HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN

-----ꕥ🕮ꕥ-----

BÁO CÁO BÀI TẬP LỚN

Lập trình Python

Sinh viên thực hiện:

Họ và tên : Lương Minh Quý

Mã SV: B22DCKH099

Hà Nội, ngày 3/11/2023

- Câu 1: Thu thập dữ liệu thống kê [*] của tất cả các cầu thủ có số phút thi đấu nhiều hơn 90 phút tại giải bóng đá ngoại hạng Anh mùa 2023-2024.
  - Ý tưởng :
    - Sử dụng thư viện BeautifulSoup và requests để lấy dữ liệu từ trên trang web về
    - Cách cài đặt : dung lệnh “pip install BeautifulSoup requests” để cài đặt hai thư viện
    - Ta chia những chỉ số thành nhiều phần trong cùng 1 table để dễ dàng sử lý
    - Ta có thể chia các chỉ số thành 10 phần sau:
      + Base_data
      + Goalkeep_data
      + Shooting_data
      + Passing_data
      + Goalshot_data
      + Defensive_data
      + Posess_data
      + Playtime_data
      + Miscell_data
  - Đầu tiên , ta thu thập dữ liệu về các đội bóng tham gia ngoại hạng Anh bằng 2 thue viện nói trên (HÌnh 1.1)

```
url='https://fbref.com/en/comps/9/2023-2024/2023-2024-Premier-League-Stats'
r=requests.get(url)
soup=BeautifulSoup(r.content,'html.parser')
table=soup.find('table',{
    'class': 'stats_table sortable min_width force_mobilize',
    'id': 'results2023-202491_overall'
})
#Danh sách chứa các đội bóng và url đến đội bóng đó
teams_data=[]
#Tìm các thẻ <a> trong <tbody> của table
tbody = table.find('tbody')
teams = tbody.find_all('a', href=True)
#Đưa dữ liệu về tên và link đội bóng vào danh sách
for team in teams :
    if "squads" in team['href']:
        team_name=team.text.strip()
        team_url="https://fbref.com" + team['href']
        teams_data.append([team_name,team_url])
```

(Hình 1.1)

- Tiếp theo tạo 10 danh sách ứng vs 10 phần dữ liệu đã được chia r ava thu thập dữ liệu (Hình 1.2)

```
for team in teams_data:
    print(f"Đang cào dữ liệu đội {team[0]}...")
    r = requests.get(team[1])
    soup = BeautifulSoup(r.content, 'html.parser')

    crawl_base_data(soup, team[0], base_data, ok)
    crawl_goalkeep_data(soup, team[0], goalkeep_data, ok)
    crawl_shooting_data(soup, team[0], shooting_data, ok)
    crawl_passing_data(soup, team[0], passing_data, ok)
    crawl_passtype_data(soup, team[0], passtype_data, ok)
    crawl_goalshot_data(soup, team[0], goalshot_data, ok)
    crawl_defensive_data(soup, team[0], defensive_data, ok)
    crawl_possess_data(soup, team[0], possess_data, ok)
    crawl_playtime_data(soup, team[0], playtime_data, ok)
    crawl_miscell_data(soup, team[0], miscell_data, ok)

    print(f'Đã cào xong đội {team[0]}...')
    ok += 1
    time.sleep(4)
```

(Hình 1.2)

- Từ danh sách các đội tìm được , tiến hành lấy thông tin từ các đội ở từng bảng và đưa vào danh sách , mỗi data sẽ tương tứng vs 1 hàm riêng , minh họa hàm tìm base_data (HÌnh 1.3)

```
def crawl_base_data(soup, team_name, base_data, ok):
    """Hàm tìm dữ liệu base_data của một đội bóng."""
    table = soup.find('table', {'class': 'stats_table sortable min_width', 'id': 'stats_standard_9'})
    if ok == 1:
        b = ["Team"]
        thead = table.find('thead').find_all('tr')
        for x in thead[1]:
            if x.text.strip() != "":
                b.append(x.text.strip())
        base_data.append(b)
    tbody = table.find('tbody').find_all('tr')
    for y in tbody:
        tmp = [team_name]
        for x in y:
            tmp.append(x.text.strip() if x.text.strip() else "N/a")
        base_data.append(tmp)
```

(Hình 1.3)

- Sau khi tìm được tất cả các dữ liệu cần thiết , t chuyển tất cả các list tìm được thành dataframe và đưa hết vào 1 list chứa , minh họa cho vc chuyển list base_data thành df và đưa vào list chưa, những hàm khác tương tự (HÌnh 1.4)

```
#Tạo 1 list chứa tất cả các df
all_df=[]
#Chuyển đổi list thành Dataframe và loại bộ những hàng không cần thiết
base_data_df=pd.DataFrame(base_data[1:],columns=base_data[0])
base_data_df=base_data_df.drop(['90s','Gls','G+A','PK','npxG+xAG','Matches'],axis=1)
all_df.append(base_data_df.drop_duplicates(subset='Player'))
```

(Hinh 1.4)

- Cuối cùng , ta lấy tổng hợp tất cả các df bằng outer join đòng thời loại bỏ trùng lặp và them điều kiện số phút thi đấu phải lớn hơn 90 (Hình 1.5)

```
#Outer join hết tất cả các df trong list all_df
result = reduce(lambda left, right: pd.merge(left, right, on=['Player'], how='outer'),all_df)
#Loại bỏ trùng lặp
result=result.drop_duplicates()
result = result.groupby('Player').first().reset_index()
result=result.T.drop_duplicates().T
#Loại các cầu thủ có chỉ số Min bé hơn 90 hoặc N/a
def ch(a:str):
    if a=='N/a' :
        return 0
    return int(a.replace(',',''))
a=list(result['Min'].values)
b=[x for x in a if ch(x)>90]
result=result[result['Min'].isin(b)].reset_index(drop=True)
print(result)
#Ghi thông tin vào file csv
# result.to_csv("results.csv",index=False)
```

(Hình 1.5)

- Câu 2 : Từ những thông tin từ file results.cvs , thực hiện các yêu cầu

- Tìm top 3 cầu thủ có điểm cao nhất và thấp nhất ở mỗi chỉ số (Hình 2.1)

  + Ý tưởng :

  _ Đầu tiên ta lọc ra các column có kiểu số

  _ TIếp đó tìm ra 3 cầu thủ có chỉ số cao nhất và thấp nhất ứng với tùng column và đưa vào list_column

  _tạo 1 dataframe được tạo nên từ các list column

```python
def find_top_bottom_players(df):
    numeric_cols = df.select_dtypes(include=['number']).columns
    result_top = pd.DataFrame()
    for col in numeric_cols:
        top_3 = df.nlargest(3, col)['Name'].values
        bottom_3 = df.nsmallest(3, col)['Name'].values
        result_top[col] = np.concatenate((top_3, bottom_3))
    print("Top 3 và Bottom 3 cầu thủ cho từng chỉ số:")
    print(result_top)
```

(Hình 2.1)

+ Kết quả sau khi chạy code :

```
top3 và bottom 3 cầu thủ của từng chỉ số
                 Age                 MP              Starts               Goals             Assists1  ...                 OG                Recov                 Won                 Lost                Won%
0        Mikey Moore       Jonny Castro         Zack Nelson  Aleksandar Mitrović           Alex Matos  ...     Đorđe Petrović       Andrey Santos      Aaron Ramsdale          Zack Nelson       Anass Zaroury
1      Ethan Nwaneri  Joshua Acheampong       Andrey Santos           Alex Iwobi           Alex Iwobi  ...       Aaron Hickey   Antwoine Hackford    Thomas Strakosha           Alex Iwobi     Andros Townsend
2       Leon Chiwome      Amadou Diallo   Antwoine Hackford           Alex Matos  Aleksandar Mitrović  ...     Aaron Ramsdale       Mason Burstow   Antwoine Hackford        Stefan Ortega   Antwoine Hackford
3   Yunus Emre Konak   Yunus Emre Konak    Yunus Emre Konak     Yunus Emre Konak     Yunus Emre Konak  ...   Yunus Emre Konak    Yunus Emre Konak    Yunus Emre Konak     Yunus Emre Konak         Zack Nelson
4         Zak Sturge         Zak Sturge          Zak Sturge           Zak Sturge           Zak Sturge  ...         Zak Sturge          Zak Sturge          Zak Sturge           Zak Sturge          Zak Sturge
5    Álvaro Carreras    Álvaro Carreras     Álvaro Carreras      Álvaro Carreras      Álvaro Carreras  ...    Álvaro Carreras     Álvaro Carreras     Álvaro Carreras      Álvaro Carreras     Álvaro Carreras

[6 rows x 154 columns]
```

- Tìm trung vị của mỗi chỉ số. Tìm trung bình và độ lệch chuẩn của mỗi chỉ số cho các cầu thủ trong toàn giải và của mỗi đội

  + Ý tưởng :

  _ Đối với toàn bộ các cầu thủ thì ta có 3 bảng tìm về median,mean,std đối với mỗi chỉ số, sau khi tìm được thì t chỉ cần nối column của chúng lại tạo thành 1 dataframe1 mới (Hình 2.2)

```python
numeric_cols = df.select_dtypes(include=['number']).columns
# Thống kê cho toàn giải
overall_stats = {
    'Team': ['All'],
    **{f'{col}_median': [df[col].median()] for col in numeric_cols},
    **{f'{col}_mean': [df[col].mean()] for col in numeric_cols},
    **{f'{col}_std': [df[col].std()] for col in numeric_cols},
}
overall_df = pd.DataFrame(overall_stats)
```

(Hình 2.2)

  _ Còn đối vs đội cũng vậy , ta cũng có 3 bảng về median,mean,std đối vs mỗi chỉ số được group theo "team" , từ đó ra nối column chúng lại tạo thành 1 dataframe2 mới (Hình 2.3)

```python
# Thống kê cho từng đội
team_stats = df.groupby('Team')[numeric_cols].agg(['median', 'mean', 'std']).round(2)
team_stats.columns = [f'{col}_{stat}' for col, stat in team_stats.columns]
team_stats.reset_index(inplace=True)
```

(Hình 2.3)

_Cuối cùng ta nối df1 và df2 theo hang bằng hàm concat và them dữ liệu vào file result2.csv (Hình 2.4)

```
# Kết hợp và lưu kết quả
final_df = pd.concat([overall_df, team_stats], ignore_index=True)
# final_df.to_csv('results2.csv', index=False)
print(final_df)
```

(Hình 2.4)

- Vẽ historgram phân bố của mỗi chỉ số của các cầu thủ trong toàn giải và mỗi đội (Hình 2.5)
  + Dầu tiên ta vẽ biểu đồ cho toàn giải (Hình 2.5) , sau đó ta tạo them thư mục để lưu trữ từng biểu đồ một (Hình 2.5)

```
def generate_histograms(df):
    def save_histograms(data, folder, title_prefix=""):
        os.makedirs(folder, exist_ok=True)
        for col in data.select_dtypes(include=['number']).columns:
            plt.figure(figsize=(8, 6))
            sns.histplot(data[col], bins=20, kde=True)
            plt.title(f'{title_prefix} {col}')
            plt.xlabel(col)
            plt.ylabel('Số lượng cầu thủ')
            plt.grid(True, linestyle='--', alpha=0.5)
            plt.savefig(os.path.join(folder, f"{col}.png"))
            plt.close()

    save_histograms(df, 'histograms_all', 'Toàn giải -')
    for team in df['Team'].unique():
        save_histograms(df[df['Team'] == team], f"histograms_teams/{team}", f"Đội {team} -")
    print("Đã lưu tất cả biểu đồ histogram.")
```

(Hình 2.5)

- Tìm các đội bóng có chỉ số cao nhất ở mỗi chỉ số
  + Ý tưởng t duyêt qua từng chỉ số của df1 (được tạo ra từ vc grouby ”team” với phương thức mean) và lưu lại vào trong list tên đội , chỉ số, giá trị max của chỉ số đó

```
def identify_best_team(df):
    numeric_cols = df.select_dtypes(include=['number']).columns
    team_means = df.groupby('Team')[numeric_cols].mean()
    best_teams = pd.DataFrame([(col, team_means[col].idxmax(), team_means[col].max()) for col in numeric_cols], columns=['Chỉ số', 'Đội', 'Giá trị'])
    print("Đội có điểm cao nhất cho từng chỉ số:")
    print(best_teams)

    # Đếm số lần xuất hiện của mỗi đội và tìm đội xuất hiện nhiều nhất
    top_team = Counter(best_teams['Đội']).most_common(1)
    print(f"Đội có điểm cao nhất tổng hợp: {top_team[0][0]} với {top_team[0][1]} chỉ số cao nhất.")
```

(Hình 2.6)

	Teams	Status	Value
0	Fulham	Age	27.16
1	Fulham	MP	23.52
2	Arsenal	Starts	16.72
3	Manchester City	Goals	3.76
4	Manchester City	Assists1	2.76
..	...	...	...
149	Sheffield Utd	OG	0.20
150	Everton	Recov	75.77
151	Everton	Won	26.00
152	Luton Town	Lost	22.54
153	Nott'ham Forest	Won%	56.16

(HÌnh 2.7 : bảng df_results)

- Từ hình 2.6 , ta thấy rằng việc tiếp theo là tìm tần xuất điểm của các đội bằng cách dung hàm couter để đếm tần xuất và sort để tìm giá trị lớn nhất

  + Kết quả của thu được sau khi chạy code : `Manchester City 61`

  ➔ Manchester City là đội có phong độ tốt nhất trong giải ngoại hạng Anh
- Trong hàm main , ta vt lại các hàm (Hinh 2.8)

```
# Thực thi các hàm
if __name__ == "__main__":
    df = pd.read_csv("results.csv")
    find_top_bottom_players(df)
    calculate_team_statistics(df)
    generate_histograms(df)
    identify_best_team(df)
```

(Hình 2.8)

- Câu 3:
- Sử dụng thuật toán K-means để phân loại các cầu thủ thành các nhóm có chỉ số giống nhau.
- Sử dụng thuật toán PCA, giảm số chiều dữ liệu xuống 2 chiều, vẽ hình phân cụm các điểm dữ liệu trên mặt 2D.

  + Hàm vẽ biểu đồ (Hinh 3.1)

```python
def plot_clusters(data_points, centroid_points, cluster_labels, num_clusters):
    plt.figure(figsize=(8, 6))

    # Màu sắc ngẫu nhiên cho các cụm
    colors = plt.cm.get_cmap('viridis', num_clusters)

    for cluster_index in range(num_clusters):
        # Lấy các điểm thuộc cụm cluster_index
        cluster_points = data_points[cluster_labels == cluster_index]
        plt.scatter(cluster_points[:, 0], cluster_points[:, 1], s=50, color=colors(cluster_index), label=f'Cụm {cluster_index}')
        # Vẽ tâm cụm
        plt.scatter(centroid_points[cluster_index, 0], centroid_points[cluster_index, 1], s=200, color=colors(cluster_index), marker='X', edgecolor='k')

    plt.title('K-means Phân cụm các cầu thủ bóng đá')
    plt.xlabel('PC1')
    plt.ylabel('PC2')
    plt.legend()
    plt.show()
```

(HÌnh 3.1.1)

+ Hàm chuẩn bị dữ liệu (HÌnh 3.1.2)

```python
def prepare_data(file_path):
    # Đọc dữ liệu và xử lý giá trị thiếu
    data_frame = pd.read_csv(file_path)
    data_frame = data_frame.select_dtypes(exclude=['object'])
    data_frame.fillna(data_frame.mean(), inplace=True)

    # Chuẩn hóa dữ liệu
    scaler = StandardScaler()
    standardized_data = scaler.fit_transform(data_frame)

    # Giảm chiều với PCA
    pca = PCA(n_components=2)
    reduced_data = pca.fit_transform(standardized_data)

    return pd.DataFrame(reduced_data, columns=['PC1', 'PC2'])
```

(Hình 3.1.2)

+ Hàm chạy K-means tùy chỉnh (HÌnh 3.1.3)

```python
def kmeans_custom(data_frame, num_clusters, max_iterations=100):
    # Khởi tạo tâm cụm ngẫu nhiên
    centroid_points = data_frame.sample(n=num_clusters).values
    cluster_labels = np.zeros(len(data_frame))

    for _ in range(max_iterations):
        # Gán nhãn cụm cho từng điểm
        for point_index in range(len(data_frame)):
            distances = np.linalg.norm(data_frame.values[point_index] - centroid_points, axis=1)
            cluster_labels[point_index] = np.argmin(distances)

        # Cập nhật tâm cụm mới
        new_centroid_points = np.array([data_frame.values[cluster_labels == cluster_index].mean(axis=0) for cluster_index in range(num_clusters)])

        # Kiểm tra sự hội tụ
        if np.all(centroid_points == new_centroid_points):
            break

        centroid_points = new_centroid_points

    # Vẽ biểu đồ kết quả cuối cùng
    plot_clusters(data_frame.values, centroid_points, cluster_labels, num_clusters)
```

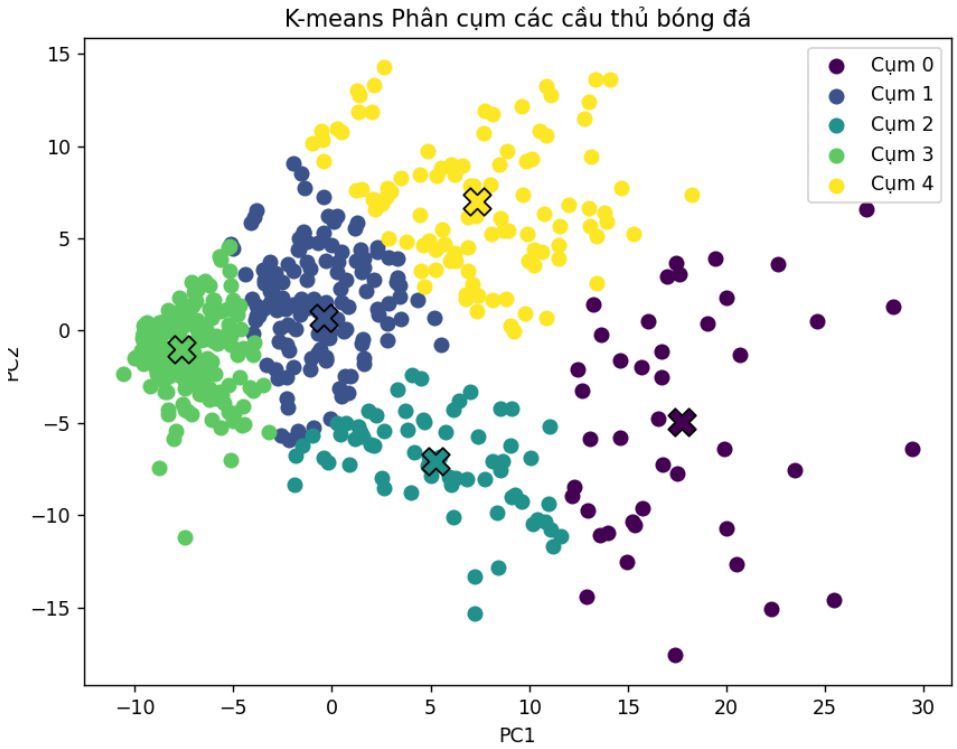
(Hình 3.1.4)

+ Trong hàm main (Hình 3.1.4)

```
# Chạy chương trình chính
if __name__ == "__main__":
    file_path = 'results.csv'
    prepared_data = prepare_data(file_path)
    num_clusters = 5  # Số cụm mong muốn
    kmeans_custom(prepared_data, num_clusters)
```

(Hình 3.1.4)

+ Sau khi chạy trương trình (Hình 3.1.5)



(Hình 3.1.5)

- Viết chương trình python vẽ biểu đồ rada (radar chart) so sánh cầu thủ :

  + hàm vẽ biểu đồ (Hình 3.2.1)

```python
def create_radar_chart(data_frame, player1_name, player2_name, attribute_list):
    # Lọc dữ liệu cho từng cầu thủ
    player1_data = data_frame[data_frame['Name'] == player1_name]
    player2_data = data_frame[data_frame['Name'] == player2_name]
    # Kiểm tra nếu không tìm thấy cầu thủ
    if player1_data.empty or player2_data.empty:
        print("Không tìm thấy cầu thủ hoặc sai tên.")
        return
    # Lấy giá trị các thuộc tính
    player1_values = player1_data[attribute_list].values.flatten()
    player2_values = player2_data[attribute_list].values.flatten()
    # Xây dựng góc của biểu đồ radar
    num_attributes = len(attribute_list)
    angles = np.linspace(0, 2 * np.pi, num_attributes, endpoint=False).tolist()
    # Hoàn thành vòng radar
    player1_values = np.concatenate((player1_values, [player1_values[0]]))
    player2_values = np.concatenate((player2_values, [player2_values[0]]))
    angles += angles[:1]
    # Vẽ biểu đồ
    fig, ax = plt.subplots(figsize=(6, 6), subplot_kw=dict(polar=True))
    ax.fill(angles, player1_values, color='blue', alpha=0.25)
    ax.fill(angles, player2_values, color='red', alpha=0.25)
    ax.plot(angles, player1_values, color='blue', linewidth=2, label=player1_name)
    ax.plot(angles, player2_values, color='red', linewidth=2, label=player2_name)
    # Cấu hình các nhãn thuộc tính
    ax.set_yticklabels([])
    ax.set_xticks(angles[:-1])
    ax.set_xticklabels(attribute_list)
    plt.title(f"So sánh chỉ số giữa {player1_name} và {player2_name}")
    plt.legend(loc='upper right')
    plt.show()
```

(Hình 3.2.1)

+ Hàm main (Hình 3.2.2)

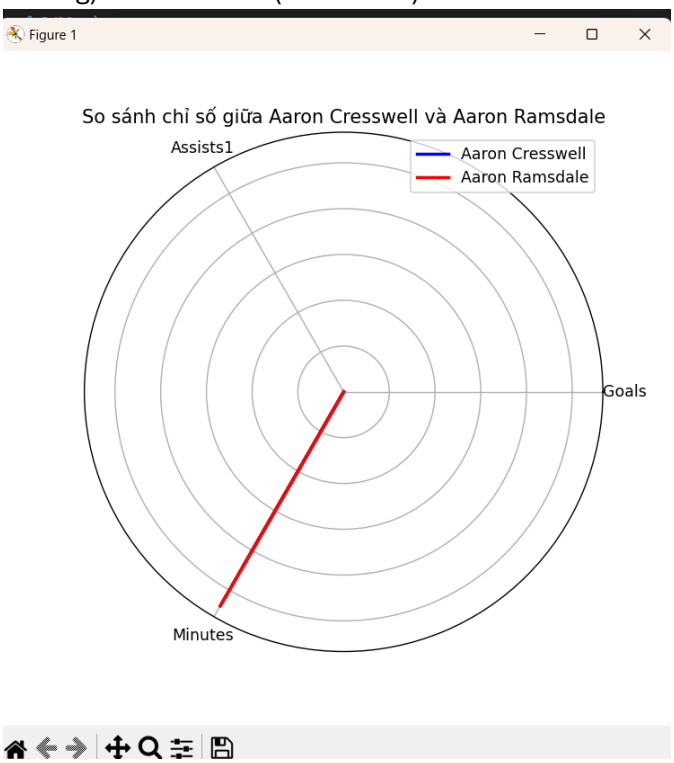
```python
if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Vẽ biểu đồ radar so sánh cầu thủ")
    parser.add_argument("--p1", type=str, required=True, help="Tên cầu thủ thứ nhất")
    parser.add_argument("--p2", type=str, required=True, help="Tên cầu thủ thứ hai")
    parser.add_argument("--Attribute", type=str, required=True, help="Danh sách các chỉ số cần so sánh, cách nhau bởi dấu phẩy")
    args = parser.parse_args()

    # Chuyển đổi chuỗi các thuộc tính thành danh sách
    attribute_list = [attribute.strip() for attribute in args.Attribute.split(",")]

    # Đọc dữ liệu và vẽ biểu đồ
    player_data = load_csv_data("results.csv")
    create_radar_chart(player_data, args.p1, args.p2, attribute_list)

    # Cú pháp chạy chương trình
    # C:/Users/luong/AppData/Local/Programs/Python/Python312/python.exe Bai3,2.py --p1 "Aaron Cresswell" --p2 "Aaron Ramsdale" --Attribute "Goals, Assists1, Minutes"
```

(Hình 3.2.2)

➔ Sau khi chạy lệnh ta dung lệnh <Đường dẫn file> <tên file> --p1 "Tên Cầu Thủ 1" --p2 "Tên Cầu Thủ 2" --Attribute "ChỉSố1,ChỉSố2,...,ChỉSốN"

_ Cụ thể sau khi dung lệnh :

C:/Users/luong/AppData/Local/Programs/Python/Python312/python.exe Bai3,2.py --p1 "Aaron Cresswell" --p2 "Aaron Ramsdale" --Attribute "Goals, Assists1, Minutes" (đã được bôi xanh dòng cuối cùng) trên terminal (Hình 3.3.3)



(Hình 3.3.3)

- Bài 4 : Thu thập giá chuyển nhượng của các cầu thủ trong mùa 2023-2024 từ trang web

- Ý tưởng :
  - Cũng như bài 1 , ta dung hai thự viện requests và BeautifulSoup để thu thập và làm sạch dữ liệu trên web
  - Đầu tiên ta lấy dữ liệu của các team
  - TIếp theo , từ từng team ta lấy dữ liệu của các cầu thủ trong team đấy (GIá chuyển nhượng)
- **fetch_teams(url)** (Hình 4.1)
  + Lấy URL của giải đấu và tải nội dung HTML về.
  + Tìm bảng chứa danh sách các đội và lấy tên đội cùng liên kết đến trang chi tiết của mỗi đội.

```python
def fetch_teams(url):
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    teams_data = []

    # Tìm bảng các đội trong trang và lấy liên kết đến các trang đội
    league_table = soup.find('table',
        {'class': 'table table-striped table-hover leaguetable mvp-table ranking-table mb-0'}
    )
    if league_table:
        team_rows = league_table.find('tbody')
        teams = team_rows.find_all('a', href=True)
        for team in teams:
            team_name = team.text.strip()
            team_url = team['href']
            teams_data.append((team_name, team_url))

    return teams_data
```

(Hình 4.1)

- **fetch_players(team_name, team_url)** (HÌnh 4.2)

  + Với mỗi đội, hàm này tải về trang chi tiết của đội đó, tìm bảng chứa danh sách cầu thủ và giá chuyển nhượng.

  + Tạo một danh sách các cầu thủ, trong đó mỗi phần tử chứa tên cầu thủ, tên đội, và chi phí chuyển nhượng.

```python
def fetch_players(team_name, team_url):
    player_data = []
    response = requests.get(team_url)
    soup = BeautifulSoup(response.text, 'html.parser')

    # Tìm bảng thông tin cầu thủ
    player_table = soup.find('table', {'class': 'table table-striped-rowspan ft-table mb-0'})
    if player_table:
        player_rows = player_table.find('tbody')
        players = player_rows.find_all('tr')

        for player in players:
            if "odd" in player.get('class', []) or "even" in player.get('class', []):
                player_name = player.find('th', class_='td-player').find('span').text.strip()
                transfer_cost = player.find_all('td')[-1].text.strip()
                player_data.append((player_name, team_name, transfer_cost))

    return player_data
```

(Hình 4.2)

- **main()** (Hình 4.3)

  + Gọi fetch_teams() để lấy danh sách đội bóng và các URL tương ứng.

  + Duyệt qua từng đội và gọi fetch_players() để lấy dữ liệu về cầu thủ.

  + Kết hợp tất cả dữ liệu vào all_players_data và tạo DataFrame.

  + In DataFrame ra màn hình hoặc lưu vào file CSV nếu cần.

```python
def main():
    url = 'https://www.footballtransfers.com/us/leagues-cups/national/uk/premier-league/2023-2024'
    teams = fetch_teams(url)
    all_players_data = []

    for team_name, team_url in teams:
        players = fetch_players(team_name, team_url)
        all_players_data.extend(players)
        print(f"Hoàn thành đội: {team_name}")
        time.sleep(3)  # Dừng 3 giây để tránh bị chặn truy cập

    # Tạo DataFrame từ dữ liệu cầu thủ thu thập được
    df = pd.DataFrame(all_players_data, columns=['Player', 'Team', 'Cost'])
    # df.to_csv("results4.csv", index=False, encoding='utf-8-sig')  # Lưu dữ liệu ra file CSV nếu cần
    print(df)
```

(Hình 4.3)