Web Scraping

Web Scraping je proces pridobivanja informacij z interneta.

S pomočjo web scrapinga lahko napišemo skripto, ki nas opozori, ko se nam približuje slabo vreme. Napišemo lahko skripto, ki nam pridobi vse tweete specifične osebe, pridobi trenutne informacije o stanju na cestah. Napišemo lahko skripto, ki se sprehodi čez članke na wikipediji in izpiše vse stavke, ki vsebujejo iskane besede, ipd.

Ponavadi ljudje uporabljamo internet preko **HTTP (HyperText Transfer Protocol)**.

> (v grobem): V browser napišemo spletni naslov katerega želimo obiskati. Browser nato izvede klic za pridobitev te spletne strani. Če spletna stran obstaja je posredovana nazaj v browser in ta nam prikaže spletno stran.

Za uporabo HTTP v pythonu obstaja knjižnjica **requests**.

> Dokumentacija: https://docs.python-requests.org/en/master/ (https://docs.python-requests.org/en/master/)

To je 3rd party knjižnjica, kar pomeni, da ne pride avtomatično z inštalacijo pythona. Zato jo moramo sami inštalirati.

Za inštalacijo 3rd party knjižnjic zapišemo ukaz **pip install <knjižnjica>** v terminal:

```
pip install requests
```

Za začetek bomo pridobili podatke o praznikih in dela prostih dneh v Republiki Sloveniji.

Informacije o podatkih lahko najdemo na sledeči spletni strani: https://podatki.gov.si/dataset/seznam-praznikov-in-dela-prostih-dni-v-republiki-sloveniji/resource/eb8b25ea-5c00-4817-a670-26e1023677c6 (https://podatki.gov.si/dataset/seznam-praznikov-in-dela-prostih-dni-v-republiki-sloveniji/resource/eb8b25ea-5c00-4817-a670-26e1023677c6)

Na spletni strani vidimo, da so podatki shranjeni v **csv** formatu.

Imajo 8 stolpcev:

- id
- Datum
- Ime praznika
- Dan v tednu
- Dela prost dan
- Dan

- Mesec
- Leto

Dejanske podatke lahko pridobimo na URL: https://podatki.gov.si/dataset/ada88e06-14a2-49c4-8748-3311822e3585/resource/eb8b25ea-5c00-4817-a670-26e1023677c6/download/seznampraznikovindelaprostihdni20002030.csv (https://podatki.gov.si/dataset/ada88e06-14a2-49c4-8748-3311822e3585/resource/eb8b25ea-5c00-4817-a670-26e1023677c6/download/seznampraznikovindelaprostihdni20002030.csv)

In [1]:

```python
import requests

url = "https://podatki.gov.si/dataset/ada88e06-14a2-49c4-8748-3311822e3585/resource

response = requests.get(url)
#print(r.encoding)
response.encoding = "utf-8" # treba dodat, ker če ne maš ISO-8859-1 kar pa ne prepo

data = response.text
print(data)
```

```
DATUM;IME_PRAZNIKA;DAN_V_TEDNU;DELA_PROST_DAN;DAN;MESEC;LETO
1.01.2000;novo leto;sobota;da;1;1;2000
2.01.2000;novo leto;nedelja;da;2;1;2000
8.02.2000;Prešernov dan, slovenski kulturni praznik;torek;da;8;2;200
0
23.04.2000;velika noč;nedelja;da;23;4;2000
24.04.2000;velikonočni ponedeljek;ponedeljek;da;24;4;2000
27.04.2000;dan boja proti okupatorju ;četrtek;da;27;4;2000
1.05.2000;praznik dela;ponedeljek;da;1;5;2000
2.05.2000;praznik dela;torek;da;2;5;2000
11.06.2000;binkoštna nedelja;nedelja;da;11;6;2000
25.06.2000;dan državnosti;nedelja;da;25;6;2000
15.08.2000;Marijino vnebovzetje;torek;da;15;8;2000
31.10.2000;dan reformacije;torek;da;31;10;2000
1.11.2000;dan spomina na mrtve;sreda;da;1;11;2000
25.12.2000;božič;ponedeljek;da;25;12;2000
26.12.2000;dan samostojnosti;torek;da;26;12;2000
1.01.2001;novo leto;ponedeljek;da;1;1;2001
2.01.2001;novo leto;torek;da;2;1;2001
8.02.2001;Prešernov dan, slovenski kulturni praznik;četrtek;da;8;2;2
```

Na začetku importiramo knjižnico **requests**, katero bomo uporabili za komuniciranje z internetom.

> requests dokumentacija: https://docs.python-requests.org/en/master/ (https://docs.python-requests.org/en/master/)

Nato v spremenljivko **url** shranimo naslov na katerem se nahajajo naši podatki.

Uporabimo **GET** metodo request knjižnjice. GET metoda ustvari HTTP Request, ki zahteva pridobitev spletne strani, oziroma v našem primeru bomo pridobili CSV podatke. Requests omogoča tudi uporabo ostalih HTTP Requests (POST, PUT, DELETE, HEAD, itd..).

Vse informacije našega request-a so shranjene v spremenljivki **response**. Da dostopamo do dejanski podatkov kličemo **response.text**.

Naša naloga bi sedaj lahko bila, da preverimo koliko praznikov pade na določen dan v tednu, za leto 2022.

In [2]:

```
import requests

url = "https://podatki.gov.si/dataset/ada88e06-14a2-49c4-8748-3311822e3585/resource

response = requests.get(url)
#print(r.encoding)
response.encoding = "utf-8" # treba dodat, ker če ne maš ISO-8859-1 kar pa ne prepo

data = response.text

rezultat = {}
for vrstica in data.split("\r\n"):
    v_splitted = vrstica.split(";")
    #print(v_splitted)
    if v_splitted[-1] == "2022":
        print(v_splitted)
        dan = v_splitted[2]
        if dan in rezultat.keys():
            rezultat[dan] += 1
        else:
            rezultat[dan] = 1

print("Število praznikov na specifični dan: ")
print(rezultat)
```

```
['1.01.2022', 'novo leto', 'sobota', 'da', '1', '1', '2022']
['2.01.2022', 'novo leto', 'nedelja', 'da', '2', '1', '2022']
['8.02.2022', 'Prešernov dan, slovenski kulturni praznik', 'torek', 'd
a', '8', '2', '2022']
['17.04.2022', 'velika noč', 'nedelja', 'da', '17', '4', '2022']
['18.04.2022', 'velikonočni ponedeljek', 'ponedeljek', 'da', '18',
'4', '2022']
['27.04.2022', 'dan boja proti okupatorju ', 'sreda', 'da', '27', '4',
'2022']
['1.05.2022', 'praznik dela', 'nedelja', 'da', '1', '5', '2022']
['2.05.2022', 'praznik dela', 'ponedeljek', 'da', '2', '5', '2022']
['5.06.2022', 'binkoštna nedelja', 'nedelja', 'da', '5', '6', '2022']
['8.06.2022', 'dan Primoža Trubarja', 'sreda', 'ne', '8', '6', '2022']
['25.06.2022', 'dan državnosti', 'sobota', 'da', '25', '6', '2022']
['15.08.2022', 'Marijino vnebovzetje', 'ponedeljek', 'da', '15', '8',
'2022']
['17.08.2022', 'združitev prekmurskih Slovencev z matičnim narodom',
'sreda', 'ne ', '17', '8', '2022']
['15.09.2022', 'vrnitev Primorske k matični domovini', 'četrtek', 'n
e', '15', '9', '2022']
['25.10.2022', 'dan suverenosti', 'torek', 'ne ', '25', '10', '2022']
['31.10.2022', 'dan reformacije', 'ponedeljek', 'da', '31', '10', '202
2']
['1.11.2022', 'dan spomina na mrtve', 'torek', 'da', '1', '11', '202
2']
['23.11.2022', 'dan Rudolfa Maistra', 'sreda', 'ne', '23', '11', '202
2']
['25.12.2022', 'božič', 'nedelja', 'da', '25', '12', '2022']
['26.12.2022', 'dan samostojnosti in enotnosti', 'ponedeljek', 'da',
'26', '12', '2022']
Število praznikov na specifični dan:
{'sobota': 2, 'nedelja': 5, 'torek': 3, 'ponedeljek': 5, 'sreda': 4,
'četrtek': 1}
```

URL katerega smo uporabili predstavlja API portala OPSI.

API (**Application Programming Interface**) predstavlja povezavo med dvema računalnikoma oziroma programoma.

V našem primeru je naš program kontaktiral portal OPSI preko API in pridobil podatke.

Veliko spletnih strani ima vzpostavljene API. Preko njihovih specifičnih URL-jev lahko tako dostopamo do njihovih urejenih podatkov.

Formati takšnih podatkov so velikokrat standardni, kot so CSV, XML, JSON, itd...

Za primer bolj naprednega API si poglejmo **coingecko.com**. To je spletna platforma za spremljanje trgovanja s kriptovalutami. Imajo informacije o trenutni ceni, volumnu, market cap, novicah, itd.

> https://www.coingecko.com/en (https://www.coingecko.com/en)

Dokumentacijo svojega API imajo lepo zapisano na:

> https://www.coingecko.com/api/documentations/v3#/ (https://www.coingecko.com/api/documentations/v3#/)

Vidimo, da so vse metode **GET** in okvirno kako so URL sestavljeni.

Za primer vzemimo nalogo, kjer moramo poiskati trenutno ceno Bitcoina v €.

API kateri nam bi lahko rešil nalogo je **GET /simple/price**. Če ga odpremo vidimo, da lahko izberamo še dodatne parametre in, da nam spletna stran sama zgenerira URL in nam tudi nudi možnost testiranja tega URL.

> https://api.coingecko.com/api/v3/simple/price?ids=bitcoin&vs_currencies=eur (https://api.coingecko.com/api/v3/simple/price?ids=bitcoin&vs_currencies=eur)

Podatke bomo dobili vrnjene v JSON formatu. JSON format je podoben python dictionary.

Če sedaj odpremo podani URL se nam v brskalniku izpišejo JSON podatki katere bi prejeli, če bi URL klicali s programom.

In [3]:

```python
import requests

url = "https://api.coingecko.com/api/v3/simple/price?ids=bitcoin&vs_currencies=eur"

r = requests.get(url)
data = r.json()
print(data)
print("Cena BTC v €: ", data["bitcoin"]["eur"])
```

```
{'bitcoin': {'eur': 50916}}
Cena BTC v €:  50916
```

Če bi sedaj podatke želeli v $ namesto v €, bi morali spremeniti URL.

In [4]:

```python
import requests

url = "https://api.coingecko.com/api/v3/simple/price?ids=bitcoin&vs_currencies=usd"

r = requests.get(url)
data = r.json()
print(data)
print("Cena BTC v $: ", data["bitcoin"]["usd"])
```

```
{'bitcoin': {'usd': 57399}}
Cena BTC v $:  57399
```

URL je v grobem sestavljen iz:

- **Base URL**, ki predstavlja pot do spletne strani. `api.coingecko.com/api/v3/simple/price`
- **Query parameters**, ki predstavljajo parametre katere lahko spreminjamo. Pričnejo se po `?`

Query parameters so sestavljeni iz:

- **imena parametra** - `id`
- **=**, enačaja
- **vrednosti parametra** - `bitcoin`

Med seboj so parametri ločeni z `&`.

Recimo, da imamo naš portfolio sestavljen iz sledečih kriptovalit:

```
["bitcoin", "ethereum", "cardano", "polkadot", "secret"]
```

Ko zaženemo naš program bi radi, da nam izpiše trenutno ceno vsakega kovanca v našem portfoliju. To pomeni, da bomo morali URL-je dinamično kreirati.

In [5]:

```python
import requests

my_portfolio = ["bitcoin", "ethereum", "cardano", "polkadot", "secret"]

for coin in my_portfolio:
    url = f"https://api.coingecko.com/api/v3/simple/price?ids={coin}&vs_currencies=

    r = requests.get(url)
    data = r.json()
    print(f"Cena {coin} v €: ", data[coin]["eur"])
```

```
Cena bitcoin v €:  50916
Cena ethereum v €:  3724.55
Cena cardano v €:  1.6
Cena polkadot v €:  35.48
Cena secret v €:  6.86
```

In [ ]:

```

```

Naloga:

Pridobite **daily** podatke o **ceni in market_cap** za do 3 dni nazaj za naš portfolijo. Podatki naj bodo v €.

```
["bitcoin", "ethereum", "cardano", "polkadot", "secret"]

OUTPUT

bitcoin
Price in €: 50120.35,      MC: 946129966385.45
Price in €: 51792.97,      MC: 977748021681.97
Price in €: 53231.48,      MC: 1004952689342.15

ethereum
Price in €: 3512.04,      MC: 415518933689.03
Price in €: 3825.25,      MC: 451060688164.44
Price in €: 3930.32,      MC: 464722167457.57

cardano
Price in €: 1.57,      MC: 50210489214.95
Price in €: 1.66,      MC: 53018653910.54
Price in €: 1.71,      MC: 54707905428.96

polkadot
Price in €: 34.24,      MC: 36178751146.04
Price in €: 36.70,      MC: 38654769090.58
Price in €: 37.37,      MC: 39428742405.97

secret
Price in €: 6.22,      MC: 926683065.11
Price in €: 6.46,      MC: 961627834.47
Price in €: 6.38,      MC: 951412154.96:
```

In [ ]:

```
import requests
my_portfolio = ["bitcoin", "ethereum", "cardano", "polkadot", "secret"]

for coin in my_portfolio:
    url = f"https://api.coingecko.com/api/v3/coins/{coin}/market_chart?vs_currency=

    r = requests.get(url)

    data = r.json()
    print(coin)
    for i in range(3):
        print(f"Price in €: {data['prices'][i][1]:.2f}, \t MC: {data['market_caps']
    print()
```

In [ ]:

```

```

Naloga:

S pomočjo webscrapinga preverite ali bi se lahko z Bicikelj odpeljali domov.

Vaša začetna postaja je TRG MDB

Vaša končna postaja je STARA CERKEV.

Preverite ali je na začetni postaji vsaj 1 prosto kolo in ali je na končni postaji vsaj 1 prosto parkirno mesto.

Podatke lahko dobite na sledečem linku v JSON formatu. Podatki o prostih mestih in kolesih se nahaja v "station" delu.

```
free nam pove koliko prostih mest je na postaji.
available nam pove koliko koles je prostih za izposojo.
```

[https://opendata.si/promet/bicikelj/list/ (https://opendata.si/promet/bicikelj/list/)](https://opendata.si/promet/bicikelj/list/)

In [ ]:

```
# Rešitev:

import requests

url = "https://opendata.si/promet/bicikelj/list/"

r = requests.get(url)

data = r.json()


free_bike = False
free_park = False
for key, station in data["markers"].items():

    if station["address"] == "TRG MDB":
        #print(station)
        if int(station["station"]["available"]) > 0:
            free_bike = True

    if station["address"] == "STARA CERKEV":
        #print(station)
        if int(station["station"]["free"]) > 0:
            free_park = True

if free_bike and free_park:
    print("Lahko greš z Bicikelj")
else:
    print("Ne moreš se odpeljati")
```

In [ ]:

```

```

Web Scraping with Beautiful Soup

Problem se nam pojavi, če spletne strani nimajo API.

Za primer vzemimo nalogo, kjer želimo pridobiti informacije o episodah serije Game of Thrones - No.overall, No. in season, Title, Directed by, Written by, Original air date, U.S. viewers (millions).

> https://en.wikipedia.org/wiki/List_of_Game_of_Thrones_episodes (https://en.wikipedia.org/wiki/List_of_Game_of_Thrones_episodes)

Spletna stran v naši nalogi je napisana v HTML (HyperText Markup Language). Ta zapis spletne strani je posredovan našemu browserju in ta ga spremeni v nam prijazno obliko (dizajn, itd.). Dejanski HTML zapis lahko vidimo s pomočjo "developers tools" - Ctrl+Shift+I (Chrome).

In celotno to kodo (HTML) dobimo, če uporabimo naš zgornji postopek in naredimo GET klic na naš URL.

In [8]:

```python
import requests

url = "https://en.wikipedia.org/wiki/List_of_Game_of_Thrones_episodes"
r = requests.get(url)

print(r.text)
```

```
<!DOCTYPE html>
<html class="client-nojs" lang="en" dir="ltr">
<head>
<meta charset="UTF-8"/>
<title>List of Game of Thrones episodes - Wikipedia</title>
<script>document.documentElement.className="client-js";RLCONF={"wgBr
eakFrames":!1,"wgSeparatorTransformTable":["",""],"wgDigitTransformT
able":["",""],"wgDefaultDateFormat":"dmy","wgMonthNames":["","Januar
y","February","March","April","May","June","July","August","Septembe
r","October","November","December"],"wgRequestId":"0344dc15-b79f-458
0-9b19-0820211bc628","wgCSPNonce":!1,"wgCanonicalNamespace":"","wgCa
nonicalSpecialPageName":!1,"wgNamespaceNumber":0,"wgPageName":"List_
of_Game_of_Thrones_episodes","wgTitle":"List of Game of Thrones epis
odes","wgCurRevisionId":1050374473,"wgRevisionId":1050374473,"wgArti
cleId":31120069,"wgIsArticle":!0,"wgIsRedirect":!1,"wgAction":"vie
w","wgUserName":null,"wgUserGroups":["*"],"wgCategories":["Use Ameri
can English from July 2020","All Wikipedia articles written in Ameri
can English","Use mdy dates from May 2020","Articles with short desc
ription","Short description is different from Wikidata","Official we
```

In [ ]:

```
```

HTML Quick Overview

Dodatna vsebina:

https://www.w3schools.com/html/default.asp (https://www.w3schools.com/html/default.asp)

HTML je sestavljena iz elementov imenovanih **tags**.

Najbolj osnoven tag je `<html> </html>`. Ta tag nam pove, da je vse znotraj njega HTML koda.

Znotraj `<html>` obstajata dva taga:

- `<head></head>` - vsebuje meta podatke o naši spletni strani

- `<body></body>` - vsebuje spletno stran katero vidimo v browserju (naslovi, text, slike, itd.)

```
<html>
    <head>
    </head>

    <body>
    </body>
</html>
```

Tage lahko vstavljamo znotraj drugih tagov, kot sta vstavljena `<head>` in `<body>` znotraj `<html>` . Tagi imajo tako lahko:

- **parent tag** - tag znotraj katerega se nahajajo
- **child tag** - tag, ki se nahaja znotraj njih
- **sibling tag** - tagi, ki se nahajajo v istem parent tag-u

Za dodajanje teksta se najbolj uporablja `<p> Text </p>` tag.

example_01.html

```
<html>
    <head>
    </head>

    <body>
        <p>Webscraping je proces pridobivanja podatkov iz interneta.</p>
    </body>
</html>
```

Če sedaj ponovno odpremo developer's tools lahko točno vidimo naši HTML kodo.

Tag-i imajo tudi določene lastnosti / atribute katere lahko spreminjamo.

Za primer vzemimo tag `<a></a>` , ki deluje kot hiperpovezava / link na drugo spletno stran.

```
<a href="https://www.google.com">Link</a>
```

Tag a ima atribut **href** katerega vrednost je *google.com*, ki nam pove na katero spletno stran naj nas hiperpovezava preusmeri, ko kliknemo na tekst *Link*.

example_02.html

```
<html>
    <head>
    </head>

    <body>
        <p>Webscraping je proces pridobivanja podatkov iz interneta.</p>
        <a href="https://www.google.com">Google brskalnik</a>
    </body>
</html>
```

Dodatno lahko spreminjamo lastnosti tag-ov s pomočjo **class** in **id** atributov. Z njimi lahko spreminjamo izgled naših elementov (barva, velikost, ...) oziroma prikazovanje (element lahko skrijemo, naredimo transparentnega, itd.).

Isti **class** si lahko deli več tag-ov, medtem ko **id** naj bi bil specifičen samo za en tag.

example_03.html

```
<html>
    <head>
        <style>
            #first_text {
                font-size: 20px;
            }

            .red_text {
                color: red;
            }
        </style>
    </head>

    <body>
        <p id="first_text">Webscraping je proces pridobivanja podatkov iz i
nterneta.</p>
        <a href="https://www.google.com">Google brskalnik</a>
        <p class="red_text"> Ta tekst naj bo obarvan rdeče.</p>
    </body>
</html>
```

Poglejmo si sedaj našo nalogo.

S pomočjo developer's tools lahko vidimo, da se podatki za prvo sezono nahajao znotraj `<table>` tag-ov, ki imajo **class="wikitable plainrowheaders wikiepisodetable"**.

In [9]:

```
import requests

url = "https://en.wikipedia.org/wiki/List_of_Game_of_Thrones_episodes"
r = requests.get(url)

print(r.content)
```

```
b'<!DOCTYPE html>\n<html class="client-nojs" lang="en" dir="ltr">\n<
head>\n<meta charset="UTF-8"/>\n<title>List of Game of Thrones episo
des - Wikipedia</title>\n<script>document.documentElement.className
="client-js";RLCONF={"wgBreakFrames":!1,"wgSeparatorTransformTable":
["",""],"wgDigitTransformTable":["",""],"wgDefaultDateFormat":"dm
y","wgMonthNames":["","January","February","March","April","May","Ju
ne","July","August","September","October","November","December"],"wg
RequestId":"0344dc15-b79f-4580-9b19-0820211bc628","wgCSPNonce":!1,"w
gCanonicalNamespace":"","wgCanonicalSpecialPageName":!1,"wgNamespace
Number":0,"wgPageName":"List_of_Game_of_Thrones_episodes","wgTitl
e":"List of Game of Thrones episodes","wgCurRevisionId":105037447
3,"wgRevisionId":1050374473,"wgArticleId":31120069,"wgIsArticle":!
0,"wgIsRedirect":!1,"wgAction":"view","wgUserName":null,"wgUserGroup
s":["*"],"wgCategories":["Use American English from July 2020","All
Wikipedia articles written in American English","Use mdy dates from
May 2020","Articles with short description","Short description is di
fferent from Wikidata","Official website not in Wikidata","Featured
lists",\n"Pages using the Graph extension","Game of Thrones episode
s","Lists of American drama television series episodes","Lists of fa
```

Sedaj bi lahko sami poiskali vse tabele sezon in ročno našli željene podatke. Vendar je to preveč zakomplicirano.

Za lažje navigiranje po HTML kodi obstaja knjižnjica **BeautifulSoup**.

```
pip install beautifulsoup4
```

In [10]:

```python
import requests
from bs4 import BeautifulSoup

url = "https://en.wikipedia.org/wiki/List_of_Game_of_Thrones_episodes"
r = requests.get(url)

soup = BeautifulSoup(r.text, "html.parser")
print(soup.prettify())
```

```
<!DOCTYPE html>
<html class="client-nojs" dir="ltr" lang="en">
 <head>
  <meta charset="utf-8"/>
  <title>
   List of Game of Thrones episodes - Wikipedia
  </title>
  <script>
   document.documentElement.className="client-js";RLCONF={"wgBreakFr
ames":!1,"wgSeparatorTransformTable":["",""],"wgDigitTransformTabl
e":["",""],"wgDefaultDateFormat":"dmy","wgMonthNames":["","Januar
y","February","March","April","May","June","July","August","Septembe
r","October","November","December"],"wgRequestId":"0344dc15-b79f-458
0-9b19-0820211bc628","wgCSPNonce":!1,"wgCanonicalNamespace":"","wgCa
nonicalSpecialPageName":!1,"wgNamespaceNumber":0,"wgPageName":"List_
of_Game_of_Thrones_episodes","wgTitle":"List of Game of Thrones epis
odes","wgCurRevisionId":1050374473,"wgRevisionId":1050374473,"wgArti
cleId":31120069,"wgIsArticle":!0,"wgIsRedirect":!1,"wgAction":"vie
w","wgUserName":null,"wgUserGroups":["*"],"wgCategories":["Use Ameri
```

In [ ]:

Za začetek lahko izberemo vse **child tags** naše spletne strani, kar nam bo vrnilo osnovno strukturo `<!DOCTYPE html>` in `<html>` tags.

In [11]:

```python
soup_children = list(soup.children)
print(type(soup_children))
print(len(soup_children))
print(soup_children)
```

```
<class 'list'>
3
['html', '\n', <html class="client-nojs" dir="ltr" lang="en">
<head>
<meta charset="utf-8"/>
<title>List of Game of Thrones episodes - Wikipedia</title>
<script>document.documentElement.className="client-js";RLCONF={"wgBr
eakFrames":!1,"wgSeparatorTransformTable":["",""],"wgDigitTransformT
able":["",""],"wgDefaultDateFormat":"dmy","wgMonthNames":["","Januar
y","February","March","April","May","June","July","August","Septembe
r","October","November","December"],"wgRequestId":"0344dc15-b79f-458
0-9b19-0820211bc628","wgCSPNonce":!1,"wgCanonicalNamespace":"","wgCa
nonicalSpecialPageName":!1,"wgNamespaceNumber":0,"wgPageName":"List_
of_Game_of_Thrones_episodes","wgTitle":"List of Game of Thrones epis
odes","wgCurRevisionId":1050374473,"wgRevisionId":1050374473,"wgArti
cleId":31120069,"wgIsArticle":!0,"wgIsRedirect":!1,"wgAction":"vie
w","wgUserName":null,"wgUserGroups":["*"],"wgCategories":["Use Ameri
can English from July 2020","All Wikipedia articles written in Ameri
can English","Use mdy dates from May 2020","Articles with short desc
```

Izberimo zadnji element, ki predstavlja našo **html** kodo.

Če preverimo njegov tip vidimo, da je to `bs4.element.Tag` - to je beautiful soup objekt, ki predstavlja naš tag.

In [12]:

```python
html = list(soup.children)[2] # equivalent to soup.html
print(type(html))
print(html)
```

```
<class 'bs4.element.Tag'>
<html class="client-nojs" dir="ltr" lang="en">
<head>
<meta charset="utf-8"/>
<title>List of Game of Thrones episodes - Wikipedia</title>
<script>document.documentElement.className="client-js";RLCONF={"wgBr
eakFrames":!1,"wgSeparatorTransformTable":["",""],"wgDigitTransformT
able":["",""],"wgDefaultDateFormat":"dmy","wgMonthNames":["","Januar
y","February","March","April","May","June","July","August","Septembe
r","October","November","December"],"wgRequestId":"0344dc15-b79f-458
0-9b19-0820211bc628","wgCSPNonce":!1,"wgCanonicalNamespace":"","wgCa
nonicalSpecialPageName":!1,"wgNamespaceNumber":0,"wgPageName":"List_
of_Game_of_Thrones_episodes","wgTitle":"List of Game of Thrones epis
odes","wgCurRevisionId":1050374473,"wgRevisionId":1050374473,"wgArti
cleId":31120069,"wgIsArticle":!0,"wgIsRedirect":!1,"wgAction":"vie
w","wgUserName":null,"wgUserGroups":["*"],"wgCategories":["Use Ameri
can English from July 2020","All Wikipedia articles written in Ameri
can English","Use mdy dates from May 2020","Articles with short desc
ription","Short description is different from Wikidata","Official we
```

Da vidimo ime našega tag-a lahko uporabimo `tag.name` . Da vidimo njegove atribute lahko uporabimo

```
tag.attrs
```

In [13]:

```
print(html.name)
print(html.attrs)
```

```
html
{'class': ['client-nojs'], 'lang': 'en', 'dir': 'ltr'}
```

- class - definira razrede tag-a
- lang - definira jezik v katerem je vsebina tag-a
- dir - specificira smer texta (ltr -> left to right) https://www.w3schools.com/tags/att_dir.asp (https://www.w3schools.com/tags/att_dir.asp)

Da se premaknemo naprej do naše tabele izberemo *children* od našega *html* tag-a. Specifično želimo **body**.

In [14]:

```
html_children = html.children
for c in html_children:
    print(c.name)
    #print(c)
```

```
None
head
None
body
```

In [15]:

```
body = list(html.children)[3]
print(body.name)
print(body.attrs)
```

```
body
{'class': ['mediawiki', 'ltr', 'sitedir-ltr', 'mw-hide-empty-elt', 'ns
-0', 'ns-subject', 'mw-editable', 'page-List_of_Game_of_Thrones_episod
es', 'rootpage-List_of_Game_of_Thrones_episodes', 'skin-vector', 'acti
on-view', 'skin-vector-legacy']}
```

In tako bi lahko nadaljevali dokler ne bi našli naših tabel.

Če želimo najti specifičen tag lahko uporabimo `.find()` metodo. V njej lahko specificiramo ime tag-a katerega iščemo, z `class_` parametrov lahko specificiramo katere **class** vrednosti ima in z `id_` parametrom lahko specificiramo njegov **id** vrednost.

In [16]:

```
table = body.find("table", class_="wikitable plainrowheaders wikiepisodetable")
print(type(table))
print(table.name)
print(table)
```

```
<class 'bs4.element.Tag'>
table
<table class="wikitable plainrowheaders wikiepisodetable" style="wid
th:100%"><tbody><tr style="color:white;text-align:center"><th scope
="col" style="background:#295354;width:5%"><abbr title="Number">No.
</abbr><br/>overall</th><th scope="col" style="background:#295354;wi
dth:5%"><abbr title="Number">No.</abbr> in<br/>season</th><th scope
="col" style="background:#295354;width:23%">Title</th><th scope="co
l" style="background:#295354;width:17%">Directed by</th><th scope="c
ol" style="background:#295354;width:27%">Written by</th><th scope="c
ol" style="background:#295354;width:12%">Original air date <span sty
le="background-color:white;padding:1px;display:inline-block;line-hei
ght:50%"><sup class="reference" id="cite_ref-Futon_20-0"><a href="#c
ite_note-Futon-20">[20]</a></sup></span></th><th scope="col" style
="background:#295354;width:10%">U.S. viewers<br/>(millions)</th></tr
><tr class="vevent" style="text-align:center;background:inherit"><th
id="ep1" rowspan="1" scope="row" style="text-align:center">1</th><td
style="text-align:center">1</td><td class="summary" style="text-alig
n:left">"<a href="/wiki/Winter_Is_Coming" title="Winter Is Coming">W
inter Is Coming</a>"</td><td style="text-align:center"><a href="/wik
```

Če si pogledamo kako je tabela sestavljena vidimo, da tabela vsebuje 1 child tag **tbody**.

tbody nato vsebuje **tr** tag-e, ki predstavljajo vrstice. tr tag vsebuje **th** oziroma **td** tage, ki predstavljajo stolpce in vsebujejo naše iskane vrednosti.

Izluščimo iz tabele prvi 2 vrstici:

In [17]:

```
for i in table.children:
    print(i.name)
```

```
tbody
```

In [18]:

```
tbody = list(table.children)[0]

for row in list(tbody.children)[:2]:
    print(row.name)
    print(row)
    print()
```

```
tr
<tr style="color:white;text-align:center"><th scope="col" style="backg
round:#295354;width:5%"><abbr title="Number">No.</abbr><br/>overall</t
h><th scope="col" style="background:#295354;width:5%"><abbr title="Num
ber">No.</abbr> in<br/>season</th><th scope="col" style="background:#2
95354;width:23%">Title</th><th scope="col" style="background:#295354;w
idth:17%">Directed by</th><th scope="col" style="background:#295354;wi
dth:27%">Written by</th><th scope="col" style="background:#295354;widt
h:12%">Original air date <span style="background-color:white;padding:1
px;display:inline-block;line-height:50%"><sup class="reference" id="ci
te_ref-Futon_20-0"><a href="#cite_note-Futon-20">[20]</a></sup></span>
</th><th scope="col" style="background:#295354;width:10%">U.S. viewers
<br/>(millions)</th></tr>

tr
<tr class="vevent" style="text-align:center;background:inherit"><th id
="ep1" rowspan="1" scope="row" style="text-align:center">1</th><td sty
le="text-align:center">1</td><td class="summary" style="text-align:lef
t">"<a href="/wiki/Winter_Is_Coming" title="Winter Is Coming">Winter I
s Coming</a>"</td><td style="text-align:center"><a href="/wiki/Tim_Van
_Patten" title="Tim Van Patten">Tim Van Patten</a></td><td style="text
-align:center"><a href="/wiki/David_Benioff" title="David Benioff">Dav
id Benioff</a> &amp; <a href="/wiki/D._B._Weiss" title="D. B. Weiss">
D. B. Weiss</a></td><td style="text-align:center">April 17, 2011<span
style="display:none"> (<span class="bday dtstart published updated">20
11-04-17</span>)</span></td><td style="text-align:center">2.22<sup cla
ss="reference" id="cite_ref-21"><a href="#cite_note-21">[21]</a></sup>
</td></tr>
```

In [19]:

```
rows = list(tbody.children)[:2]

for row in rows:
    print(row.name)
    for column in row.children:
        print(column.name, column.text)
    print()
```

```
tr
th No.overall
th No. inseason
th Title
th Directed by
th Written by
th Original air date [20]
th U.S. viewers(millions)

tr
th 1
td 1
td "Winter Is Coming"
td Tim Van Patten
td David Benioff & D. B. Weiss
td April 17, 2011 (2011-04-17)
td 2.22[21]
```

Da najdemo več kot en tag lahko uporabimo metodo `find_all()` .

In [20]:

```
tables = soup.find_all("table", class_="wikitable plainrowheaders wikiepisodetable"
print("Našli smo ",len(tables), "tabel.")
print(tables)
```

```
Našli smo  9 tabel.
[<table class="wikitable plainrowheaders wikiepisodetable" style="wi
dth:100%"><tbody><tr style="color:white;text-align:center"><th scope
="col" style="background:#295354;width:5%"><abbr title="Number">No.
</abbr><br/>overall</th><th scope="col" style="background:#295354;wi
dth:5%"><abbr title="Number">No.</abbr> in<br/>season</th><th scope
="col" style="background:#295354;width:23%">Title</th><th scope="co
l" style="background:#295354;width:17%">Directed by</th><th scope="c
ol" style="background:#295354;width:27%">Written by</th><th scope="c
ol" style="background:#295354;width:12%">Original air date <span sty
le="background-color:white;padding:1px;display:inline-block;line-hei
ght:50%"><sup class="reference" id="cite_ref-Futon_20-0"><a href="#c
ite_note-Futon-20">[20]</a></sup></span></th><th scope="col" style
="background:#295354;width:10%">U.S. viewers<br/>(millions)</th></tr
><tr class="vevent" style="text-align:center;background:inherit"><th
id="ep1" rowspan="1" scope="row" style="text-align:center">1</th><td
style="text-align:center">1</td><td class="summary" style="text-alig
n:left">"<a href="/wiki/Winter_Is_Coming" title="Winter Is Coming">W
inter Is Coming</a>"</td><td style="text-align:center"><a href="/wik
```

In [21]:

```
for table in tables[:]:
    #print(table)
    rows = table.find_all("tr")
    #print(rows)
    for row in rows[:]:
        #print(row)
        tds = row.find_all("td")
        for td in tds[:]:
            print(td.text)

        print()
    print()
```

```

1
"Winter Is Coming"
Tim Van Patten
David Benioff & D. B. Weiss
April 17, 2011 (2011-04-17)
2.22[21]

2
"The Kingsroad"
Tim Van Patten
David Benioff & D. B. Weiss
April 24, 2011 (2011-04-24)
2.20[22]

3
"Lord Snow"
Brian Kirk
David Benioff & D. B. Weiss
```

In [ ]:

Naloga:

Ustvarite skripto, ki pridobi informacije o 250 najbolje ocenjenih filmih.

https://www.imdb.com/chart/top/?ref_=nv_mv_250 (https://www.imdb.com/chart/top/?ref_=nv_mv_250)

Skripta naj pridobi naslov filma, oceno filma in trajanje filma. Trajanje filma dobite, če odprete specifični film.

```
Output:
Kaznilnica odrešitve
9.2
2h 22m

Boter
9.1
2h 55m

Boter, II. del
9.0
3h 22m

Vitez teme
9.0
2h 32m

...
```

In [ ]:

```python
# Rešitev
import requests
from bs4 import BeautifulSoup
url = "https://www.imdb.com/chart/top/?ref_=nv_mv_250"

r = requests.get(url)
soup = BeautifulSoup(r.content, "html.parser")

table = soup.find_all("tbody", class_="lister-list")[0]

trs = table.find_all("tr")
for tr in trs[:10]:
    title_col = tr.find_all(class_="titleColumn")[0]
    a = title_col.find_all("a")[0]
    title = a.text
    print(title)

    rating_col = tr.find_all(class_="ratingColumn imdbRating")[0]
    rating = rating_col.find_all("strong")[0].text
    print(rating)

    href = a["href"]
    #print(a.attrs["href"])
    url = f"https://www.imdb.com{href}"
    #print(url)

    r2 = requests.get(url)
    soup2 = BeautifulSoup(r2.content, "html.parser")
    #print(soup2.html)

    ul = soup2.find_all("ul", class_="ipc-inline-list")
    #print(len(ul))
    #print(ul)
    lis = ul[0].find_all("li")
    li = lis[-1]
    print(li.text)

    print()
```

In [ ]:

```

```

Web Scraping with Selenium

Selenium je orodje, s katerim lahko naš program kontrolira browser (Chrome, Mozzila, ...). Selenium je napisan v večih jezikih (Java, C#, ...) med drugim tudi v Pythonu.

Uporablja se za pisanje avtomatičnih testov za vaše spetne aplikacije oziroma, če je potrebno pridobiti podatke iz bolj zaščitenih spletnih strani oziroma spletnih strani, ki uporabljajo veliko JavaScript-a.

```
pip install selenium
```

Za delovanje potrebujemo še browser driver:

> https://selenium-python.readthedocs.io/installation.html (https://selenium-python.readthedocs.io/installation.html)

1.5. Drivers

> Selenium requires a driver to interface with the chosen browser. Firefox, for example, requires geckodriver, which needs to be installed before the below examples can be run. Make sure it’s in your PATH, e. g., place it in /usr/bin or /usr/local/bin.

> Failure to observe this step will give you an error selenium.common.exceptions.WebDriverException: Message: ‘geckodriver’ executable needs to be in PATH.

> Other supported browsers will have their own drivers available. Links to some of the more popular browser drivers follow.

- Chrome: https://sites.google.com/chromium.org/driver/ (https://sites.google.com/chromium.org/driver/)
- Edge: https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/ (https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/)
- Firefox: https://github.com/mozilla/geckodriver/releases (https://github.com/mozilla/geckodriver/releases)
- Safari: https://webkit.org/blog/6900/webdriver-support-in-safari-10/ (https://webkit.org/blog/6900/webdriver-support-in-safari-10/)

For more information about driver installation, please refer the official documentation.

Tekom naše naloge bomo parsali podatke iz sledeče spletne strani - https://livetoken.co/listings/topshot (https://livetoken.co/listings/topshot)

Na tej spletni strani si lahko pogledamo market z NBA Top Shot Moments - na splošno povedano so izseki iz NBA tekem, katere lahko zbiralci kupujejo in prodajajo.

Naš cilj je ustvariti skripto, ki preveri cene naših momentov.

```
my_portfolio = [
        {"Name": "LUKA DONČIĆ" ,
         "Type": "Assist",
         "Date": "1/17/2021",},

        {"Name": "JAMYCHAL GREEN",
         "Type": "Dunk",
         "Date": "1/3/2021",},

        {"Name": "T.J. MCCONNELL",
         "Type": "Assist",
         "Date": "12/23/2020",},

]
```

In [25]:

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service

my_portfolio = [
        {"Name": "LUKA DONČIĆ" ,
         "Type": "Assist",
         "Date": "1/17/2021",},

        {"Name": "JAMYCHAL GREEN",
         "Type": "Dunk",
         "Date": "1/3/2021",},

        {"Name": "T.J. MCCONNELL",
         "Type": "Assist",
         "Date": "12/23/2020",},

]
print(my_portfolio)

s = Service("./chromedriver_linux_96-0-4664-45")
with webdriver.Chrome(service=s) as driver:
    # uporabi se with, da se driver na koncu lepo samodejno ugasne. Če ne moramo mi
    driver.maximize_window()
    driver.get("https://livetoken.co/listings/topshot")

    input("Press ENTER to quit")
```

```
[{'Name': 'LUKA DONČIĆ', 'Type': 'Assist', 'Date': '17/01/2021'}, {'Na
me': 'JAMYCHAL GREEN', 'Type': 'Dunk', 'Date': '03/01/2021'}, {'Name':
'T.J. MCCONNELL', 'Type': 'Assist', 'Date': '23/12/2020'}]
Press ENTER to quit
```

Finding elements

Sedaj želimo izbrati prvo polje v katerega bomo vnesli ime igralca.

Če si pogledamo stran s pomočjo developer's tools vidimo, da je element sestavljen nekako takole:

```
<input aria-autocomplete="list" aria-labelledby="vs1__combobox" aria-contro
ls="vs1__listbox" type="search" autocomplete="off" class="vs__search">
```

To je element katerega želimo klikniti in vanj vnesti določen string.

V seleniumu izberemo določen element na sledeče načine:

```
find_element_by_id
find_element_by_name
find_element_by_xpath
find_element_by_link_text
find_element_by_partial_link_text
find_element_by_tag_name
find_element_by_class_name
find_element_by_css_selector


# To find multiple elements (these methods will return a list):

find_elements_by_name
find_elements_by_xpath
find_elements_by_link_text
find_elements_by_partial_link_text
find_elements_by_tag_name
find_elements_by_class_name
find_elements_by_css_selector
```

V našem primeru bomo uporabili **css_selector** s katero lahko kar natančno določimo element.

CSS SELECTORS:

> https://www.w3schools.com/cssref/css_selectors.asp (https://www.w3schools.com/cssref/css_selectors.asp)

div.vs__selected-options

> Iščemo element DIV, ki vsebuje class *vs__selected-options*

```
In [ ]:
```

In [26]:

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service

my_portfolio = [
        {"Name": "LUKA DONČIĆ" ,
         "Type": "Assist",
         "Date": "1/17/2021",},

        {"Name": "JAMYCHAL GREEN",
         "Type": "Dunk",
         "Date": "1/3/2021",},

        {"Name": "T.J. MCCONNELL",
         "Type": "Assist",
         "Date": "12/23/2020",},

]
print(my_portfolio)

s = Service("./chromedriver_linux_96-0-4664-45")
with webdriver.Chrome(service=s) as driver:
    # uporabi se with, da se driver na koncu lepo samodejno ugasne. Če ne moramo mi
    driver.maximize_window()
    driver.get("https://livetoken.co/listings/topshot")

    # vvv   HERE vvv
    name_field = driver.find_elements_by_css_selector("div.vs__selected-options")[0
    # DeprecationWarning - to je neki novega.. par mescev nazaj še ni blo tega
    print(name_field)
    # ^^^   HERE   ^^^

    input("Press ENTER to quit")
```

```
[{'Name': 'LUKA DONČIĆ', 'Type': 'Assist', 'Date': '17/01/2021'}, {'Na
me': 'JAMYCHAL GREEN', 'Type': 'Dunk', 'Date': '03/01/2021'}, {'Name':
'T.J. MCCONNELL', 'Type': 'Assist', 'Date': '23/12/2020'}]

<ipython-input-26-af19189275aa>:27: DeprecationWarning: find_elements_
by_* commands are deprecated. Please use find_elements() instead
  name_field = driver.find_elements_by_css_selector("div.vs__selected-
options")[0]

<selenium.webdriver.remote.webelement.WebElement (session="bf7e9e6a452
a52342af89cd9ed3197eb", element="29221be4-8eb1-47e8-9c26-179d602f731
0")>
Press ENTER to quit
```

Sedaj bomo vnesli tekst v to polje s pomočjo `send_keys` metode.

In [27]:

```python
from selenium import webdriver
from selenium.webdriver.chrome.service import Service

my_portfolio = [
        {"Name": "LUKA DONČIĆ" ,
         "Type": "Assist",
         "Date": "1/17/2021",},

        {"Name": "JAMYCHAL GREEN",
         "Type": "Dunk",
         "Date": "1/3/2021",},

        {"Name": "T.J. MCCONNELL",
         "Type": "Assist",
         "Date": "12/23/2020",},

]
print(my_portfolio)

s = Service("/media/balki/E8A255DFA255B334/Mine/Shared_folder/Python/LTFE/LTFE Pyth
with webdriver.Chrome(service=s) as driver:
    # uporabi se with, da se driver na koncu lepo samodejno ugasne. Če ne moramo mi
    driver.maximize_window()
    driver.get("https://livetoken.co/listings/topshot")

    print(f"Checking price for {my_portfolio[0]['Name']}, {my_portfolio[0]['Type']}

    name_field = driver.find_elements_by_css_selector("input.vs__search")[0]
    # DeprecationWarning - to je neki novega.. par mescev nazaj še ni blo tega
    print(name_field)

    # vvv   HERE   vvv
    name_field.send_keys(my_portfolio[0]["Name"])
    # ^^^   HERE   ^^^

    input("Press ENTER to quit")
```

```
[{'Name': 'LUKA DONČIĆ', 'Type': 'Assist', 'Date': '17/01/2021'}, {'Na
me': 'JAMYCHAL GREEN', 'Type': 'Dunk', 'Date': '03/01/2021'}, {'Name':
'T.J. MCCONNELL', 'Type': 'Assist', 'Date': '23/12/2020'}]
Checking price for LUKA DONČIĆ, Assist, 17/01/2021
<selenium.webdriver.remote.webelement.WebElement (session="7a0a698841e
be580973ce02038a8af0e", element="72fb0e8a-3058-4f1e-a728-9228c1d5be6
2")>

<ipython-input-27-76a7da4c6880>:28: DeprecationWarning: find_elements_
by_* commands are deprecated. Please use find_elements() instead
  name_field = driver.find_elements_by_css_selector("input.vs__searc
h")[0]

----------------------------------------------------------------------
-----
ElementNotInteractableException           Traceback (most recent call
 last)
<ipython-input-27-76a7da4c6880> in <module>
     31
     32     # vvv   HERE   vvv
---> 33     name_field.send_keys(my_portfolio[0]["Name"])
```

```
     34     # ^^^   HERE   ^^^
     35

~/anaconda3/lib/python3.8/site-packages/selenium/webdriver/remote/webe
lement.py in send_keys(self, *value)
    537                 value = '\n'.join(remote_files)
    538
--> 539         self._execute(Command.SEND_KEYS_TO_ELEMENT,
    540                       {'text': "".join(keys_to_typing(value)),
    541                        'value': keys_to_typing(value)})

~/anaconda3/lib/python3.8/site-packages/selenium/webdriver/remote/webe
lement.py in _execute(self, command, params)
    691             params = {}
    692         params['id'] = self._id
--> 693         return self._parent.execute(command, params)
    694
    695     def find_element(self, by=By.ID, value=None):

~/anaconda3/lib/python3.8/site-packages/selenium/webdriver/remote/webd
river.py in execute(self, driver_command, params)
    416         response = self.command_executor.execute(driver_comman
d, params)
    417         if response:
--> 418             self.error_handler.check_response(response)
    419             response['value'] = self._unwrap_value(
    420                 response.get('value', None))

~/anaconda3/lib/python3.8/site-packages/selenium/webdriver/remote/erro
rhandler.py in check_response(self, response)
    241                 alert_text = value['alert'].get('text')
    242             raise exception_class(message, screen, stacktrace,
alert_text)  # type: ignore[call-arg]  # mypy is not smart enough here
--> 243         raise exception_class(message, screen, stacktrace)
    244
    245     def _value_or_default(self, obj: Mapping[_KT, _VT], key: _
KT, default: _VT) -> _VT:

ElementNotInteractableException: Message: element not interactable
  (Session info: chrome=96.0.4664.45)
Stacktrace:
#0 0x55886c56fee3 <unknown>
#1 0x55886c03d49f <unknown>
#2 0x55886c06e02e <unknown>
#3 0x55886c06d5ba <unknown>
#4 0x55886c091272 <unknown>
#5 0x55886c068063 <unknown>
#6 0x55886c09137e <unknown>
#7 0x55886c0a43bc <unknown>
#8 0x55886c091163 <unknown>
#9 0x55886c066bfc <unknown>
#10 0x55886c067c05 <unknown>
#11 0x55886c5a1baa <unknown>
#12 0x55886c5b7651 <unknown>
#13 0x55886c5a2b05 <unknown>
#14 0x55886c5b8a68 <unknown>
#15 0x55886c59705f <unknown>
#16 0x55886c5d3818 <unknown>
#17 0x55886c5d3998 <unknown>
#18 0x55886c5eeeed <unknown>
#19 0x7f2aec6ba609 <unknown>
```

Sedaj dobimo **ElementNotInteractableException**. To pomeni, da v element še ne moremo vnašati črk. Ponavadi se stran še nalaga oziroma kakšen drug element stoji v ospredju (na primer gumb, ki čaka da sprejmemo ali zavrnemo piškotke).

Najbolj osnovna rešitev je, da preprosto počakamo še nekaj časa:

In [29]:

```
# vvv   HERE   vvv
import time
# ^^^   HERE   ^^^

from selenium import webdriver
from selenium.webdriver.chrome.service import Service



my_portfolio = [
        {"Name": "LUKA DONČIĆ" ,
         "Type": "Assist",
         "Date": "1/17/2021",},

        {"Name": "JAMYCHAL GREEN",
         "Type": "Dunk",
         "Date": "1/3/2021",},

        {"Name": "T.J. MCCONNELL",
         "Type": "Assist",
         "Date": "12/23/2020",},

]
print(my_portfolio)

s = Service("/media/balki/E8A255DFA255B334/Mine/Shared_folder/Python/LTFE/LTFE Pyth
with webdriver.Chrome(service=s) as driver:
    # uporabi se with, da se driver na koncu lepo samodejno ugasne. Če ne moramo mi
    driver.maximize_window()
    driver.get("https://livetoken.co/listings/topshot")

    name_field = driver.find_elements_by_css_selector("input.vs__search")[0]
    # DeprecationWarning - to je neki novega.. par mescev nazaj še ni blo tega
    print(name_field)

    # vvv   HERE   vvv
    time.sleep(5)
    # ^^^   HERE   ^^^
    name_field.send_keys(my_portfolio[0]["Name"])

    input("Press ENTER to quit")
```

```
[{'Name': 'LUKA DONČIĆ', 'Type': 'Assist', 'Date': '17/01/2021'}, {'Na
me': 'JAMYCHAL GREEN', 'Type': 'Dunk', 'Date': '03/01/2021'}, {'Name':
'T.J. MCCONNELL', 'Type': 'Assist', 'Date': '23/12/2020'}]

<ipython-input-29-4073e8983538>:32: DeprecationWarning: find_elements_
by_* commands are deprecated. Please use find_elements() instead
  name_field = driver.find_elements_by_css_selector("input.vs__searc
h")[0]

<selenium.webdriver.remote.webelement.WebElement (session="472e2c9a078
c4e8d67537f07b2bd62bc", element="595ff5f9-35bf-48e9-86fe-644fb2d130a
5")>
Press ENTER to quit
```

Namesto čakanja lahko v seleniumu določimo specifični vzrok čakanja.

V našem primeru čakamo, da naš element postane "clickable".

In [31]:

```
import time

from selenium import webdriver
from selenium.webdriver.chrome.service import Service

# vvv   HERE   vvv
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
# ^^^   HERE   ^^^

my_portfolio = [
        {"Name": "LUKA DONČIĆ" ,
         "Type": "Assist",
         "Date": "1/17/2021",},

        {"Name": "JAMYCHAL GREEN",
         "Type": "Dunk",
         "Date": "1/3/2021",},

        {"Name": "T.J. MCCONNELL",
         "Type": "Assist",
         "Date": "12/23/2020",},

]
print(my_portfolio)

s = Service("/media/balki/E8A255DFA255B334/Mine/Shared_folder/Python/LTFE/LTFE Pyth
with webdriver.Chrome(service=s) as driver:
    # uporabi se with, da se driver na koncu lepo samodejno ugasne. Če ne moramo mi
    driver.maximize_window()
    driver.get("https://livetoken.co/listings/topshot")

    name_field = driver.find_elements_by_css_selector("input.vs__search")[0]
    # DeprecationWarning - to je neki novega.. par mescev nazaj še ni blo tega
    print(name_field)

    # vvv   HERE   vvv
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable(name_field))
    # ^^^   HERE   ^^^
    name_field.send_keys(my_portfolio[0]["Name"])

    input("Press ENTER to quit")
```

```
[{'Name': 'LUKA DONČIĆ', 'Type': 'Assist', 'Date': '17/01/2021'}, {'Na
me': 'JAMYCHAL GREEN', 'Type': 'Dunk', 'Date': '03/01/2021'}, {'Name':
'T.J. MCCONNELL', 'Type': 'Assist', 'Date': '23/12/2020'}]

<ipython-input-31-677815ef6194>:34: DeprecationWarning: find_elements_
by_* commands are deprecated. Please use find_elements() instead
  name_field = driver.find_elements_by_css_selector("input.vs__searc
h")[0]

<selenium.webdriver.remote.webelement.WebElement (session="22a96f85b37
70145949d1739722af1e1", element="40fe8879-8adc-4953-8413-d016ab8a7bd
7")>
Press ENTER to quit
```

Naš driver bo sedaj eksplicitno počakal 10sekund, da je naš element "clickable". Če naš element ne postane clickable, driver vrže error.

Sedaj moramo klikniti "ENTER" in nato ponoviti postopek še za **All Moments** element.

Privzamemo, da za specifičen datum obstaja le ena vrednost tako, da bomo vpisali le datum.

In [32]:

```
import time

from selenium import webdriver
from selenium.webdriver.chrome.service import Service

from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

# vvv   HERE   vvv
from selenium.webdriver.common.keys import Keys
# ^^^   HERE   ^^^

my_portfolio = [
        {"Name": "LUKA DONČIĆ" ,
         "Type": "Assist",
         "Date": "1/17/2021",},

        {"Name": "JAMYCHAL GREEN",
         "Type": "Dunk",
         "Date": "1/3/2021",},

        {"Name": "T.J. MCCONNELL",
         "Type": "Assist",
         "Date": "12/23/2020",},

]
print(my_portfolio)

s = Service("/media/balki/E8A255DFA255B334/Mine/Shared_folder/Python/LTFE/LTFE Pyth
with webdriver.Chrome(service=s) as driver:
    # uporabi se with, da se driver na koncu lepo samodejno ugasne. Če ne moramo mi
    driver.maximize_window()
    driver.get("https://livetoken.co/listings/topshot")

    name_field = driver.find_elements_by_css_selector("input.vs__search")[0]
    # DeprecationWarning - to je neki novega.. par mescev nazaj še ni blo tega
    print(name_field)

    WebDriverWait(driver, 10).until(EC.element_to_be_clickable(name_field))
    name_field.send_keys(my_portfolio[0]["Name"])

    # vvv   HERE   vvv
    name_field.send_keys(Keys.ENTER)

    all_moments_field = driver.find_elements_by_css_selector("input.vs__search")[2]
    print(all_moments_field)
    #print(all_moments_field.get_attribute("outerHTML"))
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable(all_moments_field))
    all_moments_field.send_keys(my_portfolio[0]["Date"])
    time.sleep(2) # otherwise the text is inputed too fast and then ENTER is presse
    all_moments_field.send_keys(Keys.ENTER)
    # ^^^   HERE   ^^^

    input("Press ENTER to quit")
```

```
[{'Name': 'LUKA DONČIĆ', 'Type': 'Assist', 'Date': '1/17/2021'}, {'N
ame': 'JAMYCHAL GREEN', 'Type': 'Dunk', 'Date': '03/01/2021'}, {'Nam
e': 'T.J. MCCONNELL', 'Type': 'Assist', 'Date': '23/12/2020'}]

<ipython-input-32-2662b50cc0ca>:35: DeprecationWarning: find_elements_
by_* commands are deprecated. Please use find_elements() instead
  name_field = driver.find_elements_by_css_selector("input.vs__searc
h")[0]

<selenium.webdriver.remote.webelement.WebElement (session="7857666c021
c71a0873075084214b841", element="da7a5da8-dfac-417c-938e-e366967d7ae
0")>

<ipython-input-32-2662b50cc0ca>:45: DeprecationWarning: find_elements_
by_* commands are deprecated. Please use find_elements() instead
  all_moments_field = driver.find_elements_by_css_selector("input.vs__
search")[2] # its [2] because 0 is the players name and then 1 is anot
her dropdown which gets hidden if you type in a players name

<selenium.webdriver.remote.webelement.WebElement (session="7857666c021
c71a0873075084214b841", element="8af1001b-32ce-4d1e-aa65-9dab654a555
d")>
<input aria-autocomplete="list" aria-labelledby="vs3__combobox" aria-c
ontrols="vs3__listbox" type="search" autocomplete="off" class="vs__sea
rch">
Press ENTER to quit
```

Sedaj moramo pridobiti informacijo o prvem in drugem najcenejšem momentu in izračunati našo ceno (ki je povprečje teh dveh).

In [48]:

```
import time

from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.keys import Keys

my_portfolio = [
        {"Name": "LUKA DONČIĆ" ,
         "Type": "Assist",
         "Date": "1/17/2021",},

        {"Name": "JAMYCHAL GREEN",
         "Type": "Dunk",
         "Date": "1/3/2021",},

        {"Name": "T.J. MCCONNELL",
         "Type": "Assist",
         "Date": "12/23/2020",},

]
print(my_portfolio)

s = Service("/media/balki/E8A255DFA255B334/Mine/Shared_folder/Python/LTFE/LTFE Pyth
with webdriver.Chrome(service=s) as driver:
    # uporabi se with, da se driver na koncu lepo samodejno ugasne. Če ne moramo mi
    driver.maximize_window()
    driver.get("https://livetoken.co/listings/topshot")

    name_field = driver.find_elements_by_css_selector("input.vs__search")[0]
    # DeprecationWarning - to je neki novega.. par mescev nazaj še ni blo tega
    print(name_field)

    WebDriverWait(driver, 10).until(EC.element_to_be_clickable(name_field))
    name_field.send_keys(my_portfolio[0]["Name"])
    name_field.send_keys(Keys.ENTER)

    all_moments_field = driver.find_elements_by_css_selector("input.vs__search")[2]
    print(all_moments_field)
    #print(all_moments_field.get_attribute("outerHTML"))
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable(all_moments_field))
    all_moments_field.send_keys(my_portfolio[0]["Date"])
    time.sleep(2) # otherwise the text is inputed too fast and then ENTER is presse
    all_moments_field.send_keys(Keys.ENTER)

    # vvv   HERE   vvv
    time.sleep(5)
    prices = driver.find_elements_by_css_selector("div.cost")
    price_1 = prices[0].get_attribute("innerText")
    price_2 = prices[1].get_attribute("innerText")
    print(price_1)
    print(price_2)
    # ^^^   HERE   ^^^

    input("Press ENTER to quit")
```

```
[{'Name': 'LUKA DONČIĆ', 'Type': 'Assist', 'Date': '1/17/2021'}, {'N
ame': 'JAMYCHAL GREEN', 'Type': 'Dunk', 'Date': '03/01/2021'}, {'Nam
e': 'T.J. MCCONNELL', 'Type': 'Assist', 'Date': '23/12/2020'}]

<ipython-input-48-1a9b8ce7fae8>:31: DeprecationWarning: find_elements_
by_* commands are deprecated. Please use find_elements() instead
  name_field = driver.find_elements_by_css_selector("input.vs__searc
h")[0]

<selenium.webdriver.remote.webelement.WebElement (session="d06782ab98c
d8ed6dc85456fe403104b", element="b0e79e68-9b68-46c0-8922-9f11938fdd9
d")>

<ipython-input-48-1a9b8ce7fae8>:39: DeprecationWarning: find_elements_
by_* commands are deprecated. Please use find_elements() instead
  all_moments_field = driver.find_elements_by_css_selector("input.vs__
search")[2] # its [2] because 0 is the players name and then 1 is anot
her dropdown which gets hidden if you type in a players name

<selenium.webdriver.remote.webelement.WebElement (session="d06782ab98c
d8ed6dc85456fe403104b", element="4d518f3a-2169-42bb-a75f-7112ce74c94
3")>

<ipython-input-48-1a9b8ce7fae8>:49: DeprecationWarning: find_elements_
by_* commands are deprecated. Please use find_elements() instead
  rows = driver.find_elements_by_css_selector("div.itemEntryReal")
<ipython-input-48-1a9b8ce7fae8>:58: DeprecationWarning: find_elements_
by_* commands are deprecated. Please use find_elements() instead
  prices = driver.find_elements_by_css_selector("div.cost")

50
<div data-v-8adc142a="" class="cost lowestAsk">$14</div>
<div data-v-8adc142a="" class="cost regularAsk">$14</div>
$14
$14
Press ENTER to quit
```

Text vrednosti imamo, sedaj je potrebno odstraniti $ znak in zadeve pretvoriti v dejanske številske vrednosti in nato izračunati našo prodajno ceno.

In [51]:

```
import time

from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.keys import Keys

my_portfolio = [
        {"Name": "LUKA DONČIĆ" ,
         "Type": "Assist",
         "Date": "1/17/2021",},

        {"Name": "JAMYCHAL GREEN",
         "Type": "Dunk",
         "Date": "1/3/2021",},

        {"Name": "T.J. MCCONNELL",
         "Type": "Assist",
         "Date": "12/23/2020",},

]
print(my_portfolio)

s = Service("/media/balki/E8A255DFA255B334/Mine/Shared_folder/Python/LTFE/LTFE Pyth
with webdriver.Chrome(service=s) as driver:
    # uporabi se with, da se driver na koncu lepo samodejno ugasne. Če ne moramo mi
    driver.maximize_window()
    driver.get("https://livetoken.co/listings/topshot")

    name_field = driver.find_elements_by_css_selector("input.vs__search")[0]
    # DeprecationWarning - to je neki novega.. par mescev nazaj še ni blo tega
    print(name_field)

    WebDriverWait(driver, 10).until(EC.element_to_be_clickable(name_field))
    name_field.send_keys(my_portfolio[0]["Name"])
    name_field.send_keys(Keys.ENTER)

    all_moments_field = driver.find_elements_by_css_selector("input.vs__search")[2]
    print(all_moments_field)
    #print(all_moments_field.get_attribute("outerHTML"))
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable(all_moments_field))
    all_moments_field.send_keys(my_portfolio[0]["Date"])
    time.sleep(2) # otherwise the text is inputed too fast and then ENTER is presse
    all_moments_field.send_keys(Keys.ENTER)

    time.sleep(5)
    prices = driver.find_elements_by_css_selector("div.cost")
    price_1 = prices[0].get_attribute("innerText")
    price_2 = prices[1].get_attribute("innerText")
    print(price_1)
    print(price_2)

    # vvv   HERE   vvv
    price_1 = int(price_1.strip("$"))
    price_2 = int(price_2.strip("$"))
    my_price = (price_1 + price_2) / 2

    print(f"Moja cena za {my_portfolio[0]} je: {my_price}")
    # ^^^   HERE   ^^^

    input("Press ENTER to quit")
```

```
[{'Name': 'LUKA DONČIĆ', 'Type': 'Assist', 'Date': '1/17/2021'}, {'Nam
e': 'JAMYCHAL GREEN', 'Type': 'Dunk', 'Date': '03/01/2021'}, {'Name':
'T.J. MCCONNELL', 'Type': 'Assist', 'Date': '23/12/2020'}]

<ipython-input-51-45dc7b299d6d>:31: DeprecationWarning: find_elements_
by_* commands are deprecated. Please use find_elements() instead
  name_field = driver.find_elements_by_css_selector("input.vs__searc
h")[0]

<selenium.webdriver.remote.webelement.WebElement (session="fdba399b591
1ff9d206a40b728c4fd3d", element="078245ef-2f55-4946-a710-4e0bef87800
b")>

<ipython-input-51-45dc7b299d6d>:39: DeprecationWarning: find_elements_
by_* commands are deprecated. Please use find_elements() instead
  all_moments_field = driver.find_elements_by_css_selector("input.vs__
search")[2] # its [2] because 0 is the players name and then 1 is anot
her dropdown which gets hidden if you type in a players name

<selenium.webdriver.remote.webelement.WebElement (session="fdba399b591
1ff9d206a40b728c4fd3d", element="5cf31ea3-9dd5-4a70-a658-afda1d85787
e")>

<ipython-input-51-45dc7b299d6d>:48: DeprecationWarning: find_elements_
by_* commands are deprecated. Please use find_elements() instead
  prices = driver.find_elements_by_css_selector("div.cost")

$14
$14
Moja cena za {'Name': 'LUKA DONČIĆ', 'Type': 'Assist', 'Date': '1/17/2
021'} je: 14.0
Press ENTER to quit
```

Dodamo zadevo v for loop preverimo delovanje.

In [54]:

```
import time

from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.keys import Keys

my_portfolio = [
        {"Name": "LUKA DONČIĆ" ,
         "Type": "Assist",
         "Date": "1/17/2021",},

        {"Name": "JAMYCHAL GREEN",
         "Type": "Dunk",
         "Date": "1/3/2021",},

        {"Name": "T.J. MCCONNELL",
         "Type": "Assist",
         "Date": "12/23/2020",},

]
print(my_portfolio)

s = Service("/media/balki/E8A255DFA255B334/Mine/Shared_folder/Python/LTFE/LTFE Pyth
with webdriver.Chrome(service=s) as driver:
    # uporabi se with, da se driver na koncu lepo samodejno ugasne. Če ne moramo mi
    driver.maximize_window()
    driver.get("https://livetoken.co/listings/topshot")

    for moment in my_portfolio:

        name_field = driver.find_elements_by_css_selector("input.vs__search")[0]
        # DeprecationWarning - to je neki novega.. par mescev nazaj še ni blo tega
        print(name_field)

        WebDriverWait(driver, 10).until(EC.element_to_be_clickable(name_field))
        name_field.send_keys(moment["Name"])
        name_field.send_keys(Keys.ENTER)

        all_moments_field = driver.find_elements_by_css_selector("input.vs__search"
        print(all_moments_field)
        #print(all_moments_field.get_attribute("outerHTML"))
        WebDriverWait(driver, 10).until(EC.element_to_be_clickable(all_moments_fiel
        all_moments_field.send_keys(moment["Date"])
        time.sleep(2) # otherwise the text is inputed too fast and then ENTER is pr
        all_moments_field.send_keys(Keys.ENTER)

        time.sleep(5)
        prices = driver.find_elements_by_css_selector("div.cost")
        price_1 = prices[0].get_attribute("innerText")
        price_2 = prices[1].get_attribute("innerText")
        print(price_1)
        print(price_2)

        price_1 = int(price_1.strip("$"))
        price_2 = int(price_2.strip("$"))
        my_price = (price_1 + price_2) / 2
        print(f"Moja cena za {moment} je: {my_price}")

    input("Press ENTER to quit")
```

```
[{'Name': 'LUKA DONČIĆ', 'Type': 'Assist', 'Date': '1/17/2021'}, {'Nam
e': 'JAMYCHAL GREEN', 'Type': 'Dunk', 'Date': '1/3/2021'}, {'Name':
'T.J. MCCONNELL', 'Type': 'Assist', 'Date': '12/23/2020'}]

<ipython-input-54-617d68517902>:33: DeprecationWarning: find_elements_
by_* commands are deprecated. Please use find_elements() instead
  name_field = driver.find_elements_by_css_selector("input.vs__searc
h")[0]

<selenium.webdriver.remote.webelement.WebElement (session="e035d7cfa3e
3ecb6250bf10480093f44", element="4bf83659-f8c1-48fd-808c-ffca8eeb0db
5")>

<ipython-input-54-617d68517902>:41: DeprecationWarning: find_elements_
by_* commands are deprecated. Please use find_elements() instead
  all_moments_field = driver.find_elements_by_css_selector("input.vs__
search")[2] # its [2] because 0 is the players name and then 1 is anot
her dropdown which gets hidden if you type in a players name

<selenium.webdriver.remote.webelement.WebElement (session="e035d7cfa3e
3ecb6250bf10480093f44", element="f4b07217-c606-4bae-8370-78a1a813cb0
1")>

<ipython-input-54-617d68517902>:50: DeprecationWarning: find_elements_
by_* commands are deprecated. Please use find_elements() instead
  prices = driver.find_elements_by_css_selector("div.cost")

$14
$14
Moja cena za {'Name': 'LUKA DONČIĆ', 'Type': 'Assist', 'Date': '1/17/2
021'} je: 14.0
<selenium.webdriver.remote.webelement.WebElement (session="e035d7cfa3e
3ecb6250bf10480093f44", element="4bf83659-f8c1-48fd-808c-ffca8eeb0db
5")>
<selenium.webdriver.remote.webelement.WebElement (session="e035d7cfa3e
3ecb6250bf10480093f44", element="f4b07217-c606-4bae-8370-78a1a813cb0
1")>
$8
$8
Moja cena za {'Name': 'JAMYCHAL GREEN', 'Type': 'Dunk', 'Date': '1/3/2
021'} je: 8.0
<selenium.webdriver.remote.webelement.WebElement (session="e035d7cfa3e
3ecb6250bf10480093f44", element="4bf83659-f8c1-48fd-808c-ffca8eeb0db
5")>
<selenium.webdriver.remote.webelement.WebElement (session="e035d7cfa3e
3ecb6250bf10480093f44", element="f4b07217-c606-4bae-8370-78a1a813cb0
1")>
$4
$5
Moja cena za {'Name': 'T.J. MCCONNELL', 'Type': 'Assist', 'Date': '12/
23/2020'} je: 4.5
Press ENTER to quit
```

Za konec bomo še odmaknili vse naše *input()* in zagnali stvar v *headless* načinu, kar pomeni, da se ne no odprlo nobeno okno in bo program deloval "v ozadju".

In [56]:

```
import time

from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.keys import Keys

# vvv   HERE   vvv
from selenium.webdriver.chrome.options import Options
# ^^^   HERE   ^^^


my_portfolio = [
        {"Name": "LUKA DONČIĆ" ,
         "Type": "Assist",
         "Date": "1/17/2021",},

        {"Name": "JAMYCHAL GREEN",
         "Type": "Dunk",
         "Date": "1/3/2021",},

        {"Name": "T.J. MCCONNELL",
         "Type": "Assist",
         "Date": "12/23/2020",},

]
print(my_portfolio)

s = Service("/media/balki/E8A255DFA255B334/Mine/Shared_folder/Python/LTFE/LTFE Pyth
# vvv   HERE   vvv
chrome_options = Options()
chrome_options.add_argument("--headless")
# ^^^   HERE   ^^^

with webdriver.Chrome(service=s, chrome_options=chrome_options) as driver:
    # uporabi se with, da se driver na koncu lepo samodejno ugasne. Če ne moramo mi
    driver.maximize_window()
    driver.get("https://livetoken.co/listings/topshot")

    for moment in my_portfolio:

        name_field = driver.find_elements_by_css_selector("input.vs__search")[0]
        # DeprecationWarning - to je neki novega.. par mescev nazaj še ni blo tega
        print(name_field)

        WebDriverWait(driver, 10).until(EC.element_to_be_clickable(name_field))
        name_field.send_keys(moment["Name"])
        name_field.send_keys(Keys.ENTER)

        all_moments_field = driver.find_elements_by_css_selector("input.vs__search"
        print(all_moments_field)
        #print(all_moments_field.get_attribute("outerHTML"))
        WebDriverWait(driver, 10).until(EC.element_to_be_clickable(all_moments_fiel
        all_moments_field.send_keys(moment["Date"])
        time.sleep(2) # otherwise the text is inputed too fast and then ENTER is pr
        all_moments_field.send_keys(Keys.ENTER)
        time.sleep(5)
        prices = driver.find_elements_by_css_selector("div.cost")
        price_1 = prices[0].get_attribute("innerText")
        price_2 = prices[1].get_attribute("innerText")
        print(price_1)
        print(price_2)

        price_1 = int(price_1.strip("$"))
        price_2 = int(price_2.strip("$"))
        my_price = (price_1 + price_2) / 2

        print(f"Moja cena za {moment} je: {my_price}")
```

```
[{'Name': 'LUKA DONČIĆ', 'Type': 'Assist', 'Date': '1/17/2021'}, {'Nam
e': 'JAMYCHAL GREEN', 'Type': 'Dunk', 'Date': '1/3/2021'}, {'Name':
'T.J. MCCONNELL', 'Type': 'Assist', 'Date': '12/23/2020'}]

<ipython-input-56-921fcd381520>:33: DeprecationWarning: use options in
stead of chrome_options
  with webdriver.Chrome(service=s, chrome_options=chrome_options) as d
river:
<ipython-input-56-921fcd381520>:40: DeprecationWarning: find_elements_
by_* commands are deprecated. Please use find_elements() instead
  name_field = driver.find_elements_by_css_selector("input.vs__searc
h")[0]

<selenium.webdriver.remote.webelement.WebElement (session="58a46a4e4b6
547fc0876282177a018f3", element="893eb9e5-6b6c-427f-bc47-966acacc8ed
9")>

<ipython-input-56-921fcd381520>:48: DeprecationWarning: find_elements_
by_* commands are deprecated. Please use find_elements() instead
  all_moments_field = driver.find_elements_by_css_selector("input.vs__
search")[2] # its [2] because 0 is the players name and then 1 is anot
her dropdown which gets hidden if you type in a players name

<selenium.webdriver.remote.webelement.WebElement (session="58a46a4e4b6
547fc0876282177a018f3", element="cf29c023-5b76-4034-b055-934255b19a8
b")>

<ipython-input-56-921fcd381520>:57: DeprecationWarning: find_elements_
by_* commands are deprecated. Please use find_elements() instead
  prices = driver.find_elements_by_css_selector("div.cost")

$14
$14
Moja cena za {'Name': 'LUKA DONČIĆ', 'Type': 'Assist', 'Date': '1/1
7/2021'} je: 14.0
<selenium.webdriver.remote.webelement.WebElement (session="58a46a4e4
b6547fc0876282177a018f3", element="893eb9e5-6b6c-427f-bc47-966acacc8
ed9")>
<selenium.webdriver.remote.webelement.WebElement (session="58a46a4e4
b6547fc0876282177a018f3", element="cf29c023-5b76-4034-b055-934255b19
a8b")>
$8
$8
Moja cena za {'Name': 'JAMYCHAL GREEN', 'Type': 'Dunk', 'Date': '1/
3/2021'} je: 8.0
<selenium.webdriver.remote.webelement.WebElement (session="58a46a4e4
b6547fc0876282177a018f3", element="893eb9e5-6b6c-427f-bc47-966acacc8
ed9")>
<selenium.webdriver.remote.webelement.WebElement (session="58a46a4e4
b6547fc0876282177a018f3", element="cf29c023-5b76-4034-b055-934255b19
a8b")>
$4
$5
Moja cena za {'Name': 'T.J. MCCONNELL', 'Type': 'Assist', 'Date': '1
2/23/2020'} je: 4.5
```

"Headless" rešitve za Firefox so malo težje:

> https://stackoverflow.com/questions/5370762/how-to-hide-firefox-window-selenium-webdriver (https://stackoverflow.com/questions/5370762/how-to-hide-firefox-window-selenium-webdriver)

In [ ]:

```

```