```python
import pygame
import random
import sys

pygame.init() #Initializes pygame module

#Loading the background image
screen_width = 1000 #Assigning pixels for screen width
screen_height = 500 #Assigning pixels for screen height
bg_x = 0
background_image = pygame.image.load("assets/background.png") #Background image
background_image = pygame.transform.scale(background_image, (screen_width,screen_height)) #Changing image dimensions to fit the whole screen
screen = pygame.display.set_mode((screen_width, screen_height)) #Creating variable for screen
pygame.display.set_caption("Realm quest") #Game name
enemies = []
player_bullets = []

score = 0
player_lives = 3
font = pygame.font.Font(None, 36)
game_over_font = pygame.font.Font(None, 64)#GAME OVER - When player's life comes down to 0

#Player 1
class Character:
   def _init_(self,x,y): #Constructor - Called everytime an object is created
      self.x = x
      self.y = y
      self.img = pygame.image.load("assets/player1.png") #Loading the image
      self.img = pygame.transform.scale(self.img, (100,100)) #Transforming the image size
      self.rect = self.img.get_rect() #Creating rectangle outside the image - Makes it easy to change coordinates of the image
      self.rect.center = (x,y) #Center fo rectangle
      self.run_animation_count = 0 #To keep track of the image being loaded
      #Each player image is a different pose
      self.img_list = ["assets/player1.png","assets/player2.png","assets/player3.png","assets/player4.png"] #List to store addresses of all the player 1 images
      self.is_jump = False
      self.jump_count = 15 #First 15 iterations, player goes up and next 15 iterations, player comes down
      self.bullet_img = 'assets/bullet.png'

   #This function is called every time a character is drawn
   def draw(self) : #self variable indicates that this function belongs to the class Character
       self.rect.center = (self.x,self.y)
       #screen.blit(self.img,(self.x,self.y))
       screen.blit(self.img,self.rect)

   #Function to make the player 1 run
   def run_animation_player(self):
    if(not(self.is_jump)): #Run animation occurs only when the player is not jumping
       self.img = pygame.image.load(self.img_list[int(self.run_animation_count)])
       self.img = pygame.transform.scale(self.img, (100,100))
       self.run_animation_count += 0.5 #For smooth running of player 1
```

```
        self.run_animation_count =  self.run_animation_count % 4
#self.run_animation_count cannot exceed 3, list index is until 3

    def jump(self): #Player jump is a parabolic path
        if(self.jump_count >-15): #-15 to 15 - 30 count
            n = 1 #Player going up - Decreasing y-coordinate
            if(self.jump_count<0):
                n = -1 #Player coming down - Increasing y-coordinate
            self.y -= ((self.jump_count**2)/10) * n
            self.jump_count -= 1
        else :
            self.is_jump = False #The player do not jump
            self.jump_count = 15
            self.y = 386 #Resetting y-coordinates of the player

    def shoot(self):
        bullet = Bullet(self.x+5, self.y-18, self.bullet_img)
        player_bullets.append(bullet)

#Enemy
class Enemy:
    def _init_(self,x,y): #Constructor - Called everytime an object is created
        self.x = x
        self.y = y
        self.img = pygame.image.load("assets/enemy1.png") #Loading the image
        self.img = pygame.transform.scale(self.img, (75,75)) #Transforming the image
size
        self.rect = self.img.get_rect() #Creating rectangle outside the image - Makes
it easy to change coordinates of the image
        self.rect.center = (x,y) #Center fo rectangle
        self.run_animation_count = 0 #To keep track of the image being loaded
        self.img_list =
["assets/enemy1.png","assets/enemy2.png","assets/enemy3.png","assets/enemy4.png"]
#List to store addresses of all the player 1 images
        self.is_jump = False
        self.jump_count = 15 #First 15 iterations, player goes up and next 15
iterations, player comes down

    #This function is called every time a character is drawn
    def draw(self) : #self variable indicates that this function belongs to the class
Character
        self.rect.center = (self.x,self.y)
        #screen.blit(self.img,(self.x,self.y))
        screen.blit(self.img,self.rect)

    #Function to make the player 1 run
    def run_animation_enemy(self):
        self.img = pygame.image.load(self.img_list[int(self.run_animation_count)])
        self.img = pygame.transform.scale(self.img, (80,80))
        self.run_animation_count += 0.5 #For smooth running of player 1
        self.run_animation_count =  self.run_animation_count % 3
#self.run_animation_count cannot exceed 3, list index is until 3


class Bullet:
    def _init_(self,x,y,img):
        self.x = x
        self.y = y
        self.img = pygame.image.load(img)
        self.img = pygame.transform.scale(self.img,(15,15))
```

```
        self.rect = self.img.get_rect()
        self.rect.center = (x,y)

    def draw(self):
        self.rect.center = (self.x, self.y)
        screen.blit(self.img, self.rect)

    def move(self,vel):
        self.x += vel #Position of the bullet increases with velocity

    def off_screen(self):
        return(self.x<=0 or self.x>=screen_width)


player = Character(100,386) #Dimensions of Player 1
running = True #Game is running
clock = pygame.time.Clock() #Clock object to perform time-related functions
speed_increase_rate = 0
last_enemy_spawn_time = pygame.time.get_ticks()

#For running the game
while running: #Loop runs when running variable is true
    score += 1 #Score increases as long as the game is running
    for event in pygame.event.get(): #Has information of all the events - User clicking the start button
       if event.type == pygame.QUIT: #User clicks on cross button (QUIT game)
           running = False #Game is not running
       if event.type == pygame.KEYDOWN: #Returns true when any key is pressed
           if event.key == pygame.K_SPACE: #Checks if spacebar is pressed
               player.is_jump = True
           if event.key == pygame.K_RIGHT: #Bullet is shot if the rigth arrow is clicked
               player.shoot()

   #Code for moving background
    speed_increase_rate += 0.004
    bg_x -= (10 + speed_increase_rate) #Decreasing x-coordinate of the first image, to increase the speed of bg image, change the number 10
    if bg_x <= -screen_width:
        bg_x = 0
    screen.blit(background_image,(bg_x,0)) #To print image onto the screen with coordinate (0,0)
    screen.blit(background_image,(screen_width + bg_x,0))

    current_time = pygame.time.get_ticks()
    if(current_time - last_enemy_spawn_time >= 3000): #Time interval between appearances of the enemy. Enemy spawns after 3 seconds
        if random.randint(0,100) < 3: #If greater than 3, enemy does not appear on the screen ; 0,1,2 - Enemy appears
            enemy_x = screen_width + 900
            enemy_y = 386
            enemy = Enemy(enemy_x,enemy_y)
            enemies.append(enemy) #All the enemies generated are appended to a list
            last_enemy_spawn_time = current_time

    for enemy in enemies:
       enemy.x -= (15 + speed_increase_rate)
       enemy.draw()
       enemy.run_animation_enemy()

```

```
157     #Collision 1
158        if enemy.rect.colliderect(player.rect):
159          speed_increase_rate = 0
160          player_lives -= 1 #Player's life is deducted every time collision happens
161          enemies.remove(enemy)
162 
163     #Collision 2
164        for bullet in player_bullets:
165            if pygame.Rect.colliderect(enemy.rect, bullet.rect):
166                player_bullets.remove(bullet)
167                enemies.remove(enemy)
168                score += 10 #Player score increases when bullet collides the enemy
169 
170     for bullet in player_bullets:
171         if(bullet.off_screen()):
172             player_bullets.remove(bullet) #If bullet is offscreen, bullet is removed
    from player bullets
173         else:
174             bullet.draw()
175             bullet.move(10)
176 
177     if player_lives <= 0:
178         game_over_text = game_over_font.render("Game Over",True,(255,255,255)) #
    (255,255,255) - Font colour is white
179         screen.blit(game_over_text,(screen_width//2 - 120, screen_height//2))
180         pygame.display.update()
181         pygame.time.wait(2000)
182         pygame.quit()
183         sys.exit()
184 
185 
186     #Display the scores and lives of the player
187     live_text = font.render(f"Lives : {player_lives}",True,(0,0,0))
188     screen.blit(live_text,(screen_width-120,10)) #Displays the lives in the top
    right corner
189     score_text = font.render(f"Score:{score}",True,(0,0,0))
190     screen.blit(score_text,(20,10)) #Displays the score in the top left corner
191 
192     if(player.is_jump):
193         player.jump()
194 
195     player.draw() #To draw the image, to display player on the screen
196     player.run_animation_player() #To make the player run
197     pygame.display.update() #Update the display
198     clock.tick(30) #This loop runs 30 times for a second. Upper limit of fps is 30
199     #This is to run the loop in the same speed
200     #More functions inside while loop, slower the execution speed
201 
202 pygame.quit()
203 
```