Wyrażenia regularne

Wyrażenia regularne

Wyrażenia regularne (ang. regular expressions, w skrócie regex lub regexp) – wzorce, które opisują łańcuchy symboli. Teoria wyrażeń regularnych jest związana z teorią języków regularnych. Wyrażenia regularne mogą określać zbiór pasujących łańcuchów, mogą również wyszczególniać stotne części łańcucha. – Wikipedia

Składnia wyrażeń regularnych

- Istnieje wiele znaków, które w wyrażeniach regularnych mają specjalne znaczenie.
- Warto korzystać z narzędzi do weryfikacji poprawności napisanych przez siebie wyrażeń regularnych.
  - https://regex101.com/ - narzędzie do sprawdzania działania regexp.
  - https://regexper.com/ - narzędzie do wizualizacji wzorców.

Składnia wyrażeń regularnych - podstawy

- `.` - dowolny znak
- `a` - jeden znak (w tym wypadku litera a)
- `abc` - ciąg znaków (w tym wypadku abc)
- `a|b` - znak a **lub** b
- `a*` - 0 lub więcej powtórzeń danego znaku (tutaj litery a)

Kwantyfikatory

- `*` - 0 lub więcej wystąpień
- `+` - 1 lub więcej wystąpień
- `?` - 0 lub 1 wystąpienie
- `{2}` - dokładnie 2 wystąpienia
- `{2, 5}` - od 2 do 5 wystąpień
- `{3,}` - 3 lub więcej wystąpień
- `{,7}` - maksymalnie 7 wystąpień

Składnia wyrażeń regularnych - grupy

- `(...)` - Grupa
- `(?P<MYGROUP>...)` - Grupa o nazwie MYGROUP
- `\1` - pierwsza znaleziona grupa

Składnia wyrażeń regularnych - klasy znaków

- `[a-cx-z]`- 1 znak ze zbioru: a,b,c,x,y,z
- `[^a-cw]` - 1 znak ze zbioru: a,b,c,w
- `\d` - 1 cyfra
- `\D` - 1 znak nie będący cyfrą
- `\s` - 1 biały znak
- `\S` - 1 znak spoza białych
- `\w` - 1 znak należący do grupy używalnych w słowach
- `\W` - 1 znak nie należący do grupy używalnych w słopakowanych

Składnia wyrażeń regularnych - asercje

- `^` - początek łańcucha znaków
- `$` - koniec łańcucha znaków
- `\A` - początek łańcucha znaków (ignoruje flagę m)
- `\Z` - koniec łańcucha znaków (ignoruje flagę m)
- `\b` - granica słowa
- `\B` - poza granicą słowa

Składnia wyrażeń regularnych

- `\n` - nowa linia
- `\r` - powrót karetki (carriage return)
- `\t` - tabulator
- `\NNN` - znak o podanym kodzie w formacie ósemkowym
- `\xNN` - znak o podanym kodzie w formacie szesnastkowym

Moduł re

```
matchObject = re.search(pattern, input_str, flags=0)
```

- Jeśli nie będzie trafień zwróci `None`
- Zwróci obiekt MatchObject jeśli udało się dopasować jakiś tekst.
- **Zatrzymuje się po znalezieniu pirwszego dopasowania**

Moduł re

```
matchList = re.findall(pattern, input_str, flags=0)
matchList = re.finditer(pattern, input_str, flags=0)
```

- Znajdują **wszystkie** dopasowania.
- Działają w podobny sposób:
  - `.findall()` zwraca listę dopasowań
  - `.finditer()` zwraca iterator zwracający kolejne dopasowania

Moduł re

```
re.sub(pattern, replacement_pattern, input_str, count,
    flags=0)
```

- Służy do zamiany łańcuchów znaków na podstawie dopasowania do wyrażenia regularnego.
- Z reguły w parametrze `replacement_pattern` używane są odwołanid do grup dopasowanych do wyrażenia regularnego.

```
import re

regex = r"([a-zA-Z]+) (\d+)"
print(re.sub(regex, r"\2 of \1", "June 24, August 9,
    Dec 12"))
```

Moduł re - Flagi

- `re.IGNORCASE` - ignoruje wielkość liter.
- `re.ASCII` - powoduje, że znaczniki `\w`, `\b`, `\s`, `\d` organiczają się tylko do znaków ASCII
- `re.DOTALL` - `.` oznacza także znak nowej linii
- `re.MULTILINE` - łańcuch znaków do przeszukania może być wieloliniowy (wpływa na symbole ^ i $)

Moduł re

```
import re

regex = re.compile(r"(\w+) World")
result = regex.search("Hello World is the easiest")

for result in regex.findall("Hello World, Bonjour
    World"):
    print(result)

print(regex.sub(r"\1 Earth", "Hello World"))
```

Kompilacja wyrażeń regularnych

- W przypadku wielokrotnego użycia tego samego wyrażenia regularnego warto je skompilować.
- Kompilacja wyrażenia przyspiesza dopasowywanie wzorców
- Dla obiektów skompilowanych wyrażeń regularnych istnieją metody takie jak dla modułu `re`.