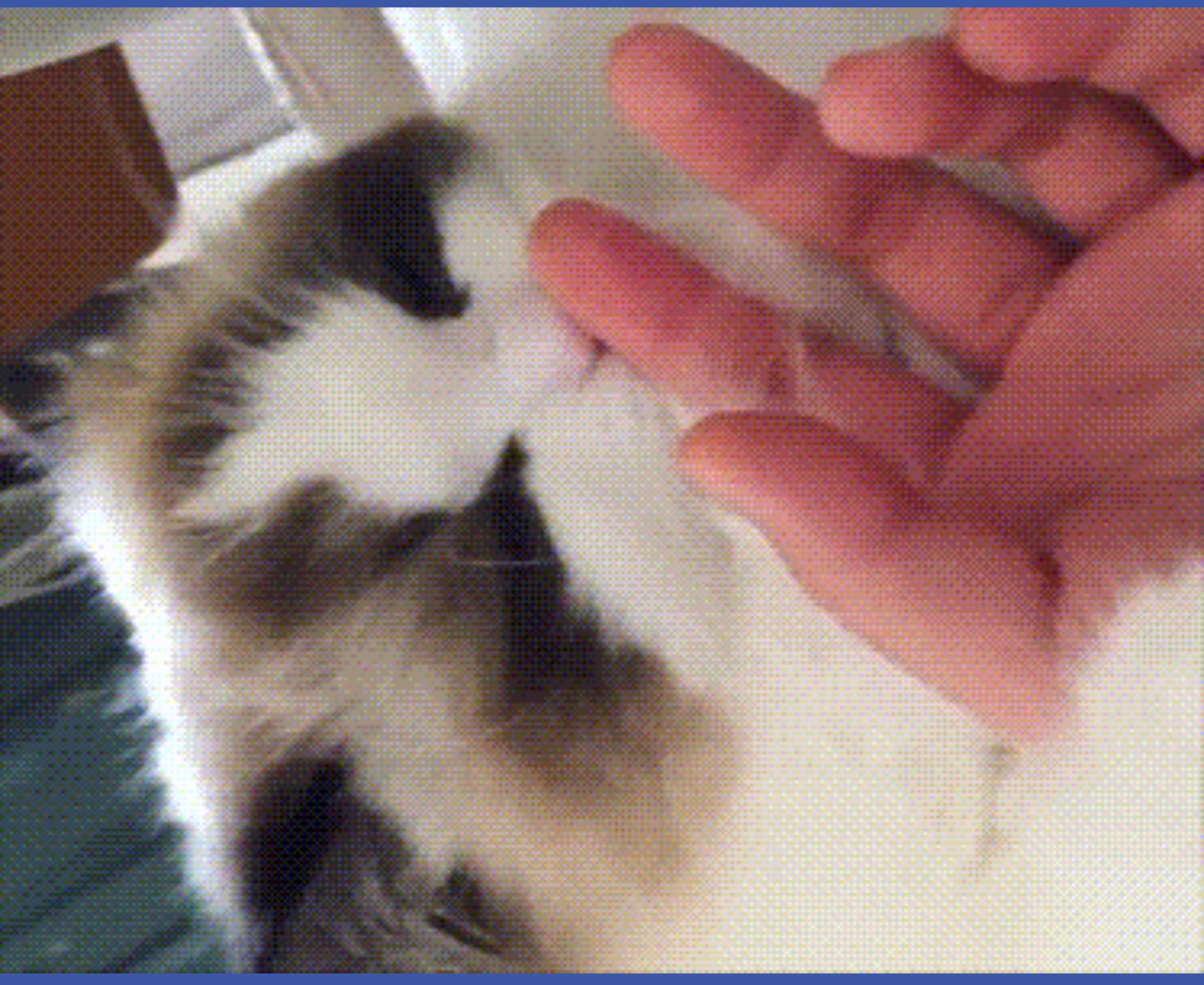
Data Science Fundamentals
Lesson 9

Hay Kranen

Monday October 21st, 2018



Today's programme

The next two weeks

Recap

Reddit Plots

Break

Functions

Web scraping

Lunch break

Week	Monday	Wednesday
43	22/10 Lesson 9	24/10 Lesson 10
44	29/10 Lesson 11	31/10 Lesson 12
45	5/11 **12.00 Deadline**	-

RECAP

```
CLIENT_ID = "Y0idhGvtxNY0oA"
CLIENT_SECRET = "_8nZnFMxafai7a0Mihqy8PCrBIc"
USER_AGENT = f"python:[{CLIENT_ID}:0.1 (by /u/Husky)"
```

```
CLIENT_ID = "x"
CLIENT_SECRET = "x"
USER_AGENT = f"python:{CLIENT_ID}:0.1 (by /u/x)"
```

In [37]:
```
print(newlist)
```

```
[{'title': 'What was the cringiest thing you did in your first relationship?', 'score': 25260, 'co
mments': 10213}, {'title': 'What's the dumbest thing you've heard someone say that made you wonder
how they function on a day to day basis?', 'score': 50720, 'comments': 28783}, {'title': 'What is
something that HAS aged well?', 'score': 5140, 'comments': 6161}, {'title': 'What have we all conv
eniently forgotten about?', 'score': 5058, 'comments': 4823}, {'title': 'What made you laugh the h
ardest in your entire life?', 'score': 7745, 'comments': 3777}, {'title': 'What is the worst thing
to cheap out on?', 'score': 477, 'comments': 713}, {'title': "What's an experience you don't ever
want to go through again?", 'score': 4880, 'comments': 4783}, {'title': "Men of reddit, what do wo
men do that they think is sexy, but actually isn't?", 'score': 546, 'comments': 1314}, {'title':
'What benefits did you notice after losing weight?', 'score': 1549, 'comments': 1278}, {'title':
"What is the most physically painful thing you've ever gone through?", 'score': 231, 'comments': 6
15}]
```

```
df = pd.DataFrame({'mass': [score, score_total]},
                  index=['High', 'Total score'])
plot = df.plot.pie(y='mass', figsize=(15, 15), fontsize=40)

fig = plot.get_figure()
fig.savefig('plot.pdf')
```

Reddit plots

Functions

A way to group code
Reuse common functionality
Local scope for variables

Simple functions

```
def say_hello():
    print("Hello!")

say_hello()
```

```
def say_hello()
```

def keyword function name

Functions with arguments

```
def say_hello(name):
    print(f"Hello, {name}!")

say_hello("Barrie")
```

```
def say_hello(name)
```

def keyword — function name — argument(s)

Return statements

```
import json

def read_json(path):
    with open(path) as f:
        data = json.load(f)

    return data

paintings = read_json("paintings.json")
```

```
def read_json(path)
```

def keyword — function name — argument(s)

Multiple arguments

```
import json

def write_json(data, path):
    with open(path, "w") as f:
        json_data = json.dumps(data)
        f.write(json_data)

colors = ["red", "green", "blue"]
write_json(colors, "colors.json")
```

Default argument values

```
import json

def write_json(data, path = "data.json"):
    with open(path, "w") as f:
        json_data = json.dumps(data)
        f.write(json_data)

colors = ["red", "green", "blue"]
write_json(colors)
```

functions

print()

len()

open()

methods

json.loads()

requests.get()

pandas.read_json

`pandas.`**`read_json`**(*path_or_buf=None*, *orient=None*, *typ='frame'*, *dtype=True*, *convert_axes=True*, *convert_dates=True*, *keep_default_dates=True*, *numpy=False*, *precise_float=False*, *date_unit=None*, *encoding=None*, *lines=False*, *chunksize=None*, *compression='infer'*) [source]

Convert a JSON string to pandas object

path_or_buf *: a valid JSON string or file-like, default: None*

The string could be a URL. Valid URL schemes include http, ftp, s3, and file. For file URLs, a host is expected. For instance, a local file could be `file://localhost/path/to/table.json`

orient *: string,*

Indication of expected JSON string format. Compatible JSON strings can be produced by `to_json()` with a corresponding orient value. The set of possible orients is:

- `'split'` : dict like `{index -> [index], columns -> [columns], data -> [values]}`
- `'records'` : list like `[{column -> value}, ... , {column -> value}]`

```
import json

def write_json(data, path):
    with open(path, "w") as f:
        json_data = json.dumps(data)
        f.write(json_data)

colors = ["red", "green", "blue"]
write_json(colors, "colors.json")
```

Keyword arguments

```
import json

def write_json(data, path):
    with open(path, "w") as f:
        json_data = json.dumps(data)
        f.write(json_data)

colors = ["red", "green", "blue"]
write_json(path = "colors.json", data = colors)
```

Functions calling other functions

```
import json

def print_json(path):
    json_data = read_json(path)
    json_str = json.dumps(json_data, indent = 4)
    print(json_str)

def read_json(path):
    with open(path) as f:
        data = json.load(f)

    return data

print_json("footballers.json")
```

```
import json

def print_json(path):
    json_data = read_json(path)
    json_str = json.dumps(json_data, indent = 4)
    print(json_str)

print_json("footballers.json")

def read_json(path):
    with open(path) as f:
        data = json.load(f)

    return data
```

Local scope

```
number = 40

def other_number():
    number = 50
    print(number)

other_number()
print(number)
```

```
number = 40

def other_number():
    number = number + 10
    print(number)

other_number()
print(number)
```

```
people = []

def add_person(person):
    people.append(person)

add_person("Barrie")
print(people)
```

Wikipedia API with functions

Write a terminal program that asks the user for a Wikipedia article and if they want the **description** or **extract**. Then lookup and **return** that data using a function that requires an article title and the value (description or extract). Print the returned data.

* Look at the **wikifunction-skeleton.py** file on how to access the Wikipedia API.
* Transform the API call to a function that accepts a **title** and **value** argument.
* Return different data, depending on what kind of **value** you get.
* Ask the user for **title** and **value**. Remember to clean up user input using **strip()**.
* Call the function and assign the new data to a variable.
* Print the returned data

Tips

* Note that a **return** statement can be used together with the **if** statement

Extended exercise

* Add the option to print the **desktop url** of the article.
* Split your function in **two** functions: one that handles the API call, one that handles returning the correct data.

Web scraping


Met deze ongelooflijke tover-
kracht ben ik werkelijk
onoverwinnelijk geworden!

* Every site is different
* Dynamic websites are difficult
* Sites may block you
* You need HTML/CSS/JS knowledge

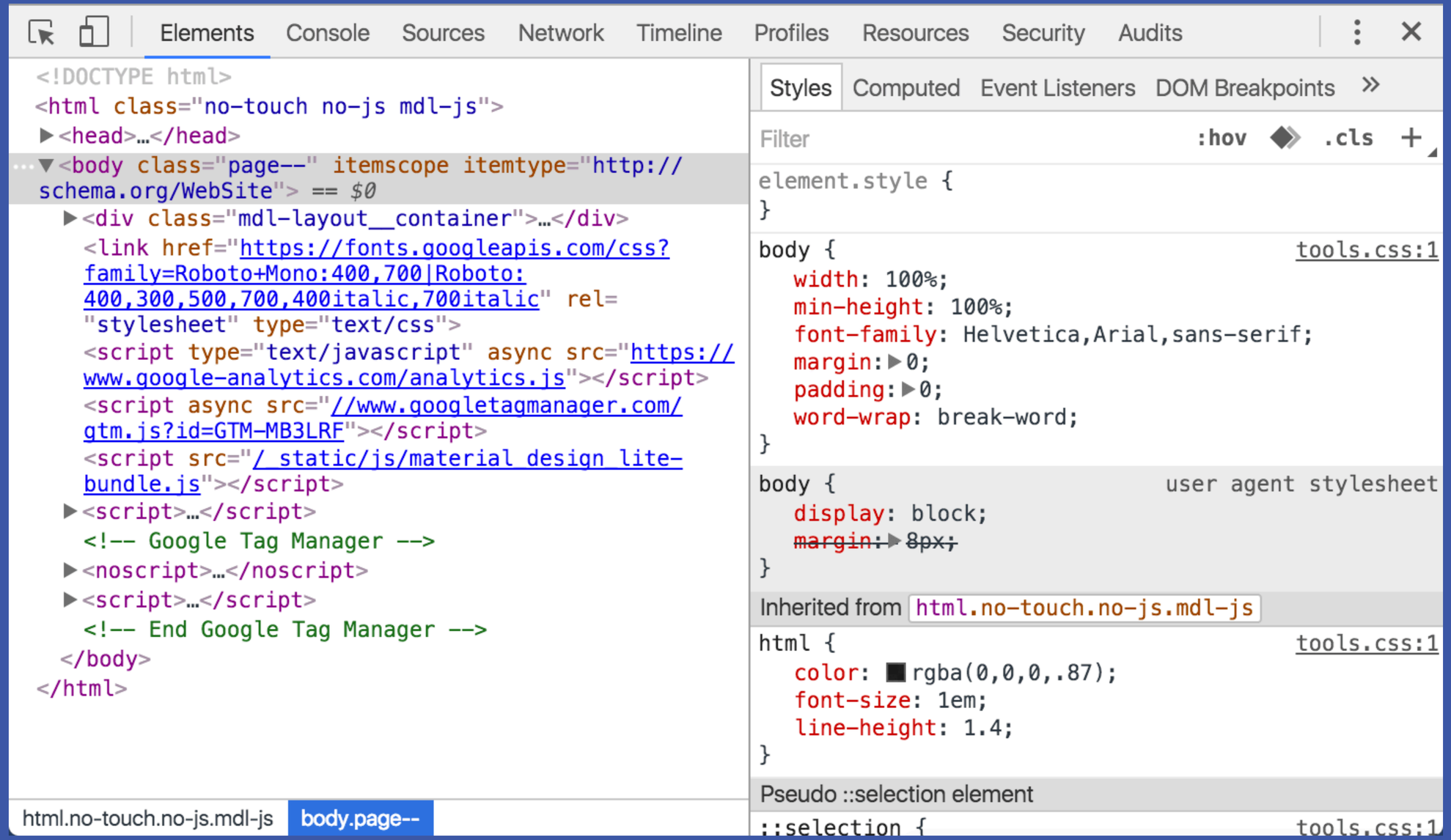

Elements
Console
Sources
Network
Timeline
Profiles
Resources
Security
Audits
<!DOCTYPE html>
<html class="no-touch no-js mdl-js">
▶<head>…</head>
▼<body class="page--" itemscope itemtype="http://schema.org/WebSite"> == $0
▶ …
<link href="https://fonts.googleapis.com/css?family=Roboto+Mono:400,700|Roboto:400,300,500,700,400italic,700italic" rel="stylesheet" type="text/css">
<script type="text/javascript" async src="https://www.google-analytics.com/analytics.js"></script>
<script async src="//www.googletagmanager.com/gtm.js?id=GTM-MB3LRF"></script>
<script src="/_static/js/material_design_lite-bundle.js"></script>
▶<script>…</script>
<!-- Google Tag Manager -->
▶<noscript>…</noscript>
▶<script>…</script>
<!-- End Google Tag Manager -->
</body>
</html>
html.no-touch.no-js.mdl-js
body.page--
Styles
Computed
Event Listeners
DOM Breakpoints
Filter
:hov
.cls
element.style {
}
body {
tools.css:1
width: 100%;
min-height: 100%;
font-family: Helvetica,Arial,sans-serif;
margin:▶0;
padding:▶0;
word-wrap: break-word;
}
body {
user agent stylesheet
display: block;
margin:▶8px;
}
Inherited from
html.no-touch.no-js.mdl-js
html {
tools.css:1
color: rgba(0,0,0,.87);
font-size: 1em;
line-height: 1.4;
}
Pseudo ::selection element
::selection {
tools.css:1

Table Of Contents

Beautiful Soup Documentation

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.

These instructions illustrate all major features of Beautiful Soup 4, with examples. I show you what the library is good for, how it works, how to use it, how to make it do what you want, and what to do when it violates your expectations.

The examples in this documentation should work the same way in Python 2.7 and Python 3.2.

You might be looking for the documentation for Beautiful Soup 3. If so, you should know that Beautiful Soup 3 is no longer being developed, and that Beautiful Soup 4 is recommended for all new projects. If you want to learn about the differences between Beautiful Soup 3 and Beautiful Soup 4, see Porting code to BS4.

This documentation has been translated into other languages by Beautiful Soup users:

- 这篇文档当然还有中文版.
- このページは日本語で利用できます(外部リンク)
- 이 문서는 한국어 번역도 가능합니다. (외부 링크)


file:///Users/hay/htdocs/git/hu
Japanese Restaurant "Arigato"
Menu
Sushi
Tempura
Ramen
Order here!

```html
<!doctype html>
<html>
<head>
    <title>Japanese Restaurant "Arigato"</title>
</head>
<body>
    <h1>Japanese Restaurant "Arigato"</h1>
    <h2>Menu</h2>

    <ul>
        <li>Sushi</li>
        <li>Tempura</li>
        <li>Ramen</li>
    </ul>

    <a class="order"
        href="http://www.example.com">
        Order here!
    </a>
</body>
</html>
```

```python
from bs4 import BeautifulSoup

with open("restaurant.html") as f:
    page = BeautifulSoup(f, "lxml")

title = page.select("title")
items = page.select("ul > li")
order = page.select(".order")

print("Title: " + title[0].get_text())

for item in items:
    print(item.get_text())

print("Order url: " + order[0]["href"])
```

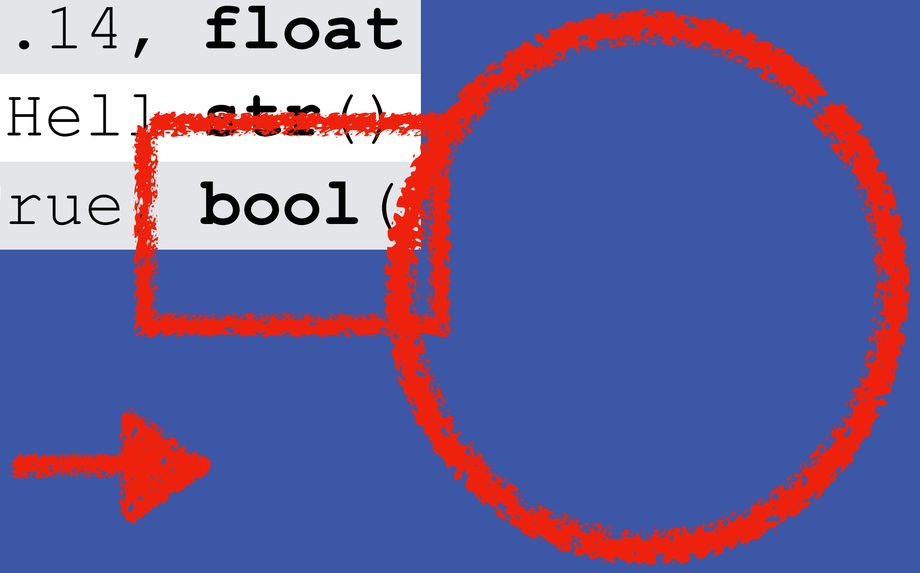
Restaurant scraping
Scrape a restaurant website and print details. You're free to use either a terminal program or a Jupyter Notebook.

* Find a restaurant website, preferably one that doesn't have a lot of dynamic content using Javascript. Try to scrape the site for the restaurant where you had dinner most recently.
* Save a page to a html file
* In a comment in your Notebook **include the URL** that you saved
* Load the HTML file using **BeautifulSoup**
* Get the **<title>** element and at least two other elements from the page.
* Try to get as other elements from the site using scraping.
* **Make sure to also upload the html file to Github!**

Tips
* Check if you have BeautifulSoup by running a Python **console** and executing **from bs4 import BeautifulSoup**
* If that doesn't work you need to do **pip install beautifulsoup4** from the **terminal** or the **Anaconda Prompt**
* Maybe you can use a function to simplify repeating tasks, like getting the first elements text after doing **.select()**

Extended exercise
* Try scraping the menu and save that to a **CSV** or **JSON** file using **Pandas**

Type	Examp	Conve
Integ	42,	**int**()
Float	3.14,	**float**
Strin	"Hell	**str**()
Boole	True	**bool**(



```
age = 20

if age < 20:
    print("option 1")
elif age <= 20 and age > 20:
    print("option 2")
else:
    print("option 3")
```

Compilation
Interpretation