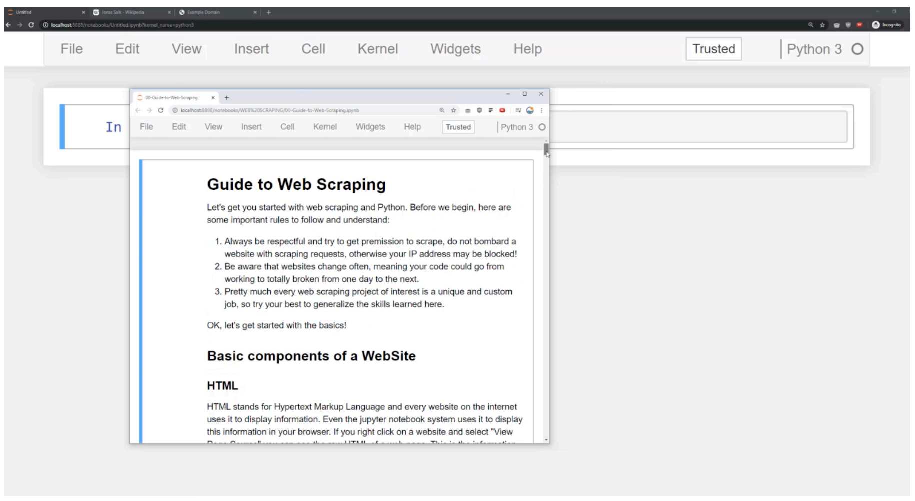
SECTION 15: WEB SCRAPING WITH PYTHON - 1 hour 40 minutes, 9 parts

3/9 Python Web Scraping - Grabbing a Title



Guide to Web Scraping

Let's get you started with web scraping and Python. Before we begin, here are some important rules to follow and understand:

1. Always be respectful and try to get premission to scrape, do not bombard a website with scraping requests, otherwise your IP address may be blocked!
2. Be aware that websites change often, meaning your code could go from working to totally broken from one day to the next.
3. Pretty much every web scraping project of interest is a unique and custom job, so try your best to generalize the skills learned here.

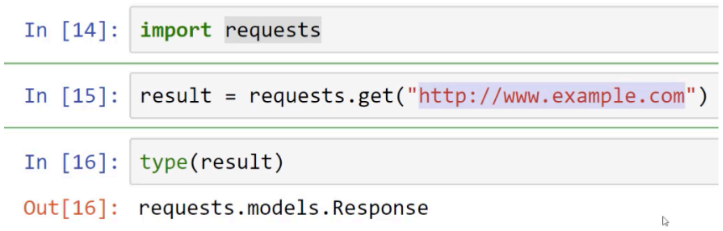
OK, let's get started with the basics!

Basic components of a WebSite

HTML

HTML stands for Hypertext Markup Language and every website on the internet uses it to display information. Even the jupyter notebook system uses it to display this information in your browser. If you right click on a website and select "View

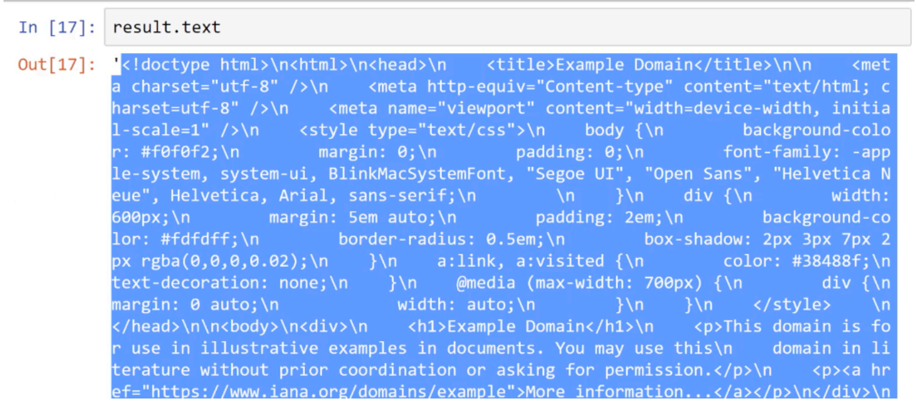
- **web scraping with Python**
  - **-> we are grabbing webpage titles with Python**
    - -> in the ipynb file, it's an HTML document with web scraping
    - -> we have an example HTML webpage
    - -> he is importing the requests module
    - -> then getting a result from this
    - -> this is via the .get method
    - -> if this returns an error, then check the firewall
    - -> the request library gets a response from the URL
    - -> this has an attribute
    - -> there is information stored as a Python string
    - **-> the returns module is used to scrape information from the webpage**
    - **-> and then beautiful soup is used to format the strings**
    - -> he has then imported the bs4 module
    - -> then created a soup object, stored in the soup variable
    - -> this is passed in in xml
    - -> then calling the name of the variable returns the content which we want
    - -> we are using soup.select() to grab things from the HTML document
    - -> raw HTML elements



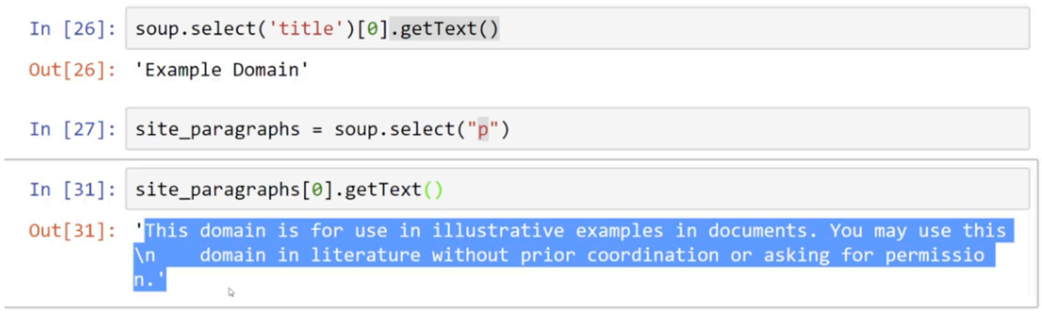
```
In [14]: import requests

In [15]: result = requests.get("http://www.example.com")

In [16]: type(result)
Out[16]: requests.models.Response
```



```
In [17]: result.text
Out[17]: '<!doctype html>\n<html>\n<head>\n    <title>Example Domain</title>\n\n    <met
         a charset="utf-8" />\n    <meta http-equiv="Content-type" content="text/html; c
         harset=utf-8" />\n    <meta name="viewport" content="width=device-width, initia
         l-scale=1" />\n    <style type="text/css">\n    body {\n        background-colo
         r: #f0f0f2;\n        margin: 0;\n        padding: 0;\n        font-family: -app
         le-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica N
         eue", Helvetica, Arial, sans-serif;\n        \n    }\n    div {\n        width:
         600px;\n        margin: 5em auto;\n        padding: 2em;\n        background-co
         lor: #fdfdff;\n        border-radius: 0.5em;\n        box-shadow: 2px 3px 7px 2
         px rgba(0,0,0,0.02);\n    }\n    a:link, a:visited {\n        color: #38488f;\n
         text-decoration: none;\n    }\n    @media (max-width: 700px) {\n        div {\n
         margin: 0 auto;\n            width: auto;\n        }\n    }\n    </style>    \n
         </head>\n\n<body>\n<div>\n    <h1>Example Domain</h1>\n    <p>This domain is fo
         r use in illustrative examples in documents. You may use this\n    domain in li
         terature without prior coordination or asking for permission.</p>\n    <p><a hr
         ef="https://www.iana.org/domains/example">More information...</a></p>\n</div>\n
```



```
In [26]: soup.select('title')[0].getText()
Out[26]: 'Example Domain'

In [27]: site_paragraphs = soup.select("p")

In [31]: site_paragraphs[0].getText()
Out[31]: 'This domain is for use in illustrative examples in documents. You may use this
         \n    domain in literature without prior coordination or asking for permissio
         n.'
```

  - **-> soup.select('')**
    - -> then passing the name of the HTML element tag into its argument
    - -> this returns a list as its default
    - -> this returns a list of all the paragraph elements on the page
    - -> we have scraped a section of a webpage, then formatted it into a more beautiful soup, and this is what we use to return a section of text from that
    - -> then we can use the .getText() method to return the string

  - **-> soup.select("p"), for example**
    - -> he sets this equal to a variable

    - -> he extracts the first element in the list
    - **-> this is a beautiful soup object**
    - -> p is the tag name in the html file
    - -> by default, it returns a list of all the paragraph tags on the page
    - -> for selecting html elements
    - -> this also automatically returns a list
    - -> we can for example add in a [0] at the end of it
    - -> and then .getText()
    - -> so, processing the information to extract certain terms
    - -> we can also store this information in variables
    - -> we also have specialised beautiful soup objects

- **-> overview of steps**
    - -> he's imported requests
    - -> then requests.get
    - -> this returns a response which has a text attribute
    - -> then importing bs4
    - -> then using bs4....
    - -> we are then selecting elements off of this and passing it into a string
    - -> we need to put the right argument into .select