Web Scraping com Python

```
import time
import requests                    → biblioteca para tratamento de dados
import pandas as pd
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.firefox.options import Options
import json
```

libs

· pq vou optar pelo firefox

Pegar conteúdo HTML a partir da URL

```
url = "                    "

option = Options()
option.headless = True
driver = webdriver.Firefox(options=option)

driver.get(url)

driver.quit()
```

~~options = option~~

↓

posso remover isso

$ python webscraping.py

* Aqui, ele entra e sai apenas ...

Agora vamos manipular o HTML...

> log

time.sleep(10)

fazer o quit pro fim...

Vamos simular um clique agora dentro do elemento

driver.find_element_by_xpath("")

// div [@ class = 'nba-stat-table'] // table // thead // tr // th [@ data-field = 'PTS'] "). click

$ python webscraping.py

// demorou por causa do sleep.

element = driver.find_element_by_xpath ("//div [@ class = 'nba-stat-table'] // table")

html_content = element.get_attribute ("outerHTML")

print (html_content)

$ python ...

ok, trouxe todo o conteúdo HTML

2. Parsear o conteúdo HTML [BeautifulSoup]

↓ lib de análise de HTML e transforma em dado estruturado

```
soup = BeautifulSoup(html_content,
       'html.parser')

table = soup.find(name = 'table')
```

3 Estruturar conteúdo em um Data Frame - Pandas

→ data frame

```
df_full = pd.read_html(str(table))[0].head(10)
```

limpeza em colunas ← limitar em 10

```
df = df_full['Unnamed: 0', 'PLAYER', 'TEAM', 'PTS']]
df.columns = ['pos', 'player', 'team', 'total']

print(df)

driver.quit()
```

$ python ...

4. Transformar os Dados em dicionário de dados próprio

```
top10 ranking = { }
top10 ranking ['points'] = df.to_dict('records')
driver.quit()
```

(df = data frame)

5. Converter e salvar em um arquivo JSON

```
js = json.dumps(top10 ranking)
fp = open('ranking.json', 'w')
fp.write(js)
fp.close()
```

Python...
veria um arquivo json → temos um ranking no formato de dicionário.
