HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CNTT - CLC

BỘ MÔN LẬP TRÌNH PYTHON

BÀI TẬP LỚN PYTHON

Giảng viên : **Kim Ngọc Bách**

Họ và tên : **Nguyễn Thiên Trung**

Mã SV : **B23DCCE093**

Lớp : **D23CQCE06 – B**

Năm học 2024 - 2025

MỤC LỤC

PHẦN I:

1. Yêu cầu đề bài

- Viết chương trình Python để thu thập dữ liệu thống kê của các cầu thủ bóng đá.
- Đối tượng: Tất cả các cầu thủ đã chơi hơn 90 phút trong mùa giải Ngoại hạng Anh 2024-2025.
- Nguồn dữ liệu**:** https://fbref.com/en/

2. Giải thích code

2.1. Các thư viện cần thiết

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from webdriver_manager.chrome import ChromeDriverManager
from bs4 import BeautifulSoup, Comment
import pandas as pd
import time
import re
```

- selenium.webdriver: Điều khiển trình duyệt web tự động.
- BeautifulSoup: Phân tích cú pháp HTML và trích xuất dữ liệu từ các phần tử HTML.
- pandas: Tạo và xử lý dữ liệu dưới dạng bảng.
- webdriver_manager: Tự động tải và cập nhật ChromeDriver.
- time: Điều khiển thời gian chờ.
- re: Biểu thức chính quy để xử lý dữ liệu văn bản.

2.2. Cấu hình URL và bảng dữ liệu

```
URL_CONFIG = {
    "standard": {"url": "https://fbref.com/en/comps/9/stats/Premier-League-Stats", "table_id": "stats_standard"},
    "keepers": {"url": "https://fbref.com/en/comps/9/keepers/Premier-League-Stats", "table_id": "stats_keepers"},
    "shooting": {"url": "https://fbref.com/en/comps/9/shooting/Premier-League-Stats", "table_id": "stats_shooting"},
    "passing": {"url": "https://fbref.com/en/comps/9/passing/Premier-League-Stats", "table_id": "stats_passing"},
    "passing_types": {"url": "https://fbref.com/en/comps/9/passing_types/Premier-League-Stats", "table_id": "stats_passing_type
    "gca": {"url": "https://fbref.com/en/comps/9/gca/Premier-League-Stats", "table_id": "stats_gca"},
    "defense": {"url": "https://fbref.com/en/comps/9/defense/Premier-League-Stats", "table_id": "stats_defense"},
    "possession": {"url": "https://fbref.com/en/comps/9/possession/Premier-League-Stats", "table_id": "stats_possession"},
    "misc": {"url": "https://fbref.com/en/comps/9/misc/Premier-League-Stats", "table_id": "stats_misc"},
}
```

- URL_CONFIG là từ điển chứa các URL và table_id tương ứng.
- Ví dụ:
    - standard: URL của bảng dữ liệu cơ bản (/stats/Premier-League-Stats).
    - keepers: URL của bảng dữ liệu thủ môn (/keepers/Premier-League-Stats).
- table_id được sử dụng để xác định bảng HTML cần trích xuất dữ liệu.

2.3. Hàm get_driver()

```
def get_driver():
    options = webdriver.ChromeOptions()
    options.add_argument(f'user-agent={USER_AGENT}')
    options.add_argument('--disable-gpu')
    options.add_argument('--no-sandbox')
    options.add_argument('--disable-dev-shm-usage')
    options.add_argument("start-maximized")
    options.add_experimental_option("excludeSwitches", ["enable-automation"])
    options.add_experimental_option('useAutomationExtension', False)
    try:
        service = Service(ChromeDriverManager().install())
        driver = webdriver.Chrome(service=service, options=options)
        driver.execute_script("Object.defineProperty(navigator, 'webdriver', {get: () => undefined})")
    except Exception as e:
        print(f"Lỗi khi khởi tạo ChromeDriver với webdriver_manager: {e}")
    return driver
```

- Tạo đối tượng ChromeOptions với các tùy chọn như:
    - Tắt GPU (--disable-gpu).
    - Vô hiệu hóa sandbox (--no-sandbox).
    - Giả lập user-agent.

- Khởi tạo webdriver.Chrome() với các tùy chọn đã cấu hình.
- Sử dụng webdriver_manager để tự động tải và cập nhật phiên bản ChromeDriver.
- Xử lý ngoại lệ khi không thể khởi tạo trình duyệt.

2.4. Hàm get_page_source_with_selenium(url, driver, table_id_hint)

```python
def get_page_source_with_selenium(url, driver, table_id_hint):
    driver.get(url)
    try:
        WebDriverWait(driver, 20).until(
            EC.presence_of_element_located((By.CSS_SELECTOR, f"div#div_{table_id_hint} table, table#{table_id_hint}"))
        )
    except Exception as e:
        pass
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
    time.sleep(0.5)
    driver.execute_script("window.scrollTo(0, 0);")
    time.sleep(0.5)
    return driver.page_source
```

- Mục đích: Lấy nội dung HTML từ URL.
- Sử dụng WebDriverWait để chờ cho đến khi bảng dữ liệu xuất hiện (table_id_hint).
- Cuộn trang lên và xuống để đảm bảo toàn bộ nội dung được tải.
- Trả về HTML của trang web dưới dạng chuỗi.

2.5. Hàm extract_table_from_html_fbref(html_content, table_id)

```python
def extract_table_from_html_fbref(html_content, table_id):
    soup = BeautifulSoup(html_content, 'html.parser')
    table_html = None
    comment = soup.find(string=lambda text: isinstance(text, Comment) and f'id="{table_id}"' in text)
    if comment:
        table_html = BeautifulSoup(comment.string, 'html.parser')
    else:
        table_html = soup
    table = table_html.find("table", {"id": table_id})
    if not table:
        return pd.DataFrame()
    data_rows = []
    for row in table.find("tbody").find_all("tr"):
        if row.has_attr('class') and ('thead' in row['class'] or \
                                      any(cls.startswith('spacer_') for cls in row['class'])):
            continue
        if not row.find_all(['th', 'td'], recursive=False):
            continue
        player_row_data = {}
        all_cells_in_row = row.find_all(['th', 'td'])
        for cell in all_cells_in_row:
            stat_name = cell.get('data-stat', None)
            if stat_name:
                stat_value = cell.get_text(strip=True)
                player_row_data[stat_name] = stat_value
        if "player" in player_row_data and player_row_data["player"]:
            data_rows.append(player_row_data)
    df = pd.DataFrame(data_rows)
    return df
```

- Mục đích: Trích xuất dữ liệu từ bảng HTML dựa trên table_id.
- Kiểm tra xem bảng có nằm trong phần bình luận HTML (Comment) hay không.
- Sử dụng BeautifulSoup để phân tích HTML và tìm bảng.
- Duyệt qua từng hàng (tr) và từng ô (td/th) để thu thập dữ liệu.
- Loại bỏ các hàng không chứa dữ liệu thực tế.
- Trả về DataFrame chứa dữ liệu đã xử lý.

2.6. Bản đồ cột dữ liệu (FBREF_TO_CSV_COLUMN_MAP):

```python
FBREF_TO_CSV_COLUMN_MAP = {
    'player': 'Player', 'nationality': 'Nation', 'team': 'Squad', 'position': 'Position',
    'age': 'Age', 'minutes': 'Minutes', 'games': 'Matches played', 'games_starts': 'Starts',
    'goals': 'Goals', 'assists': 'Assists', 'cards_yellow': 'Yellow cards', 'cards_red': 'Red cards',
    'xg': 'Expected: xG', 'xg_assist': 'Expected: xAG',
    'progressive_carries': 'Progression: PrgC', 'progressive_passes': 'Progression: PrgP', 'progressive_passes_
    'goals_per90': 'Per 90: Gls', 'assists_per90': 'Per 90: Ast', 'xg_per90': 'Per 90: xG',
    'gk_xg_against_per90': 'Per 90: xGA',
    'gk_goals_against_per90': 'Performance: GA90', 'gk_save_pct': 'Performance: Save%', 'gk_clean_sheets_pct':
    'shots_on_target_pct': 'Standard: SoT%', 'shots_on_target_per90': 'Standard: SoT/90', 'goals_per_shot': 'St
    'passes_completed': 'Total: Cmp', 'passes_pct': 'Total: Cmp%', 'passes_progressive_distance': 'Total: TotDi
    'passes_pct_short': 'Short: Cmp%', 'passes_pct_medium': 'Medium: Cmp%', 'passes_pct_long': 'Long: Cmp%',
    'assisted_shots': 'Expected: KP', 'passes_into_final_third': 'Expected: 1/3', 'passes_into_penalty_area': '
    'sca': 'SCA', 'sca_per90': 'SCA90', 'gca': 'GCA', 'gca_per90': 'GCA90',
    'tackles': 'Tackles: Tkl', 'tackles_won': 'Tackles: TklW',
    'challenges_attempted': 'Challenges: Att', 'challenges_lost': 'Challenges: Lost',
    'blocks': 'Blocks: Blocks', 'blocked_shots': 'Blocks: Sh', 'blocked_passes': 'Blocks: Pass', 'interceptions
    'touches': 'Touches: Touches', 'touches_def_pen_area': 'Touches: Def Pen', 'touches_def_3rd': 'Touches: Def
    'touches_mid_3rd': 'Touches: Mid 3rd', 'touches_att_3rd': 'Touches Att 3rd', 'touches_att_pen_area': 'Touch
    'take_ons_attempted': 'TakeOns: Att', 'take_ons_successful_pct': 'TakeOns: Succ%', 'take_ons_tackled_pct':
    'carries': 'Carries: Carries', 'carries_progressive_distance': 'Carries: PrgDist',
    'carries_into_final_third': 'Carries: 1/3', 'carries_into_penalty_area': 'Carries: CPA',
    'carries_miscontrols': 'Carries: Mis', 'carries_dispossessed': 'Carries: Dis',
    'passes_received': 'Receiving: Rec', 'fouls': 'Performance: Fls', 'fouled': 'Performance: Fld', 'offsides':
    'aerials_won': 'Aerials: Won', 'aerials_lost': 'Aerials: Lost', 'aerials_won_pct': 'Aerials: Won%'
```

- Chuyển đổi tên cột từ định dạng fbref sang định dạng yêu cầu (results.csv).
- Ví dụ:
  - player → Player
  - nationality → Nation
  - minutes → Minutes

2.7. Lọc cầu thủ >90 phút và sắp xếp theo tên riêng

```python
if 'Minutes' in final_df.columns:
    final_df['Minutes_temp_filter'] = final_df['Minutes'].astype(str).str.replace(',', '', regex=False)
    final_df['Minutes_temp_filter'] = pd.to_numeric(final_df['Minutes_temp_filter'], errors='coerce')
    final_df.dropna(subset=['Minutes_temp_filter'], inplace=True)
    final_df = final_df[final_df['Minutes_temp_filter'] > 90].copy()
    final_df.drop(columns=['Minutes_temp_filter'], inplace=True)
if 'Player' in final_df.columns and not final_df.empty:
    try:
        final_df['FirstNameTemp'] = final_df['Player'].astype(str).apply(
            lambda x: x.split(' ')[0] if x and ' ' in x else (x if x else "N/A_Player")
        )
        final_df.sort_values(by='FirstNameTemp', ascending=True, inplace=True)
        final_df.drop(columns=['FirstNameTemp'], inplace=True)
    except Exception as e:
        pass
```

- Lọc >90 phút:

- Nếu cột Minutes tồn tại:
  - Chuyển thành chuỗi, loại bỏ dấu phẩy.
  - Chuyển thành số, các giá trị không hợp lệ thành NaN.
  - Loại bỏ hàng có NaN trong cột tạm.
  - Lọc các hàng có Minutes_temp_filter > 90.
  - Xóa cột tạm.

- Sắp xếp theo tên riêng:

- Nếu cột Player tồn tại và DataFrame không rỗng:
  - Tạo cột tạm FirstNameTemp, lấy tên riêng (phần đầu tiên của tên, ví dụ: "Mohamed Salah" → "Mohamed").
  - Nếu tên không có khoảng trắng hoặc rỗng, sử dụng tên đầy đủ hoặc "N/A_Player".
  - Sắp xếp theo FirstNameTemp tăng dần.
  - Xóa cột tạm.
- Xử lý lỗi bằng try-except để tránh dừng chương trình.

2.8. Xuất file results.csv

```
results_df.fillna("N/a", inplace=True)
for col in results_df.columns:
    results_df[col] = results_df[col].apply(lambda x: "N/a" if str(x).strip() == "" else x)
try:
    results_df.to_csv("results.csv", index=False, encoding='utf-8-sig')
    print("✅ Lưu thành công dữ liệu")
except Exception as e:
    print(f"❌ Lỗi khi lưu file: {e}")
```

- Phần mã này chịu trách nhiệm:

- Đảm bảo tất cả giá trị thiếu hoặc rỗng trong DataFrame results_df được thay bằng "N/a".
- Lưu DataFrame results_df vào file results.csv với định dạng yêu cầu (không có cột chỉ số, mã hóa UTF-8-SIG để hỗ trợ ký tự đặc biệt).
- Thông báo thành công hoặc lỗi khi lưu file.

PHẦN II:

1. Yêu cầu đề bài

- Xác định top 3 cầu thủ có điểm cao nhất và thấp nhất cho mỗi chỉ số thống kê. Lưu kết quả vào file top_3.txt.
- Tìm giá trị trung vị (median) cho mỗi chỉ số thống kê.
- Tính giá trị trung bình (mean) và độ lệch chuẩn (standard deviation) cho mỗi chỉ số thống kê
- Lưu các kết quả (trung vị, trung bình, độ lệch chuẩn) vào file results2.csv theo định dạng được chỉ định (tổng quan và theo từng đội).
- Vẽ biểu đồ histogram thể hiện sự phân phối của mỗi chỉ số thống kê
- Xác định đội có điểm số cao nhất cho mỗi chỉ số thống kê.
- Dựa trên phân tích, đưa ra nhận định đội nào đang thi đấu tốt nhất trong mùa giải Ngoại hạng Anh 2024-2025.

2. Giải thích code

2.1. Các thư viện cần thiết

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
```

- pandas: Xử lý và phân tích dữ liệu.
- numpy: Hỗ trợ xử lý dữ liệu số.
- matplotlib.pyplot: Vẽ biểu đồ histogram.
- os: Quản lý thư mục và file.

2.2. Hàm identify_statistic_columns(df, exclude_cols=None)

```python
def identify_statistic_columns(df, exclude_cols=None):
    if exclude_cols is None:
        exclude_cols = ['Player', 'Nation', 'Team', 'Squad', 'Position']
    potential_stat_cols = []
    for col in df.columns:
        if col not in exclude_cols:
            try:
                numeric_col = pd.to_numeric(df[col].astype(str).str.replace('%', '', regex=False), errors='coerce')
                if numeric_col.notna().sum() > len(df) / 2:
                    potential_stat_cols.append(col)
            except:
                continue
    return potential_stat_cols
```

- Mục đích:

  - Xác định các cột trong DataFrame df chứa dữ liệu số hợp lệ để phân tích, loại bỏ các cột không phải số hoặc không cần thiết (như Player, Nation, Team, Squad, Position).

- Tham số:

  - df: DataFrame chứa dữ liệu (từ results.csv).
  - exclude_cols: Danh sách các cột cần loại bỏ (mặc định là ['Player', 'Nation', 'Team', 'Squad', 'Position']).

2.3. Hàm clean_numeric_column(series)

```python
def clean_numeric_column(series):
    series_str = series.astype(str).str.replace('%', '', regex=False).str.strip()
    series_str.replace(['', 'N/a', 'NaN', 'nan', 'None'], np.nan, inplace=True)
    return pd.to_numeric(series_str, errors='coerce')
```

- Mục đích:

  - Làm sạch và chuyển đổi một cột dữ liệu (Series) thành dạng số, xử lý các giá trị không hợp lệ hoặc chuỗi đặc biệt.

- Cách hoạt động:

- Tham số:
  - series: Một cột dữ liệu từ DataFrame (ví dụ: cột Goals hoặc Save%).
- Quy trình:
  - Chuyển cột thành chuỗi, loại bỏ ký tự % và khoảng trắng thừa.
  - Thay thế các giá trị rỗng hoặc không hợp lệ ('', 'N/a', 'NaN', 'nan', 'None') bằng np.nan.
  - Dùng pd.to_numeric để chuyển thành số, các giá trị không chuyển được thành NaN.
  - Trả về Series đã được làm sạch.

2.4. Hàm main_exercise_2()

Bước 1: Xác định cột đội và cột thống kê

```
team_column_name = 'Team' if 'Team' in df.columns else 'Squad'
if team_column_name not in df.columns:
    print(f"Lỗi: Không tìm thấy cột đội bóng trong results.csv.")
    return
stat_cols_to_analyze = identify_statistic_columns(df, exclude_cols=['Player', 'Nation', team_column_name, 'Position'])

if not stat_cols_to_analyze:
    print("Không xác định được cột thống kê nào để phân tích.")
    return
for col in stat_cols_to_analyze:
    df[col] = clean_numeric_column(df[col])
```

Mục đích:

- Xác định cột chứa tên đội (có thể là Team hoặc Squad).
- Xác định các cột thống kê số để phân tích.
- Làm sạch các cột thống kê.

Bước 2: Tạo thư mục lưu kết quả

```
if not os.path.exists("bai2_results"):
    os.makedirs("bai2_results")
if not os.path.exists("bai2_results/histograms"):
    os.makedirs("bai2_results/histograms")
```

- Tạo thư mục bai2_results và thư mục con histograms để lưu các file đầu ra.

Bước 3: Tìm top 3 cầu thủ cao nhất/thấp nhất

```
try:
    with open("bai2_results/top_3.txt", "w", encoding="utf-8") as f_top3:
        for stat in stat_cols_to_analyze:
            df_stat_cleaned = df[['Player', stat]].copy()
            df_stat_cleaned.dropna(subset=[stat], inplace=True)

            if df_stat_cleaned.empty:
                f_top3.write(f"\n--- Chỉ số: {stat} ---\n")
                f_top3.write("Không có đủ dữ liệu cầu thủ hợp lệ.\n")
                continue
            df_sorted_highest = df_stat_cleaned.sort_values(by=stat, ascending=False)
            df_sorted_lowest = df_stat_cleaned.sort_values(by=stat, ascending=True)

            f_top3.write(f"\n--- Chỉ số: {stat} ---\n")
            f_top3.write("\nTop 3 Cao nhất:\n")
            for i, row in df_sorted_highest.head(3).iterrows():
                f_top3.write(f"  {row['Player']}: {row[stat]:.2f}\n")

            f_top3.write("Top 3 Thấp nhất:\n")
            for i, row in df_sorted_lowest.head(3).iterrows():
                f_top3.write(f"  {row['Player']}: {row[stat]:.2f}\n")
    print("Hoàn thành: top_3.txt đã được tạo.")
except Exception as e:
    print(f"Lỗi khi tạo file top_3.txt: {e}")
```

- Mục đích: Xác định 3 cầu thủ có giá trị cao nhất và thấp nhất cho mỗi chỉ số, lưu vào top_3.txt.

- Chi tiết:

- Mở file top_3.txt để ghi (sử dụng mã hóa UTF-8 để hỗ trợ tiếng Việt hoặc ký tự đặc biệt).

- Duyệt qua từng cột thống kê:
  - Tạo DataFrame con chỉ chứa cột Player và cột thống kê.
  - Loại bỏ các hàng có giá trị NaN trong cột thống kê.
  - Nếu DataFrame rỗng (không có dữ liệu hợp lệ), ghi thông báo vào file và bỏ qua.
  - Sắp xếp DataFrame theo giá trị giảm dần (highest) và tăng dần (lowest).
  - Ghi 3 cầu thủ đầu tiên từ mỗi danh sách vào file, định dạng số với 2 chữ số thập phân.

Bước 4: Tính trung vị, trung bình, độ lệch chuẩn

```
results2_data = []
all_players_stats = {"Group": "all"}
for stat in stat_cols_to_analyze:
    all_players_stats[f"Median of {stat}"] = round (df[stat].median(), 2)
    all_players_stats[f"Mean of {stat}"] = round (df[stat].mean(), 2)
    all_players_stats[f"Std of {stat}"] = round (df[stat].std(), 2)
results2_data.append(all_players_stats)

teams = df[team_column_name].unique()
for team in teams:
    if pd.isna(team): continue
    df_team = df[df[team_column_name] == team]
    team_stats = {"Group": team}
    for stat in stat_cols_to_analyze:
        team_stats[f"Median of {stat}"] = round (df_team[stat].median(), 2)
        team_stats[f"Mean of {stat}"] = round (df_team[stat].mean(), 2)
        team_stats[f"Std of {stat}"] = round (df_team[stat].std(), 2)
    results2_data.append(team_stats)

df_results2 = pd.DataFrame(results2_data)
try:
    df_results2.to_csv("bai2_results/results2.csv", index=False, encoding="utf-8-sig")
    print("Hoàn thành: results2.csv đã được tạo.")
except Exception as e:
    print(f"Lỗi khi tạo file results2.csv: {e}")
```

- Mục đích: Tính trung vị, trung bình, và độ lệch chuẩn cho mỗi chỉ số, cả trên toàn bộ cầu thủ và từng đội, lưu vào results2.csv.

- Chi tiết:

- Tạo danh sách results2_data để lưu kết quả.
- Cho toàn bộ cầu thủ:
  - Tạo dictionary all_players_stats với khóa "Group": "all".
  - Duyệt qua các cột thống kê, tính median, mean, và std, làm tròn đến 2 chữ số thập phân.
  - Thêm dictionary vào results2_data.

- Cho từng đội:
  - Lấy danh sách các đội duy nhất từ cột team_column_name.
  - Bỏ qua các đội có giá trị NaN.
  - Tạo DataFrame con cho mỗi đội, tính median, mean, và std tương tự.
  - Thêm kết quả vào results2_data.
- Chuyển results2_data thành DataFrame và lưu vào results2.csv với mã hóa UTF-8-SIG (hỗ trợ tiếng Việt).

Bước 5: Vẽ biểu đồ histogram

```
for stat in stat_cols_to_analyze:
    plt.figure(figsize=(10, 6))
    df[stat].dropna().plot(kind='hist', bins=20, alpha=0.7, label='All Players', density=True)
    plt.title(f"Phân bố của chỉ số: {stat} (Toàn bộ cầu thủ)")
    plt.xlabel(stat)
    plt.ylabel("Tần suất (chuẩn hóa)")
    plt.legend()
    try:
        plt.savefig(f"bai2_results/histograms/hist_all_players_{stat.replace(':', '').replace('/', '_')}.png")
    except Exception as e:
        print(f"Lỗi khi lưu histogram (chỉ số {stat}): {e}")
    plt.close()
print("Hoàn thành: Biểu đồ Histogram đã được tạo.")
```

- Mục đích: Vẽ histogram cho phân bố của mỗi chỉ số trên toàn bộ cầu thủ, lưu vào thư mục bai2_results/histograms.

- Chi tiết:

- Duyệt qua các cột thống kê.
- Tạo biểu đồ mới với kích thước 10x6.

- Vẽ histogram với 20 bin, loại bỏ giá trị NaN, sử dụng density=True để chuẩn hóa tần suất.
- Thêm tiêu đề, nhãn trục, và chú thích.
- Lưu biểu đồ thành file PNG, thay thế các ký tự không hợp lệ (:, /) trong tên file.
- Đóng biểu đồ để tránh chiếm bộ nhớ.

Bước 6: Xác định đội có điểm số cao nhất

```
highest_scoring_teams_summary = []
for stat in stat_cols_to_analyze:
    if df[stat].dropna().empty: continue
    try:
        team_mean_stat = df.groupby(team_column_name)[stat].mean().sort_values(ascending=False)
        if not team_mean_stat.empty:
            top_team_for_stat = team_mean_stat.index[0]
            top_score_for_stat = team_mean_stat.iloc[0]
            highest_scoring_teams_summary.append(f"Chỉ số '{stat}': Đội cao nhất là {top_team_for_stat} (Trung bình:
    except Exception as e:
        highest_scoring_teams_summary.append(f"Chỉ số '{stat}': Lỗi khi tính toán ({e})")

try:
    with open("bai2_results/top_3.txt", "a", encoding="utf-8") as f_top3:
        f_top3.write("\n\n\n--- TÓM TẮT ĐỘI CÓ ĐIỂM SỐ TRUNG BÌNH CAO NHẤT CHO MỖI CHỈ SỐ ---\n\n")
        for summary_line in highest_scoring_teams_summary:
            f_top3.write(summary_line + "\n")
except Exception as e:
    print(f"Lỗi khi ghi tóm tắt đội điểm cao nhất: {e}")
```

- Mục đích: Tìm đội có giá trị trung bình cao nhất cho mỗi chỉ số và lưu vào top_3.txt.

- Chi tiết:

- Tạo danh sách highest_scoring_teams_summary.
- Duyệt qua các cột thống kê:
  - Nếu cột không có dữ liệu hợp lệ, bỏ qua.
  - Nhóm dữ liệu theo đội, tính trung bình, sắp xếp giảm dần.
  - Lấy đội có giá trị trung bình cao nhất và giá trị đó
  - Nếu có lỗi, ghi lỗi vào danh sách.
- Mở top_3.txt ở chế độ append ("a"), thêm tiêu đề và danh sách đội cao nhất.

Phần III:

1. Yêu cầu đề bài

- Đọc dữ liệu từ results.csv.
- Xác định các cột thống kê số phù hợp để phân cụm.
- Làm sạch và chuẩn hóa dữ liệu.
- Xác định số lượng cụm tối ưu (k) bằng phương pháp Elbow và Silhouette.
- Áp dụng K-means để phân cụm và PCA để giảm chiều.
- Vẽ các biểu đồ: Elbow, Silhouette, và biểu đồ phân cụm 2D (PCA).
- Lưu các biểu đồ vào thư mục bai3_results.

2. Giải thích code

2.1. Các thư viện cần thiết

- *pandas: Xử lý và phân tích dữ liệu.*
- *numpy: Hỗ trợ xử lý mảng số.*
- *sklearn.preprocessing.StandardScaler: Chuẩn hóa dữ liệu.*
- *sklearn.impute.SimpleImputer: Xử lý giá trị thiếu.*
- *sklearn.cluster.KMeans: Phân cụm bằng thuật toán K-means.*
- *sklearn.decomposition.PCA: Giảm chiều dữ liệu.*
- *sklearn.metrics.silhouette_score: Đánh giá chất lượng cụm.*
- *matplotlib.pyplot: Vẽ biểu đồ.*
- *seaborn: Tạo biểu đồ phân cụm đẹp hơn.*
- *os: Quản lý thư mục và file.*

2.2. identify_statistic_columns_for_clustering(df,excludecols=None)

```python
def identify_statistic_columns_for_clustering(df, exclude_cols=None):
    if exclude_cols is None:
        exclude_cols = ['Player', 'Nation', 'Squad', 'Position', 'Team']
    potential_stat_cols = []
    for col in df.columns:
        if col not in exclude_cols:
            try:
                if pd.api.types.is_numeric_dtype(df[col]):
                    if df[col].nunique(dropna=True) > 1:
                        potential_stat_cols.append(col)
            except Exception:
                continue
    return potential_stat_cols
```

- Mục đích:

- Xác định các cột trong DataFrame df chứa dữ liệu số hợp lệ để sử dụng cho phân cụm, loại bỏ các cột không phải số hoặc không cần thiết (như Player, Nation, Squad, Position, Team)

- Cách hoạt động:

- Tham số:
  - df: DataFrame chứa dữ liệu từ results.csv.
  - exclude_cols: Danh sách các cột cần loại bỏ (mặc định: ['Player', 'Nation', 'Squad', 'Position', 'Team']).

2.3. Hàm clean_and_convert_to_numeric(df, stat_cols)

```python
def clean_and_convert_to_numeric(df, stat_cols):
    df_cleaned = df.copy()
    for col in stat_cols:
        if df_cleaned[col].dtype == 'object':
            df_cleaned[col] = df_cleaned[col].astype(str).str.replace('%', '', regex=False)
            df_cleaned[col] = pd.to_numeric(df_cleaned[col], errors='coerce')
        elif not pd.api.types.is_numeric_dtype(df_cleaned[col]):
            df_cleaned[col] = pd.to_numeric(df_cleaned[col], errors='coerce')
    return df_cleaned
```

- Mục đích:

- Làm sạch và chuyển đổi các cột thống kê thành dạng số, xử lý các giá trị không hợp lệ hoặc chuỗi (như phần trăm).

- Cách hoạt động:

- Tạo bản sao của DataFrame (df_cleaned).
- Duyệt qua từng cột trong stat_cols
- Trả về DataFrame đã làm sạch.

2.4. Hàm main_exercise_3()

Bước 1: Lưu thông tin cầu thủ và xác định cột thống kê

```
player_info_df = df_input[['Player', 'Team' if 'Team' in df_input.columns else 'Squad']].copy()

stat_cols_for_clustering = identify_statistic_columns_for_clustering(df_input)
if not stat_cols_for_clustering:
    print("Không xác định được cột thống kê nào phù hợp cho clustering.")
    return
```

- Mục đích:

- Lưu thông tin cơ bản của cầu thủ (Player, Team hoặc Squad).
- Xác định cột số để phân cụm.

- Chi tiết:

- Tạo player_info_df chứa cột Player và cột đội (Team nếu có, nếu không thì Squad).
- Gọi identify_statistic_columns_for_clustering() để lấy danh sách cột số.
- Thoát nếu không tìm thấy cột phù hợp.

Bước 2: Làm sạch và chuẩn hóa dữ liệu

```
df_stats = df_input[stat_cols_for_clustering].copy()
df_stats = clean_and_convert_to_numeric(df_stats, stat_cols_for_clustering)
imputer = SimpleImputer(strategy='mean')
df_imputed = imputer.fit_transform(df_stats)
df_processed = pd.DataFrame(df_imputed, columns=df_stats.columns, index=df_stats.index)
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df_processed)
df_scaled = pd.DataFrame(df_scaled, columns=df_processed.columns, index=df_processed.index)
```

- Mục đích: Chuẩn bị dữ liệu cho phân cụm bằng cách làm sạch, thay thế giá trị thiếu, và chuẩn hóa.

- Chi tiết:

- Tạo df_stats chứa các cột thống kê.
- Gọi clean_and_convert_to_numeric() để chuyển dữ liệu thành số.
- Sử dụng SimpleImputer(strategy='mean') để thay thế NaN bằng giá trị trung bình của cột.
- Tạo df_processed từ dữ liệu đã thay thế.
- Sử dụng StandardScaler để chuẩn hóa dữ liệu (trừ trung bình, chia cho độ lệch chuẩn), tạo df_scaled.

Bước 3: Phương pháp Elbow để chọn k

```
wcss = []
k_range = range(2, 11)
for i in k_range:
    kmeans_elbow = KMeans(n_clusters=i, init='k-means++', n_init='auto', random_state=42)
    kmeans_elbow.fit(df_scaled)
    wcss.append(kmeans_elbow.inertia_)
plt.figure(figsize=(10, 6))
plt.plot(k_range, wcss, marker='o', linestyle='--')
plt.title('Phương pháp Elbow để xác định k tối ưu')
plt.xlabel('Số lượng nhóm (k)')
plt.ylabel('WCSS (Inertia)')
plt.xticks(list(k_range))
elbow_plot_path = os.path.join(output_dir_bai3, "kmeans_elbow_plot.png")
plt.savefig(elbow_plot_path)
plt.close()
print(f"Đã lưu biểu đồ Elbow")
```

- Mục đích: Sử dụng phương pháp Elbow để xác định số lượng cụm tối ưu (k) bằng cách đo WCSS (Within-Cluster Sum of Squares).

- Chi tiết:

- Thử k từ 2 đến 10.
- Với mỗi k, chạy K-means (init='k-means++' để khởi tạo tốt hơn, random_state=42 để cố định kết quả).
- Lưu WCSS (inertia_) vào danh sách wcss.
- Vẽ biểu đồ đường với k trên trục x và WCSS trên trục y, lưu vào kmeans_elbow_plot.png.

Bước 4: Phương pháp Silhouette để chọn

```
silhouette_scores = []
for i in k_range:
    kmeans_silhouette = KMeans(n_clusters=i, init='k-means++', n_init='auto', random_state=42)
    cluster_labels = kmeans_silhouette.fit_predict(df_scaled)
    try:
        silhouette_avg = silhouette_score(df_scaled, cluster_labels)
        silhouette_scores.append(silhouette_avg)
    except ValueError:
        silhouette_scores.append(-1)
plt.figure(figsize=(10, 6))
plt.plot(k_range, silhouette_scores, marker='o', linestyle='--')
plt.title('Phân tích Silhouette để xác định k tối ưu')
plt.xlabel('Số lượng nhóm (k)')
plt.ylabel('Silhouette Score Trung bình')
plt.xticks(list(k_range))
silhouette_plot_path = os.path.join(output_dir_bai3, "kmeans_silhouette_plot.png")
plt.savefig(silhouette_plot_path)
plt.close()
print(f"Đã lưu biểu đồ Silhouette")
```

- Mục đích: Sử dụng Silhouette Score để đánh giá chất lượng cụm và chọn k.

- Chi tiết:

- Thử k từ 2 đến 10, chạy K-means, lấy nhãn cụm, tính silhouette
- Nếu lỗi (ví dụ: cụm không hợp lệ), gán score -1.
- Vẽ biểu đồ đường với k trên trục x và Silhouette Score trên trục y, lưu vào kmeans_silhouette_plot.png.

Bước 5: Chọn k và phân cụm

```
optimal_k_silhouette = -1
if any(score > -1 for score in silhouette_scores):
    optimal_k_silhouette = list(k_range)[silhouette_scores.index(best_silhouette_score)]
else:
    print("Không thể tính toán Silhouette Score hợp lệ cho các giá trị k đã thử.")
    print("Vui lòng chọn k dựa trên biểu đồ Elbow hoặc kiến thức chuyên môn.")
chosen_k = optimal_k_silhouette if optimal_k_silhouette > 1 else 3
kmeans = KMeans(n_clusters=chosen_k, init='k-means++', n_init='auto', random_state=42)
cluster_labels = kmeans.fit_predict(df_scaled)
df_results_with_clusters = df_input.loc[df_scaled.index].copy()
df_results_with_clusters['Cluster'] = cluster_labels
cluster_analysis_df = df_processed.copy()
cluster_analysis_df['Cluster'] = cluster_labels
cluster_summary = cluster_analysis_df.groupby('Cluster')[stat_cols_for_clustering].mean().round(2)
```

- Mục đích: Chọn số lượng cụm k tối ưu và phân cụm cầu thủ.

- Chi tiết:

  - Kiểm tra xem có Silhouette Score hợp lệ không:
    - Nếu có, chọn k có score cao nhất (optimal_k_silhouette).
    - Nếu không, in thông báo và yêu cầu chọn k từ biểu đồ Elbow.
  - Chọn chosen_k: Dùng optimal_k_silhouette nếu hợp lệ, nếu không thì mặc định k=3.
  - Chạy K-means với chosen_k, lấy nhãn cụm (cluster_labels).
  - Tạo df_results_with_clusters: Thêm cột Cluster vào dữ liệu gốc.
  - Tạo cluster_analysis_df: Thêm cột Cluster vào dữ liệu đã xử lý.
  - Tính trung bình các chỉ số theo cụm (cluster_summary), làm tròn đến 2 chữ số.

Bước 6: Giảm chiều và trực quan hóa

```
pca = PCA(n_components=2, random_state=42)
df_pca = pca.fit_transform(df_scaled)
df_pca_plot = pd.DataFrame(data=df_pca, columns=['Principal Component 1', 'Principal Component 2'])
df_pca_plot['Cluster'] = cluster_labels
plt.figure(figsize=(12, 8))
sns.scatterplot(
    x="Principal Component 1", y="Principal Component 2",
    hue="Cluster",
    palette=sns.color_palette("hsv", chosen_k),
    data=df_pca_plot,
    legend="full",
    alpha=0.7
)
plt.title(f'Biểu đồ phân cụm cầu thủ 2D sử dụng PCA và K-means')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
pca_plot_path = os.path.join(output_dir_bai3, "pca_kmeans_2d_plot.png")
plt.savefig(pca_plot_path)
plt.close()
print(f"Đã lưu biểu đồ PCA 2D")
```

- Mục đích: Giảm chiều dữ liệu xuống 2D bằng PCA và vẽ biểu đồ phân cụm.

- Chi tiết:

- Sử dụng PCA(n_components=2) để giảm chiều dữ liệu thành 2 thành phần chính.
- Tạo df_pca_plot với cột Principal Component 1, Principal Component 2, và Cluster.
- Vẽ biểu đồ phân tán bằng seaborn.scatterplot, màu sắc theo cụm, lưu vào pca_kmeans_2d_plot.png.

Phần IV:

1. Yêu cầu đề bài

- Thu thập giá trị chuyển nhượng của cầu thủ mùa giải 2024-2025 từ https://www.footballtransfers.com.
- Lưu ý quan trọng: Chỉ thu thập cho những cầu thủ có thời gian thi đấu lớn hơn 900 phút (khác với yêu cầu >90 phút ở Phần I).
- Đề xuất một phương pháp để ước tính giá trị cầu thủ.
- Giải thích cách bạn chọn đặc trưng (features) và mô hình (model) cho việc ước tính này.

2. Giải thích code

2.1. Các thư viện cần thiết

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from webdriver_manager.chrome import ChromeDriverManager
from bs4 import BeautifulSoup
import pandas as pd
import time
```

- selenium và webdriver_manager: Thu thập dữ liệu động từ trang web.
- bs4 (BeautifulSoup): Phân tích HTML.
- pandas: Xử lý và lưu trữ dữ liệu.
- time: Tạm dừng giữa các lần truy cập web để tránh bị chặn.

2.2. Hàm get_driver()

```python
def get_driver():
    chrome_options = Options()
    chrome_options.add_argument(f'user-agent={USER_AGENT}')
    chrome_options.add_argument('--disable-gpu')
    chrome_options.add_argument('--no-sandbox')
    chrome_options.add_argument('--disable-dev-shm-usage')
    chrome_options.add_argument("start-maximized")
    chrome_options.add_experimental_option("excludeSwitches", ["enable-automation"])
    chrome_options.add_experimental_option('useAutomationExtension', False)
    try:
        service = Service(ChromeDriverManager().install(), service_args=['--log-level=OFF'])
        driver = webdriver.Chrome(service=service, options=chrome_options)
        driver.execute_script("Object.defineProperty(navigator, 'webdriver', {get: () => undefined})")
    except Exception as e:
        print(f"Lỗi khi khởi tạo ChromeDriver với webdriver_manager: {e}")
        try:
            driver = webdriver.Chrome(options=chrome_options)
        except Exception as e_fallback:
            raise
    return driver
```

- Mục đích:

- Khởi tạo WebDriver (Chrome) với các tùy chọn để thu thập dữ liệu động từ FootballTransfers.com, tránh bị phát hiện là bot.

- Cách hoạt động:

- Tùy chọn Chrome:
  - user-agent: Giả lập trình duyệt Chrome trên Windows.
  - --disable-gpu, --no-sandbox, --disable-dev-shm-usage: Tối ưu hóa hiệu suất và ổn định.
  - start-maximized: Mở trình duyệt toàn màn hình.
  - excludeSwitches và useAutomationExtension: Vô hiệu hóa các dấu hiệu tự động hóa.
- Khởi tạo WebDriver:
  - Sử dụng webdriver_manager để tự động cài đặt ChromeDriver.
  - Nếu thất bại, thử khởi tạo trực tiếp ChromeDriver (fallback).
  - Vô hiệu hóa thuộc tính navigator.webdriver để tránh bị phát hiện.

2.3. Hàm get_page_source_with_selenium_and_wait(driver, url, wait_selector_css)

```python
def get_page_source_with_selenium_and_wait(driver, url, wait_selector_css):
    driver.get(url)
    try:
        WebDriverWait(driver, 20).until(
            EC.presence_of_element_located((By.CSS_SELECTOR, wait_selector_css))
        )
    except Exception as e:
        print(f"⚠️ Timeout hoặc không tìm thấy element. Lỗi: {e}")
        return None
    return driver.page_source
```

- Mục đích:

Tải nội dung HTML của một trang web bằng Selenium, chờ cho đến khi một phần tử cụ thể xuất hiện.

- Cách hoạt động:

- Tham số:
  - driver: WebDriver đã khởi tạo.
  - url: URL của trang cần tải.
  - wait_selector_css: Bộ chọn CSS của phần tử cần chờ (ở đây là bảng dữ liệu).
- Quy trình:
  - Truy cập url bằng driver.get(url).
  - Sử dụng WebDriverWait để chờ tối đa 20 giây cho đến khi phần tử có CSS wait_selector_css xuất hiện.
  - Nếu thành công, trả về driver.page_source (nội dung HTML).
  - Nếu lỗi (timeout hoặc phần tử không tìm thấy), in thông báo và trả về None.

2.4. Hàm extract_data_using_confirmed_selectors(html_content, url_for_logging="")

```python
def extract_data_using_confirmed_selectors(html_content, url_for_logging=""):
    soup = BeautifulSoup(html_content, 'html.parser')
    players_data = []
    table_class_confirmed = 'table table-hover no-cursor table-striped leaguetable mvp-table similar-players-table mb-0'
    table = soup.find('table', class_=table_class_confirmed)
    if not table:
        print(f"  Cảnh báo: Không tìm thấy bảng với class '{table_class_confirmed}' trên trang được phân tích (URL: {url_
        return players_data
    tbody = table.find('tbody')
    if not tbody:
        print(f"  Cảnh báo: Không tìm thấy tbody trong bảng trên trang (URL: {url_for_logging}).")
        return players_data
    rows = tbody.find_all('tr')
    print(f"  Phân tích {len(rows)} hàng từ bảng (URL: {url_for_logging}).")
    for i, row_html in enumerate(rows):
        try:
            skill_div = row_html.find('div', class_='table-skill__skill')
            pot_div = row_html.find('div', class_='table-skill__pot')
            skill_text = skill_div.text.strip() if skill_div and skill_div.text else None
            pot_text = pot_div.text.strip() if pot_div and pot_div.text else None
            skill_pot_value = "N/A"
            if skill_text and pot_text:
                try:
                    skill_val = float(skill_text)
                    pot_val = float(pot_text)
                    skill_pot_value = f"{skill_val}/{pot_val}"
                except ValueError:
                    pass
            player_span = row_html.find('span', class_='d-none')
            player_name_val = player_span.text.strip() if player_span and player_span.text else None
            team_span = row_html.find('span', class_='td-team__teamname')
            team_name_val = team_span.text.strip() if team_span and team_span.text else None
            etv_span = row_html.find('span', class_='player-tag')
            etv_value_val = etv_span.text.strip() if etv_span and etv_span.text else None
            if player_name_val:
                players_data.append({
                    "Player": player_name_val,
                    "Team": team_name_val,
                    "ETV": etv_value_val,
                    "Skill/Pot": skill_pot_value
                })
        except Exception as e:
            print(f"    Lỗi khi xử lý hàng {i+1} (URL: {url_for_logging}): {e}")
            continue
    return players_data
```

- Mục đích:

Trích xuất dữ liệu từ HTML của trang FootballTransfers.com, bao gồm tên cầu thủ, đội, ETV, và Skill/Pot.

- Cách hoạt động:

- Tham số:
    - html_content: Nội dung HTML của trang.
    - url_for_logging: URL để ghi log lỗi.

- Quy trình:
    - Tìm bảng với class xác định (table table-hover no-cursor...).
    - Phân tích HTML bằng Beautifulsoup
    - Nếu không tìm thấy bảng hoặc tbody, in cảnh báo và trả về danh sách rỗng.
    - Duyệt qua các hàng (tr) trong tbody:
    - Xử lý lỗi từng hàng, in thông báo nếu lỗi và tiếp tục.

2.5. Hàm normalize_player_name(name)

```
def normalize_player_name(name):
    if pd.isna(name) or name is None:
        return ""
    return str(name).lower().strip()
```

- Mục đích:

Chuẩn hóa tên cầu thủ để khớp giữa results.csv và dữ liệu từ FootballTransfers.com.

- Cách hoạt động:

- Tham số: name (tên cầu thủ).
- Quy trình:
    - Nếu tên là NaN hoặc None, trả về chuỗi rỗng.
    - Chuyển tên thành chuỗi, chuyển thành chữ thường, loại bỏ khoảng trắng thừa.

2.6. Khối chính (if __name__ == "__main__":)

Bước 1: Đọc và lọc dữ liệu từ results.csv

```
if __name__ == "__main__":
    try:
        df_results1 = pd.read_csv("results.csv", na_filter=False)
    except FileNotFoundError:
        print("❌ Lỗi: File 'results.csv' không tìm thấy. Hãy đảm bảo bạn đã chạy Bài 1 thành công.")
        exit()
    except Exception as e:
        print(f"❌ Lỗi khi đọc 'results.csv': {e}")
        exit()
    if 'Minutes' not in df_results1.columns or 'Player' not in df_results1.columns:
        print("❌ Lỗi: File 'results.csv' phải có cột 'Minutes' và 'Player'.")
        exit()
    df_results1['Minutes_Numeric'] = df_results1['Minutes'].astype(str).str.replace(',', '', regex=False)
    df_results1['Minutes_Numeric'] = df_results1['Minutes_Numeric'].replace(['N/a', ''], pd.NA)
    df_results1['Minutes_Numeric'] = pd.to_numeric(df_results1['Minutes_Numeric'], errors='coerce')
    df_results1.dropna(subset=['Minutes_Numeric'], inplace=True)
    players_over_900_min_df = df_results1[df_results1['Minutes_Numeric'] > 900].copy()
```

- Mục đích: Đọc results.csv, lọc các cầu thủ có thời gian thi đấu > 900 phút.

- Chi tiết:

- Đọc results.csv với na_filter=False để giữ nguyên các giá trị như "N/a".
- Kiểm tra lỗi: thoát nếu file không tồn tại hoặc có lỗi đọc.
- Kiểm tra cột Minutes và Player: thoát nếu thiếu.
- Chuyển cột Minutes thành số:
    - Loại bỏ dấu phẩy (ví dụ: "1,234" → "1234").
    - Thay "N/a" và chuỗi rỗng bằng pd.NA.
    - Chuyển thành số, loại bỏ hàng có NaN.
- Lọc các cầu thủ có Minutes_Numeric > 900, tạo players_over_900_min_df.

Bước 2: Chuẩn hóa tên cầu thủ

```
if players_over_900_min_df.empty:
    print("Thông báo: Không tìm thấy cầu thủ nào thi đấu trên 900 phút trong 'results.csv'. Kết thúc.")
    exit()
players_over_900_min_df['Player_Normalized_Results'] = players_over_900_min_df['Player'].apply(normalize_player_name)
set_players_over_900_min_normalized = set(players_over_900_min_df['Player_Normalized_Results'])
print(f"Thông tin: Đã xác định {len(set_players_over_900_min_normalized)} cầu thủ thi đấu > 900 phút từ results.csv.")
```

*- **Mục đích**: Chuẩn hóa tên cầu thủ và tạo tập hợp tên chuẩn hóa.*

*- **Chi tiết**:*

- *Thoát nếu không có cầu thủ nào >900 phút.*
- *Tạo cột Player_Normalized_Results bằng cách áp dụng normalize_player_name.*
- *Tạo tập hợp set_players_over_900_min_normalized để kiểm tra khớp tên.*

Bước 3: Thu thập dữ liệu từ FootballTransfers.com

```
try:
    active_driver = get_driver()
    print(f"\nThông tin: Bắt đầu cào dữ liệu từ {len(URLS_TO_SCRAPE)} trang trên footballtransfers.com...")
    for page_idx, current_page_url in enumerate(URLS_TO_SCRAPE, 1):
        html_source = get_page_source_with_selenium_and_wait(active_driver, current_page_url, wait_for_table_selector)
        if html_source:
            df_page_transfer_data = extract_data_using_confirmed_selectors(html_source, current_page_url)
            if df_page_transfer_data:
                all_scraped_data_dfs.extend(df_page_transfer_data)
                print(f"  Trang {page_idx}: Trích xuất được {len(df_page_transfer_data)} mục.")
            else:
                print(f"  Trang {page_idx}: Không trích xuất được dữ liệu nào từ HTML.")
        else:
            print(f"  Trang {page_idx}: Không lấy được HTML source.")
        time.sleep(1)
except Exception as e:
    print(f"LỖI nghiêm trọng đã xảy ra trong quá trình cào dữ liệu: {e}")
finally:
    if active_driver:
        active_driver.quit()
```

- Mục đích: Thu thập dữ liệu từ 22 trang của FootballTransfers.com.

- Chi tiết:

- Khởi tạo WebDriver (get_driver()).
- Duyệt qua danh sách URL (URLS_TO_SCRAPE):
  - Tải HTML bằng get_page_source_with_selenium_and_wait, chờ bảng table.mvp-table.
  - Trích xuất dữ liệu bằng extract_data_using_confirmed_selectors.
  - Thêm dữ liệu vào all_scraped_data_dfs nếu có.
  - Tạm dừng 1 giây để tránh bị chặn.

Bước 4: Xử lý và lọc dữ liệu

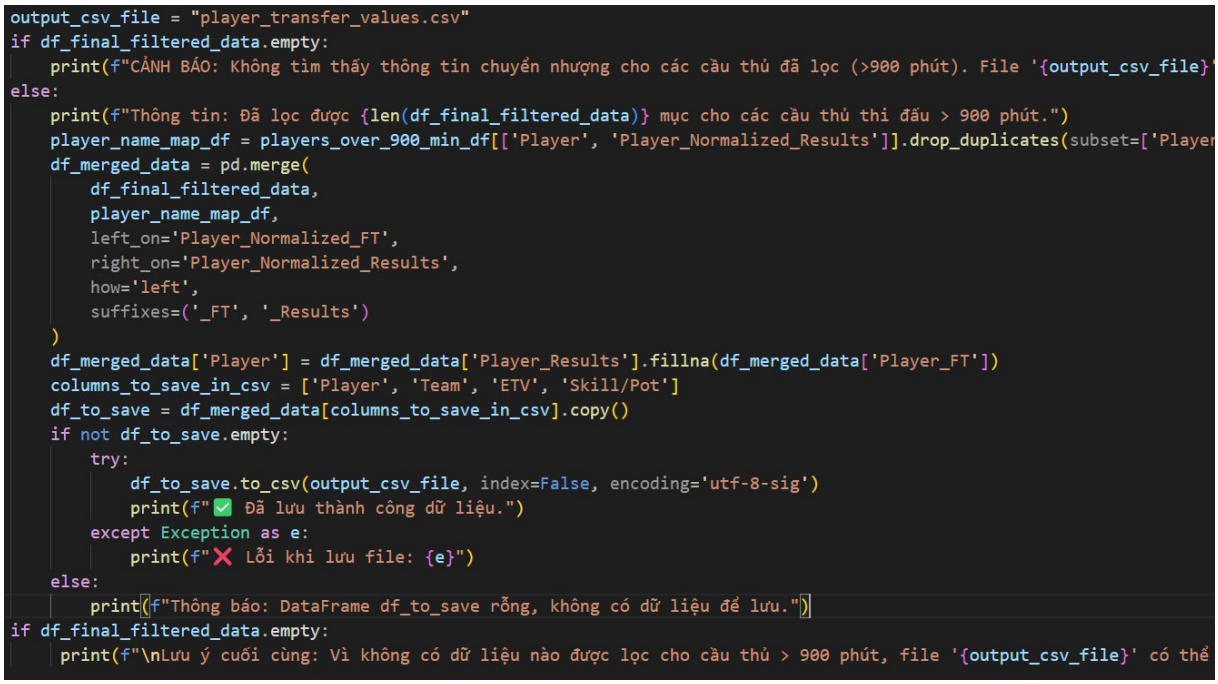
```
if not all_scraped_data_dfs:
    print("Thông báo: Không thu thập được dữ liệu nào từ footballtransfers.com. Kết thúc.")
    exit()
df_all_transfers_raw = pd.DataFrame(all_scraped_data_dfs)
if df_all_transfers_raw.empty:
    print("Thông báo: DataFrame thô rỗng sau khi thu thập. Kết thúc.")
    exit()
df_all_transfers_raw.drop_duplicates(subset=['Player'], keep='first', inplace=True)
if df_all_transfers_raw.empty:
    print("Thông báo: Không có dữ liệu thô nào sau khi xử lý trùng lặp. Kết thúc.")
    exit()
df_all_transfers_raw['Player_Normalized_FT'] = df_all_transfers_raw['Player'].apply(normalize_player_name)
df_final_filtered_data = df_all_transfers_raw[
    df_all_transfers_raw['Player_Normalized_FT'].isin(set_players_over_900_min_normalized)
].copy()
```

- Mục đích: Tạo DataFrame từ dữ liệu thu thập, loại bỏ trùng lặp, và lọc theo danh sách cầu thủ >900 phút.

- Chi tiết:

- Thoát nếu không thu thập được dữ liệu.
- Tạo df_all_transfers_raw từ all_scraped_data_dfs.
- Loại bỏ hàng trùng lặp dựa trên cột Player.
- Chuẩn hóa tên cầu thủ trong df_all_transfers_raw
- Lọc các hàng có Player_Normalized_FT nằm trong set_players_over_900_min_normalized.

Bước 5: Gộp dữ liệu và lưu kết quả



```
output_csv_file = "player_transfer_values.csv"
if df_final_filtered_data.empty:
    print(f"CẢNH BÁO: Không tìm thấy thông tin chuyển nhượng cho các cầu thủ đã lọc (>900 phút). File '{output_csv_file}'
else:
    print(f"Thông tin: Đã lọc được {len(df_final_filtered_data)} mục cho các cầu thủ thi đấu > 900 phút.")
    player_name_map_df = players_over_900_min_df[['Player', 'Player_Normalized_Results']].drop_duplicates(subset=['Player
    df_merged_data = pd.merge(
        df_final_filtered_data,
        player_name_map_df,
        left_on='Player_Normalized_FT',
        right_on='Player_Normalized_Results',
        how='left',
        suffixes=('_FT', '_Results')
    )
    df_merged_data['Player'] = df_merged_data['Player_Results'].fillna(df_merged_data['Player_FT'])
    columns_to_save_in_csv = ['Player', 'Team', 'ETV', 'Skill/Pot']
    df_to_save = df_merged_data[columns_to_save_in_csv].copy()
    if not df_to_save.empty:
        try:
            df_to_save.to_csv(output_csv_file, index=False, encoding='utf-8-sig')
            print(f"✅ Đã lưu thành công dữ liệu.")
        except Exception as e:
            print(f"❌ Lỗi khi lưu file: {e}")
    else:
        print(f"Thông báo: DataFrame df_to_save rỗng, không có dữ liệu để lưu.")
if df_final_filtered_data.empty:
    print(f"\nLưu ý cuối cùng: Vì không có dữ liệu nào được lọc cho cầu thủ > 900 phút, file '{output_csv_file}' có thể
```

- Mục đích: Gộp dữ liệu, ưu tiên tên cầu thủ từ results.csv, và lưu vào player_transfer_values.csv.

- Chi tiết:

- Nếu df_final_filtered_data rỗng, in cảnh báo.
- Tạo player_name_map_df để ánh xạ tên chuẩn hóa và tên gốc từ results.csv.
- Gộp df_final_filtered_data với player_name_map_df bằng pd.merge (left join).
- Tạo cột Player ưu tiên tên từ results.csv (Player_Results), nếu thiếu thì dùng tên từ FootballTransfers.com (Player_FT).
- Chọn các cột yêu cầu (Player, Team, ETV, Skill/Pot) và lưu vào player_transfer_values.csv.