
	Politechnika Bydgoska im. J. J. Śniadeckich Wydział Telekomunikacji, Informatyki i Elektrotechniki		
Przedmiot	Skryptowe języki programowania		
Prowadzący	mgr inż. Martyna Tarczewska		
Temat	<i>Regex - drugie starcie</i>		
Student			
Nr ćw.	8	Data wykonania	
Ocena		Data oddania spr.	

1. Cel ćwiczenia

Celem ćwiczenia jest poznanie dalszych informacji związanych z wyrażeniami regularnymi w języku skryptowym Python. Podczas laboratorium zostaną wykonane zadania, które mają na celu utrwalić informacje z zakresu tworzenia wyrażeń regularnych.

2. Informacje podstawowe

2.1. Kwantyfikatory zachłanne i leniwe

Dotychczas korzystaliśmy z kwantyfikatorów zachłannych. Ich cechą charakterystyczną jest wyszukiwanie tak długiego dopasowania, jak to tylko możliwe. Istnieją też ich wersje leniwe, które wyszukują najkrótsze pasujące elementy.

```
print(re.findall('a+', 'aaaa'))
print(re.findall('a+?', 'aaaa'))
```

2.2. Otoczenie

Bardzo ciekawym rozszerzeniem wyrażeń regularnych jest przeglądanie otoczenia dookoła naszego ciągu znaków. W języku angielskim nazywamy to lookahead, a dokładniej lookahead and lookbehind. Te konstrukcje nie konsumują znaków, a jedynie je przeglądają. Zapis **(?=kot)** zwróci dopasowania o długości 0 przed każdym ciągiem „kot”. Dopiero kiedy dodamy do wyrażenia **.{3}** zaczniemy otrzymywać dopasowania o długości 3. W tabeli przedstawiono różne konfiguracje lookahead.

Lookaround	Nazwa	Przykład	Wynik
?=	Positive lookahead	(?=kot){3}	Trzy znaki, które są równe „kot”
?!	Negative lookahead	(?!kot){3}	Trzy znaki, które nie są równe „kot”
?<=	Positive lookbehind	(?<=kot){3}	Trzy znaki, które poprzedza „kot”

?<!	Negative lookbehind	(?<!kot){3}	Trzy znaki, których nie poprzedza „kot”
-----	---------------------	-------------	---

Otoczenia można łączyć, np. `(?=kot)(?<!kot){3}` Takie wyrażenie będzie poszukiwało dowolnych trzech znaków równych „kot”, ale nie poprzedzonych napisem „kot”.

Prawidłowym będzie również zapis `{3}(?=kot)`. Taki wzorzec wyszuka trzy znaki, po których występuje ciąg „kot”.

3. Przebieg ćwiczenia

3.1. Zadanie 1.

Na podstawie kodu w punkcie 2.1. przetestować działanie leniwych i zachłannych kwantyfikatorów. Sprawdzić, jakie dopasowania uzyskamy za pomocą kwantyfikatorów: `+`, `+`, `*`, `*`, `?` oraz `?` Wyniki opisać w sprawozdaniu.

3.2. Zadanie 2.

Za pomocą lookahead z Inwokacji (plik z poprzednich zajęć):

- wypisać słowa, po których występuje „!”
- wypisać słowa z polskimi znakami,
- zliczyć wystąpienia słowa cię/ci.

3.3. Zadanie 3.

Wczytać plik adresy.txt. Zaproponować wyrażenia regularne do wydobycia nazwy ulicy, kodu pocztowego oraz numeru mieszkania.

3.4. Zadanie 4.

Zaproponowane w 3 zadaniu wyrażenia nie zawsze będą skuteczne. Zaproponować wyrażenie regularne do wydobycia nazwy ulicy dla adresu PBŚ:

Al. prof. S. Kaliskiego 7 85-796 Bydgoszcz

3.5. Zadanie 5.

W funkcji `match` można ustawić flagi, np. `I`. Przetestować działanie tej flagi korzystając np. z poniższej składni:

```
re.match("[a-z]{3}", "Ala ma kota a kot ma Ale", flags=re.I)
```

3.6. Zadanie 6.

Zaproponować wyrażenia regularne do walidacji haseł – stworzyć 3 progi siły hasła: słabe, średnie, mocne.

Wskazówka: skorzystać z lookahead.

3.7. Zadanie 7.

Typowym błędem przy szybkim wpisywaniu tekstu jest pisanie drugiej litery wyrazu dużą literą, np. BYdgoszcz (zamiast Bydgoszcz) czy POlska (zamiast Polska). Zaproponować wyrażenie regularne, które wyszuka wszystkie takie błędy w tekście (także te wewnątrz wyrazów np. poliTEchnika). Wyrazy dłuższe niż dwie litery mają być poprawiane automatycznie, natomiast o podmiannę wyrazu dwuliterowego (np. IT na It) spytać użytkownika przy każdym pojawieniu się takiego elementu.

Wskazówka: skorzystać z funkcji *sub*.

4. Sprawozdanie

Sprawozdanie z laboratorium powinno zawierać:

- wypełnioną tabelę z początku instrukcji,
- kody programów będących rozwiązaniami wszystkich zadań wraz z komentarzami,
- demonstrację działania programu,
- odpowiedzi na pytania zawarte w zadaniach,
- wnioski.