Data Extraction from the Web

aka webscraping

doc. Ing. Radek Burget, Ph.D.
burgetr@fit.vut.cz

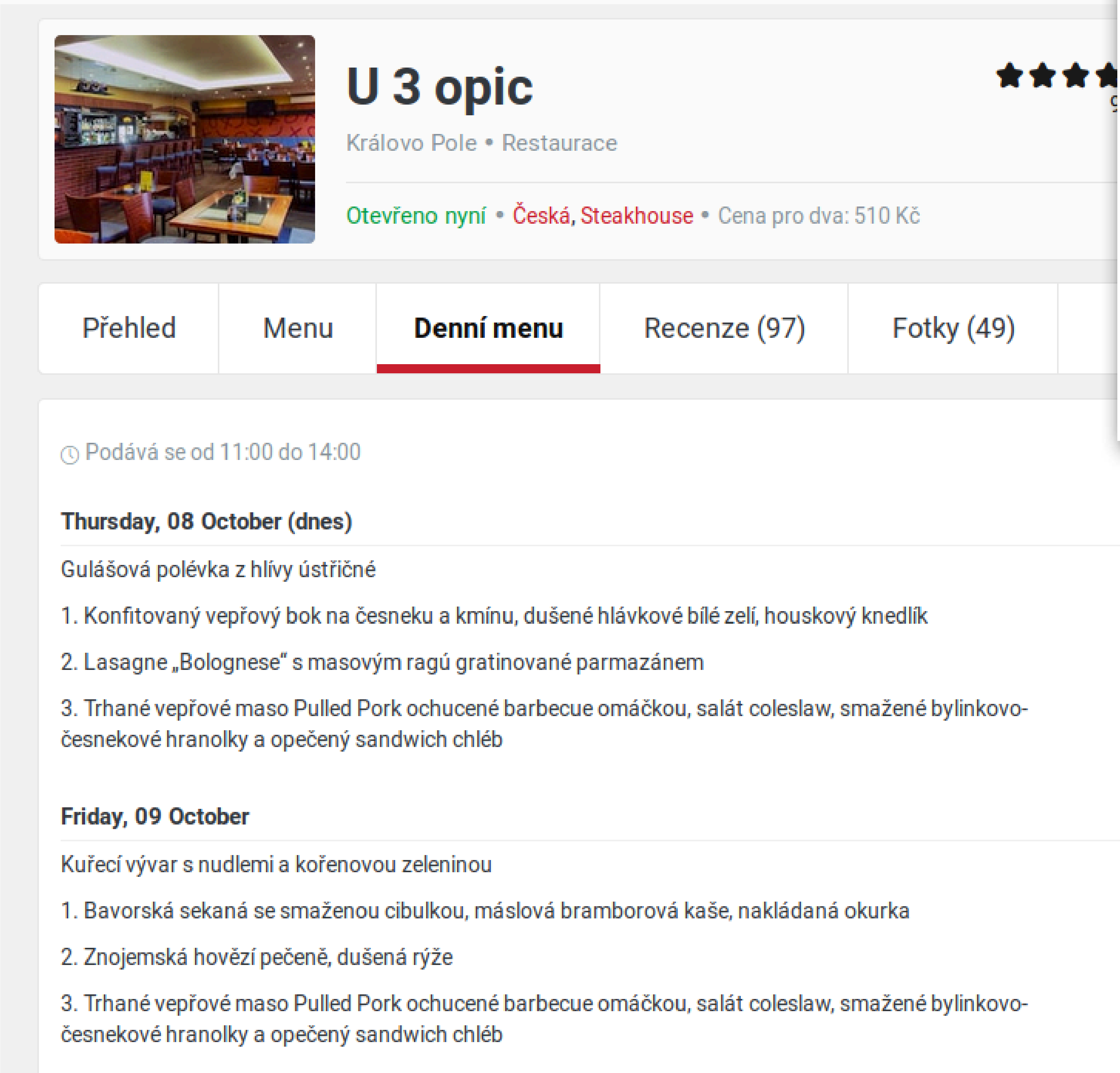
Motivation

- **There's a lot of data on the web** burried in web pages
- We need to further process this data in computer applications
  - Linking to your own data
  - Aggregation – combining results from different sources
  - Analysis – statistics, knowledge discovery
  - … and many more
- We need **structured data**
  - That can be stored in relational database tables
  - or at least XML, JSON, etc. *with a fixed structure*

Data on the Web

- Web pages are not strongly structured
- Mostly in HTML
  - *Visual presentation* is the primary goal
  - The code is secondary, *only implements the primary goal*
  - Not intended for further processing (only for browsing)

The presentation an



U 3 opic

Královo Pole • Restaurace

Otevřeno nyní • Česká, Steakhouse • Cena pro dva: 510 Kč

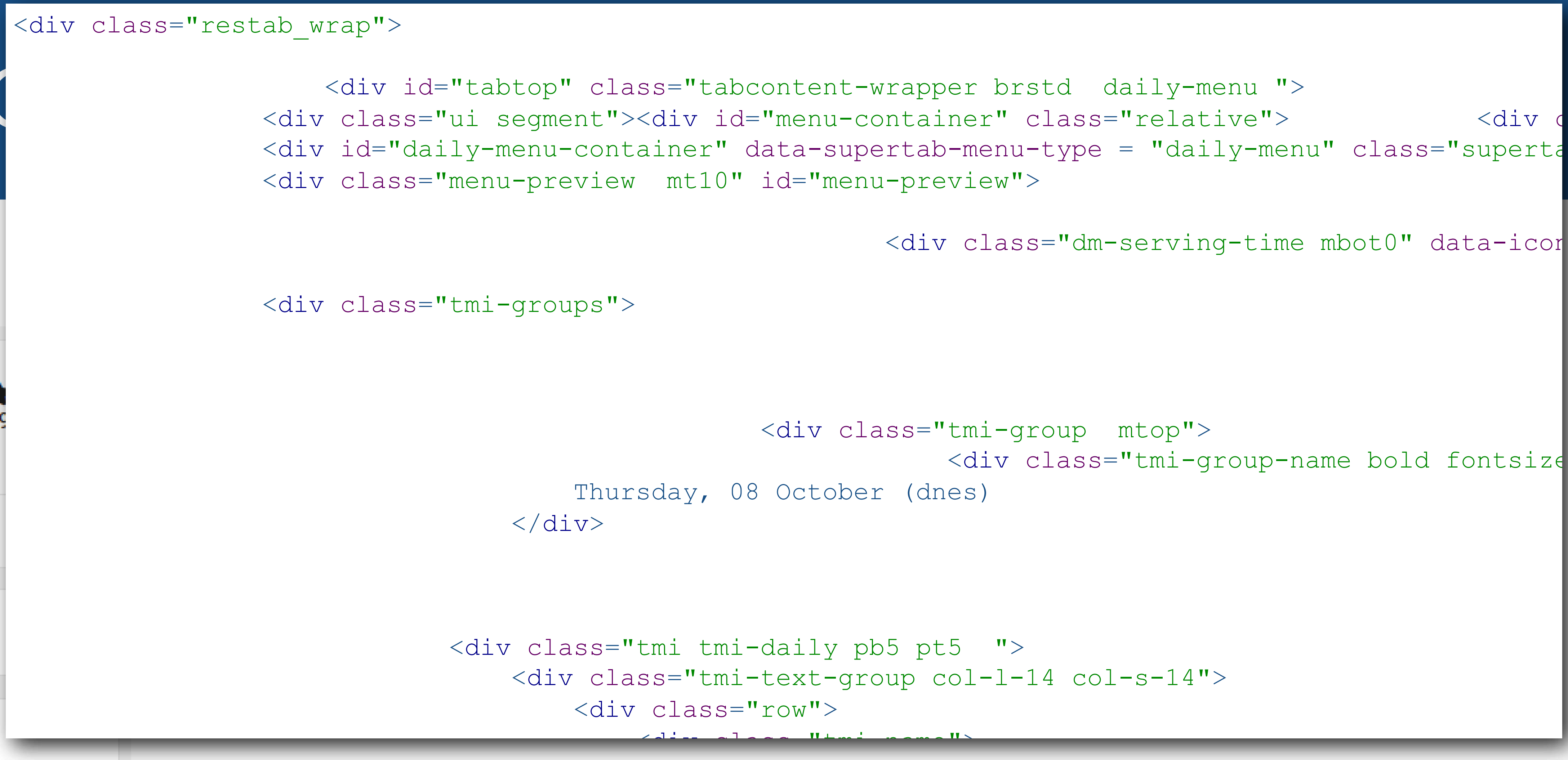
Podává se od 11:00 do 14:00

Thursday, 08 October (dnes)

Gulášová polévka z hlívy ústřičné	
1. Konfitovaný vepřový bok na česneku a kmínu, dušené hlávkové bílé zelí, houskový knedlík	109 Kč
2. Lasagne „Bolognese“ s masovým ragú gratinované parmazánem	119 Kč
3. Trhané vepřové maso Pulled Pork ochucené barbecue omáčkou, salát coleslaw, smažené bylinkovo-česnekové hranolky a opečený sandwich chléb	139 Kč

Friday, 09 October

Kuřecí vývar s nudlemi a kořenovou zeleninou	
1. Bavorská sekaná se smaženou cibulkou, máslová bramborová kaše, nakládaná okurka	109 Kč
2. Znojemská hovězí pečeně, dušená rýže	119 Kč
3. Trhané vepřové maso Pulled Pork ochucené barbecue omáčkou, salát coleslaw, smažené bylinkovo-česnekové hranolky a opečený sandwich chléb	139 Kč



```
<div class="restab_wrap">

          <div id="tabtop" class="tabcontent-wrapper brstd  daily-menu ">
        <div class="ui segment"><div id="menu-container" class="relative">           <div c
        <div id="daily-menu-container" data-supertab-menu-type = "daily-menu" class="superta
        <div class="menu-preview  mt10" id="menu-preview">

                                            <div class="dm-serving-time mbot0" data-icon

        <div class="tmi-groups">



                                <div class="tmi-group  mtop">
                                      <div class="tmi-group-name bold fontsize
                  Thursday, 08 October (dnes)
                </div>



             <div class="tmi tmi-daily pb5 pt5  ">
                <div class="tmi-text-group col-l-14 col-s-14">
                  <div class="row">
```

More data sources

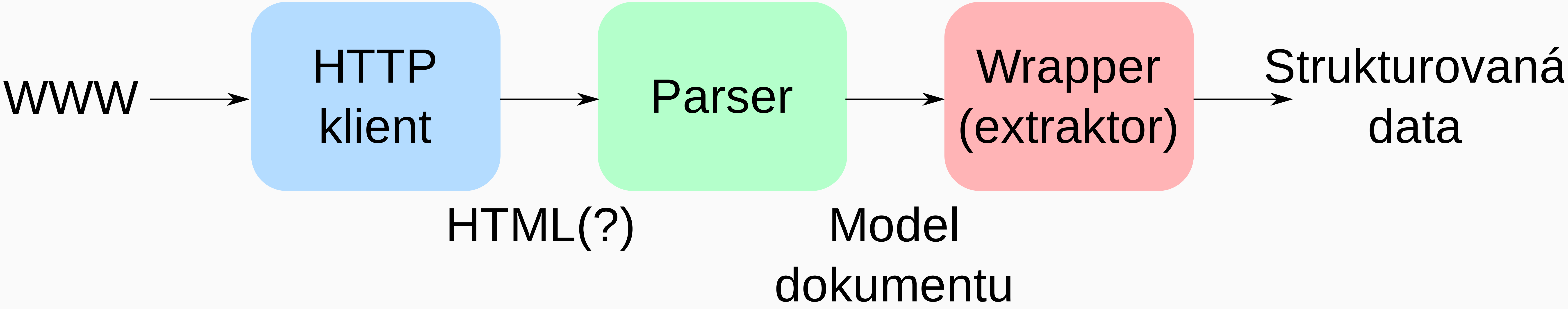
- Commercial
  - e-shops, real estate servers, flight tickets, sports results, competition monitoring
- Search results
  - E.g. position monitoring
- Public registers
  - time tables
  - trade register
  - websites of municipal councils
- Advertisement monitoring
- ∞ more

Partial problems

- Source data acquisition
  - How to download the necessary documents from the WWW (so that they contain what they are supposed to?)
  - Parallelization
- **Data identification and extraction**
  - Identification of the requested data in the page
- Storage of results
  - There can be **a lot** of them.

Architecture

Basic architecture



Shell is your friend

Motivation: Evolution of a programmer

Before we start programming:

- wget, curl
- cat, grep, sed, cut
- awk (for true geeks :-)

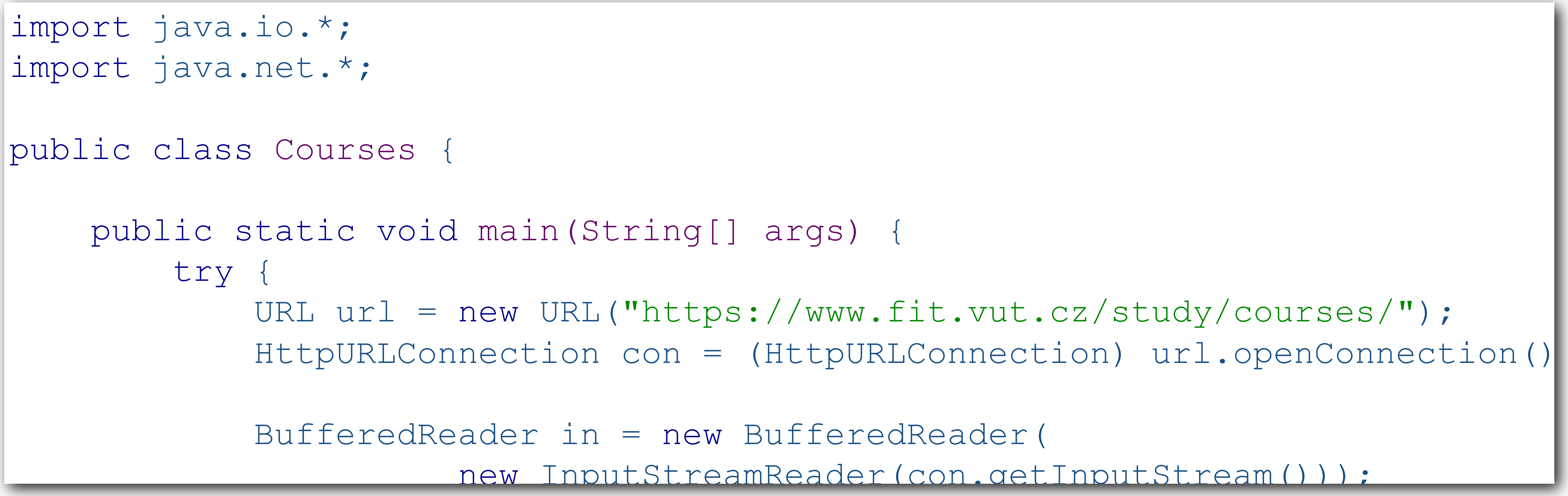
```
wget https://www.fit.vut.cz/study/courses/ -O out.html
cat out.html | grep 'list-links__link' | sed 's/<[^<>]*>/;/g' | sed 's/;;*/;
cat data.csv | cut -f2 -d';'
```

```
wget https://www.fit.vut.cz/study/courses/ -O - | grep 'list-links__link' |
```

The same in python3

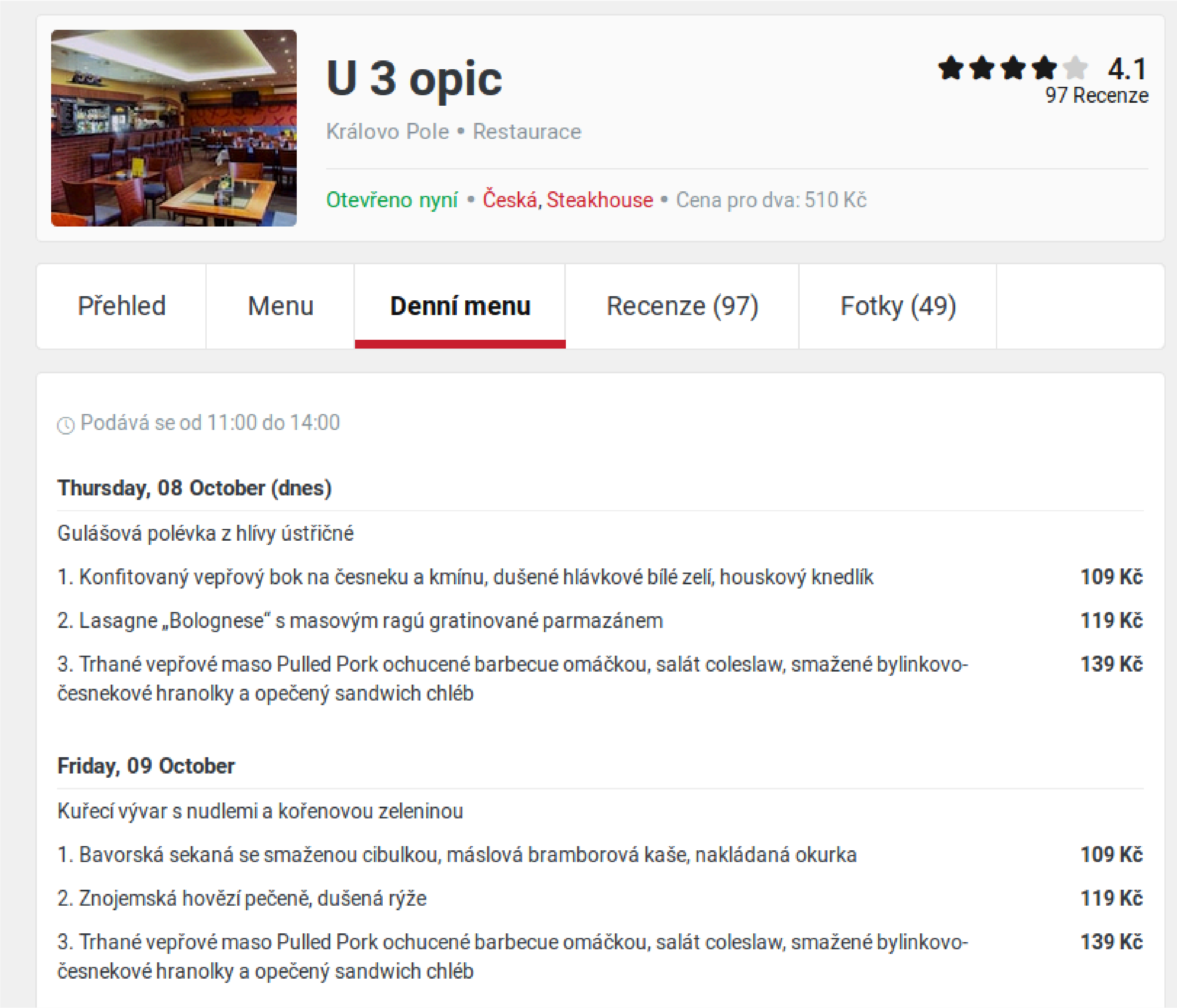
```
import urllib.request
import re

fid = urllib.request.urlopen('https://www.fit.vut.cz/study/courses/')
webpage = fid.read().decode('utf-8')
for line in webpage.split('\n'):
    if ('list-links__link') in line:
        line = re.sub(r"<[^<>]*>", ";", line);
        line = re.sub(r";;*", ";", line);
        print(line)
```

Java



```
import java.io.*;
import java.net.*;

public class Courses {

    public static void main(String[] args) {
        try {
            URL url = new URL("https://www.fit.vut.cz/study/courses/");
            HttpURLConnection con = (HttpURLConnection) url.openConnection()

            BufferedReader in = new BufferedReader(
                      new InputStreamReader(con.getInputStream()));
```

Limitations of the simple approach



U 3 opic

97 Recenze

Královo Pole • Restaurace

Otevřeno nyní • Česká, Steakhouse • Cena pro dva: 510 Kč

Podává se od 11:00 do 14:00

Thursday, 08 October (dnes)

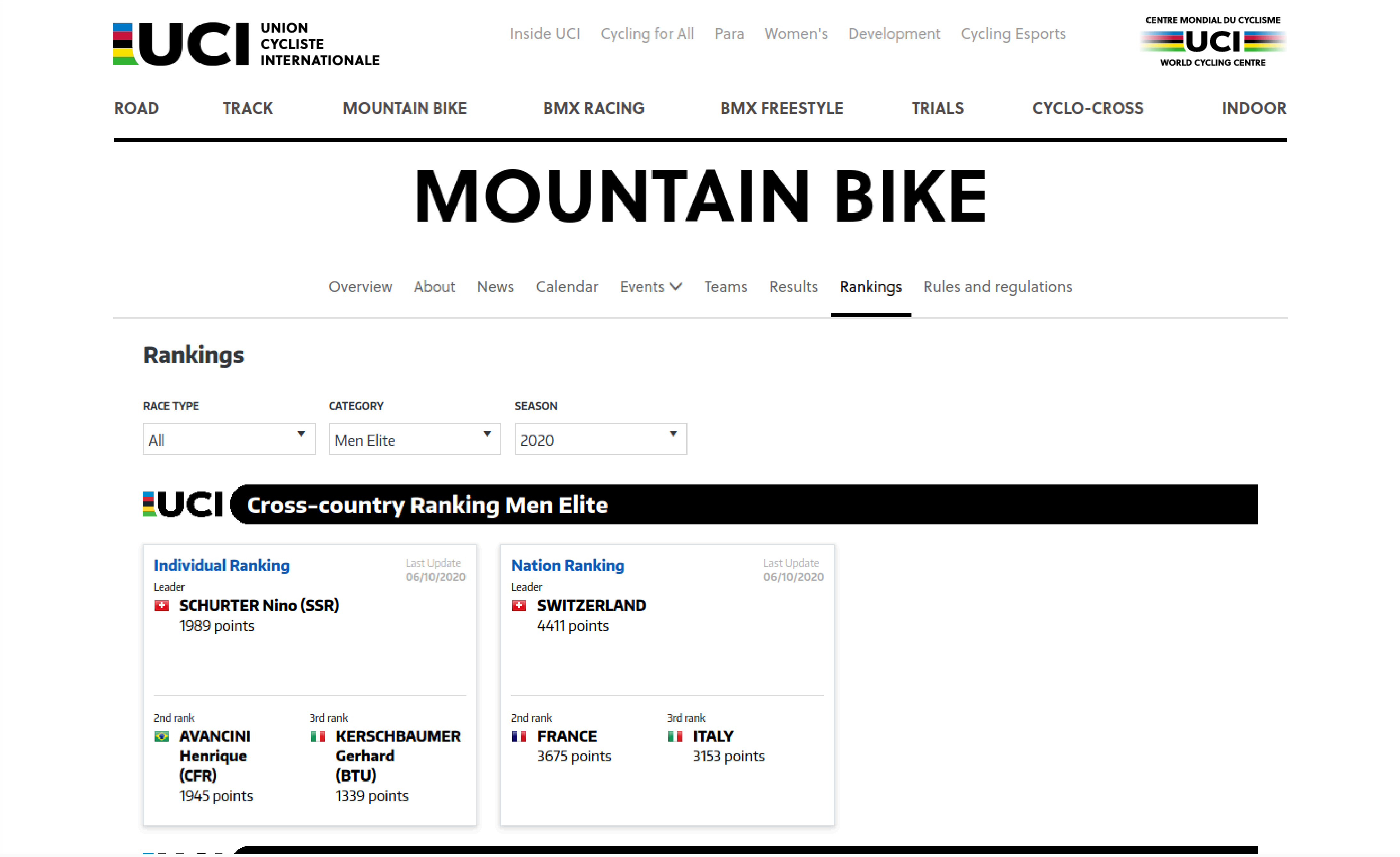
Gulášová polévka z hlívy ústřičné	
1. Konfitovaný vepřový bok na česneku a kmínu, dušené hlávkové bílé zelí, houskový knedlík	**109 Kč**
2. Lasagne „Bolognese“ s masovým ragú gratinované parmazánem	**119 Kč**
3. Trhané vepřové maso Pulled Pork ochucené barbecue omáčkou, salát coleslaw, smažené bylinkovo-česnekové hranolky a opečený sandwich chléb	**139 Kč**

Friday, 09 October

Kuřecí vývar s nudlemi a kořenovou zeleninou	
1. Bavorská sekaná se smaženou cibulkou, máslová bramborová kaše, nakládaná okurka	**109 Kč**
2. Znojemská hovězí pečeně, dušená rýže	**119 Kč**
3. Trhané vepřové maso Pulled Pork ochucené barbecue omáčkou, salát coleslaw, smažené bylinkovo-česnekové hranolky a opečený sandwich chléb	**139 Kč**

Source page – regex?!

Limitations of the simple approach



MOUNTAIN BIKE

Rankings

RACE TYPE: All

CATEGORY: Men Elite

SEASON: 2020

UCI Cross-country Ranking Men Elite

Individual Ranking
Last Update 06/10/2020
Leader
SCHURTER Nino (SSR)
1989 points

2nd rank
AVANCINI Henrique (CFR)
1945 points

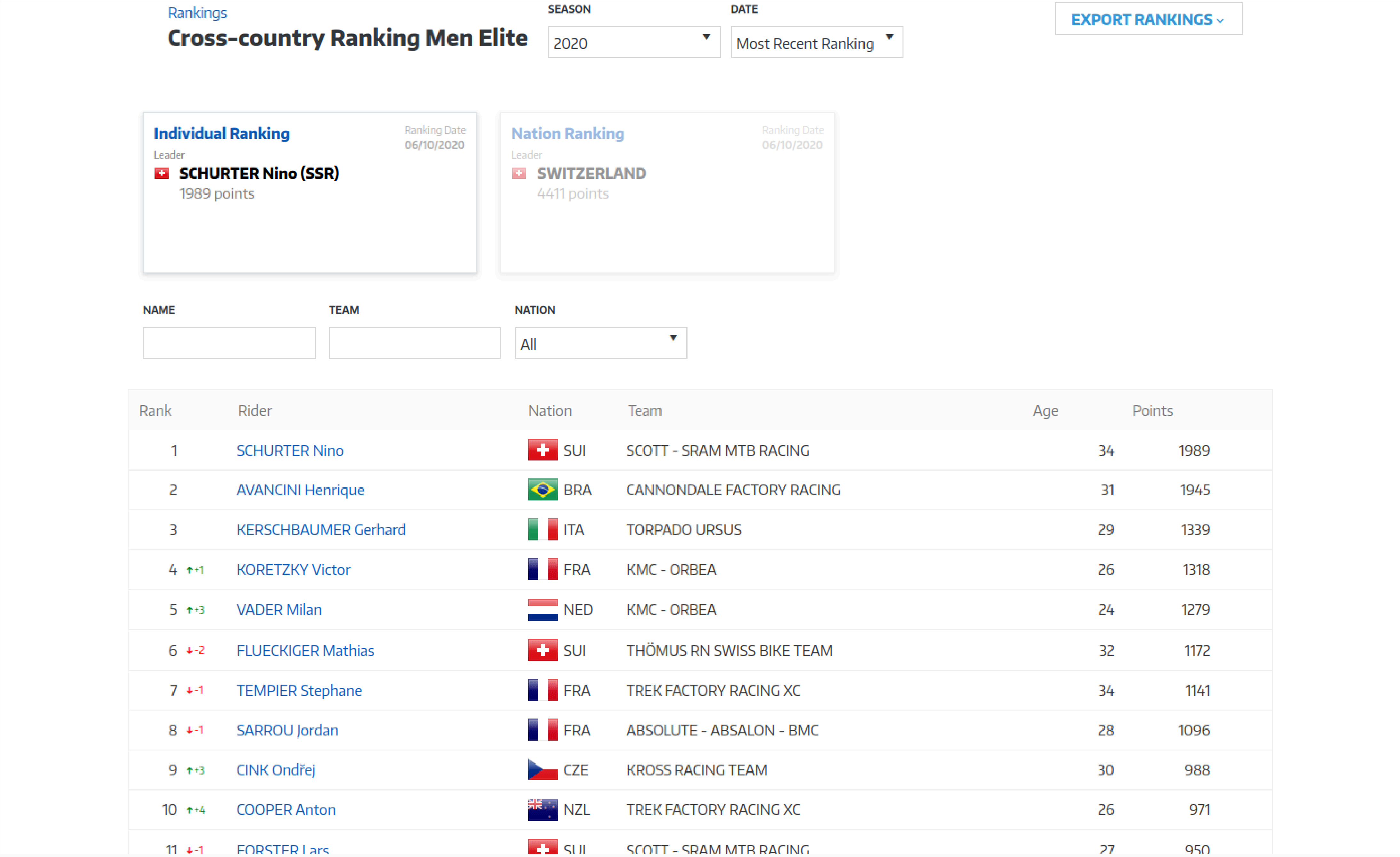
3rd rank
KERSCHBAUMER Gerhard (BTU)
1339 points

Nation Ranking
Last Update 06/10/2020
Leader
SWITZERLAND
4411 points

2nd rank
FRANCE
3675 points

3rd rank
ITALY
3153 points

Limitations of the simple approach



Rankings

Cross-country Ranking Men Elite

SEASON 2020

DATE Most Recent Ranking

Individual Ranking — Ranking Date 06/10/2020
Leader
SCHURTER Nino (SSR)
1989 points

Nation Ranking — Ranking Date 06/10/2020
Leader
SWITZERLAND
4411 points

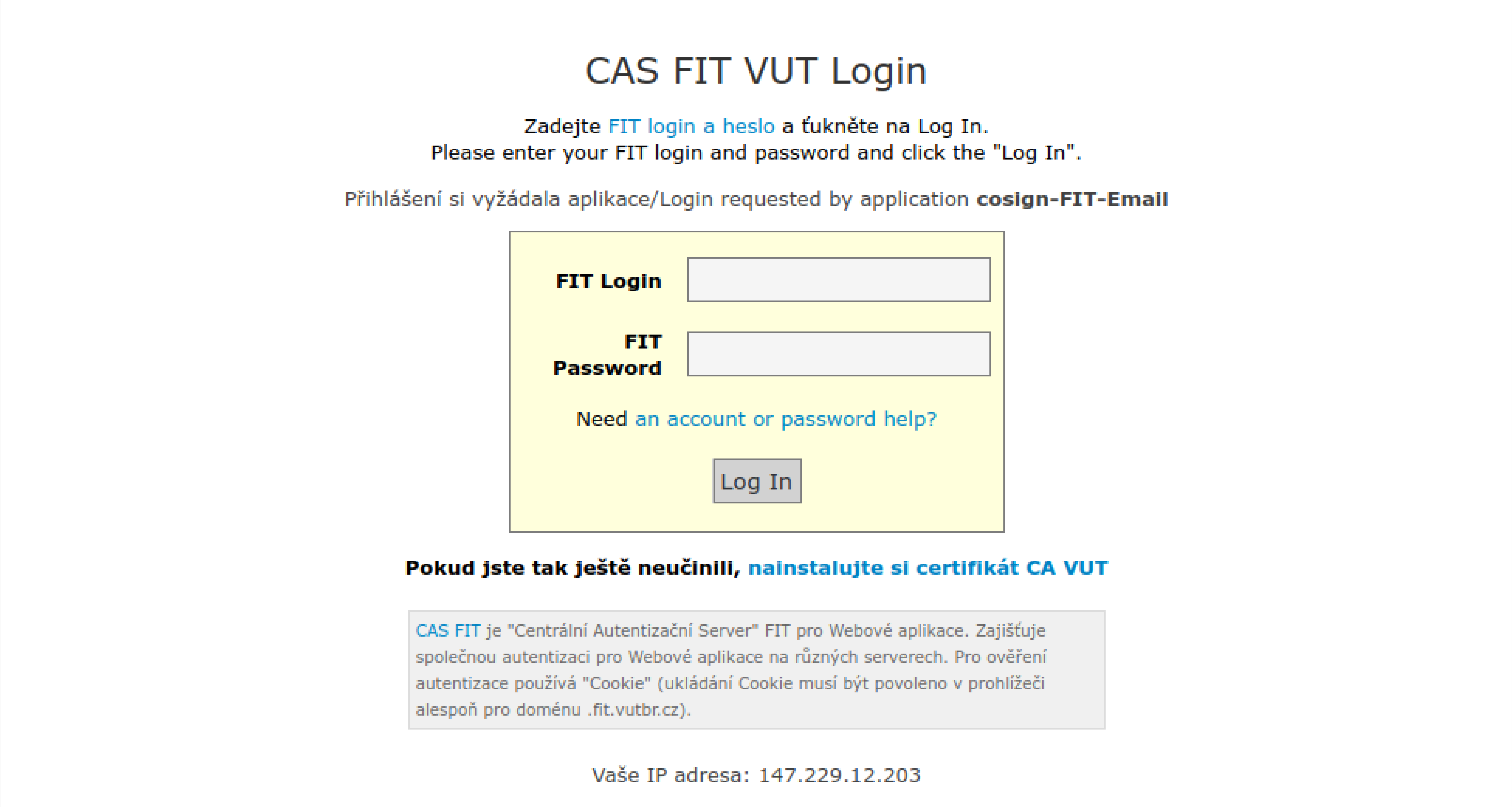
NAME

TEAM

NATION All

Rank	Rider	Nation	Team	Age	Points
1	SCHURTER Nino	SUI	SCOTT - SRAM MTB RACING	34	1989
2	AVANCINI Henrique	BRA	CANNONDALE FACTORY RACING	31	1945
3	KERSCHBAUMER Gerhard	ITA	TORPADO URSUS	29	1339
4 ↑+1	KORETZKY Victor	FRA	KMC - ORBEA	26	1318
5 ↑+3	VADER Milan	NED	KMC - ORBEA	24	1279
6 ↓-2	FLUECKIGER Mathias	SUI	THÖMUS RN SWISS BIKE TEAM	32	1172
7 ↓-1	TEMPIER Stephane	FRA	TREK FACTORY RACING XC	34	1141
8 ↓-1	SARROU Jordan	FRA	ABSOLUTE - ABSALON - BMC	28	1096
9 ↑+3	CINK Ondřej	CZE	KROSS RACING TEAM	30	988
10 ↑+4	COOPER Anton	NZL	TREK FACTORY RACING XC	26	971
11 ↓-1	FORSTER Lars	SUI	SCOTT - SRAM MTB RACING	27	950

Limitations of the simple approach



CAS FIT VUT Login

Zadejte FIT login a heslo a ťukněte na Log In.
Please enter your FIT login and password and click the "Log In".

Přihlášení si vyžádala aplikace/Login requested by application **cosign-FIT-Email**

FIT Login	
FIT Password	

Need an account or password help?

Pokud jste tak ještě neučinili, nainstalujte si certifikát CA VUT

CAS FIT je "Centrální Autentizační Server" FIT pro Webové aplikace. Zajišťuje společnou autentizaci pro Webové aplikace na různých serverech. Pro ověření autentizace používá "Cookie" (ukládání Cookie musí být povoleno v prohlížeči alespoň pro doménu .fit.vutbr.cz).

Vaše IP adresa: 147.229.12.203

- A login form with a redirect (and possible protection against machine filling)
- Manageable but complicated

Models of HTML documents

- Strings of characters
  - Easy implementation, speed, scalability
  - Regular expressions, HLRT wrappers
- Strings of tokens
  - A lexical analyzer recognizes tags, entities, text, etc.
  - HLRT wrappers
- Hierarchical models
  - Mostly DOM
  - Or its „lightweight“ variants

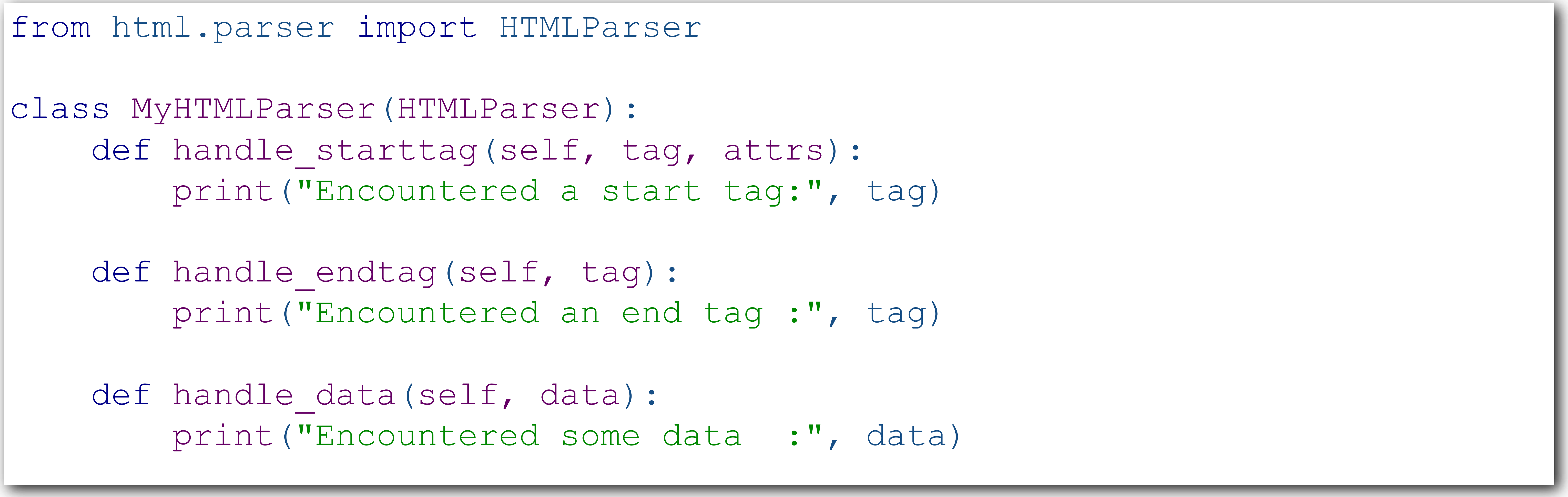
Wrapper

- Let us consider *n* data fields to be extracted from the document
- Different wrapper classes: LR, HLRT, ...
- HLRT wrapper:
  - **H**ead – a substring before the data block
  - **L**eft – left separator (for each data field)
  - **R**ight – right separator (for each data field)
  - **T**tail – a substring after the data block

$$wrapper = (h, t, l_1, r_1, l_2, r_2, \dots, l_n, r_n)$$

Token strings

- Event-driven parsers, e.g. `html.parser` in python



```python
from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):
    def handle_starttag(self, tag, attrs):
        print("Encountered a start tag:", tag)

    def handle_endtag(self, tag):
        print("Encountered an end tag :", tag)

    def handle_data(self, data):
        print("Encountered some data  :", data)
```

https://docs.python.org/3/library/html.parser.html

Document Object Model

- HTML code is represented as a **tree of objects**
- Different object types

HTML Element

- A section of a document content delimited by *tags*

```html
<p>Element content</p>

<div class="menu" id="mainmenu">
Element content<br> another content.
</div>

<div>Some <em>emphasized</em> text.</div>
```

- Always a single *root element*

A DOM tree

- The root node of type `Document`
- It has a single `Element` child node ~ the Document element (root)
- Elements can have child nodes of different types
  - `Element` – nested elements
  - `text` – text content (leaf nodes)
  - other types in some cases (e.g. `Entity`)

DOM navigation

- Standard methods of the `Document` and `Element` DOM classes
  - Element lookup: `getElementById()`, `getElementsByTagName()`
  - Tree navigation: `parentNode`, `childNodes`, …
  - Reading the content: `textContent`
- CSS selectors
  - They address a *set of elements*
  - `#main header .info`
- XPath
  - They address a *set of elements* too
  - `*[@id="main"]//table/tr[position() > 3]`

XPath

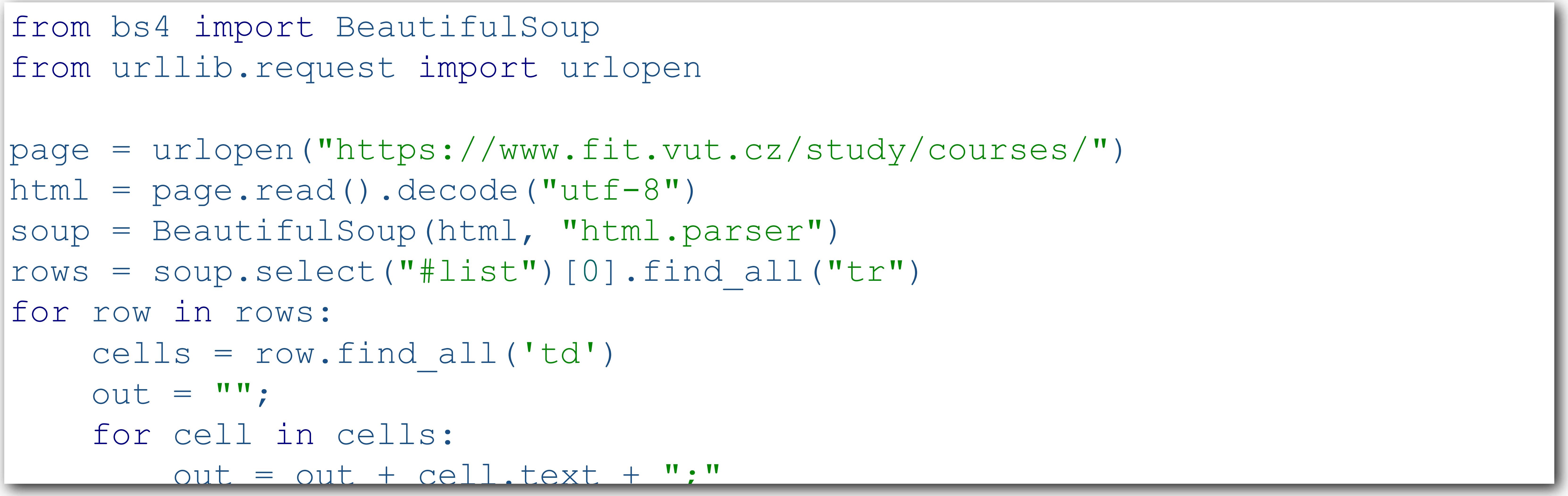
- Originally for XML documents, but also supported by some HTML libraries
- More complex syntax than CSS, but more options:
  - A generic expression for element properties in `[]` including attribute values, element order, and more
  - Navigation in different directions („axis“)
- See e.g. [MDN Documentation]

```
var res = document.evaluate('//head/title', document.documentElement,
    null, XPathResult.ANY_TYPE, null);
console.log(res.iterateNext().textContent);
```

Practical use of DOM

- A full-featured HTML 5 DOM parser is hard to find
  - Basically only in the web browser
- In practice, often simplified parsers with their own interface
  - Python: BeautifulSoup
  - Java: jSoup
  - JavaScript: cheerio

BeautifulSoup



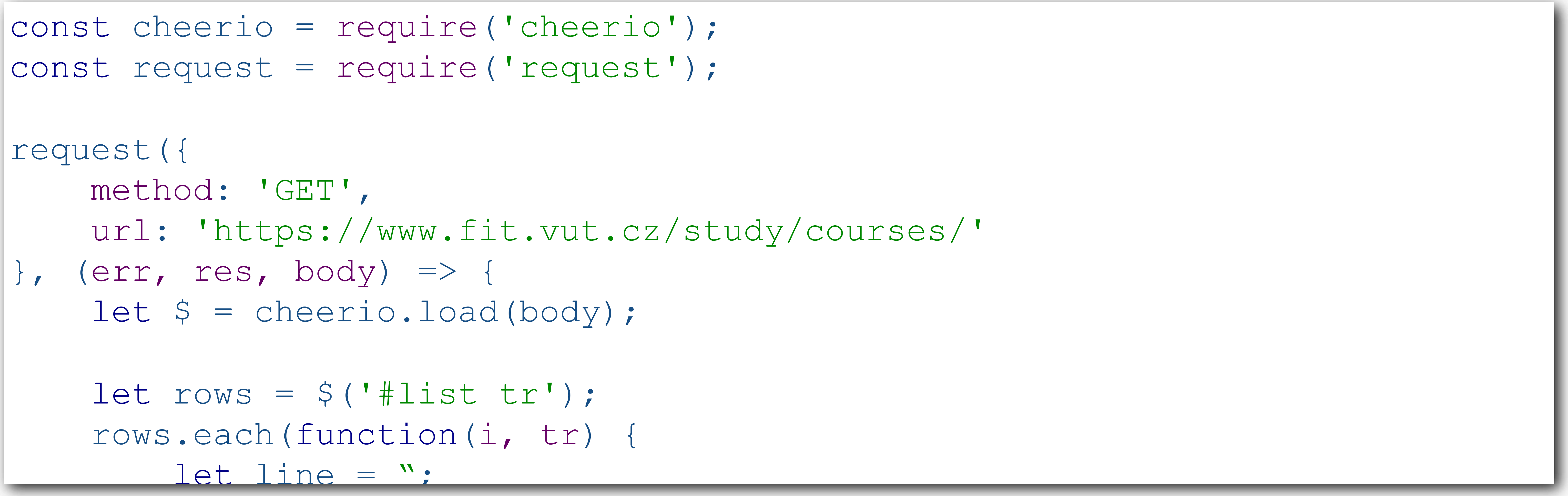
```
from bs4 import BeautifulSoup
from urllib.request import urlopen

page = urlopen("https://www.fit.vut.cz/study/courses/")
html = page.read().decode("utf-8")
soup = BeautifulSoup(html, "html.parser")
rows = soup.select("#list")[0].find_all("tr")
for row in rows:
    cells = row.find_all('td')
    out = "";
    for cell in cells:
        out = out + cell.text + ";"
```

jSoup

```
Document doc = Jsoup.connect("https://en.wikipedia.org/").get();
log(doc.title());
Elements newsHeadlines = doc.select("#mp-itn b a");
for (Element headline : newsHeadlines) {
  log("%s\n\t%s",
    headline.attr("title"), headline.absUrl("href"));
}
```

- It also provides a subset of the standard DOM interface

cheerio



```
const cheerio = require('cheerio');
const request = require('request');

request({
    method: 'GET',
    url: 'https://www.fit.vut.cz/study/courses/'
}, (err, res, body) => {
    let $ = cheerio.load(body);

    let rows = $('#list tr');
    rows.each(function(i, tr) {
        let line = '';
```

Mechanical Soup

- „Browser“ automation for BeautifulSoup
    - Project pages
- Classes and methods that simulate basic HTTP operations
    - „Clicking on links“ – getting the target and generating a GET request
    - „Form filling“ – getting the `action` and `method`, filling in the field values and sending the corresponding HTTP request.
- JavaScript is not supported

Puppeteer

- A Chrome browser remotely controlled from node.js
  https://github.com/puppeteer/puppeteer
- We can control the browser navigation
  - Entering URLs, clicking on links, filling out forms
  - API documentation
- Execution of JS code in the browser context (`page.evaluate()`)
  - E.g. for extracting data from DOM

Puppeteer Pros & Cons

- + Browser navigation
- + Convenient data extraction
  - DOM, CSS Selectors, **XPath**, any JavaScript code
- - Time and space demanding solution
  - The entire Chrome is started
- - Challenging handling of external conditions
  - E.g. race conditions, regional variants of the pages, ...
- - Difficult debugging
  - A part of the JS code runs in node.js, another part in the browser (different contexts)

Web APIs

- Some pages load the interesting content dynamically with JavaScript
  - `XMLHttpRequest` or `fetch()`
  - (formerly known as AJAX)
- The data source is an *HTTP endpoint*, that typically returns
  - HTML code snippets
  - or *Serialized structured data* – JSON, XML, ...

E.g. https://www.uci.org/road/rankings again

API Usage

- Slightly easier access to data
  - Usually one HTTP request is enough (GET or POST)
  - We parse a structured document
- The data format can be even more variable than a web page
  - Purely internal format of the application creators
- Efforts to complicate third party access
  - Authorization tokens, etc.
- There exist public endpoints with a well documented data format
  - E.g. The official portal for European data

Web Page Annotations

- Microformats
  - Annotation of HTML elements using predefined `class` values
  - A narrow set of defined formats
  - Easy implementation into an existing website
- Semantic technology, e.g. RDFa
  - HTML extension with new attributes (`resource`, `property`, …)
  - Allows transformation of HTML to *linked data* represented by RDF
  - Identification of objects and properties using **URI**
  - There are a number of dictionaries (*ontologies*) for different domains
    - E.g. FOAF, schema.org, …
- See the Semantic Web lecture

Intelligent extraction

That is, without „manual work“ in the form of searching for elements, regular expressions, CSS selectors, XPath expressions, etc.

1. Machine learning
    - Manually annotated examples of web pages
    - The wrapper/extractor parameters are automatically derived from them
2. Model-driven extraction
    - Specification of the expected data structure
        - Entities, attributes, relationships (ER diagram?)
        - Method of recognizing individual attributes
    - Finding the occurrence of the required data groups in the source page

Machine learning scenario

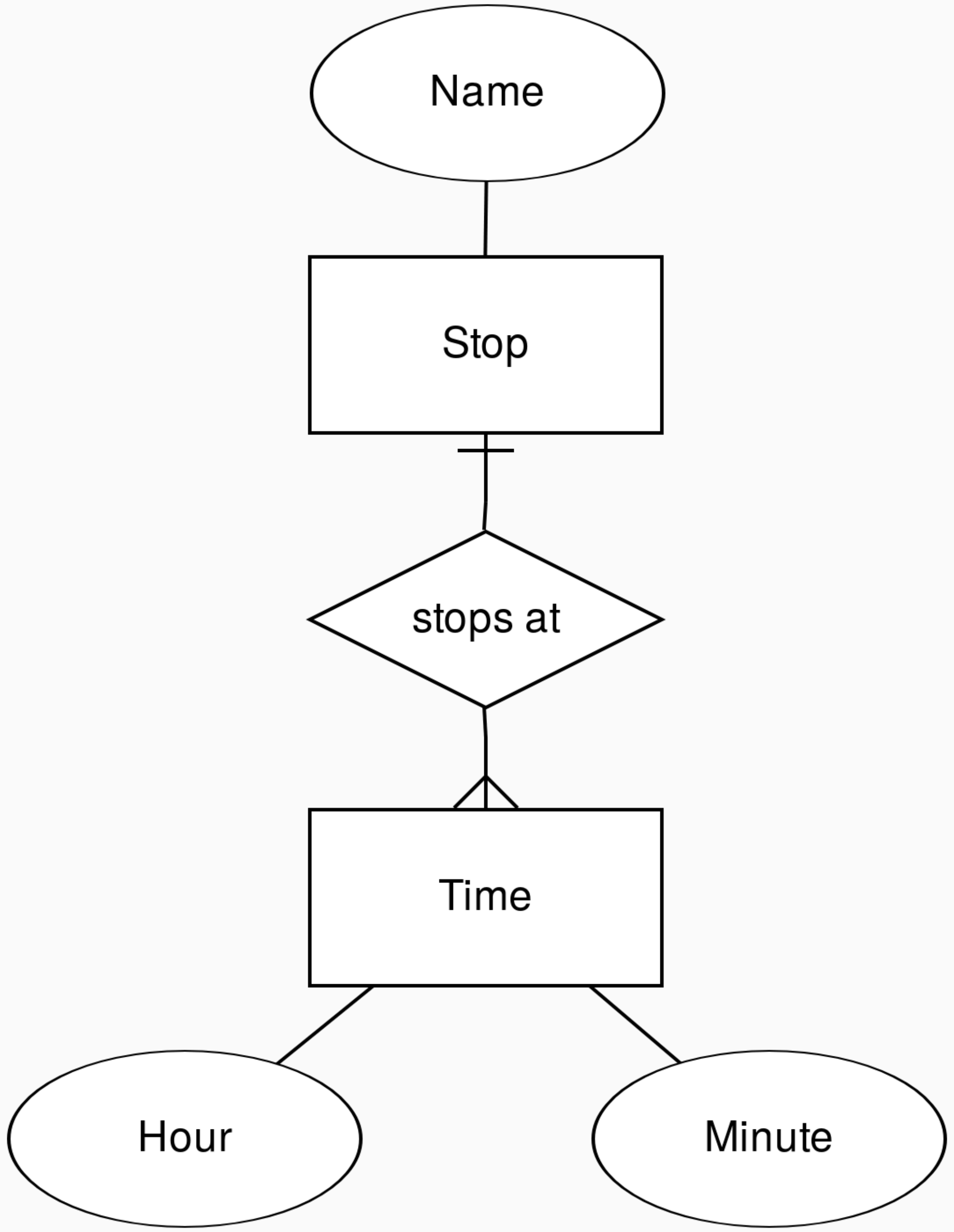
- Training set of documents
  - Usually documents from a single source
  - Annotation of the parts of the content to be extracted
  - Deriving extraction rules
- A set of new, unknown documents
  - Data extraction based on derived rules

Machine learning – methods

- Sequential page models (characters, tokens)
  - Grammar inference (*wrapper induction*), hidden Markov models, ...
- Hierarchical models
  - Generalized DOM (removal of implementation details)
  - Tree automata
- Visual document models
  - Page segmentation
  - Classification based on visual features

Model-driven extraction

- Input: Entities, attributes, relationships
- Approximate recognition of individual data fields
  - Regular expressions
  - Classification of text or visual properties
  - Mapping to a knowledge base (DBPedia, ...)
- Finding data records
  - Exploiting regularity, repeating patterns



That’s all!

Questions?