



Web Scraping



Complete Python Bootcamp

- Web scraping is a general term for techniques involving automating the gathering of data from a website.
- In this section we will learn how to use Python to conduct web scraping tasks, such as downloading images or information off a website.



Complete Python Bootcamp

- In order to web scrape with Python we need to understand the basic concepts of how a website works.
- When a browser loads a website, the user gets to see what is known as the “front-end” of the website.



Complete Python Bootcamp



WIKIPEDIA
The Free Encyclopedia



Complete Python Bootcamp



WIKIPEDIA
The Free Encyclopedia





Complete Python Bootcamp



WIKIPEDIA
The Free Encyclopedia





Complete Python Bootcamp



WIKIPEDIA
The Free Encyclopedia





Complete Python Bootcamp

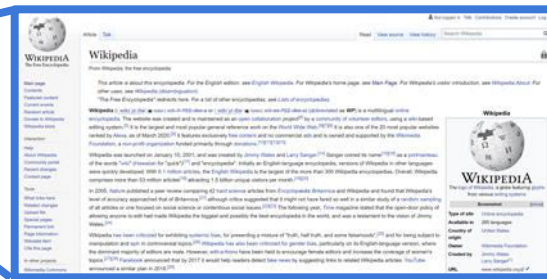


WIKIPEDIA
The Free Encyclopedia





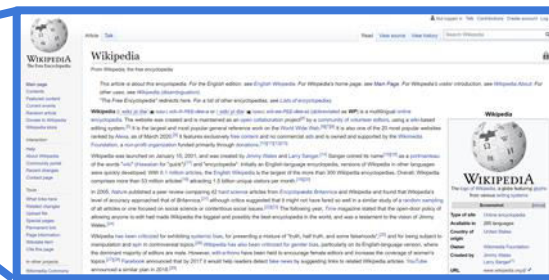
Complete Python Bootcamp





Complete Python Bootcamp

```
<!DOCTYPE html>
<html>
  <head>
    <title>Title on
Browser Tab</title>
  </head>
  <body>
    <h1> Website
Header </h1>
    <p> Some
Paragraph </p>
  </body>
</html>
```





Complete Python Bootcamp

```
<!DOCTYPE html>
<html>
  <head>
    <title>Title on
    Browser Tab</title>
  </head>
  <body>
    <h1> Website
    Header </h1>
    <p> Some
    Paragraph </p>
  </body>
</html>
```

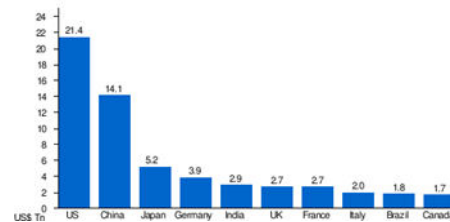


WIKIPEDIA
The Free Encyclopedia



Complete Python Bootcamp

```
<!DOCTYPE html>
<html>
  <head>
    <title>Title on
    Browser Tab</title>
  </head>
  <body>
    <h1> Website
    Header </h1>
    <p> Some
    Paragraph </p>
    <body>
  </html>
```





Complete Python Bootcamp

```
<!DOCTYPE html>
<html>
  <head>
    <title>Title on
Browser Tab</title>
  </head>
  <body>
    <h1> Website
Header </h1>
    <p> Some
Paragraph </p>
  </body>
</html>
```



[“Germany”, “France”, “Spain”]



Complete Python Bootcamp

- Main things we need to understand
 - Rules of Web Scraping
 - Limitations of Web Scraping
 - Basic HTML and CSS



Complete Python Bootcamp

- Rules of Web Scraping
 - Always try to get permission before scraping!
 - If you make too many scraping attempts or requests your IP Address could get blocked!
 - Some sites automatically block scraping software.



Complete Python Bootcamp

- Limitations of Web Scraping
 - In general every website is unique, which means every web scraping script is unique.
 - A slight change or update to a website may completely break your web scraping script.



Complete Python Bootcamp

Main front end components of a website

```
<!DOCTYPE html>
<html>
  <head>
    <title>Title on Browser
    Tab</title>
  </head>
  <body>
    <h1> Website Header </h1>
    <p> Some Paragraph </p>
  </body>
</html>
```

HTML



```
p{
  color: red;
  font-family: courier;
  font-size: 160%;
}
.someclass{
  color: green;
  font-family: verdana;
  font-size: 300%;
}
#someid{
  color: blue;
}
```

CSS



```
var values = ["Volvo", "Saab",
"Fiat"];

var person = {
  firstName: "John",
  lastName: "Doe",
  age: 50,
  eyeColor: "blue"
};
```

JS



Complete Python Bootcamp

- When viewing a website, the browser doesn't show you all the source code behind the website, instead it shows you the HTML and some CSS and JS that the website sends to your browser.



Complete Python Bootcamp

- HTML is used to create the basic structure and content of a webpage
- CSS is used for the design and style of a web page, where elements are placed and how it looks
- JavaScript is used to define the interactive elements of a webpage



Complete Python Bootcamp

- For effective basic web scraping we only need to have a basic understanding of HTML and CSS.
- Python can view these HTML and CSS elements programmatically, and then extract information from the website.
- Let's explore HTML and CSS in more detail.



Complete Python Bootcamp

- HTML is Hypertext Markup Language and is present on every website on the internet.
- You can right-click on a website and select “View Page Source” to get an example.
- Let’s see a small example of HTML code.



Complete Python Bootcamp

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>Title on Browser Tab</title>
```

```
  </head>
```

```
  <body>
```

```
    <h1> Website Header </h1>
```

```
    <p> Some Paragraph </p>
```

```
  </body>
```

```
</html>
```



Complete Python Bootcamp

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>Title on Browser Tab</title>
```

```
  </head>
```

```
  <body>
```

```
    <h1> Website Header </h1>
```

```
    <p> Some Paragraph </p>
```

```
  </body>
```

```
</html>
```



Complete Python Bootcamp

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>Title on Browser Tab</title>
```

```
  </head>
```

```
  <body>
```

```
    <h1> Website Header </h1>
```

```
    <p> Some Paragraph </p>
```

```
  </body>
```

```
</html>
```




Complete Python Bootcamp

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>Title on Browser Tab</title>
```

```
  </head>
```

```
  <body>
```

```
    <h1> Website Header </h1>
```

```
    <p> Some Paragraph </p>
```

```
  </body>
```

```
</html>
```



Complete Python Bootcamp

- CSS stands for Cascading Style Sheets.
- CSS gives “style” to a website, such as changing colors and fonts.
- CSS uses tags to define what html elements will be styled.



Complete Python Bootcamp

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" href="styles.css">
        <title>Some Title</title>
    </head>
    <body>
        <p id='para2'> Some Text </p>
    </body>
</html>
```



Complete Python Bootcamp

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<link rel="stylesheet" href="styles.css">
```

```
<title>Some Title</title>
```

```
</head>
```

```
<body>
```

```
<p id='para2'> Some Text </p>
```

```
</body>
```

```
</html>
```



Complete Python Bootcamp

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" href="styles.css">
        <title>Some Title</title>
    </head>
    <body>
        <p id='para2'> Some Text </p>
    </body>
</html>
```



Complete Python Bootcamp

Example of the style.css file:

```
#para2 {  
    color: red;  
}
```



Complete Python Bootcamp

```
<!DOCTYPE html>
```

```
<html>
```

```
    <head>
```

```
    <link rel="stylesheet" href="styles.css">
```

```
    <title>Some Title</title>
```

```
  </head>
```

```
  <body>
```

```
    <p class='cool'> Some Text </p>
```

```
  </body>
```

```
</html>
```



Complete Python Bootcamp

Example of the style.css file:

```
.cool {  
    color: red;  
    font-family: verdana;  
}
```




Complete Python Bootcamp

```
p{
    color: red;
    font-family: courier;
    font-size: 160%;
}
.someclass{
    color: green;
    font-family: verdana;
    font-size: 300%;
}
#someid{
    color: blue;
}
```



Complete Python Bootcamp

- Don't worry about memorizing this! We'll see lots of examples, main ideas to note:
 - HTML contains the information
 - CSS contains the styling
 - We can use HTML and CSS tags to locate specific information on a page



Complete Python Bootcamp

- To web scrape with Python we can use the BeautifulSoup and requests libraries.
- These are external libraries outside of Python so you need to install them with either conda or pip at your command line.



Complete Python Bootcamp

- Directly at your command line use:
 - `pip install requests`
 - `pip install lxml`
 - `pip install bs4`
- Or for Anaconda distributions, use `conda install` instead of `pip install`.



Complete Python Bootcamp

- Let's work through some examples of web scraping with Python!



Setting Up For Web Scraping



Complete Python Bootcamp

- Install the necessary libraries
- Explore how to inspect elements and view source of a webpage
- Note: We will suggest you use Chrome so you can follow along exactly as we do, but these tools are available in all major browsers.



Grabbing a Page Title



Grabbing All Elements of a Class



Complete Python Bootcamp

- We previously mentioned a big part of web scraping with the BeautifulSoup library is figuring out what string syntax to pass into the `soup.select()` method.
- Let's go through a table with some common examples (these make a lot of sense if you know CSS syntax)



Syntax	Match Results
<code>soup.select('div')</code>	All elements with 'div' tag
<code>soup.select('#some_id')</code>	Elements containing id='some_id'
<code>soup.select('.some_class')</code>	Elements containing class = 'some_class'
<code>soup.select('div span')</code>	Any elements named span within a div element.
<code>soup.select('div > span')</code>	Any elements named span directly within a div element, with nothing in between.



Grabbing an Image



Complete Python Bootcamp

- Now that we understand how to grab text information based on tags and element names, let's explore how to grab images from a website.
- Images on a website typically have their own URL link (ending in .jpg or .png)



Complete Python Bootcamp

- Beautiful Soup can scan a page, locate the `` tags and grab these URLs.
- Then we can download the URLs as images and write them to the computer.
- Note: You should always check copyright permission before downloading and using an image from a website.



Working with Multiple Pages and Items



Complete Python Bootcamp

- We've seen how to grab elements one at a time, but realistically, we want to be able to grab multiple elements, most likely across multiple pages.
- This is where we can combine our prior python knowledge with the web scraping libraries to create powerful scripts!



Complete Python Bootcamp

- We will use a site specifically designed to practice web scraping: www.toscrape.com
- We will practice grabbing elements across multiple pages.
- Let's get started!



Working with Multiple Pages



Web Scraping Exercises Overview



Web Scraping Exercises Solutions



Web Scraping Exercises Solutions - Part Two