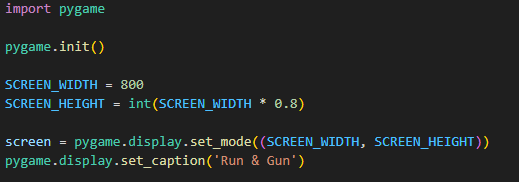
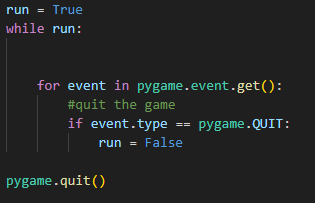
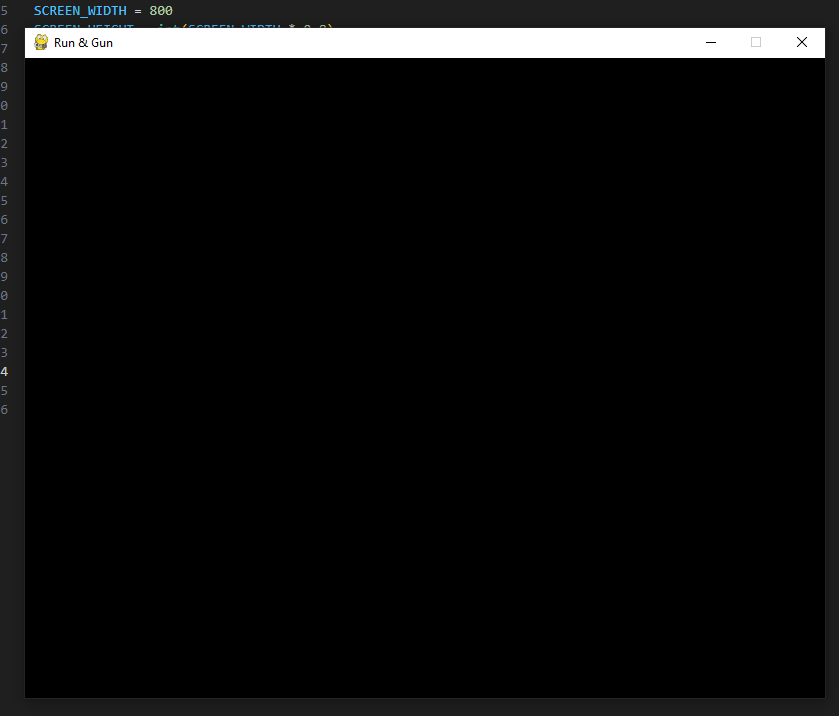
Setting up the game infrastructure and screen



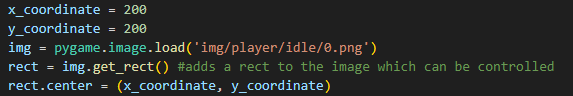
Set up game loop and allow for game exit, create event handler



Game now pops up in window and doesn’t exit immediately



set initial character sprite on screen



Add blit function inside of game loop to show character

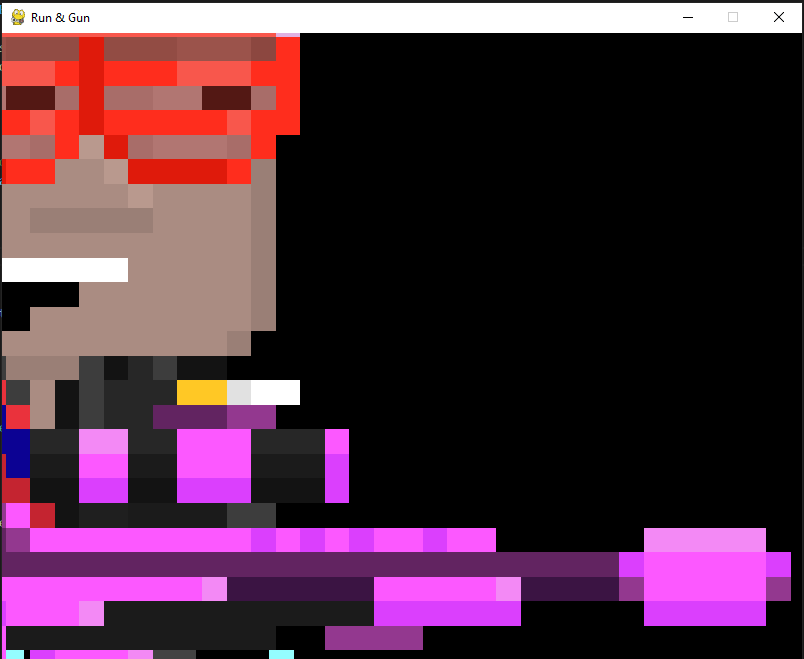


Error – player cannot be seen

Solution – update display within game loop



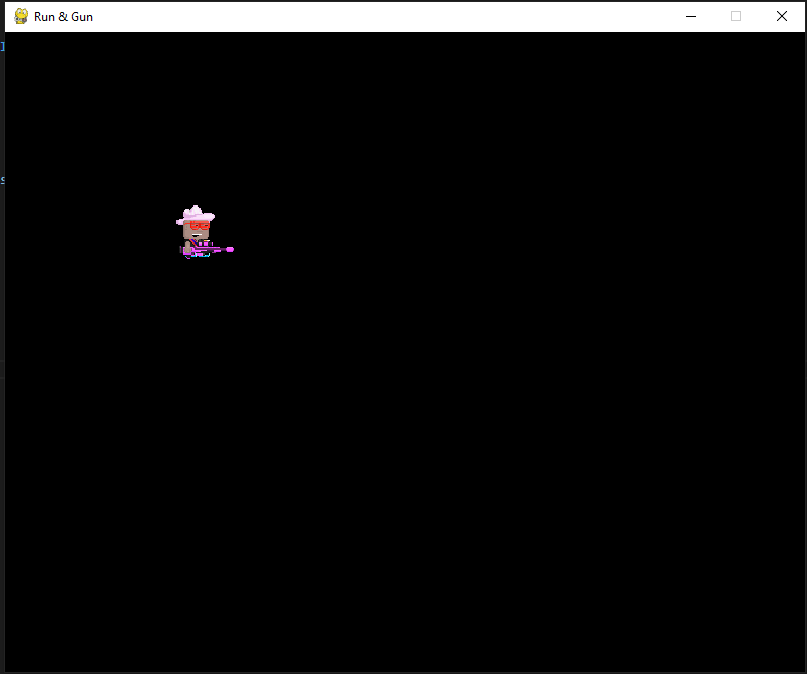
Character appears but is too large for display



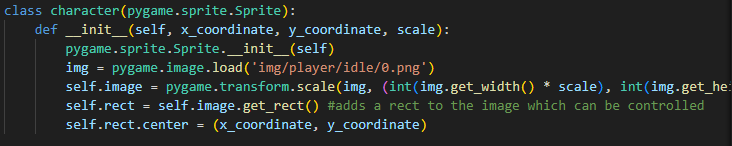
Character can be scaled with the scale function



Character is now a more appropriate size



Begin work on generic superclass for characters



Create first instance and load

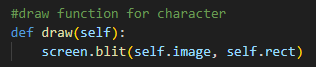




Successfully loaded player



Add draw method

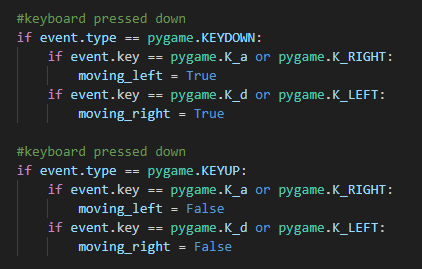


Receive following error:



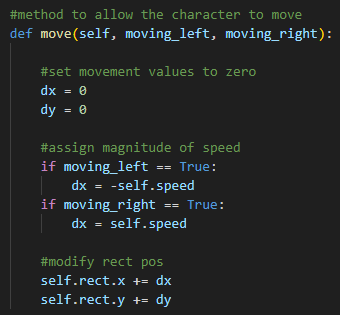
Solution: the method wasn’t indented properly below \_\_init\_\_

Begin working on movement for the player



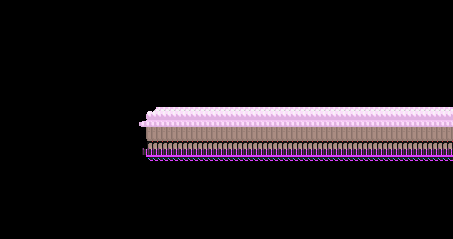
#second comment meant to say keyboard button released

Player is marked as moving when a key is pressed down, player is marked as not moving when key is released



Speed is set when instantiating the character

We set the magnitude of the speed (positive or negative) depending on Boolean value of moving\_left / moving\_right

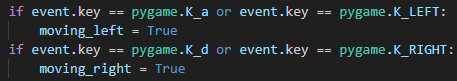


Character appears on screen but

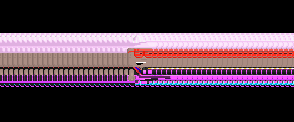
1.moves too fast

2. has multiple sprite instances

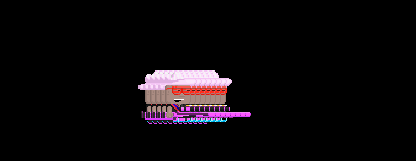
3. can only be moved in one direction



SOLUTION: incorrect use of or function, key press should be preceeded by event.key ==, also there were opposing buttons set for the same direction (a and Right Key to move left…)

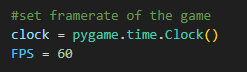


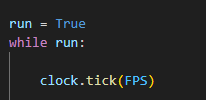
Player can now move left and right



Player can now move slower

Fix: adding a framerate limit since computer was processing the movement too quickly





To fix the sprite trail, we had to draw a new background in each tick to mask the previous sprite position

We did this by implementing a new function called Draw\_background and calling it at the start of the game loop

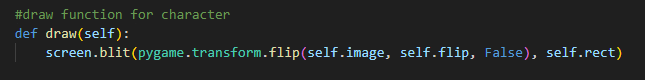


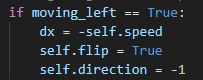
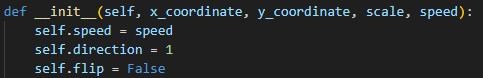




However, it’d be good if the player flipped around when moving towards the left

We did this by adding a Boolean value self.flip in the class definition to define whether we are flipping or not, changing this value depending on direction, and modifying the sprite image in the blit function when called using this command, which will flip the sprite horizontally when self.flip is True.



Player can now face left and right



Now we need to modify the character superclass so it can load different sprites depending on whether it is a main character or an enemy.

We can modify the sprite image load code to change from which directory it pulls a sprite from.



We first have to set a new parameter in the instantiation which decides if the object is the player or an enemy:

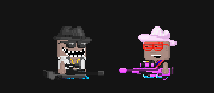


Using python’s string modification features we can replace “player” with the objects’ respective type



Now we can call both object types





And now both an enemy, and the main character appear



Added an alive variable into class definition for later



And begin working on the ability for character to jump



Add control for jump using space or W, and only allow if player is alive

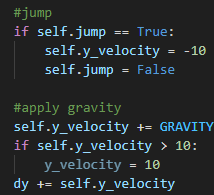
We can begin working on gravity now. First we set a y velocity in the character class



We set a global value for gravity

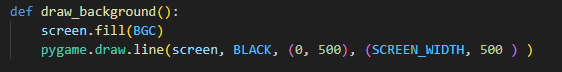


And begin working on jumping inside the move method

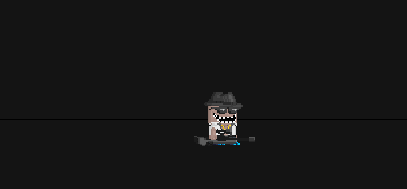
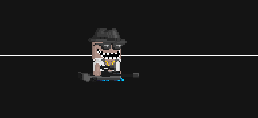


The player can now jump, but continues falling off the screen

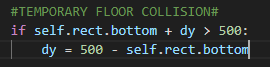
Temporarily for development, we can add a floor so the player cannot move past a certain limit



We can draw a line when creating the background, making it go horizontally across

A line is now present across the screen – ill change the colour to white



Now we can add a temporary floor collision check to ensure the player cannot move below where the line is



Player can now not move below the line

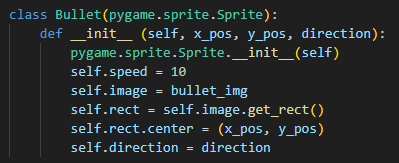
Next we need to begin working on the ability for the characters to shoot

This will be done by creating a bullet class

First we load the bullet png



Next we can create the barebones for the bullet class



And create a sprite group to efficiently manage the bullet objects



Next we can set up controls for shooting, and I’ve rewritten space to shoot instead of as a second jump button

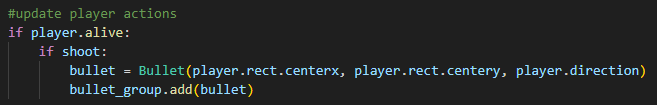


Event.key line is place both into Keyup and keydown

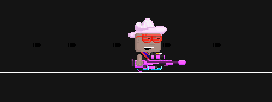
We also need to initialise shoot as False under the the character action status block at the top



The player should only be able to call the shoot method whils theyre alive. We can achieve this through creating an update player action block in the game loop, so shoot can only be called if player.alive is True.



Pressing space now spawns a bullet sprite on screen



One problem is that the bullet appears at the centre of the player, which would later cause collision problems. It also appears above where the player’s gun is

We can modify the bullet location using the player rect’s dimensions





And now the bullet appears from the end of the gun nozzle (I’ve also changed background color to make elements more clearly visible)

Another problem is that when the player turns around (the sprite is flipped) the bullets seem to come out behind him. This is fixed by multiplying the x position of the bullet by player.direction which is either 1 or -1, so we place the bullet behind or infront of the sprite. Y value stays the same.

**UNFIXED**

The bullets are facing the wrong way however when shot

Now we need to make the bullets move. This is achieved by updating the bullets x position

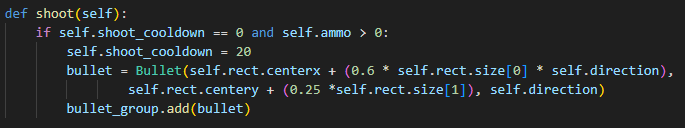




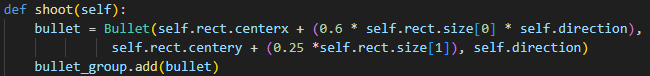
The bullets now move. But we need to remove any that are off the screen so they don’t take up memory



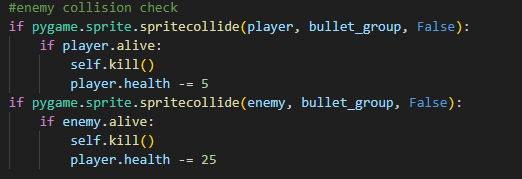
Now we need to fix the stream of bullets that come out when we press space by adding a cooldown. It’s also a good opportunity to begin working on ammo counts.



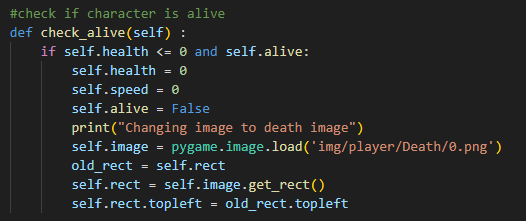
I’ve also created a shoot method in the main character class so both player and enemies can shoot using the same bullet class. Now the character can shoot by simply using player.shoot()



Now we need to make the bullet disappear if it collides with a character and reduce the player’s health

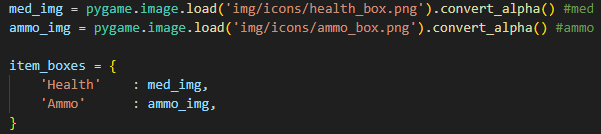


Now we need to check if the player is alive, and set it as dead if so. We also will replace the character’s sprite to a tombstone.

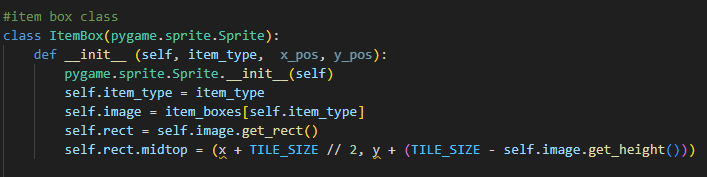


Now that the characters can shoot and die, we need a way to replenish their health and ammo count using pickups

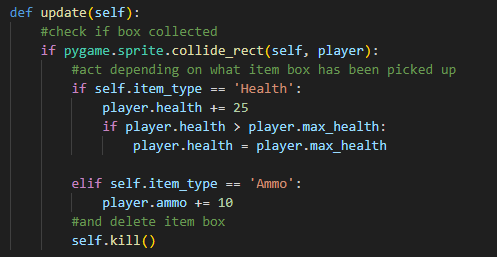
First we’ll load the sprites for the collectibles and put them in a dictionary



Next we’ll create the class



And give functionality to each type using an update method



Then we create an item box group to hold the item boxes



And using the built in draw and update functions we can load them in



Create a text draw function to display key details



Display number of bullets character has



Ammo is now displayed

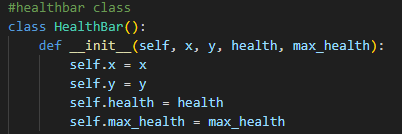


Issue encountered: although only one bullet is shot, holding space continues to decrease the ammo count

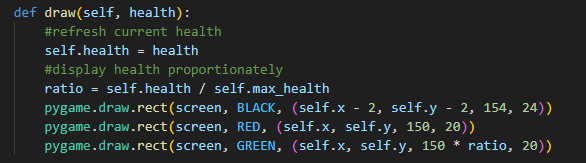


**Unresolved**

Next we can create a healthbar

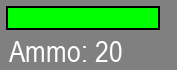


And using the parsed variables we can create a method to draw it

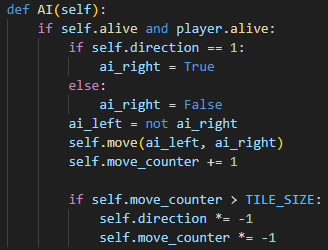


Then once called with player.health, it appears on screen





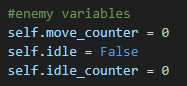
Begin working on enemy AI



Makes them move back and forth between two points

Next we want the enemy to randomly stop moving occasionally and idle

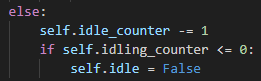
First we add these two enemy specific variables



Then we can set a random condition to make them idle for a specific amount of time



And decrease the idle counter when idle



Now we need to give the enemies vision so they can see the player and shoot if we are in their area of sight. We can achieve this by using rectangles attached to the enemy

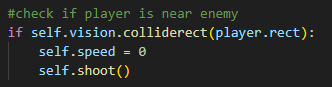




We can temporarily make the rectangles visible to see the area of sight



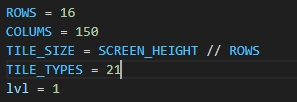
As we can see, the enemy now has an area of sight marked by the red rectangle

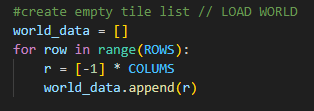


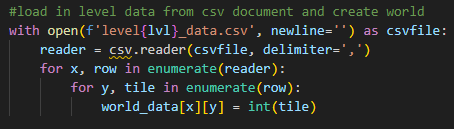
Now if their line of sight interacts with the player’s rect, they’ll stop and begin shooting

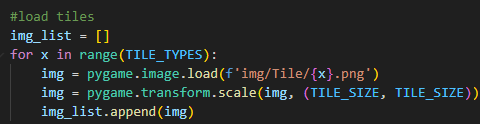
Loading world data

Set up

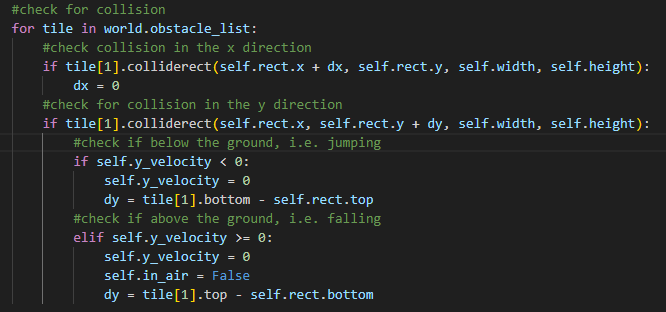






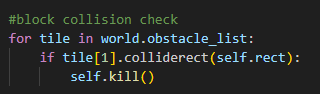




Check collision with blocks



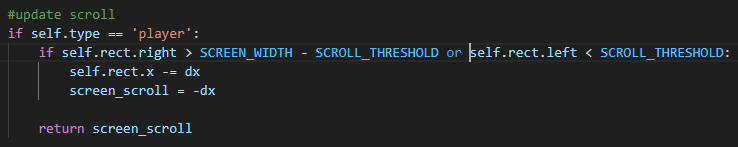
Check bullet collision with blocks



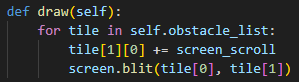
Loaded background



Begin screen scroll



Update draw function in world class to move tiles based on screen scroll



Main tiles have successfully been moved

Adding this across all world tiles





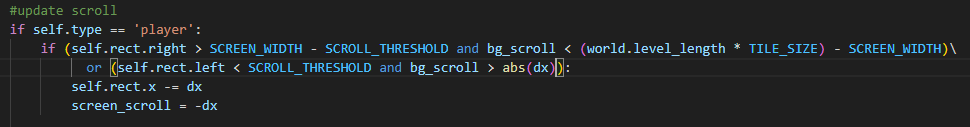


World doesn’t stop scrolling when you reach the end

We can find length of world

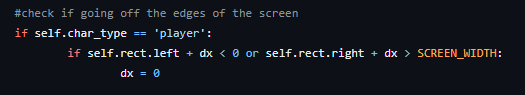


Then use it to modify the update scroll part of the move method



World no longer moves when player is at edge, but player can walk off edge



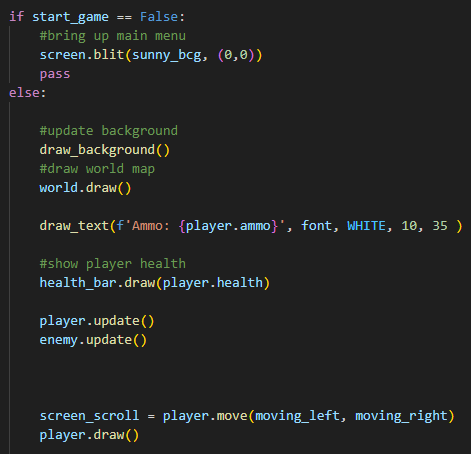


Player can now no longer walk off

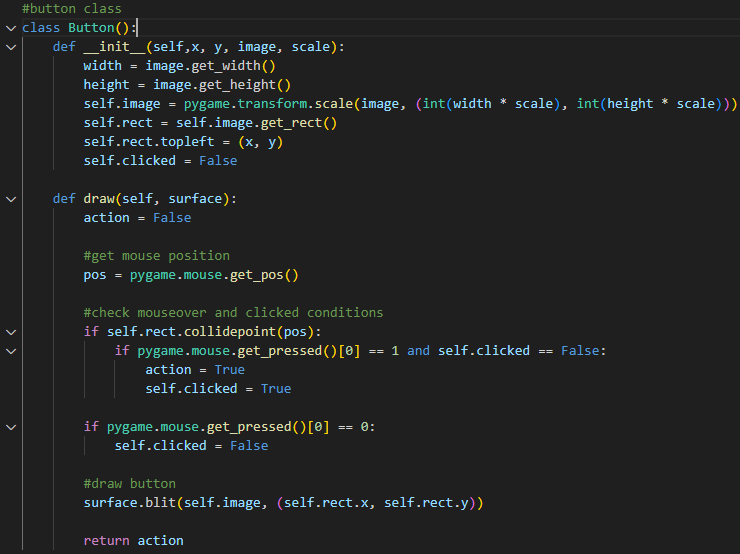
Create a start game Boolean function



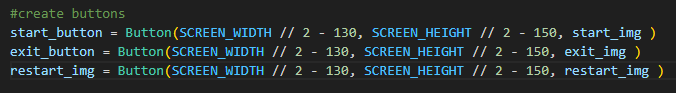
Only run rest of game loop if start\_game == True



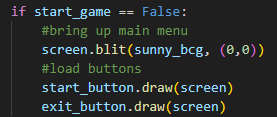
Create a button class for buttons



Create button objects



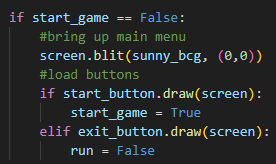
Draw buttons on menu



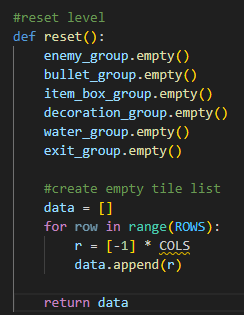
Buttons successfully loaded, but do not yet take input/output



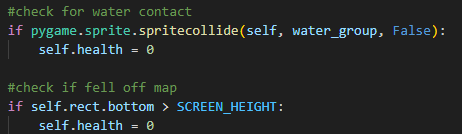
Added button output



Create map reset function



Add more death conditions





Restart level button shows up upon death, and works