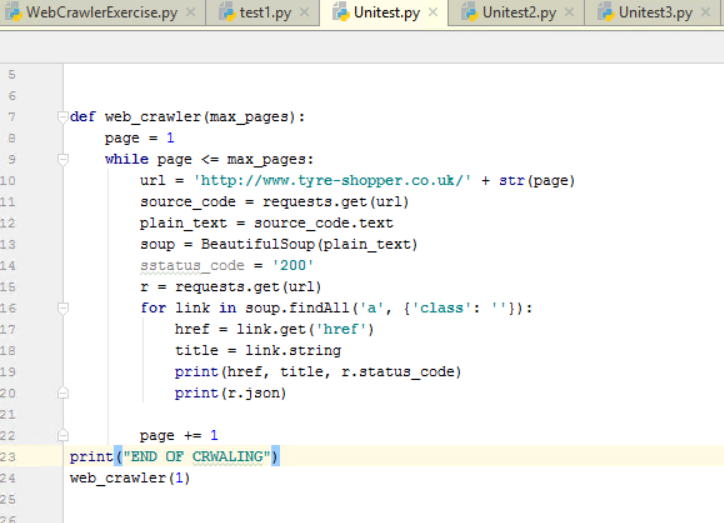
## Unit Testing

I have been spending two evening attempting to puzzle out a good way of testing: “How to Build a Web Crawler to crawl a single web domain”. I am enamoured with the idea of unit tests, and I have the feeling that I could save myself a bit of misery if I had a mechanism to build tests for each bug I close. It is a great dream, but after these two marvelous evening 3 hours each approximately, I have not come up with a solid solution for ‘http status code’ of my work.

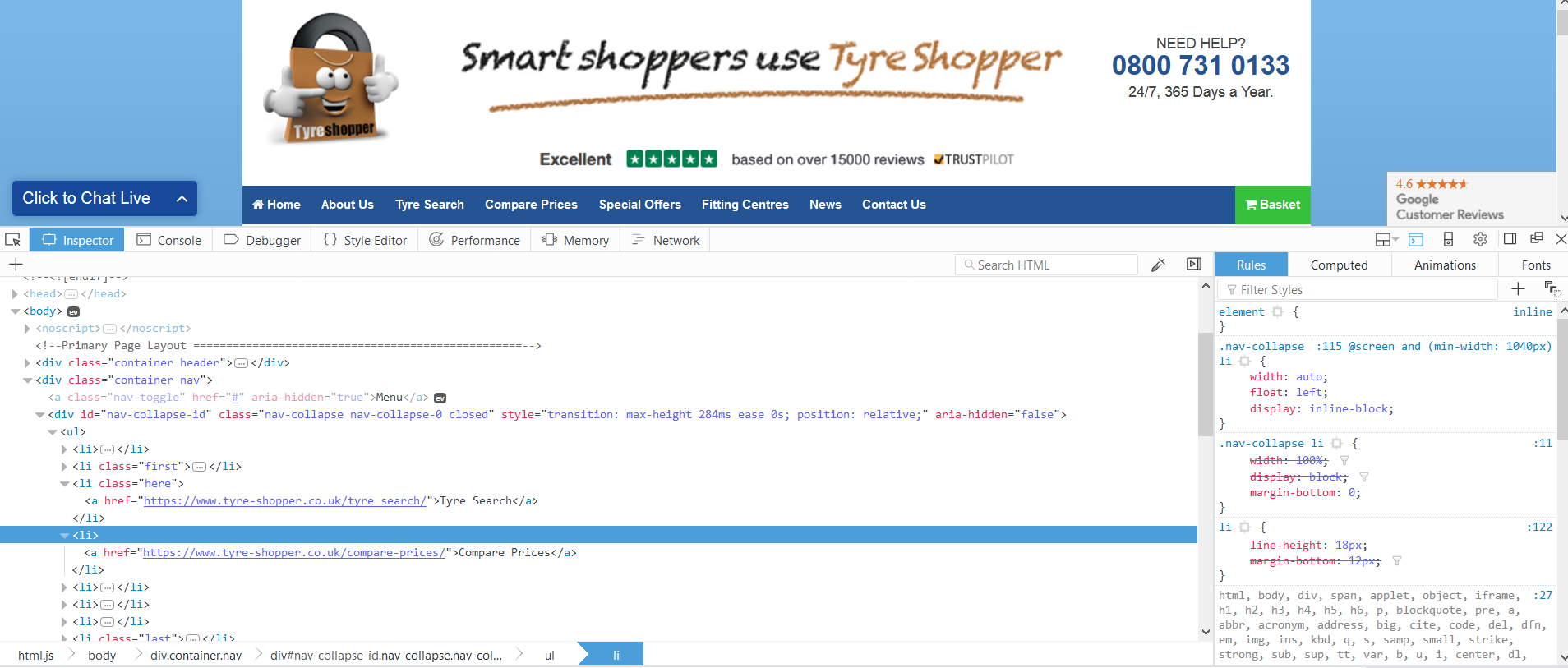
Sketching out the broad outlines of what I want is pretty simple. There are two broad categories of things I want to test:

1. **Core functionality:** I should end up with a list of URL, html title page and http status code of the crawled web domain. For example; if the status code is 200; the page is accessible else with status code 404; the page shall not be viewed.
2. **Presentation tests:** Each line captured should be displayed in a CSV file which generally reflects the actual outcome of URLs, html title page and http status code.

I was free to choose the programming language to build the web crawler engine and I chose python. PyCharm is the IDE I used to complete the web crawler program; which have helped a lot indeed. Building the first Crawling engine in Python, using beautiful soup. Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.



Inspecting element of the web domain [**http://www.tyre-shopper.co.uk/**](http://www.tyre-shopper.co.uk/) has responded with some difficulties. Links are contained in Anchors which are actually classified in multiple classes. This methodic way of classifying anchors for the URLs were basically in several classes. So; by choosing the appropriate name of the class; results were terrific whereby results was null or at times only two or three links were displayed as outcome of the web crawler program built. So broadly taken the anchor with no class that displayed me pretty good results which obviously could have been ameliorated.

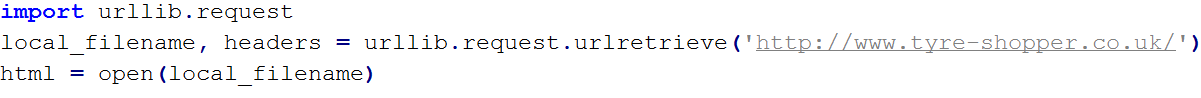




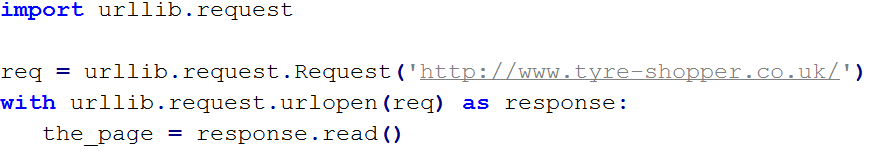
Some testing was done over and over again to display the **Http Status code**. **Urllib.request** is a Python module for fetching URLs (Uniform Resource Locators). It offers a very simple interface, in the form of the *urlopen* function. This is capable of fetching URLs using a variety of different protocols. It also offers a slightly more complex interface for handling common situations - like basic authentication, cookies, proxies and so on. These are provided by objects called handlers and openers.

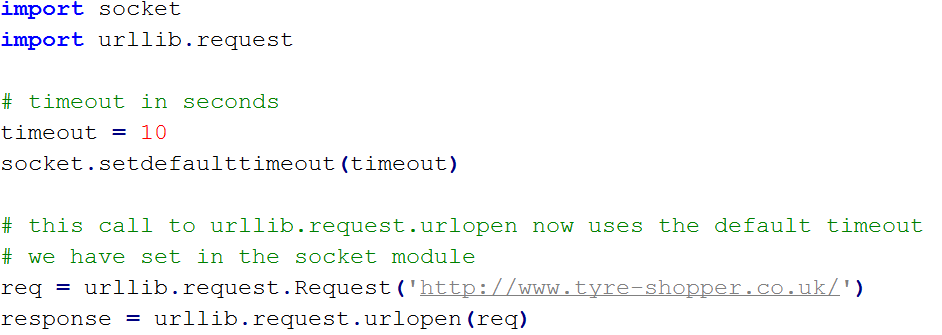
Certain cases of anomaly was at the **HTTPLINK** was has not been imported. PyCharm does not recognize httplink or httplib for the import.





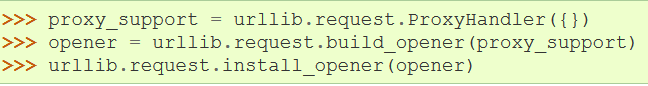
**HTTP** is based on requests and responses - the client makes requests and servers send responses. urllib.request mirrors this with a Request object which represents the HTTP request you are making. In its simplest form you create a Request object that specifies the URL you want to fetch. Calling urlopen with this Request object returns a response object for the URL requested. This response is a file-like object, which means you can for example call .read() on the response:



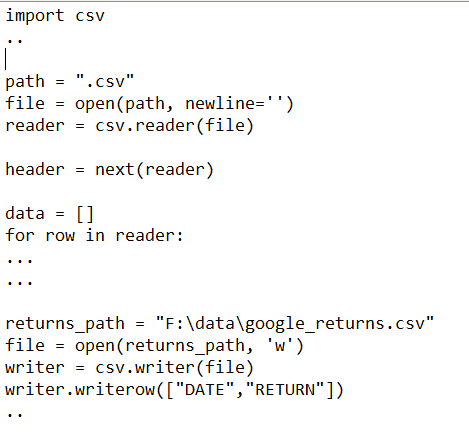


## **Proxies**

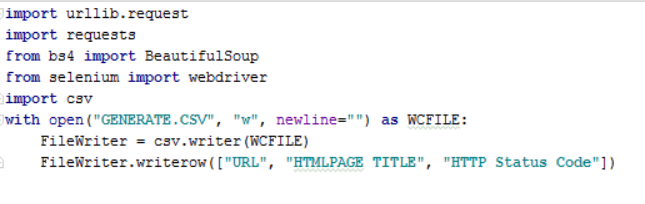
urllib will auto-detect your proxy settings and use those. This is through the ProxyHandler, which is part of the normal handler chain when a proxy setting is detected. Normally that’s a good thing, but there are occasions when it may not be helpful whereby the piece of data shown below has not responded in the Pycharm IDE.

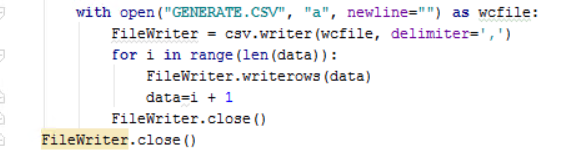


I have tried some read / write exercise just like the below screenshot:



Tried to execute as many examples so as to optimize the way data has been captured from the given web domain to finally come up with a set of code which looks like:





The generation of the result was as follows:



Several Testing were performed to finally outcome the requested data in the below CSV format:

