

Remote Hosts and Web Servers

ITX2000 - CW2

Submitted by:

Clerissa Lewis – M00692462

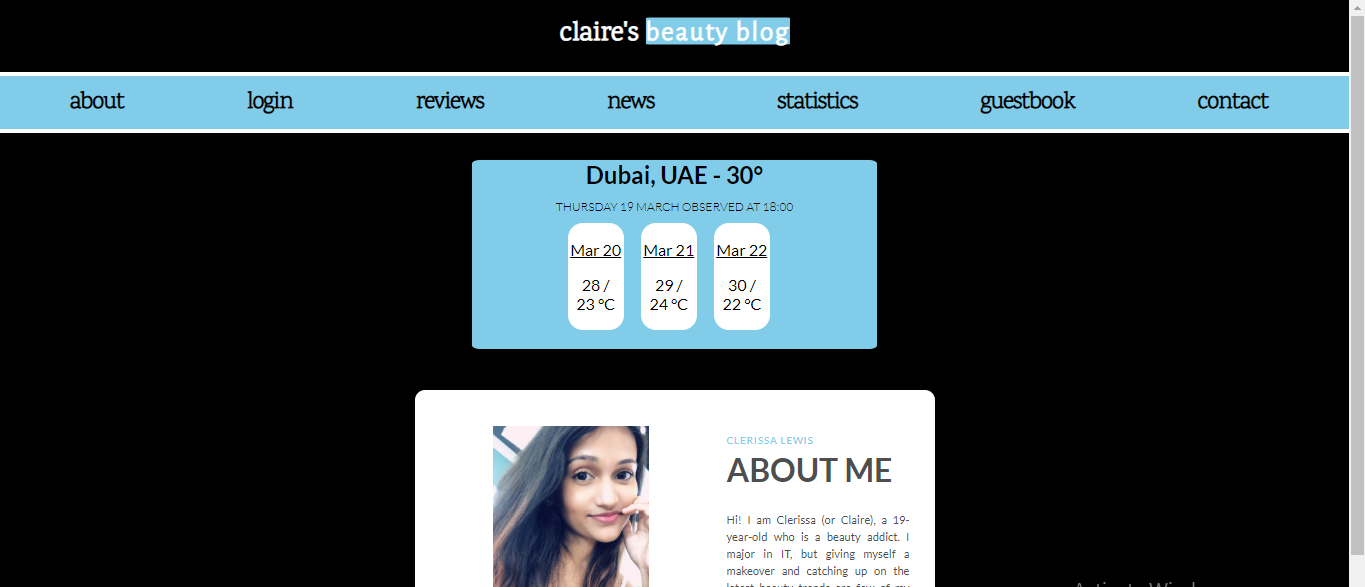
**My Report**

**About This Coursework**

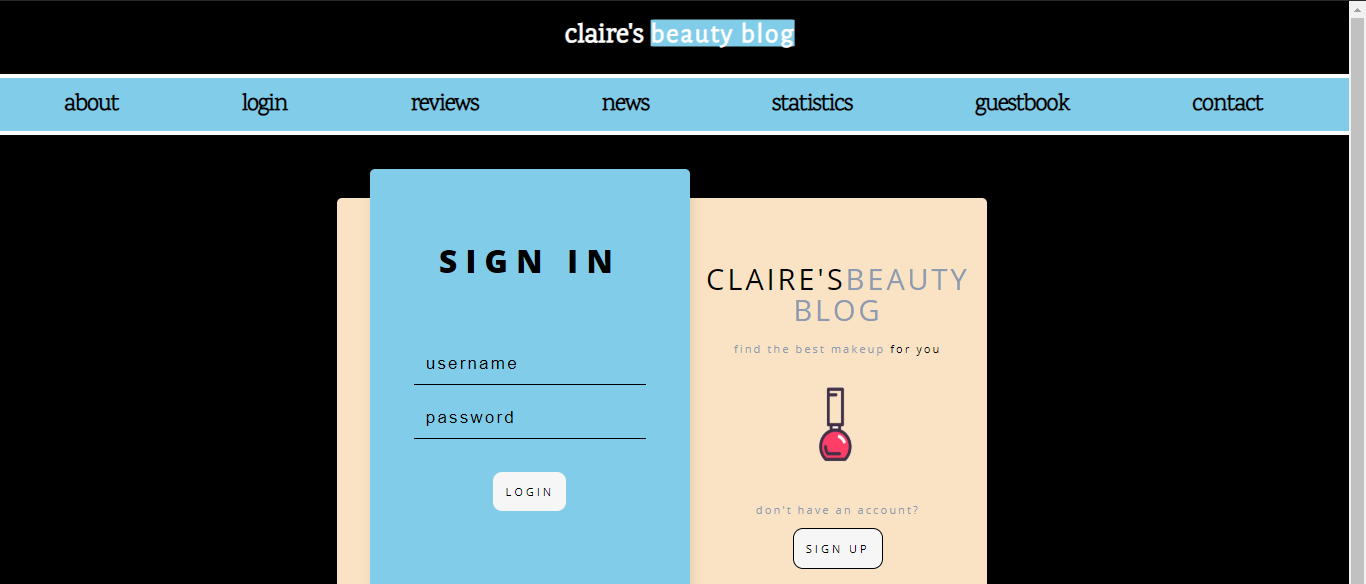
For this coursework we were expected to use the website from Coursework 1 (based on makeup), extract/scrape information from different websites using Beautiful Soup and subsequently display them on my website.

**Screenshots of All My Pages**

Home Page:



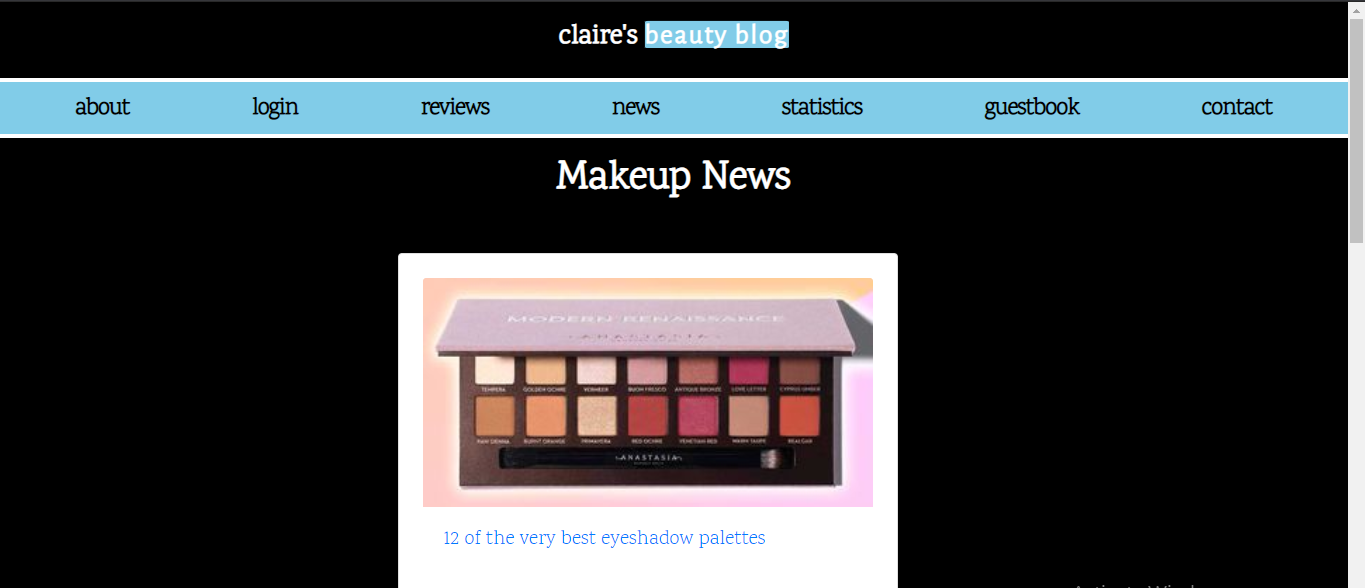
Account Page:



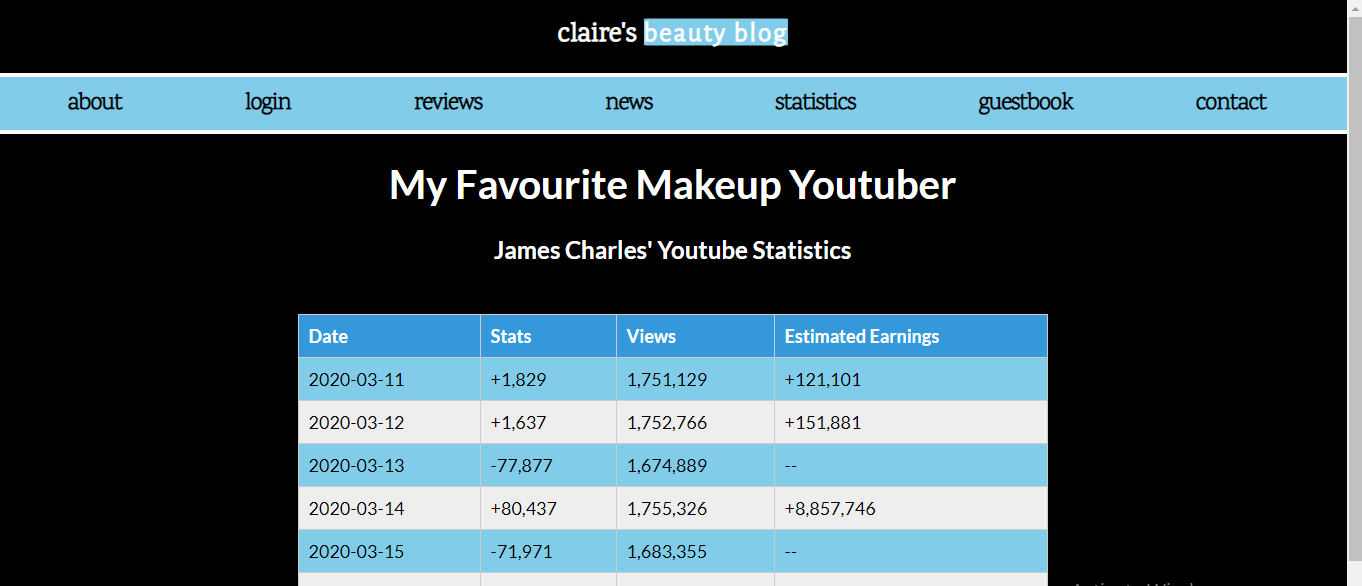
Search Results Page:



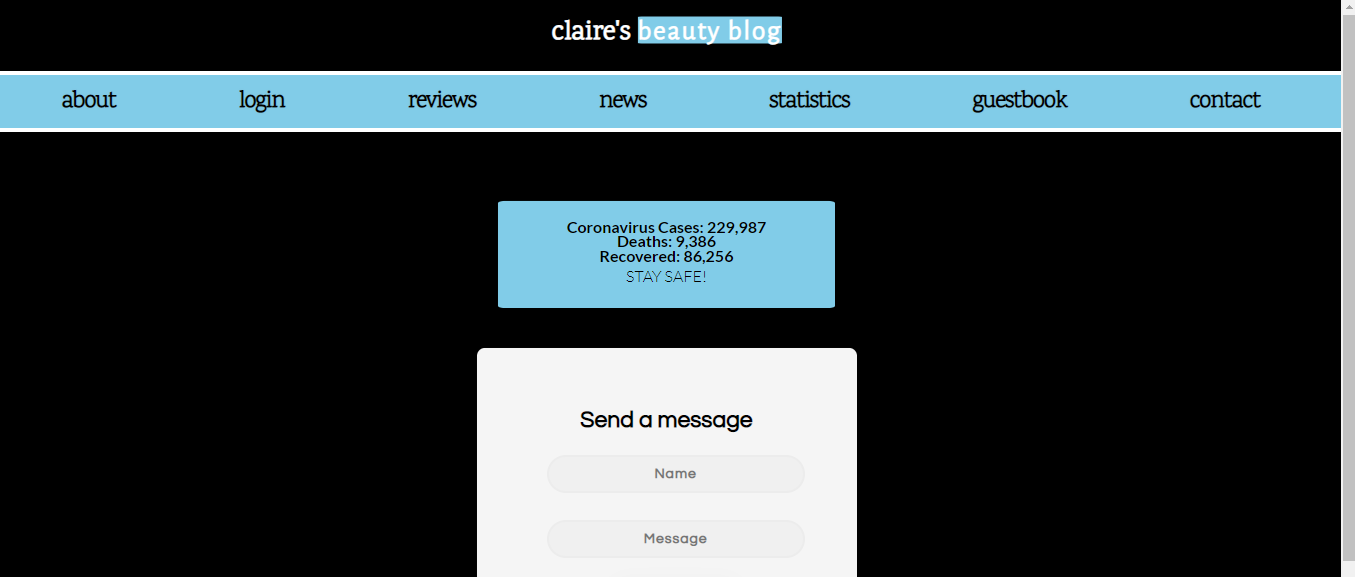
Makeup Related News Page:



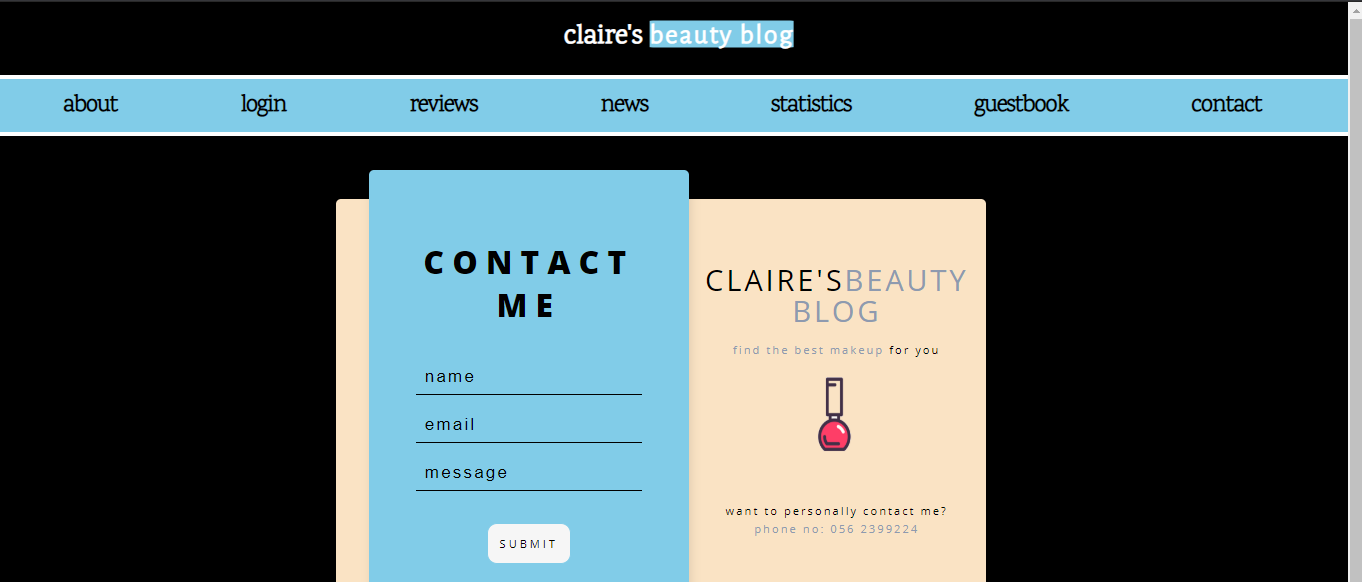
YouTube Statistics Page:



Guestbook Page:



Contact Page:



**What Did I Scrape?**

**Dynamic Information:**

I scraped a total of 6 different blocks of information from 6 different websites. All of these blocks changed in content at one or the other point on the original website. As a result, the information gets updated on my website as well. The different blocks are explained in brief as follows:

1. **Search Bar from Mirabeauty.com**

I created a simple search bar on my website that extracts details about any query that is searched. For example, if the query “best foundation” is searched, the website displays a foundation with the highest rating. I scraped this data and displayed it below my search bar.

A code snippet of this section is shown later in the report. Even though this information might seem static, the best product that satisfies the query can change through time. Another product with a higher rating might surpass it. Thus, this is dynamic in nature. A screenshot of the search part is shown as follows:

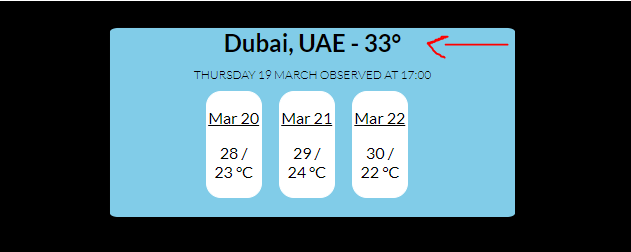


1. **Today’s Weather**

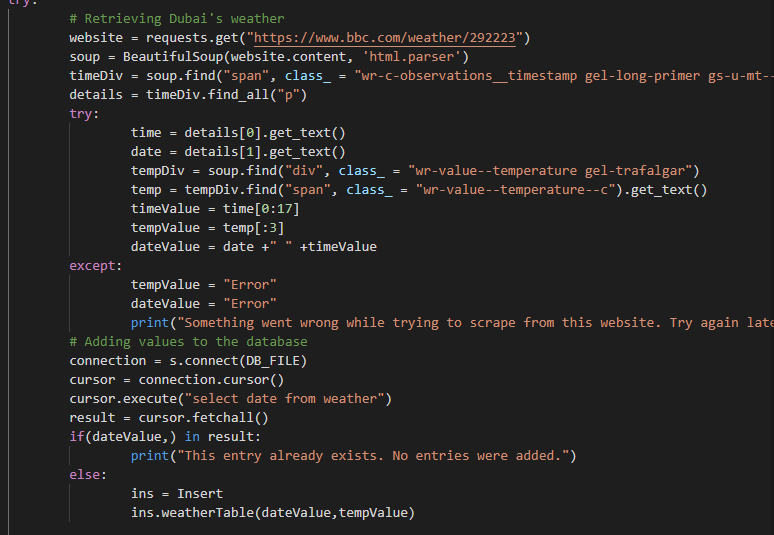
I used BBC’s website to scrape Dubai’s temperature. This information was updated *every hour* on my website. I used a simple find\_all() query to extract the <p> and <span> tags and subsequently styled it on my website. I also stored the scraped fields (date, time and temperature) in the database as a table named ‘weather’. This table avoids adding duplicated fields through the use of an if statement.

**Problem(s) Faced and How I Solved It:** When extracting the temperature, I realized that get\_text() scraped another piece of information that I did not require. Thus, I sliced the variable to only include the temperature value.

A screenshot of this part is shown as follows:



A snippet of the code is shown as follows:



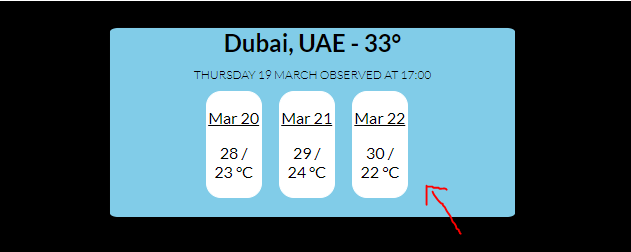
1. **Weather Forecast for the Next Three Days**

This information is updated every day. Even though my forecast information is stored in the same div as the daily weather, I consider it to be different. I used a different website, and also a completely different approach to extract the information. Instead of using a ‘for in range()’ loop to extract information about the next three days, I used Python iterators. It gave me the same result and I also had the chance of learning something new. I stored this information in the database in a table called ‘threeday’.

**Problem(s) Faced and How I Solved It:**

* The website stored the days in Arabic, which I didn’t want. Thus, I did not scrape the days from the website and instead used the ‘datetime’ module in Python to get the dates of the next three days in “mm dd” format from the current day.
* When displaying the three days on the website from the database, it did not display it in order. Thus, I had to change the execute statement and use two ORDER BY clauses to display the dates in ascending order and the “rowid” column in descending order.

The screenshot of this part is displayed as follows:



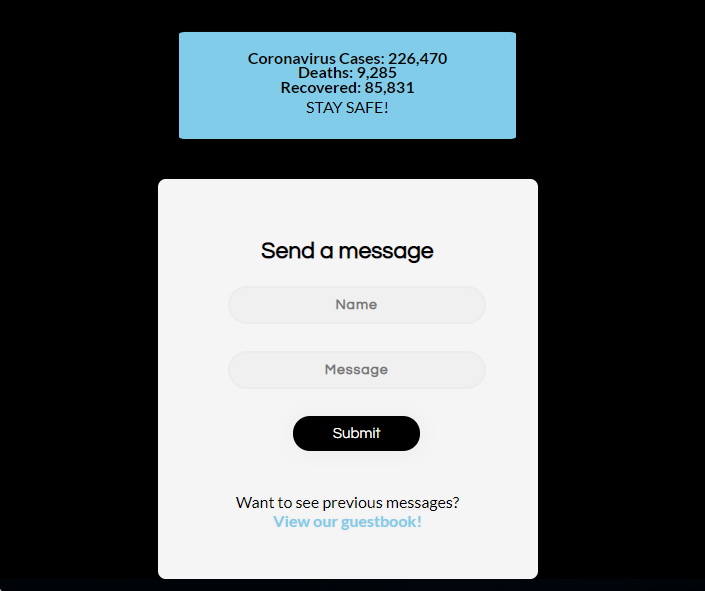
1. **Live Numbers of Coronavirus**

Due to the recent Covid-19 outbreak, I decided to display the numbers of how many people are infected on my website as a cautionary alert to spread awareness. This information is dynamic as the original website updates the numbers really frequently.

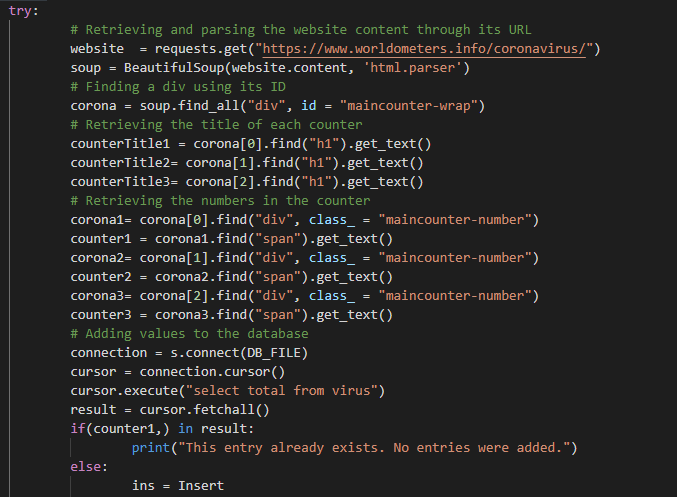
I extracted the information by using the find statement and subsequently using get\_text(). This is displayed on my guestbook page, as anyone can access it. I stored this in a table called ‘virus’.

**Problem(s) Faced and How I Solved It:** The website sometimes faces downtimes. To tackle this, I used ‘try-except’ that would display “Error” in case of this situation.

The screenshot of this part is shown as follows:



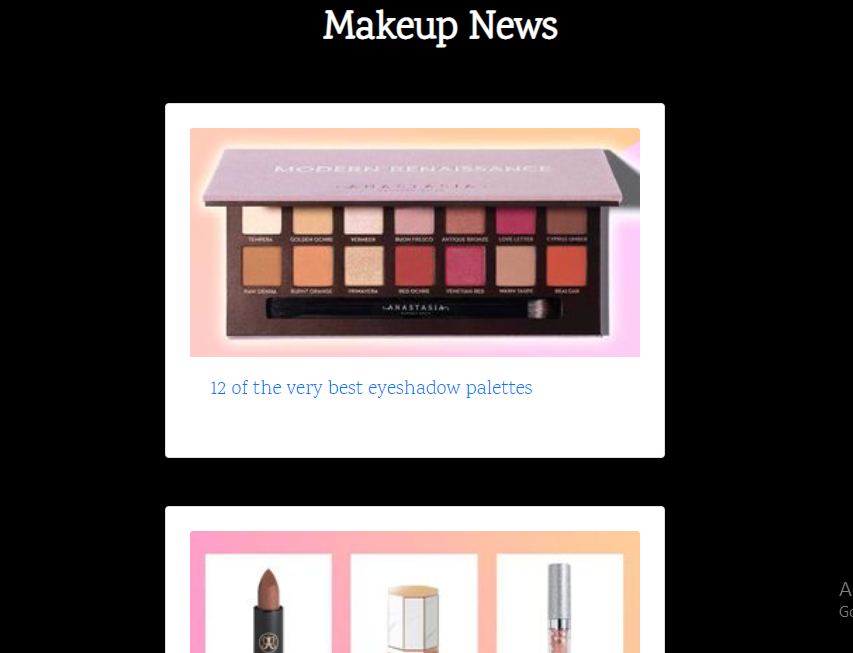
A snippet of the code is shown as follows:



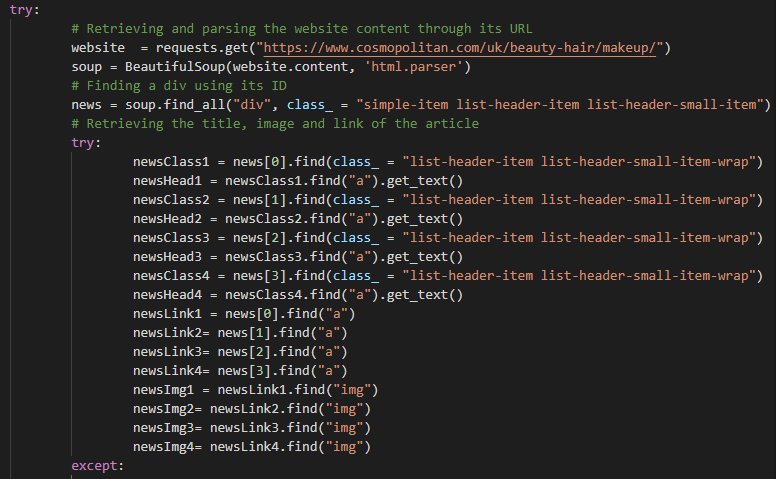
1. **Makeup News**

I displayed four news articles which included links to the original article from the Cosmopolitan website. The information gets updated on a weekly basis. These were extracted using simple find and find\_all statements. To scrape the images, I used the ‘src’ part of the <img> tag and did a similar thing to scrape the links (‘href’ part of the <a> tag).

A screenshot of this section is displayed as follows:



A snippet of the code is shown as follows:

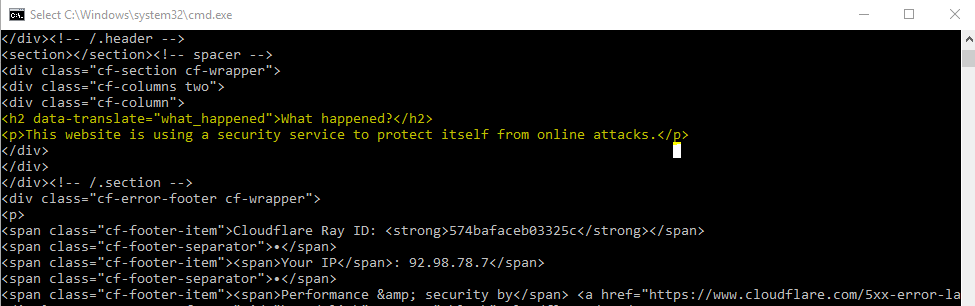


1. **YouTube Statistics of my Favourite Makeup YouTuber**

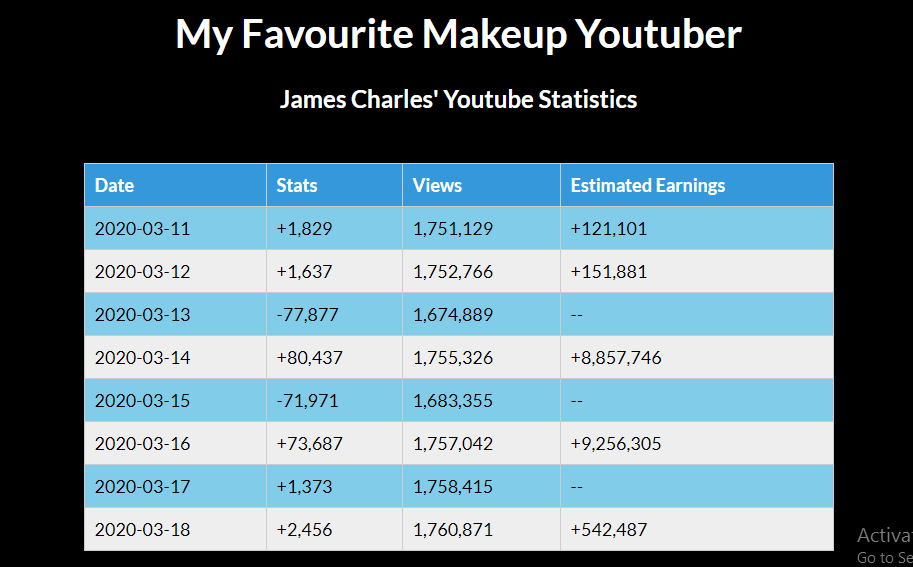
I decided to extract a table that displayed the statistics of James Charles in the past few days. I used a ‘for in range’ loop to extract each row and its data, and subsequently add it to the database in a table called ‘stats’. This information gets updated daily.

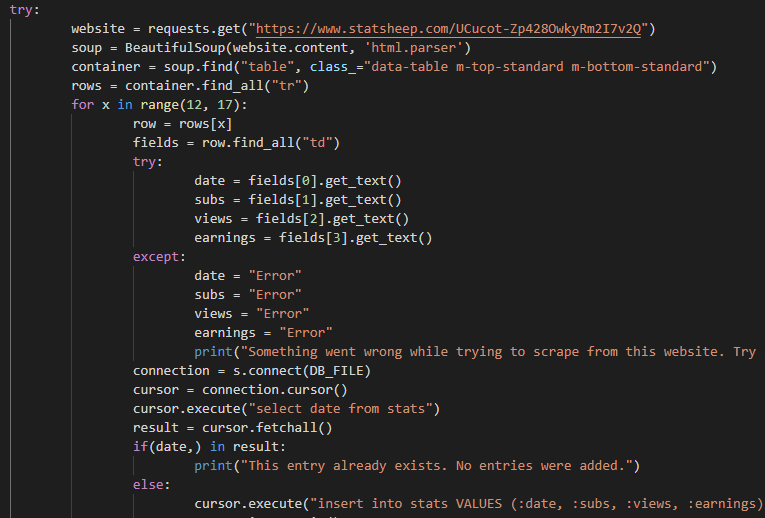
**Problem(s) Faced and How I Solved It:**

* When directly trying to scrape and display the table on my website without the use of a database, it only displayed the last column. This is because I could not put the render\_template statement which passes the variables in the for loop. Thus, to solve this, the rows first get added to the database while ensuring no duplication takes place. Only after this is the table displayed on my website.
* Before scraping from statsheep.com, I tried scraping a similar table from socialblade.com. However, the site displayed the following error:



A screenshot of this part is shown as follows:





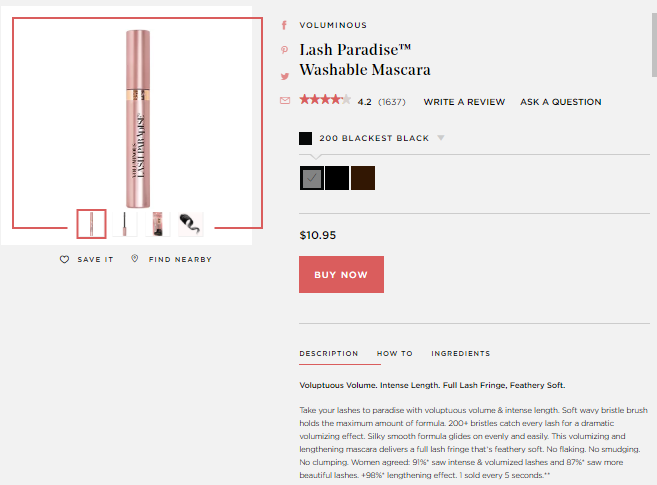
**Static Information:**

1. **Descriptions of Makeup Products**

Instead of hard coding the description, title and price of my products from Coursework 1, I scraped information from actual websites. I used a total of 4 different websites to do the same. So I extracted at least 10 different static pieces of information to achieve this.

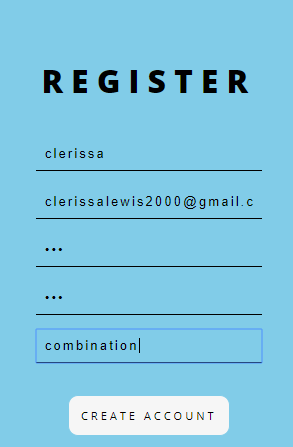
A comparison of the original product page and my page is shows as follows:

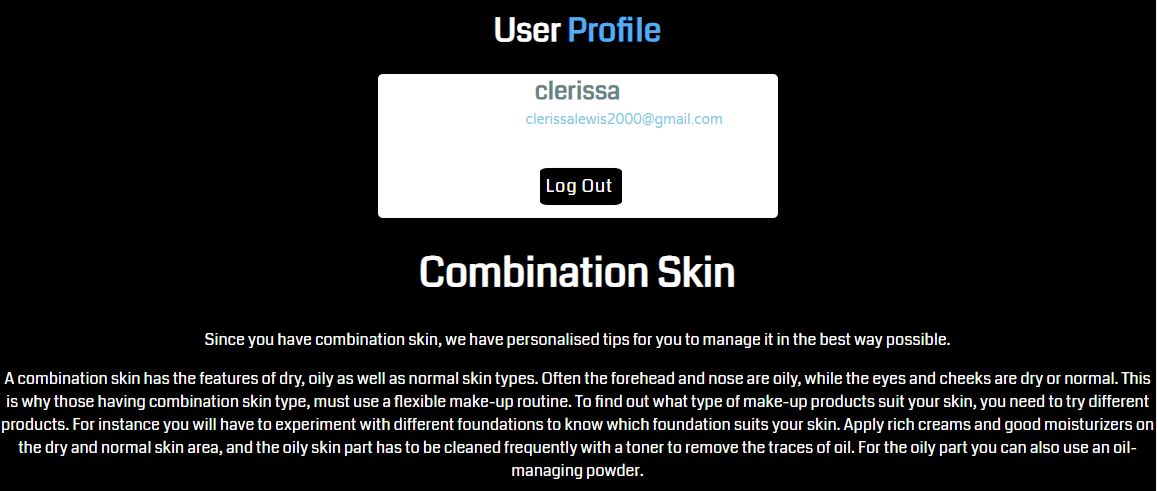




1. **Makeup Tips**

I displayed tips from a website on the account page. This is shown as follows.





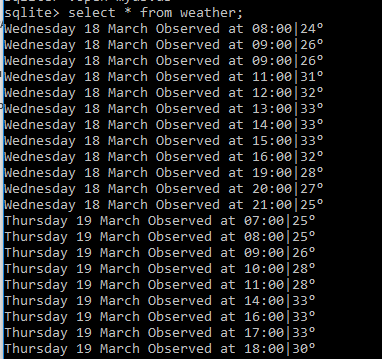
1. **Coronavirus Headings**

As mentioned above, I scraped live numbers of the outbreak which constitutes as dynamic. But I also extracted three different headings, ‘Coronavirus Cases’, ‘Deaths’ and ‘Recovered’. These constitute as static as they don’t change.

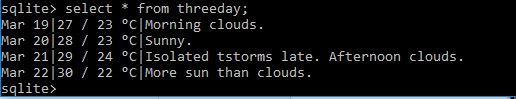
**Table Contents in my Database**

I have a total of four tables in my database that store dynamic data and are subsequently displayed on my website. Screenshots are displayed as follows:

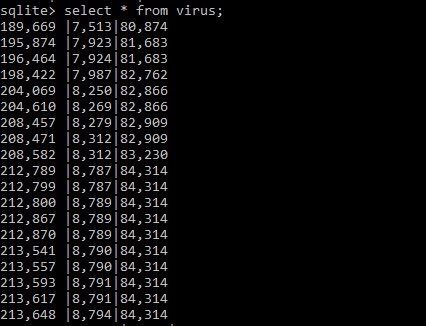
Weather table:



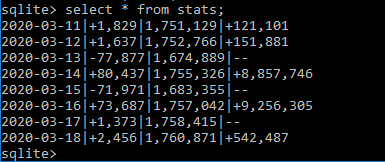
Three Day Forecast table:



Coronavirus table:



Youtube stats table:



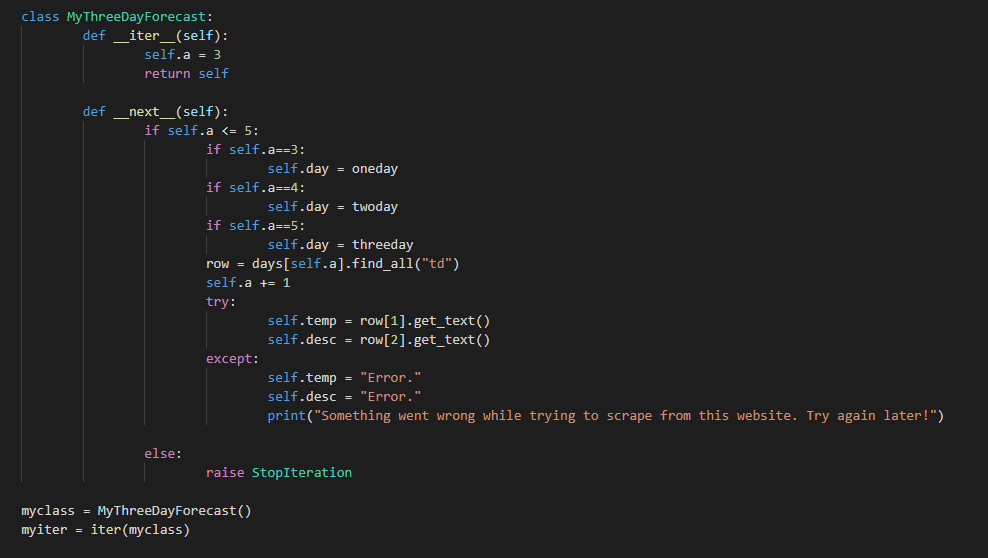
**Special Features in Terms of Code**

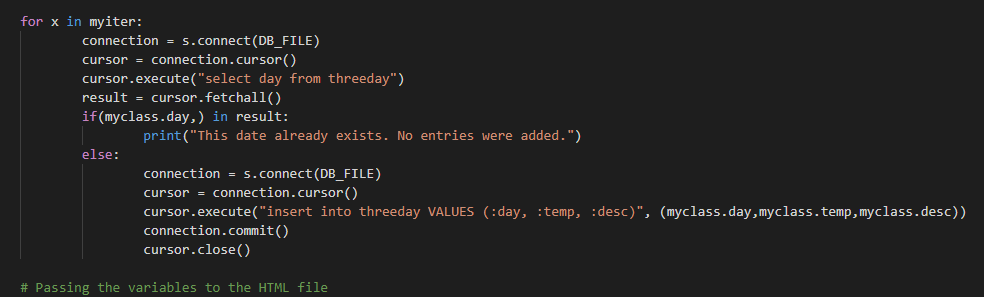
1. **Python Iterators**

I used the concept of iteration in Python to convert objects and classes to be able to store multiple values through the use of methods like \_\_iter\_() and \_\_next\_\_(). To stop the iteration process, I used the StopIteration statement.

This object can then be iterated through to subsequently retrieve the values. I used this for my three day weather forecast.

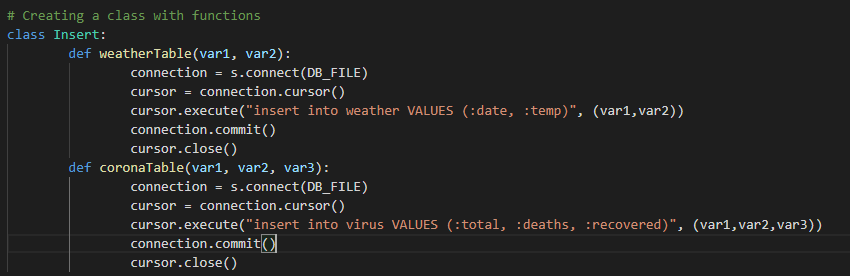
A screenshot of the code is displayed as follows:





1. **Using Classes**

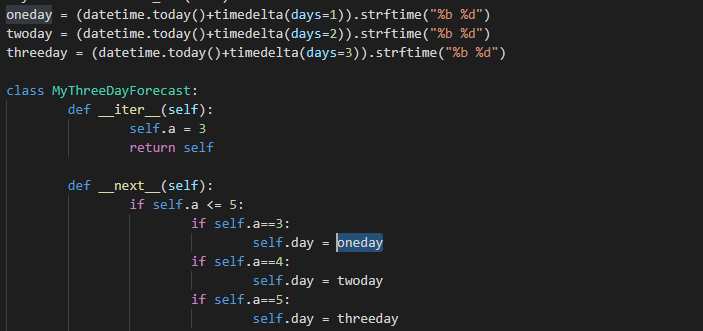
I used a class that contained methods that could be called to insert values into my database. A screenshot of this is shown as follows:



1. **Python Datetime Module**

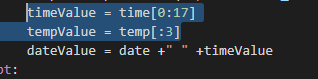
As mentioned previously, I used the datetime module to retrieve the dates of the next three days from the current day. I also changed the format of the way it is printed.

A screenshot of this is provided as follows:



1. **Slicing of Strings**

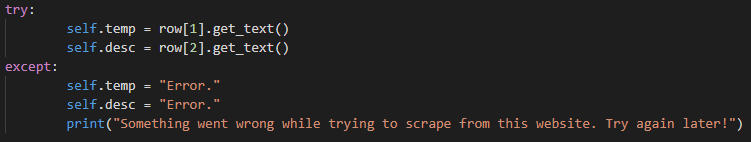
As mentioned before, I used slicing to split a string and only retrieve the section I needed. A screenshot is shown as follows:

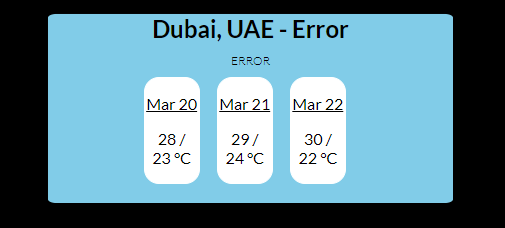


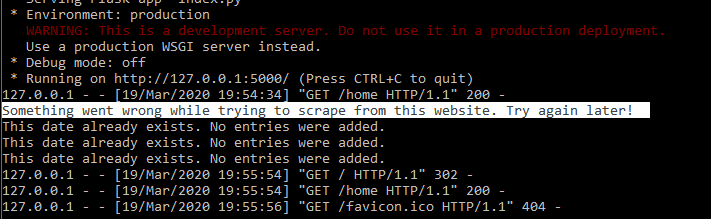
In the screenshot above, the timeValue variable is sliced from the 1st character to the 17th character. In the tempValue variable, only the first 3 letters are taken.

**Exception Handling – Try and Except**

Apart from the try and except statements that are in my Coursework 1 code, I also included a few others in case the websites I scraped from faced any downtime. A depiction of this taking place is shown as follows with the code:



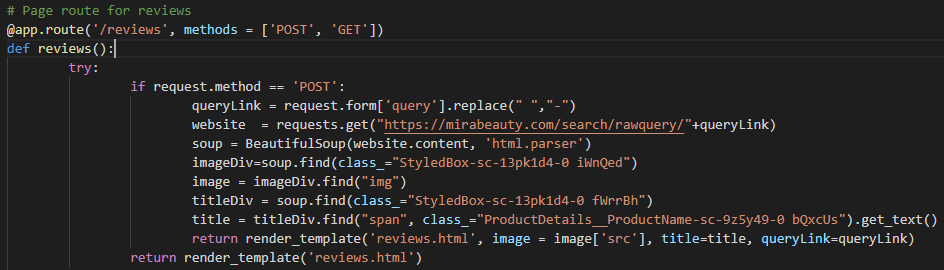




**Features/ Concepts I Learnt Myself in Terms of Scraping**

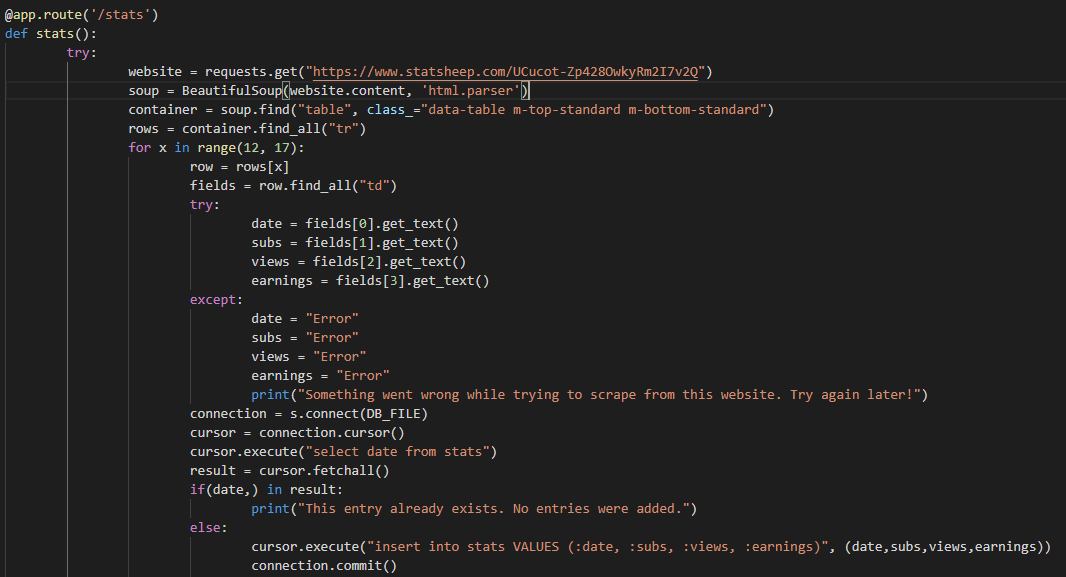
1. **Search Bar – Makeup Products**

I implemented a search bar that takes a query, adds it to a URL and shows the first search result from mirabeauty.com. This is an idea I thought of myself. The code is shown as follows:



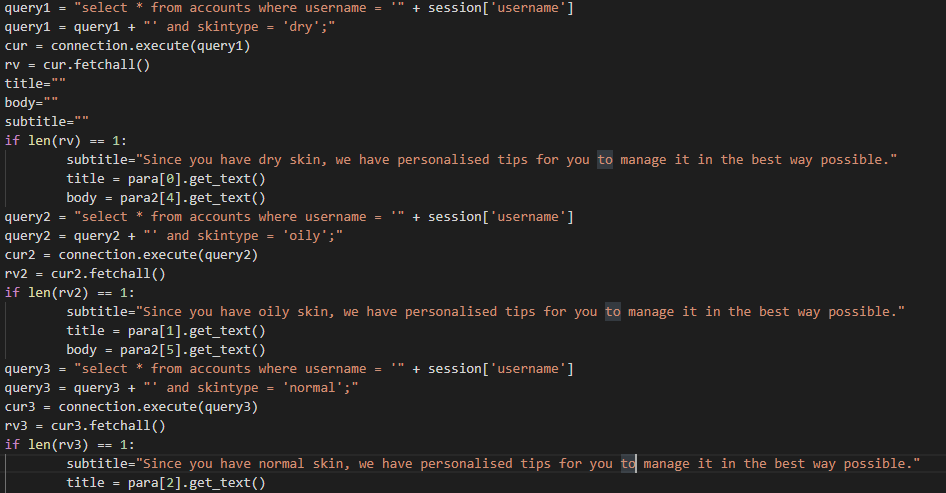
1. **Table Extraction**

Scraping a table from a website was not easy, and took a lot of trial-and-error to get right. I also had to ensure that no rows got duplicated in the database, and only the updated information would be added. Since this was not taught in class, I consider it something I learnt myself. The code is shown as follows:



1. **Tips Based on Skin type**

I changed my registration form to include a field that asks for the user’s skin type (normal, oily, dry and combination). Based on their skin type, tips are displayed in the profile page under account information. The code is shown as follows:



**List of Links I Scraped Information From**

https://www.bbc.com/weather/292223

https://www.timeanddate.com/weather/united-arab-emirates/dubai/ext

https://www.cosmopolitan.com/uk/beauty-hair/makeup/

https://www.statsheep.com/UCucot-Zp428OwkyRm2I7v2Q

https://mirabeauty.com/search/rawquery/foundation

https://www.worldometers.info/coronavirus/

https://www.makeupfornoobs.com/make-up-tips-according-to-skin-types/

https://www.lorealparisusa.com/products/makeup/eye/mascara/voluminous-lash-paradise-washable-mascara.aspx?shade=200-blackest-black

https://www.ulta.com/prep-prime-fix-primer-setting-spray?productId=xlsImpprod15921204

https://www.hourglasscosmetics.com/products/veil-translucent-setting-powder

https://www.ulta.com/barepro-performance-wear-liquid-foundation-broad-spectrum-spf-20?productId=xlsImpprod16321440