# Wyrażenia regularne – informacje podstawowe

# Metaznaki

Każde wyrażenie regularne powinno być ujęte w ukośniki. Zawarte między ukośnikami znaki oraz ich sekwencje mają specjalne znaczenie i są nazywane metaznakami. Na przykład gwiazdka (\*) oznacza: „tekst, który próbujesz dopasować, może być poprzedzony dowolną liczbą znaków albo żadnym”.

## Przykład:

Dzieła Pana Nowaka można zaklasyfikować jako....

Chcemy wyszukać słowa pasujące do „Pan Nowak”

Możemy użyć wyrażeń regularnych:

/Pan\* Nowak/

Przypuśćmy, że chcemy znaleźć dokładnie wyrażenie „Pan Nowak”. Jeśli wiemy, że, szukane słowa są zawsze rozdzielone przynajmniej jedną spacją możemy użyć znaku plus (+), gdyż wymaga on, by poprzedzający go znak wystąpił co najmniej jednokrotnie.

/Pan +Nowak/

Znak (.) pasuje do dowolnego znaku z wyjątkiem znaku przejścia do nowej linii.

## Przykład

Chcemy wyszukać znaczniki <html>

Mona użyć wyrażenia: /<.\*>/

Znaczenie: „pasuję do dowolnego ciągu znaków ograniczonego znakami < oraz >, nawet jeśli między nimi nie ma nic”.

Wyrażenie: /<.+>/

Znaczenie: „pasuję do dowolnego ciągu znaków ograniczonego znakami < oraz >, ale tylko jeśli pomiędzy nimi znajduje się przynajmniej jeden znak”. Będzie szukać po znaku”<” wielu znaków i na koniec będzie szukać „>” sprawdzając całą linijkę, więc wejdą też wyrażenia takie jak:

<h1><b>Wstęp</b></h1>

Jeśli chcemy wyszukać samą kropkę, musimy poprzedzić ją /

## Przykład:

Chcemy wyszukać liczbę 5.0. Musimy zapisać wyrażenie w postaci /5\.0/

# Grupowanie przy użyciu nawiasów

Chcemy wyszukać wartości, które odpowiadają przedrostkom w układzie SI, takim jak kilo-, mega-, giga- i tera-.

Czyli chcemy znaleźć wartości:

1,000  
1,000,000  
1,000,000,000  
1,000,000,000,000  
...

W tym przypadku również można użyć znaku plusa, ale trzeba jednocześnie zgrupować ciąg znaków ,000, aby plus odpowiadał mu w całości. Wówczas wyrażenie będzie miało postać:

/1(,000)+ /

Znaczenie: przy stosowaniu modyfikatorów takich jak plus potraktuj ten ciąg jako grupę”

# Klasy znaków

[] – odpowiadają jednemu znakowi ale możemy w nich podać listę akceptowalnych znaków.

## Przykład

Kolor „Szary” po angielsku to grey (brytyjski) lub gray (amerykański). Żeby móc wyszukać to angielskie słowo w dwóch wariantach możemy zastosować wyrażanie:

/gr[ae]y/

# Określanie zakresu

Zakres liczb od 0 do 9:

/[0-9]/

Do wyszukiwania cyfr możemy użyć metaznaku:

/\d/

# Zaprzeczanie

^ - umieszczamy po otwarciu nawiasu „[„ – zaprzeczamy wtedy zakres, czyli będą wyszukiwane wszystkie z wyjątkiem tego co w nawiasie np.

## Przykład

Chcemy znaleźć ciąg znaków Yahoo, ale bez kończącego nazwę wykrzyknika. Możemy użyć wyrażenia:

/Yahoo[^!]/

Uwaga!: Nie zadziała, jeśli „Yahoo” będzie ostatnim słowem w linii — a to ze względu na brak jakiegokolwiek znaku po nim — ponieważ zastosowanie nawiasów w powyższy sposób wymaga istnienia dowolnego znaku po szukanym słowie

# Podsumowanie metaznaków w wyrażeniach regularnych[[1]](#footnote-1):

| **Metaznaki** | **Opis** |
| --- | --- |
| / | Rozpoczyna i kończy wyrażenie regularne. |
| . | Pasuje do dowolnego znaku oprócz znaku nowego wiersza. |
| Element\* | Dopasowuje element zero lub większą liczbę razy. |
| Element+ | Dopasowuje element co najmniej raz. |
| Element? | Dopasowuje element zero razy lub raz. |
| [znaki] | Dopasowuje jeden znak spośród zawartych w nawiasach kwadratowych. |
| [^znaki] | Dopasowuje jeden znak, który nie znajduje się wśród podanych w nawiasach. |
| (regex) | Traktuje regex jako grupę do zliczania albo do opatrzenia znakiem \*, + lub ?. |
| lewo|prawo | Dopasowuje lewo lub prawo. |
| [l-r] | Zakres znaków od l do r. |
| ^ | Wymaga, aby dopasowanie nastąpiło na początku tekstu. |
| $ | Wymaga, aby dopasowanie nastąpiło na końcu tekstu. |
| \b | Dopasowanie następuje na granicy słowa. |
| \B | Dopasowanie następuje, jeśli nie jest to granica słowa. |
| \d | Dopasowuje do pojedynczej cyfry. |
| \D | Dopasowuje do pojedynczego znaku niebędącego cyfrą. |
| \n | Dopasowuje znak nowego wiersza. |
| \s | Dopasowuje biały znak. |
| \S | Dopasowuje znak niebędący białym znakiem. |
| \t | Dopasowuje znak tabulacji. |
| \w | Dopasowuje znak alfanumeryczny (a-z, A-Z, 0-9 oraz \_). |
| \W | Dopasowuje znak niebędący alfanumerycznym (wszystko oprócz a-z, A-Z, 0-9 oraz \_). |
| \x | Znak o kodzie x (przydaje się, jeśli x jest metaznakiem, a należy go dopasować). |
| {n} | Dopasowanie nastąpi dokładnie n razy. |
| {n,} | Dopasowanie nastąpi co najmniej n razy. |
| {min, max} | Dopasowanie nastąpi przynajmniej min razy i najwyżej maxrazy. |

Źródło:

# Przykłady wyrażeń regularnych:

| **Przykład** | **Dopasowania** |
| --- | --- |
| r | Pierwsza litera r w zdaniu *Szybki rudy lis* |
| od[bi][bi]era | Pasuje do *odbiera* lub *odibera* (a także do *odbbera* i *odiiera*) |
| od[bi]{2}era | Pasuje do *odbiera* lub *odibera* (a także do *odbbera* i *odiiera*) |
| od(bi|ib)era | Pasuje do *odbiera* lub *odibera* (ale nie do *odbbera* i *odiiera*) |
| kot | Ciąg *kot* w zdaniu *Fajne koty i duży piesek* |
| kot|pies | Ciągi *kot* lub *pies* w zdaniu *Fajne koty i duży piesek* |
| \. | . (ukośnik \ jest konieczny, bo . jest metaznakiem) |
| 5\.0\* | *5.*, *5.0*, *5.00*, *5.000* itd. |
| [a-f] | Dowolna z liter *a*, *b*, *c*, *d*, *e* albo *f* |
| koty$ | Tylko ostatnie słowo *koty* w tekście *Wszystkie moje koty to* |
| *przyjazne koty* |
| ^moje | Tylko pierwsze słowo *moje* w tekście *moje koty to moje pupilki* |
| \d{2,3} | Dowolna dwu- albo trzycyfrowa liczba (od *00* do *999*) |
| 7(,000)+ | *7,000*; *7,000,000*; *7,000,000,000*; *7,000,000,000,000* itd. |
| [\w]+ | Dowolne słowo składające się z jednego lub kilku znaków |
| [\w]{5} | Dowolne pięcioliterowe słowo |

źródło: Robert Nixon, PHP, MySQL i JavaScript wprowadzenie, Wyd. IV, Helion, Gliwice 2015,

Zadanie

Przerobić tutorial dotyczący wyrażeń regularnych ze strony:

https://regexone.com

1. Źródło: [↑](#footnote-ref-1)