

Python średnio zaawansowany

Dzień 5





Blok nr 3:

Przetwarzanie danych



AGENDA

- Wyrażenia regularne
- Przetwarzanie HTML



Wyrażenia regularne



Wyrażenia regularne

<u>Wyrażenie regularne</u> (ang. regular expression, regex lub regexp) jest notacją (językiem) zapisu ciągu znaków, który określa poszukiwany **wzorzec**, np..:

- Polski kod pocztowy składa się z dwóch cyfr, po których następuje znak myślnika a następnie pojawiają się kolejne trzy cyfry: 30-415
- Wzorzec regularny: [0-9][0-9]-[0-9][0-9]



Wyrażenia regularne

```
Dowolna cyfra z zakresu od 0

do 9

Znak "-"

[0-9][0-9] [0-9][0-9][0-9]

Trzy dowolne cyfry
```

Kolejność elementów ma kluczowe znaczenie!



Określenie znaków:

- \d dowolna cyfra
- \s dowolny biały znak
- \w dowolny znak (litera, cyfra, podkreślnik)
- \D dowolny znak nie będący cyfrą
- \S dowolny znak nie będący białym znakiem
- W dowolny znak nie będący literą, cyfrą lub znakiem



Określenie krotności:

- ? zero lub jedno wystąpienie dowolnego znaku
- * zero lub więcej wystąpień dowolnego znaku
- + jedno lub więcej wystąpień dowolnego znaku
- . dowolny, pojedynczy znak

{min, max} - liczba wystąpień



Inne elementy wyrażeń regularnych:

\\ − po prostu ukośnik

\. – po prostu kropka (nie jest to dowolny znak)

[AB] – A lub B (umożliwia zdefiniowanie liczby powtórzeń)

A|B - A |B|



```
[0-9], [a-z] – znak ze zbioru
```

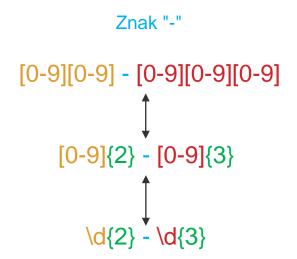
[^x] – wykluczenie x

- ^ początek wiersza
- \$ koniec wiersza



Optymalizacja

Dowolna cyfra z zakresu od 0 do 9



Dowolna cyfra z zakresu od 0 do 9

Liczba wystąpień



Wyrażenie regularne w Pythonie

```
import re

text = 'Some long text with 30-415 post code somewhere in it.'
pattern = '[0-9][0-9]-[0-9][0-9][0-9]'

match = re.search(pattern, text)
if match:
    start_pos = match.start()
    end_pos = match.end()
    print(text[start_pos:end_pos])
```



Wyrażenia regularne w Pythonie

Przydatne metody dostępne w module <u>re</u>:

- compile(pattern) kompiluje, co przyspiesza wyszukiwanie
- findall(pattern, string) zwraca obiekt z wszystkimi dopasowaniami
- search(pattern, string) zwraca obiekt z pierwszym dopasowaniem
- match(pattern, string) zwraca dopasowanie o ile jest na początku string'a



Parsowanie HTML

Moduł bs4 - BeautifulSoup



```
html doc = """
<html><head><title>The Dormouse's story</title></head>
<body>
<b>The Dormouse's story</b>
Once upon a time there were three little sisters; and their
names were
<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
and they lived at the bottom of a well.
...
```



Obiekt klasy BeautifulSoup odpowiada za interakcję z dokumentem HTML

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html_doc, 'html.parser')
soup.title
# <title>The Dormouse's story</title> - tag
soup.title.string
# u'The Dormouse's story' - text
soup.p
# <b>The Dormouse's story</b> - tag
```



```
soup.p['class'] - atrybut
# u'title'

soup.a
# <a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>
```



```
soup.find_all('a')
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
# <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
# <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]
soup.find(id='link3')
# <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>
```





Dzięki!