number is: {random_number} you return to the center of the room.' )
        self.puzzleRoom()
    else:
        print('You return to the center of the room')
```

3. Random Number

- Player able to input number to escape
- Can leave puzzle to reset attempts
- At 3 attempts player dies and game quits



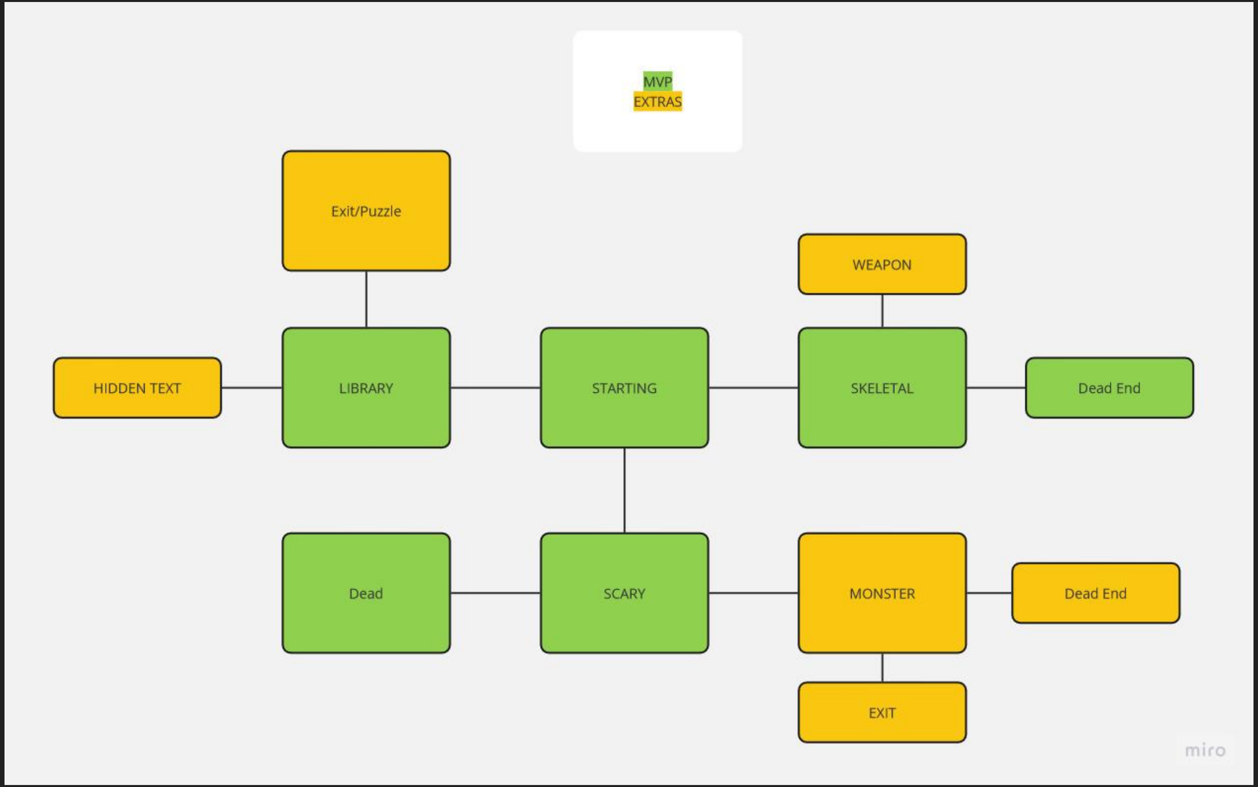
```python
def puzzleEscape(self):
    attempts = 0
    guessing = True

    while guessing:

        print ("As you look closer at the door and you try to fill in a 8 digit code)")
        user_input = input ("Enter an 8 digit number\n")
        if len(user_input) == 8 and user_input.isdigit():
            # Check if player is correct
            if user_input == random_number:
                print('As soon as you enter the numbers doors begin to move and you escape the dungeon!')
                print (complete)
                quit()
            else:
                again = input('Sorry, your guess is incorrect. Try again? (y/n)\n')
                if again.lower() == "n":
                    print ('You hear something above you turning as you return to the center of the room')
                    guessing = False
                    self.puzzleRoom()
                elif again.lower() == "y":
                    attempts += 1
                    print(f"You have {3-attempts} attempts left.")

                    if attempts == 3:
                        break
                    else:
                        # reset guessing to True and prompt for input again
                        guessing = True
                else:
                    print("Invalid input. Please enter 'y' or 'n'.")
        else:
            print("Invalid input. Please enter an 8-digit number")
    else:
        print("Invalid input. Please enter 'y' or 'n'.")

    print ("The ceiling suddenly opens up and a spike trap comes hurtling down killing you.")
    print (death)
    quit()          # on death quit
```

Validation

- Validation throughout code
- Ensures valid inputs only by the user using while loops and ifs
- Ensures that puzzle input is only 8 digits, containing only numbers using len and is digit()
- Use of lower() throughout to ensure that whether input is upper or lower case text still works

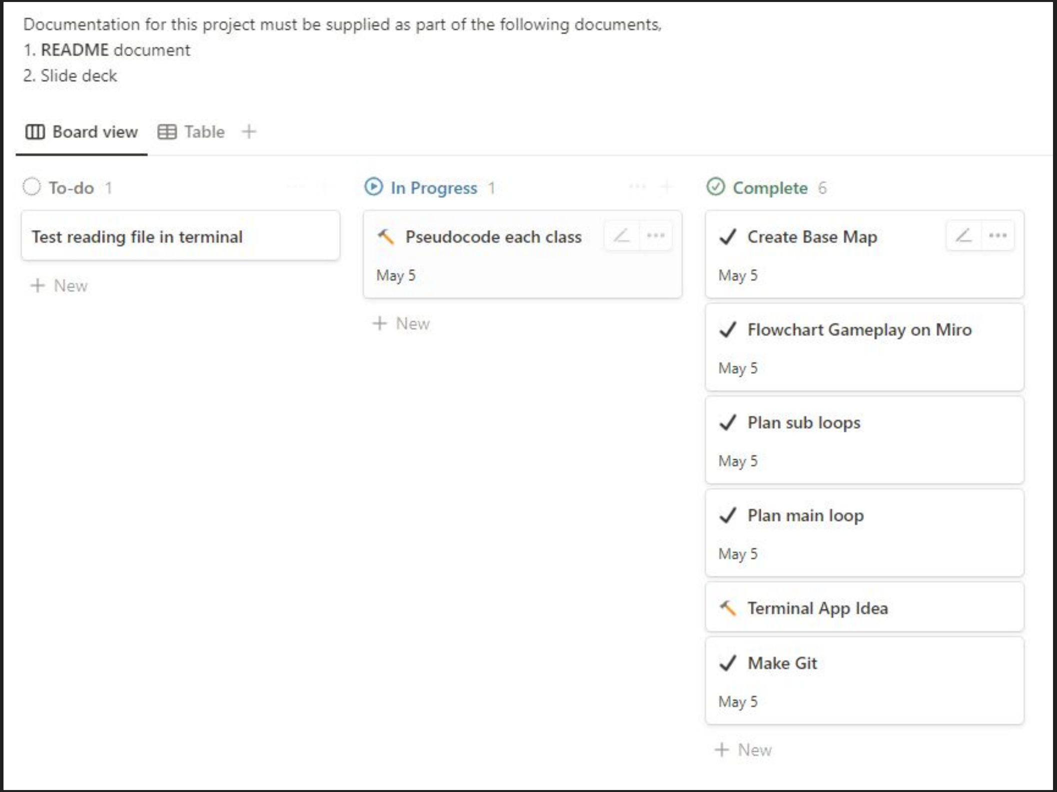
Development Plan

- Agile plan
- Create an MVP
- Green rooms bare minimum functionality
- Yellow rooms to achieve if possible



Development Plan

- Used Notion
- Used to it
- Less functionality than a traditional Kanban board



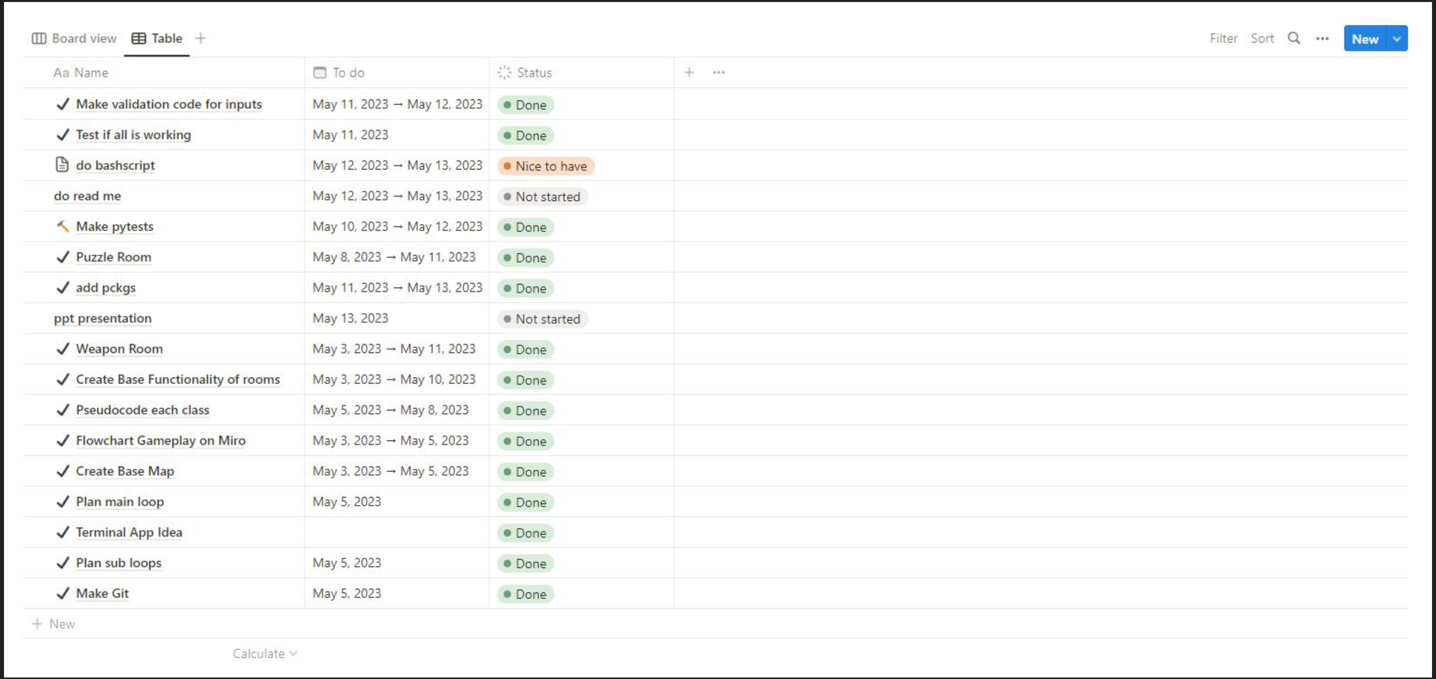
Documentation for this project must be supplied as part of the following documents,
1. README document
2. Slide deck

Board view | Table

To-do 1	In Progress 1	Complete 6
Test reading file in terminal	Pseudocode each class May 5	Create Base Map May 5
		Flowchart Gameplay on Miro May 5
		Plan sub loops May 5
		Plan main loop May 5
		Terminal App Idea
		Make Git May 5
+ New	+ New	+ New

Development Plan

- Different formatting options such as table format or calendar format

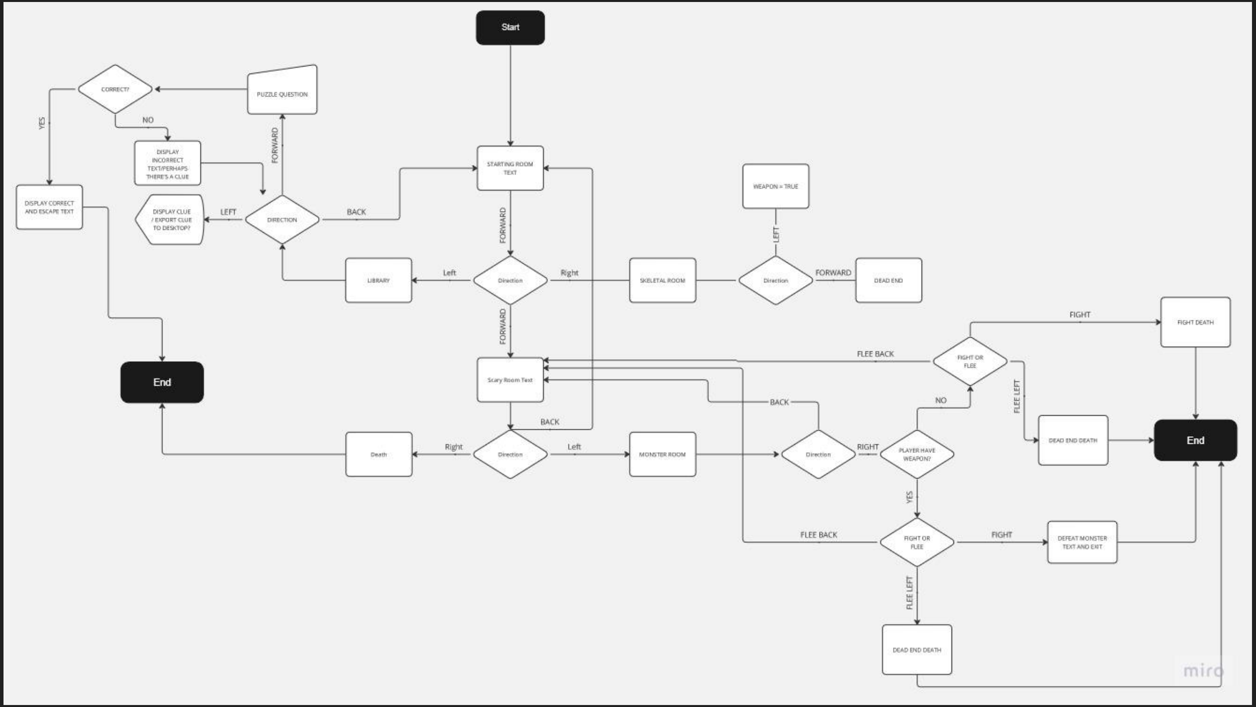


Name	To do	Status
✓ Make validation code for inputs	May 11, 2023 → May 12, 2023	Done
✓ Test if all is working	May 11, 2023	Done
do bashscript	May 12, 2023 → May 13, 2023	Nice to have
do read me	May 12, 2023 → May 13, 2023	Not started
Make pytests	May 10, 2023 → May 12, 2023	Done
✓ Puzzle Room	May 8, 2023 → May 11, 2023	Done
✓ add pckgs	May 11, 2023 → May 13, 2023	Done
ppt presentation	May 13, 2023	Not started
✓ Weapon Room	May 3, 2023 → May 11, 2023	Done
✓ Create Base Functionality of rooms	May 3, 2023 → May 10, 2023	Done
✓ Pseudocode each class	May 5, 2023 → May 8, 2023	Done
✓ Flowchart Gameplay on Miro	May 3, 2023 → May 5, 2023	Done
✓ Create Base Map	May 3, 2023 → May 5, 2023	Done
✓ Plan main loop	May 5, 2023	Done
✓ Terminal App Idea		Done
✓ Plan sub loops	May 5, 2023	Done
✓ Make Git	May 5, 2023	Done

Development Plan

- Flowchart how game should go if all completed

- Helped to understand logic behind choices and directions



Review of the Build

- Most challenging portion: testing and using pytest
- With time can make it DRY’er and more modular
- Less time planning and more doing