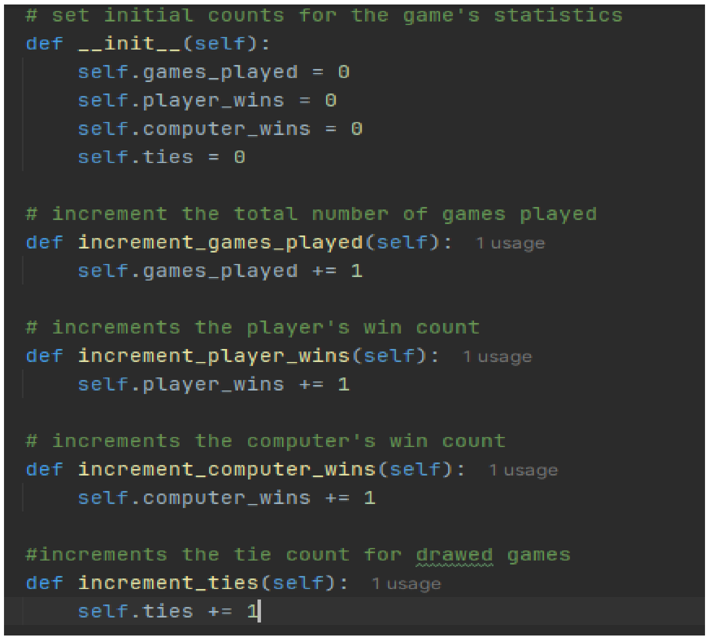
Audy Vee

Good Principles

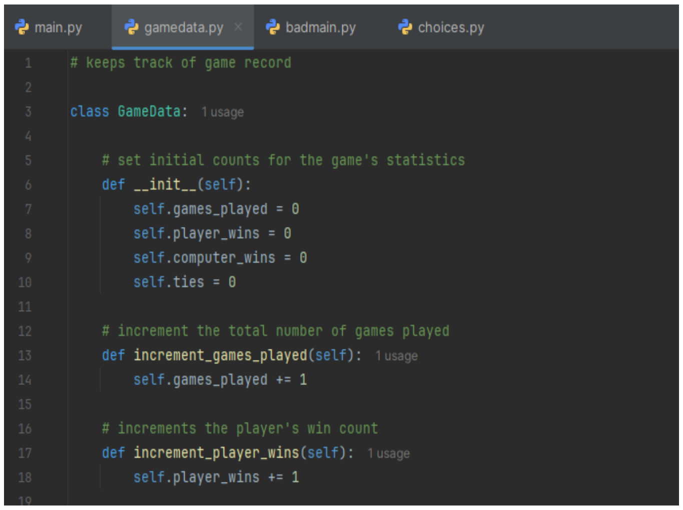
KISS



```python
# set initial counts for the game's statistics
def __init__(self):
    self.games_played = 0
    self.player_wins = 0
    self.computer_wins = 0
    self.ties = 0

# increment the total number of games played
def increment_games_played(self):  1 usage
    self.games_played += 1

# increments the player's win count
def increment_player_wins(self):  1 usage
    self.player_wins += 1

# increments the computer's win count
def increment_computer_wins(self):  1 usage
    self.computer_wins += 1

#increments the tie count for drawed games
def increment_ties(self):  1 usage
    self.ties += 1
```

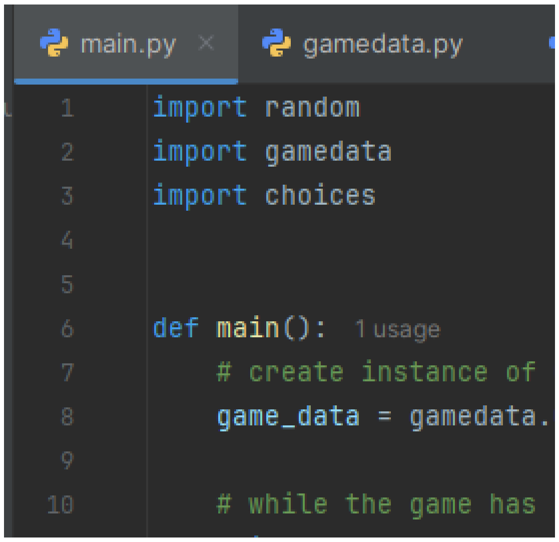
KISS is being used as most of the methods in the GameData class are simple and straightforward to understand. Can understand how each method is being used for instance, the increment methods.

Single Responsibility



main.py gamedata.py badmain.py choices.py

```
# keeps track of game record

class GameData:  1 usage

    # set initial counts for the game's statistics
    def __init__(self):
        self.games_played = 0
        self.player_wins = 0
        self.computer_wins = 0
        self.ties = 0

    # increment the total number of games played
    def increment_games_played(self):  1 usage
        self.games_played += 1

    # increments the player's win count
    def increment_player_wins(self):  1 usage
        self.player_wins += 1
```

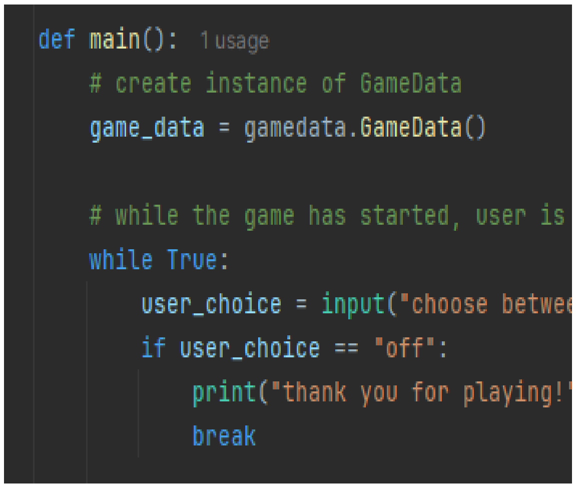


main.py gamedata.py

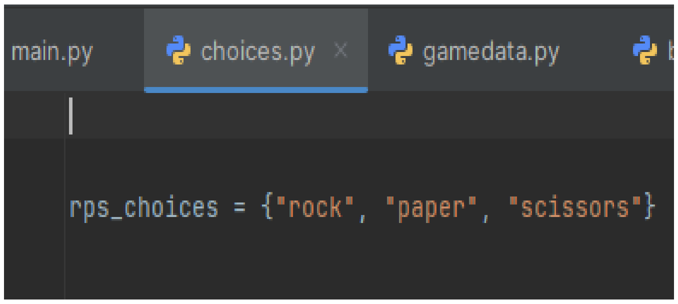
```
import random
import gamedata
import choices


def main():  1 usage
    # create instance of
    game_data = gamedata.

    # while the game has
```

The gamedata.py handles the game's statistics and keeps track of it. The main.py handles the game logic and the user's inputs. This ties in with Single Responsibility, as separating each role can help with code management.

Separation of Concerns

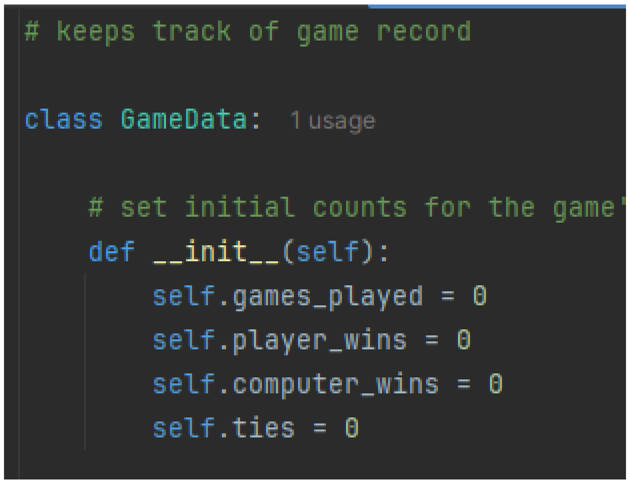


```
def main():  1 usage
    # create instance of GameData
    game_data = gamedata.GameData()

    # while the game has started, user is
    while True:
        user_choice = input("choose betwee
        if user_choice == "off":
            print("thank you for playing!"
            break
```



main.py   choices.py ×   gamedata.py

```
rps_choices = {"rock", "paper", "scissors"}
```

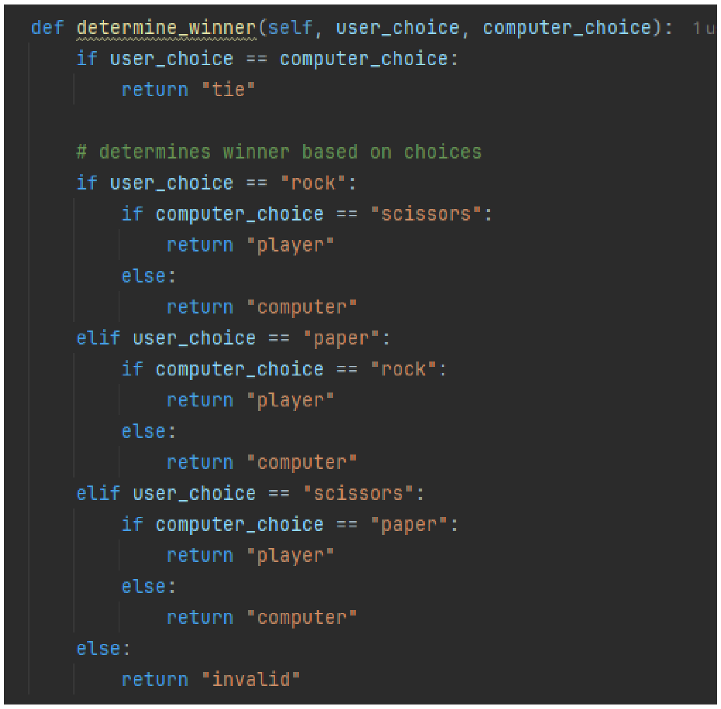


```
# keeps track of game record

class GameData:  1 usage

    # set initial counts for the game'
    def __init__(self):
        self.games_played = 0
        self.player_wins = 0
        self.computer_wins = 0
        self.ties = 0
```

Rather having everything in the main function, choices.py and gamedata.py were created for their single responsibility. By using separation of concerns, this helps with maintaining and understanding the code.

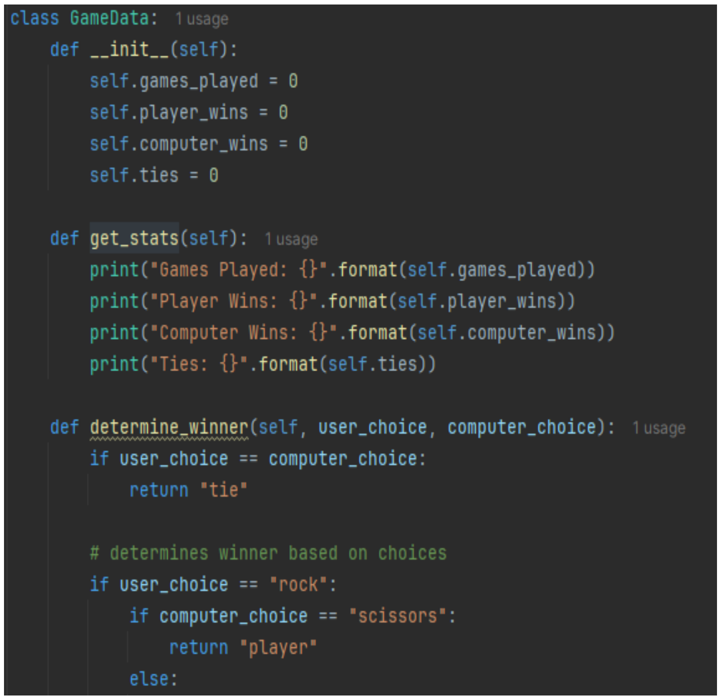
Bad Principles

KISS

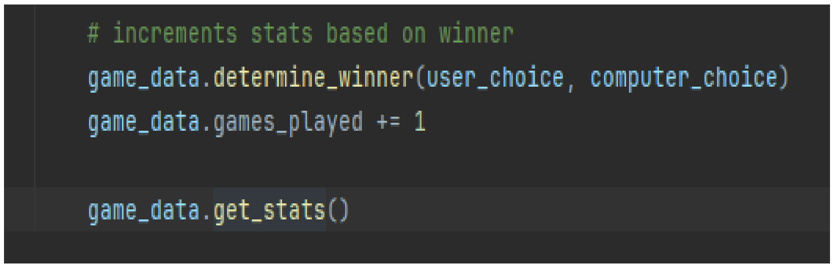

```python
def determine_winner(self, user_choice, computer_choice):
    if user_choice == computer_choice:
        return "tie"

    # determines winner based on choices
    if user_choice == "rock":
        if computer_choice == "scissors":
            return "player"
        else:
            return "computer"
    elif user_choice == "paper":
        if computer_choice == "rock":
            return "player"
        else:
            return "computer"
    elif user_choice == "scissors":
        if computer_choice == "paper":
            return "player"
        else:
            return "computer"
    else:
        return "invalid"
```

Although the code is simple to understand, it is however not efficient as it violates the KISS principles. The long chain of if statements can make the code harder to read.

Single Responsibility



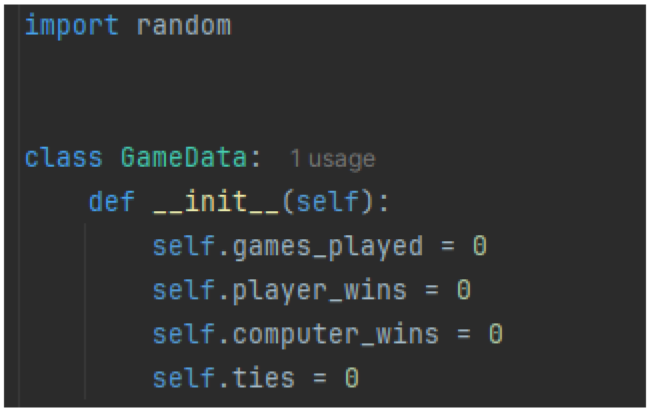
```
class GameData:  1 usage
    def __init__(self):
        self.games_played = 0
        self.player_wins = 0
        self.computer_wins = 0
        self.ties = 0

    def get_stats(self):  1 usage
        print("Games Played: {}".format(self.games_played))
        print("Player Wins: {}".format(self.player_wins))
        print("Computer Wins: {}".format(self.computer_wins))
        print("Ties: {}".format(self.ties))

    def determine_winner(self, user_choice, computer_choice):  1 usage
        if user_choice == computer_choice:
            return "tie"

        # determines winner based on choices
        if user_choice == "rock":
            if computer_choice == "scissors":
                return "player"
            else:
```



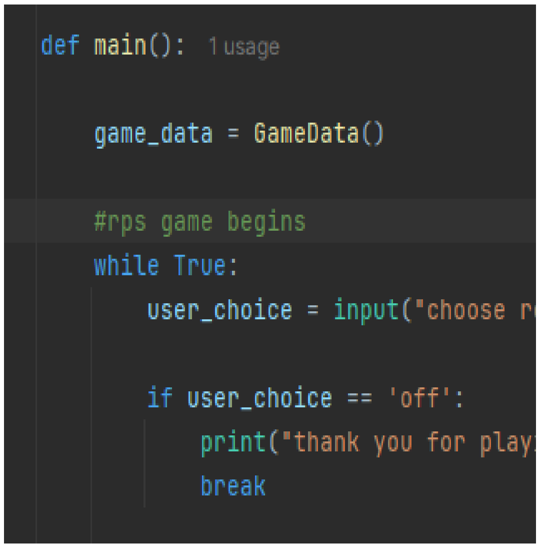
```
    # increments stats based on winner
    game_data.determine_winner(user_choice, computer_choice)
    game_data.games_played += 1

    game_data.get_stats()
```

The GameData class is handling both the game logic and statistics. It is best to keep one role for each class to avoid violation of the KISS and single responsibility principle.

Separation of Concerns



```
import random


class GameData:  1 usage
    def __init__(self):
        self.games_played = 0
        self.player_wins = 0
        self.computer_wins = 0
        self.ties = 0
```



```
def main():  1 usage

    game_data = GameData()

    #rps game begins
    while True:
        user_choice = input("choose r

        if user_choice == 'off':
            print("thank you for play
            break
```

GameData class and the main game logic are defined in the same file. This leads to the code becoming less readable and harder to manage, this violates separation of concerns.