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

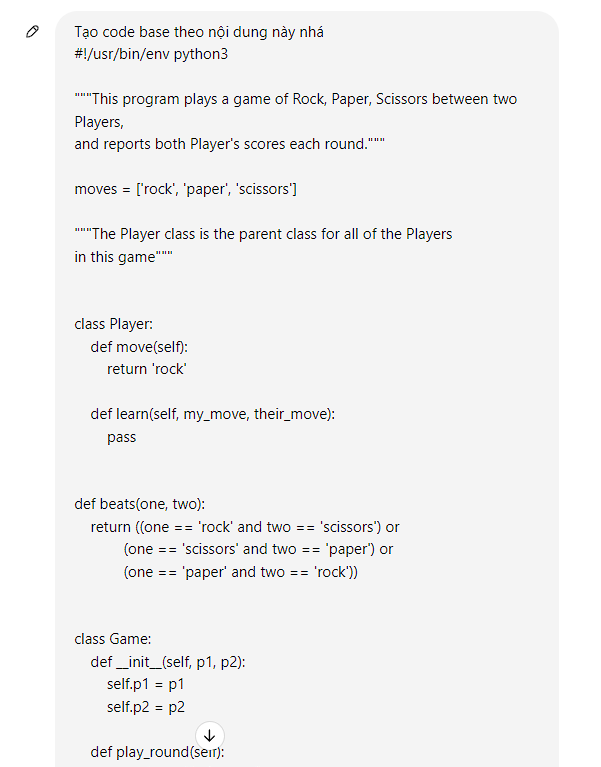
if \_\_name\_\_ == '\_\_main\_\_':

human = Human()

computer = random.choice([RockPlayer(), RandomPlayer(), ReflectPlayer(), CyclePlayer()])

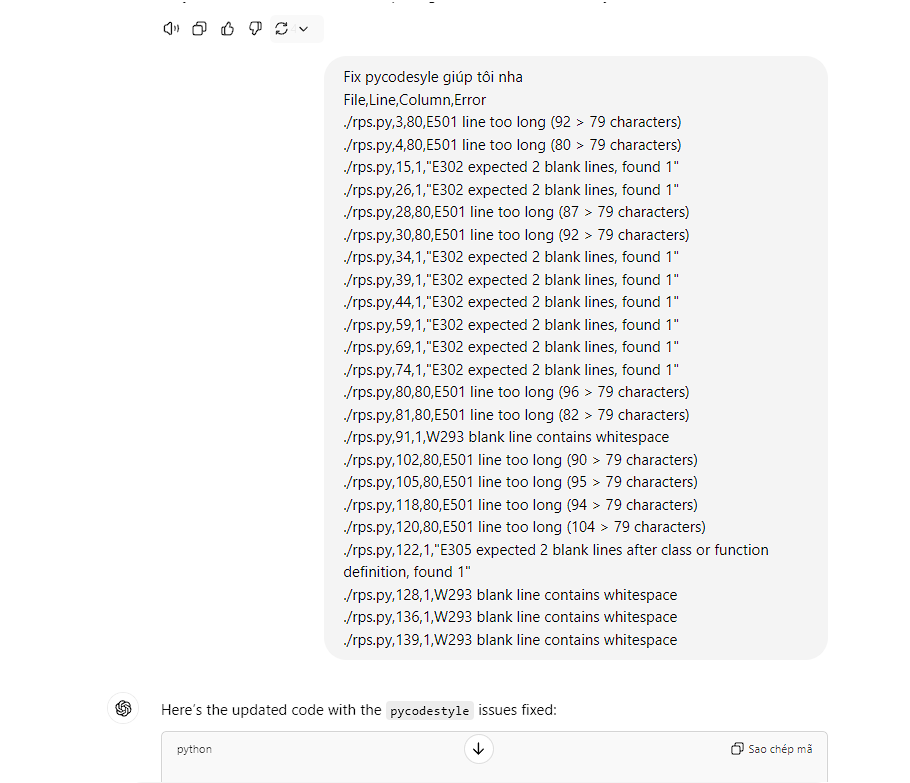
game = Game(human, computer)

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  


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



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.