Web scraping and Covid-19: Covid 2

Welcome to the fourth lesson in our web scraping to build social research data training series. In the previous lesson, we looked at our first two examples of scraping data from the web. And the first we looked at how we could extract text stored on a webpage. And in the second part, we looked at how we could directly download a file from our webpage. Now today, we're going to look at a more complicated, real example. So we're going to see how we can extract detailed statistics and tables of statistics from a website containing information on COVID-19 data.

So again, real live website that gets updated, I think multiple times per day with information from different countries. So we're gonna follow the exact same process as we did in lesson three. So if you remember, we've got a six step process for web scraping. We're going to follow that once more. But this time, the Python code needs to be a slightly more advanced to deal with the structure of the web page, but it's nothing special. It's nothing too difficult. And I'm sure you'll have fun trying it yourself. So again, first step, we need a piece of information, which is the web page containing the data we want to scrape. There's a website that's been tracking the development of COVID-19 globally, so it's on the world dot mushers dot info website. And it's got a page relating to Coronavirus. Worldometer in general is reasonably interesting. It's it's got lots of real world, quote unquote real time population level statistics. And so you can look at developments of the world's population in real time. So again, it'll do its best to retrieve good quality data about demographics, globally. So we can see again this like, quote, unquote, real time changes in the world's population using official statistics. And so this is a good reputable source of data. So we're going to be interested in the Coronavirus data. And in particular for this first part of the lesson, we're interested in three simple statistics global Coronavirus cases, and global deaths and the total number of people who have recovered. So this is the information we're interested in scraping from the webpage.

So we've done the first bit we've we've identified the web page of interest. Now on that webpage, we need to identify where the information actually is. So given we've done our visual manual inspection, we can use our previous trick, which is if we highlight the content we want. If we right click and select inspect, and I'm on Google, Google Chrome at the moment, you'll be able to see the underlying HTML, as it specifically relates to what I've highlighted here. So what I'm looking for is an element called div which is section. So div is divider. And so a div tag basically identifies sections on a webpage. So I'm looking for a div tag with this ID here. So a main counter dash wrap ID, so the unique identifier of this particular section. And then within this section, we've got a heading one, which is here. And then the actual figure or statistic itself is contained in a sub div, div tag. And then within two span tags, we've got the actual statistic, we want to scrape. So you can see now it's getting a little bit more complicated on our previous simple examples. But again, we've been able to identify that there is a section with a unique ID. And within that section, there's a set of span tags, that contains the statistic of interest.

So now we have our two key pieces of information, we have the web address of the web page, and we have where on that web page, the information we want resides. So let's get straight into it. So we've done the visual inspection of the code. And I've just put it here in the notebook if you want to check again. So we need to move on to the next stage, which is requesting the webpage. So the few more modules that we've done, we've been used to but again, it's the it's the same group of preliminaries, you know, we need the requests module for requesting web pages, we need the Beautiful Soup module and for parsing the structure of the web page, so Python knows what it's dealing with. And a couple of kind of data handling modules. And also, again, I've written quite detailed notes on what's happening here. So we'll just get straight stuck into the web scraping itself. So again, we define the web address, and we store it in a variable called URL. We request that URL, and then we check the status code. So has it been successfully requested or not? So yes, so we've requested the web page, I've just shown you in the browser, we get a response code of 200. Therefore, it's been a successful request.

This is again, just to show you different way of, you know, naming the variables. You know, instead of calling it URL, I can call it web address. Instead of, you know, calling the results of the request your response, I can call it, you know, scrape result, or get the exact same result. Yep. So a successful request. So it's just a quick example showing you, you know, that, while variable names have some limits on what they can be, and, you know, you've got largely unlimited choice and how you name it. So it's up to you, whatever makes sense. It's your interpretable by you, you can choose. So again, just a very quick look at the metadata associated with the request. So here's the date and time that I've made the request. And again, a lot of stuff, which is really only of interest to your web developer. So how long did it take for the server to return the webpage to me, stuff like that, you know, it's not, it's not terribly relevant to our purposes.

So let's take a look at a snippet of that web page. So again, we haven't told Python, we're dealing with a web page, it just thinks we're dealing with a mass of text. So here's the first 1000 characters, you know, off the webpage. And we've requested, which is, in essence, just this kind of top bit of code here. And you can see the doctype. So that's just the first 1000 characters. That moves us on to our second step, which again, is telling Python, hey, we're dealing with a web page. So let's, you know, work with it on those grounds. So given us the the Beautiful Soup module, it's a very detailed web page with lots of content. So what we're going to do, again, is just a quick little sample for demonstration purposes. So we'll just interpret the first 1000 characters as if they were an HTML page. And now take a look at it in Python. Yeah, so here's the first 1000 characters. Now Python recognises that it's HTML. As we've learned before, HTML is hierarchical. And now we can see near the tag, and the nested structure of HTML. So now Python is understanding that what we've requested at this link is a web page. And now let's start picking out the bits of information we need. So again, what I want to find is well, rather our sections then that are called divs. So I'm going to try and find all of the div tags that have the unique ID of mean counter dash, right. So it sounds obviously a little bit counterintuitive that, you know, finding all the sections that have a unique ID. This is obviously not unique to an individual section, there's three sections. But that's absolutely fine. So basically, the results of the screen, or the results of this process, will find all of the sections. So this is section one, this is Section two, and this is section three. So that's what we're asking Python to find. But let's take a look at the results.

So great. That seems to have found certainly the first one with the Coronavirus cases. It's found the second section with the third statistic, and it's fun. The third one with the numbers of recovered patients. So great. So what that has returned that the Python is something called list. So I've now got a list of results. Because it's a list, then I can start counting how many elements are in the list. And I can loop over the list, I can say for every element in the list, you know, print its contents, for example. So if I use the length function in Python, it'll tell me how many sections I found. So I found three sections. Excellent. That's good to know. And then I can loop over those sections and say, you know, print, print the details and try. So here's the first div. Here's the second. And here's the part. So because now I know I have a list and the list has an order. So the first element in the list is relates to the Coronavirus cases, the second element in the list refers to the debts and the third refers to the numbers have recovered. And I can use that structure of the list to pull out the information that I need. So for example, and element one of the list which perhaps counter intuitively is identified by the number zero. So in Python, and it starts counting from zero. So the first element in the list is identified by the index zero, the second element by the number one, and so on. So it's just a little piece of information you need to keep in mind when you're using Python. But it starts counting from zero. So what I want to do then is extract the information within those div tags, specifically the span tags that are within the div tags. And then I want to save those to a variable.

So it's best usually to work from right to left and perhaps counter intuitively. So the first element of the list that's called section Find the span tags, I know there's only one set of span tags within the div. And here it is. So find the span tags, take the text that's contained within the span tags. And then I'm just doing a little bit of cleaning up, I'm saying if you find any whitespace, and just deleted, and if you find any comments as well take, take those out. So that's just a bit of data cleaning and formatting. That's not essential to the extraction. That's just a little bit of tidying up. And again, so for debts again, I go to the second element of the list, that's called sections, find the span tag, extract the text from the span tag, replace any columns that I see and save the results to this variable. If there's enough time for me, let's actually see if it works. So excellent. It also have a variable called cases, which contains this statistic, a variable called that switch contains this one and a variable called recoveries, which contains that statistic, as well. So we very successfully requested this web page, parsed it as HTML, and extracted the three key statistics that we were interested in. Again, we go to the the final step in the web scraping process, I haven't really scraped a lot of data here, you know, we can do a lot more than just two or three statistics. But just so we were doing things properly. And what I do now is I'm going to create a Downloads folder, if it doesn't exist, this is just a good idea. Just so you know, everything your web scraping goes into a particular folder. And it already exists on my machine, so it doesn't create it. All I'm doing now is basically creating a CSV file. So basically a spreadsheet. And the spreadsheet will have three variable names or three columns, you know, the cases, that's recoveries, and then to the first row of that file, I'll write those three statistics that we extracted. So that's all that's going on here. It's just setting up the file itself, and you're creating the variable names for the file. It's opening the new file, it's writing to it. And it's writing the information that we constructed. And here we go. So we're getting today's these two so we can name the file a bit more appropriately. Again, Did I do anything that actually worked, so we can check using Python? So I can list the contents of the Downloads folder? So again, you can see a couple of files have been created. Excellent. And is there anything in any of those files? Yes, there is. So the information that I scraped, has now been written to the CSV file. And just to have, again, a manual kind of demonstration of how that has worked. Just so I'm not pulling any wool over your eyes. Here are the statistics. Great, so another very successful web scraping, not terribly exciting. One roll of observations is not really gonna justify any kind of research funding herati support for what I'm trying to do. So this next part of the lesson, we're going to focus on actually scraping Coronavirus data for every country in the world.