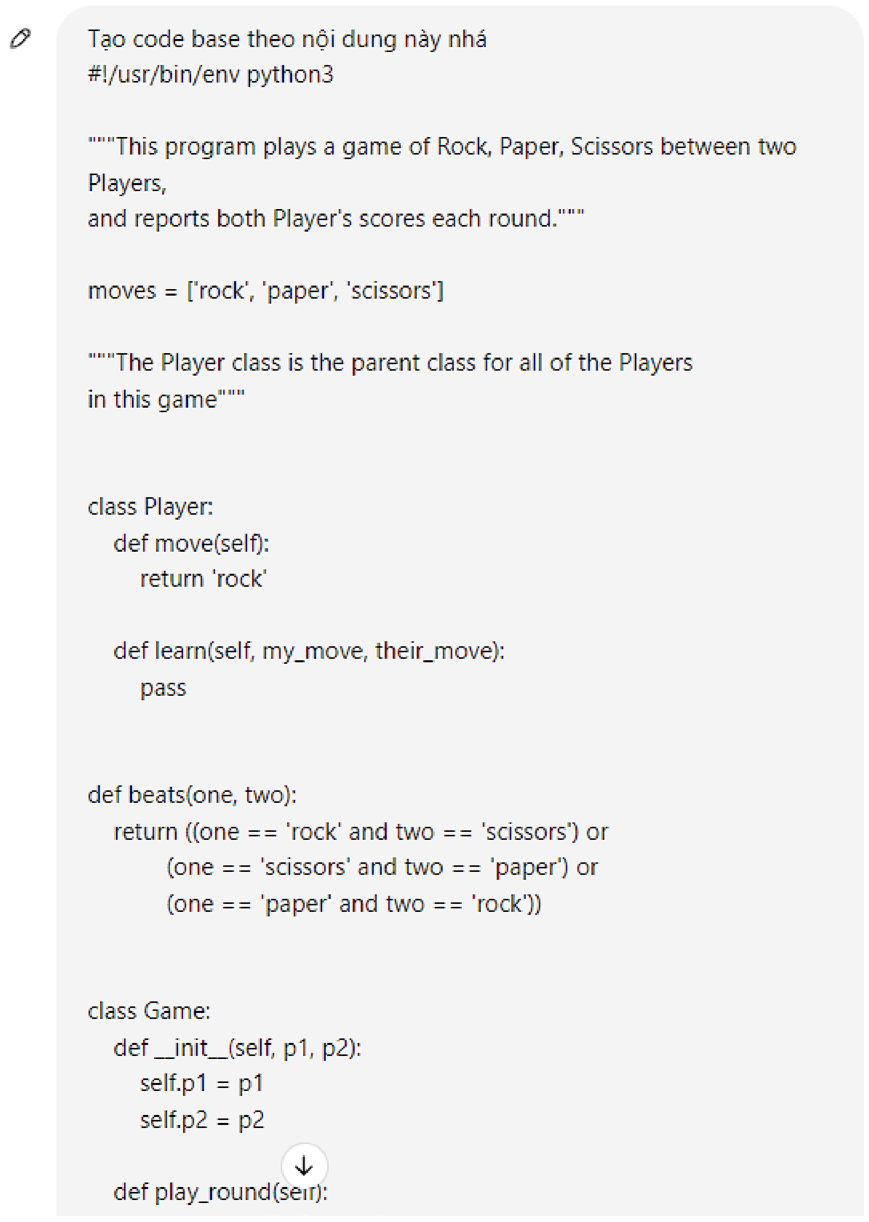
I did use AI to assist me, but it doesn't seem like I took other people's work. I'm sure I worked independently.
This is my first post.

```
import random


# Constants for the game

ROCK = 'rock'

PAPER = 'paper'

SCISSORS = 'scissors'

MOVES = [ROCK, PAPER, SCISSORS]


# Base class for a player

class Player:

    def move(self):

        pass


    def learn(self, opponent_move):

        pass


# Human player

class Human(Player):

    def move(self):

        move = input("Enter your move (rock, paper, scissors): ").lower()

        while move not in MOVES:

            move = input("Invalid move. Please try again (rock, paper, scissors): ").lower()

        return move


# Computer player that always plays 'rock'

class RockPlayer(Player):
    def move(self):
        return ROCK


# Computer player that chooses a move randomly
class RandomPlayer(Player):
    def move(self):
        return random.choice(MOVES)


# Computer player that mimics the opponent's last move
class ReflectPlayer(Player):
    def __init__(self):
        self.opponent_last_move = None

    def move(self):
        if self.opponent_last_move:
            return self.opponent_last_move
        else:
            return random.choice(MOVES)

    def learn(self, opponent_move):
        self.opponent_last_move = opponent_move


# Computer player that cycles through the three moves
class CyclePlayer(Player):
    def __init__(self):
        self.index = 0

    def move(self):
        move = MOVES[self.index]
        self.index = (self.index + 1) % len(MOVES)
        return move

# Main class for the game
class Game:
    def __init__(self, player1, player2):
        self.player1 = player1
        self.player2 = player2
        self.score1 = 0
        self.score2 = 0

    def play_round(self):
        move1 = self.player1.move()
        move2 = self.player2.move()
        print(f"Player 1 chooses: {move1} - Player 2 chooses: {move2}")

        winner = self.determine_winner(move1, move2)
        if winner == 1:
            self.score1 += 1
            print("Player 1 wins this round!")
        elif winner == 2:
            self.score2 += 1
            print("Player 2 wins this round!")
        else:
            print("It's a tie!")

        print(f"Scores: Player 1 - {self.score1}, Player 2 - {self.score2}")

        self.player1.learn(move2)
        self.player2.learn(move1)

    def determine_winner(self, move1, move2):
        if move1 == move2:
            return 0
        elif (move1 == ROCK and move2 == SCISSORS) or \
             (move1 == PAPER and move2 == ROCK) or \
             (move1 == SCISSORS and move2 == PAPER):
            return 1
        else:
            return 2

    def play_game(self, rounds):
        print("Starting the Rock-Paper-Scissors game!")
        for round in range(1, rounds + 1):
            print(f"\nRound {round}:")
            self.play_round()
        print("\nGame over!")
        print(f"Final scores: Player 1 - {self.score1}, Player 2 - {self.score2}")

# Run the program
if __name__ == '__main__':
    human = Human()
    computer = random.choice([RockPlayer(), RandomPlayer(), ReflectPlayer(), CyclePlayer()])
    game = Game(human, computer)
    game.play_game(3)
```

Then I realized that the assignment was wrong compared to the architecture of the rps.py file in workspace, so I asked GPT to help me edit it.

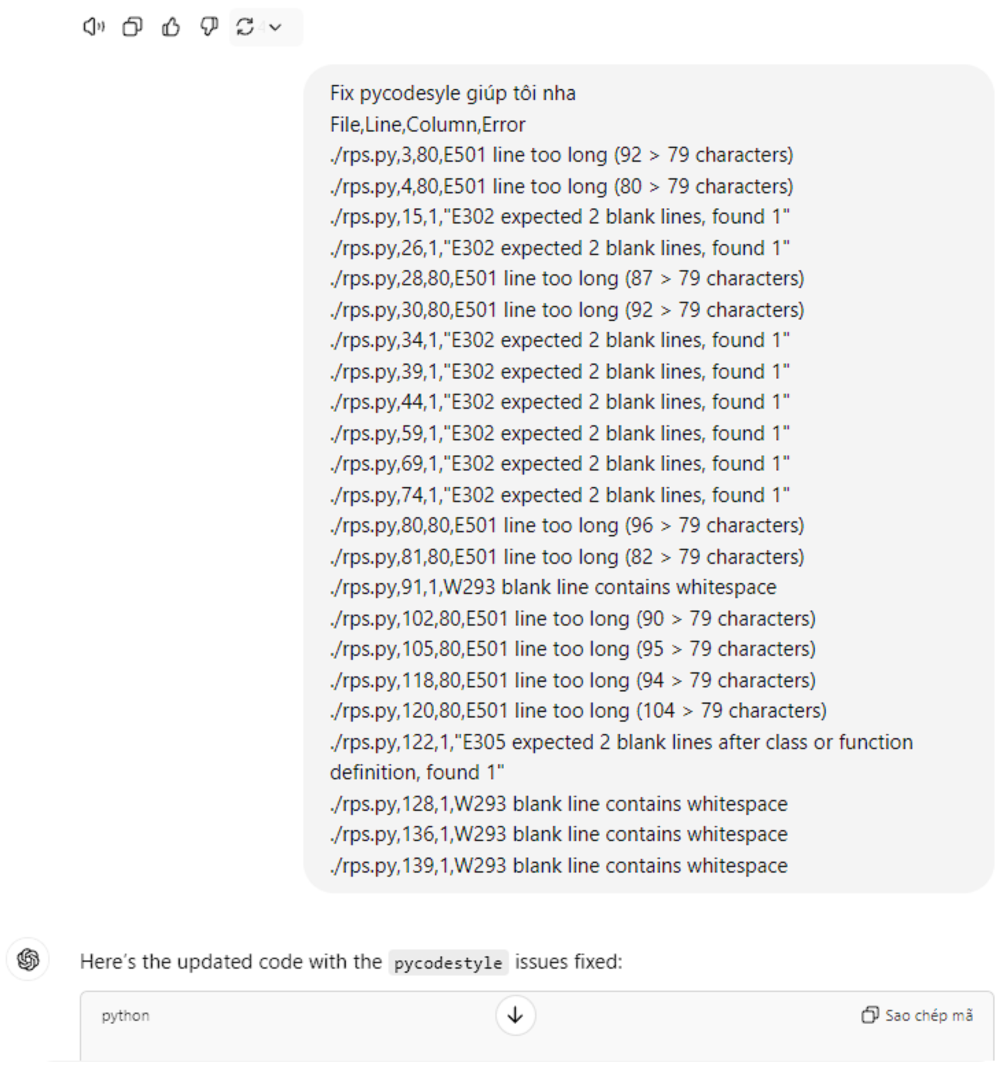
This my prompt



```
Tạo code base theo nội dung này nhá
#!/usr/bin/env python3

"""This program plays a game of Rock, Paper, Scissors between two
Players,
and reports both Player's scores each round."""

moves = ['rock', 'paper', 'scissors']

"""The Player class is the parent class for all of the Players
in this game"""


class Player:
    def move(self):
        return 'rock'

    def learn(self, my_move, their_move):
        pass


def beats(one, two):
    return ((one == 'rock' and two == 'scissors') or
            (one == 'scissors' and two == 'paper') or
            (one == 'paper' and two == 'rock'))


class Game:
    def __init__(self, p1, p2):
        self.p1 = p1
        self.p2 = p2

    def play_round(self):
```

Later in the previous lesson I added the function to record scores so I added them to my work.
Finally I used the file: pycodestyles_export_csv.py to check for pycodestyle errors

It will output pycodestyle errors to csv result.csv
I asked GPT to fix the pycodestyle error for me



Fix pycodesyle giúp tôi nha

```
File,Line,Column,Error
./rps.py,3,80,E501 line too long (92 > 79 characters)
./rps.py,4,80,E501 line too long (80 > 79 characters)
./rps.py,15,1,"E302 expected 2 blank lines, found 1"
./rps.py,26,1,"E302 expected 2 blank lines, found 1"
./rps.py,28,80,E501 line too long (87 > 79 characters)
./rps.py,30,80,E501 line too long (92 > 79 characters)
./rps.py,34,1,"E302 expected 2 blank lines, found 1"
./rps.py,39,1,"E302 expected 2 blank lines, found 1"
./rps.py,44,1,"E302 expected 2 blank lines, found 1"
./rps.py,59,1,"E302 expected 2 blank lines, found 1"
./rps.py,69,1,"E302 expected 2 blank lines, found 1"
./rps.py,74,1,"E302 expected 2 blank lines, found 1"
./rps.py,80,80,E501 line too long (96 > 79 characters)
./rps.py,81,80,E501 line too long (82 > 79 characters)
./rps.py,91,1,W293 blank line contains whitespace
./rps.py,102,80,E501 line too long (90 > 79 characters)
./rps.py,105,80,E501 line too long (95 > 79 characters)
./rps.py,118,80,E501 line too long (94 > 79 characters)
./rps.py,120,80,E501 line too long (104 > 79 characters)
./rps.py,122,1,"E305 expected 2 blank lines after class or function definition, found 1"
./rps.py,128,1,W293 blank line contains whitespace
./rps.py,136,1,W293 blank line contains whitespace
./rps.py,139,1,W293 blank line contains whitespace
```

Here's the updated code with the `pycodestyle` issues fixed:

Then I checked again and saw if GPT fixed pycodestyle by running the pycodestyles_export_csv.py file again.

Then I checked that the code was correct according to the requirements on the rubric and I submitted the codes.