Capgemini

DATA SCIENCE CONSULTING
Session 1

January 25th, 2020

Agenda

1. **Who are we?**
2. Course modalities
3. Analysis objectives & approach
4. Case presentation
5. Data collection
6. Html presentation & Selectors
7. Scraping with Scrapy
8. Summary of the session

Capgemini Invent, a leader in digital & data transformation

Strategic focus,
of our firm since 2012, now standing for 50% of our project portfolio

Leader on the market,
with award winning thought leadership, partnership with MIT, and world class recognition

World class footprint,
16 offices, 5.000 consultants, coverage of more than 80% companies of CAC 40 and DAX 30 at CxO level

Acknowledgements of business expertize

ALM

Full spectrum digital specialist, Best-of breed provider - 2016

Gartner

Leader position in the Magic Quadrant for Business Analytics Services 2017

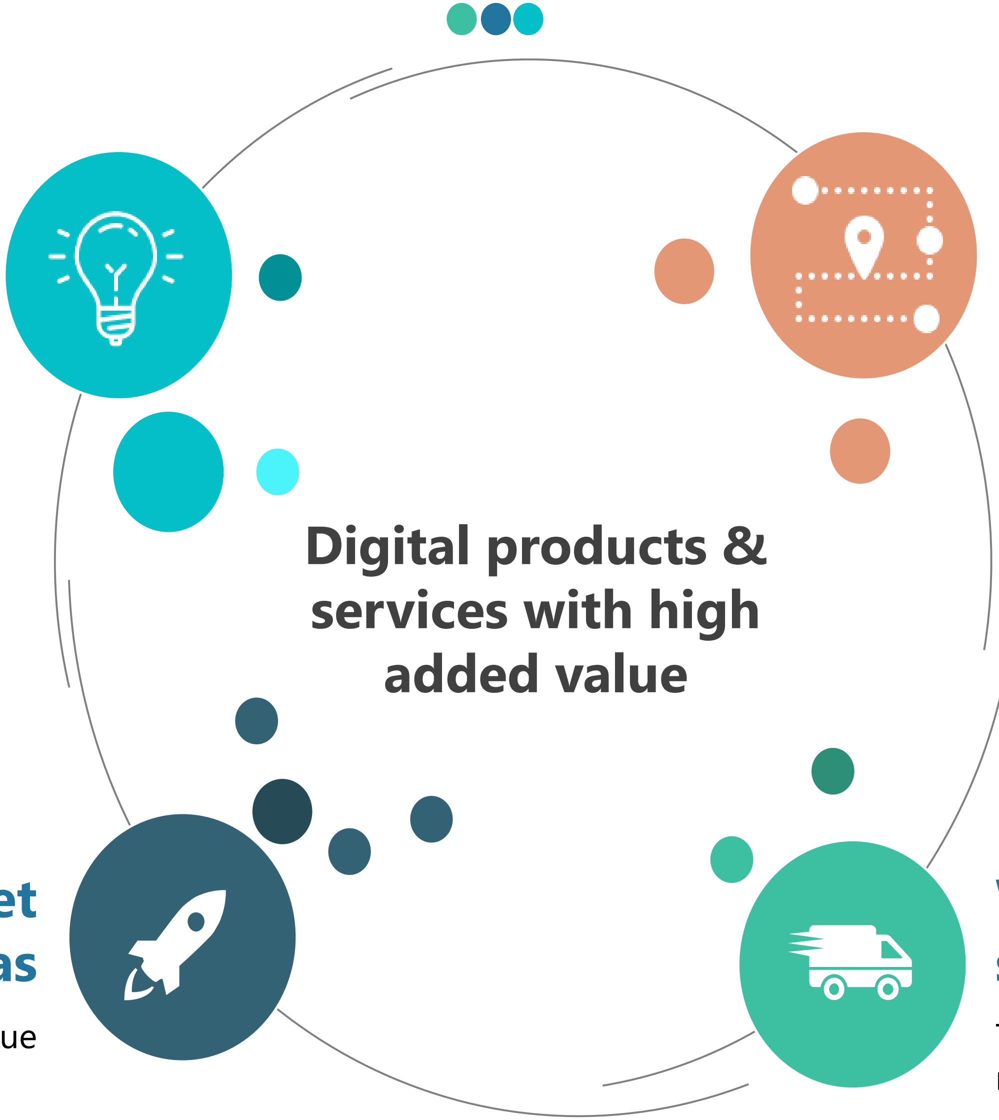
Capital

N°1 consultancy for digital, data and transformation - 2016

source forconsulting.com

Top 3 worldwide thought leadership consultancy - 2016

Insight Driven Enterprise, a hybrid agency orchestrating consulting, data science, technologies, creative skills from vision to delivery



We stimulate ideation & disruptive vision

By helping you finding the unconventional ideas to hack your business

We provide end-to-end commitment & delivery

By involving & orchestrating the different required capabilities

Digital products & services with high added value

We ensure go-to-market for your best ideas

From digital assets to business value

We innovate at startup speed for a fast delivery

Through proven agile and lean methodologies

The team with whom you will spend your Bootcamp sessions

Olivier AULIARD
Chief Data Scientist
olivier.auliard@capgemini.com

Johan ATTIA
Data Scientist
johan.attia@capgemini.com

Khemon BEH
Managing Data Scientist
khemon.beh@capgemini.com

Sim BOZKO
Senior Consultant
sim.bozko@capgemini.com

Thomas CLAVIER
Managing Data Scientist
thomas.clavier@capgemini.com

Maëva DERRIEN
Senior Consultant
maeva.derrien@capgemini.com

Mariam IKKEN
Consultant
mariam.ikken@capgemini.com

François LEMEILLE
Data Scientist
francois.lemeille@capgemini.com

Hamza MASSAOUDI
Data Scientist
hamza.massaoudi@capgemini.com

Ismail MEBSOUT
Data Scientist
ismail.mebsout@capgemini.com

Sami MHIRECH
Data Scientist
sami.mhirech@capgemini.com

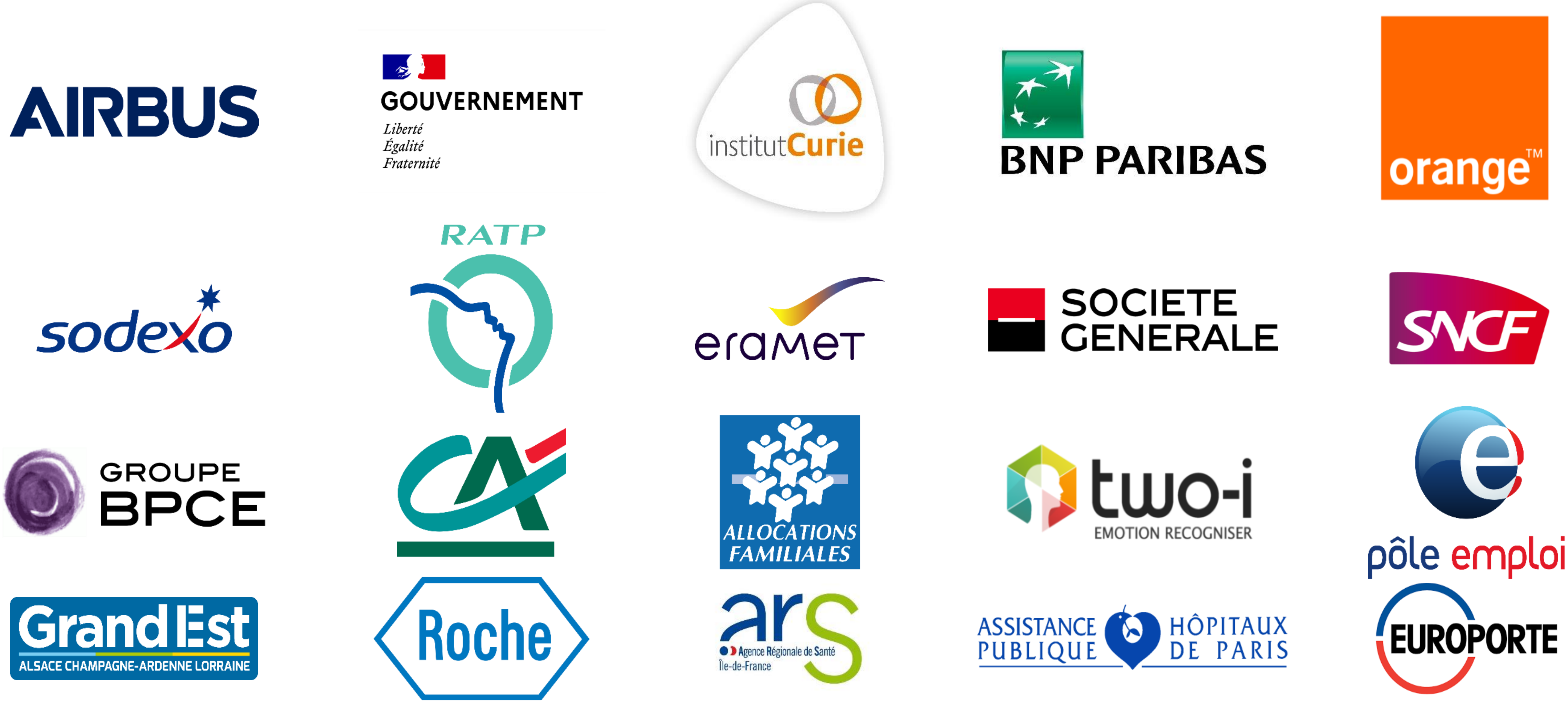
Augustin MONY
Data Scientist
augustin.mony@capgemini.com

Ziad NADER
Senior Data Scientist
ziad.nader@capgemini.com

Svetlana OLLIVIER
Managing Consultant
svetlana.ollivier@capgemini.com

Marie VACHELARD
Data Scientist
marie.vachelard@capgemini.com

Business expertise from projects with diverse clients



Agenda

1. Who are we?
2. **Course modalities**
3. Analysis objectives & approach
4. Case presentation
5. Data collection
6. Html presentation & Selectors
7. Scraping with Scrapy
8. Summary of the session

Objectives of the case study

Handle a business problematic associated to data

Increase both knowledge and skills on these topics

Learn how to determine & realize the required analysis

Handle a data project from the beginning to the end

Understand the strategic & transformation stakes

Qualify and quantify the associated stakes

Grasp the consulting aspects

Learn how to manage these kinds of projects

Organization of the course

Content

- Each session focused on a **specific topic**
- **Balance** between theoretical explanations, brainstorming and hands-on applications
- **Tasks** to be prepared between sessions and to be presented in the beginning of the next session

Format

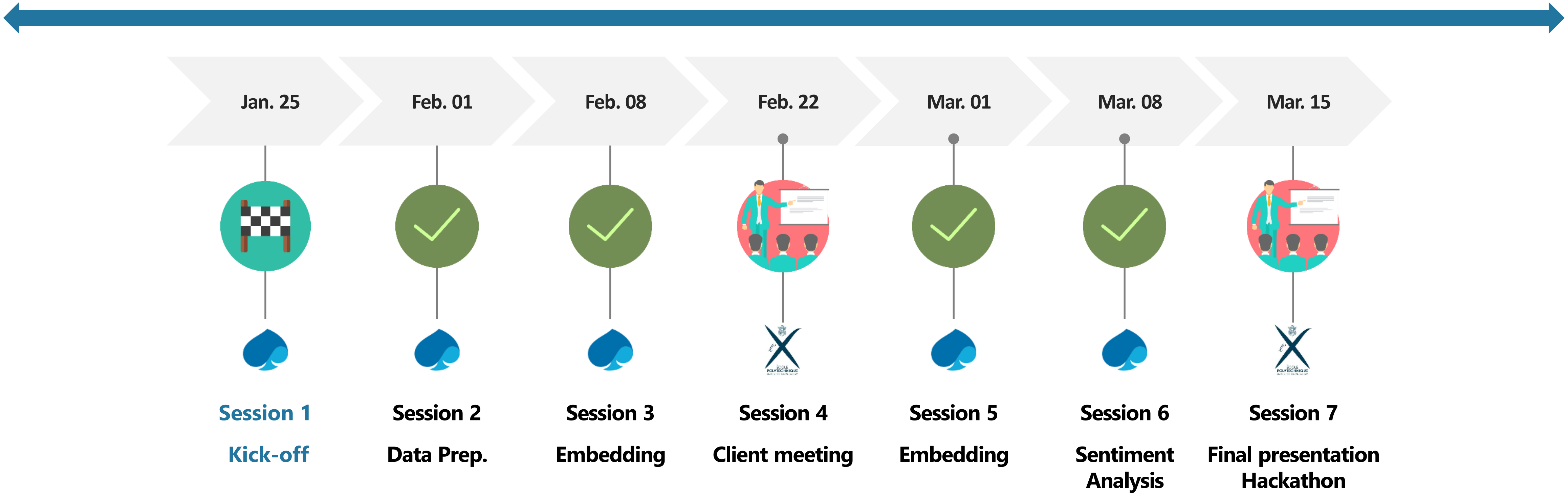
- **Weekly sessions on Monday afternoons** – from 2pm to 6pm
- **Online sessions only** in virtual classes on Microsoft Teams
- **Animation by Data Scientists & Consultants** from Insight Driven Enterprise (IDE)

Practical organization

- **Teamwork** **:** divided in groups of 4, each group with an assigned coach
- **Data Science tools** : Google Collab for all the sessions and Anaconda for the 1st session
- **The team is available via Slack & Teams** https://nlpinvent2021.slack.com in order to :
  - **Discuss** among yourselves
  - **Ask questions** to coaches
  - **Get access to all the questions & answers** – including from the other students & groups

Planning & key steps of the course

7 courses



Jan. 25 – **Session 1** – **Kick-off**

Feb. 01 – **Session 2** – **Data Prep.**

Feb. 08 – **Session 3** – **Embedding**

Feb. 22 – **Session 4** – **Client meeting**

Mar. 01 – **Session 5** – **Embedding**

Mar. 08 – **Session 6** – **Sentiment Analysis**

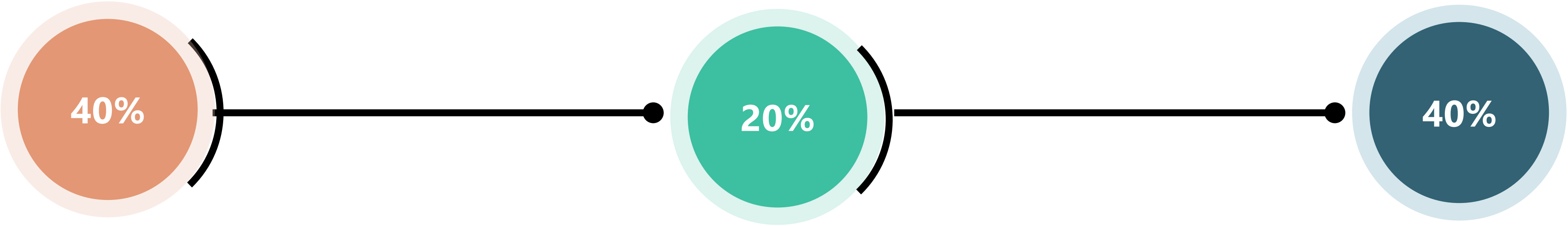
Mar. 15 – **Session 7** – **Final presentation Hackathon**

Legend :

✓ **Regular sessions:** Synthetic status update on the progress made and foreseen objectives to be prepared

Committees: Complete pres. of the progress made, results; difficulties and foreseen next steps to be prepared

Evaluation



40%

Homework

At each session you might have homework. We will ask you to send us your results and will evaluate them !

20%

Intermediate restitution

At the 4th session, we will simulate a client meeting with intermediary results. It will be a presentation where you need to summarize what you’ve seen/done until then.

40%

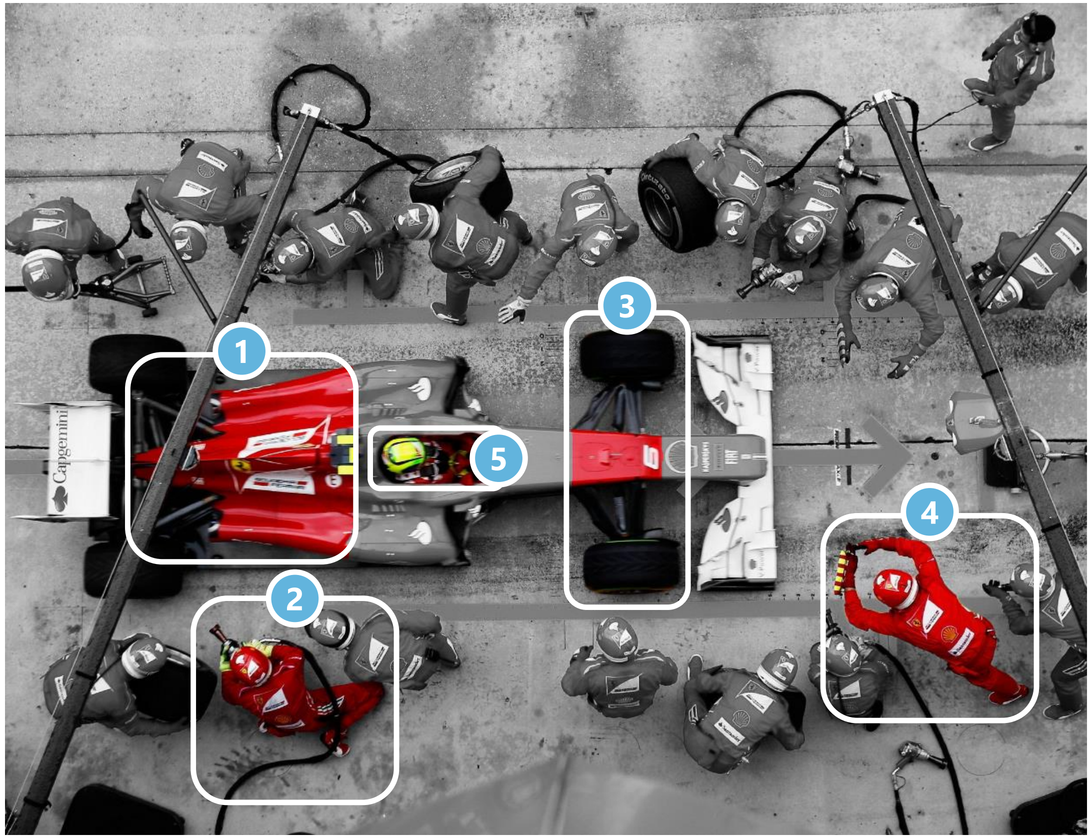
Final restitution

On the last day, you will make a final presentation that will enable you to present the business insights produced throughout from your data Project.

Agenda

1. Who are we?
2. Course modalities
3. **Analysis objectives & approach**
4. Case presentation
5. Data collection
6. Html presentation & Selectors
7. Scraping with Scrapy
8. Summary of the session

First of all … what is a data use case?



1 **Motor**
Robust technical platforms

2 **Fuel**
Comprehensive data sources

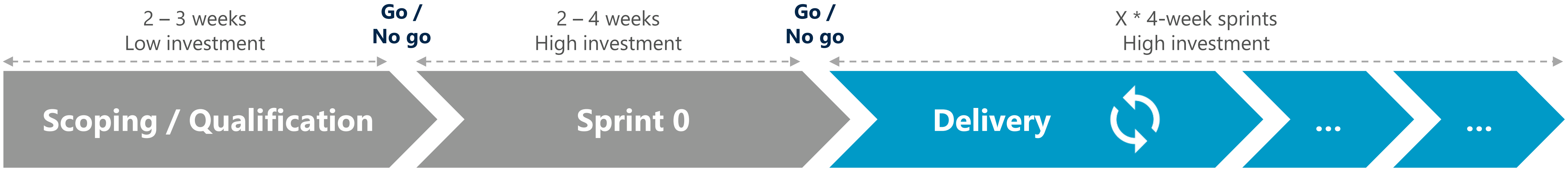
3 **Tires, wheel**
Digital solutions & touchpoints

4 **Technical team**
Advanced analytics capability

5 **Driver**
Identified end-users

Typical approach for use case delivery projects

The operating model is based on a progressive approach to ensure the correct product delivery



2 – 3 weeks
Low investment

Go / No go

2 – 4 weeks
High investment

Go / No go

X * 4-week sprints
High investment

Scoping / Qualification

Sprint 0

Delivery

…

…

Is the project **mature enough**?

Is **everything ready** to launch the project?

Delivery of industrialized data products

PoC / PoV

If necessary, launch of a PoC / PoV to prove the project feasibility / value

Zoom on Scoping / Qualification

1 Business needs

- Elaborate a description of needs and matching features to respond
- Present the breakdown of features into user stories
- Design the wireframe / prototype of the solution (if applicable)
- Process data and evaluate the algorithms & logic to be implemented

2 Data integration

- Confirm required & available data
- Identify data sources
- Define data to be ingested
- Build and initate data ingestion process

3 IT architecture

- Identify required elements for solution setup
- Adapt existing architecture to the target solution
- Define data ingestion architecture
- Ensure availability of necessary architecture models

4 Team

- Define competences to gather
- Ensure teams' availability (including Product Owner & Support)
- Define delivery model

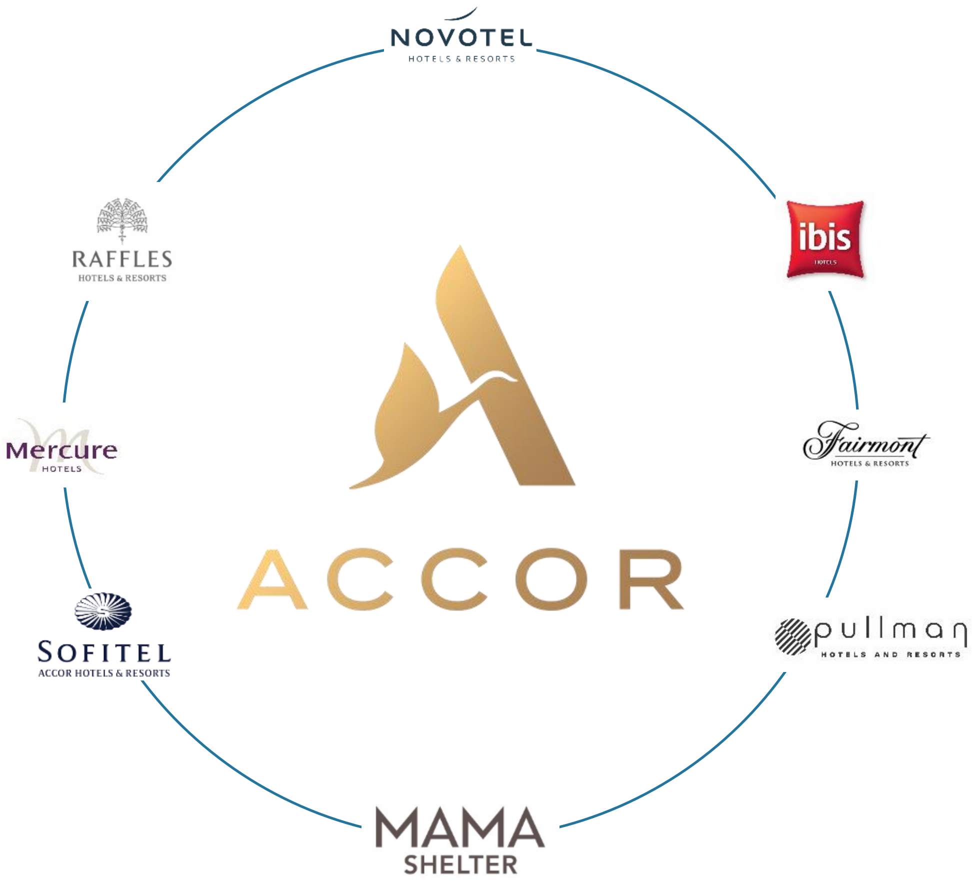
5 Planification & project organisation

- Set up steering and delivery model : injection planning, sprint 1 planning
- Evaluate and prioritize features (business value & complexity)

Agenda

1. Who are we?
2. Course modalities
3. Analysis objectives & approach
4. **Case presentation**
5. Data collection
6. Html presentation & Selectors
7. Scraping with Scrapy
8. Summary of the session

Presentation of the client



Multinational hospitality group founded in 1967 in France

Covers variety of segments – luxury, midscale and economy – and have subsidiaries in events organization and digital hospitality (catering, coworking…)

Operates in 100 countries – 4800 hotels and 280 000 employees worldwide

Rebranding strategy since 2011 on existing assets and new acquisitions

« Life styled for you » : complementarity between group brands for better customer experience

Focus of case study : Novotel Canary Wharf

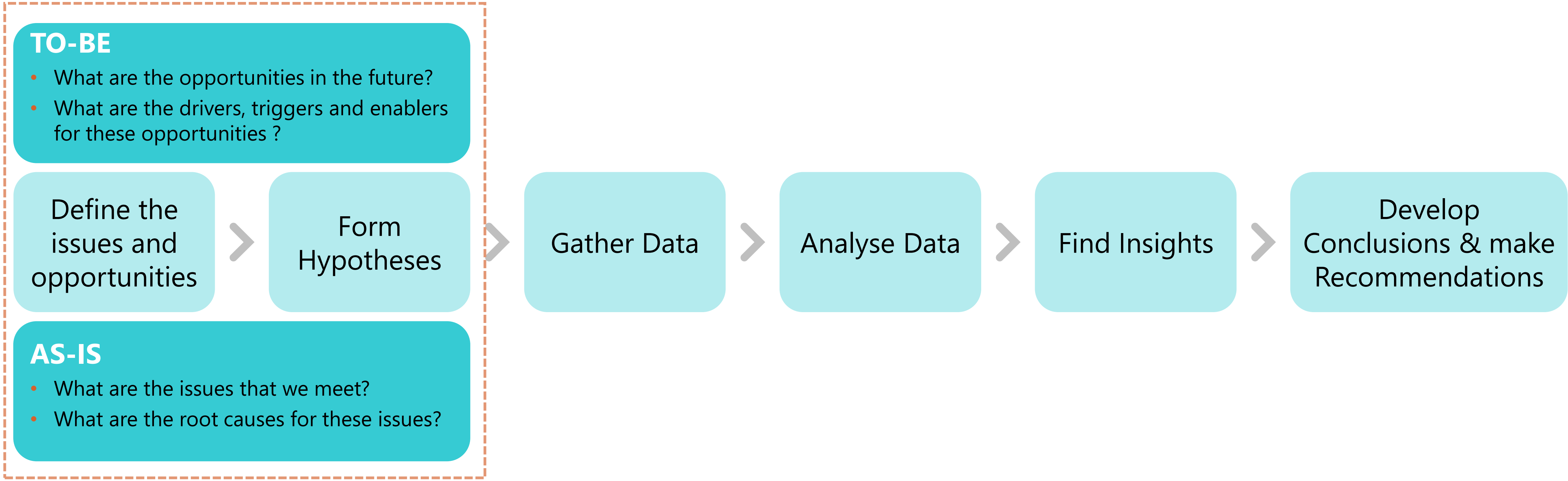
➢ Prepare and adapt

Main steps to deliver our case study

- Ensure business during lockdown and prepare for future opening to attract a larger scale of customers
- Provide suggestions for branding and menu content
- Use data to make the right choices for the rebranding of the space
- Estimate the costs, benefits and breakeven point

Homework for next session (01/02)

In group, perform an **As-is and To-be analysis** of Bokan 39 based on its components and characteristics.
Do not forget to adapt your analysis to the current Covid-19 **sanitary context**.



TO-BE

- What are the opportunities in the future?
- What are the drivers, triggers and enablers for these opportunities ?

Define the issues and opportunities › Form Hypotheses › Gather Data › Analyse Data › Find Insights › Develop Conclusions & make Recommendations

AS-IS

- What are the issues that we meet?
- What are the root causes for these issues?

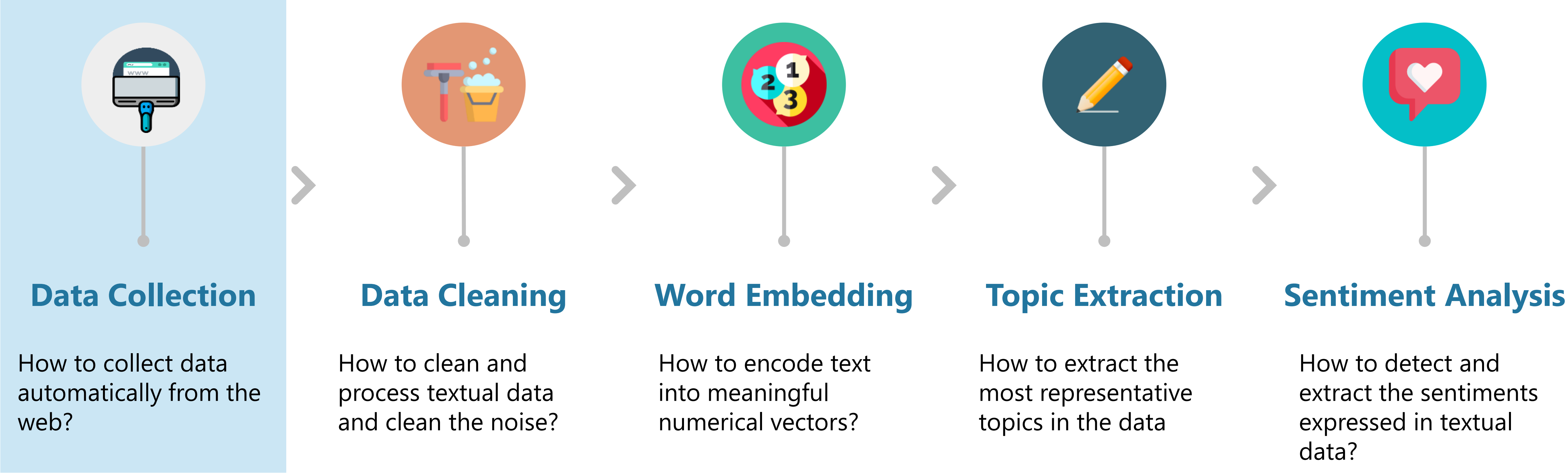
As-is & To-be analysis are a project management tool that **enable the definition of current problematics and future opportunities** in the context of a change in the business model or process

Agenda

1. Who are we?
2. Course modalities
3. Analysis objectives & approach
4. Case presentation
5. **Data collection**
6. Html presentation & Selectors
7. Scraping with Scrapy
8. Summary of the session

Data pipeline of a Nature Language Processing (NLP) project

Subject of today's session



Data Collection

How to collect data automatically from the web?

Data Cleaning

How to clean and process textual data and clean the noise?

Word Embedding

How to encode text into meaningful numerical vectors?

Topic Extraction

How to extract the most representative topics in the data

Sentiment Analysis

How to detect and extract the sentiments expressed in textual data?

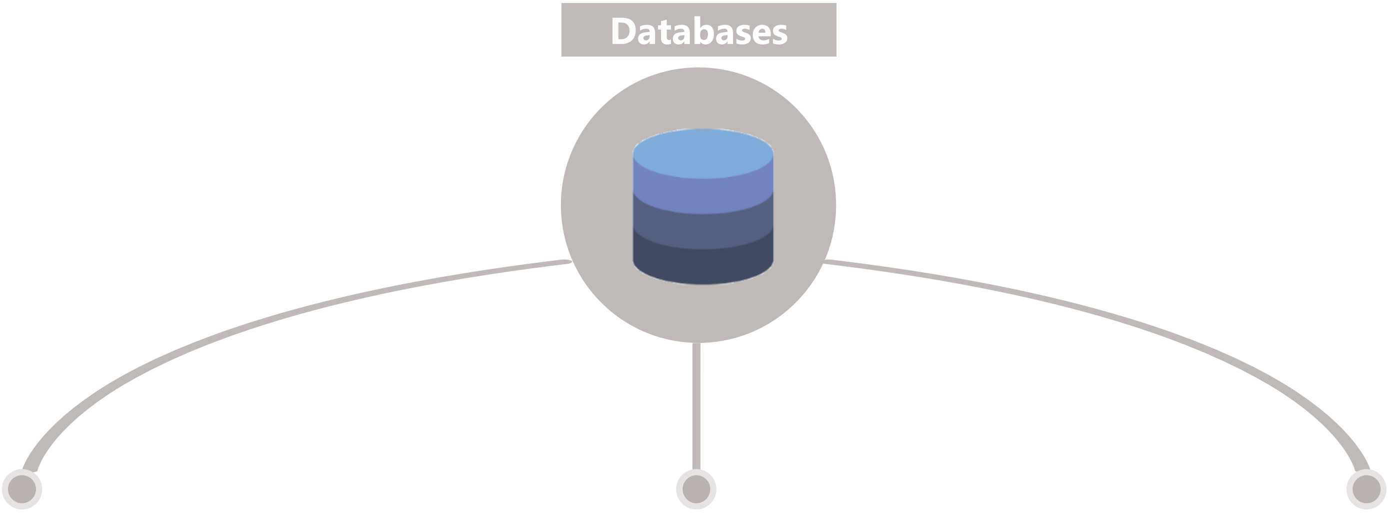
Data can be collected through several data channels

Databases

APIs

Web scraping

Collecting data from databases



Databases

Extracting data directly from databases

A user can retrieve data directly from certain type of databases by selecting and copying it through a user interface (*e.g. Excel files*)

Writing queries

Queries are written statements that describes and executes read and write operations on databases (*e.g. SQL queries*). In order to write such queries, the user needs to know how the database is structured

Using database connectors

Some data platforms provide connectors that facilitates extraction from databases stored into the platform (*e.g. Power BI provides connectors Azure DBs and Amazon DBs*)

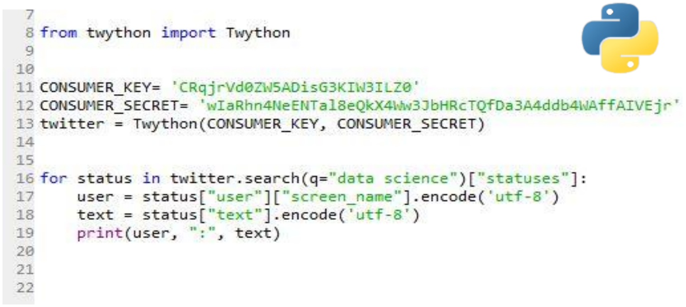
Application Programming Interfaces (API)

APIs enable interactions with applications

Several websites provide APIs for the purpose of data sharing

Few lines of code are needed to use them

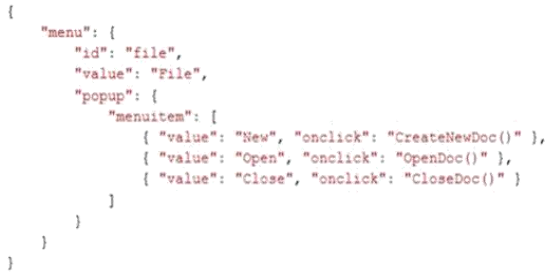
The main advantage of using APIs is that they require less programming than scraping and are in general well documented



```python

from twython import Twython


CONSUMER_KEY= 'CRqjrVd0ZW5ADisG3KIW3ILZ0'
CONSUMER_SECRET= 'wIaRhn4NeENTal8eQkX4Ww3JbHRcTQfDa3A4ddb4WAffAIVEjr'
twitter = Twython(CONSUMER_KEY, CONSUMER_SECRET)


for status in twitter.search(q="data science")["statuses"]:
    user = status["user"]["screen_name"].encode('utf-8')
    text = status["text"].encode('utf-8')
    print(user, ":", text)



```

APIs provides structured data output

The output of an API is in general a JSON file which can be easily turned into a database



```json
{
    "menu": {
        "id": "file",
        "value": "File",
        "popup": {
            "menuitem": [
                { "value": "New", "onclick": "CreateNewDoc()" },
                { "value": "Open", "onclick": "OpenDoc()" },
                { "value": "Close", "onclick": "CloseDoc()" }
            ]
        }
    }
}
```

Social networks have different policies concerning APIs

- No direct scraping unless authorized (fill in authorization form – 2 weeks for FB answer) : click here to see the terms
- APIs exist for app developers

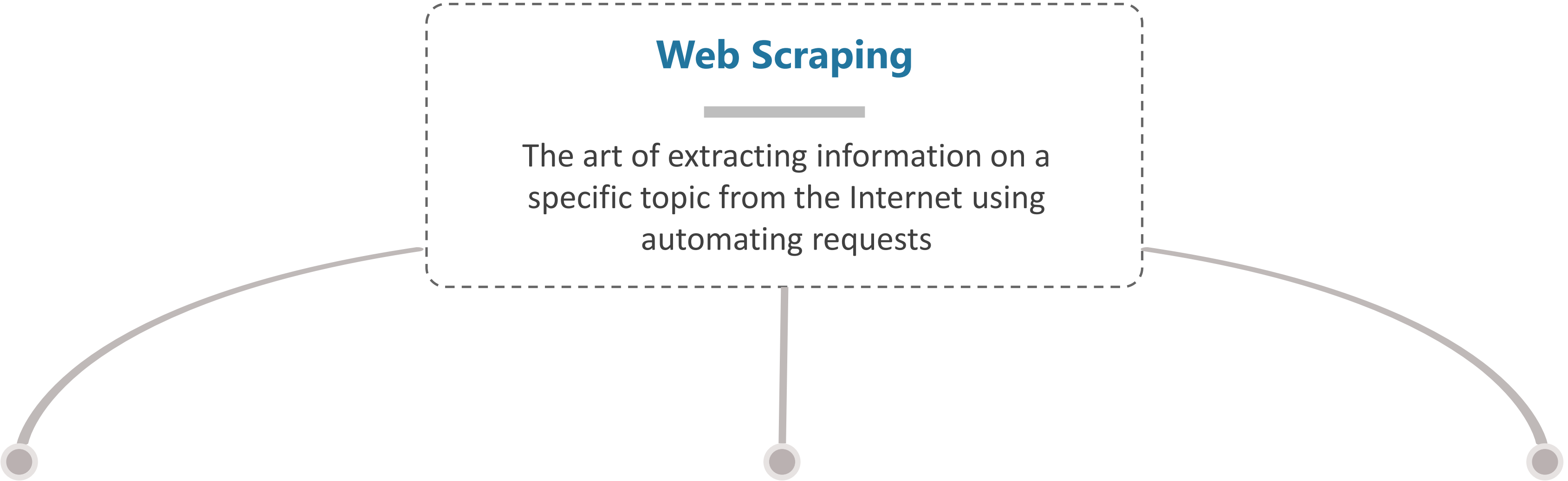
- Limited collection of the data (speed/volume is compared to what « human can reasonably produce »), it allows read & write operations on videos with a limited quota
- No personal data

- No direct scraping
- API with limited number of call by 15 min window

- Prohibition of scraping software
- APIs are for app development

- The free API is for “non-automated” apps, user authorization needed, Python/Ruby versions exists, 5000 calls per hour : it is being depreciated in favour of the new “Business version” (Instagram Graph API)
- Sensitive to user content/media (owned by users)

Web scraping enables information retrieval from the web



Web Scraping

The art of extracting information on a specific topic from the Internet using automating requests

Why should we use web scraping?

- Market Price Analysis (*e.g. real-time competitiveness*)
- Market Intelligence (*e.g. competitive benchmarks*)
- Sentiment Analysis (*e.g. social listening*)

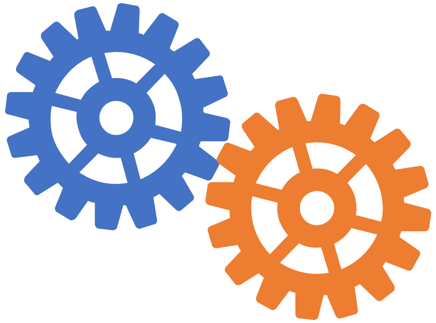
Which data are we looking for?

- Textual data available on websites (*e.g. articles, reviews, prices*)
- Metadata (*e.g. number of connections*)
- Social Media data (*e.g. tweets*)

What is the typical process?

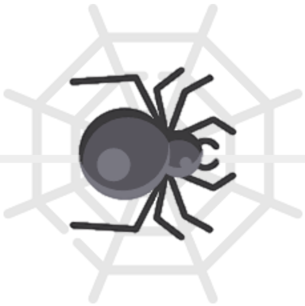
- Crawl the web looking for needed information
- Centralize collected data and make it structured

Web scraping consists in two main blocks of tasks



Parsing

Breaking down the scraped data into smaller bits to understand and structure it



Crawling

Going through specified website and related links, to get raw data (source code)

Main tools used in web scraping

BeautifulSoup

HTML parser in Python, which transforms HTML code into a tree object usable with Python to extract content like text, images' URLs…

Scrapy

Python package combining a crawler AND a parser.

Selenium

Tool allowing to automate web browsers, mainly used in scraping to get data which is written in JavaScript through AJAX requests.

RVest

HTML parser like BeautifulSoup, but for R. Great documentation online.

Scrapy

In this course we will focus on **Scrapy** exclusively

Agenda

1. Who are we?
2. Course modalities
3. Analysis objectives & approach
4. Case presentation
5. Data collection
6. **Html presentation & Selectors**
7. Scraping with Scrapy
8. Summary of the session

How does the web work (1/2)?

First, some internet culture…

What are the differences between the Internet and the Web ?

Internet

« **Inter**connected **net**work »

Global system of interconnected computer networks using the internet protocol suite to link devices worldwide

The Web

Information space where documents and other resources are identified by Uniform Resource Locator (URLs), interlinked by hypertext links, and accessible via the Internet

Web Browsers

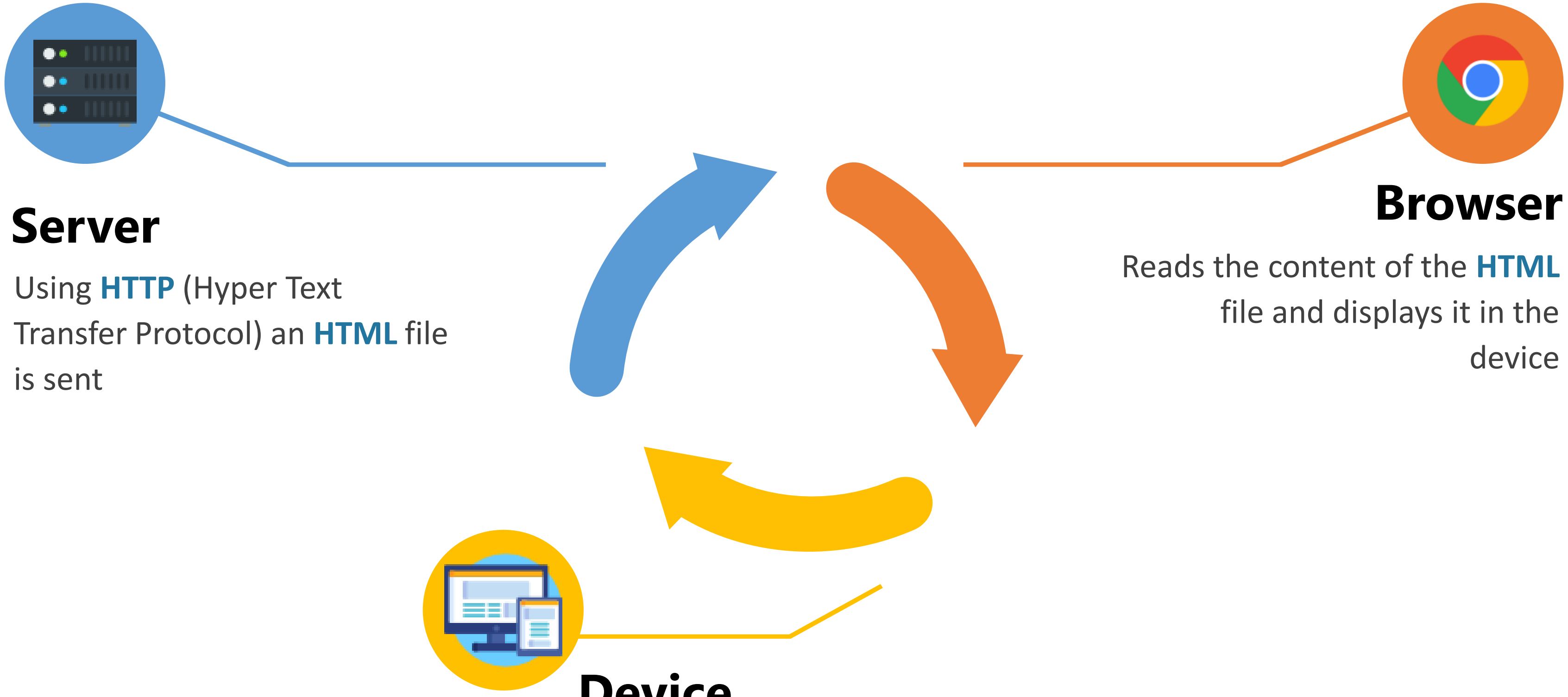
Software application used to « surf » the Web.

E.g. Google Chrome, Mozilla Firefox, Safari, Microsoft Edge…

The web was built using a markup language, **HTML** (HyperText Markup Language), invented in the early 90's. Understanding **HTML** is useful to learn how to scrape properly the web.

How does the web work (2/2)?

How the web works : global summary



Server

Using **HTTP** (Hyper Text Transfer Protocol) an **HTML** file is sent

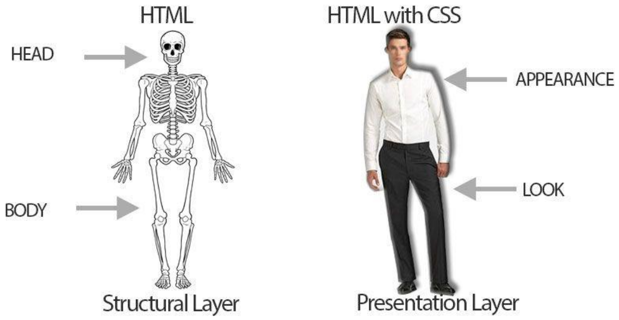
Browser

Reads the content of the **HTML** file and displays it in the device

Device

Requests web pages (originally **HTML** files) using their URLs which works as a « web address »

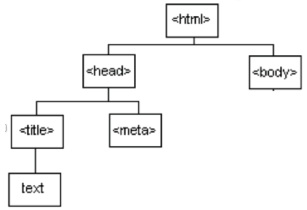
How is a web page structured ?



Structure

- A web page is structured by two main elements:
  - **HTML** : backbone of the web page, it contains text arranged into blocks, which have attributes
  - **CSS** : describes the style of the webpage

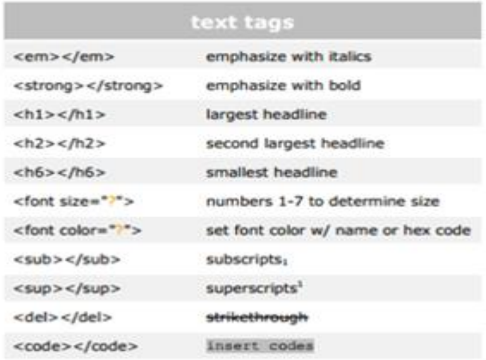
```
<!DOCTYPE html>
<html>
  <head>
      <meta charset="utf-8" />
      <title>Titre</title>
  </head>

  <body>

  </body>
</html>
```



Sequence

- It is a sequence of HTML tags which can be seen as a tree

`<tag attribute="valeur"> ... </tag>`



text tags	
<em></em>	emphasize with italics
<strong></strong>	emphasize with bold
<h1></h1>	largest headline
<h2></h2>	second largest headline
<h6></h6>	smallest headline
<font size="?">	numbers 1-7 to determine size
<font color="?">	set font color w/ name or hex code
	subscripts$_1$
	superscripts^1
<del></del>	~~strikethrough~~
<code></code>	`insert codes`

E.g. Text tags

Tags

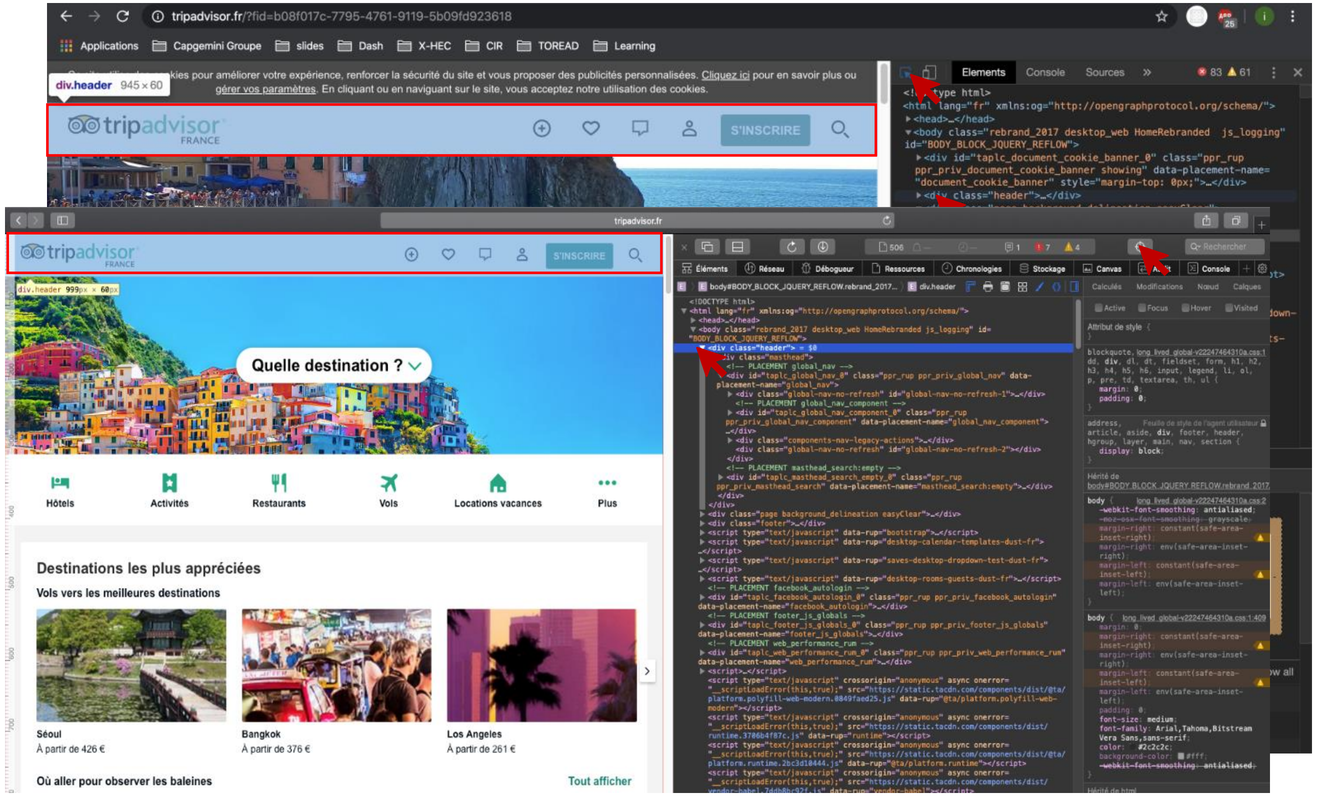
- Each tag has a specific format
- There are several attributes per tag: class, href, etc.

Scraping enables information retrieval from HTML source code

- Scraping packages enable the user to extract information from HTML pages and to structure it into databases
- Some browsers (Chrome, Firefox…) provide access to an interactive "Inspect mode", which enables the user to navigate within the source code of a web page :

CTRL + SHIFT + I

OPTION + CMD + I



…kies pour améliorer votre expérience, renforcer la sécurité du site et vous proposer des publicités personnalisées. Cliquez ici pour en savoir plus ou gérer vos paramètres. En cliquant ou en naviguant sur le site, vous acceptez notre utilisation des cookies.

```
<!doctype html>
<html lang="fr" xmlns:og="http://opengraphprotocol.org/schema/">
 ▶<head>…</head>
 ▼<body class="rebrand_2017 desktop_web HomeRebranded  js_logging"
 id="BODY_BLOCK_JQUERY_REFLOW">
   ▶<div id="taplc_document_cookie_banner_0" class="ppr_rup
   ppr_priv_document_cookie_banner showing" data-placement-name=
   "document_cookie_banner" style="margin-top: 0px;">…</div>
   ▶<div class="header">…</div>
```

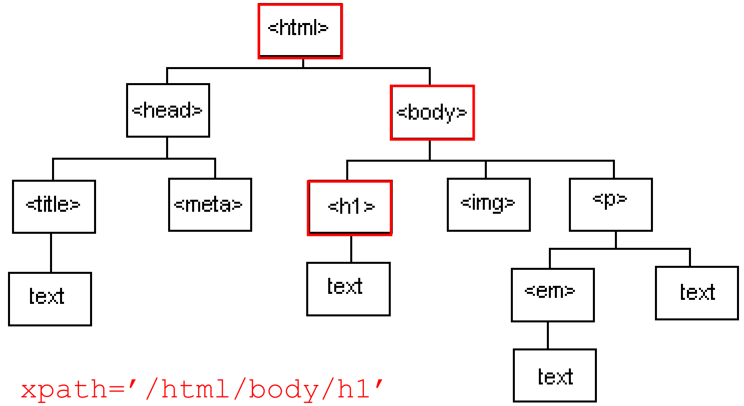
```
<!DOCTYPE html>
▼<html lang="fr" xmlns:og="http://opengraphprotocol.org/schema/">
  ▶<head>…</head>
  ▼<body class="rebrand_2017 desktop_web HomeRebranded js_logging" id=
  "BODY_BLOCK_JQUERY_REFLOW">
    ▼<div class="header"> = $0
       <div class="masthead">
         <!-- PLACEMENT global_nav -->
        ▼<div id="taplc_global_nav_0" class="ppr_rup ppr_priv_global_nav" data-
        placement-name="global_nav">
          ▶<div class="global-nav-no-refresh" id="global-nav-no-refresh-1">…</div>
           <!-- PLACEMENT global_nav_component -->
          ▶<div id="taplc_global_nav_component_0" class="ppr_rup
           ppr_priv_global_nav_component" data-placement-name="global_nav_component">
           …</div>
          ▶<div class="components-nav-legacy-actions">…</div>
           <div class="global-nav-no-refresh" id="global-nav-no-refresh-2"></div>
         </div>
         <!-- PLACEMENT masthead_search:empty -->
        ▶<div id="taplc_masthead_search_empty_0" class="ppr_rup
        ppr_priv_masthead_search" data-placement-name="masthead_search:empty">…</div>
        </div>
      </div>
    ▶<div class="page background_delineation easyClear">…</div>
    ▶<div class="footer">…</div>
    ▶<script type="text/javascript" data-rup="bootstrap">…</script>
    ▶<script type="text/javascript" data-rup="desktop-calendar-templates-dust-fr">
    …</script>
    ▶<script type="text/javascript" data-rup="saves-desktop-dropdown-test-dust-fr">
    …</script>
    ▶<script type="text/javascript" data-rup="desktop-rooms-guests-dust-fr">…</script>
      <!-- PLACEMENT facebook_autologin -->
    ▶<div id="taplc_facebook_autologin_0" class="ppr_rup ppr_priv_facebook_autologin"
    data-placement-name="facebook_autologin">…</div>
      <!-- PLACEMENT footer_js_globals -->
    ▶<div id="taplc_footer_js_globals_0" class="ppr_rup ppr_priv_footer_js_globals"
    data-placement-name="footer_js_globals">…</div>
      <!-- PLACEMENT web_performance_rum -->
    ▶<div id="taplc_web_performance_rum_0" class="ppr_rup ppr_priv_web_performance_rum"
    data-placement-name="web_performance_rum">…</div>
    ▶<script>…</script>
      <script type="text/javascript" crossorigin="anonymous" async onerror=
      "__scriptLoadError(this,true);" src="https://static.tacdn.com/components/dist/@ta/
      platform.polyfill-web-modern.0849faed25.js" data-rup="@ta/platform.polyfill-web-
      modern"></script>
      <script type="text/javascript" crossorigin="anonymous" async onerror=
      "__scriptLoadError(this,true);" src="https://static.tacdn.com/components/dist/
      runtime.3706b4f87c.js" data-rup="runtime"></script>
      <script type="text/javascript" crossorigin="anonymous" async onerror=
      "__scriptLoadError(this,true);" src="https://static.tacdn.com/components/dist/@ta/
      platform.runtime.2bc3d10444.js" data-rup="@ta/platform.runtime"></script>
      <script type="text/javascript" crossorigin="anonymous" async onerror=
      "__scriptLoadError(this,true);" src="https://static.tacdn.com/components/dist/
      vendor-babel.7ddb8bc92f.js" data-rup="vendor-babel"></script>
```

- The user chooses the information he wants to extract from a web page by exploring its source code and by adding to his scraping code the tags he wants to extract

XPath notation to navigate HTML

What is XPath ?

- Xpath is a string which selects nodes in an HTML tree
- It can also be seen as the linear representation of the requested element



xpath='/html/body/h1'

- XPath allows you to select:
  - The content of a markup
  - The content of its attributes (hypertext links for example)

Requested Element	Corresponding Xpath
Second division of the body	'/html/body/div[2]' (index start at 1)
All tables	'//table' #all tables
All tables descendants of the 2nd division	'/html/body/div[2]//table'
All paragraphs directly bellow the body	'//p'
Conditional division	'//div[@id="uid"]'
Wildcard	'/html/body/*'
All elements with a condition on the class	'//*[contains(@class, "class-1")]'
Selection of the attribute href cond. paragraph	'//p[@id="p2"]/a/@href'

From XPath to CSS Locators

What is a CSS Locator ?

- If XPath allows the selection of a node using the HTML tree, the CSS locators on the other hand access the node using CSS attributes
- The following tables displays on each row XPath and CSS locators that enable the selection of similar nodes :

XPath	CSS Locator
'/html/body/div'	'html>body>div'
'//div/span//p'	' div > span p'
'//div/p[2]'	' div > p:nth-of-type(2)'
'/html/body//div/p[2]'	'html > body div >p:nth-of-type(2)'
'//div[@id="uid"]'	' div#uid'
'//p[@class="class-1"]'	' p.class-1'
'//div[@id="uid"]/a/@href'	'div#uid >a::attr(href)'
'//p[@id="p-example"]/text()'	'p#p-example::text'
//p[@id="p-example"]//text()'	'p#p-example ::text'

Introduction to the Scrapy Selector

What is a selector ?

Hello World!

Enjoy Scraping!

```
text_html='''

<html>
    <head>
          <meta charset="utf-8" />
          <title>Titre</title>
    </head>

    <body>

         <div>
             <p>Hello World!</p>
         </div>

         <div>
             <p>Enjoy Scraping!</p>
         </div>

    </body>
</html>
'''
```

- A selector is a Scrapy object which is used to select portions of the HTML source code using XPath or CSS Locators
- Base on the HTML structure on the left, we can extract all the paragraphs:

  From scrapy import Selector

  Sel=Selector(text=text_html)

  Sel.xpath('//p').extract()

  ['<p>Hello world!</p>', '<p>Enjoy Scraping!</p>']

  *Sel.xpath('//p**/text()**').extract()*

  ['Hello world!', 'Enjoy Scraping!']

  *Sel.xpath('//p**/text()**')**.extract_first()***

  'Hello world!'

- Selectors allow chaining:

  Sel.xpath('/html/body/div[2]') == Sel.xpath('/html').xpath('./body/div[2]')

XPath	CSS Locator
Sel.xpath('div/p')	Sel.css('div > p')

Hands-on 1

40’

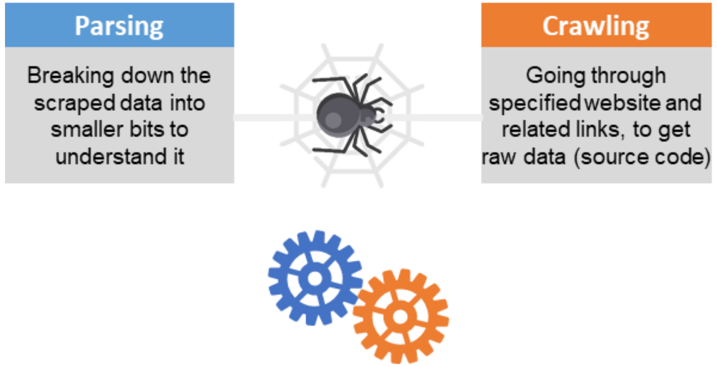
Use the notebook 1 to discover selectors and get some information from the web

If you have any question about Python set-up, feel free to contact us about that !

Agenda

1. Who are we?
2. Course modalities
3. Analysis objectives & approach
4. Case presentation
5. Data collection
6. Html presentation & Selectors
7. **Scraping with Scrapy**
8. Summary of the session

Scrapy uses a Spider Class to parse and crawl the web

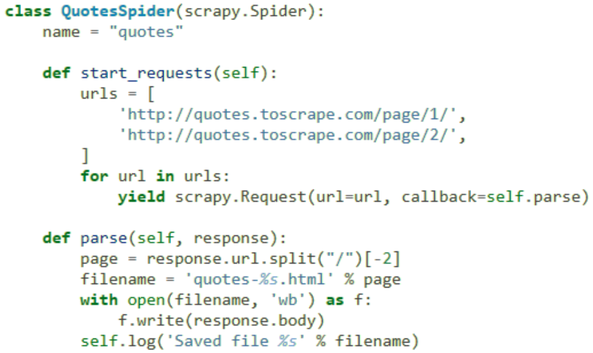


Parsing

Breaking down the scraped data into smaller bits to understand it

Crawling

Going through specified website and related links, to get raw data (source code)



```
class QuotesSpider(scrapy.Spider):
    name = "quotes"

    def start_requests(self):
        urls = [
            'http://quotes.toscrape.com/page/1/',
            'http://quotes.toscrape.com/page/2/',
        ]
        for url in urls:
            yield scrapy.Request(url=url, callback=self.parse)

    def parse(self, response):
        page = response.url.split("/")[-2]
        filename = 'quotes-%s.html' % page
        with open(filename, 'wb') as f:
            f.write(response.body)
        self.log('Saved file %s' % filename)
```

Scrapy

- Scrapy is a powerful Python scraping package to scrape which combines a crawler and a parser, so it's pretty complete for a scraping project
- Scrapy works with spiders, which are classes that the user defines and that Scrapy uses to crawl the web through multiple pages automatically according to the procedures the user programmed

Spider

- The spider is a python class and must contain :
  - a start_requests method which defines which web pages are to be scraped
  - a parse method (or methods) to define what we want to do with the scraped information
- You can write a spider inside a .py script, or you can use Jupyter's notebook.

Check the documentation for more info : https://docs.scrapy.org/

Scrapy Spiders are Python class objects

- Before getting into spiders we first need to define the structure of a Python class which is the basic structure of the spider itself
- A python class enables to :
  - Creates new objects
  - Defines the internal structure of an object
  - Defines the methods of an object

```
class class_name:
```

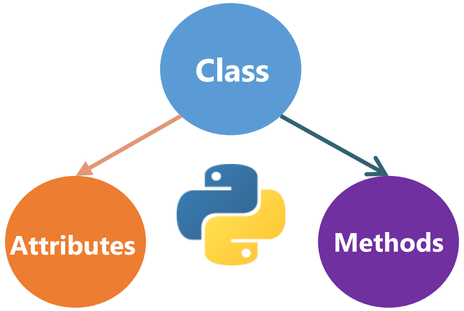
Attributes

```
def __init__(self, var1, var2):
    self.attr1=var1
    self.attr2=var2
```

Methods

```
def method1(self, x):
    …
    return …


def method2(self, y):
    …
    return …
```

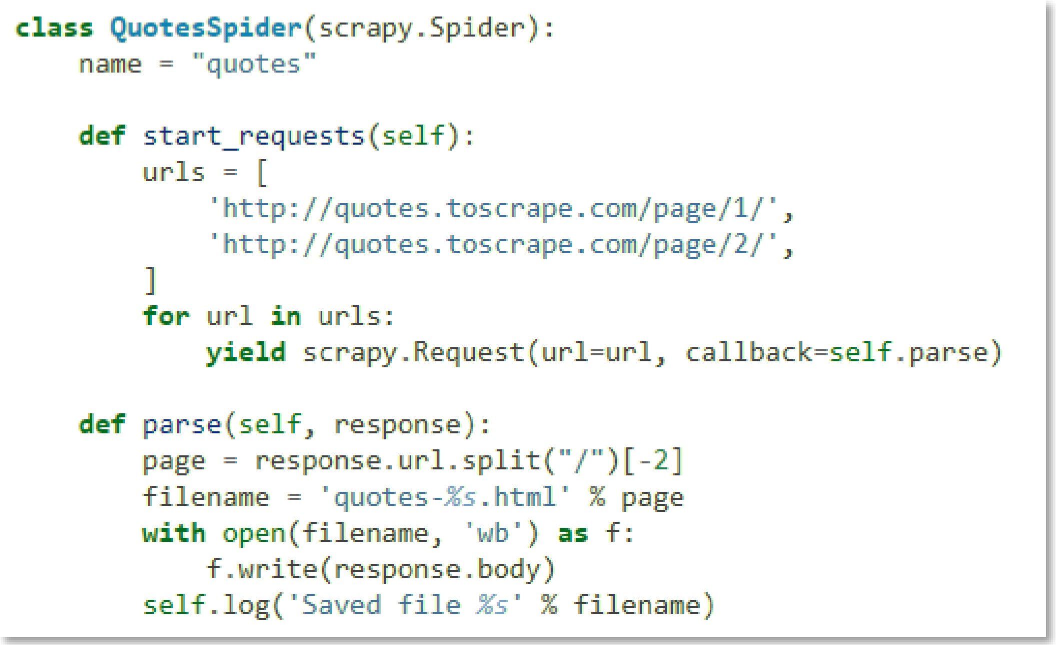
```
object = class_name(var1, var2)
print(object.attr1)  # returns var1
print(object.method1(x))
```



General form of a spider

```
Class
SpiderClassName(scrapy.Spider):
    name = "spider_name"
    # the code for your spider
    # what website?
    # which techniques
```

How to define a Spider as a class ?



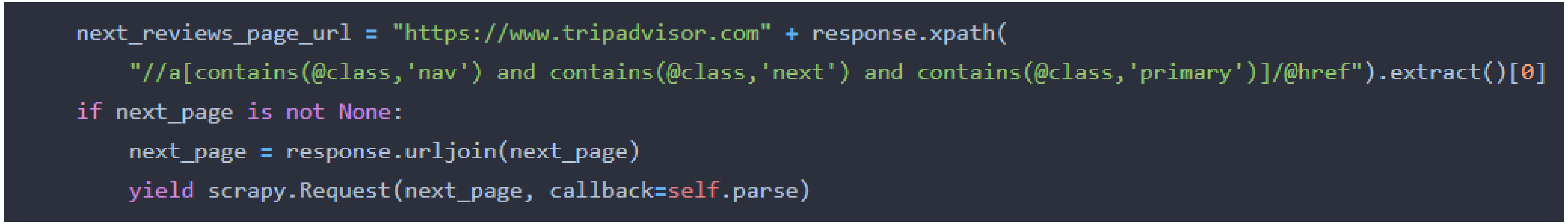
```
class QuotesSpider(scrapy.Spider):
    name = "quotes"

    def start_requests(self):
        urls = [
            'http://quotes.toscrape.com/page/1/',
            'http://quotes.toscrape.com/page/2/',
        ]
        for url in urls:
            yield scrapy.Request(url=url, callback=self.parse)

    def parse(self, response):
        page = response.url.split("/")[-2]
        filename = 'quotes-%s.html' % page
        with open(filename, 'wb') as f:
            f.write(response.body)
        self.log('Saved file %s' % filename)
```

- We define our spider with the subclass scrapy.Spider, along with some attributes and methods:
  - name: identifies the Spider, must be unique within a project
  - start_requests(): must return an iterable of URL Requests which the Spider will begin to crawl from
  - parse(): method that will be called to handle the response downloaded for each of the requests made

- The parse() method usually parses the response, extracting the scraped data as dictionaries, finding new URLs to scrape and creating new requests to scrape them

Scraping with Scrapy: Crawling

- Now that you know how to extract data from one page, we need to see how to follow links from one page to another, so that you'll be able to scrape a full website on your own !
- Crawling using Scrapy takes 3 steps:
  - Identify the link the user wants to follow with Selectors in the Scrapy Shell (or using tools like SelectorGadget…)
  - Extract this link using the .extract() method
  - Modify the spider to follow the extracted link by adding it to your parse() method:



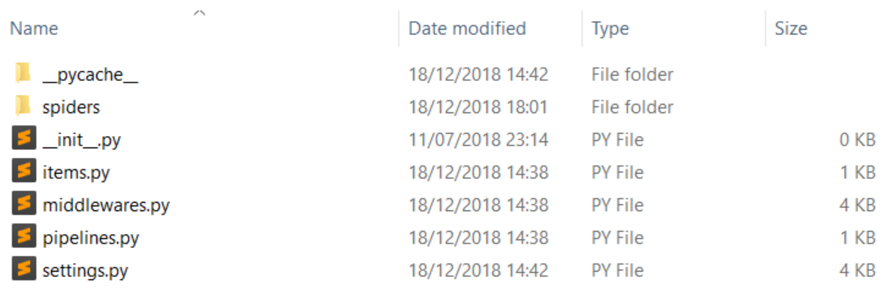
```
next_reviews_page_url = "https://www.tripadvisor.com" + response.xpath(
    "//a[contains(@class,'nav') and contains(@class,'next') and contains(@class,'primary')]/@href").extract()[0]
if next_page is not None:
    next_page = response.urljoin(next_page)
    yield scrapy.Request(next_page, callback=self.parse)
```

Scraping with Scrapy: creating a new project

- We can create a new project using the following command line:

```
scrapy startproject project_name
```

- This will create a directory with different files and subfolders:
  - *__pycache__* : contains the python cache
  - *spiders* : contains the spiders used for scraping
  - *__init__.py* : project initialization file
  - *items.py* : project items definition file
  - *middlewares.py* : project middlewares file
  - *pipelines.py* : project pipelines file
  - *settings.py* : project settings file



Name	Date modified	Type	Size
__pycache__	18/12/2018 14:42	File folder	
spiders	18/12/2018 18:01	File folder	
__init__.py	11/07/2018 23:14	PY File	0 KB
items.py	18/12/2018 14:38	PY File	1 KB
middlewares.py	18/12/2018 14:38	PY File	4 KB
pipelines.py	18/12/2018 14:38	PY File	1 KB
settings.py	18/12/2018 14:42	PY File	4 KB

- Having already defined a spider in a .py file stored in the folder « spiders », we can put it to work using the following command line:

```
scrapy crawl spider_name
```

Scraping with Scrapy: storing data scraped in JSON files

- The easiest way to store your scraped data is by using Feed exports, with the following command:

```
scrapy crawl spider_name -o file.json
```

- Saving your data using the JSON format will only **work once**, since Scrapy will append to a given file instead of overwriting its contents
- If you run the command twice without deleting the file before the second time, you'll end up with a broken JSON file

- You can also use JSON Lines format (file.jl) instead, which doesn't have the same problem of JSON if you run twice

Hands-on 2

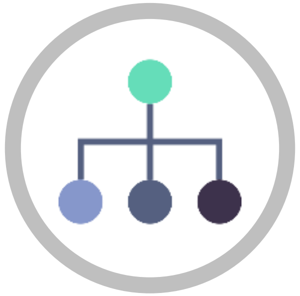
⏱ 30’

Try to scrap TripAdvisor – You can get inspiration from *x_actu_spider* !

Agenda

1. Who are we?
2. Course modalities
3. Analysis objectives & approach
4. Case presentation
5. Data collection
6. Html presentation & Selectors
7. Scraping with Scrapy
8. **Summary of the session**

Summary of the session – To remember



Approach & organization of a data science consulting project

Typical approach of this type of project

- Data science workstream: Data scraping, cleansing & feature engineering, running of the different analyses, restitutions (visualizations etc.)
- Business workstream: Diagnosis of the current situation & transformation stakes (as-is/to-be analysis) with a quantification of the impacts

Organization & governance :

- Several dedicated meetings all along the project (e.g. weekly status updates, steering committees) to track progress and escalate potential issues

Scraping as a tool to retrieve information from the web

Scraping as a data collection tool:

- Data can be collected through several channels: databases, APIs, web scraping

Zoom on web scraping:

- Scraping tools (Python: Scrapy, BeautifulSoup, Selenium ; R: Rvest)
- Key steps and tools used to perform them :
    - Parsing : Xpath and CSS locators
    - Crawling : Spider classes
    - Storing : JSON file

Work for next week

Instructions

To practice what we learnt today, for next week, you'll have to :

- Prepare an As-is and To-be analysis of Bokan 39
- Create a spider which gets reviews and ratings from multiple pages of reviews for a given restaurant
- As a reminder, you must be able to link a review to its restaurant
- It might be interesting to get other information available on the webpages
- Bonus (+1 pt on final mark) : create a spider which crawls multiple pages of restaurants to get multiple pages of reviews. At least 500 full reviews per restaurant for 100 restaurants.

We expect you to send your code and scraped file by **Friday 29th evening** to *hamza.massaoudi@capgemini.com* and *sami.mhirech@capgemini.com*

If you have any questions, feel free to contact us through the slack channel or by email.

Course evaluation

Did you like that first course ? It’s time to share your feedbacks !

Capgemini

Thank you for your attention

See you next week online !

GOODBYE !

January 25th, 2021