Intro

The main objective of this project was to help a bariatrician to study their market, and find better opportunities to offer a fair price and better service.
For this task we decide to make a Web Scrapping on "Doctoralia.com", a page where doctors from all areas offer their services. There, we search for "Cirugía Bariátrica" and find some pages with useful information. We decided to take name, speciality, number of opinions, address, link to map location, and price.

Extract while transform

It didn't take much time to realize that we can transform the data while we extract it. In first place, as usual, we had to use the "strip" and "replace" functions to clean data, which normally came with html, extra symbols or characters. For data like name, address, link map and specialities this was enough. As a precaution we add a `try:` and `except:` function to add a NaN where it could be a missing value.

```
### Name
  try:
      doct_inf = {}
  name = result.find("a", {"data-ga-label":"Doctor Name"})
  clean_name = name.text.strip()
  except:
  clean_name= "NaN"
      doct_inf["name"] = clean_name
```

But "price" and "opinion" gave us some extra problems. Price and opinion came with a lot of extra text or characters as a currency symbol among others. So we have to use several replace functions for get rid of them. And also transform the request data into an integer, using the *int()* function of python. Once we did this, we got the problem that some values were missing or not given, so we had to use a "try" and "exception" function to fill the value with "0" once it find some error.

```
  #### Price
try:
   price = result.find("p", class_="text-nowrap")
   clean_price =  int(price.text.replace("•","").replace("$","").
```

```
    replace("desde","").replace(",","").strip())
except:
    clean_price=0
    doct_inf["price"] = clean_price
```

First we try to use *splinter* to visit the different search pages, but we find unexpected navigation problems who couldn't find the button. So we decide to use directly the url, iterate through the result pages using a rise counter from a for loop, and using a "if" statement to indicate to the computer where to stop, while telling it to just explore pages where the url has the word "&page".

```
url = "https://www.doctoralia.com.mx/buscar?
filters%5Bservices%5D%5B0%5D=1189&q=Cirug%C3%ADa+bari%C3%A1trica&loc=Ciudad+de+
M%C3%A9xico&page="
for pages in range (0,10):
    try:
      page_counter += 1
      browser.visit(url+str(page_counter))
      time.sleep(2)
      if "&page" in browser.url:
```

LOAD

As the data was incomplete and had duplicates, we decided to save it into a list and then transform the data into a **pandas Data Frame**. There, we use the functions drop_na and drop_duplicates to get rid of this deficiency. Then we transform our DataFrame into a dictionary using .to_dict('records'), who give us a list of dictionaries. Then we iterate through the list to add each dictionary in our Mongo data base. We decided to use Mongo because most of the data were incomplete and because we wanted to practice.

```
doctor_df = pd.DataFrame(complete_data)
doctor_no_NAN = doctor_df.replace("NaN", np.nan)
doctor_df_clean = doctor_no_NAN.dropna(axis=0, how="any")
```

```
doctor_no_dup = doctor_df_clean.drop_duplicates("name")

final_df = doctor_no_dup.sort_values("price", ascending=False)

dicts = final_df.to_dict('records')

for i in dicts:

    collection.insert_one(i)
```