

The Big Picture : Scrape any Website in 4 Steps



In this chapter, we overview on common web scraping steps. Actually this is really simple process.

Step 1 : Web scraping always start with a web page and data items which we want to scrape out from the page. At first step, we need to understand web page then find out HTML tags contain our wanted data. Result from this step will be used in final step to actually scrape data.

Step 2 : Wanted Data is located inside HTML page. We download HTML to local. We use Selenium for download.

Step 3 : After have HTML page, we use Beautiful Soup to parse HTML content in to object. Then we search for HTML tags which contain our data.

Step 4 : Final step is scraping data and store it to file or database. In this book we will make it simple by store data to file.

Selenium

We use [Selenium](#) to control browser and download HTML content.

Why we not just use simple library like requests to download HTML content ?.

Have 2 reasons:

- Many modern web page use a lot of JavaScript for dynamic HTML render, requests package could not render HTML from JavaScript.
- In some web pages, in order access wanted data, we need to do actions like : login, click link to navigate. Selenium can do that perfectly.

Beautiful Soup

We need [Beautiful Soup](#) to parse HTML in to object. Beautiful Soup provide functions help us to search HTML tags inside HTML object.

After have HTML tags, final step just about access wanted data and save---

layout: default

Inspecting Web Page with Chrome Developer Tool

At step 1, we will go in deep detail how to use `Chrome Developer Tool` to inspect web page.

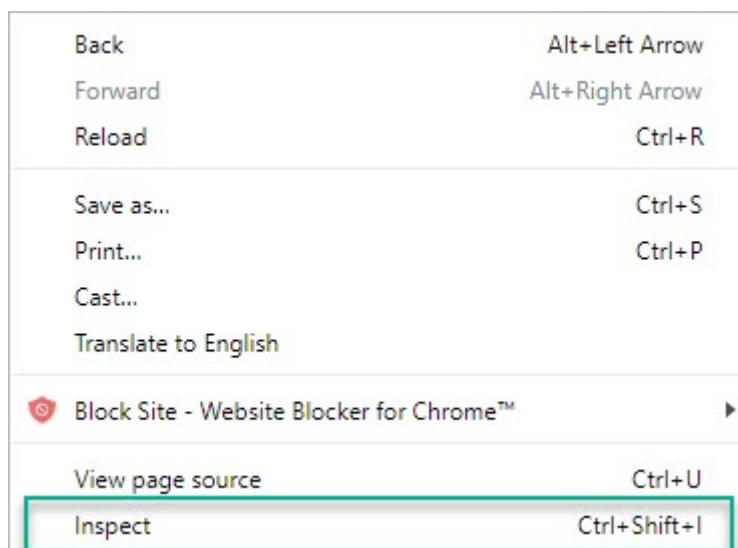
At end of this chapter you will understand how to do inspecting web page and have clear strategies to scrape data from HTML page.

We will use [NBA](#) for our demo, let access NBA with Chrome browser.



Chrome Developer Tool

We will use `Chrome Developer Tool` to inspect NBA home page. Now mouse over web page then right click, then select `Inspect` from context menu.



A panel will open and show up corespond HTML content of NBA home page. We saw that HTML show on a tree structure and very easy to navigate from tag to tag.

Example 1 : Inspect NBA News

Suppose we want to scrape all ***title and link*** inside the box "LATEST AROUND THE NBA".

Move mouse up and down on HTML tree view to findout what web page items is selected when we select corespond tag.

We found that when mouse select `div` tag which has class name `content_list--collection`, the whole news area will be selected. So the first step should be find the `div` tag with class name `content_list--collection`.

Now after know the `div` tag which contain all news, we need continue go to each new inside the box.

Continue try to moving the mouse around, we will see that titles and link are inside `a` tag which have class name `content_list--item clearfix`.

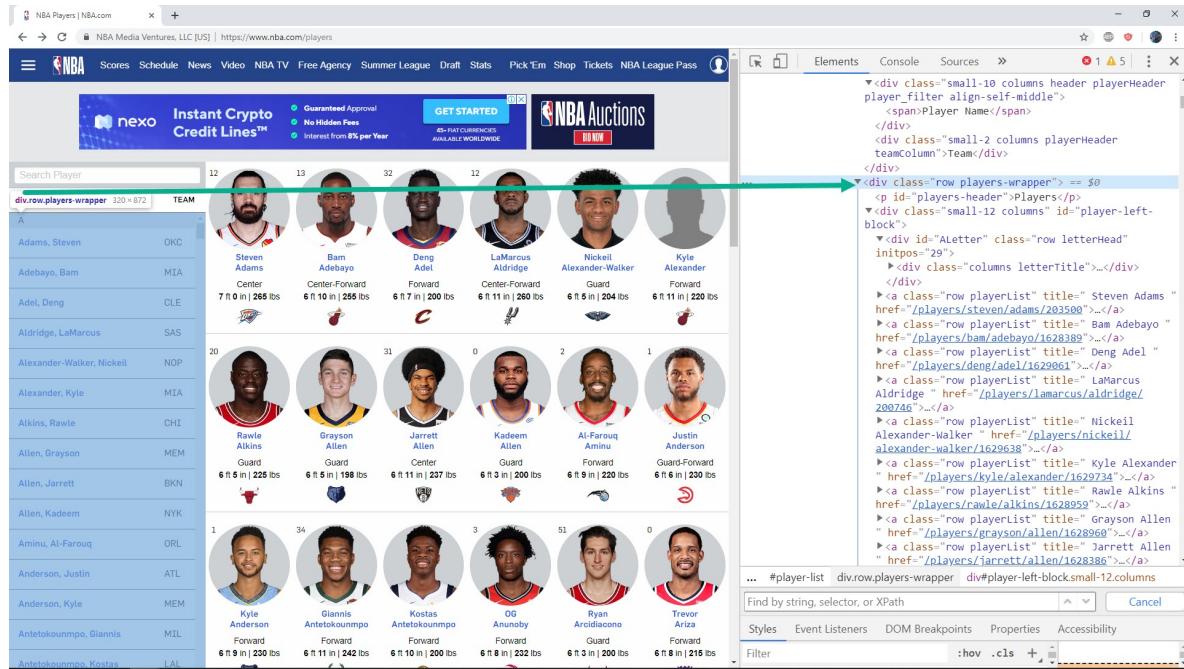
So for conclusion we have following strategy to scrape all new title inside "LATEST AROUND THE NBA"

- Search for `div` tag which has class name `content_list--collection`.
- Inside above `div` tag, search for all `a` tags which have class name `content_list--item clearfix`.

Example 2 : Inspect NBA Players

Suppose we want to **scrape detail data for each players** on link <https://www.nba.com/players>.

From web page we see that we want to find out HTML tag which contain all player name. Right click mouse and select `inspect`, then move mouse around we know that we need to find `div` tag with class name `players-wrapper`.



The screenshot shows the NBA Players page with various player cards. The developer tools are open, specifically the Elements tab, showing the DOM structure. A green arrow points from the text 'we need to find all a tag with class name playerList' to the `>a class="row playerList"` line in the DOM tree, which corresponds to the `playerList` class mentioned in the text.

```
... #player-list div.row.players-wrapper div#player-left-block.small-12.columns
Find by string, selector, or XPath
Cancel
Styles Event Listeners DOM Breakpoints Properties Accessibility
Filter :hover .cls +
```

Inside the `div` we found above, we nee do to find all `a` tag with class name `playerList` which contain name and link to detail data for each player.

NBA Players | NBA.com

Guaranteed Approval
No Hidden Fees
Interest from 8% per Year

GET STARTED

40+ FLAT CURRENCIES AVAILABLE WORLDWIDE

Search Player

PLAYER NAME	TEAM
Adams, Steven	OKC
Adebayo, Bam	MIA
Adel, Deng	CLE
Aldridge, LaMarcus	SAS
Alexander-Walker, Nickeil	NOP
Alexander, Kyle	MIA
Alkins, Rawle	CHI
Allen, Grayson	MEM
Allen, Jarrett	BKN
Allen, Kadeem	NYK
Aminu, Al-Farouq	ORL
Anderson, Justin	ATL
Anderson, Kyle	MEM
Antetokounmpo, Giannis	MIL
Antetokounmpo, Kostas	LAL

Steven Adams
Center
7 ft 0 in | 265 lbs

Bam Adebayo
Center-Forward
6 ft 10 in | 255 lbs

Deng Adel
Forward
6 ft 7 in | 200 lbs

LaMarcus Aldridge
Center-Forward
6 ft 11 in | 260 lbs

Nickeil Alexander-Walker
Guard
6 ft 5 in | 204 lbs

Kyle Alexander
Forward
6 ft 11 in | 220 lbs

Rawle Alkins
Guard
6 ft 5 in | 225 lbs

Grayson Allen
Guard
6 ft 5 in | 198 lbs

Jarrett Allen
Center
6 ft 11 in | 237 lbs

Kadeem Aminu
Guard
6 ft 3 in | 200 lbs

Al-Farouq Aminu
Forward
6 ft 9 in | 220 lbs

Justin Anderson
Guard-Forward
6 ft 6 in | 230 lbs

Kyle Anderson
Forward
6 ft 9 in | 230 lbs

Giannis Antetokounmpo
Forward
6 ft 11 in | 242 lbs

Kostas Antetokounmpo
Forward
6 ft 10 in | 200 lbs

OG Anunoby
Forward
6 ft 8 in | 232 lbs

Ryan Arcidiacono
Guard
6 ft 3 in | 200 lbs

Trevor Ariza
Forward
6 ft 8 in | 215 lbs

...
...

Find by string, selector, or XPath

Styles Event Listeners DOM Breakpoints Properties Accessibility

From link for each player above, we go to detail page, for example we go to detail page for player Steven Adams at reference. In this page, we need to find `div` tag with class name `nba-player-vitals` which contain all detail informations

NBA Players | NBA.com

Steven Adams stats, details, video

2018 - 19 POSTSEASON CAREER STATS

	MPG	F% 0	3P% 0	FT% 37.5	PPG 11.8	RPG 7.2	APG 1.4	BPG 1
2018-19	31.8	66.7			11.8	7.2	1.4	1
POSTSEASON	28.8	58.8	0	55.3	9.7	7.4	1	1
CAREER STATS	28.8	58.8	0	55.3	9.7	7.4	1	1

Info Stats

section.nba-player-vitals.small-12 large-6

HEIGHT
7 ft 0 in / 2.13m
WEIGHT
265 lbs / 120 kg

BORN
07/20/1993
AGE
26 years
FROM
Pittsburgh
NBA DEBUT
2013
YEARS IN NBA
8
PREVIOUSLY
OKC 2013-19

LATEST VIDEOS

2018-19 SEASON AVERAGES:
Appeared in 231 games (167 starts) and averaged 6.2 points, 6.0 rebounds and 1.0 blocks in 21.6 minutes

2015-16 SEASON AVERAGES:
Appeared in 80 games (80 starts) and averaged 7.9 points, 6.7 rebounds and 1.1 blocks in 25.2 minutes.

2015-16:

PLAYER BIO

Professional History

CAREER AVERAGES:
Appeared in 231 games (167 starts) and averaged 6.2 points, 6.0 rebounds and 1.0 blocks in 21.6 minutes

2015-16 SEASON AVERAGES:
Appeared in 80 games (80 starts) and averaged 7.9 points, 6.7 rebounds and 1.1 blocks in 25.2 minutes.

... div #player-tabs-Info section section.nba-player-vitals.small-12.large-6

Styles Event Listeners DOM Breakpoints Properties Accessibility

That it, now let summary again strategy to get detail information for each players:

- From url <https://www.nba.com/players> search for `div` tag which has class name `players-wrapper`.
 - Inside `div` above, search for all `a` tags which has class name `playerList`.
 - Which each `a` tags above, we scrape `href` attribute to have link to detail information for each player.
 - Inside each detail url, for example <https://www.nba.com/players/steven/adams/203500> we search for `div` tag which has class name `nba-player-vitals`. Inside this `div` tag contain all detail data we need.
- layout: default

Download HTML Page Content

At step 2, we will download HTML page content with selenium package. And try to control Chrome web browser in two mode normal mode and headless mode.

Installation

First of all we need to install selenium package in order to control browser with python. Let open command prompt on Windows (terminal in Mac) and typing in.

```
pip install selenium
```

To control Chrome, We also need to download [Chrome driver](#). Let choose stable version and place some where in your local.

All versions available in Downloads

- Latest stable release: [ChromeDriver 75.0.3770.140](#)
- Latest beta release: [ChromeDriver 76.0.3809.68](#)



Control browser in normal mode

We will use `webdriver` to control Chrome browser, so the first step is import `webdriver` from `selenium` package.

```
from selenium import webdriver
```

Then we want to create `chrome` object, note that we need to specify absolute path where we place downloaded `chrome driver` above.

```
driver_path = r'C:\chromedriver_win32\chromedriver.exe'  
driver = webdriver.Chrome(executable_path=driver_path)
```

Now with `driver` object, we could download HTML content with `get` function. Support we want to download HTML from <https://www.nba.com/>. You could see that a Chome browser instance is created then NBA home page is loaded.

```
driver.get('https://www.nba.com/')
```

Now to access HTML content, we just need to use property `page_source`

```
print(driver.page_source)
```

At here we complete download HTML content and print it out, at final step we should `close` driver so you will see Chrome driver close.

```
driver.close()
```

That it, let summary every thing again

```
# import webdriver
from selenium import webdriver

# create Chrome object
driver_path = r'C:\chromedriver_win32\chromedriver.exe'
driver = webdriver.Chrome(executable_path=driver_path)

# access url
driver.get('https://www.nba.com/')

# access HTML content
print(driver.page_source)

# close browser
driver.close()
```

Control browser in headless mode

Headless mode mean we still create a Chrome browser instance to download HTML but Chrome browser **do not show up**.

Why we do not want browser to show up ?

The answer is in `headless` mode every thing will faster and our scraping script consume less cpu, ram resource. So after we build up script completely, we should running it in `headless` mode. We do normal mode when we want to debug and develop script.

To run brower in `headless` mode, the first step we need to create `ChromeOptions` object and set value to `headless` property.

```
# create option for headless mode
options = webdriver.ChromeOptions()
options.headless = True
```

Now when we create `Chrome` object, we need put in `option` parameter.

```
# create Chrome object with option
driver_path = r'C:\chromedriver_win32\chromedriver.exe'
driver = webdriver.Chrome(executable_path=driver_path,
                           options=options)
```

That it, let summary every thing again

```
# import webdriver
from selenium import webdriver

# create option for headless mode
options = webdriver.ChromeOptions()
options.headless = True

# create Chrome object
driver_path = r'C:\chromedriver_win32\chromedriver.exe'
```

```

driver = webdriver.Chrome(executable_path=driver_path,
                           options=options)

# access url
driver.get('https://www.nba.com/')

# access HTML content
print(driver.page_source)

# close browser
driver.close()
```
layout: default
```

### Create Soup and Search for Tag

At step 3, after have HTML content downloaded with selenium, we use `beautiful soup` to parse the HTML and start search for tags which contain our wanted data.

#### Install Beautiful Soup

we need to install `beautiful soup` package before using it.

```python
pip install beautifulsoup4

```

## Create Soup Object

For easy of explain, we will use a predefine HTML content as following code. When we do actual web scraping, step 2 will provide HTML content with `driver.page_source`

```

import beautifulsoup object
from bs4 import BeautifulSoup

predefine a html content
html = """
<html>
<head>
 <title>The Dormouse's story</title>
</head>
<body>
 <p class="title">
 The Dormouse's story
 </p>
 <p class="story">
 Once upon a time there were three little sisters. And their names were
 Elsie,
 Lacie
 and
 <a href="http://example.com/tillie" class="sister"
id="link3">Tillie;
 and they lived at the bottom of a well.
 </p>
 <p class="story">...</p>
</body>
</html>

```

```
"""
create soup object by HTML content
soup = BeautifulSoup(html, 'lxml')
```

## Search for First Tag by Name

To search the first tag by name, we use function `find` and pass in the tag name. For example following code return the first `p` tag

```
first_p_tag = soup.find('p')
print(first_p_tag)
```

Result print out

```
"""
<p class="title">
 The Dormouse's story
</p>
"""
```

## Search for All Tags by Name

To search for all tags by name, we use function `find_all` then pass in the tag name. For example following code return all `a` tags in a list.

```
a_tags = soup.find_all('a')
print(a_tags)
```

Result when print out

```
"""
[
Elsie,
Lacie,
Tillie
]
"""
```

## Search with Tag Name and Class Attribute

In real project, many time we want to search for a tag with specific class name. To do that we put in tag name and `class_` parameter.

```
p_tag = soup.find('p', class_ = 'story')
print(p_tag)
```

Above code will return the first `p` tag which have `class` is "story"

```
....
<p class="story">
 Once upon a time there were three little sisters; and their names were
 Elsie,
 Lacie and
 Tillie;
 and they lived at the bottom of a well.
</p>
....
```

## Search with Tag Name and Other Attribute

Beside of using `class` attribute for search, we could use any other attribute. To do that we put in tag name and attribute parameter with grammar `{'attribute_name': attribute_value}`

For example following code will find first `a` tag which have `id` is "link1"

```
a = soup.find('a', {'id':'link1'})
print(a)
```

And it will return

```
....
Elsie
....
```

## Search with Tag Name and Text Inside

Another way to search is using tag name and text inside that tag. To do that we use `string` parameter.

For example following code will return all `a` tag which has text inside is "Elsie"

```
a_elsie = soup.find_all('a', string = 'Elsie')
print(a_elsie)
```

And it will return

```
....
[Elsie]
....
```

## Search Scope

You could see that HTML content page is structured in tree. So after find out the parent tag, we can find out child inside that parent.

For example following code will return the first `b` tag inside the first `p` tag.

```

first_p = soup.find('p')
first_b_inside_first_p = first_p.find('b')

print(first_b_inside_first_p)

```

And it print out

```

"""
The Dormouse's story
"""

```
layout: default
---


### Scrape Data from Tag

```

Congratulations, so you come to the final step. And at this step we will harvest our result.

For easy of demo, let `continue` using the following soup.

```

```python
import beautifulsoup object
from bs4 import BeautifulSoup

predefine a html content
html = """
<html>
<head>
 <title>The Dormouse's story</title>
</head>
<body>
 <p class="title">
 The Dormouse's story
 </p>
 <p class="story">
 Once upon a time there were three little sisters. And their names were
 Elsie,
 Lacie
 and
 Tillie;
 and they lived at the bottom of a well.
 </p>
 <p class="story">...</p>
</body>
</html>
"""

create soup object by HTML content
soup = BeautifulSoup(html, 'lxml')

```

## Scrape for text

To scrape the text inside a tag we just need to use grammar `tagname.text`

As following code example

```
p = soup.find('p')
print(p.text)
```

will return result

```
.....
The Dormouse's story
.....
```

## Scrape for link

To scrape the link inside `a` tag, we just need to access attribute `href`

Following code will print out all link inside a tags.

```
a_tags = soup.find_all('a')
for a in a_tags:
 print(a['href'])
```

And we have result

```
.....
http://example.com/elsie
http://example.com/lacie
http://example.com/tillie
.....
```

---

## layout: default

---

### NBA projects introduction

[NBA](#) is popular site when you want to find information about men's professional basketball league in North America. In this chapter we will do 4 difference missions to scraping data on this site.

- scrape name of all players
- scrape name and detail link of all players
- scrape detail information of all players
- scrape image of all players

### Mission 1 : scrape name of all active player

Let access url <https://stats.nba.com/players/list/> , you will see full list of player order in alphabet list.

Last Initial	First Name	Middle Name	Last Name	Position
A	Abrines, Alex		Aldridge, LaMarcus	Aminu, Al-Farouq
	Acy, Quincy		Alexander, Kyle	Anderson, Justin
	Adams, Jaylen		Alexander-Walker, Nickell	Anderson, Kyle
	Adams, Steven		Alkins, Rawle	Anderson, Ryan
	Adebayo, Bam		Allen, Grayson	Anigbogu, Ike
	Adel, Deng		Allen, Jarrett	Antetokounmpo, Giannis
	Akoon-Purcell, DeVaughn		Allen, Kadeem	Antetokounmpo, Kostas
B	Bacon, Dwyane		Beasley, Michael	Bogdanovic, Bojan
	Bagley III, Marvin		Belinelli, Marco	Bogut, Andrew
	Baker, Ron		Bell, Jordan	Bol, Bol
	Baldwin IV, Wade		Bembry, DeAndre'	Bolden, Jonah
	Ball, Lonzo		Bender, Dragan	Bone, Jordan
	Bamba, Mo		Bertans, Dairis	Bonga, Isaac
	Barea, J.J.		Bertans, Davis	Booker, Devin
	Barnes, Harrison		Beverley, Patrick	Boucher, Chris
	Barrett, RJ		Birch, Khem	Bowen II, Brian

Right click and select 'Inspect' you will open chrome developer tool panel

The screenshot shows the NBA.com/Stats Players List page with the Chrome Developer Tools open. The 'Elements' tab is active, and the 'stats-player-list' class is selected in the DOM tree. The right panel displays the HTML structure of the selected element, showing nested 

 and - tags for the player list.

Now move the mouse around and find the tag which contain all names inside it. You will find out that is `div` tag with `class` name is `stats-player-list`

The screenshot shows the NBA Advanced Stats Players List page. The developer tools are open, highlighting the `div.stats-player-list.players-list` element. This element contains a list of player names, each enclosed in a `li` tag with the class `players-list__name`. The browser's address bar shows the URL `https://stats.nba.com/players/list/`.

Now mouse over one name and inspect, you will see that it inside `li` tag with class `players-list__name`

The screenshot shows the NBA Advanced Stats Players List page. A green arrow points from the highlighted `li.players-list_name` element in the developer tools to the same element on the page. The developer tools show the expanded HTML structure of the list item, revealing the `a` tag with the href attribute set to `/player/203518/Abrines, Alex`.

So after inspection we have following strategy to scrape all name.

- search for `div` tag with `class` is `stats-player-list`
- search all `li` tags with class `players-list__name` inside above `div` tag
- scrape text inside each `li` tag above

Now let really do it

```
from selenium import webdriver
from bs4 import BeautifulSoup

driver =
webdriver.Chrome(executable_path=r'C:\chromedriver_win32\chromedriver.exe')

url = 'https://stats.nba.com/players/list/'
driver.get(url)

soup = BeautifulSoup(driver.page_source, 'lxml')
```

```

div = soup.find('div', class_= 'stats-player-list')

for a in div.find_all('a'):
 print(a.text)

driver.quit()

```

And at out put you will have 638 players

```

Abrines, Alex
Acy, Quincy
Adams, Jaylen
Adams, Steven
Adebayo, Bam
Adel, Deng
Akoon-Purcell, DeVaughn
Aldridge, LaMarcus
Alexander, Kyle
...

```

## Mission 2 : scrape name and detail link of all active player (try it your self)

In this mission we scrape name and correspond link to detail information of that player.

Final result should look like

```

Abrines, Alex
/player/203518/

```

```

Acy, Quincy
/player/203112/

```

```

Adams, Jaylen
/player/1629121/

```

## Mission 3 : scrape detail information for each player (try it your self)

When click to detail link, you will know player detail information

HT	WT	PRIOR	PTS	REB	AST	PIE
6-5	225 lbs	University of Arizona/USA	3.7	2.6	1.3	5.1
AGE	BORN	DRAFT				
21 2740	10/29/1997	Undrafted	1 yrs			

In this mission, we will need to scrape detail data for each player like : name, PTS, REB, AST, PIE

The result should be

```
name : Abrines, Alex
link : https://stats.gleague.nba.com/player/203518/
pts : 0.0
reb : 0.0
ast : 0.0
pie : 0.0

name : Acy, Quincy
link : https://stats.gleague.nba.com/player/203112/
pts : 21.7
reb : 8.0
ast : 1.3
pie : 12.0

...
```

## Mission 4 : scrape image for all players (try it your self)

Now we want to get image for each player



Result should as following video

---

## layout: default

### Steam projects

Steam is a video game digital distribution platform developed by Valve Corporation. We have 2 mission when scrape data from Steam

- Scrape top selling items
- Scrape top selling items with detail information description
- Login to steam before doing scraping

## Mission 1 : scrape top selling items

Let access [this url](#)

We will see a table list out the top seller

The screenshot shows the Steam Store's 'TOP SELLING' page. It displays a list of 15 games, each with a thumbnail, title, release date, discount percentage, and price. The games include Grand Theft Auto V, Oxygen Not Included, Stay the Spire, Survivor Pass 4: Aftermath, ARK: Survival Evolved, PLAYERUNKNOWN'S BATTLEGROUNDS, Warhammer 40,000: Inquisitor - Martyr, Hide Or Die, Grand Theft Auto V: Premium Online Edition, NieR Automata™, Shadow of the Tomb Raider, Hunt: Showdown, and HUNT: The Tрошада.

Now our job is scrape title and link of top selling at first page.

Let's do some inspecting and we found that `div` contain all top selling items has

`id=search_result_container`

The screenshot shows the same Steam Store page with developer tools (F12) open. The 'Elements' tab is selected, and a green arrow points to the `div id="search_result_container"` element in the DOM tree. This div contains the list of top-selling games.

And then each game is inside `a` tag with `class` is `search_result_row`

The screenshot shows the Steam store's top sellers page. A green arrow points from the search bar at the top left to the browser's DOM inspector window on the right. The DOM inspector highlights the search results container, specifically the part where game tiles are listed.

And inside each game to get the game tile, we need to find for `span` tag with class `title`.

This screenshot is similar to the one above, but it focuses on a specific game tile for "Grand Theft Auto V". A green arrow points from the search bar to the DOM inspector, which is now highlighting the HTML for the "Grand Theft Auto V" listing. The `span` tag with the `title` class is visible within the game's listing.

That it, now let do the code.

```
from selenium import webdriver
from bs4 import BeautifulSoup
import requests

driver =
webdriver.Chrome(executable_path=r'C:\chromedriver_win32\chromedriver.exe')

def top_seller():
 # search web scraping
 url = 'https://store.steampowered.com/search/?filter=topsellers&os=win'
 driver.get(url)

 soup = BeautifulSoup(driver.page_source, 'lxml')
```

```

search for div contain all game
div = soup.find('div', {'id':'search_result_container'})

search for a contain one game
for a in div.find_all('a', class_='search_result_row'):

 # search for title
 span_name = a.find('span', class_='title')
 print(span_name.text)
 print(a['href'])
 print('\n')

topSeller()

driver.close()

```

and when running, we should see result

```

.....
Grand Theft Auto V
https://store.steampowered.com/app/271590/Grand_Theft_Auto_V/?
snr=1_7_7_topsellers_150_1

Oxygen Not Included
https://store.steampowered.com/app/457140/Oxygen_Not_Included/?
snr=1_7_7_topsellers_150_1

Slay the Spire
https://store.steampowered.com/app/646570/slay_the_spire/?
snr=1_7_7_topsellers_150_1
.....

```

## Mission 2 : scrape top selling items with detail information (try it your self)

Try click to one game item, we will go to detail page. In this page have session to have detail information

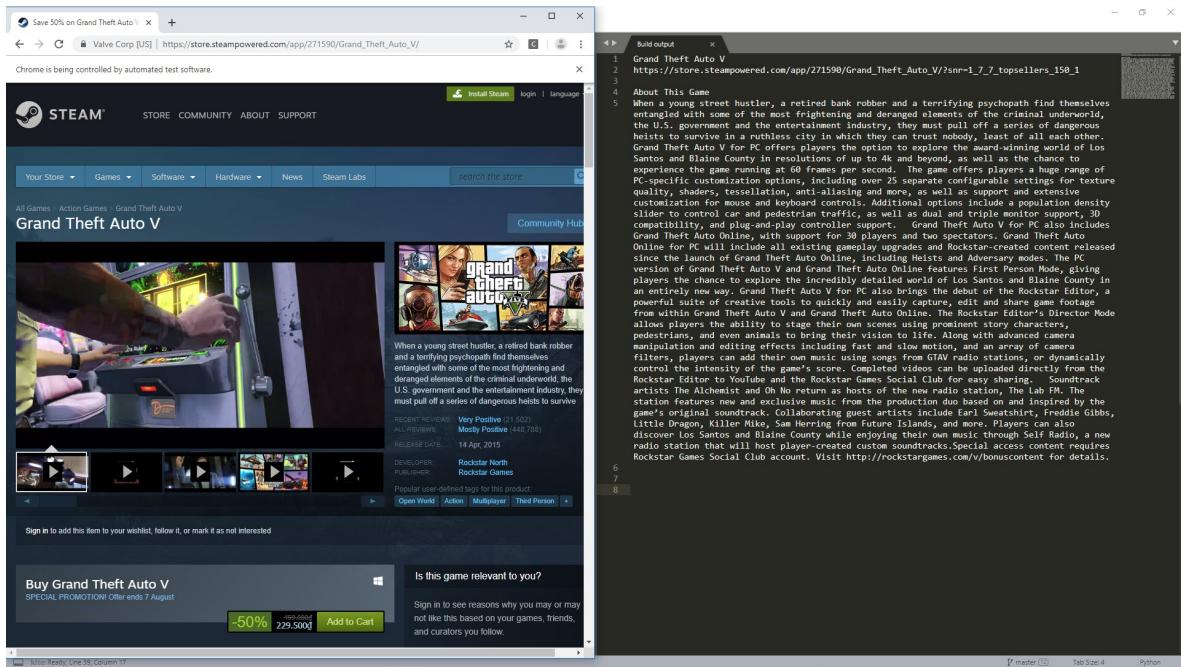
### ABOUT THIS GAME

When a young street hustler, a retired bank robber and a terrifying psychopath find themselves entangled with some of the most frightening and deranged elements of the criminal underworld, the U.S. government and the entertainment industry, they must pull off a series of dangerous heists to survive in a ruthless city in which they can trust nobody, least of all each other.

Grand Theft Auto V for PC offers players the option to explore the award-winning world of Los Santos and Blaine County in resolutions of up to 4k and beyond, as well as the chance to experience the game running at 60 frames per second.

Let scrape this section for each game. Please try this your self first

When running your code result should some thing like below video



## Mission 3 : login to steam before do scraping (try it your self)

In mission 2 you will see from result that some of game require you above age of 18 in order to view description.

To solve this issue we will try to login to steam before doing scraping.

## layout: default

### IMDb projects

IMDb (Internet Movie Database) is an online database of information related to films, television programs, home videos, video games, and streaming content online – including cast, production crew and personal biographies, plot summaries, trivia, fan and critical reviews, and ratings.

We will do following tasks when scrape IMDb site :

- Scrape list of best ever movies
- Scrape best movies with detail information
- Scrape large posters for best movies

## Mission 1 : scrape list of best ever movies

Access this [url](#) , a list of 250 best ever movies will show up. In this mission, we will try to scrape all of this movie title.

The screenshot shows the IMDb Top Rated Movies chart page. At the top, there's a search bar and navigation links for 'Movies, TV & Showtimes', 'Celebs, Events & Photos', 'News & Community', and 'Watchlist'. A 'Sign in' button is also present. The main content area is titled 'Top Rated Movies' and shows the top 250 movies as rated by IMDb users. The list includes titles like 'The Shawshank Redemption', 'The Godfather', 'The Dark Knight', etc., each with a small poster, rating (e.g., 9.2), and a 'SHARE' button. To the right, there's a sidebar titled 'IMDb Charts' with links to various charts like 'Box Office', 'Most Popular Movies', and 'Top Rated English Movies'. Below the sidebar is a section titled 'Top Rated Movies by Genre' with a list of genres.

Let do some inspection, we easy see that entire list is inside `table` tag with class `chart`

This screenshot shows the same IMDb page with the browser's developer tools open, specifically the 'Elements' tab. A green arrow points from the text in the previous step to the 'Elements' panel. In the panel, the HTML code for the top-rated movies list is visible, showing a `table` element with the class `chart full-width`. The table has a header row with columns for 'Rank & Title', 'IMDb Rating', and 'Your Rating'. Below the header, there are 10 rows, each representing a movie from the list. The first row is highlighted with a blue background. The developer tools also show the entire page structure, including the header, sidebar, and footer.

And each individual movie is inside `td` tag with class `titleColumn`

So we have following code to scrape all movies title

```

from bs4 import BeautifulSoup
from selenium import webdriver

chrome_driver_path = r'C:\chromedriver_win32\chromedriver.exe'
driver = webdriver.Chrome(executable_path=chrome_driver_path)

url = 'https://www.imdb.com/chart/top?ref_=nv_mv_250'
driver.get(url)
soup = BeautifulSoup(driver.page_source, 'lxml')

search for table contain all title
table = soup.find('table', class_ = 'chart')

search for each individual title
for td in table.find_all('td', class_ = 'titleColumn'):

 # get title and clean it up
 print(td.text.strip().replace('\n', '').replace(' ', ''))

```

And then running above code we will have following result

```

.....
1.The Shawshank Redemption(1994)
2.The Godfather(1972)
3.The Godfather: Part II(1974)
4.The Dark Knight(2008)
5.12 Angry Men(1957)
6.Schindler's List(1993)
7.The Lord of the Rings: The Return of the King(2003)
8.Pulp Fiction(1994)
9.The Good, the Bad and the Ugly(1966)
10.Fight Club(1999)
.....

```

## Mission 2 : scrape list of best ever movies with detail infor (try it your self)

From above full title we could extract following information

- Rank
- Movie title
- Year of show
- Link to detail page

Let try your self to scrape all above information. And out put print out should as following

```
#####
rank : 1
title : The Shawshank Redemption
year : 1994
link : /title/tt0111161/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=e31d89dd-322d-4646-
8962-327b42fe94b1&pf_rd_r=5J8K5RAZ3N1Q80SH0JMQ&pf_rd_s=center-
1&pf_rd_t=15506&pf_rd_i=top&ref_=chttp_tt_1

#####
rank : 2
title : The Godfather
year : 1972
link : /title/tt0068646/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=e31d89dd-322d-4646-
8962-327b42fe94b1&pf_rd_r=5J8K5RAZ3N1Q80SH0JMQ&pf_rd_s=center-
1&pf_rd_t=15506&pf_rd_i=top&ref_=chttp_tt_2
#####
```

## Mission 3 : Scrape large poster for all movies (try it your self)

With each individual movie, when click to detail link, detail page will show up. In detail page contain large poster that we want to scrape in this mission.



Let's try your self to scrape these posters, it will be very interesting. Result should as below

# layout: default

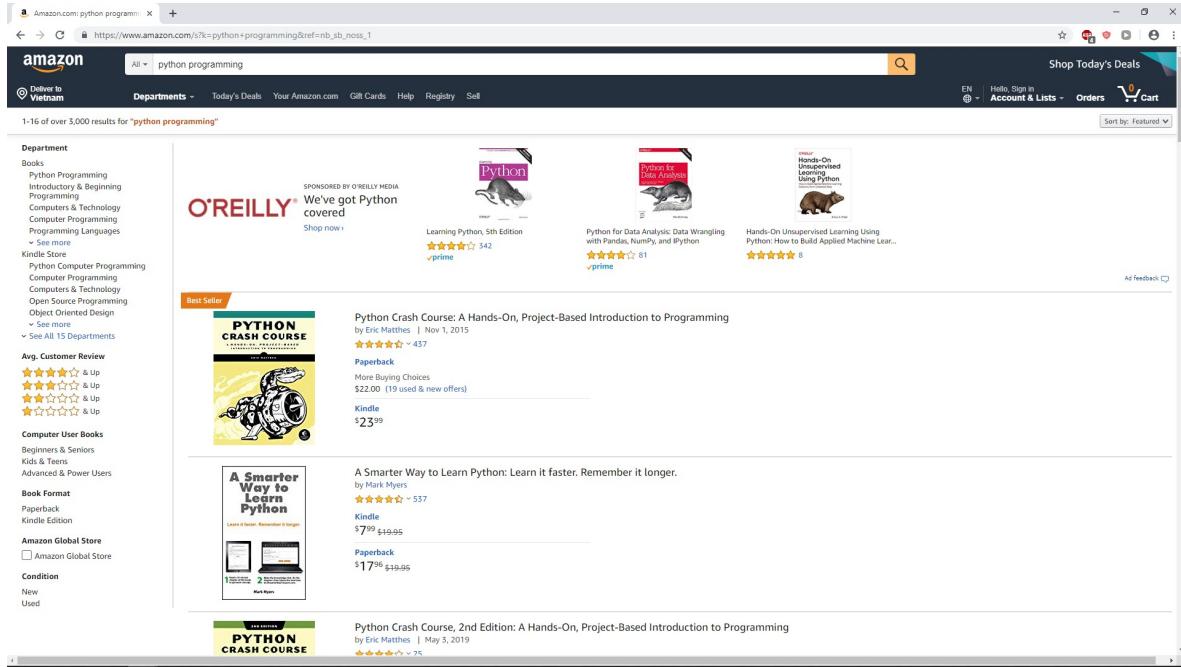
## Amazon projects

Amazon is largest ecommerce site in the world. We will do following web scraping task on Amazon:

- Scrape book list "python programming", get book title and link to detail page.
- Scrape detail information for each book in side "python programming" list
- Scrape customer comment for each book inside "python programming" list
- Scrape search result from multiple pages

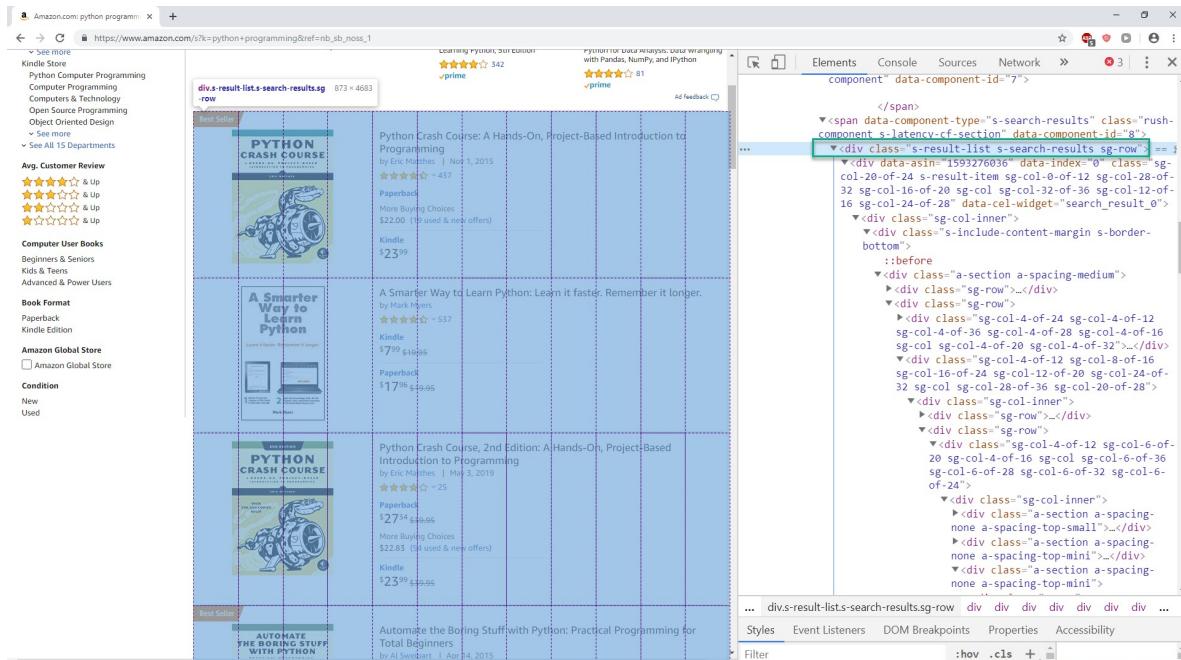
### Mission 1 : scrape title and detail link

Access the [link](#) and you will see a list of python programming book



The screenshot shows the Amazon search results for "python programming". The search bar at the top has "python programming" entered. Below the search bar, there are several filters and categories on the left, including "Department" (Books, Python Programming, Introductory & Beginning Programming, Computers & Technology, Computer Programming, Programming Languages, Kindle Store, Python Computer Programming, Computer Programming, Computers & Technology, Open Source Programming, Object Oriented Design, See more, and "See All 15 Departments"). There are also sections for "Avg. Customer Review" (4.5 stars & up, 4.5 stars & up, 4.5 stars & up, 4.5 stars & up, 4.5 stars & up), "Computer User Books" (Beginners & Seniors, Kids & Teens, Advanced & Power Users), "Book Format" (Paperback, Kindle Edition), and "Amazon Global Store" (Amazon Global Store). The main content area displays a grid of book results. One book, "Python Crash Course: A Hands-On, Project-Based Introduction to Programming" by Eric Matthes, is highlighted as a "Best Seller". Other books shown include "Python", "Python for Data Analysis", "Hands-On Unsupervised Learning Using Python", "Python Crash Course, 2nd Edition", "A Smarter Way to Learn Python", and "Automate the Boring Stuff with Python". Each book listing includes the title, author, price, and a "View Details" button.

Let do some inspection on this page, we see that `div` with class `s-result-list` contain all books



The screenshot shows the same Amazon search results page with developer tools (Elements tab) open in the browser. The cursor is hovering over the first book listing for "Python Crash Course: A Hands-On, Project-Based Introduction to Programming". The developer tools highlight the `div` element with the class `s-result-list`. The full DOM structure for this book listing is visible on the right side of the developer tools interface, showing nested `div`, `span`, and `a` elements. The book's title, author, price, and other details are also present in the DOM structure.

Now to access each book, we see that `a` tag with class `a-link-normal a-text-normal` contain book title and link to detail for each book

The screenshot shows a search results page on Amazon for the keyword "python+programming". Four books are listed:

- Python Crash Course: A Hands-On, Project-Based Introduction to Programming** by Eric Matthes (Nov 1, 2015) - \$22.00
- A Smarter Way to Learn Python** by Mark Myers (Kindle Edition) - \$7.99
- Python Crash Course, 2nd Edition: A Hands-On, Project-Based Introduction to Programming** by Eric Matthes (May 3, 2019) - \$27.94
- Automate the Boring Stuff with Python: Practical Programming for Total Beginners** by Al Sweigart (Apr 14, 2015) - \$23.99

The developer tools (Elements tab) are open, highlighting the first book's title and link (`a.a-link-normal.a-text-normal`) in the DOM tree.

So we have following code to scrape book title and link to detail page

```
from bs4 import BeautifulSoup
from selenium import webdriver

chrome_driver_path = r'C:\chromedriver_win32\chromedriver.exe'
driver = webdriver.Chrome(executable_path=chrome_driver_path)

def get_book_list():
 url = 'https://www.amazon.com/s/ref=nb_sb_noss_1?url=search-
alias%3Daps&field-keywords=python+programming'
 driver.get(url)

 soup = BeautifulSoup(driver.page_source, 'lxml')
 div = soup.find('div', class_='s-result-list')

 for a in div.find_all('a', class_ = 'a-link-normal a-text-normal'):
 print('title : ',a.text.replace('\n', ''))
 print('link : ', a['href'])
 print('\n')
 print('\n')

get_book_list()

driver.close()
```

After running we will have book title and link print out as below

title : Python Crash Course: A Hands-On, Project-Based Introduction to Programming  
link : /Python-Crash-Course-Hands-Project-Based/dp/1593276036/ref=sr\_1\_1?keywords=python+programming&qid=1564924835&s=gateway&sr=8-1

title : A Smarter Way to Learn Python: Learn it faster. Remember it longer.  
link : /Smarter-Way-Learn-Python-Remember-ebook/dp/B077Z55G3B/ref=sr\_1\_2?keywords=python+programming&qid=1564924835&s=gateway&sr=8-2

title : Python Crash Course, 2nd Edition: A Hands-On, Project-Based Introduction to Programming  
link : /Python-Crash-Course-2nd-Edition/dp/1593279280/ref=sr\_1\_3?keywords=python+programming&qid=1564924835&s=gateway&sr=8-3

.....

## Mission 2 : scrape detail infor for each book (try it your self)

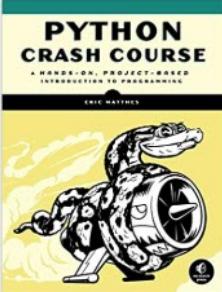
When click to each book, We will go to detail page, in this page for example we want to scrape for ISBN number and the book description.

Back to results

**Look inside** ↗

**PYTHON CRASH COURSE**  
A HANDS-ON, PROJECT-BASED INTRODUCTION TO PROGRAMMING

ERIC MATTHES



See all 9 images

There is a newer edition of this item:

**Python Crash Course, 2nd Edition: A Hands-On, Project-Based Introduction to Programming** Paperback – November 1, 2015

by Eric Matthes (Author)  
★ ★ ★ ★ ★ (437 customer reviews)

#1 Best Seller in Teen & Young Adult Computer Programming

See all 2 formats and editions

Kindle \$23.99 Paperback from \$22.00

Read with Our Free App 14 Used from \$22.00 5 New from \$26.63

There is a newer edition of this item:  
**Python Crash Course, 2nd Edition: A Hands-On, Project-Based Introduction to Programming** Paperback – November 1, 2015

\$27.54 ★ ★ ★ ★ ★ (25) In Stock.

Python Crash Course is a fast-paced, thorough introduction to Python that will have you writing programs, solving problems, and making things that work in no time.

In the first half of the book, you'll learn about basic programming concepts, such as lists, dictionaries, classes, and loops, and practice writing clean and readable code with exercises for each topic. You'll

+ Read more

Follow the Author

Eric Matthes + Follow

### Editorial Reviews

#### Review

Recommended reading for a "shining tech career" by Techradar India

Python Crash Course was selected as one of the best books for learning Python by Real Python  
"It has been interesting to see, over the last few years, No Starch Press, which produces this book, growing and producing future classics that should be alongside the more traditional O'Reilly Press programming books. Python Crash Course is one of those books."  
—Greg Laden, ScienceBlogs

"All of these projects are well thought out and presented in such a way that learning the subject matter and implementing it is much more an enjoyable pastime rather than an onerous task that must be completed. Eric took the time to deal with some rather complex projects and lay them out in a consistent, logical and pleasant manner that draws the reader into the subject willingly, which unfortunately, many authors fail to do."  
—Full Circle Magazine

"The book is well presented with good explanations of the code snippets. It works with you, one small step at a time, building more complex code, explaining what's going on all the way."  
—flickThrough Reviews

"Learning Python with Python Crash Course was an extremely positive experience! A great choice if you're new to Python."  
—Mikke Goes Coding

#### About the Author

Eric Matthes is a high school science and math teacher living in Alaska where he teaches Introduction to Python. He has been writing programs since he was five years old.

### Product details

Paperback: 560 pages

Publisher: No Starch Press; 1 edition (November 1, 2015)

Language: English

**ISBN-10:** 1593276036

**ISBN-13:** 978-1593276034

Product Dimensions: 7 x 1.3 x 9.2 inches

Shipping Weight: 2.4 pounds

Average Customer Review: ★ ★ ★ ★ ★ (437 customer reviews)

Amazon Best Sellers Rank: #10,067 in Books (See Top 100 in Books)

#16 in Introductory & Beginning Programming

#1 in Teen & Young Adult Computer Programming

#5 in STEM Education

When running your code, console result should as below

```
....
isbn : 1593276036
description : Python Crash Course is a fast-paced, thorough introduction to
Python that will
have you writing programs, solving problems, and making things that work in no
time. In the first half of the book, you'll learn about basic programming
concepts, such as lists, dictionaries, classes, and loops, and practice
writing clean and readable code with exercises for each topic. You'll also
learn how to make your programs interactive and how to test your code safely
before adding it to a project. In the second half of the book, you'll put your
new knowledge into practice with three substantial projects: a Space
Invaders-inspired arcade game, data visualizations with Python's super-handy
libraries, and a simple web app you can deploy online. As you work through
Python Crash Course you'll learn how to:-use powerful Python libraries and
tools, including matplotlib, NumPy, and Pygal-Make 2D games that respond to
keypresses and mouse clicks, and that grow more difficult as the game
progresses-work with data to generate interactive visualizations-Create and
customize web apps and deploy them safely online-Deal with mistakes and errors
so you can solve your own programming problemsIf you've been thinking
seriously about digging into programming, Python Crash Course will get you up
to speed and have you writing real programs fast. Why wait any longer? Start
your engines and code!uses Python 2 and 3
....
```

### **Mission 3 : scrape customer comment for each book (try it your self)**

In this mission, we also scrape inside detail page, but we want to get comment from reader. Just the most valueable comment. It is the top comment.

## Read reviews that mention



Showing 1-8 of 437 reviews

Top Reviews ▾



Megan

★★★★★ I'm taking programming courses to complete a teaching authorization in computer science and this book is 10x better than the required text

January 11, 2018

Format: Paperback | Verified Purchase

This book is a life-saver! I'm taking programming courses to complete a teaching authorization in computer science and this book is 10x better than the required text. Each programming concept is explained well and followed by a chunk of code that has a line-by-line narrative explanation of what the code does. At the end of each section, there are "Try It Yourself" mini-projects to apply your learning. They can take anywhere from 5-45 minutes to complete but they are perfect for checking your own understanding of each concept. If/when I teach an Intro to Python course, this will be MY required text!

59 people found this helpful

Helpful

| Comment | Report abuse

When running your code it should print out as below

```
....
comment : This book is a life-saver! I'm taking programming courses to complete
a
teaching authorization in computer science and this book is 10x better than
the required text. Each programming concept is explained well and followed by
a chunk of code that has a line-by-line narrative explanation of what the code
does. At the end of each section, there are "Try It Yourself" mini-projects to
apply your learning. They can take anywhere from 5-45 minutes to complete but
they are perfect for checking your own understanding of each concept. If/when
I teach an Intro to Python course, this will be MY required text!
....
```

## Mission 4 : scrape book from multiple pages (try it your self)

We could see that search result for "python programming" contain multiple pages (here we see that have 20 pages).

In this mission we will try to get book infor from all pages

Amazon.com: python programming

<https://www.amazon.com/s?k=python+programming>

**PYTHON PROGRAMMING**  
A Step By Step Guide From Beginner To Expert  
Anthony Brun

Advanced)  
by Anthony Brun | Jan 27, 2019  
★ ★ ★ ★ ★ ~ 194  
paperback  
\$16<sup>69</sup> \$17.99  
More Buying Choices  
\$11.87 (12 used & new offers)  
Kindle  
\$0<sup>00</sup> KindleUnlimited  
Free with Kindle Unlimited membership  
Or \$2.99 to buy

Inspired by your views

**PYTHON TRICKS THE BOOK**  
A Buffet Of Awesome Python Features  
by Dan Bader  
★ ★ ★ ★ ★ ~ 202  
Kindle Edition  
\$4<sup>99</sup> \$29.99

**PYTHON CRASH COURSE**  
A Hands-On, Project-Based Introduction to Programming  
by Eric Matthes  
★ ★ ★ ★ ★ ~ 437  
Kindle Edition  
\$23<sup>99</sup>

**A Smarter Way to Learn Python**  
Learn it faster. Remember it longer.  
by Mark Myers  
★ ★ ★ ★ ★ ~ 537  
Paperback  
\$17<sup>99</sup> \$19.95  
Kindle Edition  
\$17<sup>99</sup> \$29.95  
prime

**AUTOMATE THE BORING STUFF WITH PYTHON**  
Practical Programming for Total Beginners  
by Al Sweigart  
★ ★ ★ ★ ★ ~ 346  
Paperback  
\$17<sup>99</sup> \$19.95  
Kindle Edition  
\$17<sup>99</sup> \$29.95

**PYTHON PROGRAMMING**  
A Practical Introduction To Python Programming For Total Beginners  
by Jason Rees  
★ ★ ★ ★ ★ ~ 106  
Kindle Edition  
\$0<sup>00</sup> KindleUnlimited  
Free with Kindle Unlimited membership

**Learn PYTHON 3 the HARD WAY**  
Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code (Zed Shaw's...  
by Zed A. Shaw  
★ ★ ★ ★ ★ ~ 72  
Paperback  
\$17<sup>94</sup> \$39.99  
Kindle  
prime

— Previous 1 2 3 ... 20 Next —

Tell us how we can improve  
If you need help, please visit the help section or contact us