**GRAVITY:** def update(self):

Works but not really accurate as it doesn’t create a gradual ascent for the player when jumping.

self.rect.y += self.gravity

if touching == True:

self.gravity = 0

else:

self.gravity = 1

#if self.gravity > 0:

# self.direction = 'down'

# self.gravity -= 1

An attempt to create a gradual ascent with a direction variable that I gave up on as I didn’t see how it would end up working

#elif self.gravity < 0:

# self.direction = 'up'

# self.gravity += 1

#elif self.gravity == 0 and self.direction == 'down':

# self.gravity += 1

#elif self.gravity == 0 and self.direction == 'up':

# self.gravity -= 1

if touching == True:

self.gravity = 0

elif touching == False:

This created the effect I wanted as once the player isn’t on the ground it adds to the gravity gradually pulling the player down to the ground.

if self.gravity == 0:

self.gravity = 1

else:

self.gravity += .35

def jump(self):

player.gravity = -8

**PLAYER GOIING THORUGH BLOCKS:**

for block in contact\_list:

touching = True

player.rect.y = block.rect.y - 30

#player.gravity = 0

if player.gravity > 0:

#player.gravity = -1

player.rect.bottom = block.rect.top

elif player.gravity < 0:

#player.gravity = 1

if player.speed > 0:

player.rect.right = block.rect.left

elif player.speed < 0:

player.rect.left = block.rect.right

player.rect.top = block.rect.bottom

player.gravity = 0

**CREATING A MAP OF CRUMBLING BLOCKS**

#for x in break\_list:

I used to use two lists for my blocks this one decreased the life of the blocks in a crumbling list however I realised this was a bad idea especially if I were to read a json file. It would be better if all the read blocks would be crumbling and I would add separately different blocks outside the json loop.

# touching = True

# player.vertical(-1)

# crumbling\_block.life = crumbling\_block.life - 1

#cb2.life += -1

# if crumbling\_block.life < 0:

# all\_sprites\_list.remove(crumbling\_block)

# all\_sprites\_list.update()

#elif cb2.life < 0:

# all\_sprite\_list.remove(cb2)

for block in contact\_list:

if block.dying == True:

block.life += -1

if block.life < 0:

all\_sprites\_list.remove(block)

all\_sprites\_list.update()

# all\_sprite\_list.update()

for y in range(20):

for x in range(54):

if level01[y][x] == 1:

num += 1

The count number ended up huge as it would iterate through level loop and going through my count loop each time a 1 is found in the json file. Also I’m not using the ordered list I created to my advantage

val = x\*30

ordered.append(val)

ordered.sort()

for i in range(num):

count += 10

print(count)

my\_block = Block(GREY, 30, 30, x\*30, y\*30, count, True)

block\_list.add(my\_block)

all\_sprites\_list.add(my\_block)

for y in range(20):

for x in range(54):

if level01[y][x] == 1:

Problem here was that my creating blocks loop was separated from the main loop so it didn’t have the x and y values of the blocks it needed to create. Still not using the ordered list.

num += 1

val = x\*30

ordered.append(val)

ordered.sort()

print(num)

if num == 32:

for i in range(num):

count += 10

print(count)

my\_block = Block(GREY, 30, 30, x\*30, y\*30, count, False)

block\_list.add(my\_block)

all\_sprites\_list.add(my\_block)