1. The purpose of our program is of course for entertainment, but it was also to explore some of the principles of artificial intelligence. The idea was to get a use, (the player) to play against an AI opponent. The AI would navigate itself through the course using rules we programmed it to follow. The program also heavily relates to principles of object oriented programming, and interaction between modules or classes. To make the entire program work, many different classes had to exist independently to handle different parts of the problem.

2. The main class uses several methods that involve many intricate series of statements. Take for example the eventChecker() method from the ERacers.py file:

   ```
   def eventChecker():
           # Complicated else-if statement
   ```

   And it is called frequently in the main game loop. This is so that, when we wish to check for an event, we can give the specifics of that task to something that handles it independently, so our code does not have to be confusing and cluttered. An entire Actors class had to be made, where the car class held the information and compatibility with pygame, while only circles appeared on the GUI.

3. The most complex algorithm in the program ended up not involving AI, but rather the detection of each increasing lap. If we were to simply check if the X and Y of each car, and add a lap every time the car was in the finish line area, then the lap counter would ascend rapidly, since there are many frames where the car is in the finish zone. Therefore, we had to create a mechanism where it would only add or subtract a lap once the car entered the finish zone, and wouldn't do it again until the car exited the finish zone and entered again.

4. The first step was to learn pygame. It wasn't difficult to find tutorial videos on how to start, and prior coding knowledge allowed us to fill in the blanks through pygame's documentation. Then we had to create an image for the background, and learn how to set it as the background. Using GIT to control our progress, we slowly added the player and CPU sprites, added keyboard control, then a pause function, then a collision detector, and finally a mechanism to restart or end the game after someone has won. Each step had its own set of unique concepts to learn and problems to overcome. Most of the problems, however, were a result of overzealous development without testing more frequently.

5. Since I was much more familiar with Python and coding in general, I was in charge of implementation, while Grant was in charge of conceptual work and design. Grant would decide how the game should run and proceed in development, while I translated the prototype into executable code. Often times, Grant would assist in coding as well, whenever a bug was giving me a particularly difficult time.