

УНИВЕРЗИТЕТ У БЕОГРАДУ  
МАТЕМАТИЧКИ ФАКУЛТЕТ



Зорана Гајић

САВРЕМЕНИ АЛАТИ ЗА ПРИКУПЉАЊЕ  
ПОДАТАКА СА ВЕБ СТРАНИЦА

мастер рад

Београд, 2023.

**Ментор:**

др Милена ВУЛОШЕВИЋ ЈАНИЧИЋ, ванредни професор  
Универзитет у Београду, Математички факултет

**Чланови комисије:**

др Ана АНИЋ, ванредни професор  
University of Disneyland, Недођија

др Лаза ЛАЗИЋ, доцент  
Универзитет у Београду, Математички факултет

**Датум одбране:** 15. јануар 2016.

*Порогици*

**Наслов мастер рада:** Савремени алати за прикупљање података са Веб страница

**Резиме:**

**Кључне речи:** прикупљање података са веб страница

# Садржај

<b>1</b>	<b>Увод</b>	<b>1</b>
<b>2</b>	<b>Прикупљање података са Веб страница</b>	<b>2</b>
2.1	Изазови . . . . .	2
2.2	Примењене технологије . . . . .	2
<b>3</b>	<b>Преглед алата за прикупљање података са Веб страница</b>	<b>6</b>
3.1	Библиотека BeautifulSoup . . . . .	6
3.2	Библиотека Selenium . . . . .	16
3.3	Библиотека Scrapy . . . . .	16
<b>4</b>	<b>Анализа резултата</b>	<b>17</b>
<b>5</b>	<b>Закључак</b>	<b>18</b>
	<b>Библиографија</b>	<b>19</b>

# Глава 1

## Увод

TODO:

## Глава 2

# Прикупљање података са Веб страница

TODO: Шта означава, процес у глобалу - прикупљање, издвајање и трансформација података, АПИ стругање, разлика између Веб и АПИ стругања

### 2.1 Изазови

TODO: Ботови, CAPTCHA, добијање забране, ИП блокирање, Noneurort замке, динамичан садржај, захтевање пријаве, смернице како избећи блокаде са сајтова

### 2.2 Примењене технологије

TODO: (HTML, Regex, XPath, Тагови, CSS Селектори)

#### Hyper Text Markup Language

TODO:

#### Regex

TODO:

## XPath

XPath (енг. XML Path Language) [1] представља флексибилан начин адресирања на различите делове XML (XML, Extensible Markup Language <sup>1</sup>) документа и због тога може да се користи као начин навигације кроз DOM (DOM, Document Object Model <sup>2</sup>) било ког документа језика сличног XML користећи *XPath израз* (енг. XPathExpression <sup>3</sup>). Садржи преко 200 уграђених функција и дефинисан је од стране WWW конзорцијума. У овом раду ће се XPath користити као начин за одабир елемената са HTML страница.

### XPath синтакса

XPath користи изразе путање за избор чворова у XML документу. Чвор се бира праћењем путање или корака. Корисни примери изрази путања су наведени у наставку:

- `//h2`  
прикупља све *h2* елементе
- `//div//p`  
прикупља све *параграф* (енг. p) елементе унутар блока *div*
- `//ul/li/a`  
прикупља све линкове унутар неуређених листи
- `//ol/li[2]`  
проналази други елемент уређене листе
- `//div/*`  
прикупља све унутар неуређених блока *div*
- `//*[@id="id"]`  
проналази елемент са задатим идентификатором

---

<sup>1</sup>XML представља прошириви (мета) језик за означавање (енгл. markup)

<sup>2</sup>DOM или објектни модел документа представља хијерархијски приказ структуре Веб сајта.

<sup>3</sup>XPath израз дефинише образац за одабир скупа чворова.



- `//*[@class="class"]`  
проналази све елементе са задатом класом
- `//a[@name or @href]`  
проналази све линкове који имају *name* или *href* атрибут
- `//button[contains(text(),"Send")]`  
проналази сву дугмад која садрже текст *Send*
- `//a[last()]`  
проналази последњи линк
- `//table[count(tr)=1]`  
проналази број редова у табели
- `string()`  
`number()`  
`boolean()`  
Типови конверзија
- `contains()`  
`starts-with()`  
`ends-with()`  
функције са нискама
- `//*`  
цви елементи
- `//section[h1[@id='section-name']]`  
унутар *section* елемента пронаћи све *h1* елементе са жељеним идентификатором
- `//a/text()`  
проналази текст линка
- `./a`  
тачка прикупља тренутни чвор  
TODO: Slike sa primerima iz Selenium-a

## **Тагови**

TODO:

## **CSS Селектори**

TODO:

## Глава 3

# Преглед алата за прикупљање података са Веб страница

TODO: Кратак увод шта ће се користити у поглављу, који програмски језик, које библиотеке, опис сајта, жељени циљеви, препреке са којима ћу се сусрести у раду

### 3.1 Библиотека BeautifulSoup

*BeautifulSoup* библиотека [7] представља једну од најједноставнијих за коришћење Пајтон библиотека за имплементацију прикупљања података из HTML и XML текстуалних докумената. Више информација на <https://www.w3.org/TR/xml/>.) датотека преласком кроз DOM. Искључиво ради само статичко прикупљање, које игнорише Јаваскрипт (енг. JavaScript), тј преузима се Веб страница са сервера без помоћи претраживача. За разлику од осталих библиотека које ће се касније у раду разматрати, ова библиотека не може сама да приступи Веб страници, већ јој је неопходна помоћна библиотека. Што значи да јој је главна функција парсирање HTML и XML датотека. Честа је пракса да се динамичан део Веб странице дохвати уз помоћ Selenium пакета [3], а да се парсирање података врши уз помоћ BeautifulSoup пакета.

### Инсталација

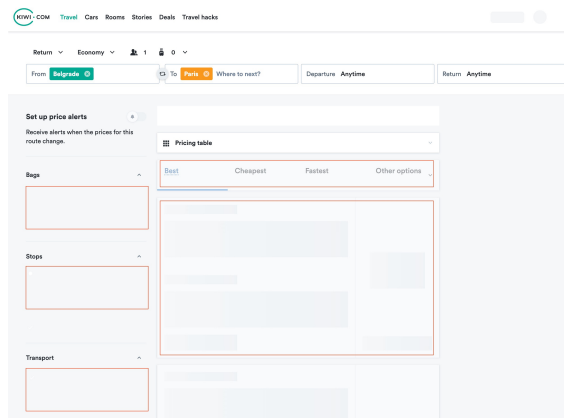
Неопходно је инсталирати *pip3*<sup>1</sup> пакет менаџер и редом *BeautifulSoup*, *requests* [6] и парсер библиотеку *lxml* [4] уз помоћ наведених испод наредби.

```
pip3 install bs4
pip3 install requests
pip3 install lxml
```

Детаљно упутство за инсталацију се може пронаћи у званичној документацији BeautifulSoup библиотеке [7].

### Провера динамичности Веб странице

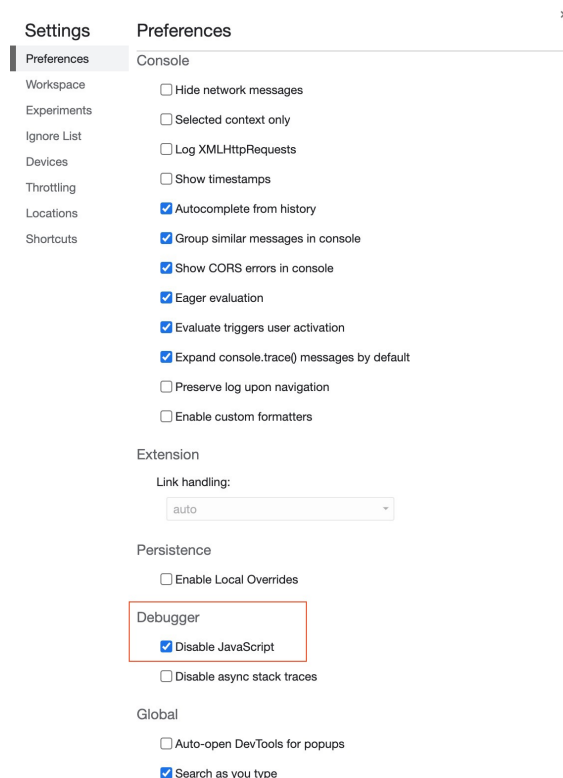
Прва препрека коришћења BeautifulSoup пакета на примеру Киви Веб странице јесте та, да <https://www.kiwi.com/> представља динамичну страницу и да се без Јаваскрипта не може учитати. Самим тим не постоји могућност добијања података конкретно о летовима коришћењем искључиво BeautifulSoup пакета. Изглед Веб странице са угашеним Јаваскриптом је приказан на слици 3.1.



Слика 3.1: Киви веб сајт са угашеним Јаваскриптом

Да би се видело како конкретна Веб страница зависи од Јаваскрипта, у подешавању коришћеног претраживача је неопходно да се угаси опција коришћења Јаваскрипта и како то урадити на примеру *Хром* (енг. Google Chrome) претраживача је приказано на слици 3.2.

<sup>1</sup><https://pip.pypa.io/en/stable/>



Слика 3.2: Подешавање претраживача

## Прикупљање и парсирање HTML

BeautifulSoup није библиотека за скрејповање података са Веба сама по себи. То је библиотека која омогућава да се ефикасно и лако извуче информација из HTML.

Дакле, за почетак, потребан је HTML документ и у ту сврху се може искористити Пайтонов пакет *requests* који омогућава слање *HTTP* (енг. HTTP, Hypertext Transfer Protocol <sup>2</sup>) захтева. Постоји неколико метода од значаја у овом пакету [5]:

- `get(url, params, args)`

Шаље HTTP GET захтев на наведену веб адресу (енг. URL, Uniform Resource Locator))

- `post(url, data, json, args)`

<sup>2</sup>HTTP је мрежни протокол који припада слоју апликације ОСИ референтног модела, представља главни и најчешћи метод преноса информација на веб.

### ГЛАВА 3. ПРЕГЛЕД АЛАТА ЗА ПРИКУПЉАЊЕ ПОДАТАКА СА ВЕБ СТРАНИЦА

---

Шаље HTTP POST захтев на наведену веб адресу

- `put(url, data, args)`

Шаље HTTP PUT захтев на наведену веб адресу

У наставку је приказан код који преузима Веб страницу. Треба имати у виду да при преузимању странице са Веба, може да се догоди да страница не буде пронађена на серверу (или је дошло до грешке у њеном преузимању) или да сервер није пронађен.

```
1 import requests
2 from bs4 import BeautifulSoup
3
4 website = 'https://www.kiwi.com/en/search/results/belgrade-serbia/
    paris-france'
5 try:
6     response = requests.get(website)
7 except requests.exceptions.RequestException as err:
8     print("Error fetching page")
9     exit()
10
11 content = response.text
```

Даље, неопходно је испарсирати добијену HTML страницу тако што се креира *BeautifulSoup* конструктор који за параметре прихвата HTML страницу и парсер.

Пајтон нуди разне библиотеке за парсирање HTML, од којих су две најзаступљеније: *lxml* [4] и *html.parser* [2]. Сваки пројекат има различите захтеве на основу којих се може одабрати најпогоднији парсер, па је за потребе овог рада одабран *lxml* парсер из једног простог разлога - брзина. *lxml* је најбржи парсер HTML страница према званичној документацији *BeautifulSoup* библиотеке [7]. Једна од мана овог парсера јесте екстерна зависност, али у овом случају то не представља проблем.

Наведени код ниже креира *BeautifulSoup* објекат, који омогућава лак приступ многим корисним информацијама, као што је наслов странице, сви линкови на страници, текст са странице...

```
1 soup = BeautifulSoup(content, 'lxml')
2
3 print(soup.title)
4 # <title>Belgrade-Paris trips</title>
```

```
5
6 print(soup.find_all('a'))
7 # [<a class="Logo_Link-sc-1uwlkrd-6 kcwlYv" data-test="Logo" href="/
   en/?destination=paris-france&origin=belgrade-serbia">...]
8
9 print(soup.get_text())
10 # Belgrade Paris trips
11 # TravelCarsRoomsStoriesDealsTravel hacksloadingManage your trips, set
   up price alerts, use...
```

### Циљање DOM елемената

У оквиру пакета BeautifulSoup постоје два метода која омогућавају једноставно и брзо филтрирање HTML странице за проналазак жељених информација [5]:

- `find(tag, attributes, recursive, text, keywords)`

проналази и враћа жељени елемент, где су аргументи редом: назив тага или листа тагова, Пајтон речник атрибута и одговарајућих тагова које садрже било који од тих атрибута, булова вредност која представља да ли се претрага врши по глобалним таговима или и по унутрашњим таговима, подударност на основу текстуалног садржаја ознака, атрибут на основу којих се бира таг

- `findAll(tag, attributes, recursive, text, limit, keywords)`

проналази и враћа листу жељених елемената, где су аргументи слични наведеним изнад са разликом што је додат аргумент лимит, који представља максималан број подударних елемената

Да би приказане информације биле занимљивије, HTML страница ће бити преузета коришћењем Selenium пакета из разлога што Selenium подржава Јаваскрипт. Детаљније о самом пакету Selenium може да се пронађе у наставку рада 3.2.

Одабрана је страница са летовима на релацији Београд-Малага <https://www.kiwi.com/en/search/results/belgrade-serbia/malaga-spain> и коришћењем Selenium пакета је направљен Пајтон скрипт који ће у одвојени фајл експортирати изворни HTML дате странице. У наставку је приказан код

### ГЛАВА 3. ПРЕГЛЕД АЛАТА ЗА ПРИКУПЉАЊЕ ПОДАТАКА СА ВЕБ СТРАНИЦА

---

који најпре затвара прозор за *колачиће* (енг. cookies <sup>3</sup>) који се отвара кад се приступи Веб страници, затим чека 10 секунди да би се страница учитала и цео изворни HTML експортује у одвојени фајл. И овим је решен проблем нединамичности пакета BeautifulSoup.

```
1 import time
2
3 from selenium import webdriver
4 from selenium.common import exceptions
5 from selenium.webdriver.chrome.service import Service
6 from selenium.webdriver.common.by import By
7
8 website = 'https://www.kiwi.com/en/search/results/belgrade-serbia/
           malaga-spain'
9 path = '/usr/local/bin/chromedriver_mac64/chromedriver'
10 service = Service(executable_path=path)
11 driver = webdriver.Chrome(service=service)
12 driver.get(website)
13 driver.maximize_window()
14
15 try:
16     # Close cookies dialog
17     cookie_dialog_close_btn = driver.find_element(By.XPATH, value="//
           section[@id='cookie_consent']//button[@aria-label='Close']")
18     cookie_dialog_close_btn.click()
19
20     time.sleep(10)
21
22     with open('flights-belgrade-malaga-page-source.txt', 'w') as file:
23         file.write(driver.page_source)
24
25     driver.quit()
26
27 except exceptions.StaleElementReferenceException as e:
28     print(e)
29     pass
```

У наставку ће бити приказан комплетан код који извучи информације из необрађеног HTML који је претходно био експортован у фајл: датум поласка, информације о одласку, датум повратка, информације о повратку, број ноћи,

---

<sup>3</sup>Колачић представља текстуалну датотеку која се чува у Веб прегледачу док корисник прегледа неку Веб страницу.



## ГЛАВА 3. ПРЕГЛЕД АЛАТА ЗА ПРИКУПЉАЊЕ ПОДАТАКА СА ВЕБ СТРАНИЦА

цена лета.

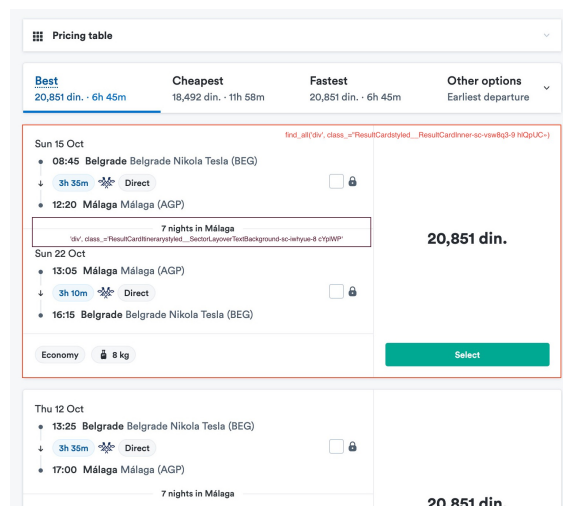
Најпре је неопходно установити како извући целе блокове са информацијама о лету. Када се детаљно прегледа структура HTML, може се приметити да је сваки лет издвојен у *div*<sup>4</sup> са дугачким именом класе што представља једино по чему се тај блок може идентификовати. Пошто на страници има више летова, најпре се користи функција која проналази све инстанце по потпуној подударности имена класе. На исти начин се извршио проналазак броја ноћи касније. 3.3.

```
1 all_flights = soup.find_all('div', class_="
    ResultCardstyled__ResultCardInner-sc-vsw8q3-9 h1QpUC")
```

Осим потпуне подударности по имену класе, елемент се може пронаћи по свом јединственом *id*<sup>5</sup>:

```
1 soup.find(id="specific_id")
```

Сада када су пронађени сви блокови са летовима, могу да се извуку информације о конкретно приказаном лету проласком кроз *for* петљу кроз дате инстанце.



Слика 3.3: Комплетно поклапање класа

```
1 import re
2
3 import pandas as pd
```

<sup>4</sup>Ознака `<div>` дефинише поделу или одељак у HTML документу.

<sup>5</sup>`id` атрибут се користи за одређивање јединственог идентификатора за HTML елемент.

### ГЛАВА 3. ПРЕГЛЕД АЛАТА ЗА ПРИКУПЉАЊЕ ПОДАТАКА СА ВЕБ СТРАНИЦА

---

```
4 from bs4 import BeautifulSoup
5
6 with open('flights-belgrade-malaga-page-source.txt', 'r') as file:
7     soup = BeautifulSoup(file, 'lxml')
8
9     all_flights = soup.find_all('div', class_="
ResultCardstyled__ResultCardInner-sc-vsw8q3-9 h1QpUC")
10
11     departure_date_regex = re.compile('.*DepartureDate.*')
12     departure_info_regex = re.compile('.*
ResultCardItineraryPlacestyled__StyledResultCardItineraryPlace.*')
13
14     flights = []
15     departure_dates = []
16     departure_info = []
17     return_dates = []
18     return_info = []
19     nights = []
20     prices = []
21
22     for flight in all_flights:
23         nights_spent = flight.find('div', class_='
ResultCardItinerarystyled__SectorLayoverTextBackground-sc-iwhyue-8
cYplWP').text.split(' ')[0]
24
25         dates = flight.find_all("p", {"class" : departure_date_regex})
26
27         departure_dates.append(dates[0].find('time').text)
28         return_dates.append(dates[1].find('time').text)
29
30         departure_time = flight.find_all('div', {"class" :
departure_info_regex})[0].find('time').text
31         departure_airport = flight.find_all('div', {"class" :
departure_info_regex})[0].find('div').text
32         departure_info.append(departure_time + ' ' + departure_airport
)
33
34
35         return_time = flight.find_all('div', {"class" :
departure_info_regex})[2].find('time').text
36         return_airport = flight.find_all('div', {"class" :
departure_info_regex})[2].find('div').text
```

## ГЛАВА 3. ПРЕГЛЕД АЛАТА ЗА ПРИКУПЉАЊЕ ПОДАТАКА СА ВЕБ СТРАНИЦА

---

```
37         return_info.append(return_time + ' ' + return_airport)
38
39         nights.append(nights_spent)
40         prices.append(flight.find('strong', {"data-test" : '
ResultCardPrice'})).text)
41
42
43     df_flights = pd.DataFrame({'Departure date': departure_dates, '
Departure info': departure_info, 'Return date': return_dates, '
Return info': return_info, 'Nights' : nights, 'Prices': prices})
44     df_flights.to_csv('flights-belgrade-malaga-results.csv', index=
False)
```

Оно што треба приметити јесте да се блок са информацијама о одласку са Веб сајта идентификовао коришћењем поклапања по регексу (енг. regex).

### Проблем прикупљања података са више страница

TODO: уколико буде превише страница, ова секција се може обрисати јер се пример морао радити коришћењем другог сајта.

У претходно наведеном примеру су вађени подаци из HTML једне Веб странице. Такође је могуће динамично преузети податке и то ће се приказати на примеру Веб страница које не зависе од Јаваскрипта, коришћењем BeautifulSoup и requests пакета. У питању је Веб страница која садржи транскрипте филмова. Да би се успешно извршило преузимање података кроз више страница, треба да се разуме на који је начин то постигнуто на конкретном Веб сајту.

На веб адреси [https://subslikescript.com/movies\\_letter-A](https://subslikescript.com/movies_letter-A) уколико се изврши навигација на следећу страницу, може да се примети да се додаје и мења параметар *страница* (енг. page) у веб адреси и онда она изгледа овако [https://subslikescript.com/movies\\_letter-A?page=2](https://subslikescript.com/movies_letter-A?page=2). Наведени код ниже са претходно стеченим знањем, проналази блок са нумерацијом страница, извлачи укупан број страница, пролази кроз сваку од њих, поставља вредност изворног HTML на нову тако што креира нову Веб адресу и уз помоћ requests пакета извлачи HTML нове странице, креира нови *BeautifulSoup* објекат за парсирање, проналази све линкове ка филмовима на датој страници, прелази на конкретан линк и преузима транскрипт филма. Кључни моменат јесте да се сваки пут иницијализује *BeautifulSoup* објекат за парсирање.

### ГЛАВА 3. ПРЕГЛЕД АЛАТА ЗА ПРИКУПЉАЊЕ ПОДАТАКА СА ВЕБ СТРАНИЦА

---

```
1 import requests
2 from bs4 import BeautifulSoup
3
4 root = 'https://subslikescript.com'
5 website = f'{root}/movies_letter-A'
6 result = requests.get(website)
7 content = result.text
8 soup = BeautifulSoup(content, 'lxml')
9
10 # pagination
11 pagination = soup.find('ul', class_='pagination')
12 pages = pagination.find_all('li', class_='page-item')
13 last_page = pages[-2].text
14
15 links = []
16
17 for page in range(1, int(last_page) + 1):
18     result = requests.get(f'{website}?page={page}')
19     content = result.text
20     soup = BeautifulSoup(content, 'lxml')
21
22     box = soup.find('article', class_='main-article')
23
24     for link in box.find_all('a', href=True):
25         links.append(link['href'])
26
27     for link in links:
28         try:
29             result = requests.get(f'{root}/{link}')
30             content = result.text
31             soup = BeautifulSoup(content, 'lxml')
32             box = soup.find('article', class_='main-article')
33
34             title = box.find('h1').get_text()
35             transcript = box.find('div', class_='full-script').
get_text(strip=True, separator=' ')
36
37             print(link)
38             with open(f'movies/{title}.txt', 'w') as file:
39                 file.write(transcript)
40
41         except:
```

42 `print('Link not working')`

## 3.2 Библиотека Selenium

TODO: Инсталација, сајтови покренути са JS, драјвери, лоцирање елемената, пагинација, имплицитно и експлицитно чекање, попуњавање формулара, логин, кретање између страница, скрејповање странице са бесконачним скролом, како променити корисничког агента и зашто

## 3.3 Библиотека Scrapy

TODO: Инсталација, scrapy шаблони, креирање паука, стругање са више линкова, стругање са више страница, стругање АПИ, попуњавање формулара, логин, како променити корисничког агента, најосновније потребне функције LUA програмског језика (неопходно за SPLASH), SPLASH, шта представља SPLASH, зашто је неопходан, како превазићи Captcha)

## Глава 4

# Анализа резултата

TODO: Поређење перформанси, дијаграм односа времена стругања сајта и коришћене библиотеке, како оценити добијене резултате, табела са поређеним карактеристикама коришћених библиотека, препоруке, смернице за унапређење коришћених технологија

## Глава 5

## Закључак

TODO:

# Библиографија

- [1] Keio) 1999 W3C® (MIT, INRIA. XPath. on-line at: <https://www.w3.org/TR/1999/REC-xpath-19991116/>.
- [2] Python Software Foundation 2001-2023. `html.parser` — Simple HTML and XHTML parser. on-line at: <https://docs.python.org/3/library/html.parser.html>.
- [3] 2023 Software Freedom Conservancy. Selenium. on-line at: <https://www.selenium.dev/documentation/>.
- [4] Stefan Behnel Martijn Faassen. Parsing XML and HTML with `lxml`. on-line at: <https://lxml.de/parsing.html>.
- [5] Ryan Mitchell. In *Web Scraping with Python*, 2015.
- [6] A Kenneth Reitz P. Requests: HTTP for Humans. on-line at: <https://requests.readthedocs.io/en/latest/>.
- [7] Leonard Richardson. BeautifulSoup Documentation, 2004-2023. on-line at: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>.



# Биографија аутора

Зорана Гајић, рођена 04.11.1997 у Москви, где је ... TODO: