Let’s get started! Let’s get some skeleton code and resources from:

Basic stuff: importing the libraries we are going to need. Start every PyGame project with:

```
import pygame
```

For this project we will also need the time and random libraries. Add these statements directly after the first import

```
import time
import random
```

Immediately after, we will initialize an instance of the game! We can do this by typing

```
pygame.init()
```

As you can see, the display dimensions are kept as variables defined at the beginning of the code! This way, we can easily modify them in just one place! Set them to a value between 700 and 900. Also set the caption to give your game a title!

```
display_width =
display_height =
car_width =

#showing the game box and title
gameDisplay = pygame.display.set_mode((display_width,display_height))
pygame.display.set_caption('???')
```

Let’s load a sprite by having pygame load the 'race_car.png image. The car_width variable will be important later, so let’s set it to the width of this picture, which is 73 (px)

We have a pair of colors defined by their RGB values here

```
color1 = (0,0,0)
color2 = (255,255,255)
```

Right now they correspond to black and white. You can set them to any colors you want, and rename the variables too! You can pick colors from https://rgbcolorcode.com/ or from a tool like MS Paint

I am going to rename color1 to sand, and make it my background by changing

```
gameDisplay.fill(???)
```

to

```
gameDisplay.fill(brown)
```

Now we need to modify the game loop to allow movement. We do this by defining what happens when the player presses a key. We are going to add the following code to the event loop

```
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                x_change =
            elif event.key == pygame.K_RIGHT:
                x_change =
        if event.type == pygame.KEYUP:
            if event.key == pygame.K_LEFT or event.key ==
pygame.K_RIGHT:
                x_change = 0
```

This will go in the same indentation level as

```
if event.type == pygame.QUIT:
```

Careful! Python uses indentation and spacing as a way to track scope, instead of braces

Back in our code before, we specify what the x_change ought to be! You can modify it for you game but two things are important:

a) The change should be negative to move left, and positive to move right
b) The value itself, other than the sign, should be the same
c) If a the left arrow or right arrow key has bee released, the change becomes zero and we stop moving!

I am doing -5 and +5

To enforce this change, right before the car(x,y) call, we add

```
x += x_change
```

Speaking of! So far we seem to be making a racecar game. What if we don't want to do that? We can find sprites online that fit better with our game design vision, but the size should be kind of similar. I am going to use this Mad Max like car instead, to go with my desert colored map. This is simply changing the filename loaded at

```
sprite = pygame.image.load('race_car.png')
```

Now, the game object we control is of car type. You can rename it anything you want, but you have to rename every instance of it! Also if you change the sprite you have to change the car_width variable!

Now the object is moving, but there are no obstacles! Let’s make some

Looking at the code, we see a things function

```
def things(thingx, thingy, thingw, thingh, color):
    pygame.draw.rect(gameDisplay, color, [thingx, thingy, thingw, thingh])
```

This function uses the pygame.draw method to create an object, and move it towards the player at a defined speed. To call this function, we need to specify what values we want in terms of where to create it, at what speed should it move, etc.

Right before car(x,y) we add

```
things(thing_startx, thing_starty, thing_width, thing_height, color)
thing_starty += thing_speed
```

The second line is super important, as it updates the start position to change as the car moves. Here we use the random library to find a random position horizontally. Here are possible values to use:

```
    thing_startx = random.randrange(0, display_width)
    thing_starty = -600
    thing_speed = 7
    thing_width = 100
    thing_height = 100
```

Feel free to experiment with different values! Also use colors you have defined earlier! These statements need to go **before** we call the things() constructor. Let’s place them at the top of the game loop

Now, after this we add:

```
        if thing_starty > display_height:
            thing_starty = 0 - thing_height
            thing_startx = random.randrange(0,display_width)
```

This lets us create multiple obstacles instead of just the one!

I am going to create steel barriers, so I will set the rectangles to be wider and for the color to be steely

Now, the game runs, but it is not challenging at all! We need to add object collisions and crashes

Let's add:

```
if x > display_width - car_width or x < 0:
       gameExit = True
```

Right by the updating part of our game loop. What this does is cause a game over if the car goes against the edges of the screen!

The barrier collision is a bit more complicated, and looks like this:

```
if y < thing_starty+thing_height:
          if x > thing_startx and x < thing_startx + thing_width or
x+car_width > thing_startx and x + car_width <
thing_startx+thing_width:
                gameExit = True
```

Improving the game….

Let's say we don't want to load the game each time we die, but want to just start again. Let's make it so crashing does not change the gameExit Boolean, but instead calls a new crash() function

```
def crash():
    game_loop()
```

This let's us start the game over! But maybe we can give it a cooldown. Let's add

```
time.sleep(t)
```

To the crash(x) method. Replace x with the number of seconds you want to wait!

What else can we do on a crash? We can display text, but that is outside the scope of this class. Let's instead put a sound bit!

In our crash method, let's put:

```
    pygame.mixer.Sound.play(crash_sound)
    pygame.mixer.music.stop()
```

We need to determine what crash_sound it. Delete the # that marks the line as a comment where this is defined. Fill in the space with the filename. It can be any .ogg file currently in your directory. I am going to use a good old Wilhem Scream

Now to add the finishing touch, let’s make the game play some music!

Let’s add, at the top of our game loop:

```
    pygame.mixer.music.load(“???”)
    pygame.mixer.music.play(-1)
```

Fill in the space with any sound clip you want!