Modifying the environment
I started with the Car Racing environment from OpenAI Gymnasium. I decided to modify this environment into a space-themed game. I created two new classes in a file named custom_carRacing.py. The first class is named CustomCarRacing and inherits from the original CarRacing environment. Functions like __init__(), reset(), _init_colors(), and _render_indicators() are overwritten in the new class to change the background color to black and the track color to blue. This class also changes the environment from using the car class to using the rocket class.
The rocket class inherits from the original car class but overrides the __init__() function to remove the wheels from the car and change its color from red to light gray to make it look more like a spaceship or rocket.

I encountered several challenges while changing the car into a rocket to fit the space theme. I originally tried to remove the car and replace it with a rocket by using different polygons in the code (the car's shape is built using polygons in the code) and leaving the wheels invisible. It would be harder to remove the wheels since they were the pieces of the car that had physics applied. When I did this the track started to shake when the game was running. I don't know what caused this issue.

I also tried to make the rocket by removing the wheels and applying the physics logic of one wheel to the rocket. This seemed to work but the rocket didn't turn sharp enough to make it around corners. I modified the values I thought would change the turn radius but it didn't.

My last attempt at changing the car to a rocket involved leaving the car in the game without printing it out and putting an image on top of it. I was able to create the invisible car but had issues getting an image to stay at the same location as the car. I believe this issue came from a difference in the way location coordinates are handled in Box2D and Pygame.

Implementing the agent
The agent uses Q learning and can be found in the agent.py file. It includes functions to get an action from the agent, update the policy weights of the agent, save the agent's policy in an XML file, or load the agent's policy from an XML file.
Rewards and penalties for the agent are already implemented in the original car racing environment. Moving to new track tiles is rewarded while each frame the agent takes adds a slight penalty.
The weights are stored in a Python dictionary that uses images as keys and a list of weights for each move as values. These weights are updated after each move made by the agent while training. The images used as keys have been reduced from 96x96 pixels to 16x16. I did this to reduce noise in the data and decrease the number of states the agent needs to learn.
I used the openAI gymnasium example to write the Q learning agent. The example was made for openAI's blackjack environment. The example made it a lot easier to apply the Q learning algorithm to the modified car racing environment.
I was able to use chatGPT to create load and save functions to keep an agent's trained policy in a file and load an existing policy from a file. This saved some work figuring out how to complete this function.

The wrapper
In the wrapper.py file, I wrote functions to train and test an agent as well as play the game without the agent. The training function goes through a specified number of training simulations on the environment and updates the agent's policy after each move. When a simulation is

completed, the epsilon value is reduced to make it more likely that the agent will choose a move based on its policy instead of choosing a random move to explore the environment.
The wrapper class also contains a function to test the agent once it has been trained. The function loads the policy from a file and then runs it once on the environment to show the user how the agent performs.
The user can also play the game without the agent using the playGame() function. This takes input from the user's keyboard to control the rocket on the screen. The up arrow makes the rocket accelerate, the down arrow makes the rocket stop, and the left and right arrows make the rocket turn in the corresponding directions.

The main file
The main file should be run to start the main menu. This menu allows the user to load an agent's policy from a file to test it, train and store a model, or play the game. The menu was created very quickly using chatGPT. This saved a lot of time and created a menu that allowed me to easily connect the functions from the wrapper class to the buttons on the menu.

Current performance of the agent
The model I trained doesn't perform very well in the environment. After 1000 simulations, the model has learned to go straight but hasn't learned to turn on a corner. I believe this may be because it hasn't been trained in enough simulations. It could also be the case that rewards need to be modified to improve training.

ChatGPT use
I used chatGPT in a few different ways while working on this project. It helped me understand parts of the example I was modifying to work with the modified racing environment instead of the blackjack environment. ChatGPT also helped me understand how I could create a custom environment similar to car racing. I was able to save time on the load and save functions in the agent class when chatGPT made these functions that worked without modification. The menu was also created by chatGPT and saved me the time I would have spent arranging different buttons and text boxes on the tkinter window. While chatGPT was very helpful for small pieces of code, it was less helpful for larger pieces of code. When trying to look for things in the car racing environment I had to narrow it down to a function or two since I couldn't copy and paste in all the code (this may be easier by sending the file with paid versions of chatGPT). Another way to use chatGPT is to find errors in your code when using libraries you aren't familiar with. chatGPT was helpful when trying to correct my code using the openAI gym library however there were issues it was unable to solve. ChatGPT wasn't as helpful when I was trying to put a picture of a rocket on top of the race car sprite.