1. **Bài tập lớn 1:**

A screen shot of a computer

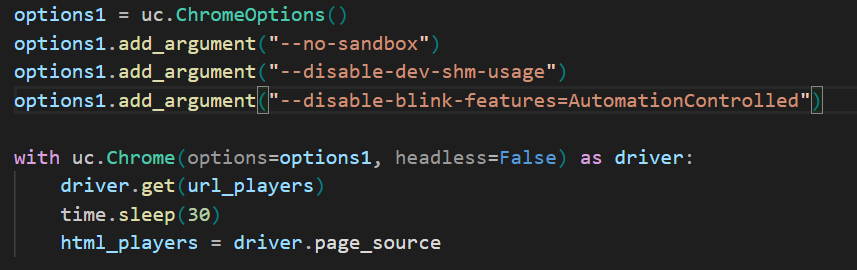
AI-generated content may be incorrect.

* Khai báo các thư viện:
* undetected\_chromedriver: chạy Chrome không bị phát hiện bởi các trang web chống bot.
* BeautifulSoup: phân tích và trích xuất dữ liệu từ mã HTML.
* time: để tạm dừng chương trình (delay khi chờ trang web tải).
* pandas, numpy: xử lý và phân tích dữ liệu bảng.

1. Lấy dữ liệu từ bảng standard:



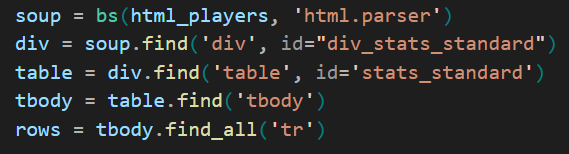
* Truy cập vào URL standard stats



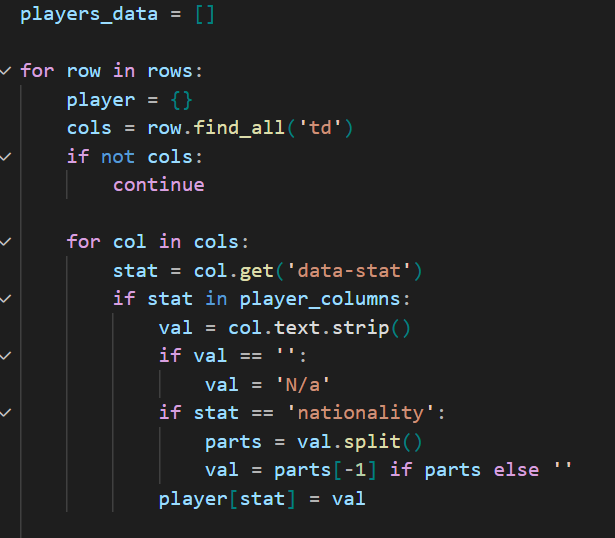
* Mở trình duyệt, chờ 30 giây cho trang tải xong, lấy mã HTML.



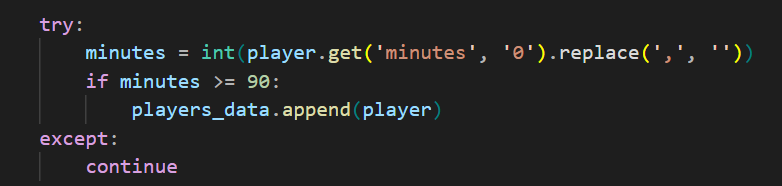
* Định nghĩa **các cột cần lấy** từ bảng HTML -> tên cột tương ứng trong DataFrame.



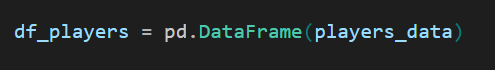
* Dùng BeautifulSoup để phân tích HTML -> tìm bảng stats\_standard



* Với mỗi hang, lấy theo từng cột data-stat



* Chỉ lưu các cầu thủ có trên 90p thi đấu.

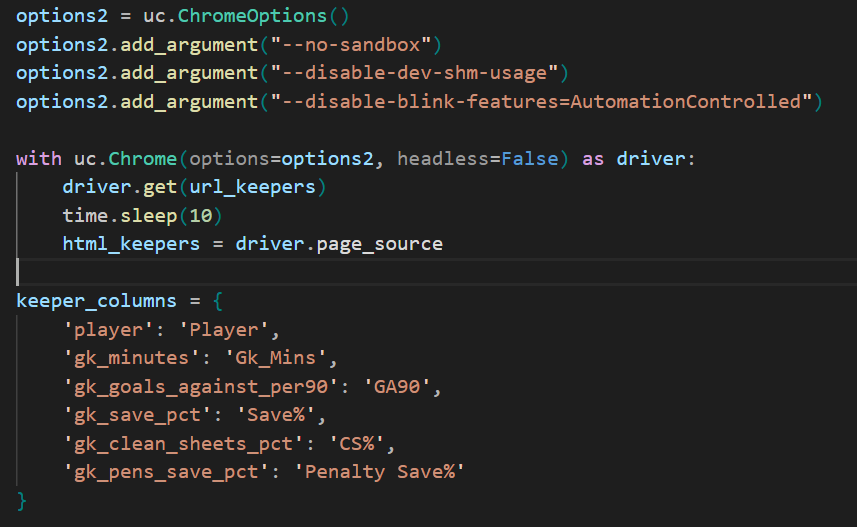


* Tạo Data Frame để chứa các chỉ số

1. Lấy dữ liệu của thủ môn:



* URL của các chỉ số thủ môn



* Mở trình duyệt, chờ 10, lấy mã HTML và khai báo các chỉ số của thủ môn



* Dùng BeautifulSoup để phân tích HTML -> tìm bảng stats\_keeper
* Lấy giữ liệu mỗi dòng, chỉ giữ lại các thủ môn có thời gian chơi trên 90p

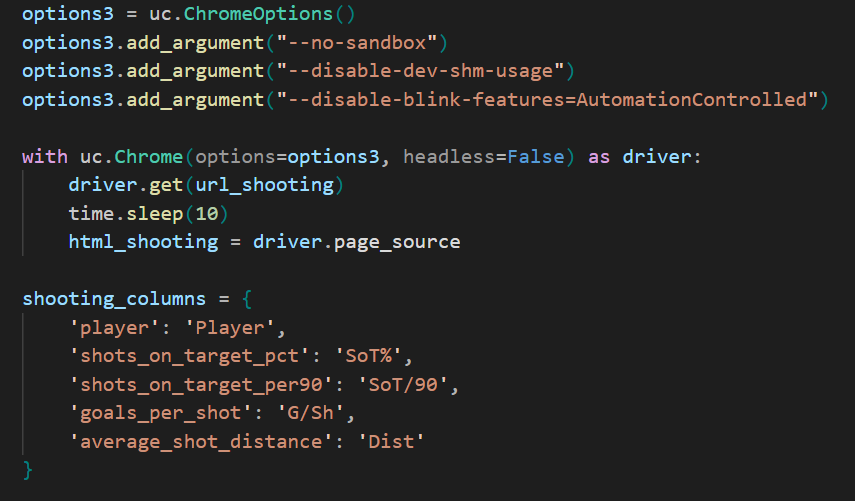


* Tạo Data Frame từ danh sách keepers\_data

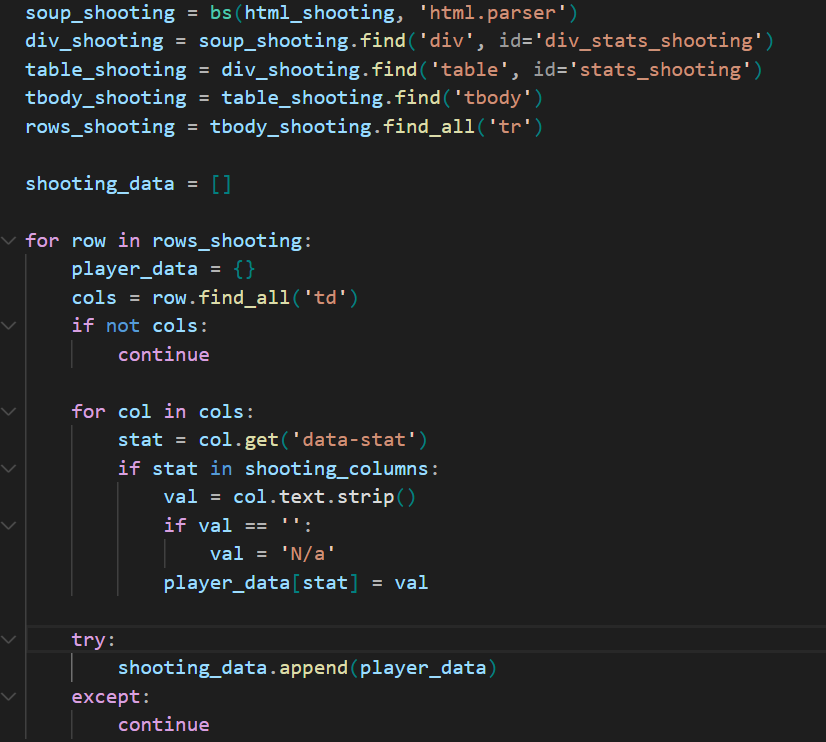
1. Lấy dữ liệu Shooting:



* URL của shooting



* Mở trình duyệt, chờ 10 giây, lấy mã HTML
* Khai báo các chỉ số và tên của cột tương ứng trong DataFrame



* Dùng BeautifulSoup để phân tích HTML -> tìm bảng stat\_shooting
* Lấy các cầu thủ có thời gian chơi trên 90p

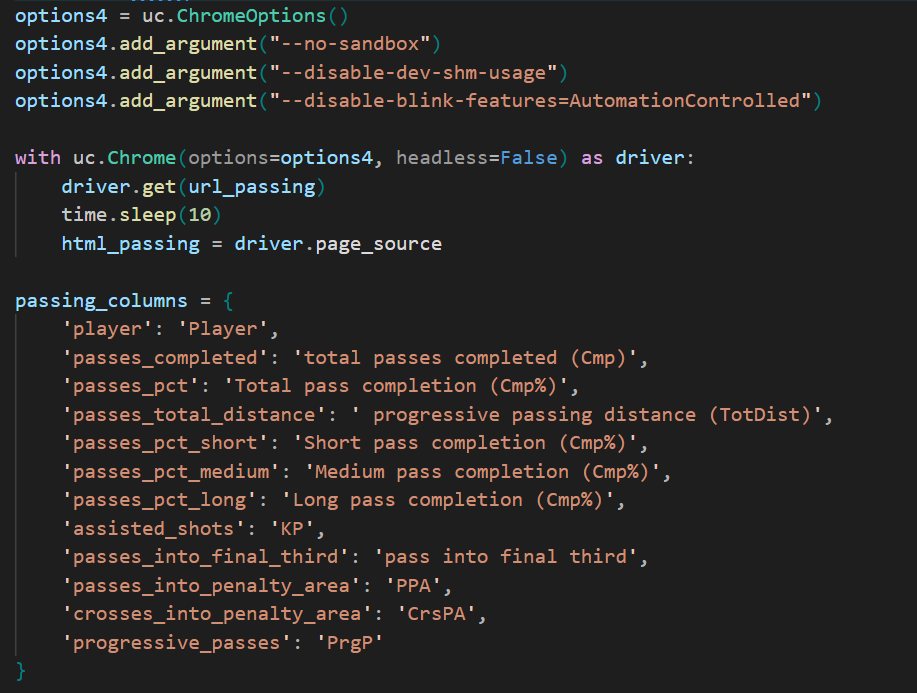


* Tạo Data Frame từ danh sách shooting\_data

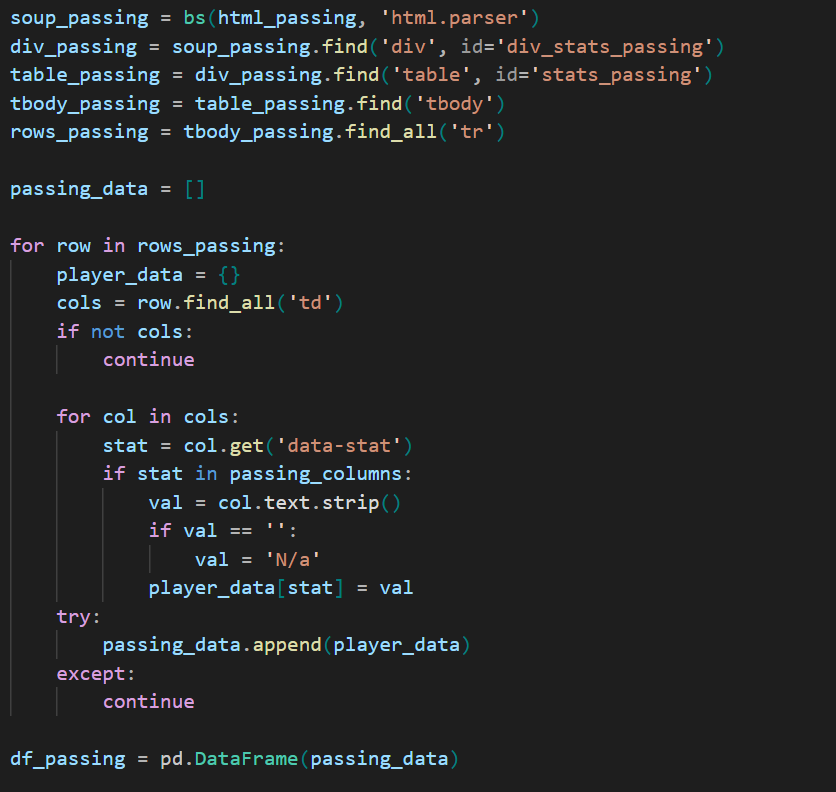
1. Lấy dữ liệu Passing:



* Truy cập vào trang stat passing

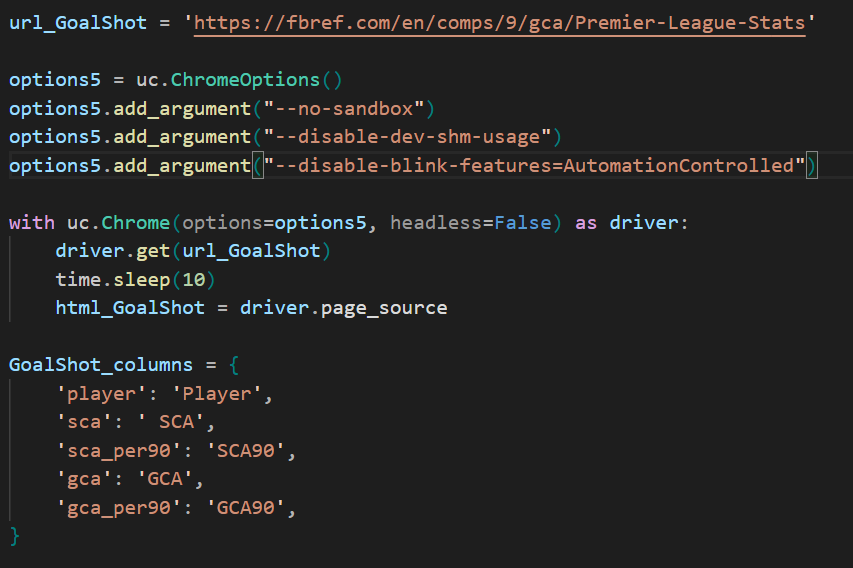


* Mở trình duyệt, chờ 10s để tải trang, lấy HTML
* Khai báo các chỉ số và tên của cột tương ứng trong DataFrame

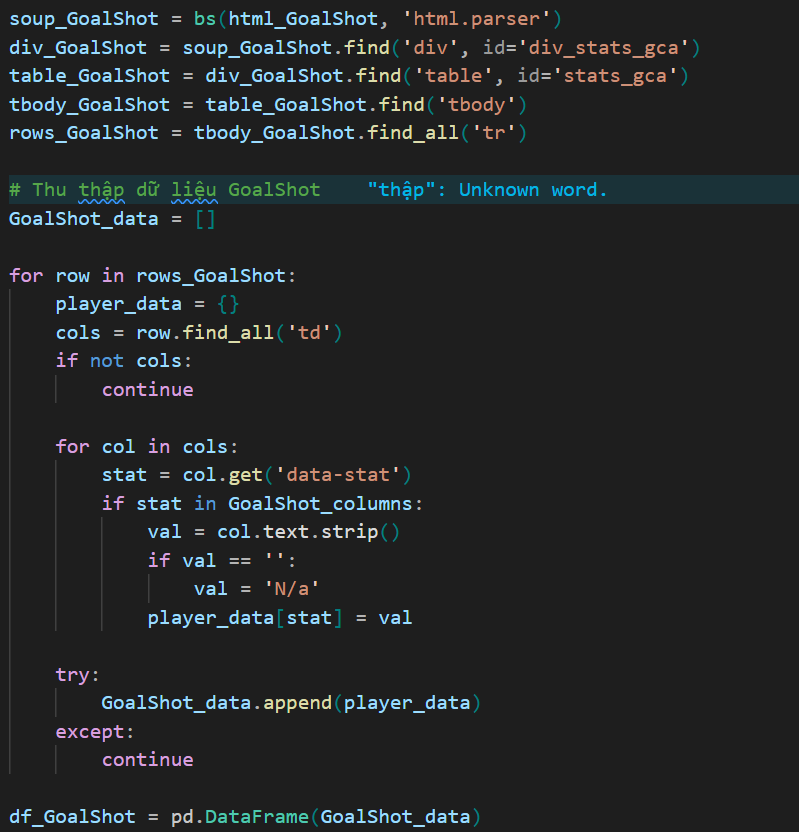


* Dùng BeautifulSoup phân tích HTML -> tìm bảng stat\_passing
* Lấy dữ liệu từ bảng với các chỉ số đã khai báo
* Tạo Data Frame từ danh sách passing\_data

1. Lấy dữ liệu Goal Shot Creation:

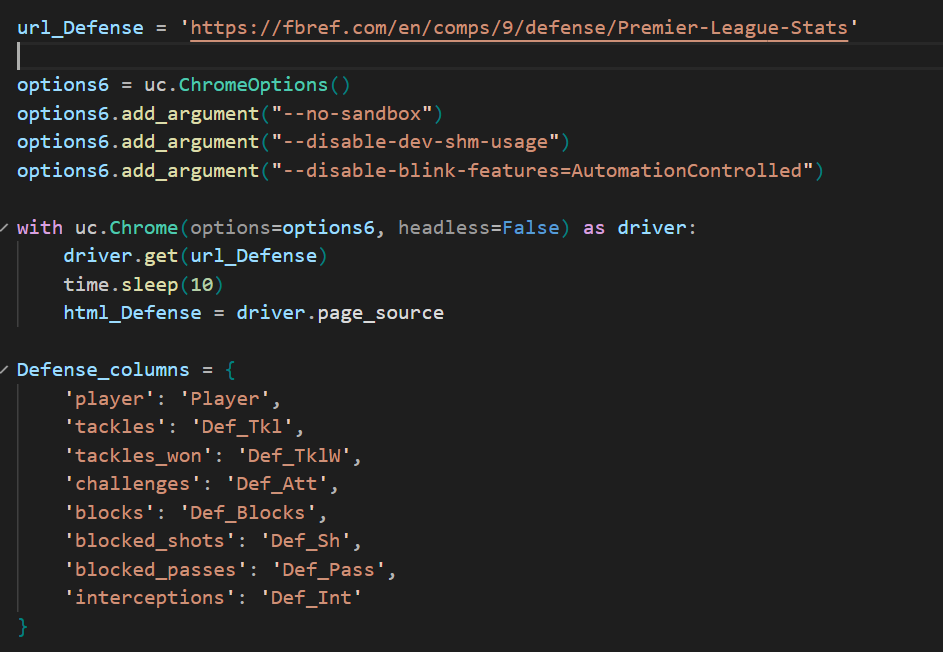


* Truy cập vào trang Goal and Shot Creation
* Mở trình duyệt, chờ 10s để tải trang, lấy mã HTML
* Khai báo các chỉ số và tên của cột tương ứng trong DataFrame

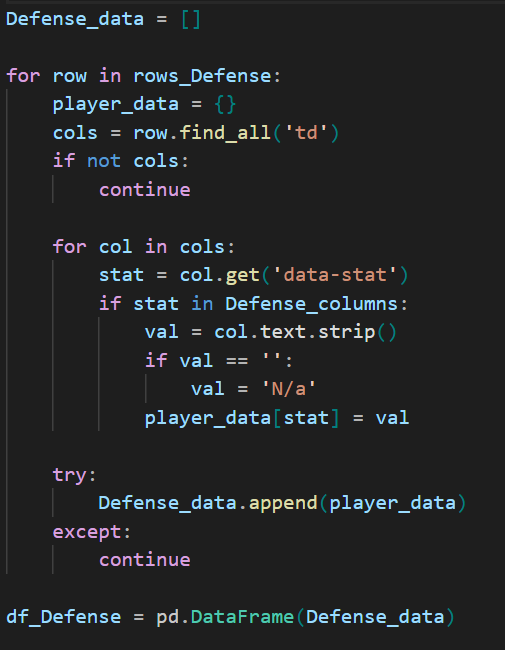


* Dùng BeautifulSoup để phân tích HTML -> tìm bảng stat\_gca
* Lấy dữ liệu từ mỗi dòng bảng và xử lý
* Tạo DataFrame từ danh sách GoalShot\_data

1. Lấy dữ liệu Defense Stat:

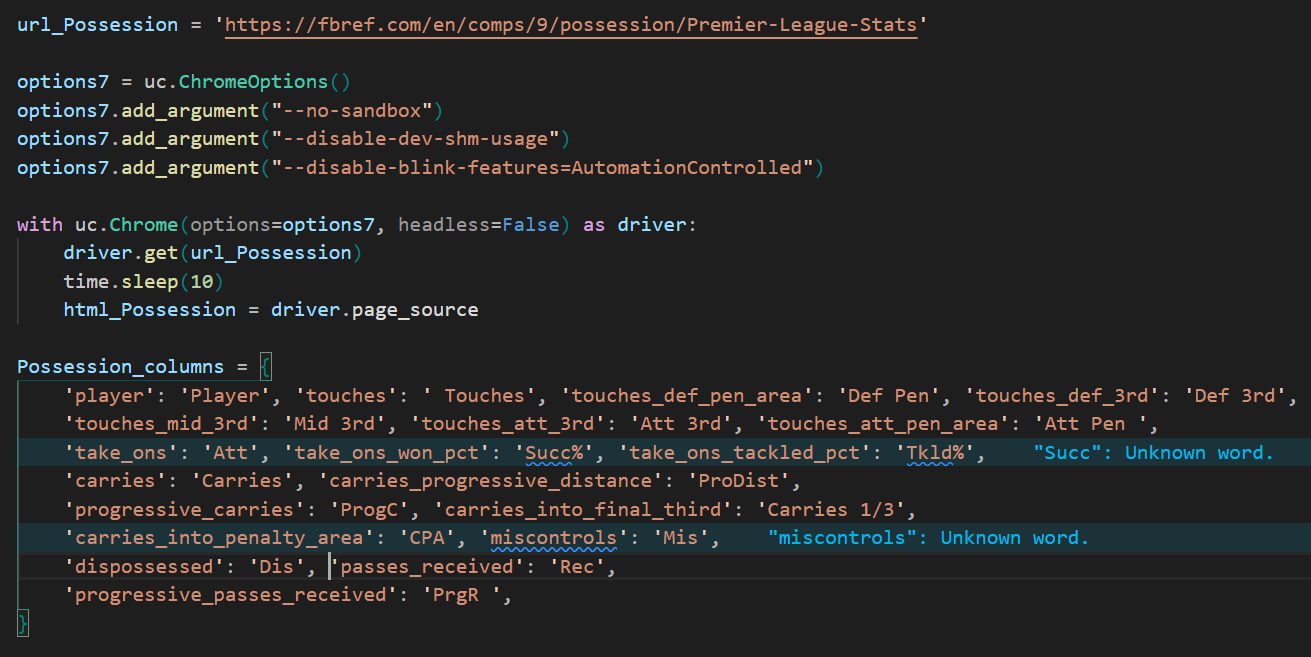


* Truy cập vào trang Defense Stat
* Mở trình duyệt, chờ 10s để tải trang, lấy mã HTML
* Khai báo các chỉ số và tên cột tương ứng trong DataFrame

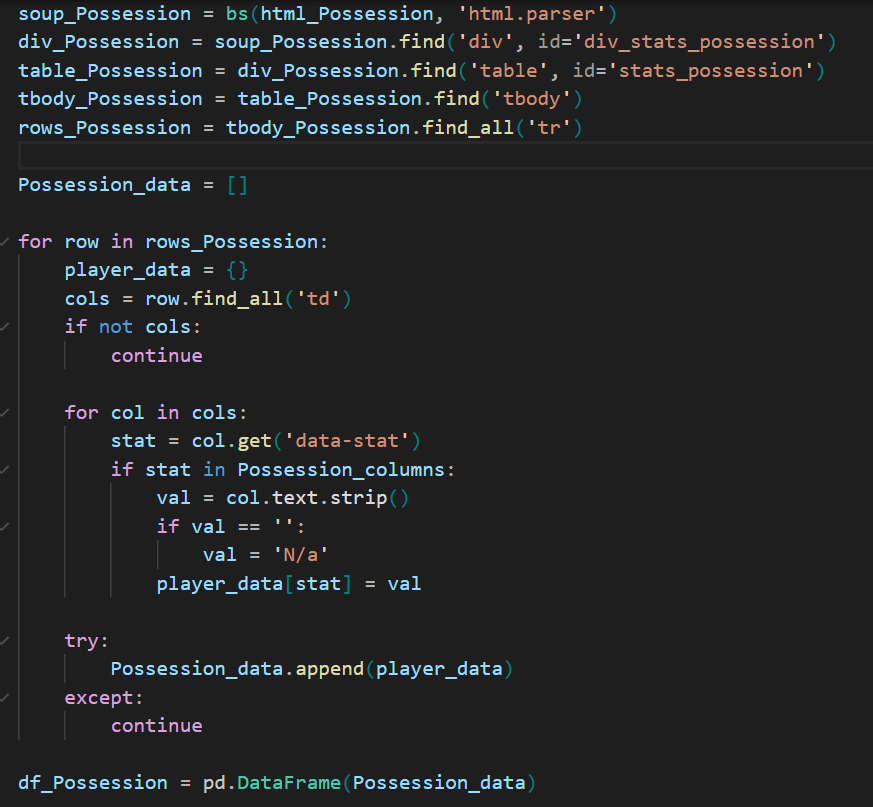


* Lấy dữ liệu từ mỗi dòng của bảng và xử lý
* Tạo DataFrame từ danh sách Defense\_data

1. Lấy dữ liệu Possession Stat

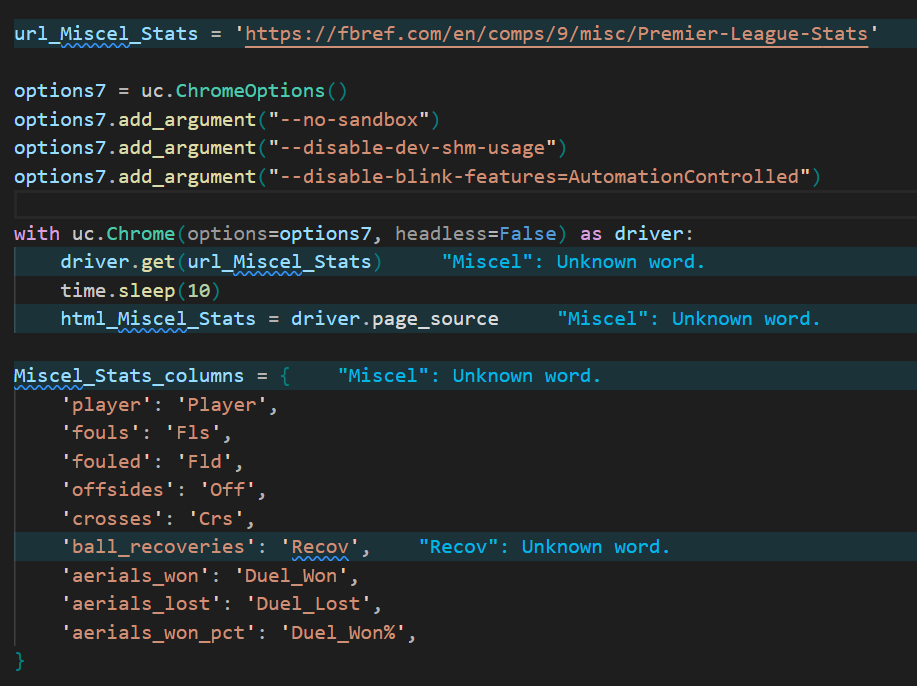


* Truy cập vào trang Possession Stat
* Mở trình duyệt, chờ 10s để tải trang, lấy mã HTML
* Khai báo các chỉ số và tên cột tương ứng trong DataFrame

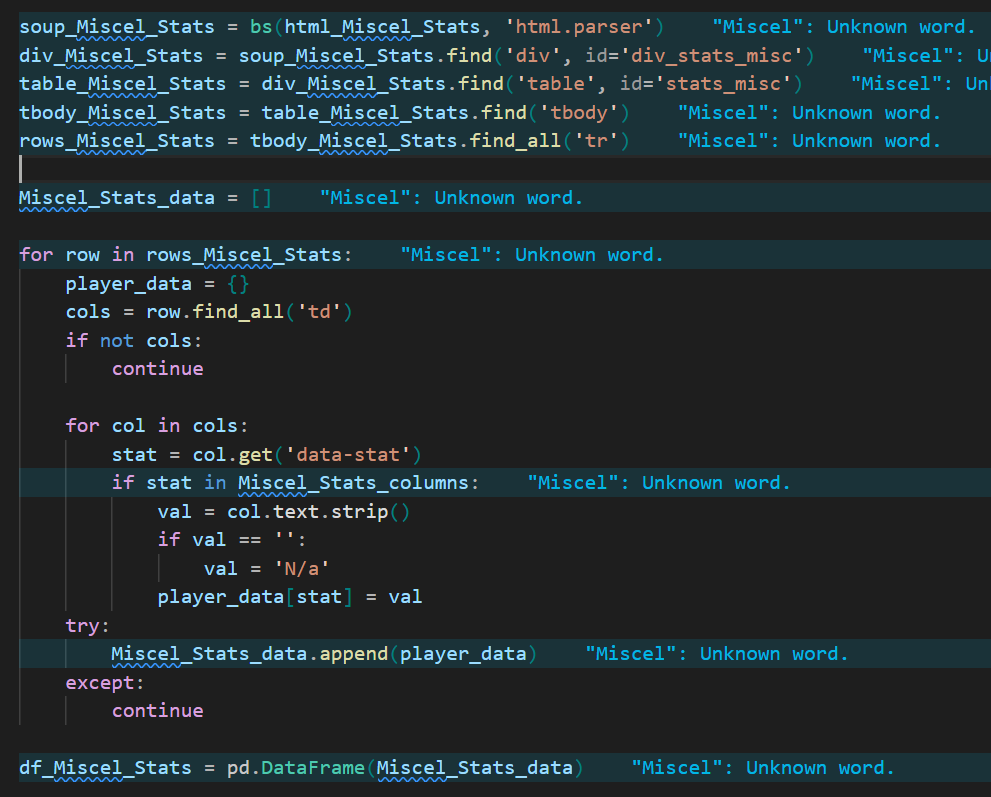


* Lấy dữ liệu từ mỗi dòng của bảng và xử lý
* Tạo DataFrame từ danh sách Possession\_data

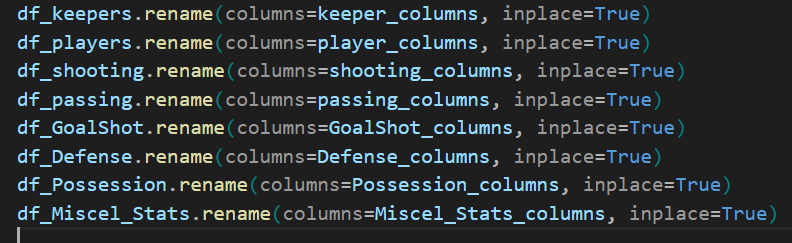
1. Lấy dữ liệu Miscellaneous Stat:



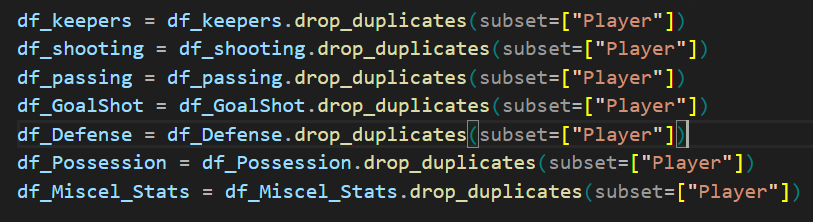
* Truy cập vào trang Miscellaneous Stat
* Mở trình duyệt, chờ 10s để tải trang, lấy mã HTML
* Khai báo các chỉ số và tên cột tương ứng trong DataFrame



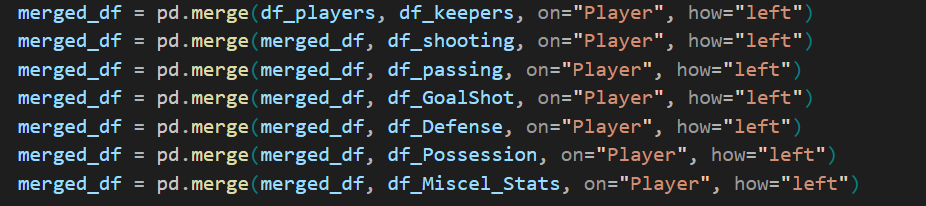
* Lấy dữ liệu từ mỗi dòng của bảng và xử lý
* Tạo DataFrame từ danh sách Miscel\_Stats\_data



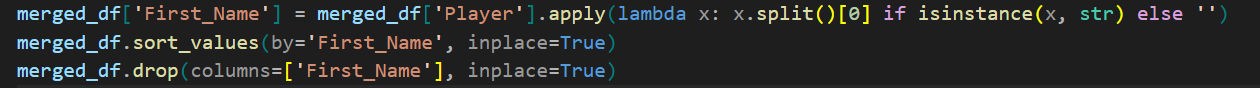
* Đổi tên các cột theo chuẩn để gộp dữ liệu.



* Xoá các trùng lặp theo tên (Player)



* Lần lượt ghép các bảng dữ liệu lại với nhau

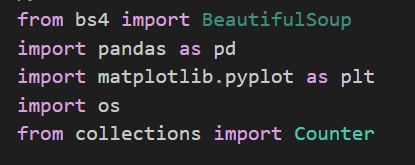


* Sắp xếp lại các cầu thủ theo First Name



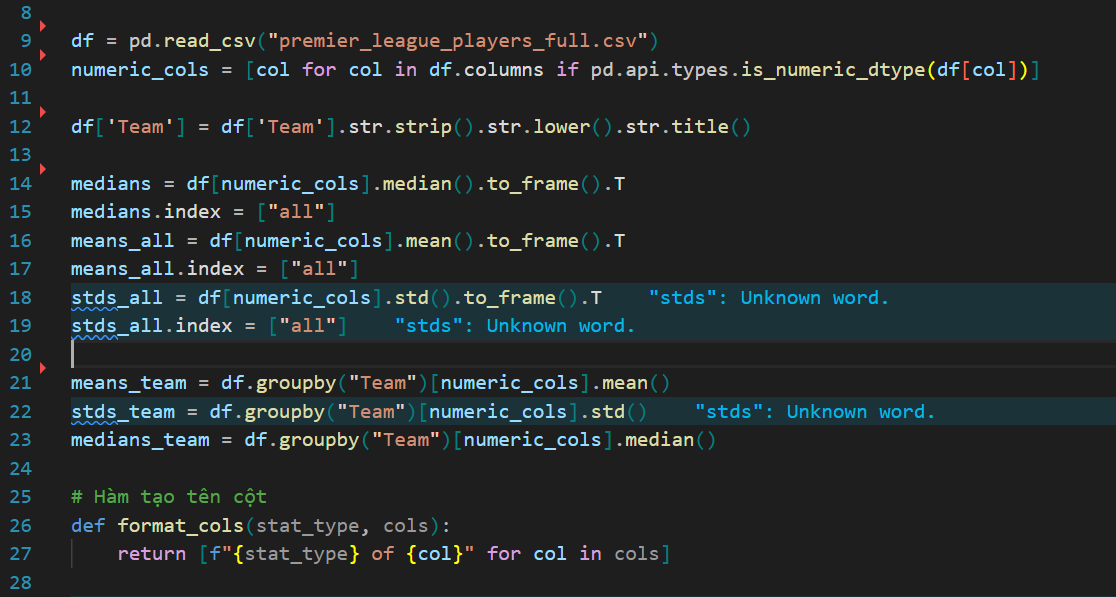
* Điền “N/a” vào các khoảng trống
* In ra file csv có tên BTL1

1. **Bài tập lớn 2:**

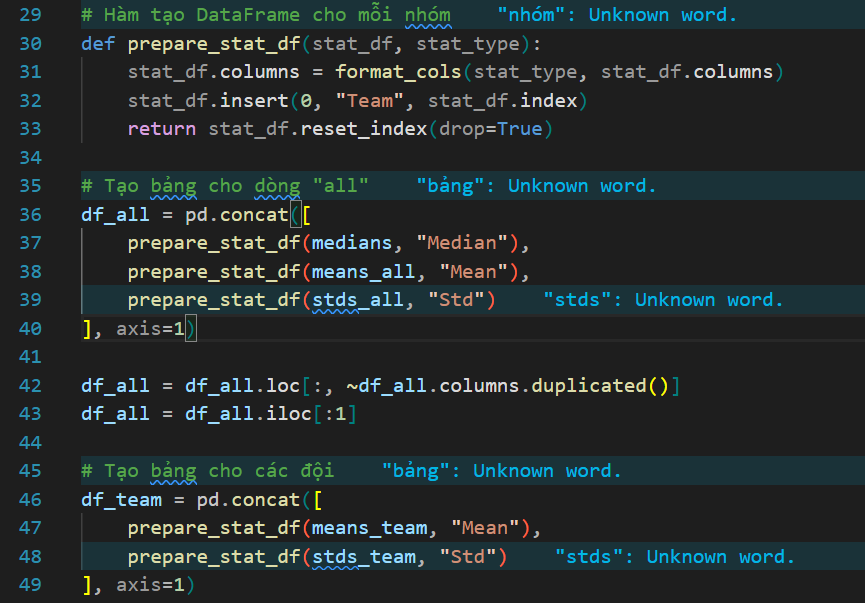
****

Khai báo các thư viện cần thiết

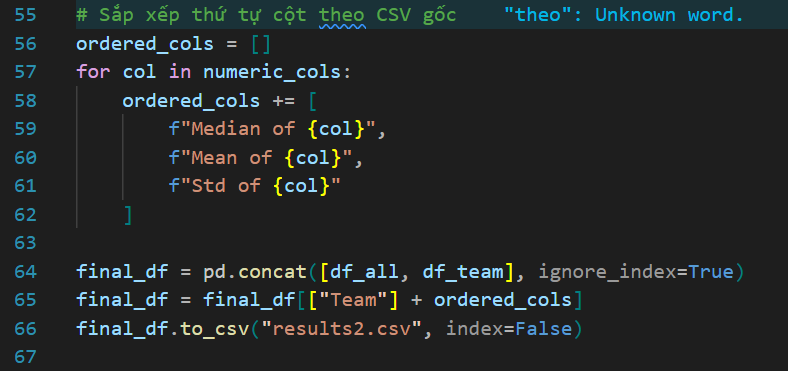
1. Phần1: Tính toán và in ra file csv:



* Dòng 9: Đọc dữ liệu từ file CSV chứa thông tin cầu thủ.
* Dòng 10: Lấy danh sách các cột có kiểu dữ liệu số (numeric) để tính toán thống kê.
* Dòng 12: Chuẩn hóa tên đội bóng: xóa khoảng trắng thừa, chuyển về chữ thường, rồi viết hoa từng chữ cái đầu.
* Dòng 14 -19: Tính trung vị, trung bình, và độ lệch chuẩn cho toàn giải.
* Dòng 21-23: Tính các chỉ số trên theo từng đội bóng.
* Dòng 21-23: Hàm tạo tên cột

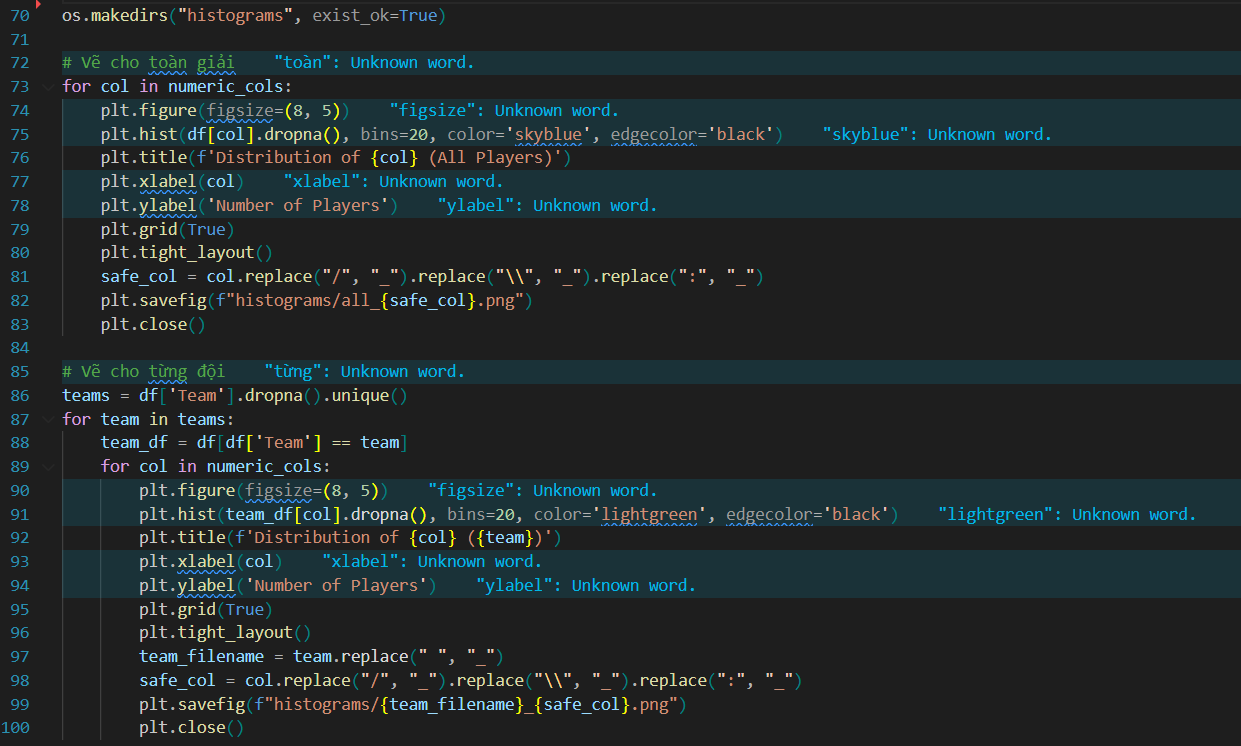


* Dòng 30-33: Hàm tạo DataFrame cho mỗi nhóm
* Dòng 36-40: Tạo bảng cho dòng “all”
* Dòng 46 -49: Tạo bảng cho các đội



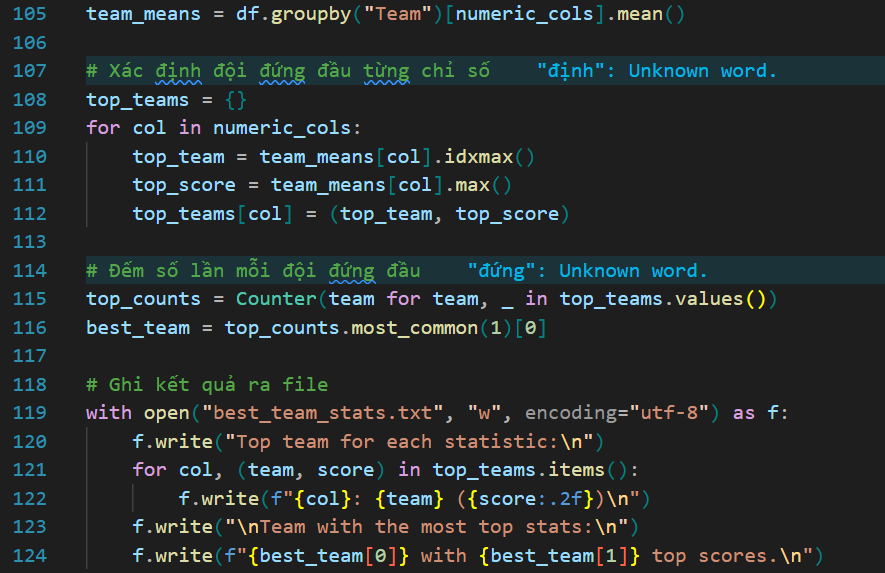
* Dòng 56-61: Sắp xếp thứ tự cột theo csv gốc
* Dòng 64-66: Gộp bảng “all” và bảng “team” -> Sắp xếp lại các cột cho đúng thứ tự -> xuất ra file csv tên “results2.csv”

1. Vẽ Histogram:



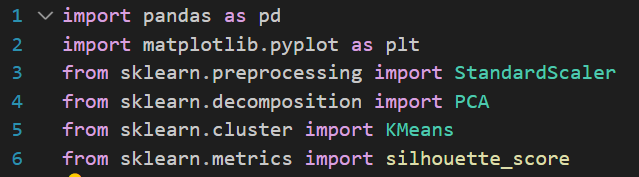
* Dòng 70: Tạo thư mục Histogram nếu chưa tồn tại
* Dòng 73-83 **( Vẽ Histogram cho toàn giải)**: Lặp qua từng chỉ số, vẽ histogram bằng matplotlib.pyplot và lưu lại thành ảnh dưới định dạng .png
* Dòng 86-100 **(Vẽ Histogram cho từng đội):** Tương tự như toàn giải, sử dụng điều kiện lọc df[df['Team'] == team] để vẽ histogram chỉ cho cầu thủ thuộc đội đó.

1. Tìm ra đội mạnh nhất:

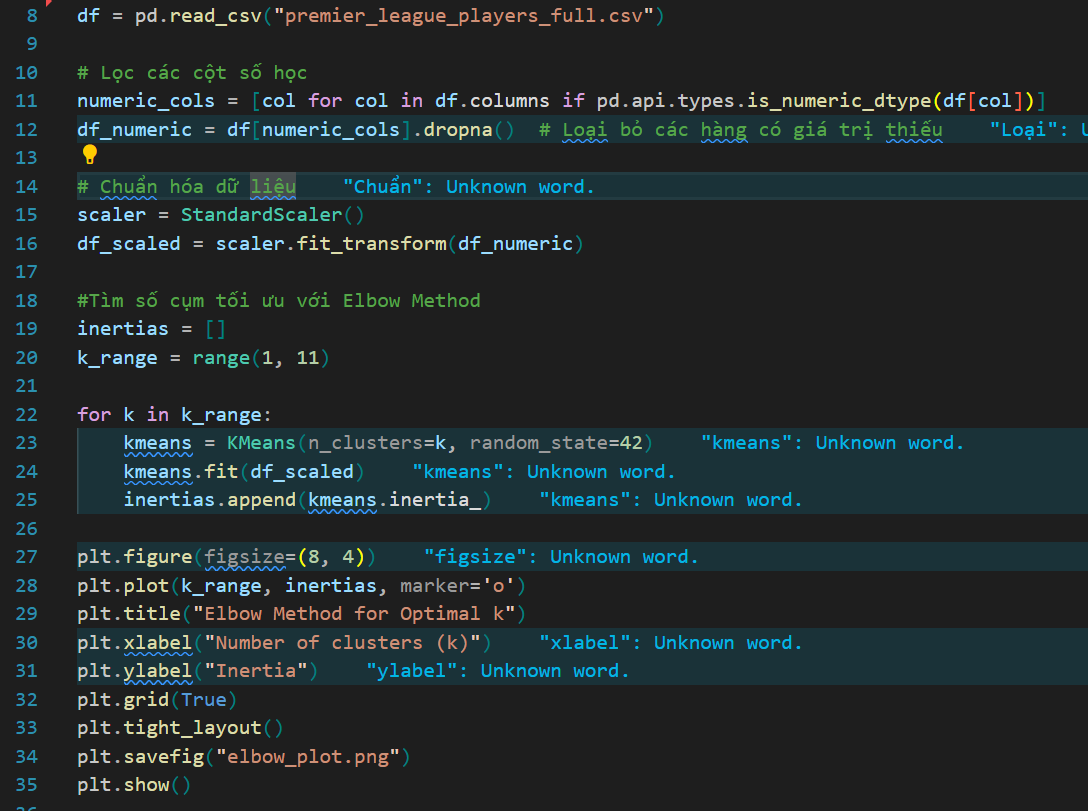


* Dòng 105: Tính trung bình các chỉ số theo đội
* Dòng 108-112: Xác định các đội đứng đầu từng chỉ số
* Dòng 114-116: Dếm số lần đứng đầu của mỗi đội
* Dòng 119-124: Ghi kết quả vào file “best\_team\_stats.txt”

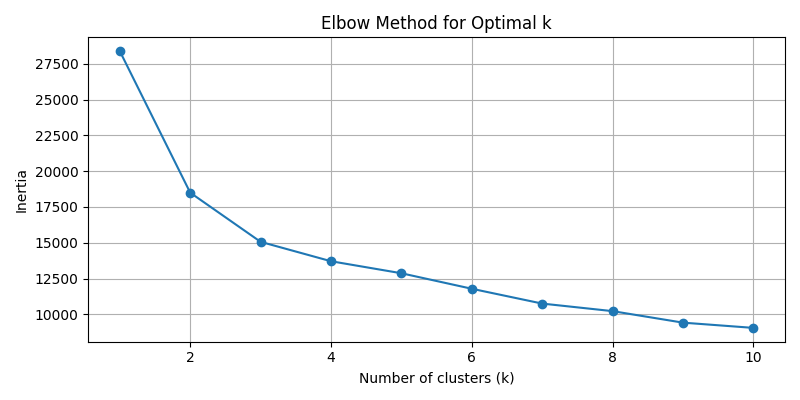
1. Bài tập lớn 3:



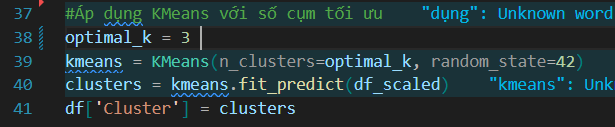
* Khai báo các thư viện cần thiết



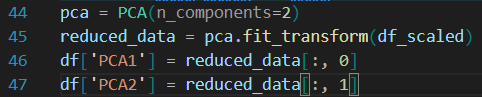
* Dòng 8: Đọc dữ liệu thống kê cầu thủ từ file CSV.
* Dòng 11-12: Lọc ra các cột có giá trị số (numeric) để dùng cho phân cụm, bỏ các hang có giá trị tối thiểu.
* Dòng 15-16: Chuẩn hóa dữ liệu về cùng thang đo (trung bình = 0, độ lệch chuẩn = 1)
* Dòng 19-25: Ta chạy thử K-means các giá trị từ 1->10. Tính tổng bình phương khoảng cách từ điểm đến tâm cụm, để xác định độ “chặt” của cụm
* Dòng 27-35: Vẽ biểu đồ Elbow Plot để tìm "k điểm gấp khúc", là số cụm tối ưu (optimal\_k)



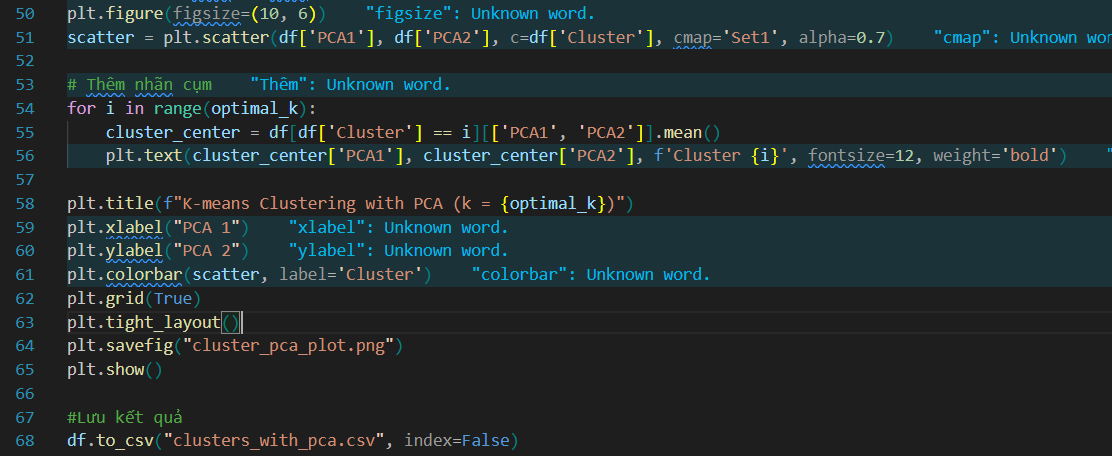
* Dựa vào biểu đồ Elbow, ta chọn được k = 3 là điểm gấp khúc



* Chạy phân cụm KMeans với số cụm tối ưu và gán nhãn cụm vào dataframe.



* Giảm chiều dữ liệu với PCA để vẽ biểu đồ

****

* Dòng 50-51: Tạo một hình mới với kích thước 10 inch × 6 inch.

Vẽ các điểm theo trục hoành là PCA1, trục tung là PCA2, tô màu theo nhãn cụm(Cluster), dùng bảng màu “Set1”, alpha = 0.7 -> làm cho điểm mơ đi để dễ nhìn thấy các điểm trông lên nhau

* Dòng 54-56: Thêm nhãn cụm vào vị trí trung tâm của từng cụm.
* Dòng 58-65:

- Đặt tiêu đề cho biểu đồ là: **"K-means Clustering with PCA (k =3)"**

- Gắn trục x là PCA1

- Gắn trục y là PCA2

- Thêm màu vào bên cạnh để biểu diễn ý nghĩa màu: mỗi màu ứng với 1 cụm (cluster)

- Bật lưới nền giúp dễ nhìn hơn.

- Tự động **canh chỉnh bố cục** để không bị chồng chữ, sát mép.

- Lưu biểu đồ thành file ảnh PNG với tên “**cluster\_pca\_plot.png**”

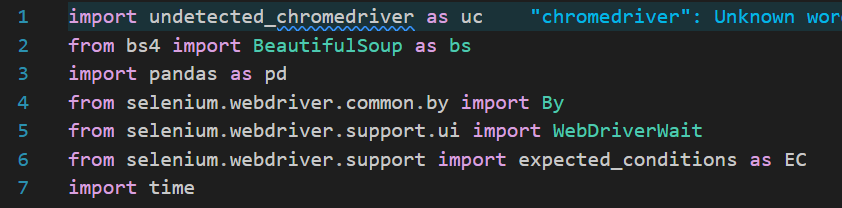
- Hiển thị biểu đồ ra màn hình.

A graph with red and grey dots

AI-generated content may be incorrect.

* Cluster 0: Các hậu vệ/ tiền vệ phòng ngự/tiền vệ trung tâm(ít bàn, ít kiến tạo, tắc bóng nhiều)
* Cluster 1: Các tiền vệ trung tâm / tiền vệ công/ tiền đạo (ghi nhiều bàn, kiến tạo cao, nhiều đường chuyền)
* Cluster 2: Các thủ môn (chỉ số cứu thua cao, ít tham gia tấn công)
* Dòng 68: Lưu toàn bộ bảng dữ liệu (gồm cột cụm và PCA) vào file CSV để phân tích tiếp.

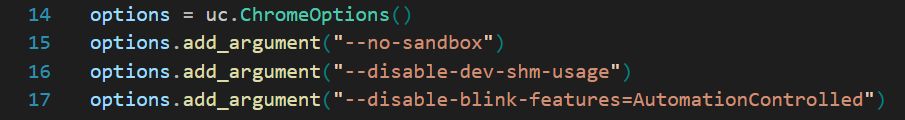
1. Bài tập lớn 4:
2. Phần 1:



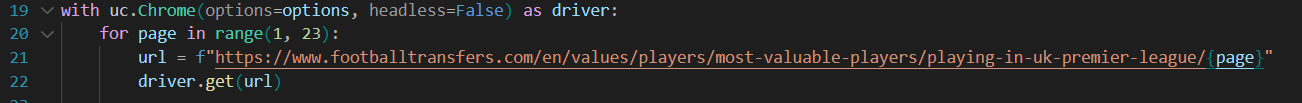
* Khai báo các thư viện cần thiết



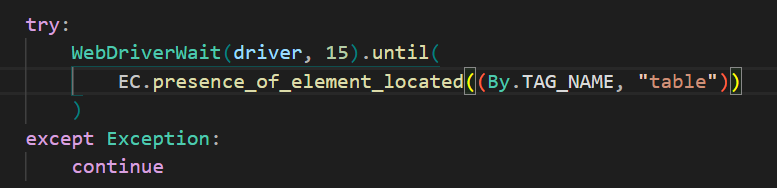
* Đọc file CSV đã có chứa danh sách đầy đủ cầu thủ
* Tạo tập hợp tên cầu thủ (existing\_players) đã được chuyển về chữ thường, xóa khoảng trắng — để so sánh tên dễ dàng hơn khi lấy từ web.



* Tạo tùy chọn trình duyệt (Chrome) sao cho trình duyệt tránh bị phát hiện là bot tự động



* Lặp qua 22 trang của bảng giá trị cầu thủ Premier League.
* Truy cập vào từng trang bằng trình duyệt tự động (driver.get(url)).



* Sử dụng WebDriverWait để đợi bảng dữ liệu HTML xuất hiện trong tối đa 15 giây trước khi phân tích.



* Dòng 31-32: Lấy mã HTML của trang hiện tại, phân tích bằng BeautifulSoup
* Dòng 33: Tìm bảng HTML chứa dữ liệu cần lấy dựa theo class định danh.
* Dòng 38: Lấy tất cả hàng (mỗi cầu thủ một hàng).
* Dòng 39-42: Duyệt qua các thẻ ”tr”, lấy phần chứa tên các cầu thủ và giá trị chuyển nhượng
* Dòng 44-53: Chuyển “name” về định dạng thường để so sánh

Nếu tên cầu thủ tồn tại trong tập existing\_players, thì thêm vào danh sách data(chuyển lại title() cho đẹp)

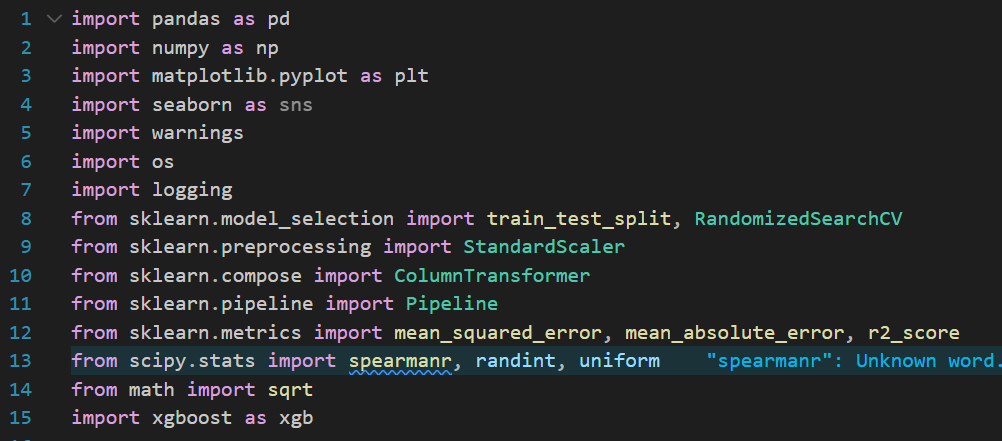
Transfer Value là giá trị

* Dòng 55: Tạm dừng 5s để tránh nguy cơ bị chặn

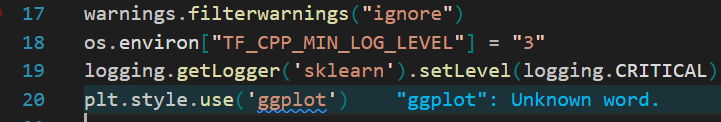


* Tạo DataFrame để lưu danh sách “data” và in ra file csv tên “BTL4.csv”

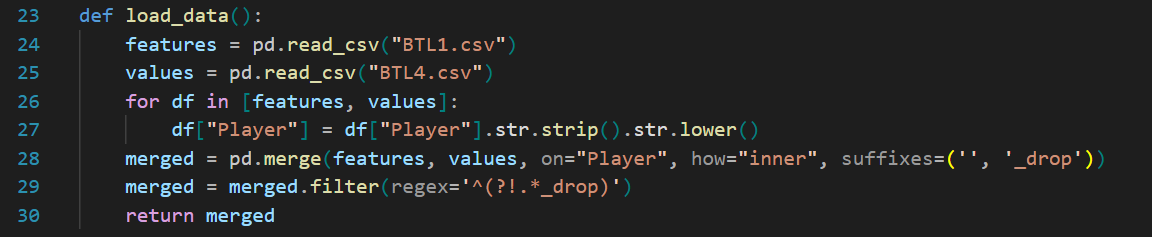
1. Phần 2:



* Khai báo các thư viện cần thiết



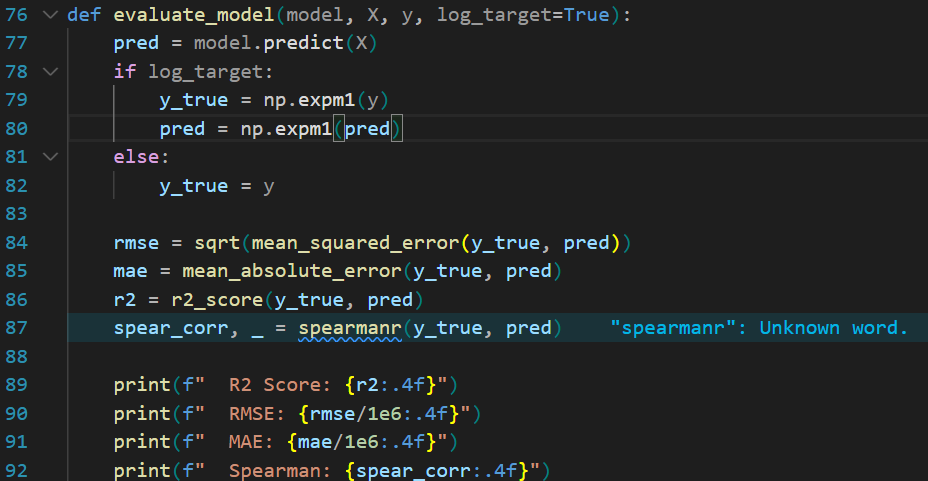
* Giảm thiểu sự nhiễu loạn đầu ra khi chạy mô hình, giúp hiển thị kết quả rõ rang



* Nhiệm vụ tải, chuẩn hóa và hợp nhất dữ liệu từ hai file CSV liên quan đến cầu thủ: đặc trưng hiệu suất và giá trị chuyển nhượng



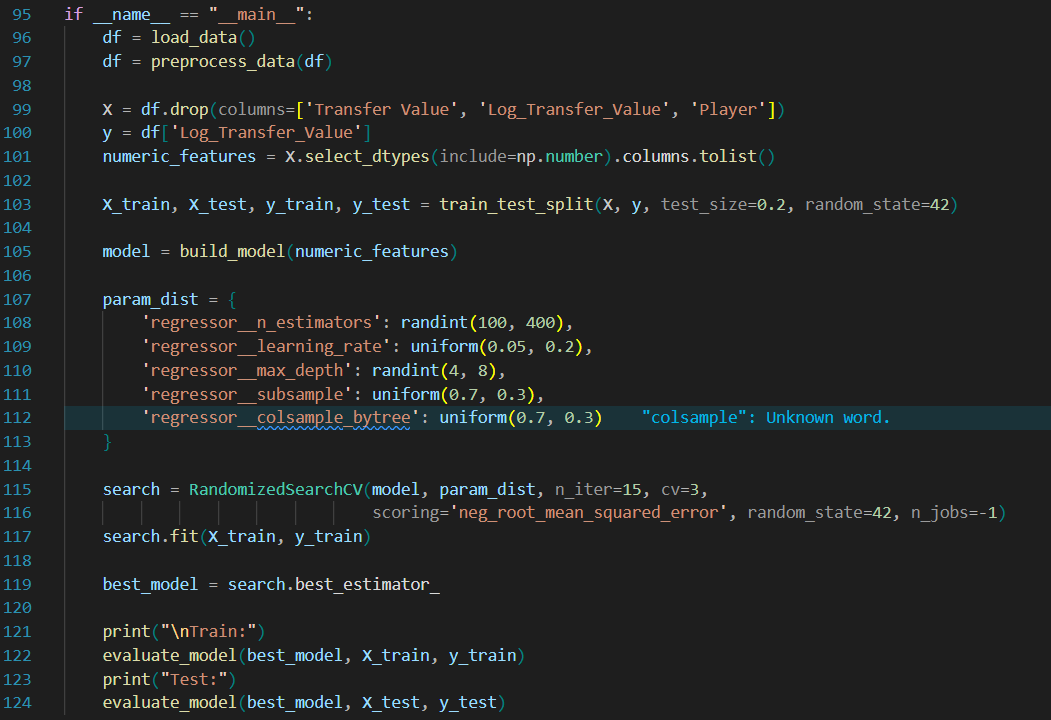
* Chuyển đổi giá trị chuyển nhượng về số (convert\_value)
* Loại bỏ cầu thủ có giá trị chuyển nhượng < 1,000 euro (lọc nhiễu).
* Chuyển đổi các cột Goal, Assists, Age về dạng số nếu cần
* Điền khuyết (fillna) bằng trung vị cho các cột số.



* Dùng để đánh giá chất lượng dự đoán của mô hình hồi quy sau khi đã huấn luyện:
* Dòng 77: Dự đoán giá trị đầu ra (Transfer Value) từ dữ liệu đầu vào X
* Dòng 78-82: Nếu log\_target=True (mặc định), có nghĩa là y là log(Transfer Value + 1)

Hàm np.expm1(x) = exp(x) - 1 được dùng để khôi phục lại giá trị gốc từ giá trị log-transformed.

* Dòng 84: RMSE: sai số trung bình bình phương căn – càng thấp càng tốt.
* Dòng 85: MAE: trung bình khoảng cách tuyệt đối giữa dự đoán và thực tế.
* Dòng 86: R²: đo mức độ mô hình giải thích phương sai trong dữ liệu thực (1 là tốt nhất, 0 là ngẫu nhiên).
* Dòng 87: Spearman correlation: đo độ tương quan thứ hạng giữa dự đoán và thực tế.
* Dòng 89-92: In kết quả đánh giá



* Dòng 95: Khai báo hàm main
* Dòng 96-97: Gọi hàm load\_data() để tải và hợp nhất dữ liệu từ các file CSV.

Gọi hàm preprocess\_data() để chuyển đổi, làm sạch và tạo đặc trưng mới

* Dòng 99-100: X: ma trận đặc trưng

Y: biến mục tiêu là giá trị chuyển nhượng đã log hóa (Log\_Transfer\_Value) để làm trơn phân phối dữ liệu.

* Dòng 101: Lọc ra các đặc trưng dạng số để chuẩn hóa (scale) sau này bằng StandardScaler
* Dòng 103: Chia dữ liệu thành 80% để huấn luyện và 20% để kiểm tra, random\_state=42 giúp tái tạo kết quả.
* Dòng 105: Gọi hàm build\_model() để tạo pipeline
* Dòng 107-113: Xác định không gian tìm kiếm siêu tham số cho mô hình XGBoost
* Dòng 115-117: Chạy tối ưu ngẫu nhiên 15 lần để chọn mô hình tốt nhất
* Dòng 119: Trích xuất pipeline tốt nhất sau quá trình RandomizedSearchCV.
* Dòng 121-124: Gọi hàm evaluate\_model() để in ra