

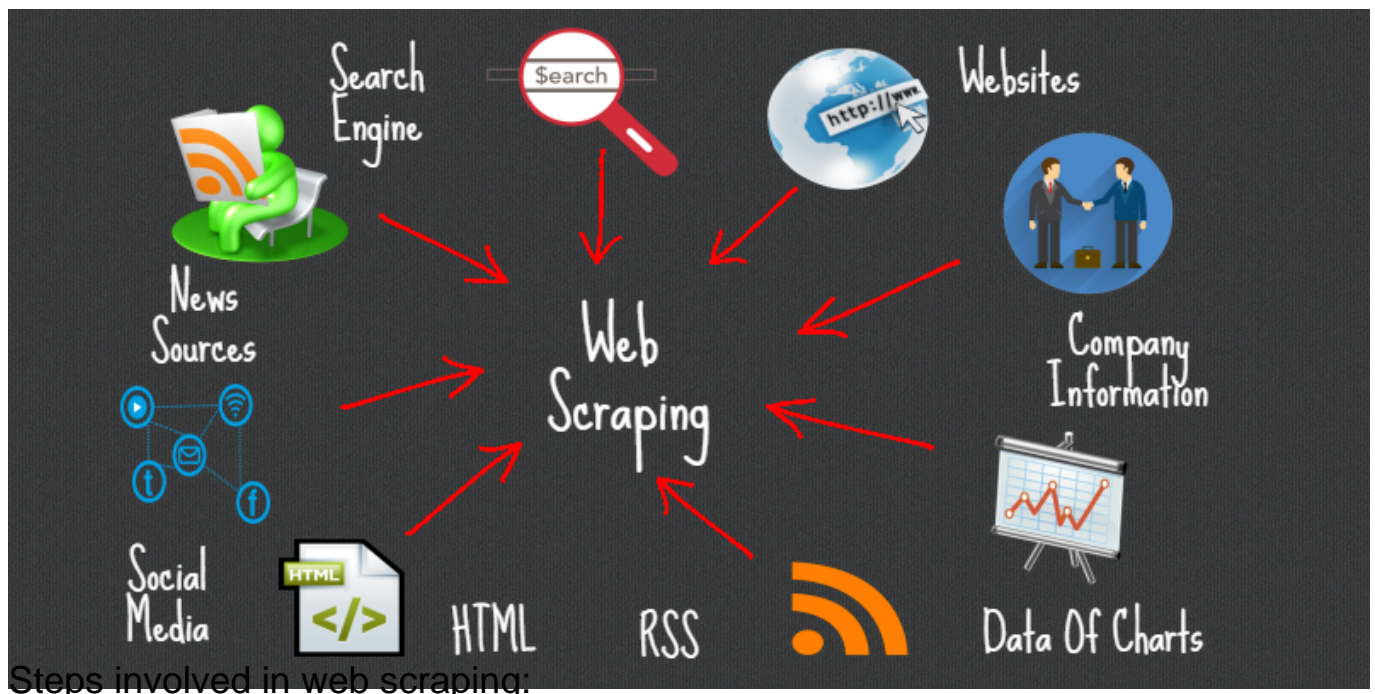
Article

[Rizmaan Marikar](#) · Jul 27, 2022 6m read

Introduction to Web Scraping with Embedded Python - Let 's Extract python job 's

What is Web Scraping:

In simple terms, Web scraping, web harvesting, or web data extraction is an automated process of collecting large data(unstructured) from websites. The user can extract all the data on particular sites or the specific data as per the requirement. The data collected can be stored in a structured format for further analysis.



1. Find the URL of the webpage that you want to scrape
2. Select the particular elements by inspecting
3. Write the code to get the content of the selected elements
4. Store the data in the required format

It 's that simple !!

The popular libraries/tools used for web scraping are:

- Selenium – a framework for testing web applications
- BeautifulSoup – Python library for getting data out of HTML, XML, and other markup languages
- Pandas – Python library for data manipulation and analysis

What is Beautiful Soup?

Beautiful Soup is a pure Python library for extracting structured data from a website. It allows you to parse data from HTML and XML files. It acts as a helper module and interacts with HTML in a similar and better way as to how

you would interact with a web page using other available developer tools.

- It usually saves programmers hours or days of work since it works with your favorite parsers like lxml and html5lib to provide organic Python ways of navigating, searching, and modifying the parse tree.
- Another powerful and useful feature of beautiful soup is its intelligence to convert the documents being fetched to Unicode and outgoing documents to UTF-8. As a developer, you do not have to take care of that unless the document intrinsic doesn't specify an encoding or BeautifulSoup is unable to detect one.
- It is also considered to be faster when compared to other general parsing or scraping techniques.

In today's article we will be using Embedded Python with Object Script to scrape python vacancies and companies on ae.indeed.com

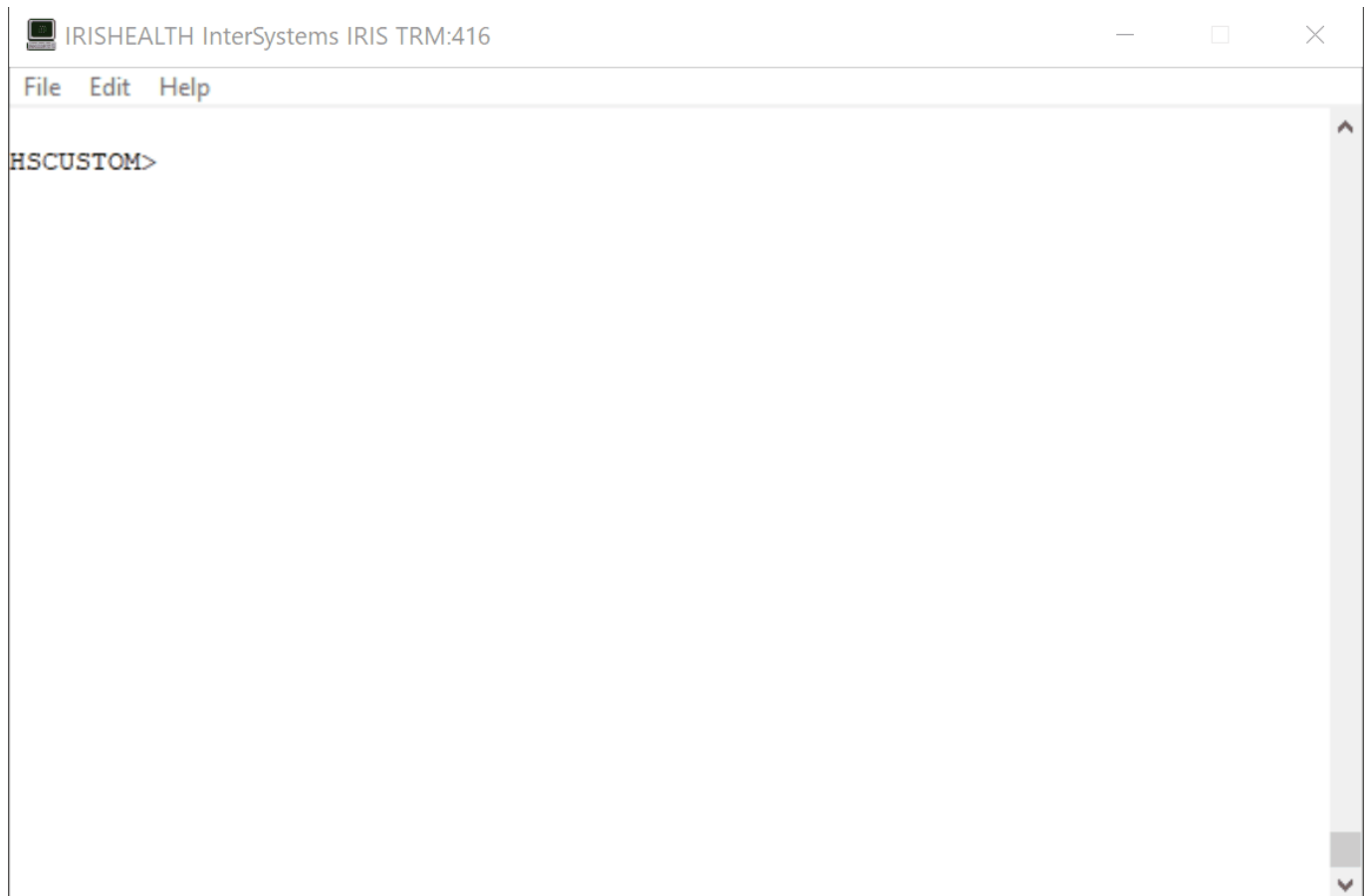
Step 1 - Find the URL of the webpage that you want to scrape.

Url = <https://ae.indeed.com/jobs?q=python&l=Dubai&start=0>

The webpage that we are gonna scrape data from looks like this

The screenshot shows the Indeed job search interface. At the top, there's the Indeed logo and a 'Sign in' button. Below that are search filters: 'What' (python) and 'Where' (Dubai). A large blue 'Find jobs' button is prominent. Below the button are several filter tabs: 'Date Posted', 'Remote', 'Job Type', 'Location', and 'Job Language'. The main content area displays a list of job results. The first result is for 'Analyst Logistics Operations' at 'Talabat' in 'Dubai', with a 4.1 star rating and a 'Full-time' label. The job description snippet mentions 'Groceries to NfV (Non-Food Verticals including retails, pharmacies, etc.)' and 'This role will own several logistics KPIs to constantly improve performance while'. To the right of the job list, there's a detailed view of the 'Analyst Logistics Operations' job, showing the company name 'Talabat' with 25 reviews, the location 'Dubai', and a message stating 'You must create an Indeed account before continuing to the company website to apply'. There are buttons for 'Apply on company site' and a heart icon for saving the job. The 'Job details' section shows 'Job Type' as 'Full-time'. The 'Full Job Description' section is partially visible at the bottom.

for simplicity and learning purposes we will be extracting "Job Title" and "Company", the output would be something similar to below screenshot.



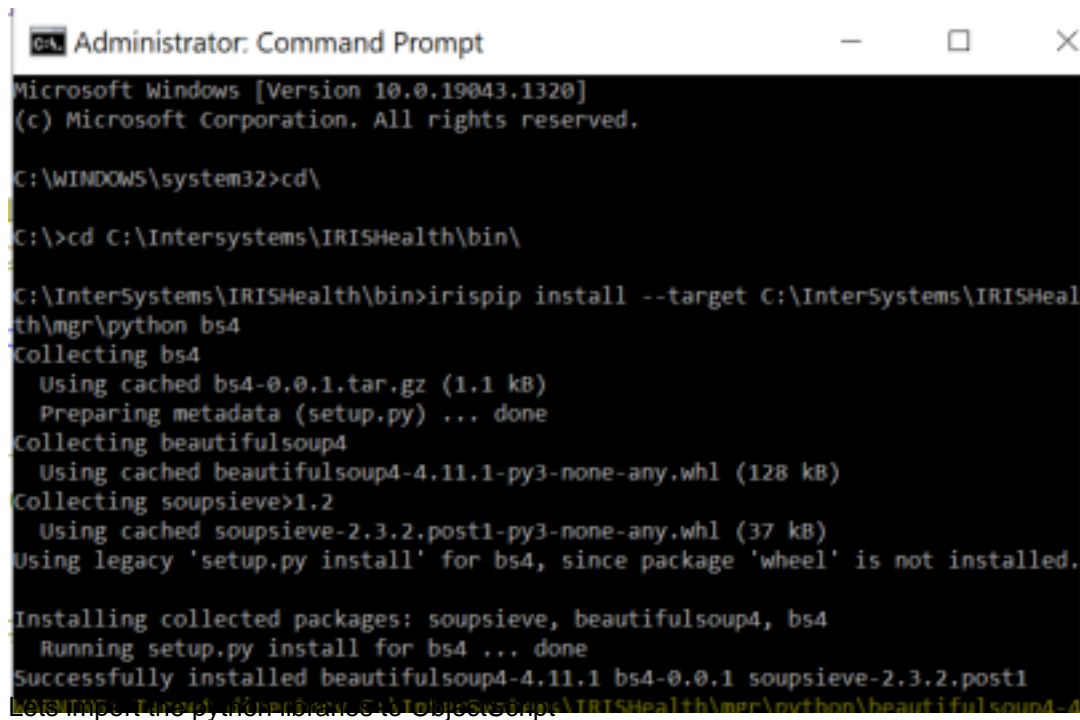
We will be using two python libraries.

- requests Requests is a HTTP library for the Python programming language. The goal of the project is to make HTTP requests simpler and more human-friendly.
- bs4 for BeautifulSoup BeautifulSoup is a Python package for parsing HTML and XML documents. It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping.

Lets install this python packages (windows)

```
irisipip install --target C:\InterSystems\IRISHealth\mgr\python bs4
```

```
irisipip install --target C:\InterSystems\IRISHealth\mgr\python requests
```



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19043.1320]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd\

C:\>cd C:\InterSystems\IRISHealth\bin\

C:\InterSystems\IRISHealth\bin>iris pip install --target C:\InterSystems\IRISHealth\mgr\python bs4
Collecting bs4
  Using cached bs4-0.0.1.tar.gz (1.1 kB)
  Preparing metadata (setup.py) ... done
Collecting beautifulsoup4
  Using cached beautifulsoup4-4.11.1-py3-none-any.whl (128 kB)
Collecting soupsieve>1.2
  Using cached soupsieve-2.3.2.post1-py3-none-any.whl (37 kB)
Using legacy 'setup.py install' for bs4, since package 'wheel' is not installed.

Installing collected packages: soupsieve, beautifulsoup4, bs4
  Running setup.py install for bs4 ... done
Successfully installed beautifulsoup4-4.11.1 bs4-0.0.1 soupsieve-2.3.2.post1
```

```
Class PythonTesting.WebScraper Extends %Persistent
{

// pUrl = https://ae.indeed.com/jobs?q=python&l=Dubai&start=
// pPage = 0
ClassMethod ScrapeWebPage(pUrl, pPage)
{
    // imports the requests python library
    set requests = ##class(%SYS.Python).Import("requests")
    // import the bs4 python library
    set soup = ##class(%SYS.Python).Import("bs4")
    // import builtins package which contains all of the built-in identifiers
    set builtins = ##class(%SYS.Python).Import("builtins")
}
```

Lets collect the html data using requests;

Note: The user agent us taken from googling "my user agent"

The url is "<https://ae.indeed.com/jobs?q=python&l=Dubai&start=>", pPage is the page number

We will do a http get request to the URL using requests and store the response on "req"

```
set headers = {"User-Agent":
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.0.0 Safari/537.36"}
set url = "https://ae.indeed.com/jobs?q=python&l=Dubai&start="_pPage

set req = requests.get(url,"headers="_headers)
```

The req object will have the html which was returned from the webpage.

Let's run this through the BeautifulSoup html parser, so that we can extract the job data.

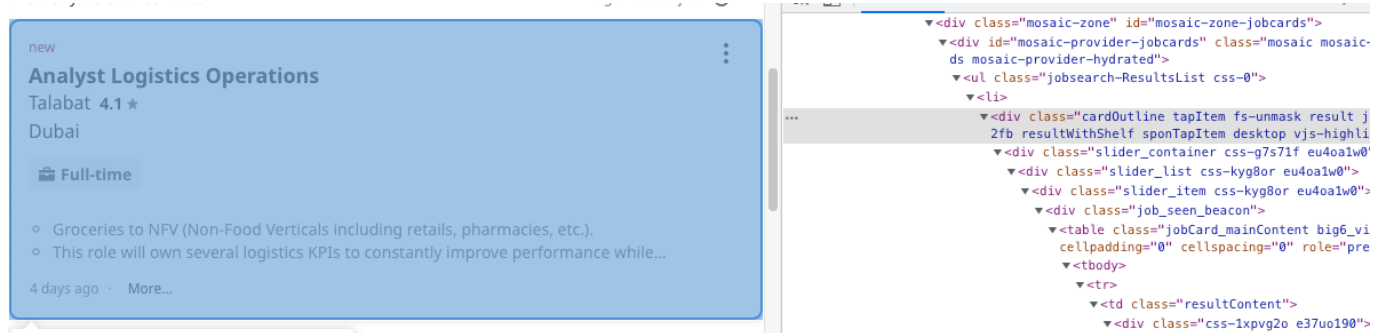
```
set soupData = soup.BeautifulSoup(req.content, "html.parser")
set title = soupData.title.text
W !,title
```

The title looks as follows

Python Jobs in Dubai (with Salaries) 2022 | Indeed.com

Step 2 :Select the required elements by inspecting.

In this scenario we are interested the list of jobs which usually sits in a <div> tag, in your browser you can inspect the element to find the div class.



In our case the required information is stored under <div class="cardOutline tapItem ... </div>

Step 3: Write the code to get the content of the selected elements

We will be using the find_all functionality on BeautifulSoup to look for all the <div> tags which contains the class name "cardOutline"

```
//parameters to python would be sent as a python dictionary
set divClass = {"class":"cardOutline"}
set divsArr = soupData."find_all"("div",divClass...)
```

This will return a list, which we can loop through and extract the Job Titles and Company

Step 4: Store/Display the data in the required format.

In the following example we will be writing the data to the terminal.

```
set len = builtins.len(divsArr)

W !, "Job Title",$C(9)_" --- "_$C(9),"Company"
for i = 1:1:len {
    Set item = divsArr."__getitem__"(i - 1)
    set title = $ZSTRIP(item.find("a").text,"<>W")
    set companyClass = {"class_":"companyName"}
    set company = $ZSTRIP(item.find("span", companyClass...).text,"<>W")
    W !,title,$C(9)," --- ",,$C(9),company
}
```

Note that we are using the builtins.len() to get the length of the divsArr list

Identifier Names:

The rules for naming identifiers are different between ObjectScript and Python. For example, the underscore (_) is allowed in Python method names, and in fact is widely used for the so-called “ dunder ” methods and attributes (“ dunder ” is short for “ double underscore ”), such as `getitem_or_class_`. To use such identifiers from ObjectScript, enclose them in double quotes:

[Intersystems Documentation on Identifier Names](#)

Example Class Method.

ClassMethod ScrapeWebPage(pUrl, pPage)

Next Steps..

Using Object Script and Embedded python and with few lines of code; we could easily scrape data form our favourite job web sites, collect the job name, company, salary, job description and emails/links.

for example if you have multiple pages you can traverse through them easily using the page

This data can be added to a pandas dataframe and remove duplicates, filters can be applied based on specific keywords that you are interested in.

Run this data through numpy, and get some lineplot's

Or perform One-Hot encoding on the data, and create/train your ML models and if there are specific vacancies that you are interested in, send a notification to yourself.

Happy Coding !!!

and don ' t forget to hit the like button

[#Best Practices](#) [#Embedded Python](#) [#ObjectScript](#) [#InterSystems IRIS for Health](#)

Source

URL:<https://community.intersystems.com/post/introduction-web-scraping-embedded-python-let%E2%80%99s-extract-python-job%E2%80%99s>