Jakubowski Adrian, Jaskuła Michał, Kruczek Michał

Inżynieria i Analiza Danych, grupa L3

Projekt

Gra piłkarska – Który droższy?

Wprowadzenie do języka Python

Dokumentacja projektu

# Temat i cel projektu

Tematem projektu jest gra piłkarska zaprogramowana w języku Python, która opiera się na wyborze, pomiędzy dwójką piłkarzy i decyzją użytkownika, który z podanych sportowców ma większą wartość rynkową. Celem projektu jest zapewnienie rozrywki osobom, które w stopniu zaawansowanym interesują się piłką nożną.

Nasza gra oferuje dwa tryby gry: nagła śmierć oraz tryb wieloosobowy. W pierwszym przypadku gracz próbuje przechodzić jak najwięcej rund, a każda pomyłka oznacza koniec gry. W ten sposób można śrubować swój osobisty rekord. W drugim trybie na starcie program pyta o liczbę graczy oraz o ilość rund przeznaczonych do gry. Ten tryb jest zatem idealny na rozgrywkę w gronie znajomych.

# Implementacja

## Biblioteki

Skrypt rozpoczynamy od załączenia odpowiednich bibliotek, do których należą:

- requests oraz BeautifulSoup – wykorzystywane podczas scrapowania danych ze strony internetowej

- pandas – do pracy na ramkach danych

- random – do losowania

import requests

from bs4 import BeautifulSoup

import pandas as pd #do dataframe

import random #do losowania

## Pobranie i obróbka danych

Najważniejszym aspektem naszego projektu było dokładne i cierpliwe przygotowanie danych oraz ich późniejsza obróbka. Dane dotyczące piłkarzy zaczerpnęliśmy z najpopularniejszej strony o tematyce piłkarskiej, a mianowicie z niemieckiego serwisu Transfermarkt. Na potrzeby projektu utworzyliśmy ramkę danych składającą się z 250 najdroższych piłkarzy na świecie.

Kod dotyczący scrapowania rozpoczęliśmy od pokazania programowi, że będziemy działać z poziomu przeglądarki.

headers = {'User-Agent':

           'Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.106 Safari/537.36'}

Następnie utworzyliśmy puste listy dotyczące atrybutów, które później będziemy umieszczać w ramce danych. Na potrzeby naszej gry, wystarczyło imię i nazwisko piłkarza oraz jego wartość.

PlayersList = []

ValuesList = []

Teraz rozpoczęła się najbardziej skomplikowana część naszego skryptu. W pętli for przechodzimy przez 10 kolejnych stron (na każdej stronie 25 piłkarzy), następnie stosujemy fragmenty kodu konieczne do dalszego pobierania danych (pageTree, pageSoup), które są powiązane z parsowaniem html. Aby ruszyć dalej, musieliśmy się zagłębić w kod html strony, aby odnaleźć pod jakimi tagami znajdują się imiona i nazwiska piłkarzy oraz ich wartości. Za pomocą funkcji find\_all znajdujemy wszystkie elementy zawarte pod tagami hauptlink (pod tym tagiem były imiona i nazwiska) oraz rechts hauptlink (wartości). Ostatnim krokiem jeśli chodzi o pobieranie danych było dodanie wartości do wcześniej utworzonych pustych list. Jeśli chodzi o PlayersList można dostrzec, że dodajemy co drugą wartość znalezioną poprzez find\_all z tagiem hauptlink. Dokonaliśmy takiego zabiegu, ponieważ funkcja find\_all znalazła wszystkie tagi, w którym zawiera się słowo hauptlink, a więc zarówno tag powiązany z imieniami i nazwiskami (hauptlink), jak i z wartościami (rechts hauptlink). Takich działań nie musieliśmy dokonywać w przypadku wartości piłkarzy, ponieważ tag rechts hauptlink dotyczył tylko i wyłącznie wyżej wspomnianych wartości.

#wybieramy strone do analizy

for pagenum in range(1, 11):

    page = "https://www.transfermarkt.pl/spieler-statistik/wertvollstespieler/marktwertetop?ajax=yw1&page=" + str(pagenum)

    pageTree = requests.get(page, headers=headers)

    pageSoup = BeautifulSoup(pageTree.content, 'html.parser')

    #wybieramy elementy na podstawie odpowiednich tagow html

    Players = pageSoup.find\_all("td", {"class": "hauptlink"})

    Values = pageSoup.find\_all("td", {"class": "rechts hauptlink"})

    #dodajemy wartosci do list (w przypadku Players biore co drugi element, ponieważ find\_all znalazło wszystkie klasy zarówno hauptlink oraz rechts hauptlink)

    for i in range(0,50,2):

        PlayersList.append(Players[i].text)

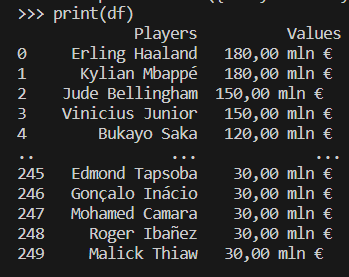
    for i in range(0,25):

        ValuesList.append(Values[i].text)

Końcowym etapem działania stricte na danych było utworzenie ramki danych oraz kosmetyka w postaci usunięcia zbędnych znaków oraz konwersja naszych danych dotyczących wartości na typ int, co było konieczne w kontekście porównywania liczb w dalszym etapie skryptu.

#tworzymy ramke danych

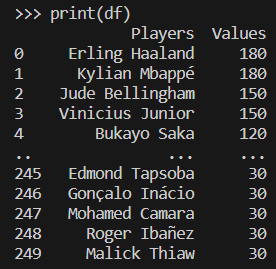
df = pd.DataFrame({"Players":PlayersList,"Values":ValuesList})



# Usunięcie zbędnych znaków i konwersja na typ int

df['Values'] = df['Values'].str.replace(' mln €', '').str.replace(',',

'.').astype(float).astype(int)



## Algorytm gry – nagła śmierć

Nasz algorytm rozpoczyna się od pętli while, która przechodzi cały czas, aż do momentu, gdy liczba zawodników wykluczonych (użytych w naszej grze) wyrówna się liczbie wszystkich piłkarzy w ramce danych. Przed główną pętlą deklarujemy zatem pustą listę wykluczonych piłkarzy, a także zmienne:

- points – do zliczania punktów

- rounds – do wyświetlania aktualnej rundy

- i – iterator

excluded\_values = []

points = 0

rounds = 1

i=1

Gra rozpoczyna się od wylosowania dwóch zawodników, sprawdzenia czy znajdują się oni w wykorzystanych już piłkarzach, a jeśli nie to przechodzimy do wizualizacji rozgrywki w konsoli. Możemy tam dostrzec numer rundy, wylosowanych zawodników oraz pytanie, na które musi odpowiedzieć gracz. Następnie przechodzimy do wyboru, który polega na wpisaniu do konsoli litery S, gdy wartość piłkarza jest taka sama jak drugiego, D jak zawodnik jest droższy oraz T, jeśli jest tańszy. Nie zapomnieliśmy również o obsłudze błędu złego wyboru. Dopóki gracz nie wybierze odpowiedniej litery, zostaje wyświetlony stosowny komunikat z prośbą o dokonanie poprawnego wyboru.

Po poprawnym wyborze następuje sprawdzenie czy odpowiedź jest prawidłowa. Jeśli tak, gracz dostaje punkt i rozpoczyna się kolejna runda. W przeciwnym wypadku gra kończy się i zostaje wyświetlony wynik końcowy.

while len(excluded\_values) < len(df['Players']): #dopóki nie wyczerpie sie liczba wykorzystanych zawodników

    x = random.randint(0, len(df['Players']) - 1) #losujemy liczbe

    while x in excluded\_values: #jesli x znajduje sie w wykluczonych liczbach

        x = random.randint(0, len(df['Players']) - 1) #losujemy ponownie

    excluded\_values.append(x) #dopisujemy wylosowaną liczbe do liczb wykluczonych

    y = random.randint(0, len(df['Players']) - 1) #losujemy drugą liczbe

    while y in excluded\_values: #jesli y znajduje sie w wykluczonych liczbach

        y = random.randint(0, len(df['Players']) - 1) #losujemy ponownie

    excluded\_values.append(y) #dopisujemy do wykluczonych

    #numer rundy oraz instrukcja wyboru

    print("--------- RUNDA " + str(rounds))

    print(df['Players'][x] + " " + str(df['Values'][x]) + " mln €   vs.   " + df['Players'][y] + " ??? mln € ")

    print("Czy " + df["Players"][y] + " jest drozszy (D)/ tanszy (T)/ ma taka sama wartosc (S) jak " + df["Players"][x])

    #obsluga bledu wyboru

    choice = input()

    while choice not in ["S","D","T"]:

        print("Wybierz S/D/T")

        choice = input()

    #sprawdzenie wyboru

    if (choice == "D" and df["Values"][x] < df["Values"][y]) or (choice == "S" and (df["Values"][x] == df["Values"][y])) or (choice == "T" and df["Values"][x] > df["Values"][y]): #jesli dobry wybor

        points += 1

        print("Zdobywasz punkt! " + df['Players'][y] + " jest warty " + str(df['Values'][y]) + " mln €. Twoja ilosc punktow to: " + str(points) + "\n")

    else: #jesli zly wybor

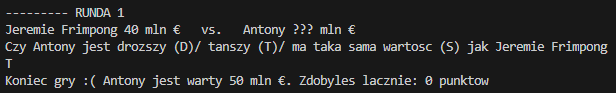
        print("Koniec gry :( " + df['Players'][y] + " jest warty " + str(df['Values'][y]) + " mln €. Zdobyles lacznie: " + str(points) + " punktow\n")

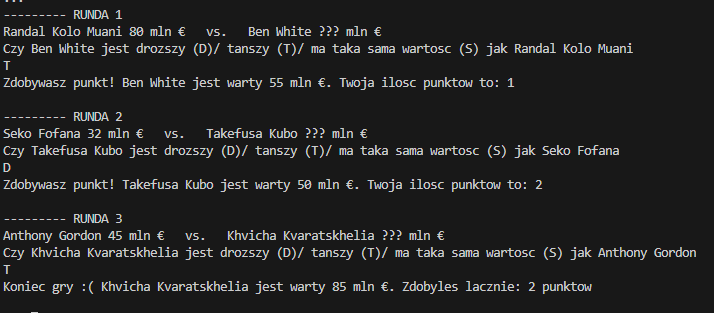
        break

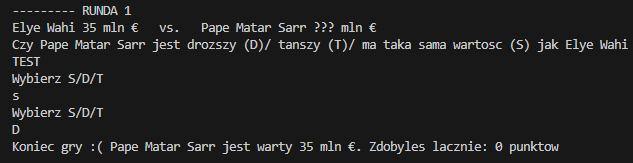
    rounds += 1

    i += 1

Tak prezentują się typowe przebiegi rozgrywki:







# Podsumowanie i wnioski