**Error Correction**

Record in the tables below any corrections you made while you were writing your programme. Use one table for each error you found and corrected. You need to include the date, line the error was in, what was wrong and how you corrected it.

An example has been given here:

| Date | 29 May |
| --- | --- |
| Line error found on | Line 6 |
| Incorrect code | number = int(input("Please input a number between 1 and 5") |
| What was wrong with the code? | I forgot to put in the last bracket and it came up with a syntax error. |
| Corrected code | number = int(input("Please input a number between 1 and 5")) |
| What was the outcome of making the changes? | Once I added it in the programme correctly asked for the user to choose a number between 1 and 5. |

| Date | Sep 4, 2025 |
| --- | --- |
| Line error found on | 60, in [main.py](http://main.py) file |
| Incorrect code | norm\_pos\_vec = np.array(car.pos) - np.array(norm\_car.pos) |
| What was wrong with the code? | The vector should be pointing from the car aligned at 0,0 to the target car. The variable names were confusing and also had to be changed. |
| Corrected code | norm\_pos\_vec = np.array(target\_car.pos) - np.array(normalized\_car.pos) |
| What was the outcome of making the changes? | The vector was correctly pointing to the correct car, meaning the corner and collision calculations worked. |

| Date | Aug 12, 2025 |
| --- | --- |
| Line error found on | 80-83 in cars file |
| Incorrect code | self.pos[0] += float(self.time\_elapsed\*self.speed\*np.cos(self.car\_angle+1.570796327))  self.pos[1] += float(self.time\_elapsed\*self.speed\*np.sin(self.car\_angle+1.570796327))  if self.car\_angle >= 6.283185307: self.car\_angle -= 6.283185307  if self.car\_angle <= -6.283185307: self.car\_angle += 6.283185307 |
| What was wrong with the code? | It was using full number radians, instead of the present math libraries representation of pi, making the lines of code much more complex then they need to be. |
| Corrected code | self.pos[0] += float(self.time\_elapsed\*self.speed\*np.cos(self.car\_angle+np.pi/2))  self.pos[1] += float(self.time\_elapsed\*self.speed\*np.sin(self.car\_angle+np.pi/2))  if self.car\_angle >= 2\*np.pi: self.car\_angle -= 2\*np.pi  if self.car\_angle <= 0: self.car\_angle += 2\*np.pi |
| What was the outcome of making the changes? | The code looks much better, and using numpy, a math library, makes it more integrated and easier to use. |

| Date | Aug 14, 2025 |
| --- | --- |
| Line error found on | 165 in cars file |
| Incorrect code | rotate = np.array([[np.cos(self.rotation), (-np.sin(self.rotation))], [np.sin(self.rotation), np.cos(self.rotation)]], dtype=np.float64) |
| What was wrong with the code? | The above rotation matrix does not correctly rotate the points passed in, it needs to rotate the desired vector into the frame of reference of the angle self.rotation, but rotating it in the opposite directrion. |
| Corrected code | rotate = np.array([[np.sin(self.rotation), (np.cos(self.rotation))], [-np.cos(self.rotation), np.sin(self.rotation)]], dtype=np.float64) |
| What was the outcome of making the changes? | The graphics of the scene now correctly rotate away from the focused cars rotation, making it look like the focused car’s png is rotating, where instead the scene is rotating about the car. |

| Date | Aug 15, 2025 |
| --- | --- |
| Line error found on | 201-237 in cars file |
| Incorrect code | m = ((first\_point[0]-second\_point[0])/(first\_point[1]-second\_point[1]))  y = (self.pos[1]+m\*self.pos[0]+(m\*\*2)\*second\_point[1]-m\*second\_point[0])/((m\*\*2)+1)  x = (m\*self.pos[1]+(m\*\*2)\*self.pos[0]+(m\*\*3)\*second\_point[1]-(m\*\*2)\*second\_point[0])/((m\*\*2)+1)-m\*second\_point[1]+second\_point[0]  else:  m = ((first\_point[1]-second\_point[1])/(first\_point[0]-second\_point[0]))  x = (self.pos[0]+m\*self.pos[1]+(m\*\*2)\*second\_point[0]-m\*second\_point[1])/((m\*\*2)+1)  y = (m\*self.pos[0]+(m\*\*2)\*self.pos[1]+(m\*\*3)\*second\_point[0]-(m\*\*2)\*second\_point[1])/((m\*\*2)+1)-m\*second\_point[0]+second\_point[1]  x\_lower\_bound = min(first\_point[0], second\_point[0])  x\_upper\_bound = max(first\_point[0], second\_point[0])  y\_lower\_bound = min(first\_point[1], second\_point[1])  y\_upper\_bound = max(first\_point[1], second\_point[1])  if m != 0:  if x > x\_upper\_bound: point = first\_point if x\_upper\_bound in first\_point else second\_point  elif x < x\_lower\_bound: point = first\_point if x\_lower\_bound in first\_point else second\_point  else: point = (x,y)  else:  if x\_lower\_bound == x\_upper\_bound:  if y > y\_upper\_bound: point = first\_point if y\_upper\_bound in first\_point else second\_point  elif y < y\_lower\_bound: point = first\_point if y\_lower\_bound in first\_point else second\_point  else: point = (x,y)  elif y\_lower\_bound == y\_upper\_bound:  if x > x\_upper\_bound: point = first\_point if x\_upper\_bound in first\_point else second\_point  elif x < x\_lower\_bound: point = first\_point if x\_lower\_bound in first\_point else second\_point  else: point = (x,y)  car\_point\_vec = np.array(self.pos) - np.array(point)  car\_point\_angle = np.arctan2(car\_point\_vec[1], car\_point\_vec[0])  seg\_vec = np.array(second\_point) - np.array(first\_point)  seg\_angle = np.arctan2(seg\_vec[1], seg\_vec[0])  result\_angle = car\_point\_angle - seg\_angle  if result\_angle < -np.pi: result\_angle += np.pi  if result\_angle > np.pi: result\_angle -= np.pi  distance = np.linalg.norm(car\_point\_vec)  result.append(distance if result\_angle > 0 else -distance)  result.append(point) |
| What was wrong with the code? | This uses normal line formula of y-y1=m(x-x1), but this has the problem where the gradient m cant be properly defined for vertical or horizontal bars, so instead, a parametric version is used. |
| Corrected code | t = ((first\_point[0]-self.pos[0])\*(first\_point[0]-second\_point[0])-(first\_point[1]-self.pos[1])\*(second\_point[1]-first\_point[1]))/((second\_point[1]-first\_point[1])\*\*2+(second\_point[0]-first\_point[0])\*\*2)  first\_point = np.array(first\_point)  second\_point = np.array(second\_point)  p = first\_point+t(second\_point - first\_point)  if abs(first\_point[1]-second\_point[1]) > abs(first\_point[0]-second\_point[0]):  dist\_sign = np.sign((p[0]-self.pos[0])/(second\_point[1]-first\_point[1]))  else:  dist\_sign = np.sign((p[1]-self.pos[1])/(first\_point[0]-second\_point[0]))  result.append(dist\_sign\*np.linalg.norm(p-np.array(self.pos)))  result.append(p) |
| What was the outcome of making the changes? | The code becomes much much simpler, achieving the exact same thing. |

| Date | Aug 16, 2025 |
| --- | --- |
| Line error found on | 18-29 in ai training file |
| Incorrect code | remain\_cars = len(cars)  while remain\_cars > 0:  for i, car in enumerate(cars):  if car.get\_alive():  car.tick(17, 0)  else:  #print(car.get\_reward())  genomes[i][1].fitness = car.get\_reward()  remain\_cars -= 1 |
| What was wrong with the code? | Since the cars stay in the cars array even if they are state = 0, they will keep incrmenting down the remain\_cars variable, even though no new cars have died, meaning it will reach zero with there still being alive cars, stopping training in its tracks. |
| Corrected code | have\_live = True  while have\_live:  have\_live = False  for i, car in enumerate(cars):  if not car.get\_alive(): continue  car.tick(17, 0)  have\_live = True  for i, car in enumerate(cars):  genomes[i][1].fitness = car.get\_reward() |
| What was the outcome of making the changes? | Now, only once all the cars have died will it exit the while loop, meaning the training goes on as long as there is a car alive, leading to much better results. |

| Date | Aug 17, 2025 |
| --- | --- |
| Line error found on | 3 in config-feedforward.txt |
| Incorrect code | fitness\_threshold = max |
| What was wrong with the code? | “max” isnt a correct setting for the fitness threshold configuration, it needs to be a int. |
| Corrected code | fitness\_threshold = 999999999999 |
| What was the outcome of making the changes? | The program no longer crashes and produces an error, and the ai will never hit the fitness threshold (ideal). |

| Date | Aug 24, 2025 |
| --- | --- |
| Line error found on | 61 main file |
| Incorrect code | t = clock.tick(FRAME\_RATE)  car.tick(t) |
| What was wrong with the code? | The ai’s were trained on a static time interval, so without the same static time intervals, they start doing weird things. |
| Corrected code | car.tick(17), reset back to 17 ms |
| What was the outcome of making the changes? | The ais correctly drive with no issues, as they have the same code as they did in training. |

| Date | Aug 22, 2025 |
| --- | --- |
| Line error found on | 44-46 graphics.py |
| Incorrect code | for event in pg.event.get():  s\_event = self.scene\_events()  if s\_event: |
| What was wrong with the code? | Forgot to pass in the event, useless other wise as it wont have the event caused by the player pressing keys |
| Corrected code | for event in pg.event.get():  s\_event = self.scene\_events(event)  if s\_event |
| What was the outcome of making the changes? | Events correctly work, recognising player input |

| Date | Aug 22, 2025 |
| --- | --- |
| Line error found on | 24-25 in graphics.py |
| Incorrect code | if scene == Scene.main\_menu: self.scene\_obj = MainMenu()  elif scene == Scene.game: self.scene\_obj = GameGraphics(cars) |
| What was wrong with the code? | The classes dont have the screen surface being passed in, so they wont be able to draw anything on the screen, and there will be an argument error. |
| Corrected code | if scene == Scene.main\_menu: self.scene\_obj = MainMenu(self.screen)  elif scene == Scene.game: self.scene\_obj = GameGraphics(self.screen, cars) |
| What was the outcome of making the changes? | Scenes now correctly have the screen object that they can draw on. |

| Date | Aug 15, 2025 |
| --- | --- |
| Line error found on | 205 in cars.py |
| Incorrect code | p = first\_point+t(second\_point - first\_point) |
| What was wrong with the code? | To get the correct p value, doing checks with the t value of the parametric function make sure that the correct point is selected. |
| Corrected code | if t < 0: p = first\_point  elif t > 1: p = second\_point  else: p = first\_point+t\*(second\_point - first\_point) |
| What was the outcome of making the changes? | the get\_data\_seg function works correctly |

| Date | Aug 8, 2025 |
| --- | --- |
| Line error found on | No line |
| Incorrect code | Problem: No requirements file |
| What was wrong with the code? | Without a requirements file, others using the program will find it hard to get the correct packages via pip, so adding a file makes that much easier as it simply becomes pip install -r requirements.txt |
| Corrected code | Added requirements file with all currently used packages |
| What was the outcome of making the changes? | It made it easier switching between devices and setting up virtual enviroments as i simply had to install the packages from one file. |