**WEB SCRAPPING TUTORIAL FOR PLANTIX**

**URL-** <https://plantix.net/en/library/plant-diseases/>

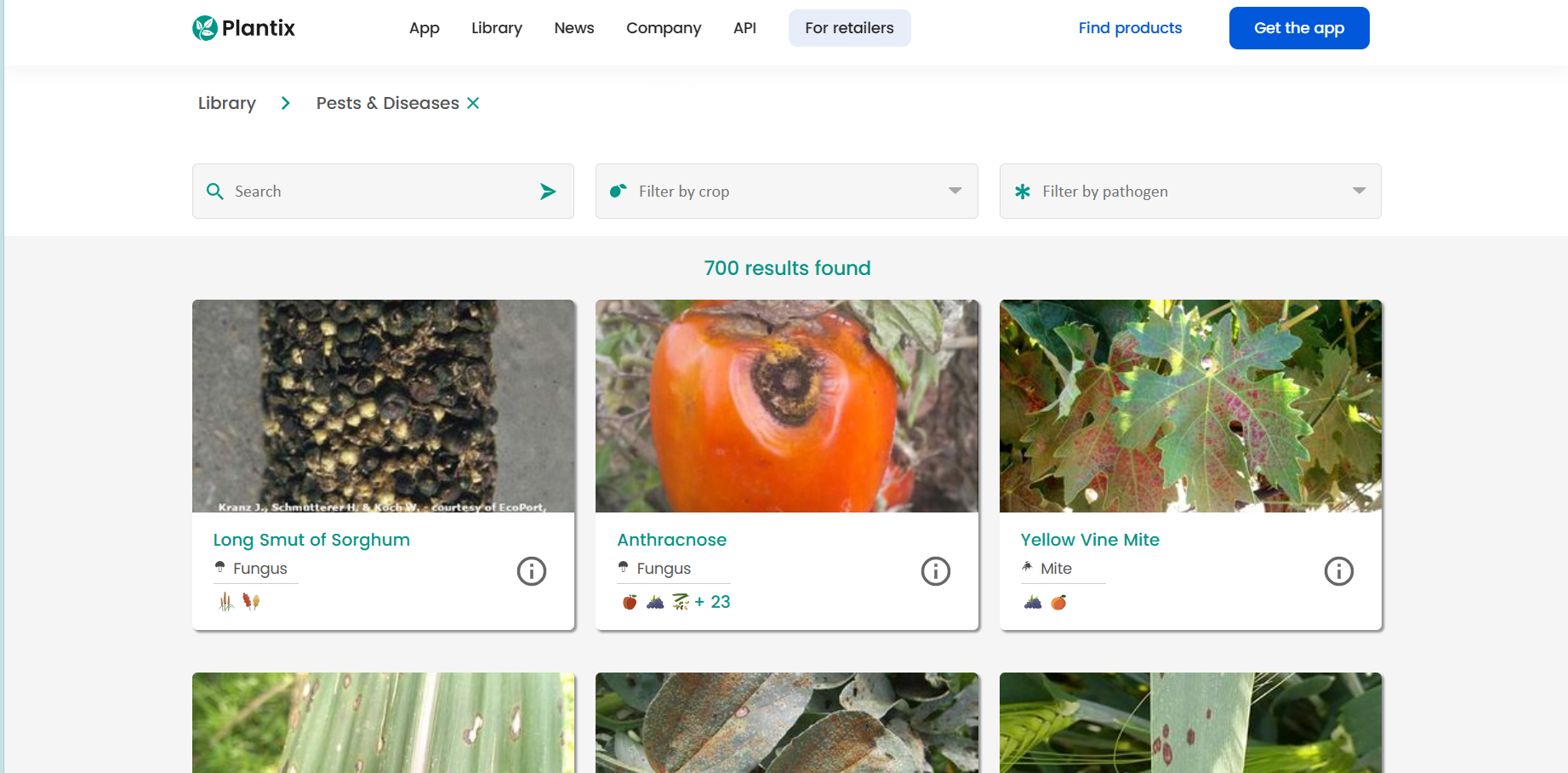
**REQUIREMENTS –**

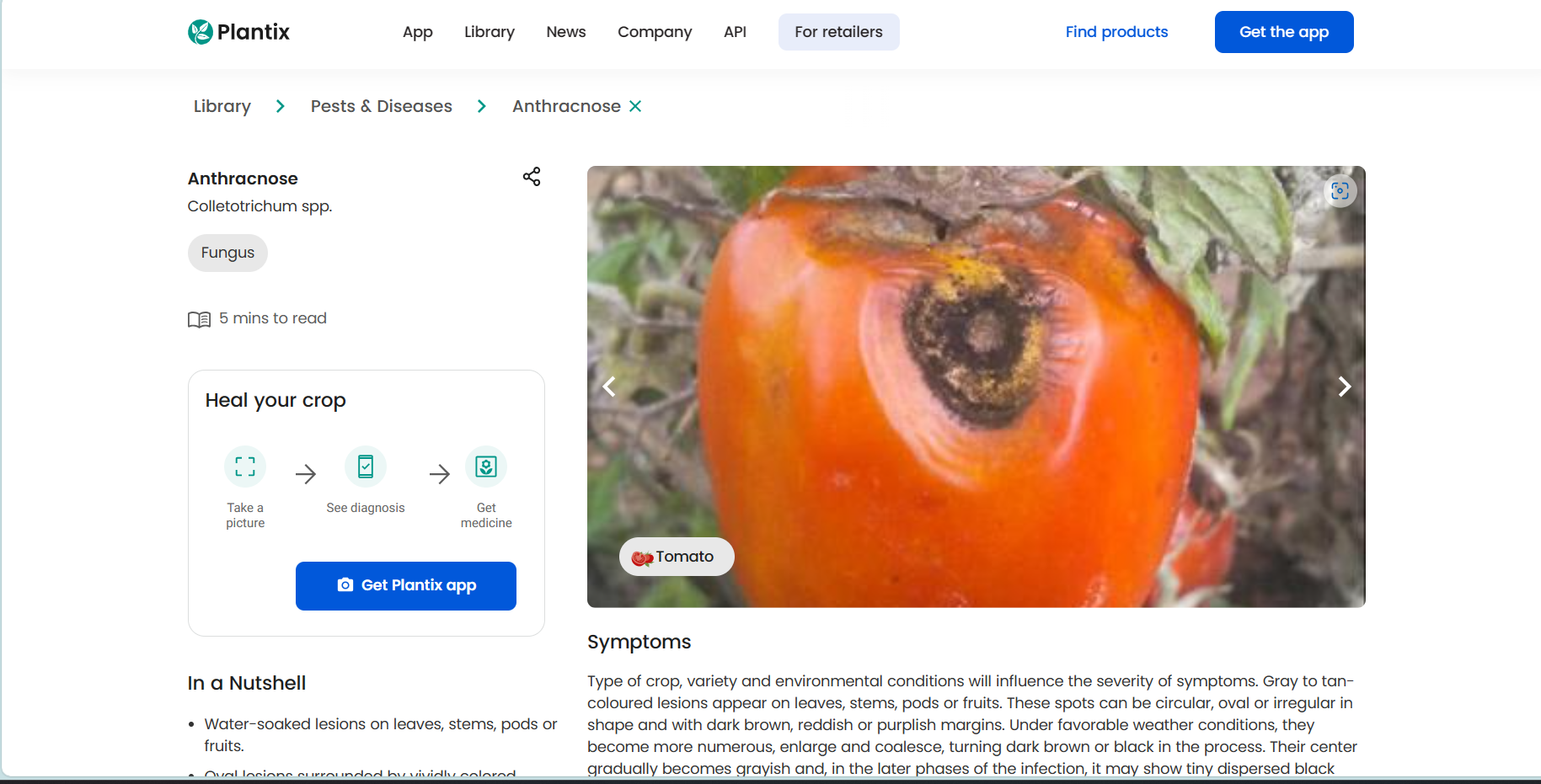
**pip install requests beautifulsoup4**

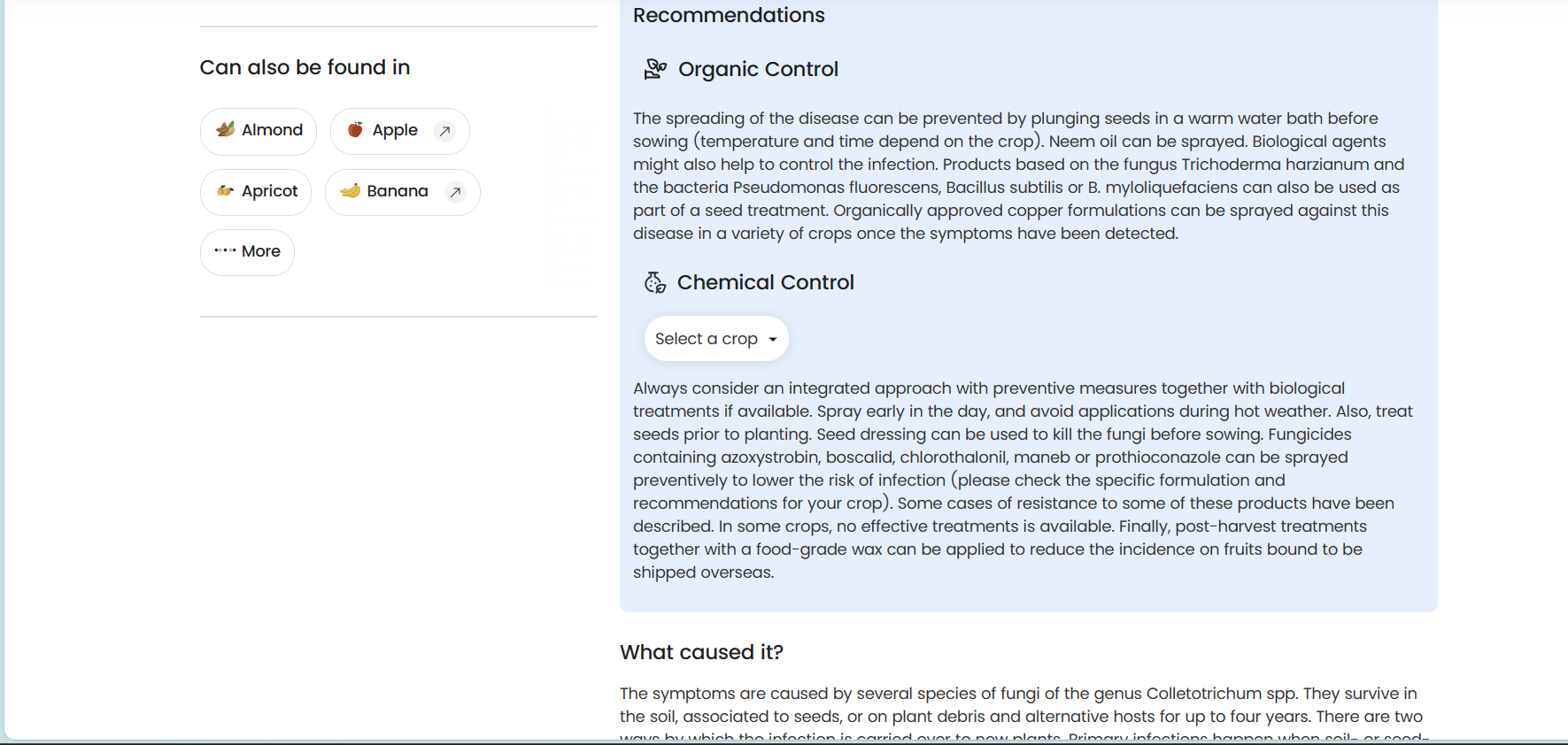
|  |  |  |
| --- | --- | --- |
| **Sr. no.** | **NAME** | **DESCRIPTION** |
| 1 | requests | Python library for making HTTP requests to web pages. |
| 2 | bs4 | BeautifulSoup library for parsing HTML. |
| 3 | time | Standard Python library for time-related functions. |
| 4 | os | Standard Python library for operating system dependent functionality (used for interacting with the file system). |
| 5 | re | Standard Python library for regular expressions, used for text processing |
| 6 | html.parser | A parser specified in BeautifulSoup to handle HTML.  In this case, it refers to the parser provided by the Python standard library. |
| 7 | selenium | A tool to scrap dynamic web pages |

**USAGE** -

First Look of the website







By this we can understand, the structure of the website.

Through this we understood

* Folders and Direct documents are assigned class “folder”
* If no a tags found with that class, we have to scrap content, in Fig 3

If we have a folders, we expand and scrap content

**HOW TO USE?**

1. First load all names of the diseases by copy-pasting HTML code or saving it offline
2. Then extract names and relevant info and add it to a dictionary

**Function load\_all\_diseases()**

Start

|

|--> Open HTML File ("html.txt")

| |

| |--> Initialize diseaseNamesList, diseaseTypeList, links, and images lists

| |

| |--> Initialize resultingDictionary as an empty dictionary

| |

| |--> Read HTML content from the file and parse it using BeautifulSoup

| |

| |--> Find all elements with class 'disease-name textfade'

| | |

| | |--> Store text of these elements in diseaseNamesList

| |

| |--> Find all elements with class 'disease-type textfade'

| | |

| | |--> Store text of these elements in diseaseTypeList

| |

| |--> Find all elements with class 'pests-and-diseases-result result-card'

| | |

| | |--> Extract links and store them in the links list

| | |

| | |--> Extract images and store them in the images list

| |

| |--> Iterate through a range (e.g., 700) for each disease

| | |

| | |--> Create a dictionary entry in resultingDictionary:

| | | |

| | | |--> Assign unique IDs ("DN" followed by index) to diseases

| | | |

| | | |--> Map disease type, link, and image URL to disease name

| |

| |--> End iteration

| |

| |--> Return resultingDictionary

|

|--> End

1. Now prepare algorithm for extracting data from one disease by providing link

**Function load\_disease() 🡪 uses selenium**

Start

|

|--> Initialize an empty string variable "str"

|

|--> Create a new WebDriver instance (Microsoft Edge)

| |

| |--> Open the specified URL using the WebDriver

| | |

| | |--> Successfully navigated to the webpage?

| | | |

| | | |--> Extract and append the title of the webpage to "str"

| | | |

| | | |--> Find and extract symptoms section content

| | | |

| | | |--> Append "Product Recommendations" to "str"

| | | |

| | | |--> Find and extract product recommendations section content

| | | |

| | | |--> Find and extract trigger card section content

| | | |

| | | |--> Append "Preventive Measures" to "str"

| | | |

| | | |--> Find and extract preventive measures section content

| |

| |--> Wait for 5 seconds

| |

| |--> Close the WebDriver

| |

| |--> Write the extracted content to a text file in the "PlantixData" directory

| |

| |--> Print a success message for debugging

| |

| |--> Return the extracted content

|

|--> Handle Exceptions:

| |

| |--> Catch any exceptions that occur during the scraping process

| |

| |--> Print an error message and return "NA"

|

|--> End (Return extracted content or "NA" in case of an exception)

Now traverse dictionary and pass links one by one and save it all in a buffer