

# Web Scrapping (& Flask API)

## Team 2:

Kazakos Christos, it22033

Katsaras Konstantinos, it22045

Linardakis Manousos, it22064

# Web Scraping

*Definition:* Python web scraping is an automated method used for collecting large amounts of data from websites and storing it in a structured form.

It serves so that we can extract – manage data that **not** are available from some ready-made API!



# Popular FrameWorks For Web Scrapping



Scrapy

BeautifulSoup



Requests-  
HTML

*HTML Scrapping for Humans.*

# FrameWorks We Used For Web Scrapping



BeautifulSoup



# Why Selenium?

We chose selenium as it offers a WebDriver Interface.

So we can directly press buttons with a function (`click()`) or wait until the page loads (`wait_until()`) like a normal user.

We will see these functions in more detail later...



# Why BeautifulSoup?

Beautifulsoup is lighter and faster than Selenium.

We export **text from html** which we get from selenium.

Below are examples (code) to better explain the web scraping methods used.

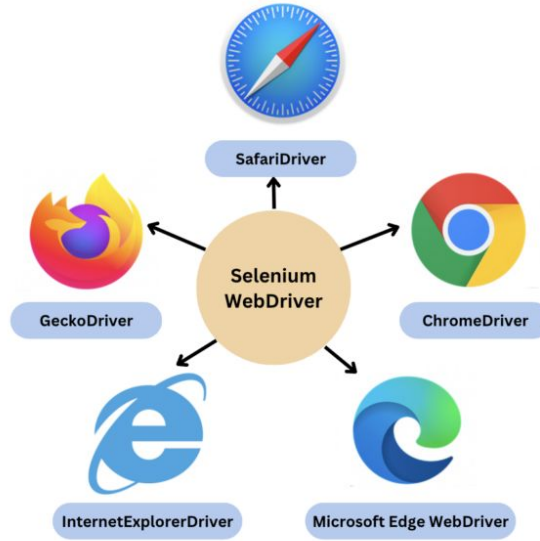
BeautifulSoup



# Web Scrapping Example Code

# Selenium Web Drivers

Selenium to interact with a web browser uses Web Drivers!



[image source](#)



# Selenium Available WebDrivers

```
from selenium import webdriver
```

```
webdriver.Firefox  
webdriver.FirefoxProfile  
webdriver.Chrome  
webdriver.ChromeOptions  
webdriver.Ie  
webdriver.Opera  
webdriver.PhantomJS  
webdriver.Remote  
webdriver.DesiredCapabilities  
webdriver.ActionChains  
webdriver.TouchActions  
webdriver.Proxy
```

# Web Drivers Example

try:

```
options=Options()
options.add_argument('--headless')# remove this if you want the browser to appear.
# create Chrome service - installs chrome driver:
service=Service(ChromeDriverManager().install())
# initialize driver:
driver=webdriver.Chrome(service=service,options=options) except:# starts
firefox
options=webdriver.FirefoxOptions()
options.add_argument('--headless')# remove this if you want the browser to appear.
service=webdriver.firefox.service.Service(executable_path=GeckoDriverManager().install()) driver=webdriver.Firefox(
service=service,options=options)# go to url:
```

# Selenium: driver.get(page\_url)

To drive him webdriver on the desired page, we use it  
driver.get(page\_url).

```
discogs_url = "https://www.discogs.com"
```




```
page_url = discogs_url + "/artist/" + artist_name + "?limit=500"
```


```
print ( 'Opening Page...' )
```

```
driver . get ( page_url )
```



# Result in Web Driver

 Search artists, albums and more...  Explore ▾ Marketplace ▾ Community ▾  Log In Register

  
[More Images](#)


## Queen

Profile: Queen is a British rock band formed in London in 1970 from the previously disbanded [Smile \(6\)](#) Rock band. Originally called Smile, later in 1970 singer [Freddie Mercury](#) came up with the new name for the band. [John Deacon](#) joined in March 1971 giving them their fourth and final bass player.

The band has released a total of 18 number-one albums, 18 number-one singles and 10 [More ▾](#)

Sites: [queenonline.com](#), [Facebook](#), [Wikipedia](#), [Twitter](#), [MySpace](#), [YouTube](#), [Wikipedia](#), [Last.fm](#), [queenworld.com](#), [queenvault.com](#), [queenpedia.com](#), [shanemcdonald.ie](#), [ultimatequeen.co.uk](#)

### Artist

 [a81013]

[Edit Artist](#)

[Share](#)

### Marketplace

[103,121 For Sale](#)

[Vinyl and CD](#)

## Let's manage your privacy

We and our partners store and/or access information on a device, such as unique IDs in cookies to process personal data. You may accept or manage your choices by clicking below, including your right to object where legitimate interest is used, or at any time in the privacy policy page. These choices will be signaled to our partners and will not affect browsing data. [Read our Cookie and Internet Advertising Policy](#)

### We and our partners process data to provide:

Actively scan device characteristics for identification. Use precise geolocation data. Measure content performance. Store and/or access information on a device. Select personalised content. Develop and improve products. Apply market research to generate audience insights. Create a personalised content profile. Measure ad performance. Select personalised ads. Create a personalised ads profile. Select basic ads.

[View our list of partners \(vendors\)](#)

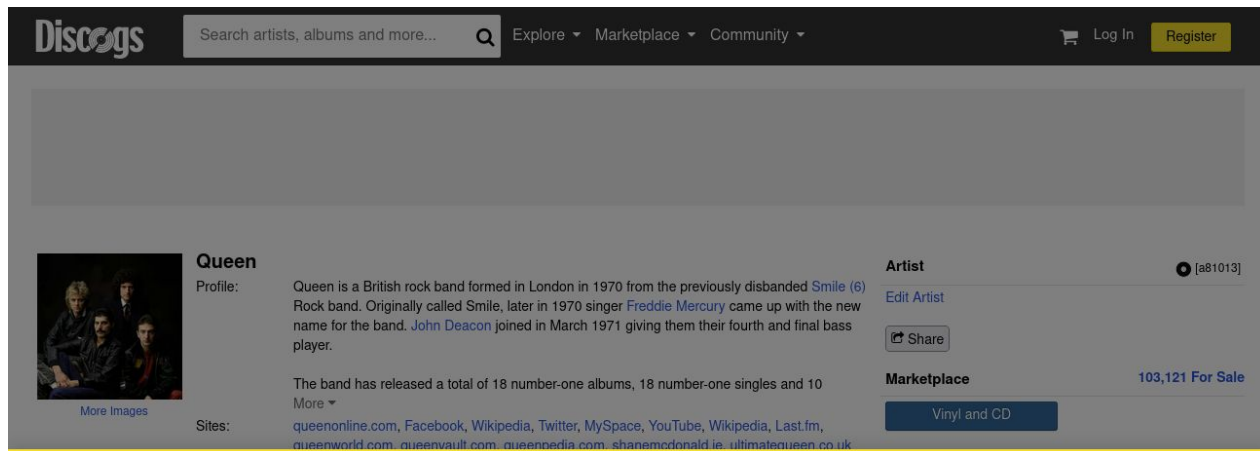
Allow all cookies

Reject all unnecessary cookies

[Show purposes](#)

# Web Scrapping – Accepting Cookies

We will then need to accept the cookies so that it is all loaded the page.



The screenshot shows the Discogs website interface. At the top is a navigation bar with the Discogs logo, a search bar, and links for Explore, Marketplace, and Community. On the right of the navigation bar are links for Log In and Register. The main content area displays the profile for the band Queen. It includes a profile picture, a bio stating they are a British rock band formed in 1970, and a list of social media sites. There is also a 'Marketplace' section showing 103,121 items for sale.

## Let's manage your privacy

We and our partners store and/or access information on a device, such as unique IDs in cookies to process personal data. You may accept or manage your choices by clicking below, including your right to object where legitimate interest is used, or at any time in the privacy policy page. These choices will be signaled to our partners and will not affect browsing data. [Read our Cookie and Internet Advertising Policy](#)

## We and our partners process data to provide:

Actively scan device characteristics for identification. Use precise geolocation data. Measure content performance. Store and/or access information on a device. Select personalised content. Develop and improve products. Apply market research to generate audience insights. Create a personalised content profile. Measure ad performance. Select personalised ads. Create a personalised ads profile. Select basic ads.

[View our list of partners \(vendors\)](#)

Allow all cookies

Reject all unnecessary cookies

[Show purposes](#)

# Web Scraping – WebDriver Wait

To accept cookies, however, we must make sure that the "Allow all Cookies" button is present on the page. For this we use:

```
wait=WebDriverWait(driver,10)  
button=wait.until(EC.presence_of_element_located((By.ID,"onetrust-accept-btn-handler")))
```

1. We initialize a WebDriverWait object with a maximum wait limit (for the page to load) of 10 seconds.
2. Then, we use wait.until, which waits until a button with the given id appears (which corresponds to the “allow all cookies” button for the discogs page).

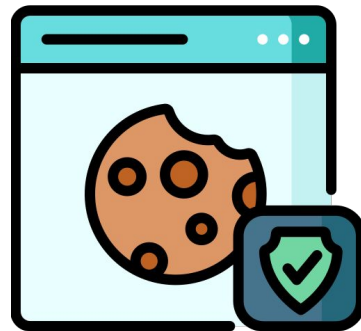


# Web Scrapping – Click (Cookies) Button

After we have verified that the cookie button has been loaded, we click it using the method:


```
button.click()
```

This is how we managed to accept cookies and load the whole page correctly.



[image source](#)

# Result of Accept All Cookies Click



[More Images](#)

## Queen

**Profile:** Queen is a British rock band formed in London in 1970 from the previously disbanded [Smile](#) (6) Rock band. Originally called Smile, later in 1970 singer [Freddie Mercury](#) came up with the new name for the band. [John Deacon](#) joined in March 1971 giving them their fourth and final bass player.

The band has released a total of 18 number-one albums, 18 number-one singles and 10 number-one DVDs, and have sold over 300 million albums worldwide, making them one of the world's best-selling [More](#) ▾

**Sites:** [queenonline.com](#), [Facebook](#), [Wikipedia](#), [Twitter](#), [MySpace](#), [YouTube](#), [Wikipedia](#), [Last.fm](#), [queenworld.com](#), [queenvault.com](#), [queenpedia.com](#), [shanemcdonald.ie](#), [ultimatequeen.co.uk](#)

**Members:** [Barry Mitchell](#), [Brian May](#), [Freddie Mercury](#), [John Deacon](#), [Roger Taylor](#)

**Variations:** [Viewing All](#) | [Queen](#)

Kween, Q, Qeen, Queen Productions, Quenn, The Queen, Куийн, Куин, クイーン, クイーン, 皇后乐队, 皇后合唱团, 皇后合唱团, 皇后樂隊

### Artist

[Edit Artist](#)

[Share](#)

**Marketplace** [103,121 For Sale](#)

[Vinyl and CD](#)




### Discography

Search

1 – 25 of 540 [◀](#) [Prev](#) [Next](#) [▶](#)

Sort [Year, 0-9](#) Show [25](#) [Grid](#) [List](#) [Menu](#)

#### Albums

	<a href="#">Queen</a> <a href="#">◀ 283 versions</a>	<a href="#">EMI, EMI</a>	1973
	<a href="#">Sheer Heart Attack</a> <a href="#">◀ 267 versions</a>	<a href="#">EMI, EMI</a>	1974
	<a href="#">Queen II</a> <a href="#">◀ 279 versions</a>	<a href="#">EMI, EMI</a>	1974

**Releases** 540

[Albums](#) 83

[Singles & EPs](#) 172

[Compilations](#) 102

[Videos](#) 42

[Miscellaneous](#) 141

[Appearances](#) 257

**TURNTABLE LAB**

SHOP ONLINE 24/7  
THANKS FOR YOUR SUPPORT!



# Web Scraping – Selecting Specific Elements

Having loaded the entire page, we will need to select specific elements of it (that is, what we are interested in for web scraping). For this we use:

```
table=driver.find_element(By.ID,'artist'):
```

**we find** the element with id “artist” – we store it in a variable.

```
table_html=table.get_attribute('outerHTML')
```

**we get** all content HTML of the specific table.



# Result of Selection "artist"

The screenshot shows the Discogs website for the artist Queen. The URL is `https://www.discogs.com/artist/81013-Queen&limit500?page=1`. The page features a sidebar with navigation links for Releases (540), Appearances (3587), and Unofficial (1569). The main content area displays a table of albums, with the first row being Queen's 1973 album. A banner at the bottom of the album list reads "10/10 Albums According To The Discogs Community".

The browser's developer tools are open, showing the HTML structure and CSS styles for the selected table. The HTML structure is as follows:

```
<div id="releases" class="content_with_ort_canvas">
  <div id="pjax_container" data-pjax-timeout="30000" aria-live="polite">
    <div class="hide-desktop">
    <nav class="pagination top" aria-label="Pagination">
    <table id="artist" class="cards table_responsive layout_normal">
      <thead>
      <tbody>
      <thead>
      <tbody>
      <thead>
      <tbody>
      <thead>
      <tbody>
```

The CSS styles for the selected table are:

```
body :: {
  line-height: 20px;
}
```

The table structure is as follows:

1 - 25 of 540		Prev	Next	Year, 0-9	Show	25
Queen	283 versions	EMI, EMI	1973			
Sheer Heart Attack	267 versions	EMI, EMI	1974			
Queen II	272 versions	EMI, EMI	1974			
Randy Newman / Queen - Stereo Pop Special-72 (LP, Transcription)		BBC Transcription Services	CN 2073/S 1974			

# Web Scraping – BeautifulSoup

Then we will need to "read" all the elements of the table → **very time consuming (with selenium, due to the web driver interface).**

That's what we use BeautifulSoup for!

BeautifulSoup



# Web Scraping – BeautifulSoup Code

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

html.parser is used to access the html and make a BeautifulSoup Object.

```
a_tags=soup.find_all('a',href=lambdahref:hrefand('/master/'inhrefor'/release/'inhref))
```

we get the hrefs of all anchors (<a>) of the table that contain the string '/master/' or '/release/'. These hrefs are the links that lead to the records of the artist (of the page we are on).

So we have stored all the urls of the artist's records in a variable a\_tags.



# Saving disc URLs in a dictionary

We create a dictionary "albums" which contains as key the name of the disk and for value the url of this disk:

```
for a_tag in a_tags:
    disc_txt = a_tag.text
    disc_txt = disc_txt.replace(' ', '-').replace('/', '-')
    if disc_txt not in albums: # keeps
        the top albums
        albums[disc_txt] = discogs_url + a_tag[href]
```

To get the disk name  
we use `a_tag.text` → returns the text of  
anchor `<a>`.



# Web Scrapping

Having stored all the artist's albums and their corresponding urls in the dictionary, we access them by going to each url with `driver.get(page_url)` → we collect information about the disk like this:

Discography

Search

Releases

540

Albums

83

Singles & EPs

172

Compilations

103

Videos

41

Miscellaneous

141

Appearances

3613

Albums

274

1 – 500 of 540

< Prev

Next >


Sort

Year, 0-9

Show

500

Albums




Queen

◀ 284 versions

EMI, EMI

1973

▼




Sheer Heart Attack

◀ 268 versions

EMI, EMI

1974

▼



Queen II

◀ 274 versions

EMI, EMI

1974

▼

# Web Scrapping

Having stored all the artist's albums and their corresponding urls in the dictionary, we access them by going to each url with `driver.get(page_url)` → we collect information about the disk like this:

Discography

Search

Q

1 – 500 of 540

◀ Prev

Next ▶

Sort

Year, 0-9 ▾

Show

500 ▾

Releases

540

Albums

83

Singles & EPs

172

Compilations

103

Videos

41

Miscellaneous

141

Appearances

3613

Albums

274

Albums

Queen

◀ 284 versions

EMI, EMI

1973

▾

Queen

◀ 268 versions

EMI, EMI

1974

▾

Queen II

◀ 274 versions

EMI, EMI

1974

▾

# Web Scrapping

Having stored all the artist's albums and their corresponding urls in the dictionary, we access them by going to each url with `driver.get(page_url)` → we collect information about the disk like this:

Discography

Search

Q

Releases

540

Albums

83

Singles & EPs

172

Compilations

103

Videos

41

Miscellaneous

141

Appearances

3613

Albums

274

1 – 500 of 540

< Prev

Next >


Sort

Year, 0-9

Show

500

Albums




Queen

◀ 284 versions

EMI, EMI

1973

▼




Sheer Heart Attack

◀ 268 versions

EMI, EMI

1974

▼



Queen II

◀ 274 versions

EMI, EMI

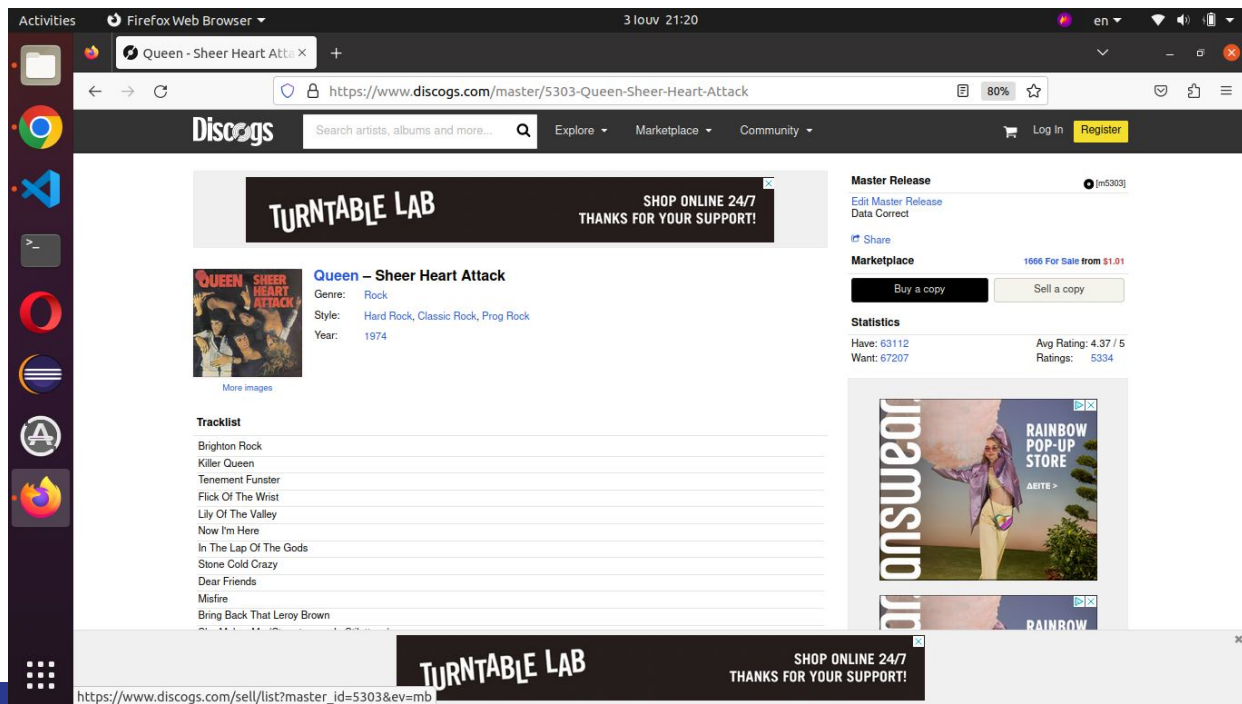
1974

▼



# Web Scrapping

By "clicking" each url of the dictionary discs, we are taken to a page where there is a "Buy Copy" button.



# Web Scraping

We choose the Buy a copy Button:

```
buy_button=soup.find('a',string='Buy a copy')
```

option anchors with the string “Buy a copy” (it's just the Buy a copy buttons on this page).

and we get the href of the button as follows:

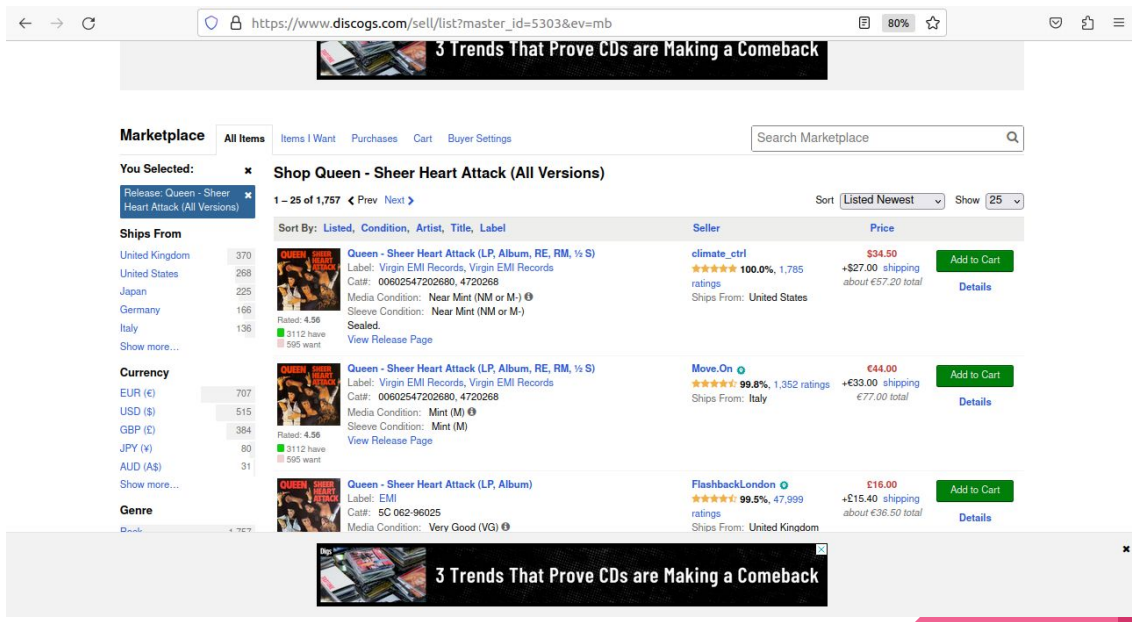
```
buy_link=buy_button.get(href)
```

then we store it again in a dictionary.



# Web Scraping

In a similar way as before, we access the new dictionary by “pressing” all the buy a copy buttons (with `driver.get(page_url)`) → we select itview release page of each disc on the page:



# Web Scrapping

By selecting each view release page we get prices for the discs for various days of the years they are released:

The screenshot shows the Discogs website interface. At the top is a navigation bar with the Discogs logo, a search bar, and links for Explore, Marketplace, and Community. Below the navigation bar is a banner for '7 Essential David Bowie Albums'. The main content area features the release page for 'Queen – Sheer Heart Attack'. The album cover is on the left, and the release details are on the right. The details include the label (Virgin EMI Records), format (Vinyl, LP), country (Europe), release date (Sep 25, 2015), genre (Rock), and style (Hard Rock, Pop Rock, Arena Rock). Below the details is a tracklist with 6 tracks. On the right side of the page, there are sections for 'Release' (with edit and version links), 'Marketplace' (with buy and sell buttons), and 'Statistics' (showing have, want, and ratings data). At the bottom of the page, there is a banner for 'Top 25 Most Expensive Items Sold on Discogs in April 2023'.

**Discogs** Search artists, albums and more... Explore Marketplace Community Log In Register

**7 Essential David Bowie Albums**

**Queen – Sheer Heart Attack**

Label: [Virgin EMI Records – 00602547202680](#), [Virgin EMI Records – 4720268](#)  
Format: [Vinyl, LP](#), Album, Reissue, Remastered,  $\frac{1}{2}$  Speed, 180 Gram  
Country: [Europe](#)  
Released: [Sep 25, 2015](#)  
Genre: [Rock](#)  
Style: [Hard Rock](#), [Pop Rock](#), [Arena Rock](#)

**Tracklist** Hide Credits

A1	Brighton Rock Written-By – <a href="#">May*</a>	5:10
A2	Killer Queen Written-By – <a href="#">Mercury*</a>	3:00
A3	Tenement Funster Written-By – <a href="#">Taylor*</a>	2:47
A4	Flick Of The Wrist Written-By – <a href="#">Mercury*</a>	3:17
A5	Lily Of The Valley Written-By – <a href="#">Mercury*</a>	1:45
A6	Now I'm Here Written-By – <a href="#">May*</a>	4:13

**Release** [7541484]  
[Edit Release](#)  
[All Versions of this Release](#)  
Recently Edited

[Add to Collection](#) [Add to Wantlist](#)

**Marketplace** [99 For Sale from \\$21.51](#)

[Buy Vinyl](#) [Sell Vinyl](#)

**Statistics**

Have:	3112	Last Sold:	May 21, 2022
Want:	596	Lowest:	\$19.06
Avg Rating:	4.56 / 5	Median:	\$24.96
Ratings:	325	Highest:	\$45.96

☆☆☆☆☆ [Share](#)

**Top 25 Most Expensive Items Sold on Discogs in April 2023**

# Web Scraping

So we have many prices for the record on various dates. We store all the results in a pandas to deal with outlier values more easily:

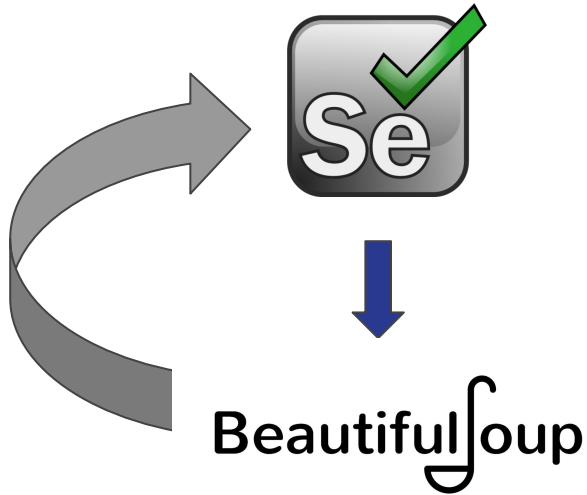
```
# Convert data list (with disc prices) into pandas DataFrame df=pd.DataFrame(  
data)
```

```
# Convert the "lowest_price", "median_price", and "highest_price" columns to floats  
df[["lowest_price","median_price","highest_price"]]=df[["lowest_price","median_price",  
"highest_price"]].applymap(lambdax:float(x.lstrip("$")))
```

```
# Convert the date column to datetime format df['date']=pd  
.to_datetime(df['date'])
```

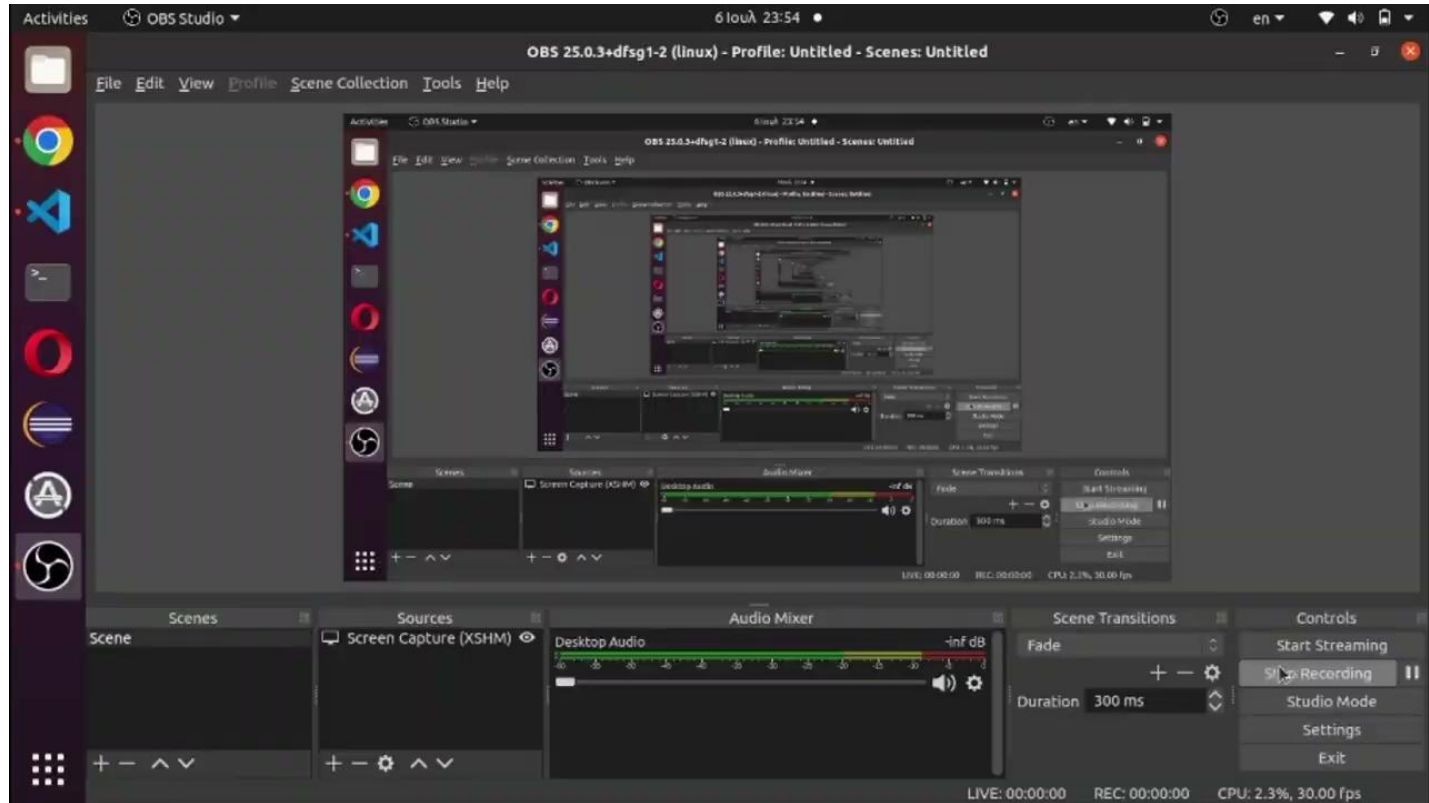


# Web Scraping Logic – Pattern



- 1) Navigate to the next page using **Selenium (easy navigation)**.
- 2) Extract data from HTML using **BeautifulSoup (fast)**.
- 3) Repeat!

# Web Scrapping Demo



<https://drive.google.com/file/d/1uammOiB9gioEWTTihNweGpIo0oQ-PIOS/view?usp=sharing>

# Flask API



source: <https://www.educative.io/blog/python-flask-tutorial>



## What is it?

With the flask api, we can **easily** and **quickly** to build a REST API in Python.

To initialize the REST API we use the command `app = Flask(__name__)` where `Flask` creates a new instance of the class `Flask` and the `__name__` refers to the name of the current module or package.

To run the API, we execute `app.run()` → as default runs on port 5000.



## @app.route('/path')

By using the decorator `@app.route('/path')` over a function, where `'/path'` is the path (URL) that the application will respond to, we specify which operation will be performed for each request made to it specific route `'/path'`. Also, we can define the `app.route` method (GET, POST, PUT...):

```
@app.route('/discs', methods=['GET'])
def get_user_discs_api():
    """
    Retrieves the discs of the authenticated user.

    Returns:
        JSON response: A JSON response containing the user's discs.
    """
```

# Flask Authentication

To get the user's information (from basic authentication), we use the flask's request object, which contains the user's username and password:

```
from flask import Flask, jsonify, request
```

```
def authenticate():
    """
    Performs authentication by checking the provided username and password against the credentials in the database.
    """
    auth_fail_msg = 'Authentication failed'
    auth = request.authorization
    if not auth:
        return jsonify({'message': auth_fail_msg}), 401
    # Get the provided username and password from the request
    username = auth.username
    password = auth.password
```

# Flask jsonify

We return a json using flask's jsonify. Thus, we can send some messages to the user along with a status code (with the return of the corresponding function):

```
@app.route('/discs', methods=['GET'])
def get_user_discs_api():
    """
    Retrieves the discs of the authenticated user.

    Returns:
        JSON response: A JSON response containing the user's discs.
    """
    # authentication ----
    auth = authenticate()
    if auth[1] != 200:
        return auth[0]
    username = auth[0]
    # -----
    discs = get_user_discs(username, conn)
    # Create a list of dictionaries with keys 'disc_name' and 'band_name'
    discs_list = [{'disc_name': disc[0], 'band': disc[1]} for disc in discs]

    # Return the user's discs as JSON response
    return jsonify({'user_discs': discs_list}), 200
```

# Cookies (1/2)

Cookies are small pieces of data stored on the customer's device by a website. In Flask, we can manage cookies using the object `request.cookies` to read them from the client and the function `make_response()` to generate a response which contains cookies.



## Cookies (2/2)

In the example, the route `/` responds to GET and POST requests. The object `request.cookies` is used to read the cookie named 'username' from the client. Then the function is used `make_response()` in order to a response containing the text 'Hello, ' plus the cookie value 'username' is generated. Finally, through the method `resp.set_cookie()` is defined cookies.

```
@app.route('/')
def index():
    # Διάβασμα του cookie 'username' από τον πελάτη
    username = request.cookies.get('username')

    # Επιστροφή απόκρισης που περιέχει ένα cookie
    resp = make_response('Hello, ' + username)

    # Ορισμός του cookie 'username' με την τιμή 'John Doe'
    resp.set_cookie('username', 'John Doe')

    return resp
```

# Other URLs:

Source Code:

<https://github.com/manouslinard/music-recommender>

Web Scraping demo:

<https://drive.google.com/file/d/1uammOiB9gioEWTTihNweGpIo0oQ-PJOS/view?usp=sharing>





Thank you for that  
you watched!