

WebScraping

Anno Accademico 2022-2023

Docente: Laura Ricci

Lezione 14

WebScarping in Python:

Form, Fields e Cookies

14 Marzo 2023

Strumenti avanzati per il web scraping

- cosa abbiamo visto nelle lezioni precedenti
 - **HTTP, HTML, CSS**
- le tecniche viste possono essere utilizzate sotto certe condizioni
 - la pagina **HTML** che cerchiamo è accessibile senza autenticazione
 - il contenuto della pagina è **statico**
 - il server non utilizza misure anti-scraping.
- cosa manca per implementare un vero web-scrafer?
 - gestione di web forms per siti che richiedono autenticazione
 - gestione di cookies
 - capire le misure opportune affinché lo scraper **looks like a human**
 - gestire il contenuto dinamico generato da **JavaScript**
- alcune di queste tecniche possono essere implementate direttamente in **BeautifulSoup**
- la gestione di contenuti gestiti da **Java Script** richiede strumenti più avanzati, come **Selenium**

Interagire con il web server: la libreria Python Request

- come interagire con il protocollo **HTTP** da **Python**
 - tramite la libreria **Request**
 - cosa abbiamo visto fino nelle lezioni precedenti: il metodo **GET** della libreria **Requests**
 - la libreria, come vedremo, offre molte altre funzionalità
 - possibilità di sottomettere **form** al server mediante **POST**
 - gestione degli header **HTTP**
 - gestione dei **cookies**

```
In [3]: import requests
import json
url = 'https://www.pythonscraping.com/pages/page1.html'
r = requests.get(url)
# Qual è il codice HTTP restituito da HTML?
print(r.status_code)
print()
# Qual è la spiegazione testuale di quel codice?
print(r.reason)
print()
# Quali sono i campi header contenuti nella risposta inviata dal server?
print(r.headers)
print()
# Che tipo ha "r.request", cioè la richiesta inviata al server?
print(r.request)
print()
# Quali erano gli header HTTP della richiesta?
print(r.request.headers)
print()
# IL contenuto della risposta HTTP:
print(r.text)
```

200

OK

```
{'Server': 'nginx', 'Date': 'Thu, 16 Mar 2023 18:31:16 GMT', 'Content-Type': 'text/html', 'Content-Length': '361', 'Connection': 'keep-alive', 'X-Accel-Version': '0.01', 'Last-Modified': 'Sat, 09 Jun 2018 19:15:58 GMT', 'ETag': '"234-56e3a58a63780-gzip"', 'Accept-Ranges': 'bytes', 'Vary': 'Accept-Encoding', 'Content-Encoding': 'gzip', 'X-Powered-By': 'PleskLin'}
```

<PreparedRequest [GET]>

```
{'User-Agent': 'python-requests/2.28.1', 'Accept-Encoding': 'gzip, deflate', 'Accept': '*/*', 'Connection': 'keep-alive'}
```

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

```
<title>A Useful Page</title>
</head>
<body>
<h1>An Interesting Title</h1>
<div>
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor i
ncidunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute i
rure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui offic
ia deserunt mollit anim id est laborum.
</div>
</body>
</html>
```

Interagire con un web server: query strings

- **HTTP** consente non solo di scaricare contenuti dal server, ma anche di interagire con il server inviandogli dei dati
- il modo più semplice per inviare i dati a un server è includere i dati direttamente nella **URL**
- **query strings**: parte opzionale della **URL** inserita dopo il **?**

`http://www.example.com/product_page.html?product_id=304`

`https://www.google.com/search?
dcr=0&source=hp&q=test&oq=test`

- quando il server riceve una richiesta **HTTP** con una query string, è in grado di "interpretare i parametri" ed effettuare le operazioni corrispondenti
- di seguito presentiamo due modi diversi per accedere agli episodi di **Game of Thrones**, nel secondo si sottomette un a query **WikiGameOfThrones**

`https://en.wikipedia.org/wiki/List_of_Game_of_Thrones_episode`

WikiGameOfThroneswithQuery

`https://en.wikipedia.org/w/index.php?
title=List_of_Game_of_Thrones_episodes`

Form HTML e richieste HTTP

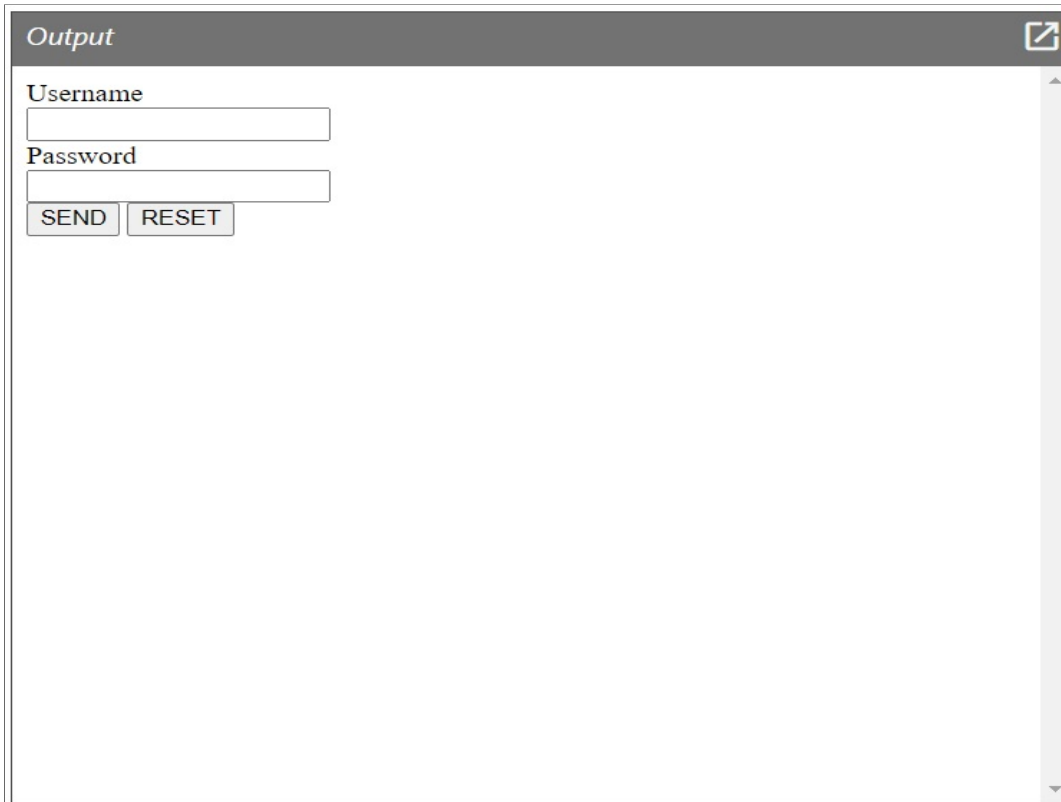
- anche se la dimensione di una **URL** è illimitata, inviare parametri tramite una **URL** diventa difficile, quando le query diventano più complesse
 - ad esempio: comprare tickets per un concerto, invinando nome, e-mail, scelta del ticket, etc.
- in questo caso si possono usare le **web forms**
- una **HTML Form (Web Form)** permette l'interazione tra l'utente e il browser web
 - è incorporata nella pagina **HTML** mediante i tag

```
<form>
```

```
</form>
```

- contiene elementi che vengono mostrati all'utente per l'immissione di informazioni da inviare al server
 - **input fields**
 - **buttons**
 - **checkboxes**
 - ...
- sul web server, un programma colleziona questi dati e restituisce una risposta calcolata dinamicamente sulla base dei dati sottomessi.

Un esempio di form HTML: il rendering



The image shows a browser window titled "Output" with a dark header bar. Inside the window, there is an HTML form. The form contains two text input fields: the first is labeled "Username" and the second is labeled "Password". Below the "Password" field, there are two buttons: "SEND" and "RESET". The "SEND" button is on the left and the "RESET" button is on the right. The form is rendered in a simple, clean style with a light gray background. A vertical scrollbar is visible on the right side of the window, indicating that the form content is scrollable.

Un esempio di form HTML: il sorgente



```
1 <!DOCTYPE html>
2 <!-- HtmlFormEx1.html -->
3 <html lang="en">
4   <head>
5     <meta charset="utf-8">
6     <title>Basic HTML Form Structure</title>
7   </head>
8   <body>
9     <form action="submit.php" method="get">
10       <label for="username">Username</label>
11       <br>
12       <!-- "for" targets "id" -->
13       <input type="text" id="username" name="username">
14       <br>
15       <!-- "name" for "name=value" pair -->
16       <label>Password <br>
17         <input type="password" name="password">
18       </label>
19       <br>
20       <input type="submit" value="SEND">
21       <input type="reset" value="RESET">
22     </form>
23   </body>
24 </html>
```

Ln: 24 Col: 7 size: 687 B

Un esempio di form HTML

- racchiusa tra i tag

```
<form>
```

```
</form>
```

- la form presenta due attributi importanti

- **action**

- indica la locazione, sul server, dove i dati della form dovranno essere sottomessi
 - nel caso della form

`submit.php`

* in generale "#" indica la stessa **URL** della form di Login

- **method**

- specifica il metodo **HTTP** (**GET** oppure **POST**) usato per la sottomissione dei dati immessi dall'utente
- importante ricordare queste informazioni da utilizzare per gestire le form tramite la libreria **request**

Un esempio di form HTML

- racchiusa tra i tag

```
<form>  
</form>
```

- attributi importanti
 - una serie di **input tag**
 - per ogni **input tag**, un attributo importante è **name**
- quando l'utente/il programma di scraping sottomette la form, il browser colleziona i valori dei parametri della form
 - i parametri devono essere sottomessi nella form **name=value**
 - ci deve essere corrispondenza tra il nome utilizzato per inviare il parametro al server e il nome dell'attributo individuato nell'elemento `< input >`
 - il valore è quello inserito in input dall'utente
- nella form precedente
 - **<input type="text">**: un campo testo per l'immissione dello Username
 - **<input type="password">**: un campo usato per l'immissione della passwords, visualizzata con asterischi (non visibile)

Un esempio di form HTML: buttons

`<input type="submit">`: Submit button.

* per inviare la form

`<input type="reset">`:

* per resettare tutti i campi della form

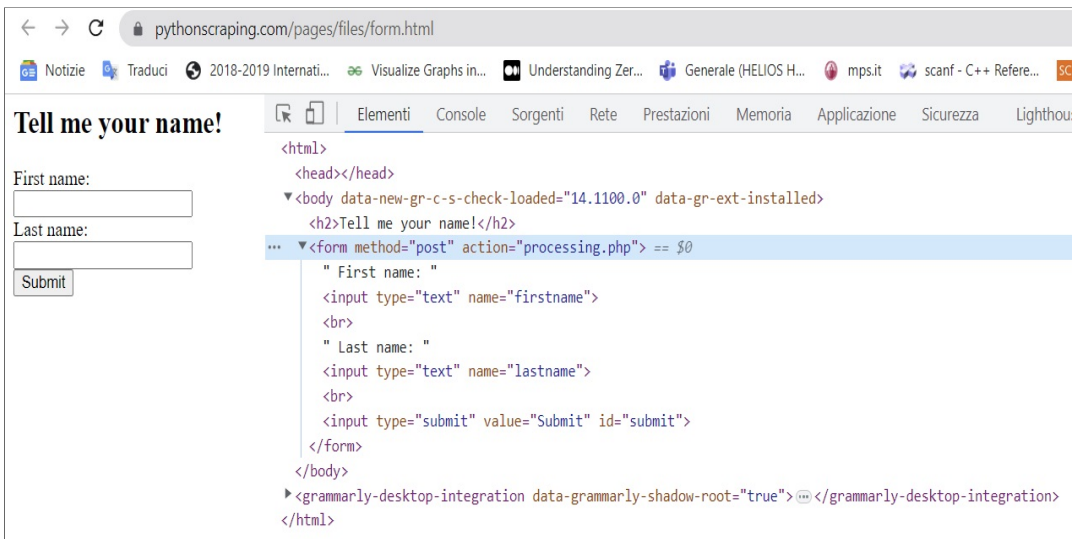
- l'attributo dei button indicano le etichette da visualizzare sul button stesso

Ispezionare form scaricate da web

- scaricare la pagina dal sito

<http://pythonscraping.com/pages/files/form.html>

- usare la funzione **"ispeziona"** del browser, come per le pagine web statiche, per individuare i campi della form



Ispezionare la form

- i campi da tenere d'occhio per scrivere un programma che sottometta la form in modo automatico?
- il campo **metodo**
 - indica il metodo utilizzato per sottomettere la **form**
 - nel caso della form precedente, non si utilizza il metodo di default, che è il metodo **GET**, ma il metodo utilizzato è **POST**
 - viene inviata una richiesta **HTTP POST**
 - il valore dei parametri viene incluso nel body della richiesta
 - invece di includere il valore dei parametri nella **URL** di un metodo **GET**
- la **action** che indica **dove** andrà sottomessa la form
- il nome degli attributi di input
 - **firstname, lastname**

Sottomettere form con la libreria Request

```
In [9]: import requests

url= "http://pythonscraping.com/pages/processing.php"

# primo passo: richiedere la pagina **GET**
r = requests.get(url)

# eseguire quindi una **POST**
formdata = {'firstname': 'Laura',
            'lastname': 'Ricci'}

r = requests.post(url, data=formdata)
print(r.text)
```

Hello there, !

- gli argomenti da includere nel payload sono passati in un **dizionario** Python
- la prima **GET** potrebbe essere eliminata, ma...
 - un browser reperirebbe prima la pagina con una **GET** e dopo effettuerebbe una **POST**
 - stesso comportamento per evitare misure anti-scraping

Strumenti di monitoring

- usare il tab "Strumenti per sviluppatori**" di Chrome per monitorare la trasmissione
- verificare che, nel momento in cui si preme **Submit**, effettivamente è stata inviata una richiesta **POST**

The screenshot shows the Chrome DevTools Network tab. The top bar indicates the page content is "Hello there, laura ricci!". The Network tab is active, showing a list of requests. The first request, "processing.php", is selected. The details panel for this request is expanded, showing the following information:

- Generali**
 - URL di richiesta: <https://pythonscraping.com/pages/files/processing.php>
 - Metodo di richiesta: POST
 - Codice di stato: 200
 - Indirizzo remoto: 142.4.205.1:443
 - Criterio relativo al referrer: strict-origin-when-cross-origin
- Intestazioni della risposta**
 - content-encoding: gzip
 - content-length: 45
 - content-type: text/html; charset=UTF-8
 - date: Sun, 12 Mar 2023 16:24:31 GMT
 - server: nginx
 - vary: Accept-Encoding
 - x-powered-by: PHP/7.4.33
 - x-powered-by: PleskLin
- Intestazioni delle richieste**
 - :authority: pythonscraping.com
 - :method: POST
 - :path: /pages/files/processing.php
 - :scheme: https

Request headers

- fino a questo momento gli **HTML Header** da inviare al server sono stati generati automaticamente dalla libreria **Request**

```
In [7]: import requests

url= "http://pythonscraping.com/pages/processing.php"
# eseguire per prima una richiesta **GET**
r = requests.get(url)
print(r.request.headers)
```

```
{'User-Agent': 'python-requests/2.28.1', 'Accept-Encoding': 'gzip, deflate', 'Accept': '*/*', 'Connection': 'keep-alive'}
```


Modificare il campo header

- considerare il seguente frammento di codice:

```
In [10]: url = 'http://www.webscrapingfordatascience.com/usercheck/'
         r = requests.get(url)
         print(r.text)
         # Attenzione: il server ha capito che stiamo facendo scraping!
         print(r.request.headers)
```

It seems you are using a scraper!

```
{'User-Agent': 'python-requests/2.28.1', 'Accept-Encoding': 'gzip, deflate', 'Accept': '*/*', 'Connection': 'keep-alive'}
```

- in questo caso il server risponde con il messaggio **It seems you are using a scraper**
- se apriamo la stessa pagina da un browser questo non accade
- che cosa sta accadendo?
 - la libreria **requests** ha inserito, in modo corretto, l'header '**User-Agent': 'python-requests/2.28.1'**
 - se il server prevede qualche meccanismo di **anti-scraping**, può inserire un semplice controllo per verificare che il richiedente non è un browser e bloccare l'accesso
 - occorre modificare l'header nello script di scraping.

Looking like a human!

```
In [ ]: import requests

url = 'http://www.webscrapingfordatascience.com/usercheck/'
my_headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 ' + ' (k
r = requests.get(url, headers=my_headers)
print(r.text)
print(r.request.headers)
```

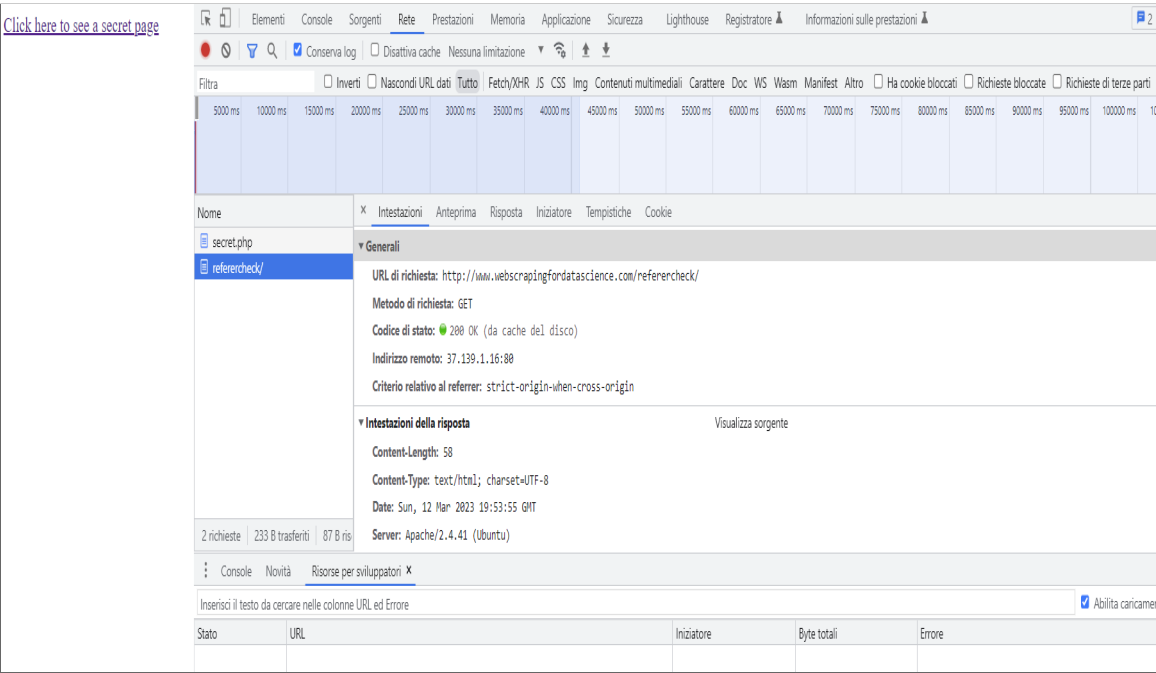
- impostare un header in modo da comportarsi come un browser

Referral headers

- collegarsi alla pagina

http://www.webscrapingfordatascience.com/referercheck/

* la pagina presenta il link ad una pagina **secret page** * si può notare il **referral header** settato a **strict-origin-when-cross-origin** * significa che le pagine linkate devono essere accedute solo attraverso questa pagina * se si accede alla **secret page** direttamente, ad esempio aprendo un nuovo tag del browser ed indicando la **URL** della pagina, verrà restituito un codice di errore * visualizzare questo header usando lo strumento per sviluppatori del browser



Referral headers

- cosa accade se accedo direttamente alla pagina segreta

```
In [12]: import requests
         url = 'http://www.webscrapingfordatascience.com/referercheck/secret.php'
         r = requests.get(url)
         print(r.text)
```

Sorry, you seem to come from another web page

- un modo per accedere direttamente alla **pagina segreta** è modificare direttamente l'header

```
In [13]: import requests
         url = 'http://www.webscrapingfordatascience.com/referercheck/secret.php'
         my_headers = {
             'Referer': 'http://www.webscrapingfordatascience.com/referercheck/'
         }
         r = requests.get(url, headers=my_headers)
         print(r.text)
```

This is a totally secret page

Gestione dei cookies

- **HTTP** è nato come un protocollo **stateless**
 - il server non mantiene alcuna informazione sulle richieste precedenti inviate dallo stesso client
 - per introdurre una forma semplice di **stato** è stato introdotto il meccanismo dei **cookies**
- **cookie**
 - un dato di piccole dimensioni (pochi kBytes) inviato da un server ad un browser e memorizzato nel browser
 - il browser automaticamente include il cookie in tutte le successive richieste che fa al medesimo server
 - in questo modo il server può identificare in modo univoco un client (un browser)
- nome **cookie**: un "opaque piece of data held by an intermediary**"
 - il dato è di interesse solo per il server e non per il client

Cookies e scraping

```
In [14]: import requests
         url = 'http://www.webscrapingfordatascience.com/cookie/login/'
         # Eseguo una post
         # mi loggo sul sito
         r = requests.post(url, data={'username': 'Laura', 'password': 'Ricci'})
         r = requests.get(url + 'secret.php')
         print(r.text)
```

Hmm... it seems you are not logged in

Cookies e scraping

```
In [15]: import requests
         url = 'http://www.webscrapingfordatascience.com/cookie/login/'
         # Eseguo una post
         # mi loggo sul sito
         r = requests.post(url, data={'username': 'Laura', 'password': 'Ricci'})
         # Accedo ai cookies ricevuti

         receivedcookies=r.cookies
         print("questo è il cookie ricevuto")
         print(receivedcookies)

         # r.cookies is a RequestsCookieJar object which can also
         # be accessed like a dictionary.

         my_cookies = r.cookies

         print("going to the secret page")
         r = requests.get(url + 'secret.php', cookies=my_cookies)
         print(r.text)
```

```
questo è il cookie ricevuto
<RequestsCookieJar[<Cookie PHPSESSID=krjcg5k3tp49n9r06ct9v6am5i for www.webscrapi
ngfordatascience.com/>]>
going to the secret page
This is a secret code: 1234
```

Gestire i cookies con la funzione Session

- aprendo una sessione **Session**, in **Request**, tutta la gestione dei cookies è effettuata automaticamente dalla libreria

```
In [18]: import requests
         url = 'http://www.webscrapingfordatascience.com/cookie/login/'
my_session = requests.Session()
r = my_session.post(url, data={'username': 'Laura', 'password': 'Ricci'})
r = my_session.get(url + 'secret.php')
print(r.text)
```

This is a secret code: 1234

```
In [19]: import requests
         url = 'http://www.webscrapingfordatascience.com/cookie/login/'
# Eseguo una post e mi loggo sul sito
r = requests.post(url, data={'username': 'Laura', 'password': 'Ricci'})

# Accedo ai cookies ricevuti
receivedcookies=r.cookies
print("questo è il cookie ricevuto")
print(receivedcookies)

# rimando il cookie che ho ricevuto al server
my_cookies = r.cookies
print("going to the secret page")
r = requests.get(url + 'secret.php', cookies=my_cookies)
print(r.text)
```

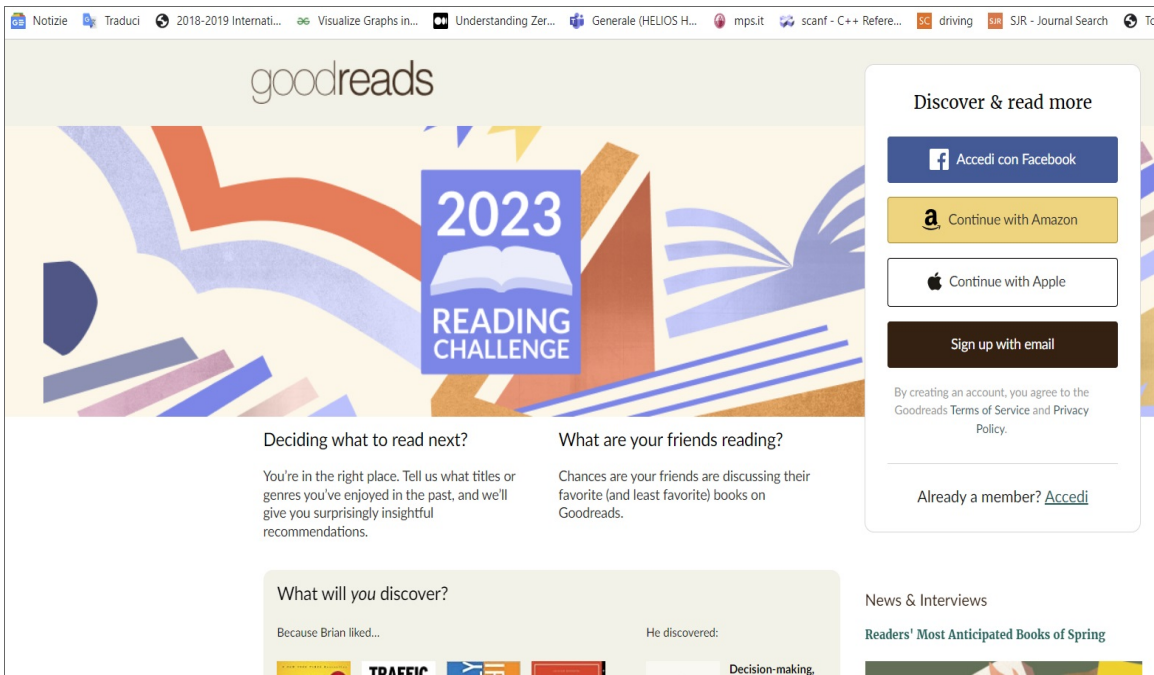
```
questo è il cookie ricevuto
<RequestsCookieJar[<Cookie PHPSESSID=gurqfsibmfs6tk6r761ehmcbqc for www.webscrapi
ngfordatascience.com/>]>
going to the secret page
This is a secret code: 1234
```


Campi hidden di forms

```
<input type="hidden">
```

- alcuni campi hanno tipo **hidden** e sono contenuti nelle **form HTML**
- permettono di rendere il valore contenuto in questo campo visibile dal browser, ma non all'utente (a meno che non si analizzi il codice sorgente)
- sono spesso usati come strumenti anti-scraping
 - in un **hidden field** viene inserito un **valore random** da parte di un server
 - se il valore non è inserito nella form che il client sottomette, è probabile che quella form sia stata postata direttamente al server, senza averla scaricata precedentemente
 - rimandare invariati i campi **hidden** al server
- come comportarsi?
 - effettuare il caricamento della pagina contenente la form
 - individuare i valori hidden random nella form scaricata
 - inserire quei valori nella **POST** successiva

Goodreads: un social network dedicato ai lettori

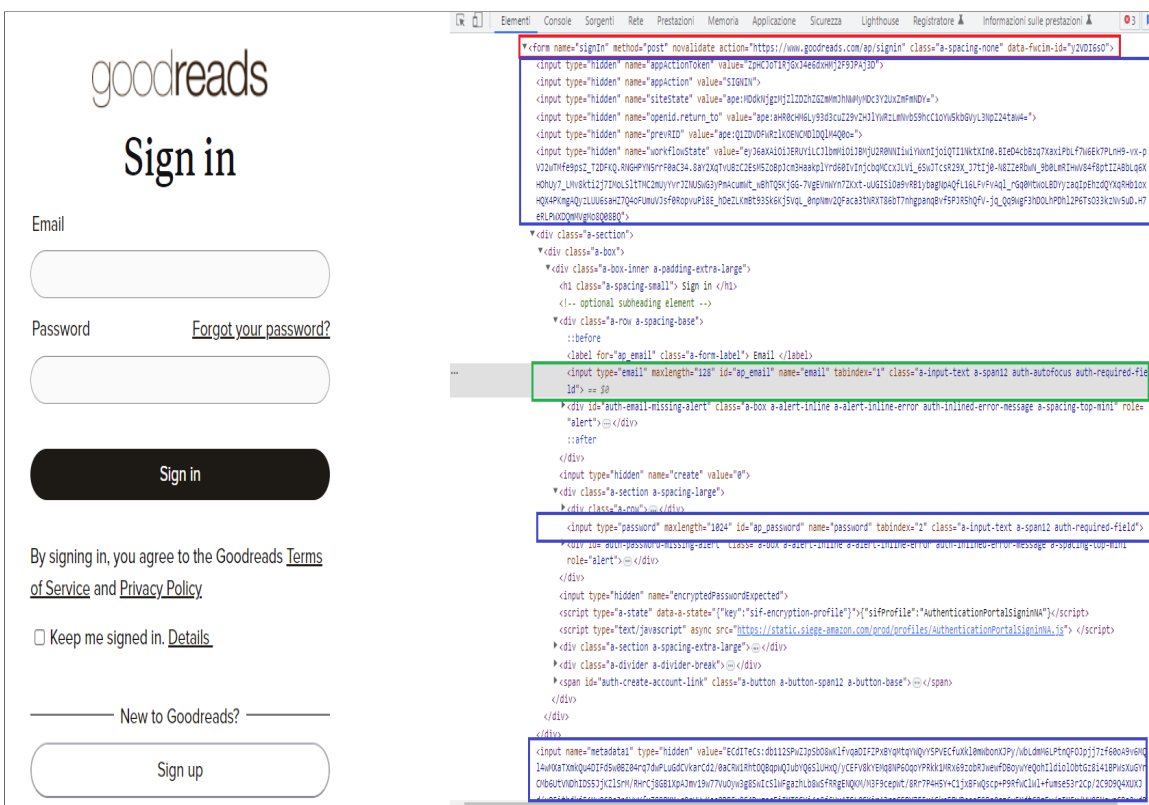


Goodreads: un social network dedicato ai lettori

- unisce funzionalità tipiche dei social networks con funzionalità specifiche dedicate ai lettori
- gli utenti
 - aggiungono ai loro profili i libri che hanno letto
 - condividono recensioni, commenti, suggerimenti
 - gruppi di discussione
- perchè scegliamo **Goodreads** per lo scraping?
 - richiede una fase di registrazione per accedere alle recensioni
 - struttura delle pagine non banale, ma può essere analizzata con gli strumenti visti fino a questo momento

Goodreads: la fase di registrazione

- analizzare la form utilizzata dal sito per la registrazione



Goodreads: la fase di autenticazione

creazione del payload della POST con i parametri richiesti dalla form

```

payload = {
# parametri pubblici

    'email': "laura.ricci@unipi.it",

    'password': ".....",

# parametri hidden

    'utf8': '✓',
    'appActionToken' :
"YUAOBJBmIpJwPj2BSAionpwFb92j2Bgj3D",

    'appAction': 'SIGNIN',

'siteState': 'ape:MTM0NTdmNzc5NDA2ZjE1MjE1MjVjYTNmZTU0NWMM0YjQ=',

'openid.return_to': 'ape:aHR0cHM6Ly93d3cuZ29vZHZJYWRzLmNvbS9hcC1c
    'prevRID': 'ape:RjFOUkFRNEQyVFFNVFZCWk5FMjE=',

    "workflowState"
:"eyJ6aXAiOiJERUYiLCJlbnMiOiJBMjU2R0NNIiwiaWwiYXNjaXoiQTI1NktXIn0.-
    XRkcaD6v0-
h7fU1XALXzRu4MyYr82rnNe3cvXXnw77Yn6DBb0ICZA.dML9-
x8kob38v2kc.YLPbkkx1GjtH0Df_VXh56g0v60UZJZauvImEq05Z

oUF1rgXxTkm0ndqfwtNXijZGuW6dsdCtcZptdHIapuXfa0WGiBYH9u8nsnnu6IZ3

```

TLIRm2kXC2Uw38Ue0zTIgZ7yRoVLkxQv6_KJ3bjBOW08fk1msY6JwE3Q51IrnHac

FldwzXIJxvJ22AJnmdp-

5MEflcrkxg4tz9sebq1vRiWiWqtBbphprzaY0nLXaUnbxD_M30at1gMekTO43wMG

BF87GKGrEI92ccOpNRq.7RwK_3C4dHpbecUvxnDU1w",

"metadata1" :

"ECdITeCs:hb2Yw0Q36NS1HePFtpW211u7G3s4qV5ThEU5

xWONHBI/JXLcYZ1ibaxiuqoVV25y9WcM5TevtQxCGTvttyonLRhuZYbLD5nIuhWjc

1kRtMzZ//Ss+Q0QtPR1lf8WrUVs9usDgfvYRfy1JxmunzxabwKdx1jm32pG7v7Vj

Yr/yTLXu20sHuRhzbior/trmg/Xymqs2kblg+XEd1doYfWTyHfnVjuN17kPA1foC

7UwZfIGkT4f9+qZSmDMx8bYSarvo+ZG6gnPI7icYKzYHTBQAsQtG4C1bQwvAPwbj

o1QgXTkvTyV+h5LE1WW487aGr+HQgTU8/QC0xQ0FPjLV38D6ddm3Dr1/63bttZTF

5btT99nN/RkdJe/i0BR0ricfhzrPwSc+WiXgCpuNNK6YDWKiWjPCmJra3FJs+vTs

bog9TzyINGqGOS/7jYR0N//6tbpXVAXw2JWE8Y9beDISnHLM25kSXsjvdcblC/G7

n5cLHHAmW8sDaDEWaq18xM6qJsgym8hdkif4ehUy.....

}

apertura della sessione per la gestione dei cookies

session = requests.Session()

LOGIN_URL = "https://www.goodreads.com/user/sign_in"

looking like a human!

session.headers = {'User-Agent': 'Mozilla/5.0'}

response = session.get(LOGIN_URL)

print(response)

```
print(f"attempting to log in as {email}")  
p = session.post(LOGIN_URL, data=payload)  
print("risposta dopo la post")  
print(response)
```

Goodreads: la fase di scraping

- una volta autenticati, è possibile visualizzare le review effettuate da ogni utente
- ogni utente è identificato da un identificatore unico
- l'identificatore dell'utente è incorporato nella **URL** della sua pagina personale e può essere ricavato da una analisi della **URL**
- si vogliono collezionare le prime 5 review effettuate da un utente il cui codice è dato in input
 - queste review si trovano sicuramente nella prima pagina associata all'utente
 - anche il numero della pagina è incorporato nella **URL**
 - un esempio nella pagina successiva

Goodreads: la fase di scraping

- **identificatore=66** corrisponde all'utente **Sean**
- analizzare la struttura della **URL** corrispondente alla prima pagina di quell'utente

<https://www.goodreads.com/review/list/66-sean?page=1&shelf=read>

The screenshot shows the Goodreads website interface. The browser address bar displays the URL: `goodreads.com/review/list/66-sean?page=1&shelf=read`. The page header includes the Goodreads logo, navigation links (Home, My Books, Browse, Community), a search bar, and links for Sign In and Join. The main content area is titled "Sean > Books: Read (70)" and features a sidebar with bookshelves (All (76), Read (70), Currently Reading (3), Want to Read (3)) and reading stats. The main table lists books with the following columns: cover, title, author, rating, my rating, read, and added. The table contains four rows of book data.

cover	title	author	rating	my rating	read	added
	Americanah	Adichie, Chimamanda Ngozi	4.32	★★★★★	not set	Jan 27, 2020
	Educated	Westover, Tara *	4.47	★★★★★	not set	Jan 27, 2020
	Noir	Moore, Christopher *	3.79	★★★★★	not set	Jan 27, 2020
	The Spy and the Traitor: The Greatest Espionage Story of the Cold War	Macintyre, Ben	4.51	★★★★★	not set	Jan 27, 2020

Goodreads: la fase di scraping, le reviews






- una tabella, ogni riga una review
- ogni review identificata da un identificatore del tipo **review_id**

Discover Spring's Biggest Books >
Sign In

[My Books](#)
[Browse ▾](#)
[Community ▾](#)

Sean > Books: Read (70) ×

Search and add books

Bookshelves	cover	title	author	rating	rate
All (76)					
Read (70)					
Currently Reading (3)					
Want to Read (3)					
<hr/>					
Reading stats					
<hr/>					
		Americanah	Adichie, Chimamanda	4.32	★
		tr#review_3163840949.booklike.r		748.4 × 98	
		view			
		Educated	Westover, Tara *	4.47	★
		Noir	Moore, Christopher *	3.79	★
		The Spy and the Traitor: The Greatest Espionage Story of the Cold War	Macintyre, Ben	4.51	★
		The Descendants	Hemmings,	3.83	★

```

<!-- CTRIFSP#mainContent -->
<div class=mainContentFloat >
  <div id=flashContainer></div>
  <div id=leadercol></div>
  <div id=columContainer class=myBooksPage>
    <div id=leftcol class=col reviewLeft></div>
    <div id=rightcol class=last col>
      <div id=shelfSettings class=controlBody style=display:none></div>
      <div class=js-dataTootlip data-use=otr.tootlip=true>
        <table id=books class=table stacked border=0>
          <thead></thead>
          <tbody id=booktbody>
            <tr id=review_3163841551 class=booklike review></tr>
            <tr id=review_3163840949 class=booklike review></tr>
            <tr id=review_3163840464 class=booklike review></tr>
            <tr id=review_3163838062 class=booklike review></tr>
            <tr id=review_3163833777 class=booklike review></tr>
            <tr id=review_3163448227 class=booklike review></tr>
            <tr id=review_67895914 class=booklike review></tr>
            <tr id=review_67894888 class=booklike review></tr>
            <tr id=review_766014 class=booklike review></tr>
            <tr id=review_649054 class=booklike review></tr>
            <tr id=review_649782 class=booklike review></tr>
            <tr id=review_649782 class=booklike review></tr>
            <tr id=review_63959 class=booklike review></tr>
            <tr id=review_67137 class=booklike review></tr>
            <tr id=review_67807 class=booklike review></tr>
            <tr id=review_67847 class=booklike review></tr>
            <tr id=review_66969 class=booklike review></tr>
            <tr id=review_66969 class=booklike review></tr>
          </tbody>
        </table>
      </div>
    </div>
  </div>
</div>

```

Stato	URL	Iniziatore	Byte totali	Errori
operazione riusc.	https://sg-assets.com/assets/text_cil_/internal_dependencies-2e2b907af.js.map	https://sg-assets.com/...	13 871 851	
operazione riusc.	https://sg-assets.com/assets/text_client_side/site_hesce-07ee155cd.js.map	https://sg-assets.com/...	2 882 427	
errore	https://amazon-adsystem.com/aaaj/apstag_pu.js.map	https://amazon-adsys-		Errore HTTP: codice di stato 403, netERR:

Goodreads: fase di scraping

- definiamo una funzione di supporto che data la pagina contenente le review, restituisce gli identificatori di tutte le reviews
 - individuare, nella pagina, ogni elemento di tipo **tr** (table row) con classe **review**
- consideriamo solo le prime 5 reviews, usando l'operatore di slicing sulla lista restituita
- per ogni identificatore nella lista risultante, eliminiamo il prefisso **_review** così isolando solo l'identificatore della review

```
def parse_review_list(html):  
    soup = BeautifulSoup(html, "html.parser")  
    rows = soup.find_all('tr', class_='review')  
    rows = rows[:5]  
    return [row.get('id').replace('review_', '') for row in rows]
```

Goodreads: fase di scraping

- il seguente gragmento di codice accede alla pagina dell'utente, invoca la precedente funzione **parse_review_list** e, ottenuti gli identificatori di ogni review, parse la review corrispondente con la funzione **parse_review**

```
# supponiamo che target_user_id identifichi l'utente di interesse
page_num=1
review_ids = set()
# costruzione dell URL che dientifica la pagina dell'utente
REVIEW_LIST_URL = "https://www.goodreads.com/review/list/{}?
view=reviews&shelf=read&page={}"
print(REVIEW_LIST_URL.format(target_user_id, page_num))
response = session.get(REVIEW_LIST_URL.format(target_user_id, page_num))
# la funzione parse_review_list
# è quella che restituisce la lista degli identificatori delle reviews
# costruzione dell URL che identifica ogni singola review
REVIEW_URL = "https://www.goodreads.com/review/show/{}"
for id in parse_review_list(response.text):
    print("parsing review id {}: {}".format(id, REVIEW_URL.format(id)))
    review_ids.add(id)
    response = session.get(REVIEW_URL.format(id))
# parse_review effettua il parsing della singola review
r = parse_review(response.text)
print(r)
r['review_id'] = id
writer.writerow(r)
session.close()
```

Approfondimento: formatting Strings in Python

- nella costruzione delle **URL** è stata usata la funzione **FORMAT** di Python per la formattazione delle stringhe

```
In [20]: txt1 = "My name is {fname}, I'm {age}".format(fname = "Mary", age = 36)
          txt2 = "My name is {0}, I'm {1}".format("Mary",36)
          txt3 = "My name is {}, I'm {}".format("Mary",36)

          print(txt1)
          print(txt2)
          print(txt3)
```

```
My name is Mary, I'm 36
My name is Mary, I'm 36
My name is Mary, I'm 36
```

```
In [21]: print("{0} love {1}!!".format("I", "Science"))
          print("{1} loves {0}!!".format("Science", "Everybody"))
          print("{2} is the most widely {3} after {1} {0}".format("language", "programming", "Python", "sought"))
```

```
I love Science!!
Everybody loves Science!!
Python is the most widely sought after programming language
```

Reperire le review di un utente su goodReads: inizializzazioni

```
In [22]: """
        Script per reperire le review effettuate da un certo utente dal sito GoodReads
        Le review sono scritte su un file CSV
        Parametri in inout
        - email: Email address per accedere a Goodreads
        - password: Password per accedere a Goodreads
        - target_user_id: User ID dell'utente che vogliamo analizzare
        """

from bs4 import BeautifulSoup
from csv import DictWriter
from datetime import datetime
import requests
```

Reperire le review di un utente su goodReads: funzione di supporto

```
In [23]: def parse_review_list(html):  
         soup = BeautifulSoup(html, "html.parser")  
         rows = soup.find_all('tr', class_='review')  
         rows = rows[:5]  
         return [row.get('id').replace('review_', '') for row in rows]
```

Reperire le review di un utente su goodReads: scraping delle review

```
In [24]: def parse_review(html):
    soup = BeautifulSoup(html, "html.parser")
    title = soup.find('a', class_='bookTitle').get_text()
    author = soup.find('a', class_='authorName').get_text()
    rating = soup.find('meta', itemprop='ratingValue').get('content')
    text = soup.find('div', class_='reviewText')
    timeline = soup.find_all('div', class_='readingTimeline__text')

    review_date = soup.find('span', itemprop='datePublished').get_text()
    if review_date:
        review_date = datetime.strptime(review_date.strip(), '%b %d, %Y').strftime('%Y-%m-%d')
    else:
        review_date = None

    for div in timeline[::-1]:
        if div.get_text():
            date, _, context = div.get_text().strip().partition('\n')

            if 'Finished Reading' in context:
                last_finished_date = datetime.strptime(date.strip().replace(' ', '0'), '%B %d, %Y').
                break
            else:
                last_finished_date = None

    return {
        'title': title.strip() if title else None,
        'author': author.strip() if author else None,
        'review_date': review_date,
        'last_finished_date': last_finished_date,
        'rating': int(rating) if rating else None,
        'text': text.get_text().strip() if text else None
    }
```


Reperire le review di un utente su goodReads

```
In [ ]: def authenticateAndScrape(email, password, target_user_id):
    payload = {
        'email': "laura.ricci@unipi.it",
        'password': "gmV91ale",
        'utf8': '&#x2713;',
        'appActionToken' : "YUA0BJBmIpJwPj2BSAionpwFb92j2Bgj3D",
        'appAction': 'SIGNIN',
        'siteState': 'ape:MTM0NTdmNzc5NDA2ZjE1MjE1MjVjYTNmZTU0NWMyQjQ=',
        'openid.return_to': 'ape:aHR0cHM6Ly93d3cuZ29vZHZlYWRzLmNvbS9hcC1oYw5kbGVyL3NpZ24taW4=',
        'prevRID': 'ape:RjFOUkFRNEQyVFFNVFZCk5FMjE=',
        "workflowState" : "eyJ6aXAiOiJERUYiLCJlbmMiOiJBMjU2R0NNIiwiaWxnIjojQTI1NktXIn0.-\
XRkcaD6v0-h7fU1XALXzRu4MyYr82rnNe3cvXXnw77Yn6DBb0ICZA.dML9-x8kob38v2kc.YLPbkx1GjtH0Df_VXh\
oUF1rgXxTkm0ndqfwtNXijZGuW6dsdCtcZptdHiapuXfaOWGiBYH9u8nsnnu6IZ3smhU9CT0YUXuoWktJjUPAM1EPy\
TLIRm2kXC2Uw38Ue0zTIgZ7yRoVLkxQv6_KJ3bjB0W08fklmsY6JwE3Q51IrnHaqto3X30TSMWGP0pHAzu7watbXPC\
FlDwXIJxvJ22AJnmdp-5MEflcrkxg4tz9sebj1vRiWiWqtBbphprzaY0nLXaUnbxD_M30at1gMekTO43WMGiAAAsA\
BF87GKGrEI92ccOpNRq.7RwK_3C4dHpbecUvxnDU1w",
        "metadata1" : "ECdITeCs:hb2Yw0Q36NS1HePFtpW211u7G3s4qV5ThEU5\
xW0NHBI/JXLcYZ1ibaxiuqoVV25y9WcM5TevtQxCGTvtYonLRhuZYbLD5nIuhWjoFb8+Js3Q15URYfCnqcg5k0/ZMT\
1kRtMzZ//Ss+Q0QtPR1lf8WrUVs9usDgfvYRfy1JxmunnxawKdx1jm32pG7v7VjijZwPwn9i3DPsz2/wRgbrz7QiZ\
Yr/yTLXu20sHuRhzbior/trmg/Xymqs2kblg+XEd1doYfWtyHfnVjuN17kPA1foObd/1dElyxsucgasR6g3tegFlc3\
7UwZfIGkT4f9+qZSmDMx8bYSarvo+ZG6gnPI7icYKzYHTBQAsQtG4C1bQwvAPwbjBjrixs9zsEVooP6LytmkrHLKL8\
o1QgXTkvTyV+h5LE1WW487aGr+HQGTU8/QC0xQ0FPjLV38D6ddm3Dr1/63bttZTFkLDwOmElafRBJ32LpuODck9GIC\
5btT99nN/RkdJe/i0BRoricfhzrPwSc+WiXgCpuNNK6YDwKiWjPCmJra3FJs+vTs4J5z8QGFJ3fOvijI96i7K4msZz\
bog9TzyINGGOS/7jYR0N//6tbpXVAXw2JWE8Y9beDISnHLM25kSXsjvdcblC/G7gUtBjNjXgZHEX11NRn1P1YX5WS\
n5cLHHAmW8sDaDEWaq18xM6qJsgym8hdkif4ehUyAiS1IQk4u0Jc81WU+fk0CYj+1c+n75Sk5y0Ha3jBD9WKH1Mc03\
CJRhsVYNur7Fape4BapT3YnhYh79uDvi5LPw7W1cp2WvcAI735cG+c+UlwF/64rpu3B/ClnjFwoc8iR1TjsjiGD6Z\
QhEJ28DQhKYkCrvp+wyPqeBmwp9+mKcecSKDu2Yq5ZDuzRNBxE2dVieXfmpfhsY8RJFR8gyy2LuGU0J6Kd211b6r+E\
bdKpCAijomy3qr0NMxIme08/L/P3hlyE5+8GawJ/Sivmy4CZyAXrL6TgzbI9/kDmQowPeTgWRVw8ET6gYd4FLRii3v\
B81JNnMVCg6Gt4LaJA0yKZj3BwGoArdK3FTYrB7h/+Vv1/j26LdBdCCMyj+1P5ZhYzdSB57bTgiDFGQE0UrtcgUdnk\
6TB7821Z9YjiIIRZ8oRKPzmKZRYcutMH9Wks70GrRtV61Wc0oYPD7112QcW0VyRawr0EohFBQLbs3r5EIrJy74Xaj\
JX4SVdSVKU3el+RSbzCdJhuwPwnCFHbJnPRN4aZ7rk1W9/wPPahYtgj2cdZX/4P+/wFT5TMK9iWAPi9vQS2qb/ugBC\
CC0Uxttu4kJKnuAXg/SYSSwCTYZdBM77o0KJKNo7AS5bIPro/vP9N8itbBpSqCu0Gz/qhJWZnPl8ucOQuqY9K/k6\
+uLp3i07xTATXWoglCtvtntQC/RQX8IicMSBZ498R2oorRsnPRCxfEePCOJEwISR6fSv6KonMAfoU6TK/bBdeXuG8/\
DAXBMX1IRHasECM9KRfKkaiAKVMwMK5Ytpzn8F6jsPiWhDPvWntno6RTJp/Sqp/ocVCuOGxOQf76uMJ5RPer0XgMeC\
k8sfvXSMtWZuHR1oZerhdXtnTdv1tQiAaD2UBm14DNHvmjRMxUrpBRq9VKfVL8P0955MXfdv1kgHOKGUJdFXpoe\
lfrnTZ/W3E7R1hngPONYwhLRtCWkuALFhxQK5f1Z0C7KG09R49DvmJdwyXPA5+yhV+9MTbzS0LK5Pvi7xd05bybC\
oYMnu/Ge9139wr/Hk4uTw7m2+dQAngHMEewtzyymDGAxNeLCQjWYnRFfDRLg5jufLJ3E4A2BEKdy7iGiFFJX7FAMj\
I9aliscUv2Byqke2j6yYzh/o66WMMOkEcXVie67jnlb7b99ZhwREj3VLtYJKfKcgMat0XU106YHRHs3AemK29Bi/FN\
3Js+XAk0BERCX9XIkxWOZCjJQUsFR0XPzZ1X6r0kKc2JjNP+ihej358PRXIOFxZiW+pq7ByZTdfFnUc0uqKXChOar\
byR1AC4Bevx50x0FvdMELsTBSN51FhTuvj8bAiZVj7D1o77TPKuR5Gw65hetwBB+fpOdXFmwYTEowZBvZMaoXX1E5c\
7ooQOpNjxP5Fb4THPAQkrcZYLRQsi3n1UyqTrRrq+z0zNGHm2ddPQA46+e2Ep3AGelJDD/h54a3QxXwNPK3d3PH8F4\
JVCkQGcef9MIVja+9bMU9hp+Nv8E1o2eAlQ3EooyQzTP89Vyj3nVsdFns45ZTNAMJcTSqKBYJpuxYtNzuyUqutoYoT\
LQ7539t1N0t2bo5/mnfyXypZq/dBJCQHbEQCvPvKrDAESkr1THoH3YbNUH1wB/GWrtMHM8F22vXpcZcqL17Bqs0LRC\
/uB9ziF2HRw7imGxSi/iokld17B2hYHN6CIOU7Y2vLf/m1GeE4DySBScn+x5tqXee8SziQpjGRE+z/EfwqWRpD4DC\
1t2wGauc7xlt3nieCDCkMupna5liWGcdlyThm/PF3IZB+c6p24Ja9gqKQzVNO63dj/+pC94pIfcm1FLQ9ddJLC6Tty\
87dIoZg0CfJSGcAGrxLaV8WcWt5zsbduotrxIgr4N94pRkNNMxCA02+VmHgdUesvmxc8W+hEvg1VfkV0+Q3gzoIHjC\
L4931jRMK7G1P37+o5pq0YJNJpQXGxc0wCpHKR4WB2Ycwy0icI6XBF2oSE2JcTt6uu320sJx/o7DpQX0rOYhgiZKE\
YXMMwiWyJTvxzRfFWJ78pejwiGN/QNQ/2VzTXGbg5ZbUffgLqVAcNh+ln/4Kvzp4qj2pX4D1DY1hlBc2gwY0FTxP2c\
WF7tQHcR0NJXJjTq3ZA4L1V5A2TE2s8yZzNyKs+vhEZPaPjKrcij+C0a2GI/hl6/3dapjgE1Z1PZRNAWHwXdsyYT2r\
B6TE9PMIR4Um13+dPs4N9vX8ddzPL96k+DX9TEHBEiFLpn1o1sGaIVTObHbp95+gKXA0daUluomIvCyLk3LDyDNkf\
P7MBKw+nubebVi+mZ+Bu0IYZyZjJhm/vgRTFhxy4kldbxN2NlrKZA0lysSJZ2+wCt3/zPueixL0X2Hm5LRN8XFUqv\
hXHuVUVfRIPzcf8IB0wuSXwj6MFnthl0tvUGJF7Rkos+IuKZFpiowUPeH84xHXj+scgS05IbYx5FCNR6RafEoPudFp\
KVGou5j4MNa0Cs8puEaL/Cqyj0+k4it/wC05shLqEYGGLd6I/BcZjCsLmMTyetZ1fAiYGITMTftjDhx+ot+twUrUPTs\
UYile3Z4pRha8opTT+UGBOjex/+Mag6mc0Jcol1zzvPQxydUNZVY67mbvkcS430zd3Zep1xXwma1H3XikBL1I7qoq\
HRTgy8WvD4+H8BmeDZISNHyzrM8up3dWcuOrNOZp0AYcrE786YofE1k2p528TbP130fey3fCKyaZUAH1xPOOVvS31L\
YqNT5qTJe/DCNyRlXOPiWAwk07ii9gu3GPTVe4B8igDxKKYQcbIx4FMHahjj3GNsxhw/UyPPXsS+OvwG3scQdwAgeC
```

```

JukK9Mcoc/eK203j/Lncksh5dWJJ8P0Ab1Yp7i+nFUIHTCwdPZ7Uufkv5/ijtoyfHYK4iWQKhm4sWcrUe+hNW3yC
JqMwrRha0C6Wj58F+JyDC20CQ19uJN0BnfaxITzKzFbuyLP4/6LhW6YmjP/TVkhxz8idsLIUBAC5ryxy1p3Y9+yds/
C4vruJznRwc6aEKvFgkx9+MbVfctLCffhkpGt9HEfgbtm50nfPj7xn1gC0KzFFfTzJTuvfS9NGqJb285LCJrvXYTT3
JBNEtb14+SUIjgteK1VKZ4m76leCjQpb0BCaMdQXLRyry31zGwrIMdaGGh4aQxuD3kNm8pGPiLISyPRSBPYX8c6G8T
k/0t/dQq5ajgFccUde9ZyFCKu4jDQmzJAi/eJp7vNywp+rHnCenoj7t05PTLHEuEqYVSRUFpY5BKt/YPGUog5WEH
GGIuLgHhW+53Hxc89vRtBNzb00Cer4K8gOyP4RtSZR5ccVHQbZZM4T16Qh8XyeqeSDFffyfGehrbas0Ym8qpjQIa2x
6y5ixDU1zVq6CSkGVkBVSIwDCWtm9tgOjmdmQ9JnnyvE3Bm9+oNDYYOwcSQ+0cqyDY58SmhEhxiiH4KXeQ5cvxyv9n
0/N6uOyOPoStC2geH6iza/epoQBUyMysOPsbnLYNhtxALX5M87C18/KTt7yatSIiqqaWpRiVwldJ+8Kd3Fw2Wo+3Z
GTu21k6N+lonNpVbM0kFN9WiUU/M6t+HATGV+3KiBVxs3GYfVXKv1cTjS1bdHbc7zHAuaK4CiJ4vZV2D9j+ozm6wAE
HmXzjbS1AGYC1yF5VDSSaVMouIHkRyjOWhguOD+ppeQhnwdrAnLaURTN4zekbk1fR8XMnZAXb5kb9iTHa4f9Db79S
yGyCa447/ZzeoEvnSB8FVux0o3EC5x+ZEKjAZ5tx05PD/Lz0QekGjQyFwvN7/prm9k1kk9tb8ohWuX8ya33io5LY7c
HZbo7ZFwKRdU74qful4ABroDvD82Gf7zrvcsA3SrN9Ph4XBCdAHls+fJxu2PFVAhRm7c4q4mFhtqexKRVg2TotSc/C
4reyk8PROL0t5rsVFnu6nWn6hXAF6wVQzW/PSd1MhyEdH0YjHbDKAFxou2M408cp7bi2CnyWR7+zseRf1xMgcX0CEg
vSD0ui2vny+KPJXRvxi4jkv7V+B6aJcxWfxeoHrIzbZzNc34Gup2cq5pytYP+c070x0qIVP5neZMsU2fSVLTx8Jw4
RGBJF+kQck4HfNUgNG39bswZWEg9rxcPaKz10BqcDeFRn5VpYSxDoOPntmtIcGwh0S0Ea3syhycHW0GraqT1YZpb/t
CgPhE/M0SVZ1SIH9JkrbV7LrSwmG3nI1VVUww626zrr24UNQP3rh3BawFam1sp5Wt2CSbQoikTROPK4orvwMGMBUkv
X+EGK3WngM4sonxQ9FxlDdThne3T05pN7d73vPPjA9Q3Z20CYplC177GH5NSTx+F+MxyOdd/sENv11Mgg8V6FCiwb0A
yXCAZBZSnCyhXFXxKrYndJvnLeGbLLbZapvObhabHbvqZVWudh7VjnEpN7Pnc6hLhWliUPn2WQV1kg1+ZorB+v9/8
3sPgEsNc99YwZBCxw3p/P9vM2f+Jv2o83L0dlv5QUAt1/sPI7phjUT+LhyBKHg7VX0Ay0zN4601stWvdhnXn4QN8Ur
JiIcjobEJPwdscYD4MF9QWFLA6p4EuJ1b7sf18Cq8KjVmEyYSJsaWaEnXQGxagDqMvM1orgmUnIQnCdlH06CV10btv
83wbkRgCCvpi02otdcknp07BUGCpJ7MJ+fAWnPGs2ouDaQHBfpp5YpUEb36Ux5M7Ub+DwB8xEFpdiVivA6oox06CXt
eKNhFaqH02B3HZFHYKiz+3MPie5WtvZphKo/eDEhQ2YBrJ1JG4LKtrFLZj0Cc8x15qokYyZ1Br7QPCuME2IqzS4gFr
2myD8Qg/gb3IdxUU5AnN4qL2+BmBNWjHuh4YCYCWpWqRskTWmk2EsjxJqaaq0cb9t8RAXGqlfYfJpKr640owK70lds
LkSk97nw0JmPoijqnwqWJD5UN+L01EbgJfogEEIHu5Bi1hUX8DpU97Q3cgzK3hN9+Bj9LP0ER02eE0Z0Q4pqevZble
vWg/mZbD+dhqoJ1T02pGXuBg8tSvAY3uBsHEoGc2l8+3lE7rk5ePDYgKwBRlfEgKUWE70SaVudrDDsPl6xAYH2cTMT
IY6X3/ua/40NcQSpFUpjtr0kU5uB+Riet/dAvCG2f2gq0T7rIX/v1z1P8L+2pBoAGZhX1NjdXfGsdJKXWgrzUsV01
H5pfXjKv0zPJ05Cs5pJ8L8/U+kTTCfvtLQAx7hSFHEmUpZsn2meaEu7bCA12fKpGEhsaDjsi+zrknkcSCBKjJUYTwc
t9iv90TgK23hL9fXRIoQTWigabsqcwohsKSGWfDM4JApN0UsYaHpoJh023x73XMBSGQVNBuAUYn518gVxxXinXCHPz
ObKBZLo0MajFCfC5ds9AN+8vY1Cp4S4ER+gJDr9sXfJ12za1hXSeZZgfw7ZxCCXC/xq2MRqMq6IYT1WgomuyVvKC
X7uSYXyNOE1texasAbjGQTBnzr1zfaBZFLv6QSh8FZV5IFpJu59Ik+1LQsnoYDxyQ6HBZhx/T8rD6s5mWR0ZSYcSnSf
pYMZ510WKDmi/NLbu1JLB1GaYR/sjSS2g6ZtX2nMbGnrjkiTRvxHPtDBcgjXW6sxcoIFVULP6vhbVN+Fp/n9sLdHwR
A20wz5TCxfHEB8LTySar9szEG7brHqnxHEHaJFMiJNDaW0BF0KnzwbUi/c7F6K7swHI52U0ce4Qq3SD5ogBIg75pk
QsMDqUeTKhu8QD2NNXqD1JKwdqDd7+P1Hn+u4KQ8/6SeQbmg5fV8JNeHB4JsgKTE+JmytuGGgiq4mbgXSACfWCSgbk
ppsXtTgWu0lDcfLVQcYt/CGxacrRVq0bAR+sud6EsiZgm1A0vrXrTQiVbe+RrlyXT1VP0E02Gq2JhvIYLVH00Htz3
dAesheAHvbsZzhTsQAEsbpPU8b+vdxwVx1cgtHvjMgXrMZPdr6jbQ2a50p9+s/8pMfGkLkF0vWlCwd4vucX13A8IQ1
6VOL/JT8PJrBPH+6t2tH17T7Jt5fCqZ16TBP6j6LigpON78CgpqMqVbQZTF7CmDIJcVRbARVUhx2Dfppm319Do0IVS
dGQkPTSFEeTWRsFDD37eNwEs4jZ/iNhMhyje0Tr95AY1p0tj599fPLiBcNtQ19Q4+qXBCMFsSuKrmX02okwDNzpLjpc
teOiQn9VJiJ9vx9mtGklcbBAG7HjVZGRj2ARFGGrFfxvimZL1IZ6/xeHq6nC/VXJrr2sSNmkuvfaJafa+Ml00zFqLF
Ty0G9h0WPiirZKVYr7f7qBD7KQzcAUDiUI9QNMRP4iseQ7Jfwb8wVYYPfLcXTGNgBtmg="
}

session = requests.Session()
LOGIN_URL = "https://www.goodreads.com/user/sign_in"

session.headers = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/
response = session.get(LOGIN_URL)
print(response)

print(f"attempting to log in as {email}")
p = session.post(LOGIN_URL, data=payload)
print("risposta dopo la post")
print(response)

outfile = open('reviews.csv', mode='w')
writer = DictWriter(outfile, fieldnames=[
    'review_id', 'title', 'author', 'review_date', 'last_finished_date', 'rating', 'text'
])

page_num=1
review_ids = set()
REVIEW_LIST_URL = "https://www.goodreads.com/review/list/{id}?view=reviews&shelf=read&page={page}"

print(REVIEW_LIST_URL.format(target_user_id, page_num))
response = session.get(REVIEW_LIST_URL.format(target_user_id, page_num))

```

```
REVIEW_URL = "https://www.goodreads.com/review/show/{}"
for id in parse_review_list(response.text):
    print("parsing review id {}: {}".format(id, REVIEW_URL.format(id)))
    review_ids.add(id)
    response = session.get(REVIEW_URL.format(id))
    r = parse_review(response.text)
    print(r)
    r['review_id'] = id
    writer.writerow(r)

session.close()
```

Reperire le review di un utente su goodReads

```
In [ ]: e=input('Enter your email: ')
        print('Hello, ' + e)
print(type(e))
p=input('Enter your password: ')
print('Hello, ' + p)
u=input('Enter user_ID: ')
print('Hello, ' + u)

authenticateAndScrape(e,p,u)
```

Assignment: Scraping di IMDb, Internet Movie Database

- è un sito web di proprietà di Amazon.com che gestisce informazioni su
 - film
 - attori
 - registi
 - personale di produzione
 - programmi televisivi
 - videogiochi.
- reperire le valutazioni di tutti gli episodi di **Game of Thrones**

Assignment: Scraping Game of Thrones

- accedere alla pagina

`https://www.imdb.com/title/tt0944947/episodes`

- effettuare lo scraping della pagina per reperire il rating di ogni episodio di ogni stagione
- plottare un barplot in cui si mettono a confronto le valutazioni date ai vari episodi della stagione