```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
1   # Lair of Doom
2   # Code Angel
3
4   # Classes: Player
5
6   import pygame
7
8   import level
9   import screen
10  import lair_of_doom
11
12  GRAVITY = 0.6
13  MOVEMENT_SPACE = 5
14  JUMP_VELOCITY = 10
15
16
17  # Player class
18  class Player:
19
20      def __init__(self):
21          self.image = lair_of_doom.load_media('image', 'player')
22
23          self.diamond_sound = lair_of_doom.load_media('audio', 'diamond')
24          self.completed_sound = lair_of_doom.load_media('audio', 'level_completed')
25          self.game_over_sound = lair_of_doom.load_media('audio', 'game_over')
26
27          self.velocity = 0
28          self.diamonds_collected = 0
29
30          self.jumping = False
31          self.jumping_left = False
32          self.jumping_right = False
33
34          self.game_over = False
35          self.game_completed = False
36
37          self.rect = pygame.Rect(0, 0, level.BLOCK_SIZE, level.BLOCK_SIZE)
38
39      # set the player location at the start of the game
40      def set_location(self, location):
41          x_coord = location[0] * level.BLOCK_SIZE
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
42          y_coord = location[1] * level.BLOCK_SIZE
43          self.rect = pygame.Rect(x_coord, y_coord, level.BLOCK_SIZE, level.BLOCK_SIZE)
44
45      # Move the player
46      def move(self, dx, dy):
47
48          # Move the player dx pixels horizontally, dy pixels vertically
49          self.rect.x += dx
50          self.rect.y += dy
51
52          # Keep the player from going off left and right edges of the screen
53          if self.rect.x >= screen.SCREEN_WIDTH - level.BLOCK_SIZE:
54              self.rect.x = screen.SCREEN_WIDTH - level.BLOCK_SIZE
55          if self.rect.x <= 0:
56              self.rect.x = 0
57
58          # Check if the moving player has hit a ledge
59          for ledge in level.ledges:
60              if self.rect.colliderect(ledge.rect):
61
62                  # player moving right hits the left edge of a ledge, so don't move any further
63                  if dx > 0:
64                      self.rect.right = ledge.rect.left
65
66                  # player moving left  hits the right edge of a ledge, so don't move any further
67                  if dx < 0:
68                      self.rect.left = ledge.rect.right
69
70                  # player jumping hits the bottom of a ledge so stop going up and start falling
71                  if dy < 0:
72                      self.rect.top = ledge.rect.bottom
73                      self.velocity = 0
74
75                  # player falling lands on a ledge, so stop falling down
76                  if dy > 0:
77                      self.rect.bottom = ledge.rect.top
78                      self.jumping = False
79                      self.jumping_left = False
80                      self.jumping_right = False
81                      self.velocity = 0
82
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```



```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
83          # Check there is a ledge directly below the player. If so, ledge_below should be True
84          self.rect.y += MOVEMENT_SPACE
85          ledge_below = False
86          for ledge in level.ledges:
87              if self.rect.colliderect(ledge.rect):
88                  ledge_below = True
89
90          self.rect.y -= MOVEMENT_SPACE
91
92          # If there is no ledge below start falling
93          if ledge_below is False and self.jumping is False:
94              self.jumping = True
95              self.velocity = 0
96
97      # The player starts a jump
98      def start_jump(self):
99          self.jumping = True
100         self.velocity = JUMP VELOCITY
101
102     # The player is still jumping and will continue to jump until they land on a ledge
103     def jump_move(self):
104
105         # Update velocity by gravity, then move the player up/down by the value of the velocity
106         self.velocity -= GRAVITY
107         self.move(0, -self.velocity)
108
109         # When jumping left/right the player moves at half the normal speed horizontally
110         if self.jumping_right is True:
111             self.move(MOVEMENT_SPACE / 2, 0)
112         elif self.jumping_left is True:
113             self.move(-MOVEMENT_SPACE / 2, 0)
114
115     # Has the player collided with a doom monster?
116     def check_doom_monsters(self):
117         for monster in level.doom_monsters:
118             if self.rect.colliderect(monster.rect):
119                 if self.game_over is False:
120                     self.game_over = True
121                     self.game_over_sound.play()
122
123     # Has the player hit water?
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
124     def check_water(self):
125         for wave in level.water:
126             if self.rect.colliderect(wave.rect):
127                 if self.game_over is False:
128                     self.game_over = True
129                     self.game_over_sound.play()
130
131     # Has the player collected a diamond?
132     def check_diamonds(self):
133         for diamond in level.diamonds:
134             if self.rect.colliderect(diamond.rect):
135                 level.diamonds.remove(diamond)
136                 self.diamonds_collected += 1
137                 self.diamond_sound.play()
138
139     # Has the player found an exit?
140     def check_exit(self, game_level):
141         for door in level.exit_doors:
142             if self.rect.colliderect(door.rect):
143
144                 # If an exit has been found, level up
145                 game_level.level_up()
146                 self.completed_sound.play()
147                 if game_level.level_number < 4:
148                     game_level.set_up()
149                     self.set_location(game_level.player_start_loc)
150                 else:
151                     self.game_completed = True
152
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```