

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN I



BÁO CÁO BÀI TẬP LỚN

Bài Tập Lập Trình Với Python

Giảng viên bộ môn : GV. Kim Ngọc Bách

Mã sinh viên : B22DCCN589

Họ và tên : Hoàng Cao Nguyên

Lớp tín chỉ : Nhóm 11

Học phần : Lập trình với Python

PHẦN I: (4 điểm)

Yêu cầu: Viết chương trình Python thu thập dữ liệu phân tích cầu thủ có thời gian thi đấu lớn hơn 90p

(*) **Code:** ex1/main.py => **kết quả:** ex1/results.csv

(*) **Các bước giải quyết bài toán:**

- Tạo mảng 2 chiều để lưu dữ liệu và định nghĩa hàm kiểm tra chỉ số null

```
8     result = []
9
10    # Hàm kiểm tra null
11    def check(v):
12        if v is None or str(v).strip() == '':
13            return 'N/a'
14        return v
```

- Hàm **get_data_playing_time()**:

+ Gửi request đến đường dẫn chứa bảng có chỉ số về thời gian thi đấu của các cầu thủ. Từ đó lấy được dữ liệu dạng html.

```
#Lấy các cầu thủ thi đấu hơn 90p
def get_data_playing_time():
    url = 'https://fbref.com/en/comps/9/2023-2024/playingtime/2023-2024-Premier-League-Stats'
    response = requests.get(url)
```

+ Dựa vào cấu trúc file html, chúng ta lấy được table chứa các chỉ số

```
if response.status_code == 200:
    soup = BeautifulSoup(response.content, 'html.parser')
    table_container = soup.find('div', attrs={'id': 'all_stats_playing_time'})
    #xử lý dữ liệu trong comment
    s = str(table_container).find('<table>')
    e = str(table_container).find('</table>')
    table_str = str(table_container)[s:e] + '</table>'
    table = BeautifulSoup(table_str, 'html.parser')
```

+ Lấy tất cả các dòng của table, thêm các cột chỉ số của bảng Playing Time vào result

```
#Lấy tất cả các dòng
rows = table.findAll('tr')
#map tên các cột để lấy vị trí (trong dòng thứ 2)
column_names = {header.get_text(): index for index, header in enumerate(rows[1].findAll('th'))}
#khởi tạo cột cho mảng kết quả
result.append(['Player', 'Nation', 'Squad', 'Pos', 'Age', 'MP', 'Starts', 'Min', 'Mn/Start',
'Compl', 'Subs', 'Mn/Sub', 'unSub', 'PPM', 'onG', 'onGA', 'onxG', 'onxGA'])
```

+ Lấy dữ liệu từ dòng thứ 2 trong table, bỏ qua các cầu thủ có thời gian chơi nhỏ hơn 90p.

```
for i in range(2, len(rows)):
    #lấy cột đầu tiên trong th
    row_data = [rows[i].find('th').get_text()]
    #từ cột thứ 2 lấy trong td
    row_data += [col.get_text() for col in rows[i].findAll('td')]

    #bỏ qua hàng tiêu đề
    if len(row_data) == 1:
        continue

    # kiểm tra phút thi đấu:
    min = str(row_data[column_names['Min']]).replace(',', '')
    if min.strip() == '' or int(min) <= 90:
        continue
```

+) Thêm dòng dữ liệu hợp lệ vào result:

```
#dòng kết quả phân tích 1 cầu thủ
row_result = []
row_result.append(check(row_data[column_names['Player']]))
row_result.append(check(row_data[column_names['Nation']]))
row_result.append(check(row_data[column_names['Squad']]))
row_result.append(check(row_data[column_names['Pos']]))
row_result.append(check(row_data[column_names['Age']]))
row_result.append(check(row_data[column_names['MP']]))
row_result.append(check(row_data[column_names['Starts']]))
row_result.append(check(min))
row_result.append(check(row_data[column_names['Mn/Start']]))
row_result.append(check(row_data[column_names['Compl']]))
row_result.append(check(row_data[column_names['Subs']]))
row_result.append(check(row_data[column_names['Mn/Sub']]))
row_result.append(check(row_data[column_names['unSub']]))
row_result.append(check(row_data[column_names['PPM']]))
row_result.append(check(row_data[column_names['onG']]))
row_result.append(check(row_data[column_names['onGA']]))
row_result.append(check(row_data[column_names['onxG']]))
row_result.append(check(row_data[column_names['onxGA']]))

#thêm dòng kết quả
result.append(row_result)

print(f'SUCCESS: Đã khởi tạo {len(result) - 1} cầu thủ thỏa mãn (>90p) từ bảng Playing Time')
else:
    print("ERROR:", "Lỗi request (playing time): " + str(response.status_code))
    sys.exit(1)
```

- Hàm `get_data_standard()`:

+ Tương tự như hàm `get_data_playing_time()`, gửi request đến đường dẫn chứa bảng Standard có chỉ số tiêu chuẩn của các cầu thủ. Sau đó phân tích lấy được table chỉ số dạng html:

```
#Lấy dữ liệu từ bảng tiêu chuẩn
def get_data_standard():
    url = 'https://fbref.com/en/comps/9/2023-2024/stats/2023-2024-Premier-League-Stats'
    response = requests.get(url)

    if response.status_code == 200:
        soup = BeautifulSoup(response.content, 'html.parser')
        table_container = soup.find('div', attrs={'id': 'all_stats_standard'})
        #xử lý dữ liệu trong comment
        s = str(table_container).find('<table>')
        e = str(table_container).find('</table>')
        table_str = str(table_container)[s:e] + '</table>'
        table = BeautifulSoup(table_str, 'html.parser')
```

+ Tiếp theo, thêm các cột chỉ số mới vào result:

```
#Lấy tất cả các dòng
rows = table.findAll('tr')
#map tên các cột để lấy vị trí (trong dòng thứ 2)
column_names = {}
for index, header in enumerate(rows[1].findAll('th')):
    name = header.get_text()
    if name not in column_names:
        column_names[name] = index

#Thêm cột cho mảng kết quả
result[0].extend(['G-PK', 'PK', 'Ast', 'CrY', 'CrR', 'xG', 'npxG', 'xAG', 'PrgC',
                  'PrgP', 'PrgR', 'Gls', 'Ast', 'G+A', 'G-PK', 'G+A-PK', 'xG', 'xAG', 'xG+xAG', 'npxG', 'npxG+xAG'])
new_col_count = 21
```

+ Tạo 1 map để lưu trữ dữ liệu chỉ số các cầu thủ theo key được tạo bởi tên và tên đội (do phải thêm các chỉ số phải đúng với cầu thủ tương ứng đã khởi tạo từ bảng Playing Time)

```

#map lưu các dòng dữ liệu lấy được
row_results = {}

#lấy dữ liệu
for i in range(2, len(rows)):
    #lấy cột đầu tiên trong th
    row_data = [rows[i].find('th').get_text()]
    #từ cột thứ 2 lấy trong td
    row_data += [col.get_text() for col in rows[i].findAll('td')]

    #bỏ qua hàng tiêu đề
    if len(row_data) == 1:
        continue

    #xác định dữ liệu thuộc về cầu thủ nào (bảng tên và tên đội)
    name = row_data[column_names['Player']]
    team = row_data[column_names['Squad']]
    key = f'{name}_{team}'

```

+ Lấy dữ liệu trong 1 dòng

```

#dòng kết quả phân tích 1 cầu thủ
row_result = []

row_result.append(check(row_data[column_names['G-PK']]))
row_result.append(check(row_data[column_names['PK']]))
row_result.append(check(row_data[column_names['Ast']]))
row_result.append(check(row_data[column_names['CrdY']]))
row_result.append(check(row_data[column_names['CrdR']]))
row_result.append(check(row_data[column_names['xG']]))
row_result.append(check(row_data[column_names['npxG']]))
row_result.append(check(row_data[column_names['xAG']]))
row_result.append(check(row_data[column_names['PrgC']]))
row_result.append(check(row_data[column_names['PrgP']]))
row_result.append(check(row_data[column_names['PrgR']]))

#bị trùng tên cột nên dùng chỉ số
row_result.append(check(row_data[26]))
row_result.append(check(row_data[27]))
row_result.append(check(row_data[28]))
row_result.append(check(row_data[29]))
row_result.append(check(row_data[30]))
row_result.append(check(row_data[31]))
row_result.append(check(row_data[32]))
row_result.append(check(row_data[33]))
row_result.append(check(row_data[34]))
row_result.append(check(row_data[35]))

```

+ Sau khi lấy được tất cả dữ liệu chỉ số của bảng, tiến hành duyệt theo từ dòng result và ghép các ô chỉ số trong map vào hàng tương ứng với key:

```
#thêm dòng kết quả vào map
if key not in row_results:
    row_results[key] = row_result
else:
    print(f"ERROR: Trùng dòng {i}, name={name}, team={team} (standard)")

#thêm dữ liệu vào result
count = 0
for i in range(1, len(result)):
    row = result[i]
    key = f'{row[0]}_{row[2]}'
    if key in row_results:
        count += 1
        row.extend(row_results[key])
    else:
        row.extend(['N/a'] * new_col_count)

#Log
print(f'SUCCESS: Thành công cập nhật {count} cầu thủ từ bảng Standard')
else:
    print("ERROR:", "Lỗi request (standard): " + str(response.status_code))
    sys.exit(1)
```

- Với các hàm còn lại:

- + **get_data_goalkeeping()** ~ tương ứng với bảng Goalkeeping
 - + **get_data_shooting()** ~ tương ứng với bảng Shooting
 - + **get_data_passing()** ~ tương ứng với bảng Passing
 - + **get_data_pass_types()** ~ tương ứng với bảng Pass Types
 - + **get_data_goal_and_shot_creation()** ~ tương ứng với bảng Goal and Shot Creation
 - + **get_data_defensive_actions()** ~ tương ứng với bảng Defensive Actions
 - + **get_data_possession()** ~ tương ứng với bảng Possession
 - + **get_data_miscellaneous_stats()** ~ tương ứng với bảng Miscellaneous Stats
- => **Các hàm này hoạt động tương tự như hàm *get_data_standard()***

- Hàm main để chạy chương trình và lưu kết quả vào file:

```
920 def main():
921     global result
922     get_data_playing_time()
923     get_data_standard()
924     get_data_goalkeeping()
925     get_data_shooting()
926     get_data_passing()
927     get_data_pass_types()
928     get_data_goal_and_shot_creation()
929     get_data_defensive_actions()
930     get_data_possession()
931     get_data_miscellaneous_stats()
932
933     # Sắp xếp
934     header = result[:1]
935     body = result[1:]
936     body = sorted(body, key=lambda x: (str(x[0]).split()[-1], x[4]))
937     print(f"SUCCESS: Đã sắp xếp")
938     result = header + body
939
940
941     # Mở file csv để ghi
942     with open('ex1/results.csv', mode='w', newline='', encoding='utf-8') as file:
943         writer = csv.writer(file)
944         writer.writerows(result)
945         print(f"SUCCESS: Dữ liệu đã ghi vào file: {file.name}")
946
947
948 if __name__ == "__main__":
949     main()
```

(*) Console

```
PS C:\Users\NGUYEN\Desktop\Python\BTL\source> & c:/Users/NGUYEN/Desktop/Python/BTL/sou
SUCCESS: Đã khởi tạo 493 cầu thủ thỏa mãn (>90p) từ bảng Playing Time
SUCCESS: Thành công cập nhật 493 cầu thủ từ bảng Standard
SUCCESS: Thành công cập nhật 37 thủ môn từ bảng Goalkeeping
SUCCESS: Thành công cập nhật 493 cầu thủ từ bảng Shooting
SUCCESS: Thành công cập nhật 493 cầu thủ từ bảng Passing
SUCCESS: Thành công cập nhật 493 cầu thủ từ bảng Pass Types
SUCCESS: Thành công cập nhật 493 cầu thủ từ bảng Goal and Shot Creation
SUCCESS: Thành công cập nhật 493 cầu thủ từ bảng Defensive Actions
SUCCESS: Thành công cập nhật 493 cầu thủ từ bảng Possession
SUCCESS: Thành công cập nhật 493 cầu thủ từ bảng Miscellaneous Stats
SUCCESS: Đã sắp xếp
SUCCESS: Dữ liệu đã ghi vào file: ex1/results.csv
PS C:\Users\NGUYEN\Desktop\Python\BTL\source>
```

PHẦN II: (2 điểm)

Yêu cầu 1: Tìm top 3 cầu thủ có điểm cao nhất và thấp nhất ở mỗi chỉ số.

(*) Code: ex2/main_1.py => kết quả: ex2/top3.pdf

(*) Các bước giải quyết bài toán:

- Từ file dữ liệu results.csv có được từ phần 1, tiến hành lấy dữ liệu chỉ số của từng cầu thủ, header chứa tên các chỉ số, body chứa giá trị các chỉ số:

```
def main():
    #Phần 1:
    with open('ex1/results.csv', mode='r', encoding='utf-8') as file:
        csv_reader = csv.reader(file)
        data = [row for row in csv_reader]
        header = data[0]
        body = data[1:]
```

- Với mỗi cột chỉ số cần lấy, ta lọc ra các cầu thủ có chỉ số này khác N/a rồi sắp xếp tìm ra top 3. Để dễ quan sát kết quả sẽ được sẽ lưu vào đối tượng PrettyTable:

```
# tạo table
table = PrettyTable()
table.field_names = ['Attribute', 'Top 3 max', 'Top 3 min']
cols = len(header)
for j in range(5, cols):
    valid = [row for row in body if row[j] != 'N/a']
    valid = sorted(valid, key=lambda x: float(x[j]))
    if len(valid) < 3:
        continue
    row = [
        header[j],
        f'{valid[-1][0]}({valid[-1][2]}),\n{valid[-2][0]}({valid[-2][2]}),\n{valid[-3][0]}({valid[-3][2]})',
        f'{valid[0][0]}({valid[0][2]}),\n{valid[1][0]}({valid[1][2]}),\n{valid[2][0]}({valid[2][2]})',
    ]
    table.add_row(row, divider=True)
```

- Sau khi hoàn thành, in table ra file pdf:

```
#print to PDF
pdf = fpdf.FPDF(format='A4')
pdf.add_page()

pdf.add_font("Courier", "", "fonts/CourierPrime-Regular.ttf", uni=True)
pdf.set_font("Courier", size=10)

table.align = 'l'
table.align['Attribute'] = 'c'
table.max_width['Attribute'] = 10
table.max_width['Top 3 max'] = 35
table.max_width['Top 3 min'] = 35
# print(table)
```



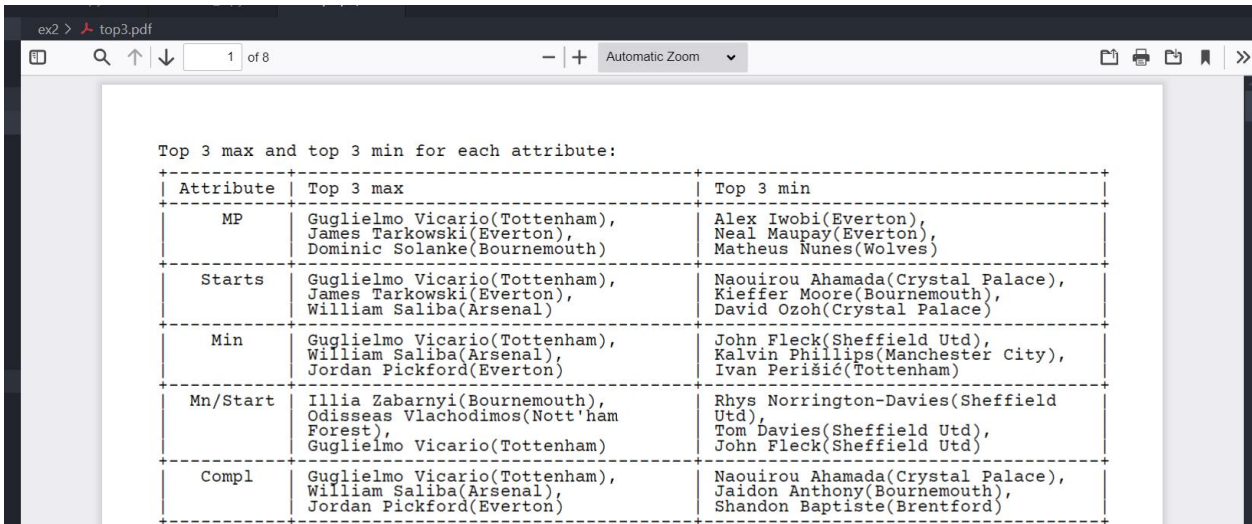
```
pdf.multi_cell(200, 6, txt="Top 3 max and top 3 min for each attribute:", align='c')
pdf.multi_cell(200, 3, txt=table.get_string())
pdf.output("ex2/top3.pdf")
print("SUCCESS: Kết quả top 3 đã được lưu vào file ex2/top3.pdf")

if __name__ == "__main__":
    main()
```

(*) Console

```
PS C:\Users\NGUYEN\Desktop\Python\BTL\source> & c:/Users/N
SUCCESS: Kết quả top 3 đã được lưu vào file ex2/top3.pdf
PS C:\Users\NGUYEN\Desktop\Python\BTL\source> █
```

(*) Minh họa kết quả:



Attribute	Top 3 max	Top 3 min
MP	Guglielmo Vicario(Tottenham), James Tarkowski(Everton), Dominic Solanke(Bournemouth)	Alex Iwobi(Everton), Neal Maupay(Everton), Matheus Nunes(Wolves)
Starts	Guglielmo Vicario(Tottenham), James Tarkowski(Everton), William Saliba(Arsenal)	Naouirou Ahamada(Crystal Palace), Kieffer Moore(Bournemouth), David Ozoh(Crystal Palace)
Min	Guglielmo Vicario(Tottenham), William Saliba(Arsenal), Jordan Pickford(Everton)	John Fleck(Sheffield Utd), Kalvin Phillips(Manchester City), Ivan Perišić(Tottenham)
Mn/Start	Illia Zabarnyi(Bournemouth), Odisseas Vlachodimos(Nott'ham Forest), Guglielmo Vicario(Tottenham)	Rhys Norrington-Davies(Sheffield Utd), Tom Davies(Sheffield Utd), John Fleck(Sheffield Utd)
Compl	Guglielmo Vicario(Tottenham), William Saliba(Arsenal), Jordan Pickford(Everton)	Naouirou Ahamada(Crystal Palace), Jaidon Anthony(Bournemouth), Shandon Baptiste(Brentford)

Yêu cầu 2: 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.

(*) Code: ex2/main_2.py => kết quả: ex2/results2.csv

(*) Các bước giải quyết:

- Định nghĩa hàm tính trung bình, trung vị và độ lệch chuẩn từ mảng 2 chiều dữ liệu chỉ số (data) của các cầu thủ (chỉ tính chỉ số từ MP trở đi). Trả về 1 danh sách kết quả sau khi tính:

```
#hàm tính median, mean và std từ danh sách cầu thủ
def calc(data):
    result = []
    for j in range(5, len(data[0])):
        row = []
        for i in range(len(data)):
            if data[i][j] != 'N/a':
                row.append(float(data[i][j]))
        result.append(str(round(numpy.median(row), 4)))
        result.append(str(round(numpy.mean(row), 4)))
        result.append(str(round(numpy.std(row), 4)))
    return result
```

- Lấy dữ liệu cầu thủ có được ở bài 1, tiến hành thêm cột theo tên chỉ số (header):

```
def main():
    #Lấy dữ liệu
    with open('ex1/results.csv', mode='r', encoding='utf-8') as file:
        csv_reader = csv.reader(file)
        players = [row for row in csv_reader]
        data = players[1:]

    # Chỉ tính các chỉ số từ cột MP trở đi
    attributes = players[0][5:]
    table = []
    table_row = []

    #tạo headers
    table_row = ["", ""]
    for attr in attributes:
        table_row.append('Median of ' + attr)
        table_row.append('Mean of ' + attr)
        table_row.append('Std of ' + attr)
    table.append(table_row)
```

- Dựa vào hàm tính toán đã xây dựng, tiến hành tính cho toàn bộ cầu thủ. Sau đó phân nhóm cầu thủ theo tên đội và tính cho từng đội

```
#tính toàn giải
table_row = [0, 'all']
table_row.extend(calc(data=data))
table.append(table_row)

#tính cho từng đội
squads = {} #chia các cầu thủ theo đội
for row in data:
    squad_name = row[2] #lấy tên cầu thủ
    if squad_name not in squads:
        squads[squad_name] = [row]
    else:
        squads[squad_name].append(row)

order = 1
for key, value in squads.items():
    table_row = [order, key]
    table_row.extend(calc(data=value))
    table.append(table_row)
    order += 1

#ghi kết quả ra file
with open('ex2/results2.csv', mode='w', newline='', encoding='utf-8') as file:
    writer = csv.writer(file)
    writer.writerows(table)
    print('SUCCESS: Tính toán thành công, kết quả lưu vào ' + file.name)

if __name__ == "__main__":
    main()
```

(*) Console

```
PS C:\Users\NGUYEN\Desktop\Python\BTL\source> & c:/Users/NGUYEN/Desk
SUCCESS: Tính toán thành công, kết quả lưu vào ex2/results2.csv
PS C:\Users\NGUYEN\Desktop\Python\BTL\source> █
```

Yêu cầu 3: Vẽ histogram phân bố của mỗi chỉ số của các cầu thủ trong toàn giải và mỗi đội.

(*) Code: ex2/main_3.py => kết quả: ex2/histograms

(*) Các bước giải quyết:

- Do cần in nhiều biểu đồ, chúng ta sẽ in chúng dưới dạng pdf
- Vẽ cho toàn bộ giải:

```
def main():
    matplotlib.use('Agg')
    df = pandas.read_csv('ex1/results.csv', na_values='N/a')

    # tính cho toàn giải
    with PdfPages("ex2/histograms/all.pdf") as pdf:
        for j in range(5, len(df.columns)):
            plt.figure(figsize=(8, 6))
            plt.hist(df.iloc[:, j].dropna(), bins=20, edgecolor='black', alpha=0.7)
            plt.title(df.columns[j])
            plt.xlabel(f'Giá trị')
            plt.ylabel('Tần xuất')
            plt.grid(True)
            pdf.savefig()
            plt.close()
    print("SUCCESS: Đã vẽ xong cho toàn giải")
```

- Vẽ cho từng đội:

```
# tính cho mỗi đội
squads = df.groupby('Squad')
for squad_name, squad_data in squads:
    with PdfPages(f'ex2/histograms/{squad_name}.pdf') as pdf:
        for j in range(5, len(squad_data.columns)):
            plt.figure(figsize=(8, 6))
            plt.hist(squad_data.iloc[:, j].dropna(), bins=20, edgecolor='black', alpha=0.7)
            plt.title(squad_data.columns[j])
            plt.xlabel(f'Giá trị')
            plt.ylabel('Tần xuất')
            plt.grid(True)
            pdf.savefig()
            plt.close()
    print(f'SUCCESS: Đã vẽ xong cho đội {squad_name}');

if __name__ == "__main__":
    main()
    print("SUCCESS: Hoàn thành! Các biểu đồ lưu tại ex2/histograms")
```

(*) Console

```
PS C:\Users\NGUYEN\Desktop\Python\BTL\source> & c:/Users/NGUYEN/Desktop/Python/BTL/source/ex2/main_4.py
SUCCESS: Đã vẽ xong cho toàn giải
SUCCESS: Đã vẽ xong cho đội Arsenal
SUCCESS: Đã vẽ xong cho đội Aston Villa
SUCCESS: Đã vẽ xong cho đội Bournemouth
SUCCESS: Đã vẽ xong cho đội Brentford
SUCCESS: Đã vẽ xong cho đội Brighton
SUCCESS: Đã vẽ xong cho đội Burnley
SUCCESS: Đã vẽ xong cho đội Chelsea
SUCCESS: Đã vẽ xong cho đội Crystal Palace
SUCCESS: Đã vẽ xong cho đội Everton
SUCCESS: Đã vẽ xong cho đội Fulham
SUCCESS: Đã vẽ xong cho đội Liverpool
SUCCESS: Đã vẽ xong cho đội Luton Town
SUCCESS: Đã vẽ xong cho đội Manchester City
SUCCESS: Đã vẽ xong cho đội Manchester Utd
SUCCESS: Đã vẽ xong cho đội Newcastle Utd
SUCCESS: Đã vẽ xong cho đội Nott'ham Forest
SUCCESS: Đã vẽ xong cho đội Sheffield Utd
SUCCESS: Đã vẽ xong cho đội Tottenham
SUCCESS: Đã vẽ xong cho đội West Ham
SUCCESS: Đã vẽ xong cho đội Wolves
SUCCESS: Hoàn thành! Các biểu đồ lưu tại ex2/histograms
PS C:\Users\NGUYEN\Desktop\Python\BTL\source> 
```

Yêu cầu 4: Tìm đội bóng có chỉ số điểm số cao nhất ở mỗi chỉ số, tìm đội có phong độ tốt nhất.

(*) Code: ex2/main_4.py => kết quả: ex2/top_squad.pdf

(*) Các bước giải quyết bài toán:

- Đọc dữ liệu từ ex1/results.csv, nhóm dữ liệu theo đội và tính trung bình. Tạo mảng lưu tên những đội đạt max ở các chỉ số tốt.

```
def main():
    df = pandas.read_csv('ex1/results.csv', na_values='N/a')

    target_cols = df.columns[5:]
    squads = df.groupby('Squad')[target_cols].mean()

    # Dữ liệu để vẽ biểu đồ phân bố với những chỉ số tốt
    positive_attrs_max_squads = []
```

- Dùng PrettyTable để lưu dữ liệu với 2 cột Attribute và tên đội. Tạo mảng lưu những chỉ số tiêu cực:

```
# chuẩn bị ghi kết quả ra pdf
table = PrettyTable()
table.field_names = ['Attribute', 'Top squad']
table.align = 'l'
pdf = fpdf.FPDF(format='A4')
pdf.add_page()
pdf.add_font("Courier", "", "fonts/CourierPrime-Regular.ttf", uni=True)
pdf.set_font("Courier", size=13)

# Những chỉ số xấu
negative_attrs = ['CrdY', 'CrdR', 'GA', 'GA90', 'Lost', 'Off', 'Mis', 'Err', 'Tkld%']
```

- Lặp qua các cột để tìm đội đạt max. Với những chỉ số tích cực, lưu lại tên đội để phân tích:

```
for col in target_cols:
    # Tìm đội có chỉ số cao nhất trong cột chỉ số
    squad = squads[col].idxmax()
    # Lưu những chỉ số tích cực
    if col not in negative_attrs:
        positive_attrs_max_squads.append(squad)
    # Cập nhật bảng kết quả
    table.add_row([col, squad])
```

- Lưu kết quả các đội đạt max các chỉ số ra file pdf. Sau đó tiến hành vẽ biểu đồ tần suất đạt max các chỉ số tích cực của các đội:

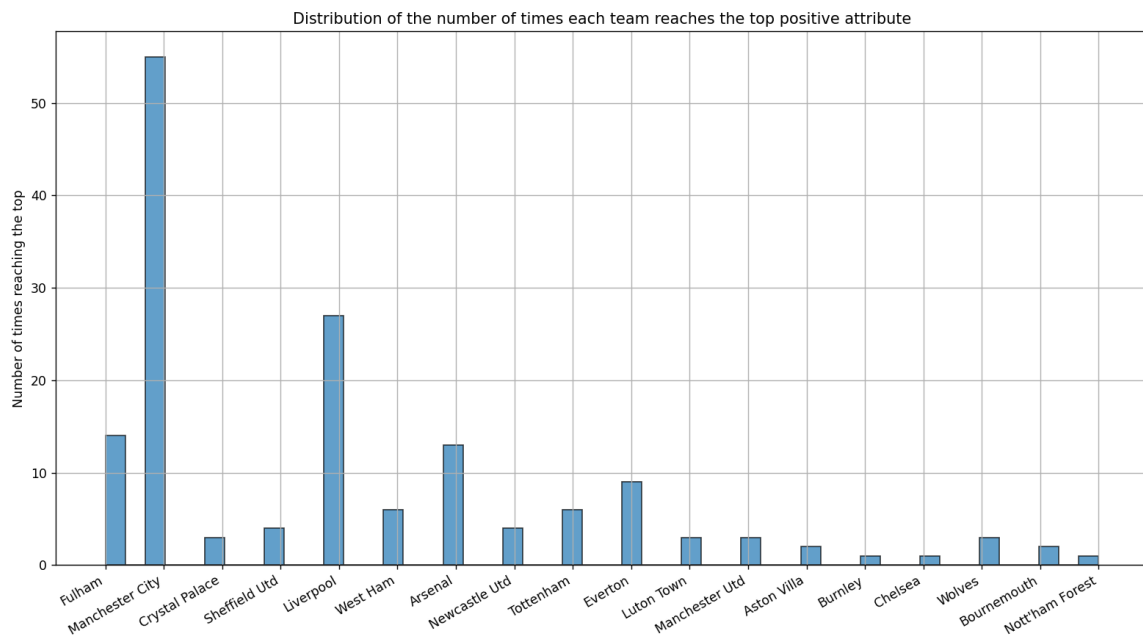
```
# ghi ra pdf bảng kết quả
pdf.multi_cell(200, 6, txt="Top squads for each attribute:")
pdf.multi_cell(200, 4, txt=table.get_string())
pdf.output("ex2/top_squads.pdf")
print("SUCCESS: Đã ghi kết quả ra ex2/top_squads.pdf")

#Phân tích số lần đạt top -> đội phong độ tốt nhất
plt.figure(figsize=(12, 6))
plt.hist(positive_attrs_max_squads, bins=50, edgecolor='black', alpha=0.7)
plt.title("Distribution of the number of times each team reaches the top positive attribute")
plt.xlabel(f'Squad name')
plt.ylabel('Number of times reaching the top')
plt.xticks(rotation=30, ha='right', fontsize=10)
plt.grid(True)
plt.show()

#kết luận
analyzed_result = pandas.Series(positive_attrs_max_squads).value_counts()
print(f'-> Đội phong độ tốt nhất: {analyzed_result.idxmax()} ({analyzed_result.max()} lần đạt top)')

if __name__ == "__main__":
    main()
```

- Kết quả sau khi vẽ biểu đồ tần số:



=> Như vậy, dễ dàng có thể thấy đội có phong độ tốt nhất là Manchester City

(*) Console:

```
PS C:\Users\NGUYEN\Desktop\Python\BTL\source> & c:/Users/NGUYEN/Desktop/Python/BTL/source/main.py
SUCCESS: Đã ghi kết quả ra ex2/top_squads.pdf
-> Đội phong độ tốt nhất: Manchester City (55 lần đạt top)
```

PHẦN III: 3 điểm

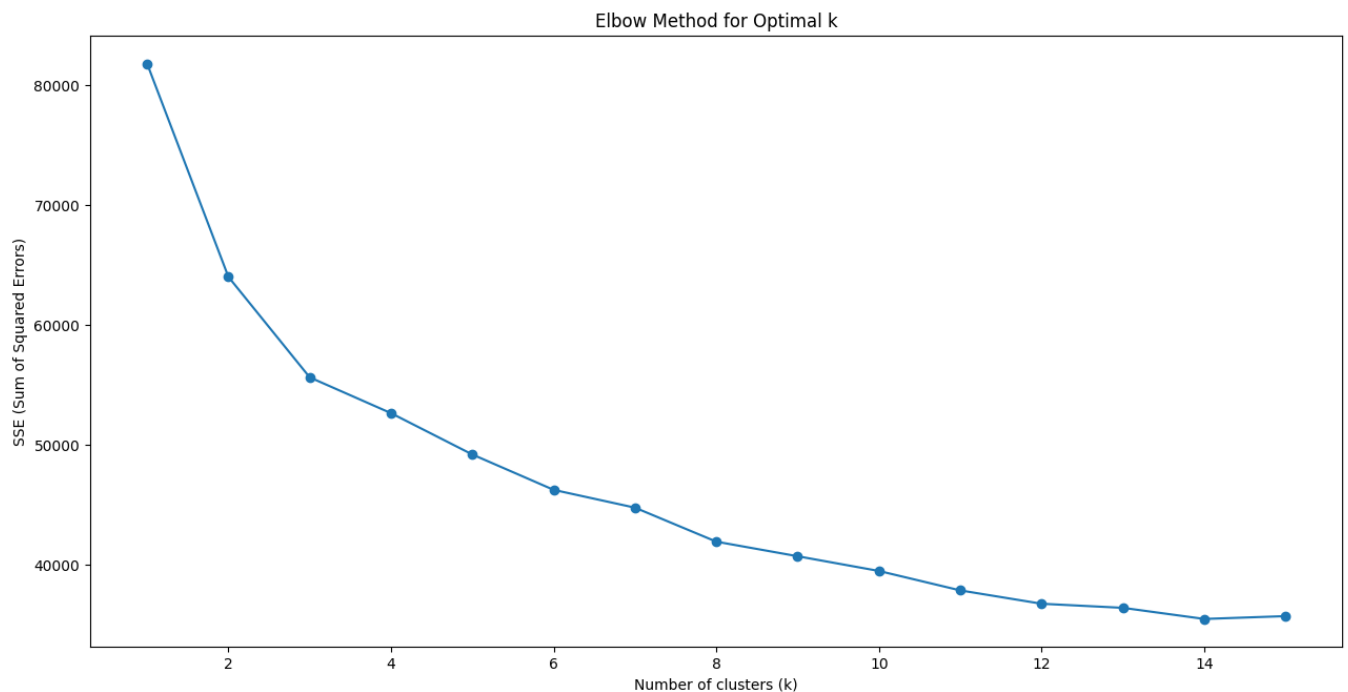
Yêu cầu 1: Phân nhóm cầu thủ

(*) Code: `ex3/select_k.py`

- Trước khi phân nhóm các cầu thủ có chỉ số tương đồng, ta tiến hành lựa chọn số nhóm cần chia bằng phương pháp Elbow:

+ Giả sử với k trong khoảng từ 1 – 15, tiến hành đo lường khoảng cách giữa các điểm dữ liệu và tâm cụm (SSE) theo từng k , và giá trị SSE nhỏ hơn đồng nghĩa với cụm dữ liệu tốt hơn.

+ Sau khi chạy code với dữ liệu từ phần 1, chúng ta được biểu đồ như sau:



(*) **Nhận xét:**

+ Khi k tăng từ 1 đến 3, SSE giảm rất nhanh vì việc thêm cụm mới giúp chia nhỏ dữ liệu thành các nhóm gần hơn với tâm cụm.

+ Từ $k = 4, 5$, sự giảm SSE chậm lại. Lúc này, việc thêm cụm mới chỉ giúp giảm SSE rất ít, trong khi chi phí tính toán và độ phức tạp của mô hình tăng lên. (1)

=> Dựa trên đặc điểm dữ liệu thực tế của các cầu thủ, chúng ta thấy có 4 vị trí chính bao gồm: **DF**(Hậu vệ), **MF**(Trung vệ), **FW**(Tiền đạo) và **GK**(Thủ môn). (2) Các cầu thủ chơi cùng 1 vị trí sẽ có điểm tương đồng nhất định về chỉ số

=> Như vậy, từ (1) & (2) việc chọn $k = 4$ cho thuật toán K-means là hợp lý.

```
# pip install sklearn
def main():
    df = pandas.read_csv('ex1/results.csv', na_values='N/a')
    target_cols = df.columns[5:]

    # Chuẩn hóa dữ liệu để tất cả các chỉ số có cùng đơn vị
    X = df[target_cols]
    X.fillna(X.mean(), inplace=True)
    X_scaled = StandardScaler().fit_transform(X)

    # Danh sách để lưu giá trị SSE = tổng bình phương khoảng cách giữa các điểm và tâm cụm
    sse = []
    k_values = range(1, 16)
    for k in k_values:
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(X_scaled)
        # SSE là thuộc tính inertia_ của KMeans trong sklearn
        sse.append(kmeans.inertia_)

    # Vẽ biểu đồ Elbow
    plt.figure(figsize=(12, 8))
    plt.plot(k_values, sse, marker='o')
    plt.xlabel('Number of clusters (k)')
    plt.ylabel('SSE (Sum of Squared Errors)')
    plt.title('Elbow Method for Optimal k')
    plt.show()

if __name__ == "__main__":
    main()
```

- Sau khi chọn được số nhóm (k), ta tiến hành phân cụm các cầu thủ theo chỉ số bằng K-means

(*) Code: ex3/kmean.py

+ Hàm hiển thị biểu đồ K-means:

```
def kmeans_display(X, label, centers, K):
    colors = ['ro', 'go', 'bo', 'co']

    for k in range(K):
        Xk = X[label == k, :]
        plt.plot(Xk[:, 0], Xk[:, 1], colors[k], markersize=4, alpha=0.8)

    plt.scatter(centers[:, 0], centers[:, 1], s=300, c='black', marker='x')
    plt.axis('equal')
    plt.title('KMeans Clustering')
```

+ Tiến hành đọc dữ liệu, chuẩn hóa đơn vị và vẽ biểu đồ K-means (Số chiều bằng số chỉ số):

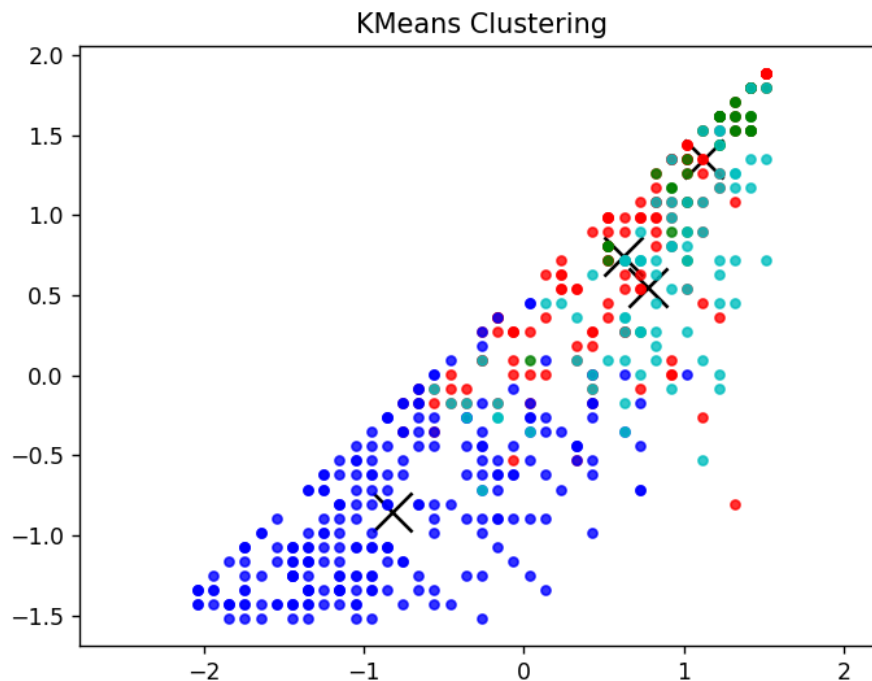
```
def main():
    df = pandas.read_csv('ex1/results.csv', na_values='N/a')
    # Dựa trên elbow ta chọn K = 4
    K = 4
    # Chọn các cột bắt đầu từ 6 ->
    target_cols = df.columns[5:]

    # Chuẩn hóa dữ liệu để tất cả các chỉ số có cùng đơn vị
    X = df[target_cols]
    X.fillna(X.mean(), inplace=True)
    X_scaled = StandardScaler().fit_transform(X)

    # Hiển thị kết quả phân cụm khi chưa giảm chiều
    kmeans_1 = KMeans(n_clusters=K, random_state=0)
    kmeans_1.fit(X_scaled)
    label = kmeans_1.predict(X_scaled)
    kmeans_display(X_scaled, label, kmeans_1.cluster_centers_, K)
    plt.show()

if __name__ == "__main__":
    main()
```

+ Sau khi chạy, ta được biểu đồ như sau:



Các tâm X là tâm cụm

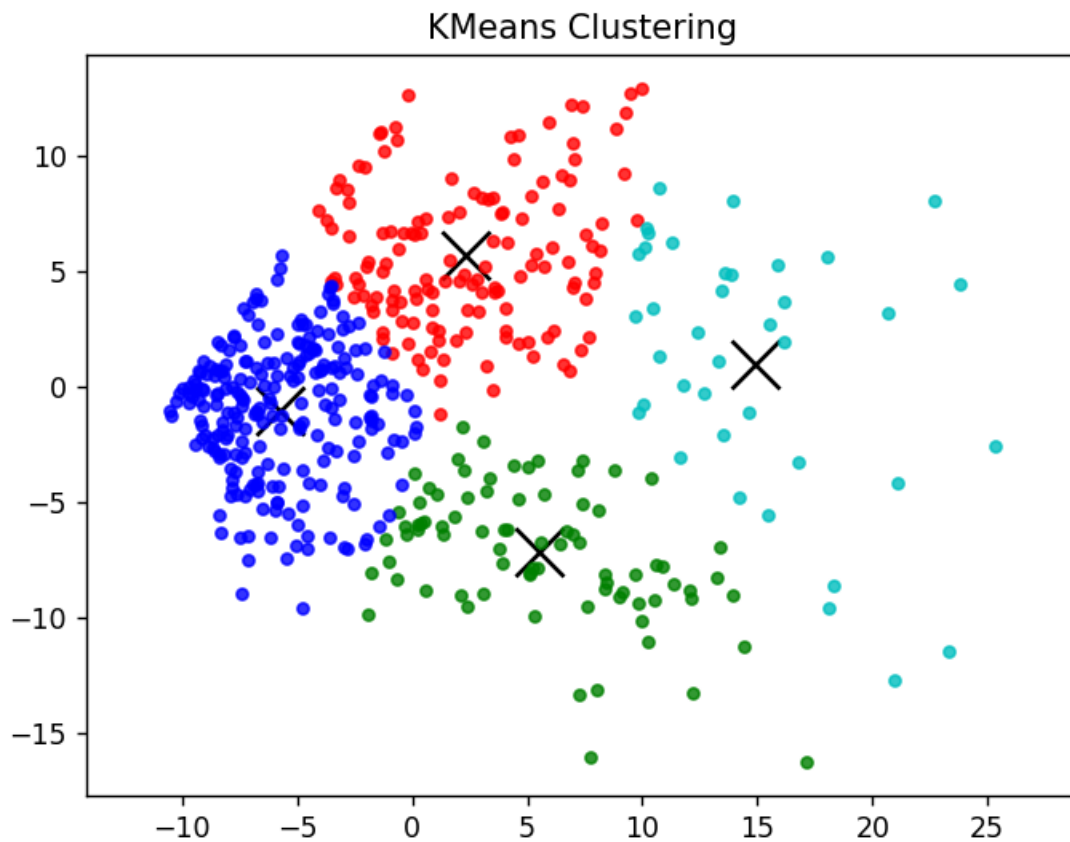
- Tương tự như trên, ta áp dụng thuật toán PCA để giảm chiều dữ liệu:

(*) Code: ex3/kmean_with_pca.py

+ Giảm xuống còn 2 chiều:

```
# Áp dụng PCA để giảm chiều dữ liệu xuống 2 chiều
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
```

+ Sau khi chạy code ta sẽ được:



=> Các cụm không phân chia rõ ràng do có nhiều cầu thủ đồng thời đảm nhiệm 2 vị trí.

Yêu cầu 2: Viết chương trình python vẽ biểu đồ rada (radar chart) so sánh cầu thủ

(*) Code: ex3/radarChartPlot.py

(*) Các bước:

- Định nghĩa hàm vẽ biểu đồ chỉ số cho 2 cầu thủ:

```
def draw_radar_chart(p1_name, p2_name, attributes, df):
    labels = attributes
    num_variables = len(attributes)
    player1 = df.loc[df['Player'] == p1_name, attributes].astype(float).values.flatten()
    player2 = df.loc[df['Player'] == p2_name, attributes].astype(float).values.flatten()

    player1 = np.append(player1, player1[0])
    player2 = np.append(player2, player2[0])

    # Tính toán góc
    angles = [n / float(num_variables) * 2 * pi for n in range(num_variables)]
    angles = np.append(angles, angles[0])

    fig, ax = plt.subplots(figsize=(8, 6), subplot_kw=dict(polar=True))

    # Vẽ cho cầu thủ 1
    ax.plot(angles, player1, linewidth=1, linestyle='solid', label=p1_name)
    ax.fill(angles, player1, 'b', alpha=0.1)

    # Vẽ cho cầu thủ 2
    ax.plot(angles, player2, linewidth=1, linestyle='solid', label=p2_name)
    ax.fill(angles, player2, 'r', alpha=0.1)

    # Thêm nhãn cho các trục
    plt.xticks(angles[:-1], labels)

    # Đặt tiêu đề và chú thích
    plt.title(f'So sánh giữa {p1_name} và {p2_name}')
    ax.legend(loc='upper right', bbox_to_anchor=(0.1, 0.1))

    plt.show()
```

- Đọc dữ liệu cầu thủ từ phần 1 + lấy các tham số từ dòng lệnh:

```
def main():
    # Đọc dữ liệu nguồn
    df = pandas.read_csv('ex1/results.csv')
    df.replace('N/a', 0, inplace=True)

    parser = argparse.ArgumentParser()
    parser.add_argument('--p1', required=True)
    parser.add_argument('--p2', required=True)
    parser.add_argument('--Attribute', required=True)
    args = parser.parse_args()
    attributes = args.Attribute.split(',')
    validAttrs = df.columns[5:]
```

- Kiểm tra tính hợp lệ của tham số và vẽ biểu đồ

```
# Check exist
if args.p1 not in df['Player'].values:
    print("ERROR: Tên cầu thủ p1 không có trong danh sách")
    sys.exit(-1)

if args.p2 not in df['Player'].values:
    print("ERROR: Tên cầu thủ p2 không có trong danh sách")
    sys.exit(-1)

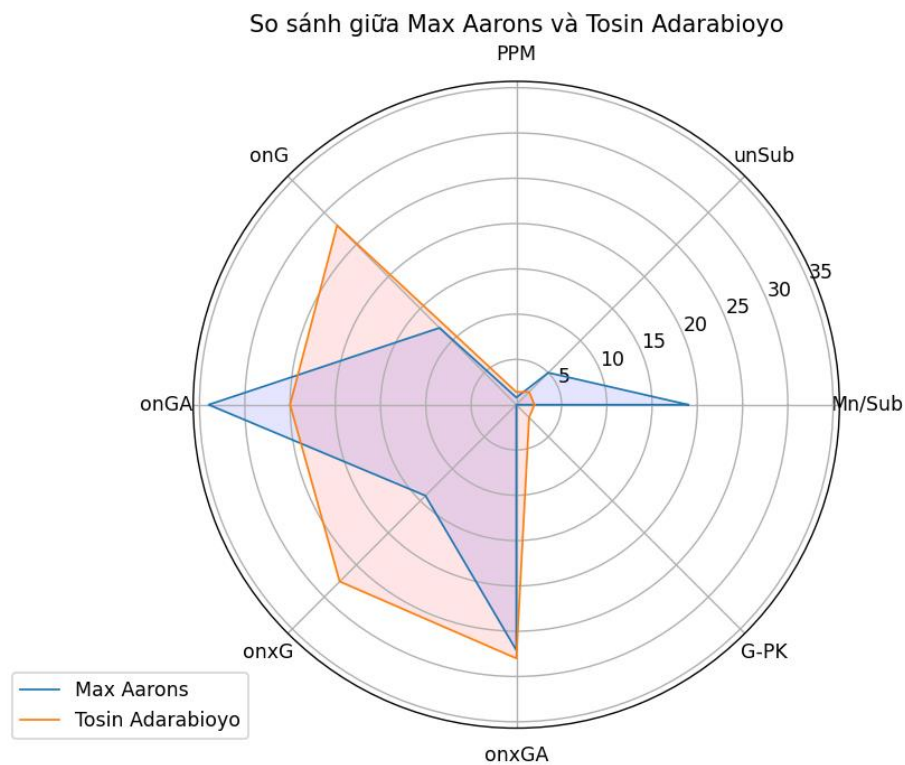
if not set(attributes).issubset(set(validAttrs)):
    print("ERROR: Tên thuộc tính hợp lệ bao gồm: " + ','.join(validAttrs))
    sys.exit(-1)

draw_radar_chart(p1_name=args.p1, p2_name=args.p2, attributes=attributes, df=df)

if __name__ == '__main__':
    main()
```

- Kết quả chạy với ví dụ:

python radarChartPlot.py --p1 'Max Aarons' --p2 'Tosin Adarabioyo' --Attribute 'Mn/Sub,unSub,PPM,onG,onGA,onxG,onxGA,G-PK'



PHẦN IV: 1 điểm

Yêu cầu 1: Thu thập giá chuyển nhượng của các cầu thủ trong mùa 2023-2024 (giải ngoại hạng anh) từ trang web

- Phương pháp: dùng paging request để lấy dữ liệu dạng json

(*) Code: ex4/main_1.py

(*) Các bước:

- Khởi tạo mảng 2 chiều lưu dữ liệu:

```
8 player_infos = [  
9     ... ['Player Name', 'Transfer price']  
10 ]
```

- Thiết lập dữ liệu gửi đi với kích thước trang tối đa => chỉ cần request trang 1 là lấy được tất cả dữ liệu cầu thủ:

```
# link và dữ liệu paging  
url = 'https://www.footballtransfers.com/en/transfers/actions/confirmed/overview'  
form_data = {  
    'orderBy': 'date_transfer',  
    'orderByDescending': 1,  
    'page': 1,  
    'pages': 0,  
    'pageItems': 500,  
    'countryId': 'all',  
    'season': 5847,  
    'tournamentId': 31,  
    'clubFromId': 'all',  
    'clubToId': 'all',  
    'transferFeeFrom': None,  
    'transferFeeTo': None,  
}  
res = requests.post(url, data=form_data)
```

- Chuỗi json nhận được sẽ như sau:

```
{
  "page": 1,
  "page_items": 500,
  "pages": 1,
  "active_filters": 2,
  "total_records": 291363,
  "filter_records": 427,
  "records": [
    {
      "id": "458593209",
      "player_id": null,
      "season_id": "5847",
      "player_name": "Demi Akarakiri",
      "player_slug": "",
      ...
      "price_tag": {
        "class": "only-tag",
        "price": "Free",
        "value": 0,
        "type": 1
      }
    },
  ],
}
```

- Dựa vào cấu trúc trên, ta dễ dàng thu thập được tên và giá chuyển nhượng của cầu thủ như sau:

```
if res.status_code == 200:
    json = res.json()

    for record in json['records']:
        player_info = []

        # Thêm tên, giá chuyển nhượng
        player_info.append(record['player_name'])
        player_info.append(record['price_tag']['price'])

        # Lưu dữ liệu
        player_infos.append(player_info)
        print(f'\r-> Đã khởi tạo {len(player_infos) - 1} cầu thủ!', end='')

    print(f'\nSUCCESS: Thời gian chạy: {time.time() - start_time}s')
else:
    print(f'ERROR: Lỗi crawl player link ({res.status_code})')
```


- Sau khi chạy xong, lưu kết quả vào file *transfer_price.csv*:

```
def main():
    getData()

    # Ghi vào file csv
    with open('ex4/transfer_price.csv', mode='w', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)
        writer.writerows(player_infos)
        print(f"SUCCESS: Dữ liệu đã ghi vào file: {file.name}")

if __name__ == "__main__":
    main()
```

(*) Kết quả sau khi chạy:

```
PS C:\Users\NGUYEN\Desktop\Python\BTL\source> & c:/Users/NGUYEN/Des
INFO: Bắt đầu thu thập thông tin...
-> Đã khởi tạo 427 cầu thủ!
SUCCESS: Thời gian chạy: 5.582660675048828s
SUCCESS: Dữ liệu đã ghi vào file: ex4/transfer_price.csv
PS C:\Users\NGUYEN\Desktop\Python\BTL\source> 
```