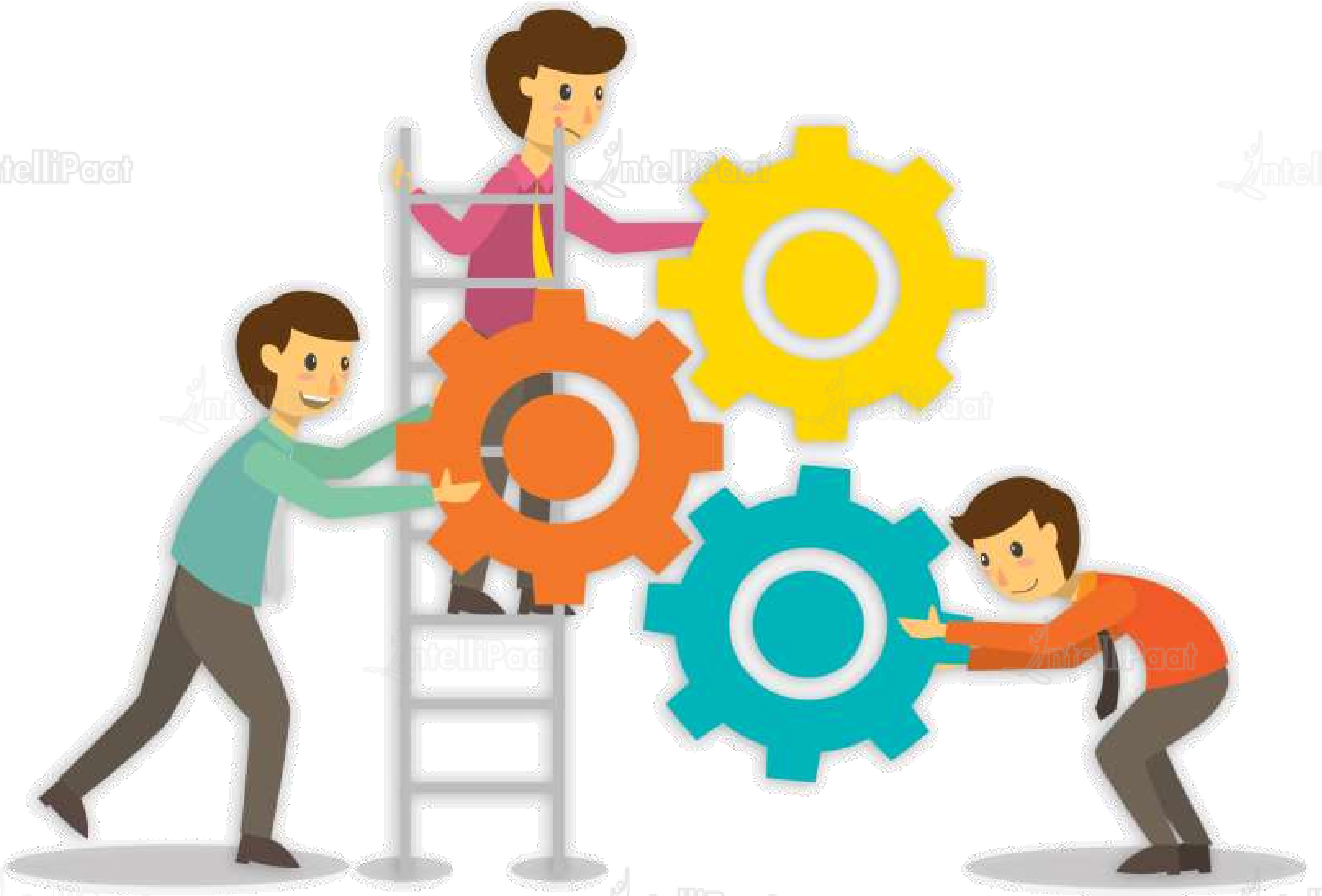
Python Training

Python Web Scrapping



Agenda

Introduction to Web Scrapping

Web Scrapping

- Web scraping is an automated method used to extract large amounts of data from websites.
- Web scraping, web harvesting, or web data extraction is data scraping used for extracting data from websites.
- The data on the websites are unstructured.
- Web scraping helps collect these unstructured data and store it in a structured form.
- There are ways to scrape websites such as online Services, APIs or writing your code.
- Web Scraping is a technique employed to extract large amounts of data from websites

Why Web Scrapping?

- Web scraping is used to collect large information from websites.
- Price Comparison: Services such as ParseHub use web scraping to collect data from online shopping websites and use it to compare the prices of products.
- Email address gathering: Many companies that use email as a medium for marketing, use web scraping to collect email ID and then send bulk emails.
- Social Media Scraping: Web scraping is used to collect data from Social Media websites such as Twitter to find out what's trending.
- Research and Development: Web scraping is used to collect a large set of data (Statistics, General Information, Temperature, etc.) from websites, which are analyzed and used to carry out Surveys or for R&D.
- Job listings: Details regarding job openings, interviews are collected from different websites and then listed in one place so that it is easily accessible to the user.

Web Scrapping

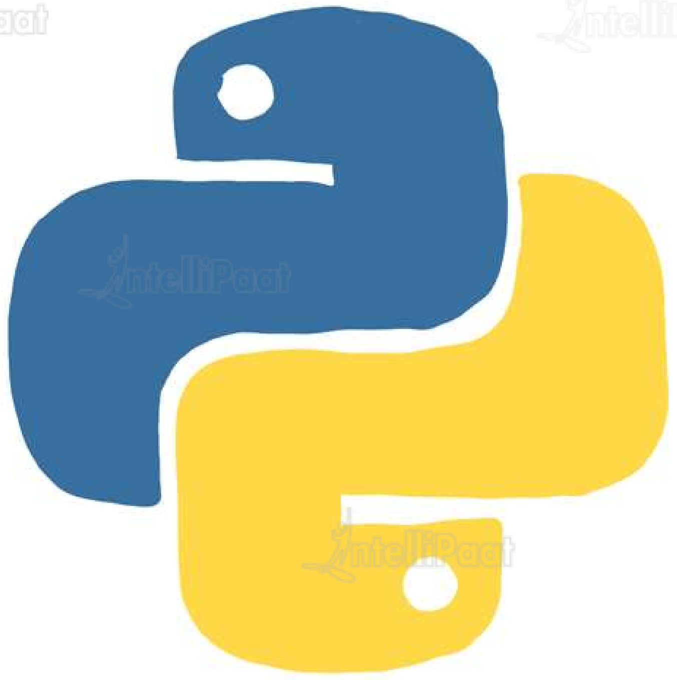
Why Python for Web Scrapping?

- Here is the list of features of Python which makes it more suitable for web scraping.
- Ease of Use: Python is simple to code. You do not have to add semicolons “;” or curly-braces “{}” anywhere.
- This makes it less messy and easy to use.
- Large Collection of Libraries: Python has a huge collection of libraries such as Numpy, Matplotlib, Pandas etc., which provides methods and services for various purposes.
- Hence, it is suitable for web scraping and for further manipulation of extracted data.
- Dynamically typed: In Python, you don’t have to define data types for variables, you can directly use the variables wherever required.
- This saves time and makes your job faster.

Web Scrapping

Why Python for Web Scrapping?

- Here is the list of features of Python which makes it more suitable for web scraping.
- Easily Understandable Syntax: Python syntax is easily understandable mainly because reading a Python code is very similar to reading a statement in English.
- It is expressive and easily readable, and the indentation used in Python also helps the user to differentiate between different scope/blocks in the code.
- Small code, large task: Web scraping is used to save time. But what’s the use if you spend more time writing the code? Well, you don’t have to.
- In Python, you can write small codes to do large tasks. Hence, you save time even while writing the code.
- Community: What if you get stuck while writing the code? You don’t have to worry.
- Python community has one of the biggest and most active communities, where you can seek help from.

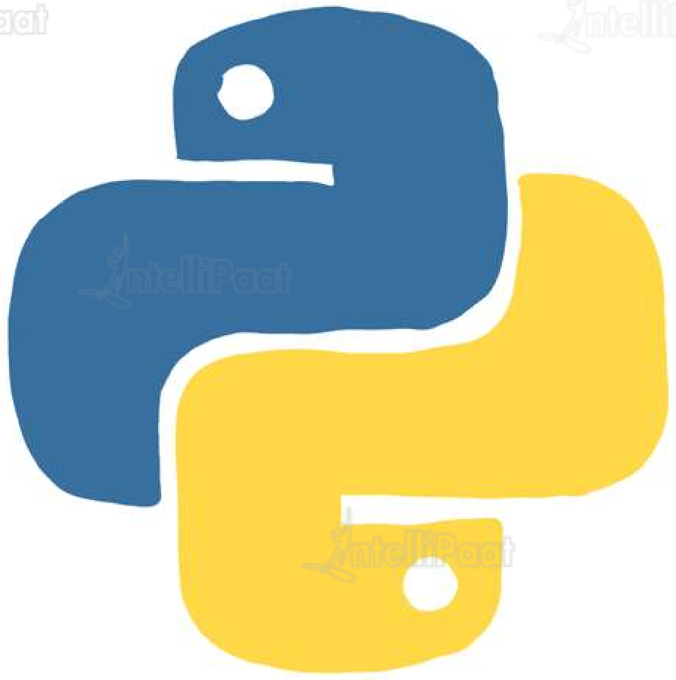


Web Scrapping

How does Web Scrapping works?

- When you run the code for web scraping, a request is sent to the URL that you have mentioned.
- As a response to the request, the server sends the data and allows you to read the HTML or XML page.
- The code then, parses the HTML or XML page, finds the data and extracts it.
- To extract data using web scraping with python, you need to follow these basic steps:

    1. Find the URL that you want to scrape
    2. Inspecting the Page
    3. Find the data you want to extract
    4. Write the code
    5. Run the code and extract the data
    6. Store the data in the required format



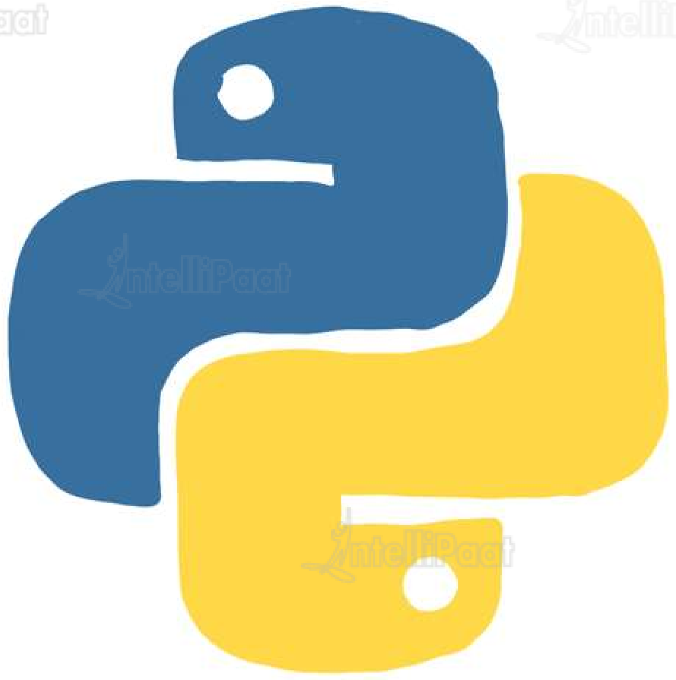
Web Scrapping

Web Scrapping libraries

- As we know, Python is used for various applications and there are different libraries for different purposes.
- In our further demonstration, we will be using the following libraries:
- Selenium: Selenium is a web testing library. It is used to automate browser activities.
- BeautifulSoup: Beautiful Soup is a Python package for parsing HTML and XML documents. It creates parse trees that is helpful to extract the data easily.
- Pandas: Pandas is a library used for data manipulation and analysis. It is used to extract the data in the desired format.

Steps involved in Web Scrapping

- Send a HTTP request to the URL of the webpage you want to access.
- The server responds to the request by returning the HTML content of the webpage.
- For this task, we will use a third-party HTTP library for python requests.
- Once we have accessed the HTML content, we are left with the task of parsing the data. Since most of the HTML data is nested, we can't extract data simply through strin.
- One needs a parser which can create a nested/tree structure of the HTML data. There are many HTML parser libraries available but the most advanced one is html5lib.
- Now, all we need to do is navigating and searching the parse tree that we created, i.e. tree traversal.
- For this task, we will be using another third-party python library, Beautiful Soup. It is a Python library for pulling data out of HTML and XML files.



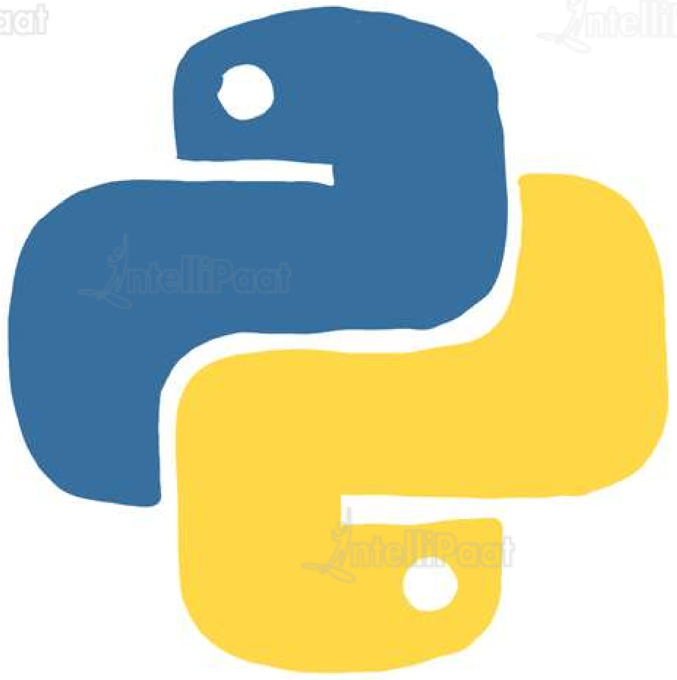
Web Scrapping

Step 1: Installing the required 3rd Party libraries

- Easiest way to install external libraries in python is to use pip.
- pip is a package management system used to install and manage software packages written in Python.
- All you need to do is:

```
pip install requests
pip install html5lib
pip install bs4
```

- Another way is to download them manually from these links:
  - Requests
  - Html5lib
  - beautifulsoup4



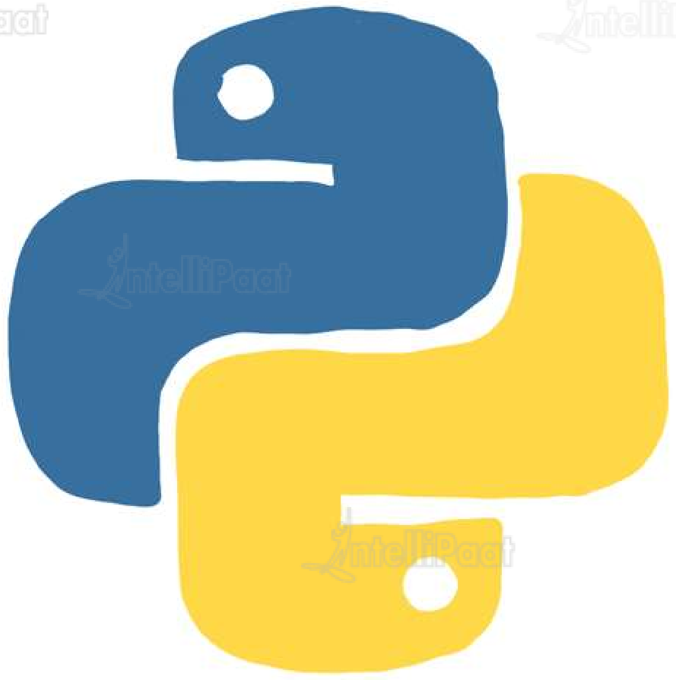
Web Scrapping

Step 2: Accessing the HTML content from Webpage

- Let us try to understand this piece of code.

```
import requests
URL = "https://docs.python.org/3/"
r = requests.get(URL)
print(r.content)
```

- First of all import the requests library.
- Then, specify the URL of the webpage you want to scrape.
- Send a HTTP request to the specified URL and save the response from server in a response object called r.
- Now, as print r.content to get the raw HTML content of the webpage. It is of 'string' type.



Web Scrapping

Step 3: Parsing the HTML content

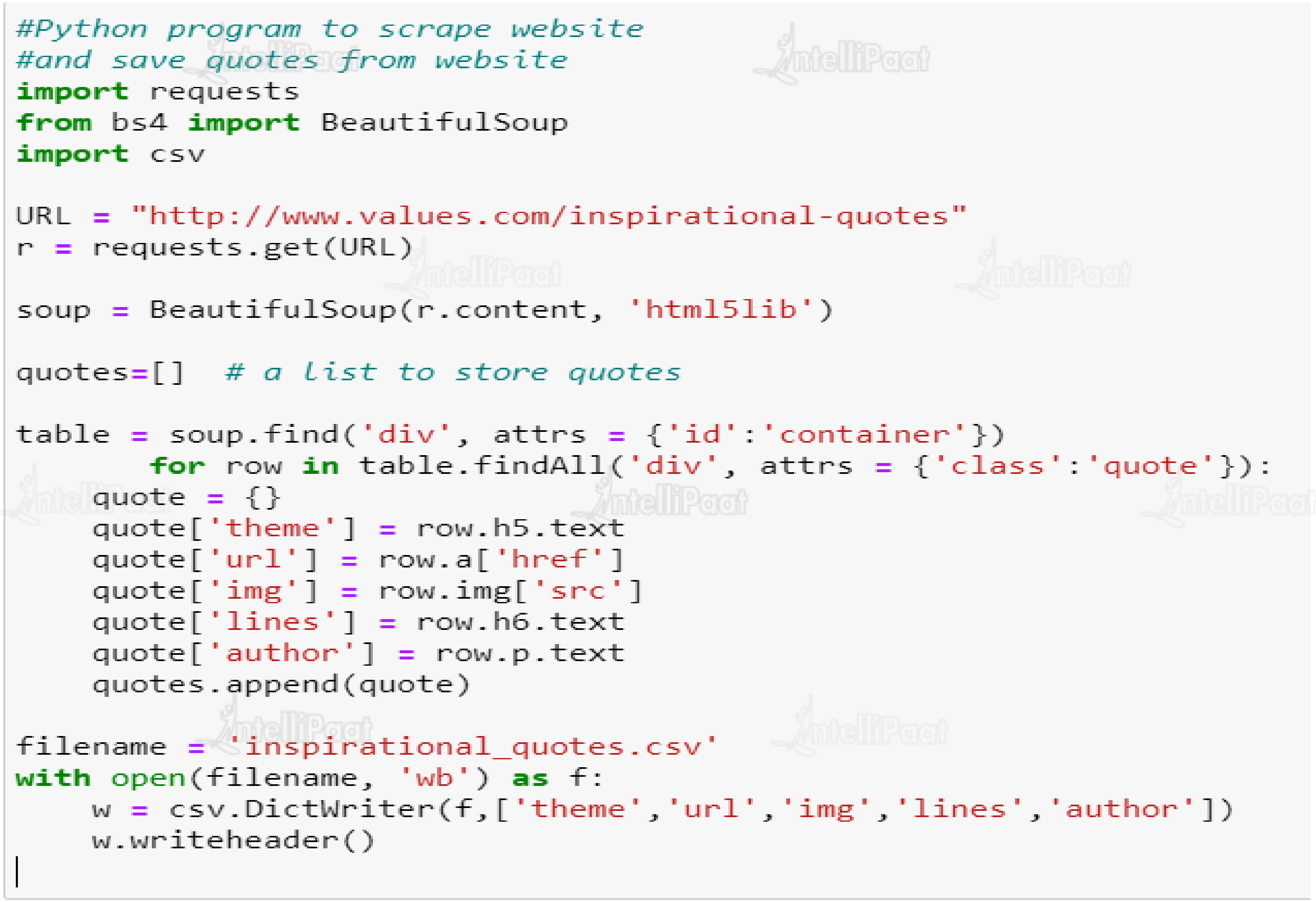
```
#This will not run on online IDE
import requests
from bs4 import BeautifulSoup
URL = "http://www.values.com/inspirational-quotes"
r = requests.get(URL)
soup = BeautifulSoup(r.content, 'html5lib')
print(soup.prettify())
```

- A really nice thing about BeautifulSoup library is that it is build on the top of the HTML parsing libraries like html5lib, lxml, html.parser, etc.
- So BeautifulSoup object and specify the parser library can be created at the same time.
- In the example above,

```
soup = BeautifulSoup(r.content, 'html5lib')
```

Web Scrapping

Step 4: Searching and Navigating through parse tree



```
#Python program to scrape website
#and save quotes from website
import requests
from bs4 import BeautifulSoup
import csv

URL = "http://www.values.com/inspirational-quotes"
r = requests.get(URL)

soup = BeautifulSoup(r.content, 'html5lib')

quotes=[]  # a list to store quotes

table = soup.find('div', attrs = {'id':'container'})
       for row in table.findAll('div', attrs = {'class':'quote'}):
    quote = {}
    quote['theme'] = row.h5.text
    quote['url'] = row.a['href']
    quote['img'] = row.img['src']
    quote['lines'] = row.h6.text
    quote['author'] = row.p.text
    quotes.append(quote)

filename = 'inspirational_quotes.csv'
with open(filename, 'wb') as f:
    w = csv.DictWriter(f,['theme','url','img','lines','author'])
    w.writeheader()
```

Web Scrapping

India: +91-7847955955

US: 1-800-216-8930 (TOLL FREE)

support@intellipaat.com

24/7 Chat with Our Course Advisor