Báo Cáo Bài Tập Lập Trình Với Python

I. Viết chương trình Python thu thập dữ liệu phân tích cầu thủ

- Hàm GetDataFromWeb, sử dụng Selenium để truy cập vào một trang web, thu thập dữ liệu từ hai bảng khác nhau rồi trả về dữ liệu đã thu thập.

  1.Thu nhập dữ liệu các cầu thủ.

```python
def GetDataFromWeb(url, Xpath_player, Xpath_squad, Data_Name):
    driver = webdriver.Chrome()
    driver.get(url)

    resultPlayerData = []
    resultSquadData = []
    try:
        table2 = driver.find_element(By.XPATH, Xpath_player)
        rows2 = table2.find_elements(By.TAG_NAME, 'tr')

        for row in rows2:  # Bỏ qua hàng tiêu đề
            cols = row.find_elements(By.TAG_NAME, 'td')
            data = []
            for id, play in enumerate(cols[:-1]):
                if id == 1:
                    a = play.text.strip().split()
                    if len(a) == 2:
                        data.append(a[1])
                    else:
                        data.append(play.text.strip())
                else:
                    s = play.text.strip()
                    if id >= 4:
                        s = s.replace(",", "")
                        s = validdata(s)
                    data.append(s)
            if len(data) != 0: resultPlayerData.append(data)
```

(variable) table2: WebElement

  2.Thu nhập dữ liệu các đội:

- Truy cập vào trang web Premier League của FBRef để lấy dữ liệu cầu thủ và đội bóng, rồi xử lý và lưu trữ dữ liệu trong các đối tượng Player và Squad thông qua các hàm trong player_manager và squad_manager

```
url = "https://fbref.com/en/comps/9/2023-2024/stats/2023-2024-Premier-League-Stats"
xpath_player = '//*[@id="stats_standard"]'
xpath_Squad = '//*[@id="stats_squads_standard_for"]'
DataName = "Standard"
list_player_result, list_Squad_result = GetDataFromWeb(url,xpath_player, xpath_Squad, DataName)

for i in list_player_result:
    p = player_manager.findPlayerByNameandTeam(i[0],i[3])
    if p == None:
        new_p=Player(i[0],i[1],i[2],i[3],i[4])
        #bo qua i5 vi i5 la nam sinh
        new_p.setPlaying_time(i[6:9])
        #bo qua i9
        new_p.setPerformance([i[13],i[14],i[11],i[16],i[17]])
        new_p.setExpected(i[18:21])
        #bo qua i22
        new_p.setProgression(i[22:25])
        new_p.setPer90(i[25:])
        player_manager.add_Player(new_p)
```

```python
#squad data
for i in list_Squad_result:
    s = squad_manager.findSquadByName(i[0])
    if s == None:
        new_s=Squad(*i[0:4])
        new_s.setPlaying_time(i[4:7])
        #bo qua i9
        new_s.setPerformance([i[11],i[12],i[9],i[14],i[15]])
        new_s.setExpected(i[16:19])
        #bo qua i22
        new_s.setProgression(i[20:22])
        new_s.setPer90(i[22:])
        squad_manager.add_Squad(new_s)
```

- Thứ tự các cầu thủ sắp xếp theo thứ tự tên (First Name), nếu trùng tên thì xếp theo độ tuổi từ lớn đến nhỏ.(sử dụng hàm sortingByName đã được tạo sẵn trong class Player_Manager)

```python
player_manager.sortingByName()
```

- Thu thập dữ liệu của tất cả các cầu thủ có số phút thi đấu cần nhiều hơn 90 phút (Sử dụng hàm filtering trong class Player_Manager để lọc)

```python
# Thu thập dữ liệu
manager.get_data()

# Lọc cầu thủ có số phút thi đấu > 90
filtered_players = manager.filtering(90)
```

- tạo file CSV (result.csv) để lưu trữ dữ liệu của cầu thủ :

```python
# Lưu kết quả vào file CSV
import csv
from bai1.tieu_de import header, row
from bai1.tieu_de import header2, row2

with open('E:/World/Code/Python/BTL/bai1/file/honglinh_result.csv', mode='w', newline='', encoding='utf-8') as file:
    writer = csv.writer(file)
    writer.writerow(header)

    for player in player_manager.list_player:
        r = row(player)
        writer.writerow(r)
print("Exam 1 Success")

with open('E:/World/Code/Python/BTL/bai1/file/honglinh_result2.csv', mode='w', newline='', encoding='utf-8') as file:
    writer = csv.writer(file)
    writer.writerow(header2)

    for squad in squad_manager.list_squad:
        r = row2(squad)
        writer.writerow(r)
print("Exam 1 Success")
```

II. (2đ)

1. Tìm top 3 cầu thủ có điểm cao nhất và thấp nhất ở mỗi chỉ số.
    - Lấy dữ liệu các cầu thủ từ file player_data.txt

```python
def read_data_from_txt(file_path):
    data = []
    with open(file_path, 'r', encoding='utf-8') as file:
        # Bỏ qua dòng tiêu đề
        file.readline()
        # Đọc từng dòng dữ liệu
        for line in file:
            # Tách các giá trị bằng dấu tab và thêm vào danh sách
            values = line.strip().split("\t")
            data.append(values)
    return data

# Đọc dữ liệu cầu thủ và đội từ file
player_data = read_data_from_txt('E:/World/Code/Python/BTL/bai1/file/honglinh/player_data.txt')
squad_data = read_data_from_txt('E:/World/Code/Python/BTL/bai1/file/honglinh/squad_data.txt')
```

- Với mỗi chỉ số chúng ta sort để xắp xếp thứ tự:

```python
# Xử lý dữ liệu cầu thủ để tính toán max và min
for index, value in enumerate(header):
    if index < 3:
        continue
    player_data = sorted(player_data, key=lambda x: x[index])
    list_mm.add_max_min([
        value,
        player_data[0][0],
        player_data[1][0],
        player_data[2][0],
        player_data[-3][0],
        player_data[-2][0],
        player_data[-1][0]
    ])
```

2. Tìm trung vị của mỗi chỉ số. Tìm trung bình và độ lệch chuẩn của mỗi chỉsố cho các cầu thủ trong toàn giải và của mỗi đội. Ghi kết quả ra file results2.csv

```python
def calculate_median_for_player_data(player_data):
    # Giả sử mỗi phần tử của player_data chứa danh sách các giá trị [name, team, ..., chỉ số 1, chỉ số 2,...]
    # Tạo các danh sách rỗng cho các chỉ số
    num_columns = len(player_data[0])  # Số cột dữ liệu
    columns = [[] for _ in range(num_columns)]

    # Trích xuất dữ liệu các chỉ số (bỏ qua cột tên và đội nếu cần)
    for row in player_data:
        for i in range(len(row)):
            try:
                # Chuyển dữ liệu thành float (bỏ qua các cột không phải số nếu cần)
                value = float(row[i])
                columns[i].append(value)
            except ValueError:
                continue  # Bỏ qua các giá trị không phải số

    # Tính trung vị cho mỗi cột chỉ số
    medians = []
    for col in columns:
        if len(col) > 0:  # Đảm bảo cột không rỗng
            medians.append(statistics.median(col))
        else:
            medians.append(None)  # Nếu cột không có giá trị nào

    return medians
```

```python
def calculate_mean_std_for_player_data(player_data):
    # Giả sử mỗi phần tử của player_data chứa danh sách các giá trị [name, team, ..., chỉ số 1, chỉ số 2,...]
    num_columns = len(player_data[0])  # Số cột dữ liệu
    columns = [[] for _ in range(num_columns)]

    # Trích xuất dữ liệu các chỉ số (bỏ qua cột tên và đội nếu cần)
    for row in player_data:
        for i in range(len(row)):
            try:
                # Chuyển dữ liệu thành float (bỏ qua các cột không phải số)
                value = float(row[i])
                columns[i].append(value)
            except ValueError:
                continue  # Bỏ qua các giá trị không phải số

    # Tính trung bình và độ lệch chuẩn cho mỗi cột chỉ số
    stats = []
    for col in columns:
        if len(col) > 1:  # Đảm bảo cột có nhiều hơn 1 giá trị để tính độ lệch chuẩn
            mean = statistics.mean(col)
            std_dev = statistics.stdev(col)
            stats.append((mean, std_dev))
        else:
            stats.append((None, None))  # Nếu cột không có giá trị hoặc quá ít

    return stats
```

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

- Lấy dữ liệu cần thiết và tạo các thư mục :

```
# Trích xuất dữ liệu các chỉ số (bỏ qua cột tên và đội)
for row in player_data:
    for i in range(len(row)):
        try:
            # Chuyển dữ liệu thành float
            value = float(row[i])
            columns[i].append(value)
        except ValueError:
            continue  # Bỏ qua các giá trị không phải số

# Tạo thư mục lưu các biểu đồ nếu chưa tồn tại
os.makedirs(output_folder, exist_ok=True)

# Lưu từng biểu đồ dưới dạng ảnh riêng
for idx, col in enumerate(columns):
    if len(col) > 1:  # Đảm bảo có dữ liệu
        plt.figure(figsize=(8, 6))
        sns.histplot(col, bins=20, kde=True)  # Histogram với đườ
        plt.title(f'Distribution of Stat {idx}')
        plt.xlabel(f'Stat {idx}')
        plt.ylabel('Frequency')
        plt.grid(True)
        output_path = os.path.join(output_folder, f'stat_{idx}.pn
        plt.savefig(output_path)  # Lưu từng biểu đồ dưới dạng fi
        plt.close()  # Đóng figure để giải phóng bộ nhớ
```

- Vẽ historgram cho các cầu thủ trong giải đấu

```
cnt = 0
# Vẽ histogram cho toàn bộ giải đấu
for att in numeric_df.columns:
    cnt += 1
    plt.figure(figsize=(5, 3))
    plt.hist(numeric_df[att], bins=10, alpha=0.7, color='blue', edgecolor='black')
    plt.title(f'Histogram of {att} (All Teams)')
    plt.xlabel(att)
    plt.ylabel('Frequency')
    plt.grid(axis='y', alpha=0.75)
    # Lưu biểu đồ vào file .png
    output_path = os.path.join(output_folder1, f'stat_{cnt}.png')
    plt.savefig(output_path)
    plt.close()
```

- Vẽ historgram cho các cầu thủ trong mỗi đội

```
    # Lưu từng biểu đồ dưới dạng ảnh riêng
    for idx, col in enumerate(columns):
        if len(col) > 1:  # Đảm bảo có dữ liệu
            plt.figure(figsize=(8, 6))
            sns.histplot(col, bins=20, kde=True)  # Histogram với đường cong KDE
            plt.title(f'Distribution of Stat {idx}')
            plt.xlabel(f'Stat {idx}')
            plt.ylabel('Frequency')
            plt.grid(True)
            output_path = os.path.join(output_folder, f'stat_{idx}.png')
            plt.savefig(output_path)  # Lưu từng biểu đồ dưới dạng file ảnh riêng
            plt.close()  # Đóng figure để giải phóng bộ nhớ

    print(f'All plots are saved in the folder: {output_folder}')

# Sử dụng hàm để vẽ và lưu từng biểu đồ vào các file riêng trong thư mục
plot_and_save_each_stat(player_data, 'honglinh/player_stats')
plot_and_save_each_stat(squad_data, 'honglinh/squad_data')
```

4. Tìm đội bóng có chỉ số điểm số cao nhất ở mỗi chỉ số. Theo bạn đội nào có phong độ tốt nhất giải ngoại Hạng Anh mùa 2023-2024

//Phần này Thiếu.

III. (3đ)

1. Sử dụng thuật toán K-means để phân loại các cầu thủ thành các nhóm có chỉ số giống nhau

- Kiểm tra dữ liệu:

```
# Kiểm tra dữ liệu
print("Dữ liệu đọc được từ file:")
print(df.head())
print("Thông tin dữ liệu:")
print(df.info())
```

- Xử lý các giá trị thiếu nếu có

```
# Tiền xử lý: Xử lý các giá trị thiếu nếu có
numeric_columns = df.select_dtypes(include=[np.number]).columns
df[numeric_columns] = df[numeric_columns].fillna(df[numeric_columns].mean())
```

- Tìm số lượng cụm tối ưu

```
# Tìm số lượng cụm tối ưu
sse = []
silhouette_scores = []
for k in range(2, 10):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X)
    sse.append(kmeans.inertia_)
    silhouette_scores.append(silhouette_score(X, kmeans.labels_))
```

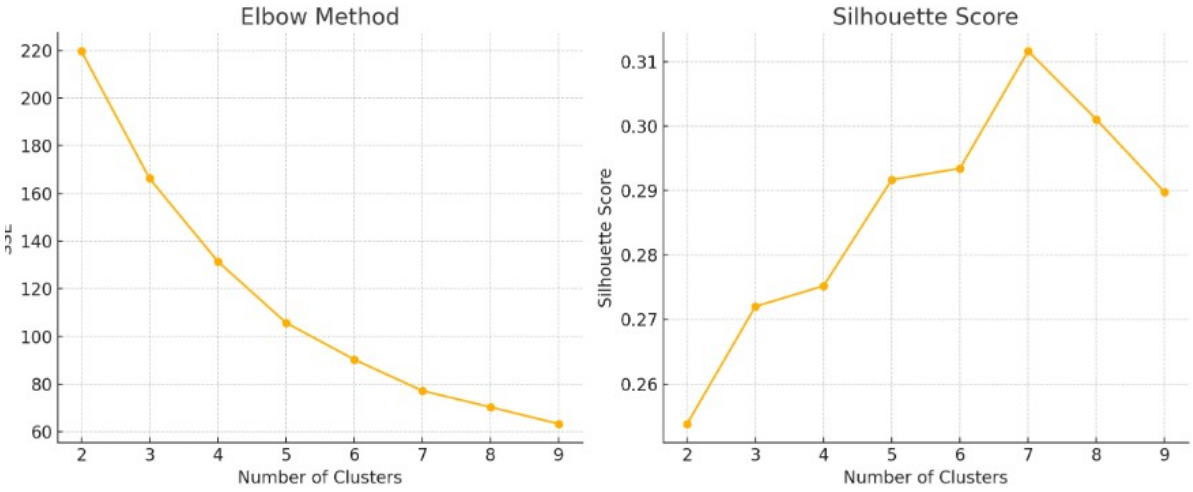
- Phương pháp Elbow để xác định SSE

```
# Phương pháp Elbow để xác định SSE
plt.subplot(1, 2, 1)
plt.plot(range(2, 10), sse, marker='o')
plt.xlabel("Number of Clusters")
plt.ylabel("SSE")
plt.title("Elbow Method")
```

- Đánh giá Silhouette Score

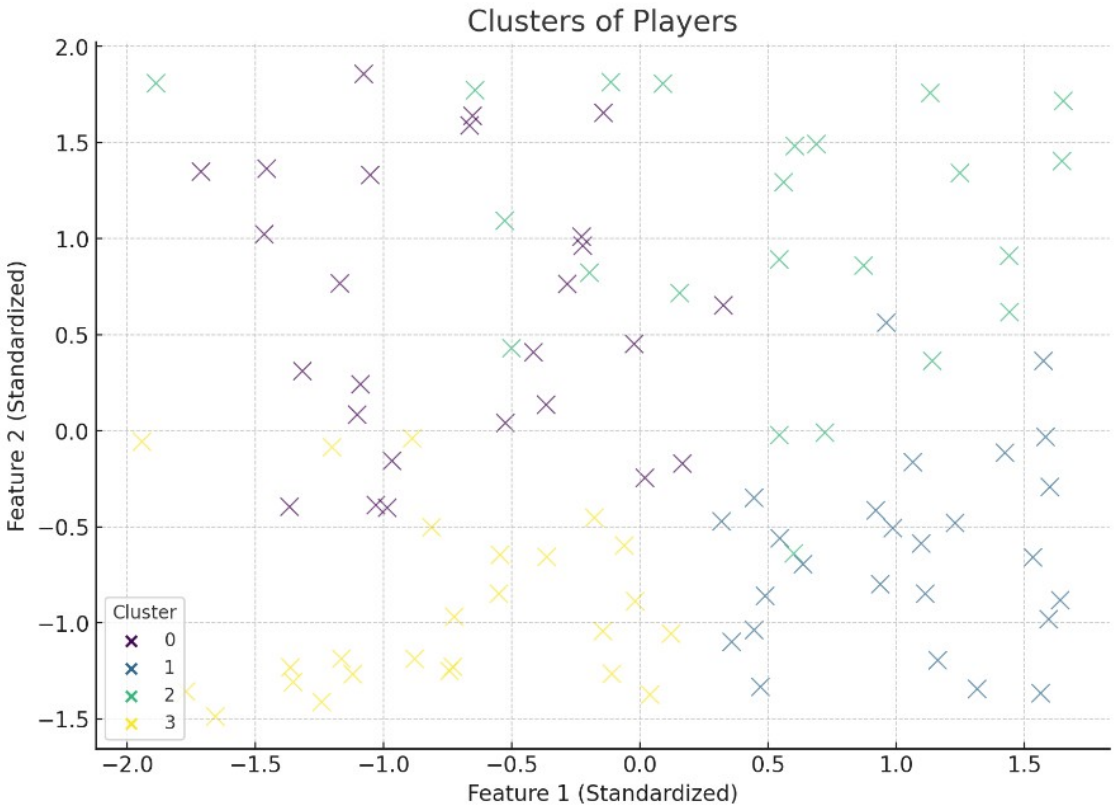
```python
# Đánh giá Silhouette Score
plt.subplot(1, 2, 2)
plt.plot(range(2, 10), silhouette_scores, marker='o')
plt.xlabel("Number of Clusters")
plt.ylabel("Silhouette Score")
plt.title("Silhouette Score")
plt.show()

# Sử dụng số cụm = 4 dựa trên kết quả từ đồ thị
optimal_k = 4
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
df['Cluster'] = kmeans.fit_predict(X)
```



- Trực quan hóa các cụm

```python
# Trực quan hóa các cụm
plt.figure(figsize=(10, 7))
sns.scatterplot(x=X[:, 0], y=X[:, 1], hue=df['Cluster'], palette='viridis', s=100, alpha=0.7)
plt.title('Clusters of Players')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend(title='Cluster')
plt.show()
```



⇨ Nhận xét:

Dựa vào hai biểu đồ, bạn chọn **số cụm tối ưu là 4**, có thể vì:

- Điểm "khuỷu tay" trong biểu đồ SSE xuất hiện xung quanh k=4, sau đó SSE giảm chậm lại.
- Silhouette Score tại k=4 có thể đạt mức cao tương đối, cho thấy các cụm khá rõ ràng.

Cân đối về mặt dữ liệu, giúp phân nhóm mà vẫn giữ sự khác biệt rõ ràng giữa cụm

2. Sử dụng thuật toán PCA, giảm số chiều dữ liệu xuống 2 chiều, vẽ hình phân cụm các điểm dữ liệu trên mặt 2D.

-Giảm số chiều dữ liệu xuống 2 chiều

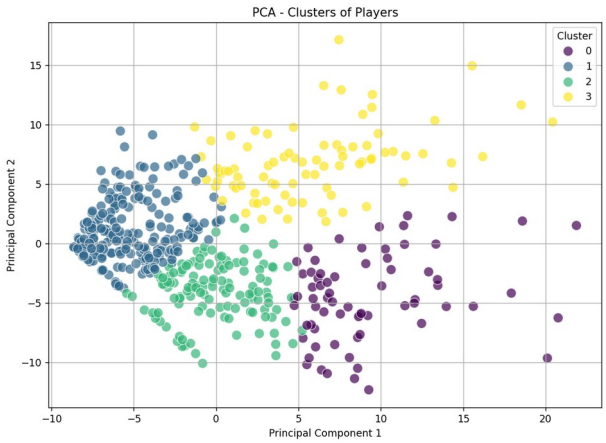
```
# Khởi tạo PCA và giảm số chiều xuống 2
pca = PCA(n_components=2)
data_reduced = pca.fit_transform(data)
```

-Phân cụm dữ liệu:

```
# Tạo DataFrame với các thành phần chính và cụm
df_pca = pd.DataFrame(data=X_pca, columns=['PC1', 'PC2'])
df_pca['Cluster'] = df['Cluster']
```

-Vẽ biểu đồ phân cụm

```
# Vẽ biểu đồ phân cụm
plt.figure(figsize=(10, 7))
sns.scatterplot(x='PC1', y='PC2', hue='Cluster', data=df_pca, palette='viridis', s=100, alpha=0.7)
plt.title('PCA - Clusters of Players')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title='Cluster')
plt.grid(True)
plt.show()
```



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

-Viết hàm create_radar_chart tạo một biểu đồ radar :

```python
def create_radar_chart(ax, player, attributes):
    # Đọc dữ liệu từ file
    df = pd.read_csv(args.file)

    # Lấy dữ liệu cho cầu thủ
    values = df.loc[df['name'] == player, attributes].values.flatten()

    # Số lượng thuộc tính
    num_vars = len(attributes)

    # Tạo một bảng cho biểu đồ radar
    angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()

    # Để hoàn thành vòng tròn
    values = np.concatenate((values, [values[0]]))
    angles += angles[:1]

    # Danh sách màu sắc cho mỗi thuộc tính
    colors = ['b', 'r', 'g', 'm', 'y']
```

```python
# Thiết lập màu nền cho biểu đồ
ax.set_facecolor('#f0f0f0')  # Bạn có thể thay đổi màu sắc này theo ý thích

# Vẽ biểu đồ radar với màu sắc cho từng thuộc tính
for i in range(num_vars):
    ax.plot(angles[i:i + 2], values[i:i + 2], color=colors[i % len(colors)], linewidth=2)
    ax.fill(angles, values, facecolor=colors[i % len(colors)], alpha=0.25)

ax.set_yticklabels([])
ax.set_xticks(angles[:-1])
ax.set_xticklabels(attributes)

# Thêm chú thích
ax.set_title(f'Radar Chart for: {player}', weight='bold', size='medium', position=(0.5, 1.1),
             horizontalalignment='center', verticalalignment='center')
```

-Tạo biểu đồ vào lưu biểu đồ vào file radar_charts.png

```python
# Đọc dữ liệu từ file
df = pd.read_csv(args.file)

# Lấy tên của hai cầu thủ từ hai dòng đầu tiên
player1 = df['name'].iloc[0]
player2 = df['name'].iloc[1]

# Đặt thuộc tính mặc định để so sánh
default_attributes = ['age', 'matches_played', 'PrgP', 'Pass_Cmp', 'Medium_Cmp', 'Pass_Live']  # Cập

# Tạo hình ảnh với hai biểu đồ radar
fig, axs = plt.subplots(1, 2, figsize=(16, 8), subplot_kw=dict(polar=True))

# Vẽ biểu đồ cho mỗi cầu thủ
create_radar_chart(axs[0], player1, default_attributes)
create_radar_chart(axs[1], player2, default_attributes)

# Lưu hình ảnh vào tệp PNG
plt.savefig('E:/World/Code/Python/BTL/bai3/file/honglinh/radar_charts.png', bbox_inches='tight')
plt.show()
```

IV. Thu thập giá chuyển nhượng của các cầu thủ trong mùa 2023-2024

- Dùng get_data_from_web để truy cập trang web và lấy dữ liệu và xử lí dữ liệu lấy được từ một bảng chứa thông tin cầu thủ.

```python
def get_data_from_web(driver, url, data_name):
    (variable) player_data: list
    player_data = []

    try:
        # Đợi cho phần tử tải lên
        WebDriverWait(driver, 10).until(
            EC.presence_of_element_located((By.XPATH, '//*[@id="content-part"]/section/div/div/div[1]/div/div[1]'))

        # Tải thêm dữ liệu nếu có
        while True:
            try:
                load_more_button = driver.find_element(By.XPATH, '//*[@id="onesignal-slidedown-allow-button"]')
                load_more_button.click()
            except Exception:
                print("All data loaded.")
                break

        # Tìm bảng dữ liệu
        div = driver.find_element(By.XPATH, '//*[@id="content-part"]/section/div/div/div[1]/div/div[1]')
        rows = div.find_elements(By.TAG_NAME, 'tr')
```

```
    # Lấy dữ liệu từ từng hàng
    for row in rows[1:]:
        data = []
        cols = row.find_elements(By.TAG_NAME, 'td')
        for index, play in enumerate(cols):
            if index == 0:
                hong_linh = play.find_element(By.CLASS_NAME, "text").text.strip().split()
                data.append(f"{hong_linh[0]} {hong_linh[1]}")
            elif index == 1:
                a = play.text.strip().split('\n')
                data.extend(a[:2])  # Lấy hai dòng đầu
            elif index == 2:
                continue
            else:
                data.append(play.text.strip())
        player_data.append(data)

except Exception as e:
    print(f"Error occurred: {e}")

print(f"Finish Page {data_name}")
return player_data
```

- Tạo 1 mảng để lưu trữ tất cả dữ liệu lấy được từ 18 web con

```
url = 'https://www.footballtransfers.com/us/leagues-cups/national/uk/premier-league/transfers/2023-2024'

# Sử dụng WebDriver
with webdriver.Chrome() as driver:
    get_player_data = get_data_from_web(driver, url, '')

    # Lặp qua các trang khác
    for index in range(2, 19):
        page_data = get_data_from_web(driver, f"{url}/{index}", str(index))
        get_player_data += page_data
```

- Thêm tiêu đề và lưu dữ liệu vào file result.csv

```
# Ghi dữ liệu vào file CSV
with open('E:/World/Code/Python/BTL/bai4/file/honglinh/result.csv', mode='w', newline='', encoding='utf-8') as file:
    writer = csv.writer(file)
    writer.writerow(header)
    writer.writerows(get_player_data)  # Sử dụng writerows để ghi tất cả dữ liệu

print("Exam 1 Success")
```