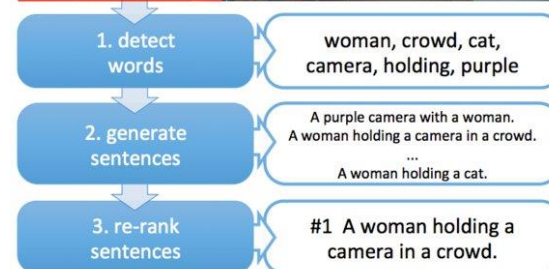
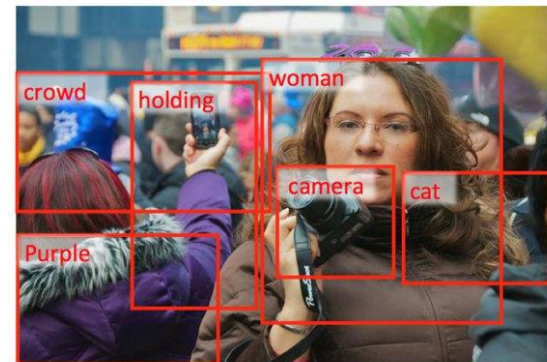


Data Science

Piotr Kałużny - <https://github.com/khashishin>

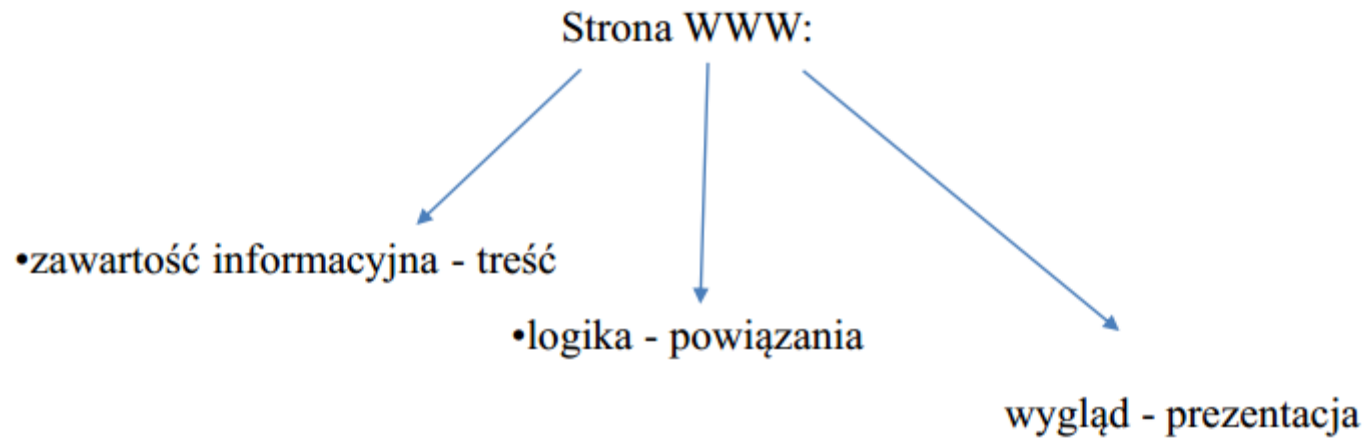
Dane

- Pracuje na różnych danych:
 - liczbowych: wyniki finansowe, odczyty z czujników,
 - tekstowych: komentarze, artykuły,
 - grafowych: komunikacja, pojęcia.
 - wielkich (Big Data)

















Budowa strony i jej zawartości



Jak jest zbudowana strona internetowa

- Strona to plik z tekstem, który jest napisany za pomocą takiego języka który może być interpretowany przez przeglądarkę np. HTML.
- Przeglądarka wie co ten tekst znaczy i jak można go „upiększyć” albo zamienić na konkretne elementy strony.
- Co jeżeli mamy wiele stron i obrazków – wtedy mamy spory folder, który albo przeładowuje jedną stronę/plik, albo ma bardzo wiele plików stron.

			
	js		16.05.2016 12:01:16
	game_design.css	2 KB	27.09.2016 17:57:34
	wyniki.csv	15 KB	09.06.2016 11:02:23
	post_1.php	2 KB	16.05.2016 12:01:43
	gra	9 KB	06.05.2016 08:36:36
	workspace_test	9 KB	05.05.2016 11:13:11
	test niekoop.png	25 KB	27.04.2016 23:10:04
	test wet niekoop.png	25 KB	27.04.2016 23:09:56
	test wet koop.png	25 KB	27.04.2016 23:09:45
	test koop.png	25 KB	27.04.2016 23:09:25
	page-loader.gif	46 KB	25.04.2016 10:08:19



- strona internetowa – HTML

```
<HTML>
  <HEAD>

    <TITLE>
      Tytuł strony
    </TITLE>

  </HEAD>
  <BODY>
    <a href="http://ue.poznan.pl">
      UE POZNAŃ
    </a>

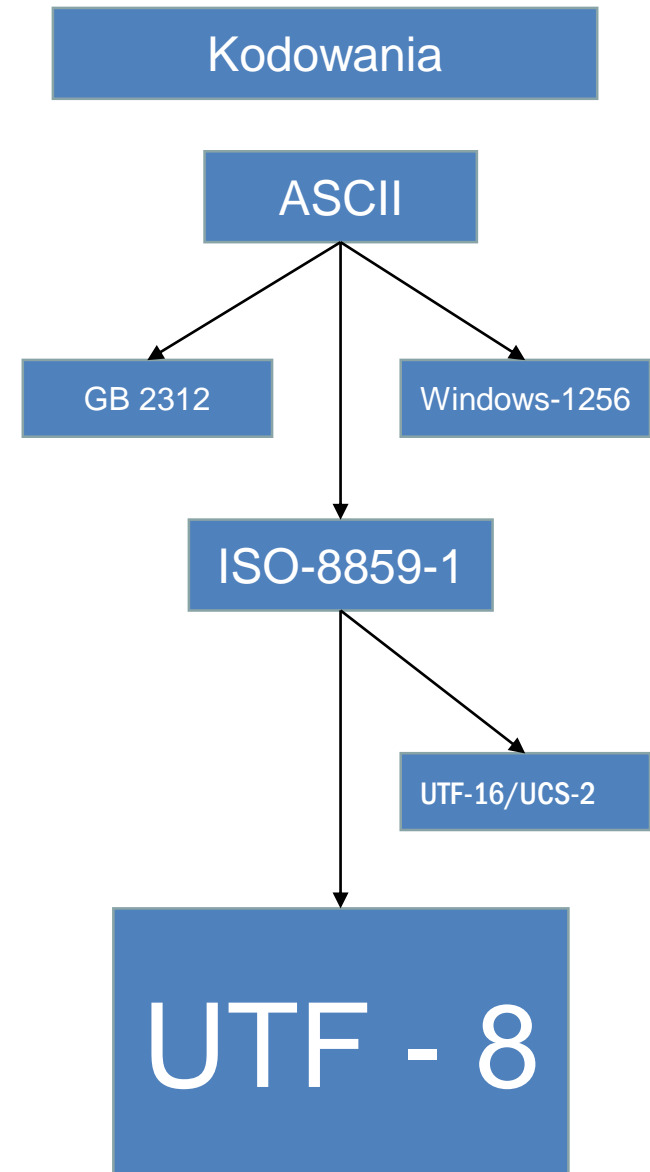
    <a href=„druga.html">
      Pierwsza strona z ćwiczeń.
    </a>

    <H1>
      Udało się podlinkować strony?
    </H1>

  </BODY>
</HTML>
```

- Często operując na jakichkolwiek plikach z tekstem spotkacie się z problemem „dziwnych znaczków”.
- Wynikają one z kodowania, sposoby zapisu specjalnych znaków, plik musi:
 - Potrafić zapisać znaki które mu wpisujecie.
 - Być odczytany z takim samym kodowaniem z jakim był zapisany (albo w przypadku strony internetowej poinformować ją w jakim kodowaniu jest).
- **Ćwiczenie 2** –
 - Wpiszcie na początku pliku:
 - `<meta charset="ISO-8859-1">`
 - A teraz zamieńcie na:
 - `<meta charset="UTF-8">`

Po co tyle kodowań? – np. kraje azjatyckie



Struktura HTML

- Jak przetwarzać taką hierarchiczną strukturę
 - [lxml package](https://lxml.de/parsing.html) (<https://lxml.de/parsing.html>)
- Korzystamy z języka „xpath”



The screenshot shows the FirePath tool interface. The 'XPath' field contains the absolute path: `html/body/div[1]/section/div[1]/div/div/div/div[1]/div/div/div/div/div[3]/div[1]/div/h4[1]/b`. An orange speech bubble points to this path with the text 'Absolute Path'. The HTML structure on the right shows the corresponding DOM tree, with the selected node highlighted in blue: `Testing`. The bottom status bar indicates '1 matching node'.

```
<br/>
- <div class="row featured-boxes">
  - <div class="col-md-3">
    - <div class="featured-box" style="height: 700px;">
      - <h4>
        - <b>Testing</b>
      - </h4>
      - <ul id="java_technologies" class="menu">
        - <p style="line-height: 15px;">
      - <h4>
```

1 matching node

Jak korzystać

- <https://docs.python-guide.org/scenarios/scrape/>

Let's start with the imports:

```
from lxml import html
import requests
```

Next we will use `requests.get` to retrieve the web page with our data, parse it using the `html` module, and save the results in `tree`:

```
page = requests.get('http://econpy.pythonanywhere.com/ex/001.html')
tree = html.fromstring(page.content)
```

```
<div title="buyer-name">Carson Busses</div>
<span class="item-price">$29.95</span>
```

Knowing this we can create the correct XPath query and use the `lxml xpath` function like this:

```
#This will create a list of buyers:
buyers = tree.xpath('//div[@title="buyer-name"]/text()')
#This will create a list of prices
prices = tree.xpath('//span[@class="item-price"]/text()')
```




Ćwiczenie 1



BeautifulSoup

- <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

```
1 from urllib import request
2 from bs4 import BeautifulSoup
3
4 start_website = "https://www.rp.pl/"
5 site = BeautifulSoup(request.urlopen(start_website).read(), "html.parser")
6
7 daty = site.findAll('div', {'class': 'article-content-text' })
8 daty = set(daty)
9
10 for data in daty:
11     print (data.text)
```



Ćwiczenie 2

- Beautiful Soup – strona
<http://www.newsweek.pl>

Regex 101 - text

Język wyrażeń regularnych (regex)

- <https://regex101.com/>
- Kiedy nie jesteśmy w stanie wyekstrahować tekstu w prosty sposób ze struktury.

Regexp ćw

- * - dowolny ciąg znaków
- . – dowolny znak
- a[b]a , aba – ciąg znaków „aba”
- a[ba]a – ciąg znaków aba, aaa
- [0-9][0-9] – 2 cyfrowa liczba
- ^ - początek
- \$ - koniec



Silnik Selenium

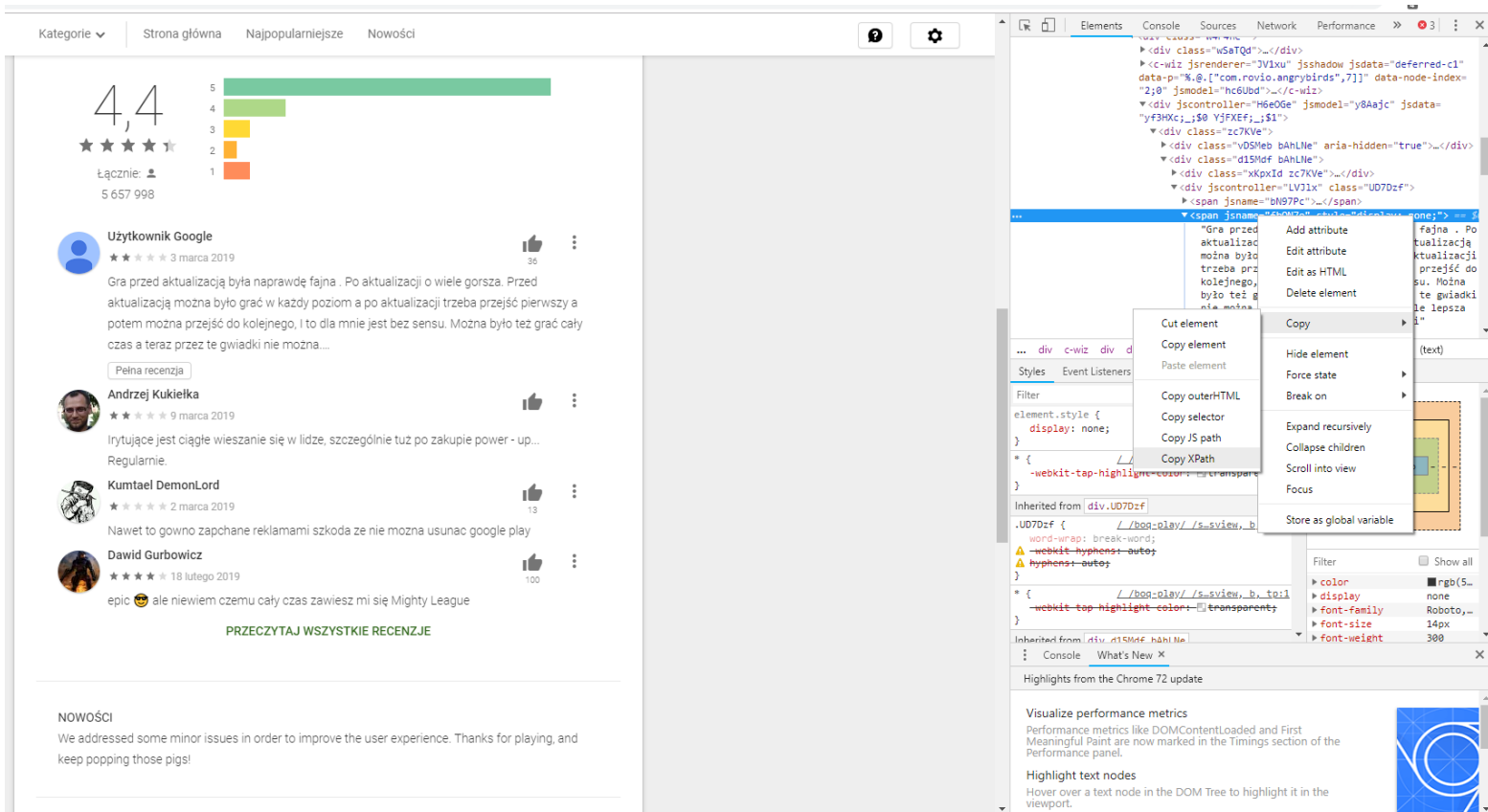
- Kiedy potrzebna jest interakcja
 - <https://selenium-python.readthedocs.io/>
- Nas interesuje webdriver

Komentarze

- XPATH:
- `//*[@id="fcxH9b"]/div[4]/c-wiz/div/div[2]/div/div[1]/div/div/div[1]/div[2]/div/div[2]/div[2]/span[2]`

Screenshot

- <https://play.google.com/store/apps/details?id=com.rovio.angrybirds&hl=pl>



NLP - materiały

- Scrapping:
- <https://github.com/alison-thaung/BigSurv/tree/master/code>
- <https://timber.io/blog/an-intro-to-web-scraping-with-lxml-and-python/>
- NLP:
- <https://github.com/rochelleterman/BigSurvText>