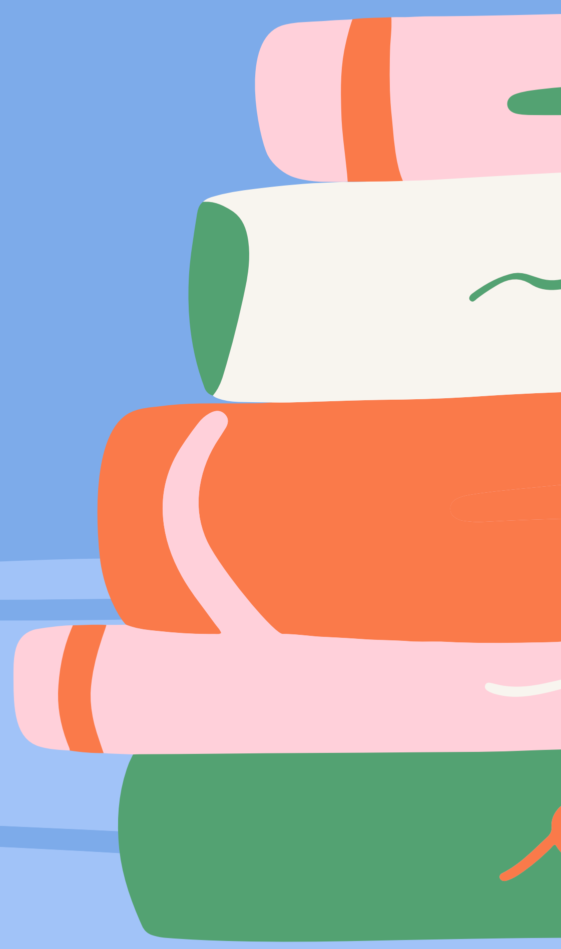
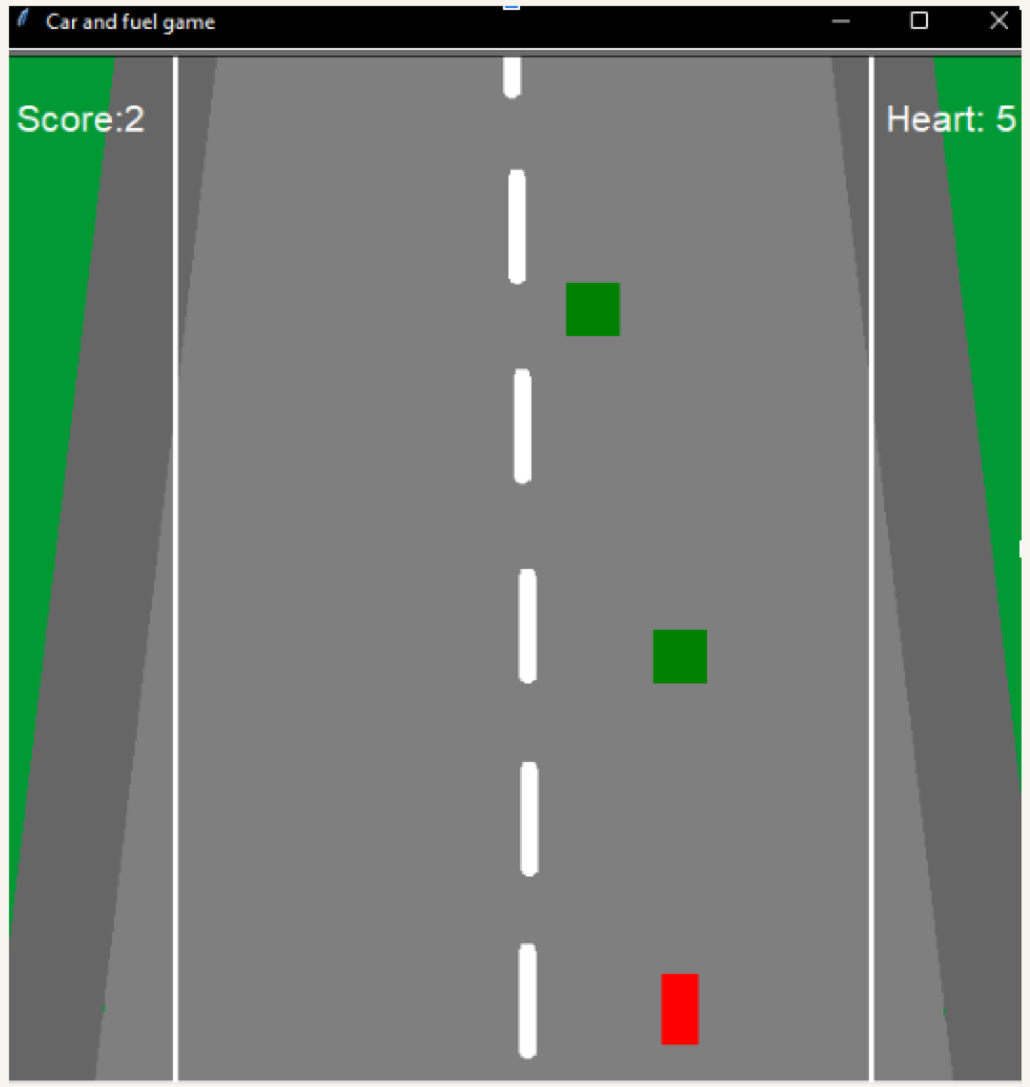
SOLFWARE AND KNOWLEDGE ENGINEERING

Car and Fuel game

by Wissarut
6510545721



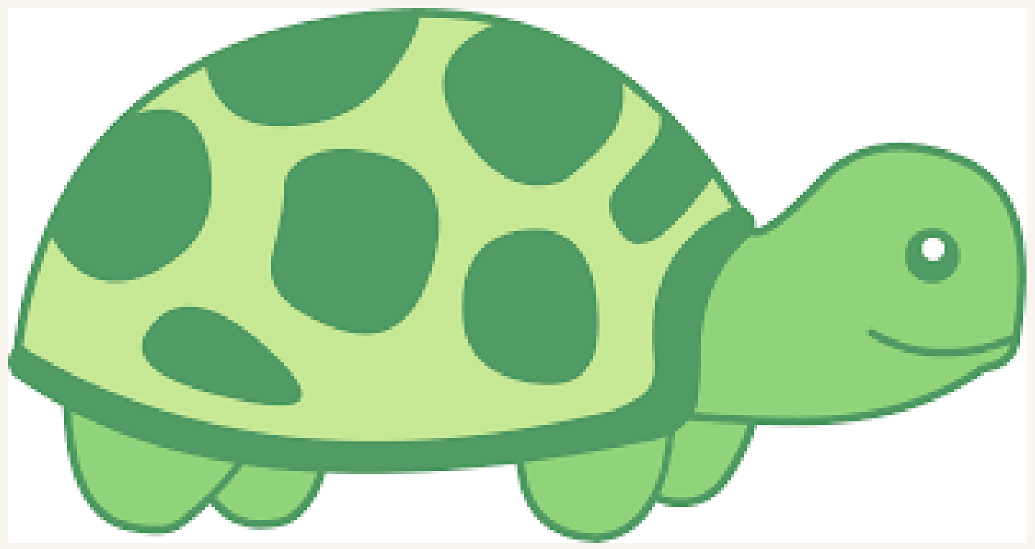
Introduction



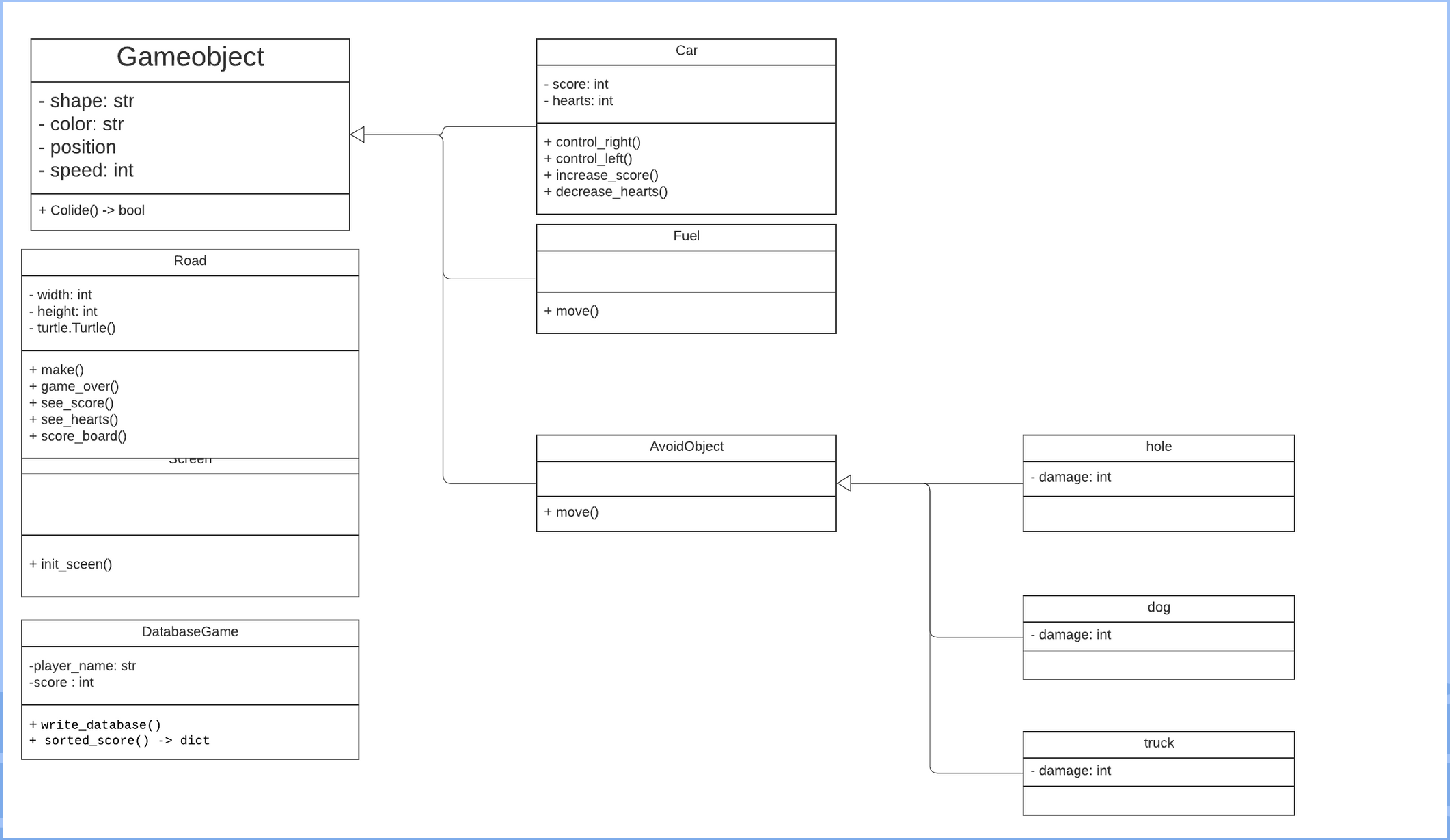
program demonstration

EXTRA FEATURES

- Git repository
- Turtle graphic
- Class inheritance



PROGRAM STRUCTURE

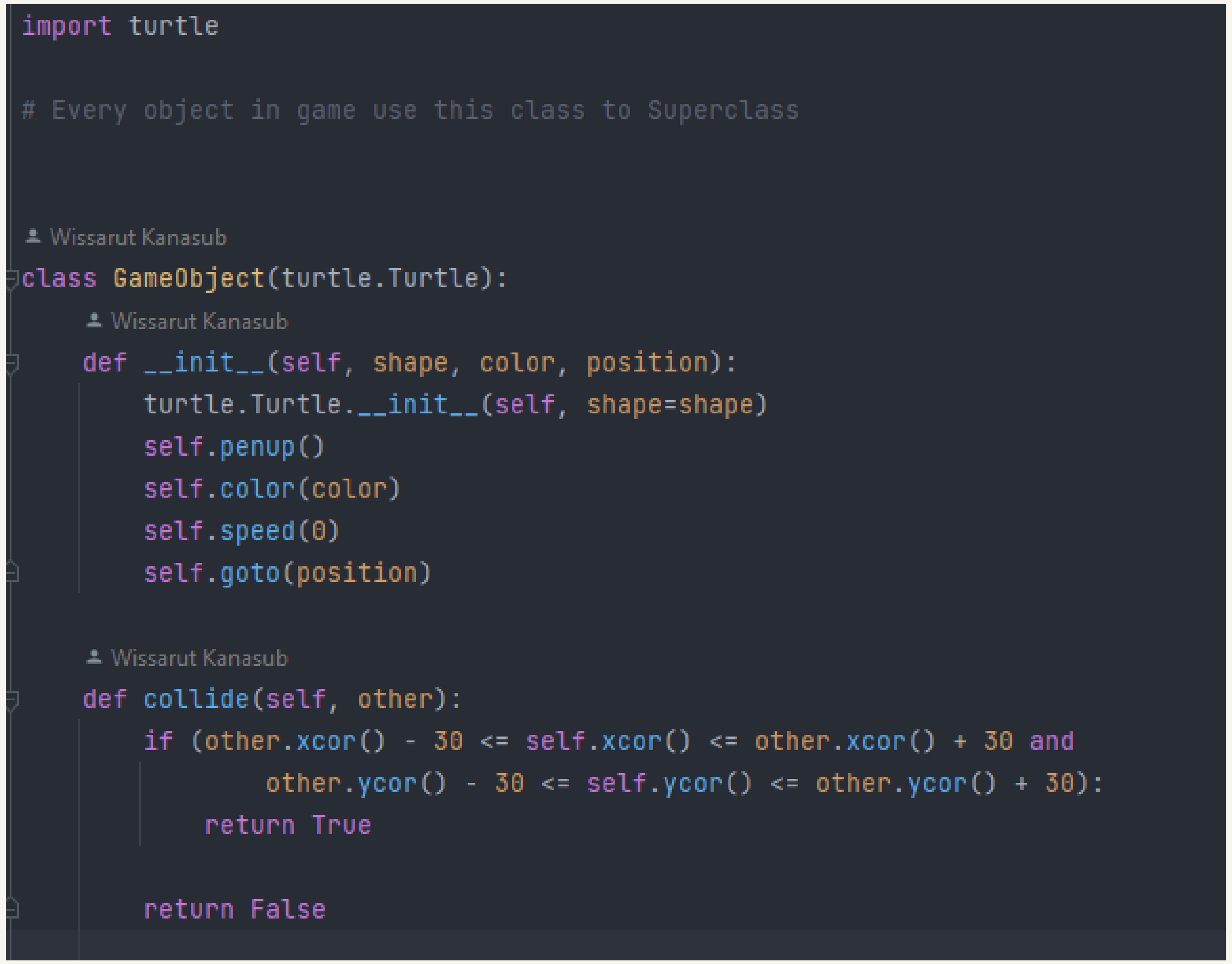


1.GameObject class (GameObject.py)

This class is an inheritance from Turtle.Superclass of all objects in the game that can move. Have 3 attributes is shape, color, position,

method

- **Collide(other) : check collide**



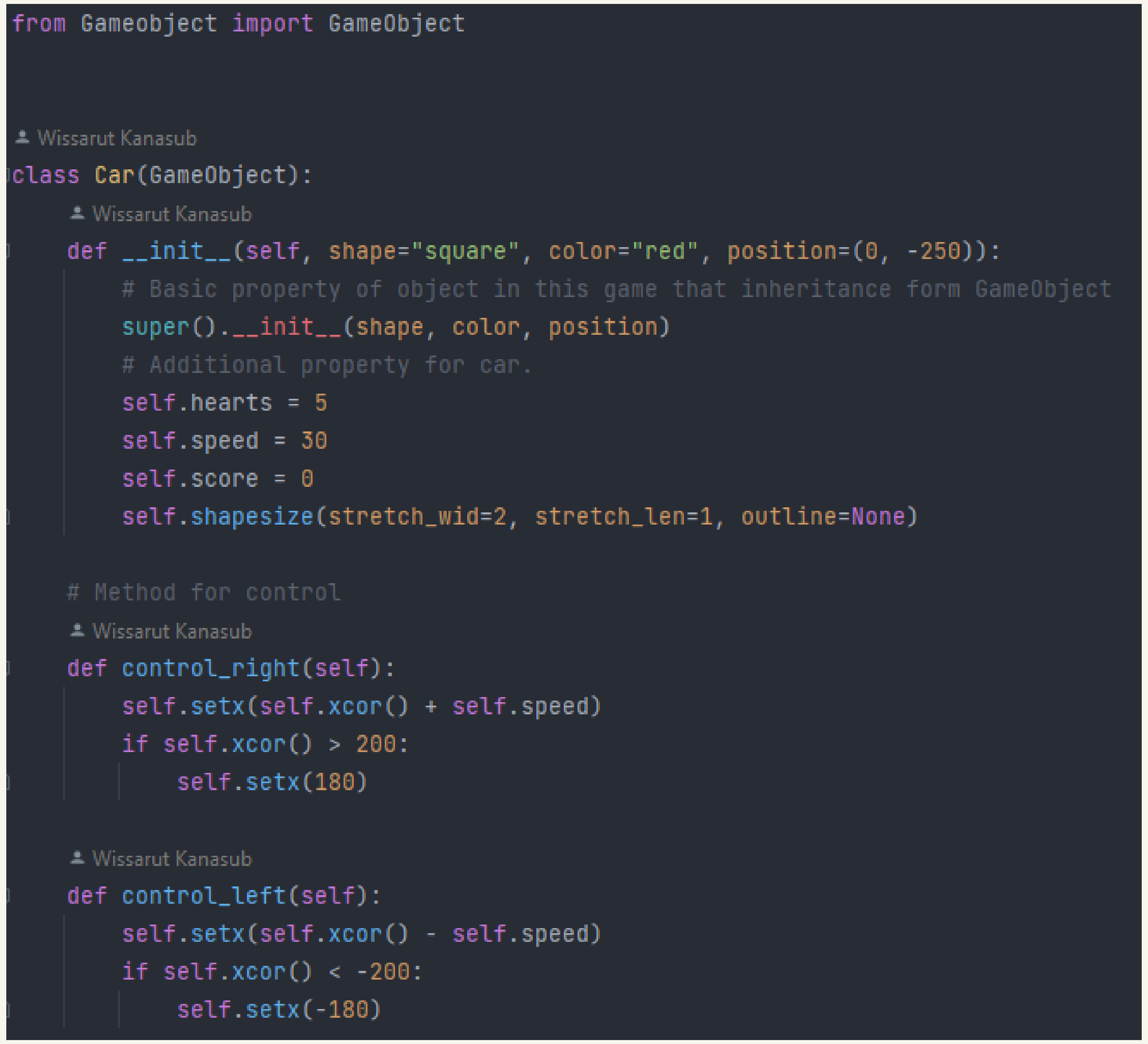
```python
import turtle

# Every object in game use this class to Superclass


class GameObject(turtle.Turtle):
    def __init__(self, shape, color, position):
        turtle.Turtle.__init__(self, shape=shape)
        self.penup()
        self.color(color)
        self.speed(0)
        self.goto(position)

    def collide(self, other):
        if (other.xcor() - 30 <= self.xcor() <= other.xcor() + 30 and
                other.ycor() - 30 <= self.ycor() <= other.ycor() + 30):
            return True

        return False
```

2.Car class (Car.py)

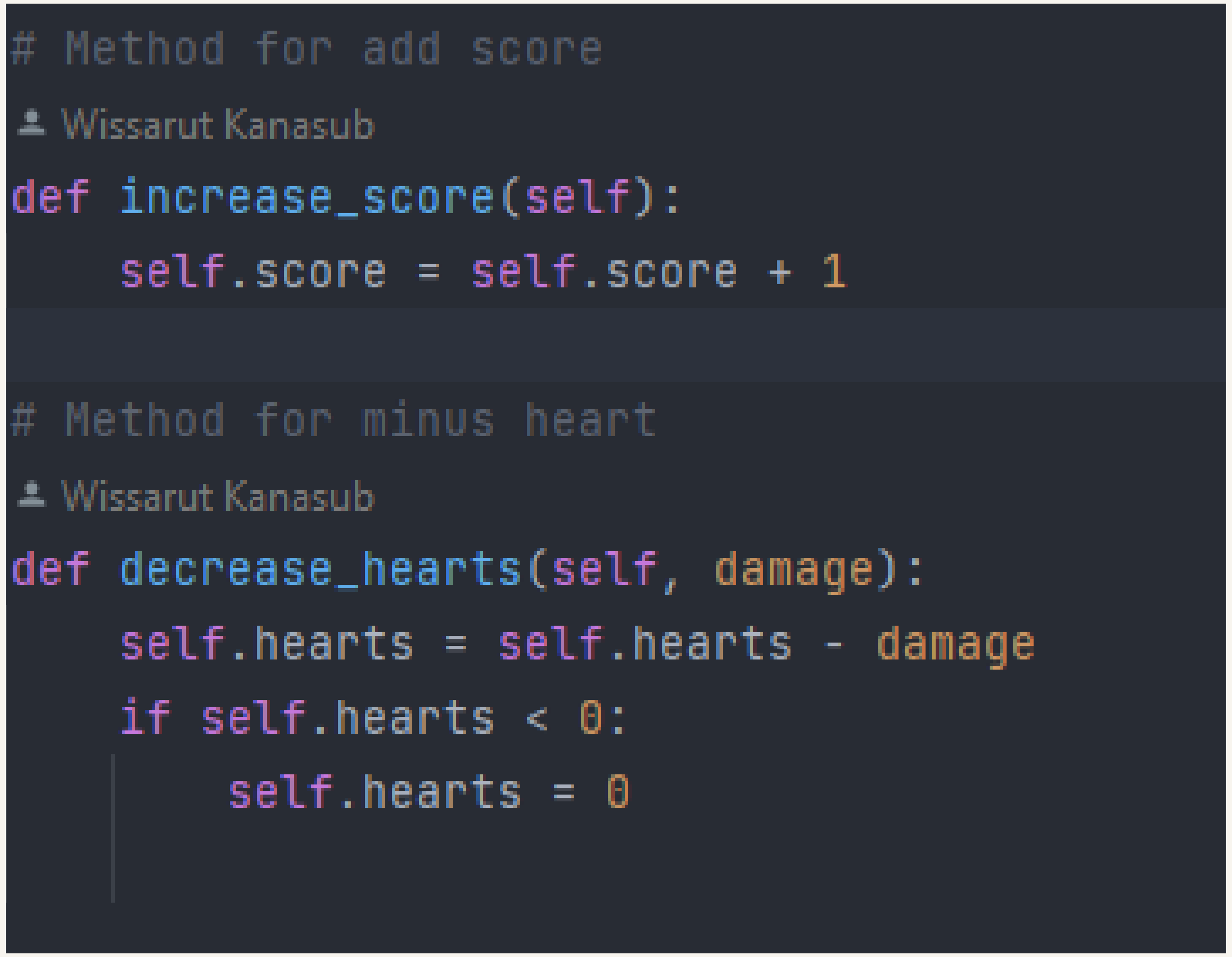
This class is a subclass of Gameobject but add more attribute are score: int, heart: int

method

- **control_left() : move car to left**
- **control_right() : move car to right**
- **decrease_hearts(damage) : decrease car hearts with damage**
- **increase_score(): increase score when car collide with fuel**



```
from Gameobject import GameObject


Wissarut Kanasub
class Car(GameObject):
    Wissarut Kanasub
    def __init__(self, shape="square", color="red", position=(0, -250)):
        # Basic property of object in this game that inheritance form GameObject
        super().__init__(shape, color, position)
        # Additional property for car.
        self.hearts = 5
        self.speed = 30
        self.score = 0
        self.shapesize(stretch_wid=2, stretch_len=1, outline=None)

    # Method for control
    Wissarut Kanasub
    def control_right(self):
        self.setx(self.xcor() + self.speed)
        if self.xcor() > 200:
            self.setx(180)

    Wissarut Kanasub
    def control_left(self):
        self.setx(self.xcor() - self.speed)
        if self.xcor() < -200:
            self.setx(-180)
```



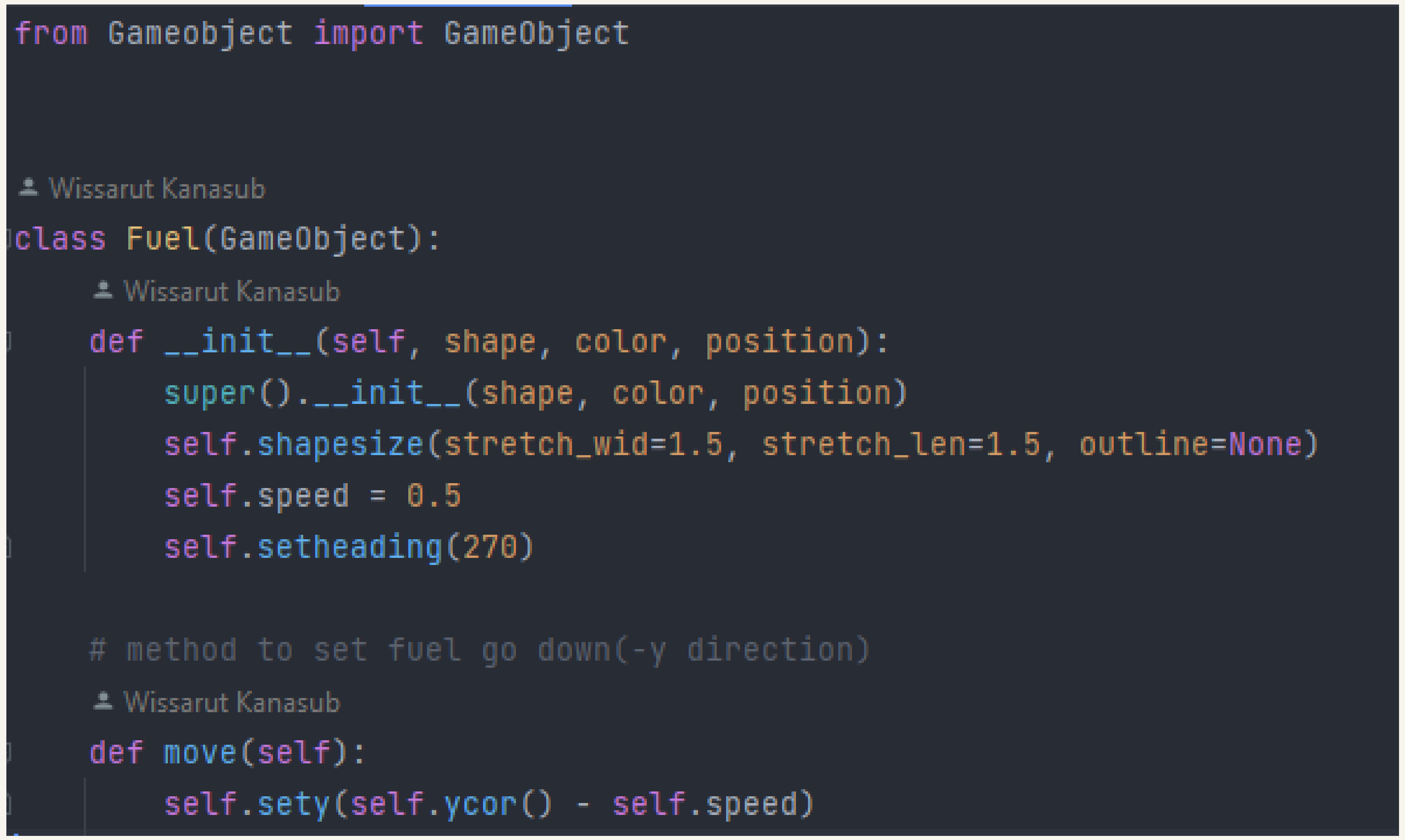
```
# Method for add score
Wissarut Kanasub
def increase_score(self):
    self.score = self.score + 1


# Method for minus heart
Wissarut Kanasub
def decrease_hearts(self, damage):
    self.hearts = self.hearts - damage
    if self.hearts < 0:
        self.hearts = 0
```

3.Fuel class (Fuel.py)

This class is a subclass of Gameobject too.

method

- **move(): move in direction (-y axis)**



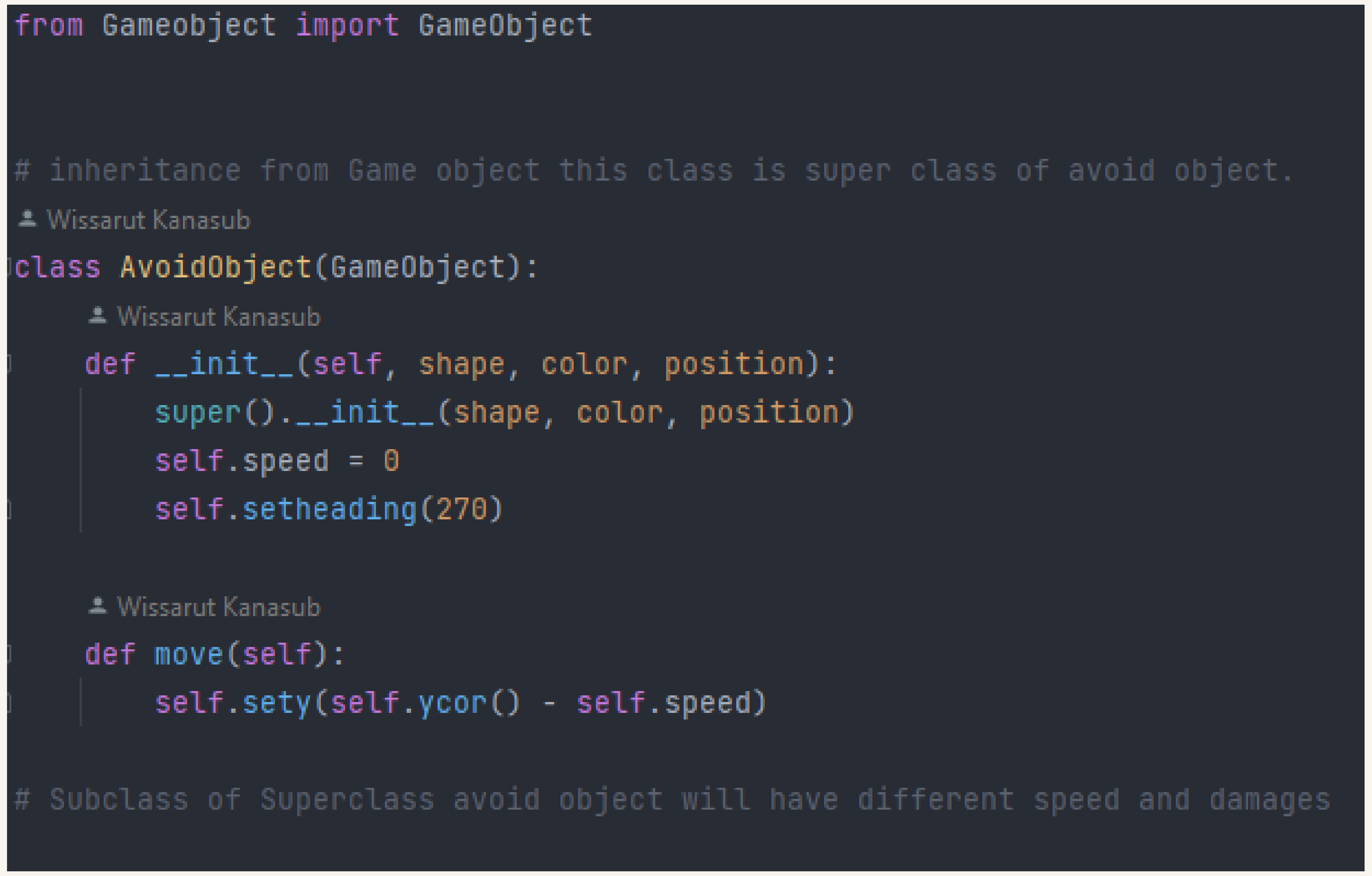
```
from Gameobject import GameObject


Wissarut Kanasub
class Fuel(GameObject):
    Wissarut Kanasub
    def __init__(self, shape, color, position):
        super().__init__(shape, color, position)
        self.shapesize(stretch_wid=1.5, stretch_len=1.5, outline=None)
        self.speed = 0.5
        self.setheading(270)

    # method to set fuel go down(-y direction)
    Wissarut Kanasub
    def move(self):
        self.sety(self.ycor() - self.speed)
```

4.AvoidObject class (AvoidObject.py)

This class is a subclass of Gameobject too. look like fuel class

method

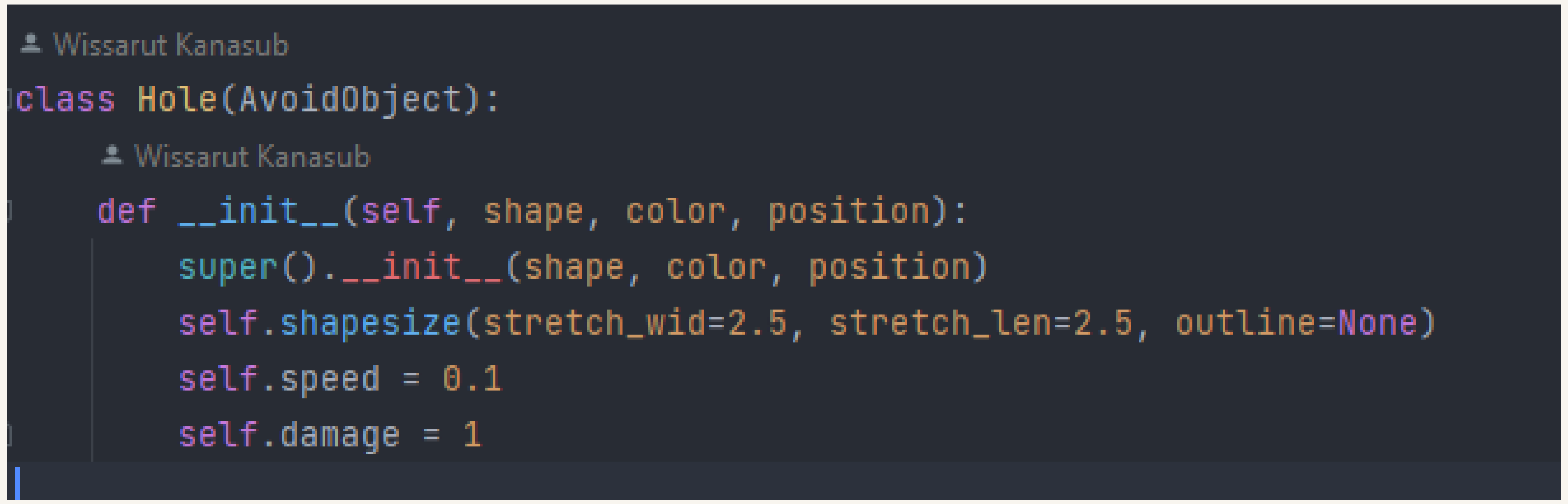
- **move(): move in direction (-y axis)**



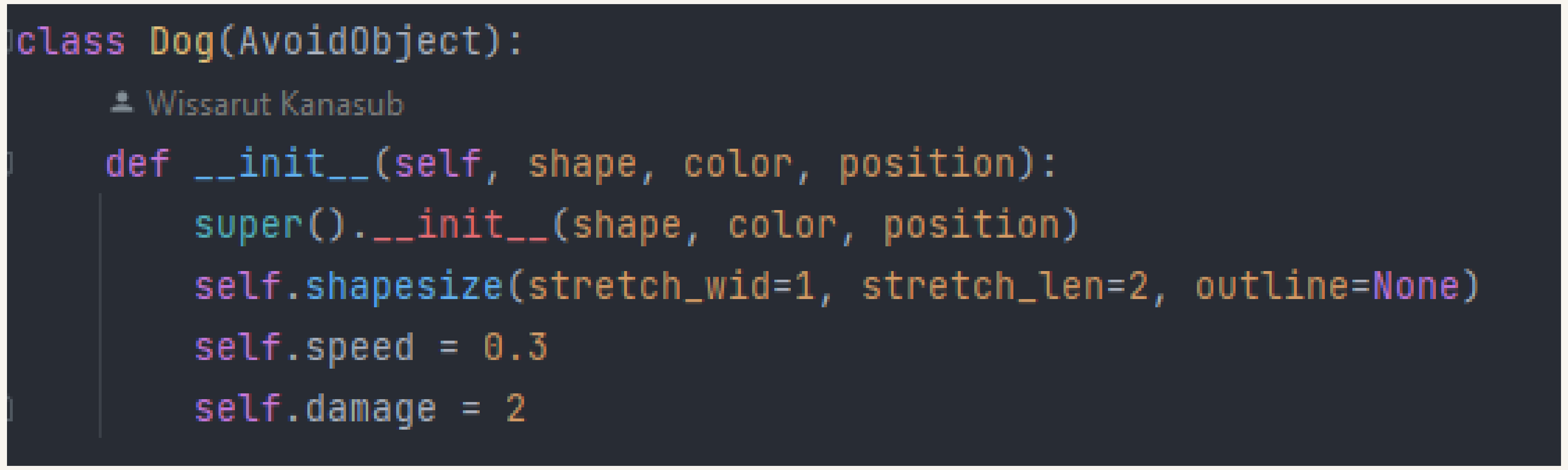
```
from Gameobject import GameObject


# inheritance from Game object this class is super class of avoid object.
Wissarut Kanasub
class AvoidObject(GameObject):
    Wissarut Kanasub
    def __init__(self, shape, color, position):
        super().__init__(shape, color, position)
        self.speed = 0
        self.setheading(270)

    Wissarut Kanasub
    def move(self):
        self.sety(self.ycor() - self.speed)

# Subclass of Superclass avoid object will have different speed and damages
```

(5 to 7).Dog, Truck, Hole: class (AvoidObject.py)

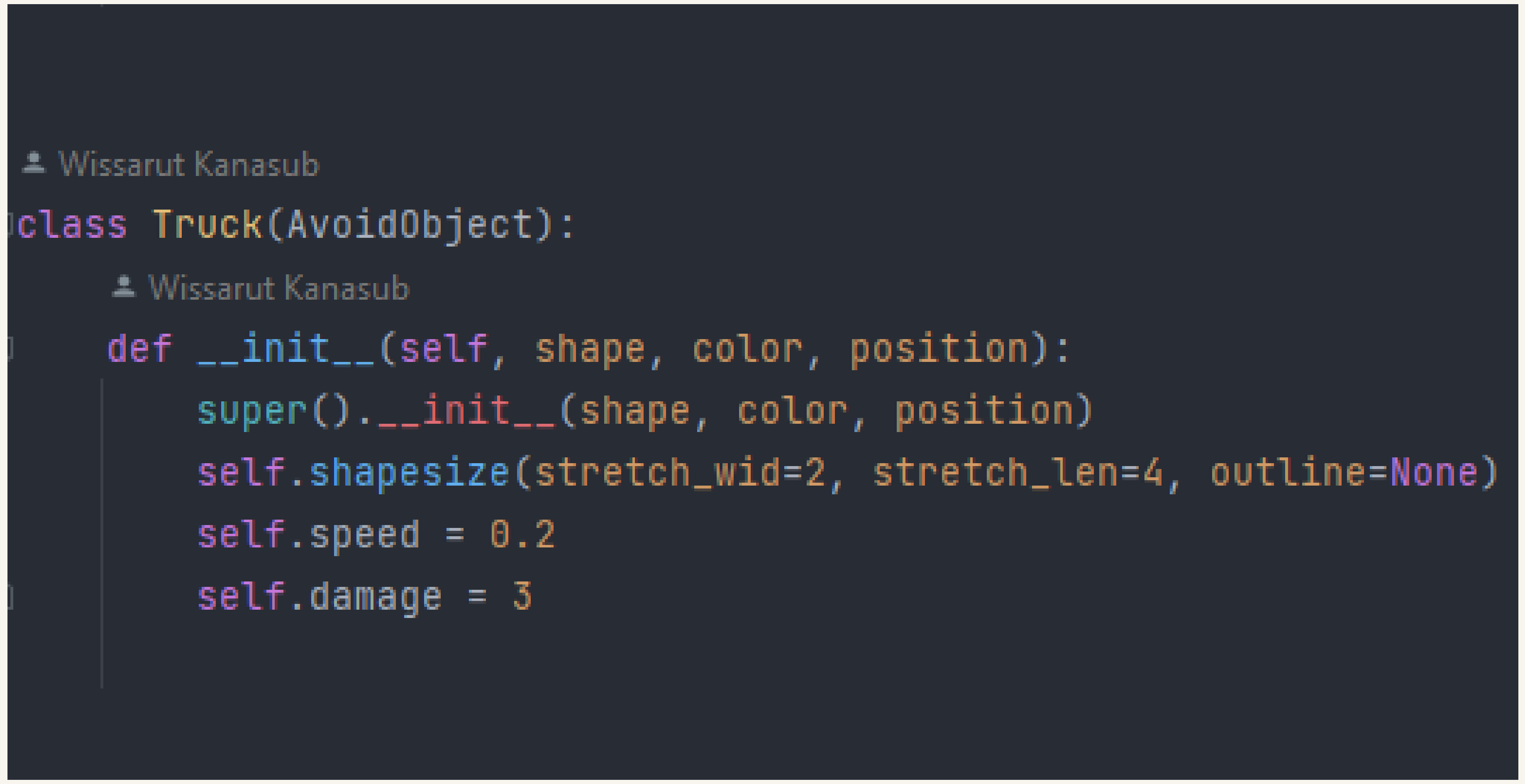
These Classes are subclass of AvoidObject class. I make it a subclass because it is simple and formal to read.
and an important attribute for this game is damage: int.



```
Wissarut Kanasub
class Hole(AvoidObject):
    Wissarut Kanasub
    def __init__(self, shape, color, position):
        super().__init__(shape, color, position)
        self.shapesize(stretch_wid=2.5, stretch_len=2.5, outline=None)
        self.speed = 0.1
        self.damage = 1
```



```
class Dog(AvoidObject):
    Wissarut Kanasub
    def __init__(self, shape, color, position):
        super().__init__(shape, color, position)
        self.shapesize(stretch_wid=1, stretch_len=2, outline=None)
        self.speed = 0.3
        self.damage = 2
```



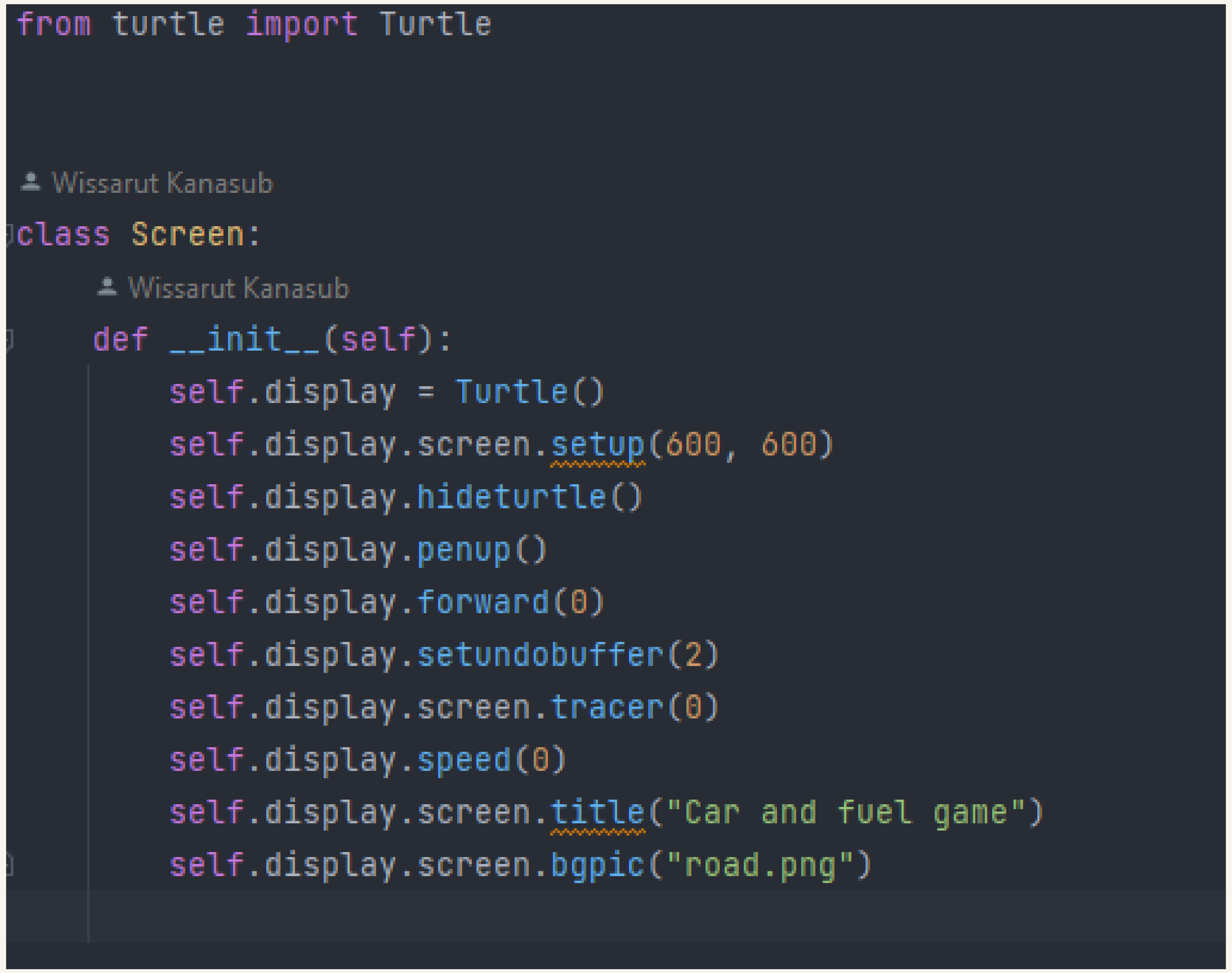
```
Wissarut Kanasub
class Truck(AvoidObject):
    Wissarut Kanasub
    def __init__(self, shape, color, position):
        super().__init__(shape, color, position)
        self.shapesize(stretch_wid=2, stretch_len=4, outline=None)
        self.speed = 0.2
        self.damage = 3
```

8.Screen class (screen.py)

This class objective is make a screen only

method

- **1.__init__() : make a screen with (600 x 600) size. and background picture**



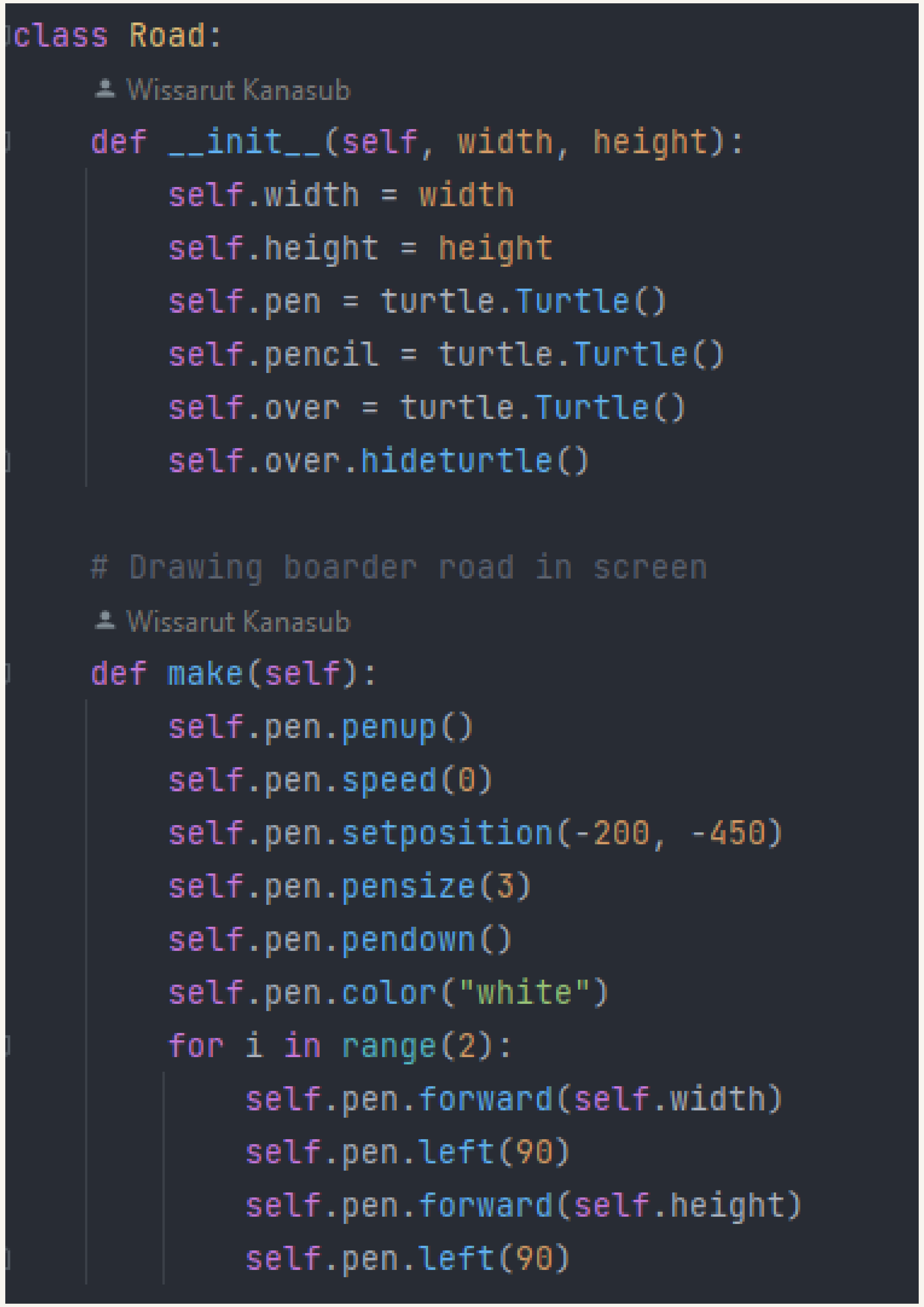
```
from turtle import Turtle


Wissarut Kanasub
class Screen:
    Wissarut Kanasub
    def __init__(self):
        self.display = Turtle()
        self.display.screen.setup(600, 600)
        self.display.hideturtle()
        self.display.penup()
        self.display.forward(0)
        self.display.setundobuffer(2)
        self.display.screen.tracer(0)
        self.display.speed(0)
        self.display.screen.title("Car and fuel game")
        self.display.screen.bgpic("road.png")
```

9.Road class (road.py)

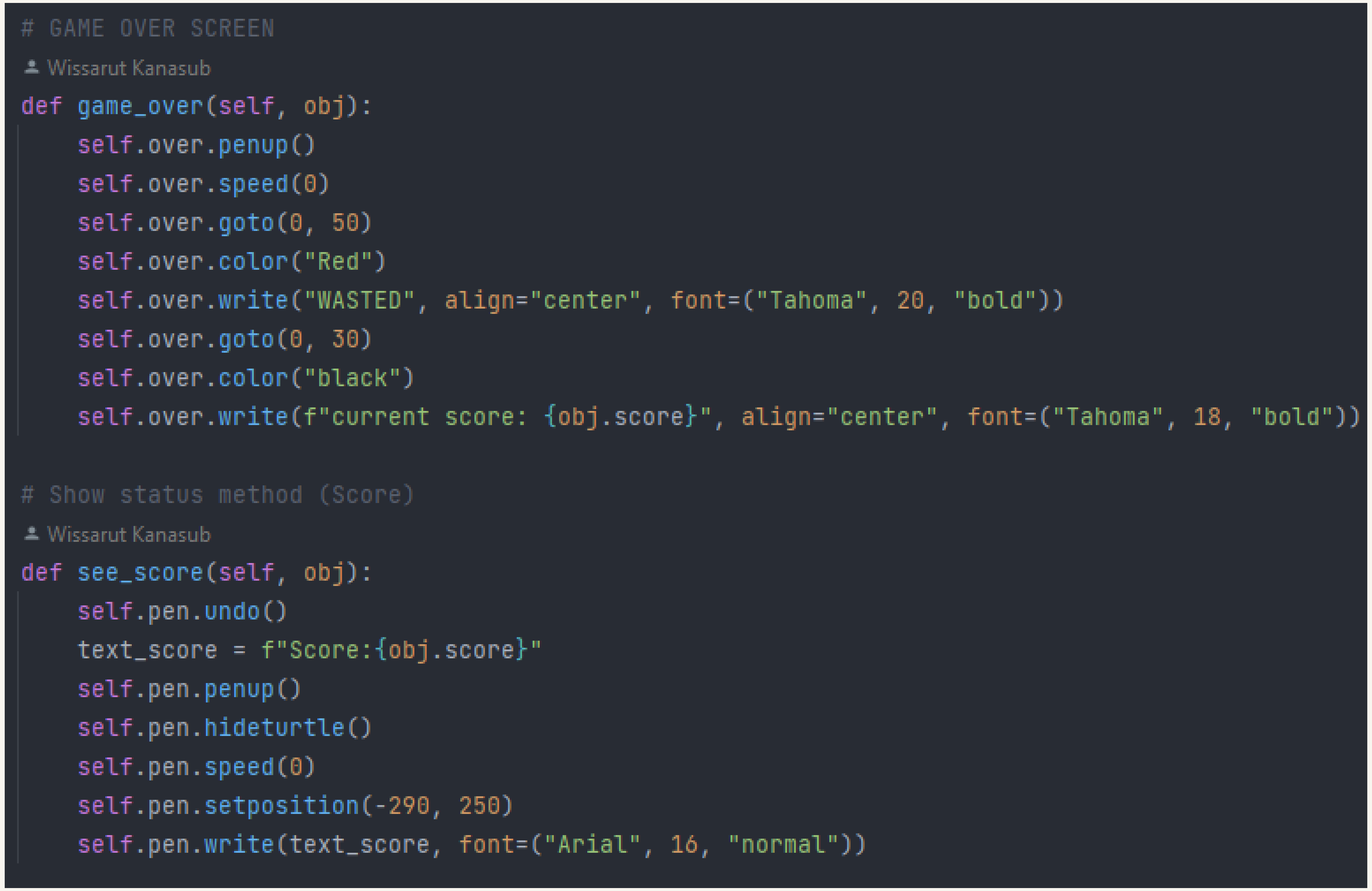
This class objective is operation all in screen without previous move objects such as show score, show heart, write road border with Height and Width, game over screen and top 3 score board

method

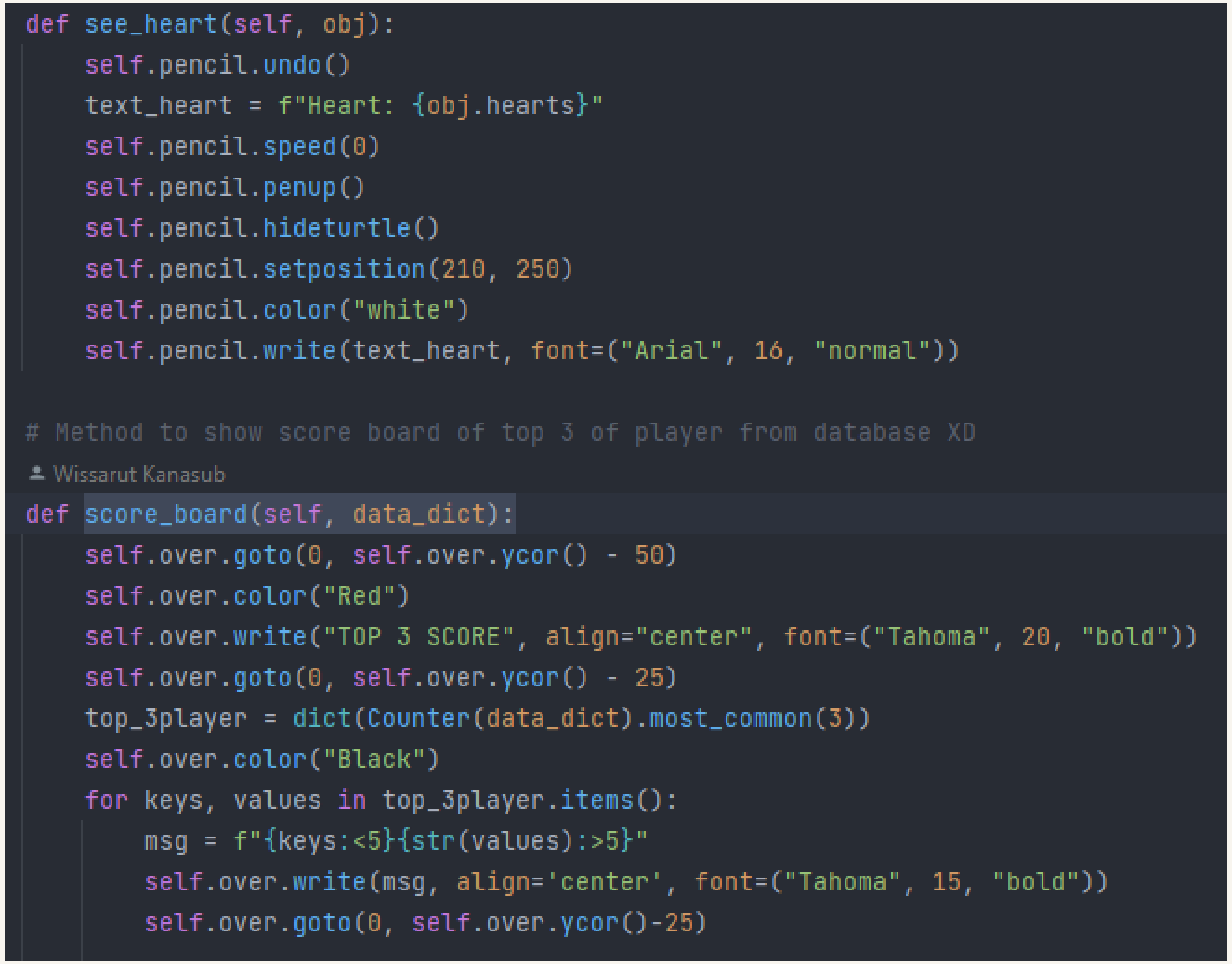
- 1.make(): writer parallel road border with 400 length
- 2.game_over(obj): show game over screen with current score of player name.
- 3.see_score(obj) : show score of player update real time on screen top left.
- 4.see_heart(obj): show hearts of player update real time on screen top right.
- 5.score_board(data_dict): show top 3 player scoreboard.
- data_dict input from DatabaseGame method



```
class Road:
    👤 Wissarut Kanasub
    def __init__(self, width, height):
        self.width = width
        self.height = height
        self.pen = turtle.Turtle()
        self.pencil = turtle.Turtle()
        self.over = turtle.Turtle()
        self.over.hideturtle()

    # Drawing boarder road in screen
    👤 Wissarut Kanasub
    def make(self):
        self.pen.penup()
        self.pen.speed(0)
        self.pen.setposition(-200, -450)
        self.pen.pensize(3)
        self.pen.pendown()
        self.pen.color("white")
        for i in range(2):
            self.pen.forward(self.width)
            self.pen.left(90)
            self.pen.forward(self.height)
            self.pen.left(90)
```



```
# GAME OVER SCREEN
👤 Wissarut Kanasub
def game_over(self, obj):
    self.over.penup()
    self.over.speed(0)
    self.over.goto(0, 50)
    self.over.color("Red")
    self.over.write("WASTED", align="center", font=("Tahoma", 20, "bold"))
    self.over.goto(0, 30)
    self.over.color("black")
    self.over.write(f"current score: {obj.score}", align="center", font=("Tahoma", 18, "bold"))

# Show status method (Score)
👤 Wissarut Kanasub
def see_score(self, obj):
    self.pen.undo()
    text_score = f"Score:{obj.score}"
    self.pen.penup()
    self.pen.hideturtle()
    self.pen.speed(0)
    self.pen.setposition(-290, 250)
    self.pen.write(text_score, font=("Arial", 16, "normal"))
```



```python
def see_heart(self, obj):
    self.pencil.undo()
    text_heart = f"Heart: {obj.hearts}"
    self.pencil.speed(0)
    self.pencil.penup()
    self.pencil.hideturtle()
    self.pencil.setposition(210, 250)
    self.pencil.color("white")
    self.pencil.write(text_heart, font=("Arial", 16, "normal"))

# Method to show score board of top 3 of player from database XD
```

Wissarut Kanasub

```python
def score_board(self, data_dict):
    self.over.goto(0, self.over.ycor() - 50)
    self.over.color("Red")
    self.over.write("TOP 3 SCORE", align="center", font=("Tahoma", 20, "bold"))
    self.over.goto(0, self.over.ycor() - 25)
    top_3player = dict(Counter(data_dict).most_common(3))
    self.over.color("Black")
    for keys, values in top_3player.items():
        msg = f"{keys:<5}{str(values):>5}"
        self.over.write(msg, align='center', font=("Tahoma", 15, "bold"))
        self.over.goto(0, self.over.ycor()-25)
```

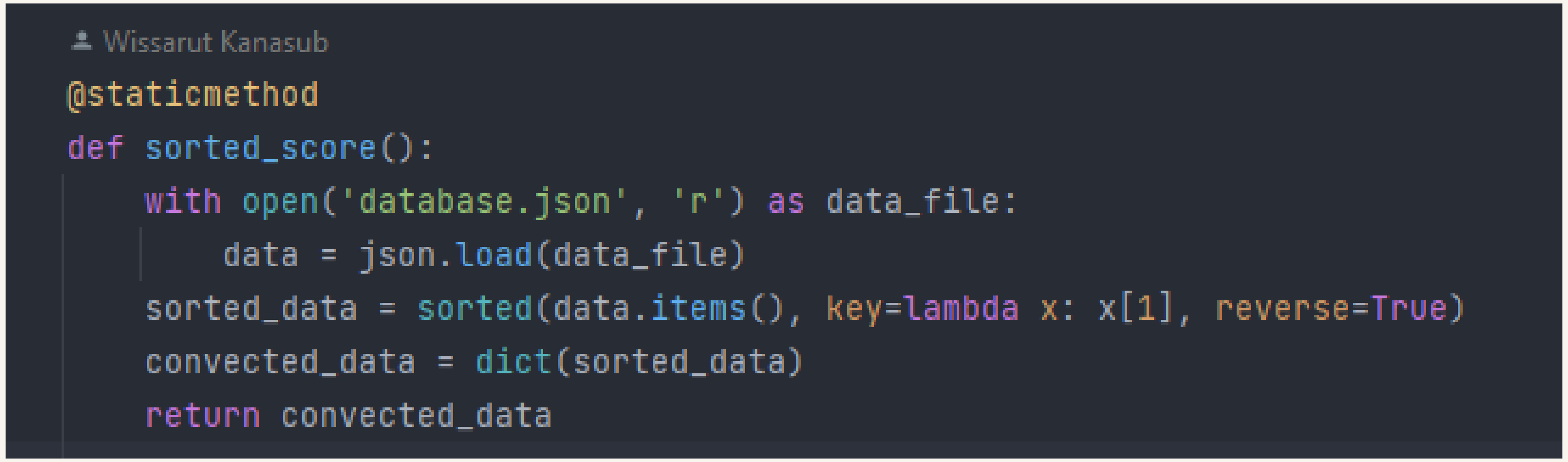
10.DatabaseGame class (DatabaseGame.py)

This class objective is save game (score and player name) in "Database.json" in format of dictionary type.

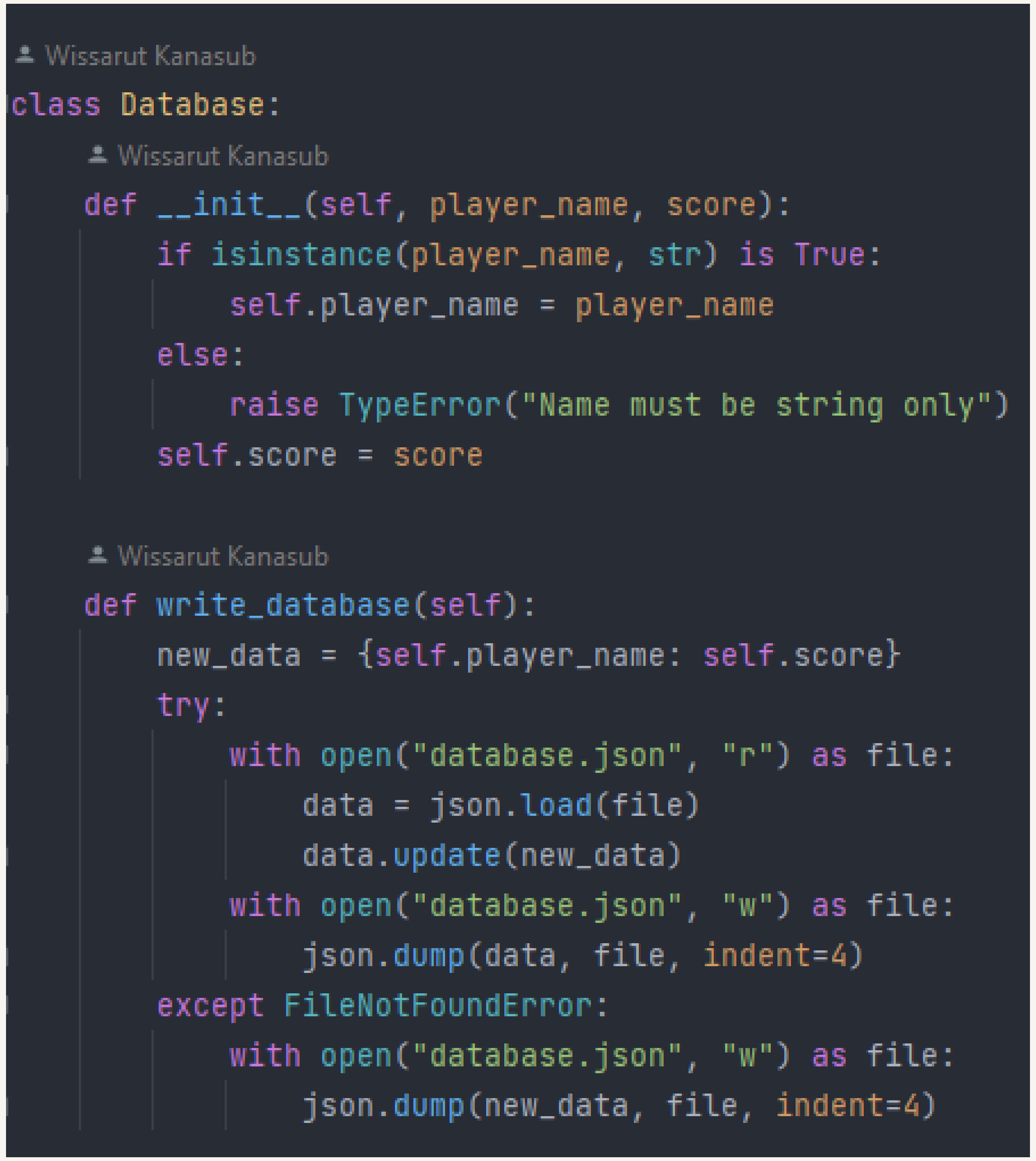
two important of this class is player_name and score

method

- **1.write_database(): Create new "Database.json" and write a save game if there is no file. if not just write only.**
- **2.sorted_score(): @staticmethod**
- **return sorted score of player high to low**



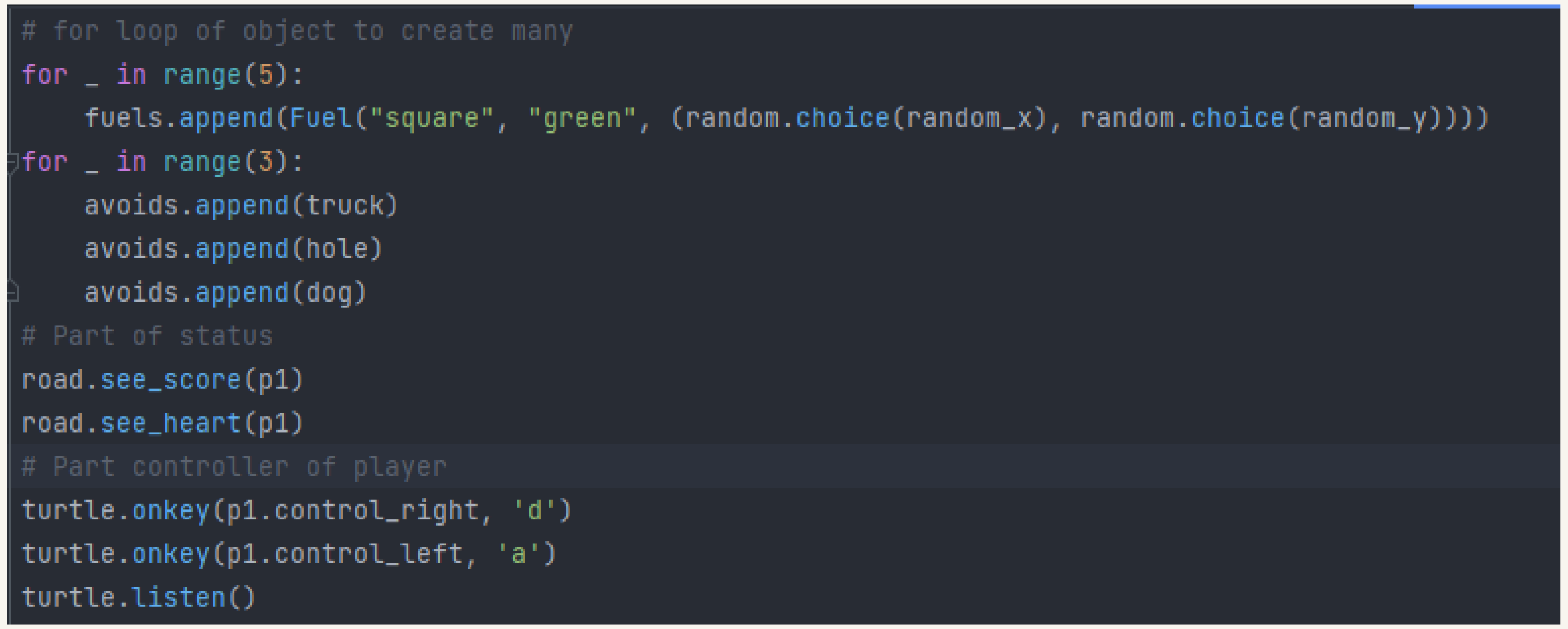
```
Wissarut Kanasub
@staticmethod
def sorted_score():
    with open('database.json', 'r') as data_file:
        data = json.load(data_file)
    sorted_data = sorted(data.items(), key=lambda x: x[1], reverse=True)
    convected_data = dict(sorted_data)
    return convected_data
```



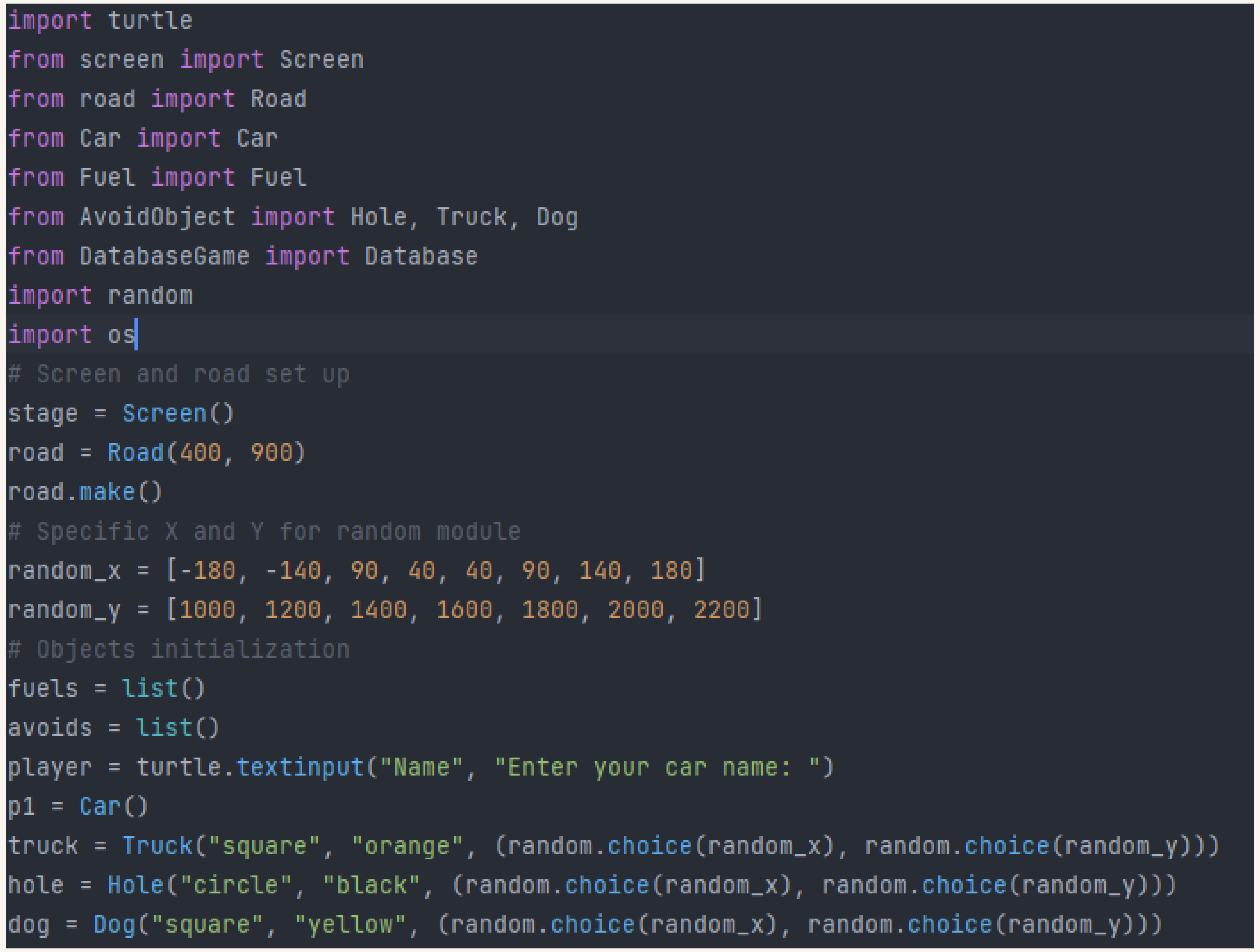
```
Wissarut Kanasub
class Database:
    Wissarut Kanasub
    def __init__(self, player_name, score):
        if isinstance(player_name, str) is True:
            self.player_name = player_name
        else:
            raise TypeError("Name must be string only")
        self.score = score

    Wissarut Kanasub
    def write_database(self):
        new_data = {self.player_name: self.score}
        try:
            with open("database.json", "r") as file:
                data = json.load(file)
                data.update(new_data)
            with open("database.json", "w") as file:
                json.dump(data, file, indent=4)
        except FileNotFoundError:
            with open("database.json", "w") as file:
                json.dump(new_data, file, indent=4)
```

11.main.py

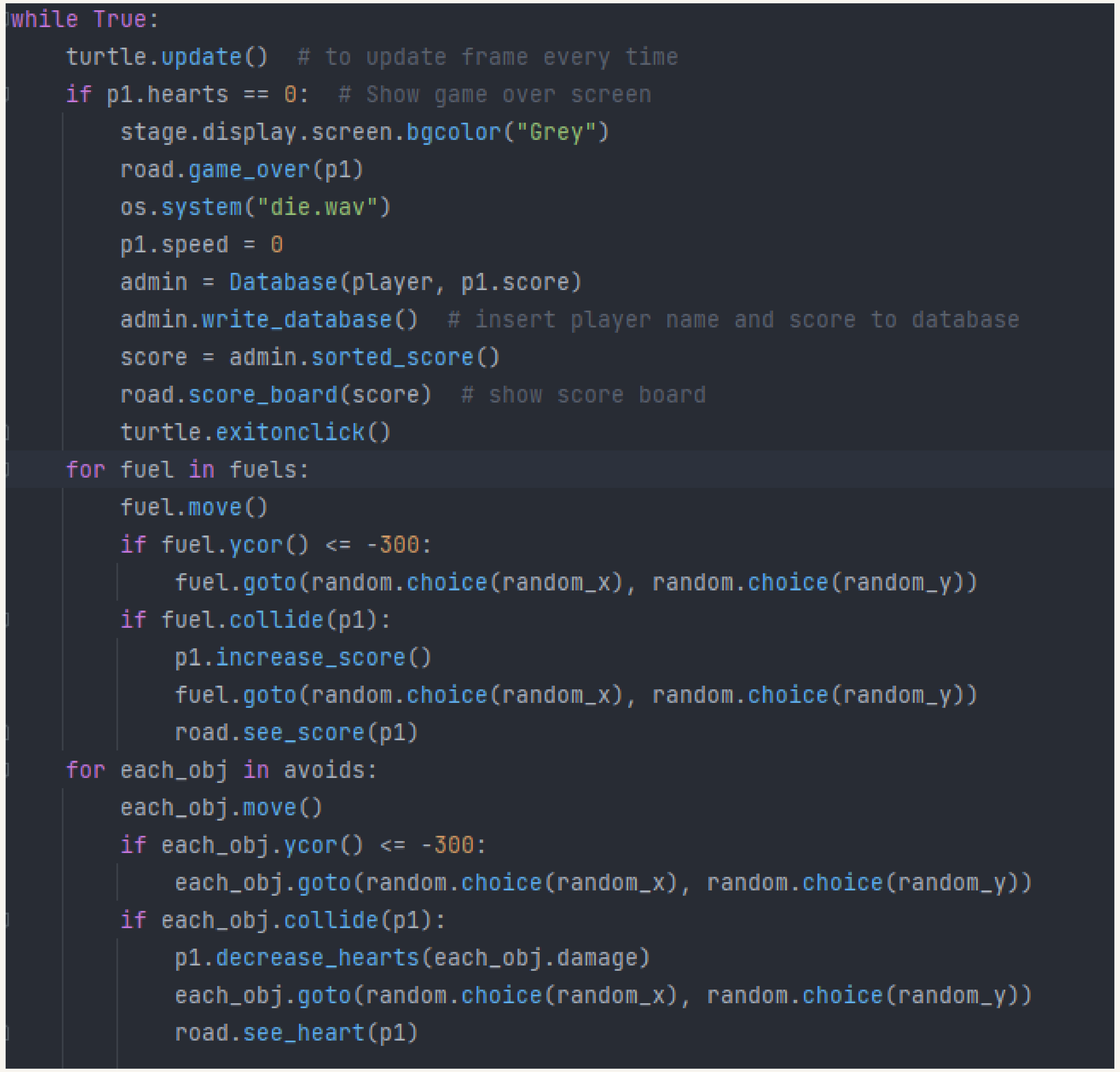
This file use all class and file to make a real game



```python
# for loop of object to create many
for _ in range(5):
    fuels.append(Fuel("square", "green", (random.choice(random_x), random.choice(random_y))))
for _ in range(3):
    avoids.append(truck)
    avoids.append(hole)
    avoids.append(dog)
# Part of status
road.see_score(p1)
road.see_heart(p1)
# Part controller of player
turtle.onkey(p1.control_right, 'd')
turtle.onkey(p1.control_left, 'a')
turtle.listen()
```



```python
import turtle
from screen import Screen
from road import Road
from Car import Car
from Fuel import Fuel
from AvoidObject import Hole, Truck, Dog
from DatabaseGame import Database
import random
import os
# Screen and road set up
stage = Screen()
road = Road(400, 900)
road.make()
# Specific X and Y for random module
random_x = [-180, -140, 90, 40, 40, 90, 140, 180]
random_y = [1000, 1200, 1400, 1600, 1800, 2000, 2200]
# Objects initialization
fuels = list()
avoids = list()
player = turtle.textinput("Name", "Enter your car name: ")
p1 = Car()
truck = Truck("square", "orange", (random.choice(random_x), random.choice(random_y)))
hole = Hole("circle", "black", (random.choice(random_x), random.choice(random_y)))
dog = Dog("square", "yellow", (random.choice(random_x), random.choice(random_y)))
```



```python
while True:
    turtle.update()  # to update frame every time
    if p1.hearts == 0:  # Show game over screen
        stage.display.screen.bgcolor("Grey")
        road.game_over(p1)
        os.system("die.wav")
        p1.speed = 0
        admin = Database(player, p1.score)
        admin.write_database()  # insert player name and score to database
        score = admin.sorted_score()
        road.score_board(score)  # show score board
        turtle.exitonclick()
    for fuel in fuels:
        fuel.move()
        if fuel.ycor() <= -300:
            fuel.goto(random.choice(random_x), random.choice(random_y))
        if fuel.collide(p1):
            p1.increase_score()
            fuel.goto(random.choice(random_x), random.choice(random_y))
            road.see_score(p1)
    for each_obj in avoids:
        each_obj.move()
        if each_obj.ycor() <= -300:
            each_obj.goto(random.choice(random_x), random.choice(random_y))
        if each_obj.collide(p1):
            p1.decrease_hearts(each_obj.damage)
            each_obj.goto(random.choice(random_x), random.choice(random_y))
            road.see_heart(p1)
```