Modul Praktikum Kuliah
Pengantar Regular Expression

versi: 18 Feb 2018, update: 11 April 2020

Yudi Wibisono (yudi@upi.edu), Masayu Leylia Khodra (masayu@informatika.org)

Regular expression (regex) adalah notasi standar yang mendeskripsikan suatu pola (pattern) berupa urutan karakter atau string. Regex digunakan untuk pencocokan string (string matching) dengan efisien. Regex sudah menjadi standar yang tersebar di semua tools dan bahasa pemrograman sehingga penting untuk dipelajari.

Tutorial ini akan menggunakan situs regexpal.com, dan kode program python di jupyter notebook.

Mencari string literal

Bentuk paling dasar adalah mencari string literal. Misalnya regex "halo" akan cocok dengan kata "halo" (gambar bawah) karena kedua string tersebut identik.

Jika terdapat lebih dari satu urutan string yang identik maka jumlah kecocokan akan menyesuaikan. Sebagai contoh, ada dua pola "halo" dalam "halohalo Bandung"

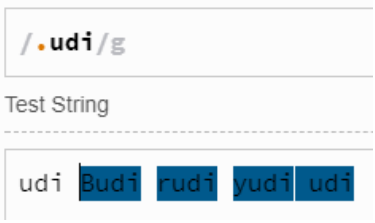
Contoh pencocokan string dengan regex di atas dalam bahasa Python:

```
import re
hasil = re.findall(r'halo', 'halohalo Bandung')
print(hasil)
```

Metakarakter

Dalam regex, terdapat sejumlah karakter khusus yang mempengaruhi proses pencocokan string. Sebagai contoh, karakter titik "." akan cocok dengan karakter apapun. Jadi regex `.udi` akan cocok dengan `Budi, Rudi, yudi` dan seterusnya.

Pada contoh di bawah, mengapa `udi` di awal string tidak cocok tetapi di bagian paling belakang cocok?



Jawab: karena "udi" dibagian belakang adalah " udi" dengan satu spasi di depan. Spasi dikenali sebagai satu karakter. Sehingga ".udi" akan cocok dengan " udi" (dengan spasi di depan 'u') tapi tidak cocok dengan "udi" (tanpa spasi di depan 'u').

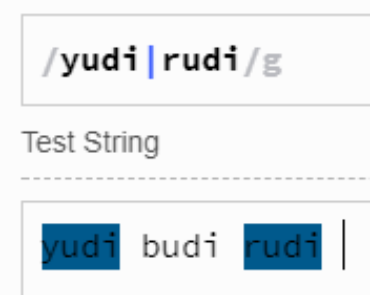
Dalam python:

```
re.findall(r'.udi', 'udi budi rudi yudi udi')
```

```
[4] import re
    hasil = re.findall(r'.udi', 'udi budi rudi yudi udi')
    print(hasil)
```

```
['budi', 'rudi', 'yudi', ' udi']
```

Contoh metakarakter lain adalah karakter "|" yang menyatakan or.



Metakarakter dalam regex adalah: `<([{\^-=$!|]})?*+.>`

Jika diperlukan metakarakter (seperti .) sebagai bagian pola regex, gunakanlah backslash '\'. Contoh untuk mencari tiga . yang berurutan: (coba buang backslash, apa yang terjadi?)

Python:

`re.findall(r'\.\.\.', 'Kalimat ini panjang ... dipotong ')`

```
[7]  import re
     hasil = re.findall(r'\.\.\.', 'Kalimat ini panjang ... dipotong ')
     print(hasil)
```

```
['...']
```

```
[8]  import re
     hasil = re.findall(r'...', 'Kalimat ini panjang ... dipotong ')
     print(hasil)
```

```
['Kal', 'ima', 't i', 'ni ', 'pan', 'jan', 'g .', '.. ', 'dip', 'oto', 'ng ']
```

Character Classes

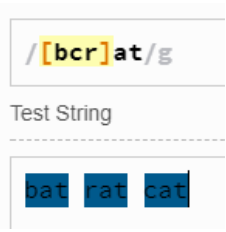
Tabel berikut memperlihatkan berbagai konstruksi regex untuk membentuk character class:

Construct	Deskripsi
`[abc]`	a, b, atau c (simple class)
`[^abc]`	Semua karakter **selain** a,b,c (negasi)
`[a-zA-Z]`	a sampai z atau A sampai Z, inclusive (range)
`[a-d[m-p]]`	a sampai d atau m sampai p (gabungan)
`[a-z&&[def]]`	d, e atau f (irisan)
`[a-z&&[^bc]]`	a sampai z, kecuali b dan c (subtraksi)
`[a-z&&[^m-p]]`	a sampai z, dan bukan m sampai p (subtraksi)

Berikut penjelasan lebih rinci untuk setiap jenis class.

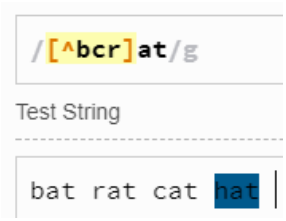
Simple Class

Bentuk paling sederhana dari character class adalah karakter dalam kurung siku. Sebagai contoh regex `[bcr]at` akan cocok dengan “bat”, “cat” dan “rat”



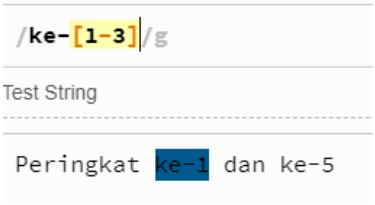
Negasi

Negasi adalah mencocokkan karakter selain dengan yang dituliskan. Tambahkan ^ pada regex dicontoh sebelumnya. [^bcr] berarti bukan "b", bukan "c" dan bukan "r". Semua yang sebelumnya cocok menjadi tidak cocok dan “hat” menjadi cocok.

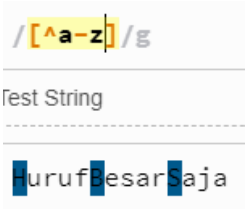


Ranges

Range digunakan untuk mencari karakter yang berada di dalam satu rentang. Misalnya a sampai e atau angka 1 sampai 5. Gunakan dash ‘-’ diantara karakter awal dan akhir. Contohnya [a-e] [1-5]

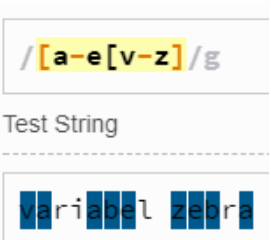


Negasi juga bisa diaplikasikan untuk range:

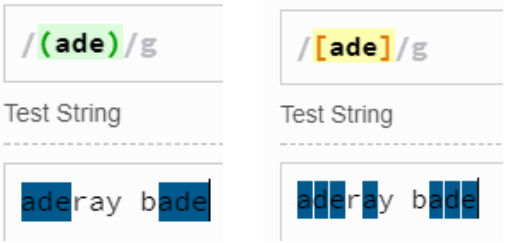


Unions

Gabungan dari range, sebagai contoh [a-e[v-z]] menyatakan huruf a sampai e digabung dengan v sampai z.



Catatan: jika menggunakan kurung biasa (bukan kurung siku), kata dianggap sebagai urutan kata biasa. Jadi jika `[ade]` menyatakan a atau d atau e, maka `(ade)` menyatakan kata “ade” (lihat gambar bawah).

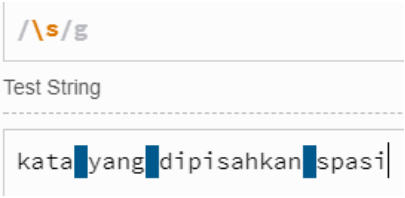


Predefined Character Class

Predefined class akan membuat regex lebih mudah dibaca dan mengurangi kesalahan. Berikut adalah predefined class.

Construct	Deskripsi
.	Semua karakter
\d	Digit [0-9]
\D	Non digit [^0-9] (hati-hati dengan huruf besar)
\s	Whitespace character [\t\n\x0B\f\r]
\S	Non whitespace character [^s]
\w	Word character [a-zA-Z_0-9]
\W	Non word character [^\w]

Contoh penggunaan \s untuk mencari spasi:



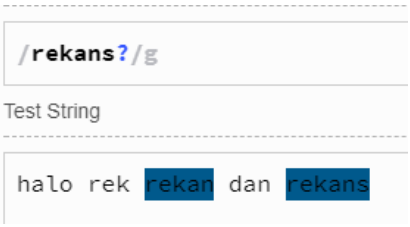
Quantifier

Quantifier digunakan untuk mendefinisikan jumlah perulangan pola. Quantifier punya peran sangat penting dalam regex.

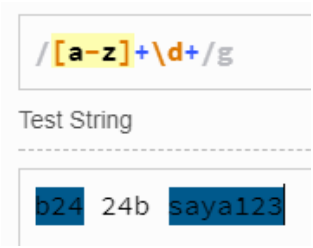
Construct	Arti
X?	X muncul **satu** atau tidak sama sekali

`X*`	X muncul **nol** atau banyak
`X+`	X muncul **satu** atau banyak
`x{n}`	X muncul tepat n kali
`x{n,}`	X muncul setidaknya n kali
`x{n,m}`	X muncul antara n sampai m kali

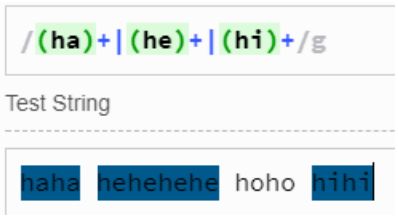
Sebagai contoh, berikut regex yang dapat menangkap kata “rekan” atau “rekans” (s bisa ada atau tidak). s? artinya karakter s dapat muncul atau tidak.



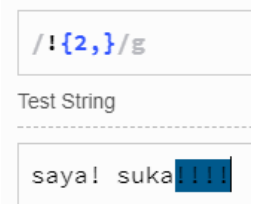
Regex berikut mengambil kata yang diawali dengan huruf a-z dan diakhiri dengan angka



Variasi tertawa:



Ambil tanda seru yang muncul lebih dari dua kali:



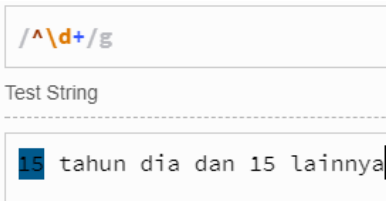
Boundary Matchers

Boundary matcher digunakan untuk mencari pola yang muncul di posisi tertentu. Misalnya di awal atau di akhir.

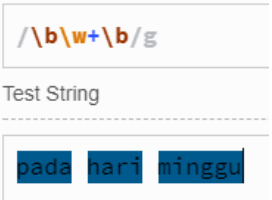
Construct	Deskripsi
^	Awal baris
$	Akhir baris
\b	Batas kata
\B	Batas bukan kata
\G	Akhir match sebelumnya
\Z	Akhir dari input tapi untuk final terminator jika ada
\z	Akhir dari input

Contoh deteksi titik yang berada di akhir kalimat . (dengan asumsi input adalah satu kalimat).

Contoh deteksi angka yang berada di depan



Contoh penggunaan \b untuk mengekstrak kata



Pada regexpal, terdapat cheat sheet di bagian kanan yang bisa digunakan sebagai referensi.

Cheat Sheet

Character classes

.	any character except newline
\w \d \s	word, digit, whitespace
\W \D \S	not word, digit, whitespace
[abc]	any of a, b, or c
[^abc]	not a, b, or c
[a-g]	character between a & g

Anchors

^abc$	start / end of the string
\b	word boundary

Escaped characters

\. * \\	escaped special characters
\t \n \r	tab, linefeed, carriage return
\u00A9	unicode escaped ©

Groups & Lookaround

(abc)	capture group
\1	backreference to group #1
(?:abc)	non-capturing group
(?=abc)	positive lookahead
(?!abc)	negative lookahead

Quantifiers & Alternation

a* a+ a?	0 or more, 1 or more, 0 or 1
a{5} a{2,}	exactly five, two or more
a{1,3}	between one & three
a+? a{2,}?	match as few as possible
ab|cd	match ab or cd

Latihan 1
(diambil dari: https://regex.sketchengine.co.uk/extra_regexps.html)
Coba kerjakan tanpa menggunakan tools terlebih dulu, setelah itu baru cek.

1. Mana yang cocok dengan regex `ab+c?`
    a. abc
    b. ac
    c. abbb
    d. bbc
2. Mana yang cocok dengan regex `a.[bc]+`
    a. abc
    b. abbbbbbbb
    c. azc
    d. abcbcbcbc
    e. ac
    f. asccbbbbcbcccc
3. Mana yang cocok dengan regex `abc|xyz`
    a. abc
    b. xyz

   c. abc|xyz
4. Mana yang cocok dengan regex `[a-z]+[\.\?!]`
   a. battle!
   b. Hot
   c. green
   d. swamping.
   e. jump up.
   f. undulate?
   g. Is.?
5. Mana yang cocok dengan regex `[a-zA-Z]*[^,]=`
   a. Butt=
   b. BotHEr,=
   c. Ample
   d. FIdDlE7h=
   e. Brittle =
   f. Other.=
6. Mana yang cocok dengan regex `[a-z][\.\?!]\s+[A-Z]`
   a. A. B
   b. c! d
   c. e f
   d. g. H
   e. i? J
   f. k L

7. Diberikan regex (regular expression) berikut ini **`b[ai]n?[rty]*[ui]`** Tentukanlah daftar kata yang cocok dengan regex tersebut:
   a. banyu, baru, ban
   b. batu, bayi, bar
   c. bau, bayi, bantu
   d. bayu, pintu, binti

8. Mana kata yang cocok dengan regex `a(ab)*a`
   a. abababa
   b. aaba
   c. aabbaa
   d. aba
   e. Aabababa

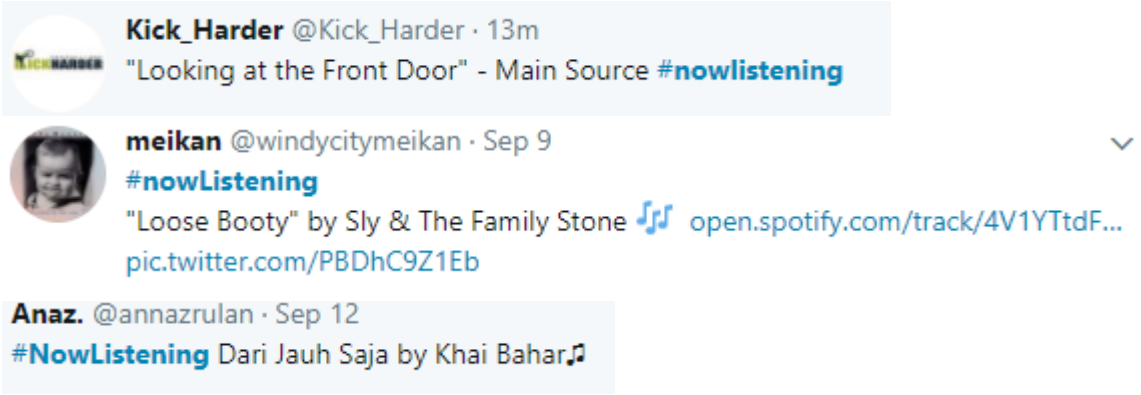
LATIHAN 2

Diberikan teks berikut ini, cobalah buat regex untuk mengekstrak semua alamat email yang tercantum. Jawaban latihan dapat dilihat pada https://www.regextester.com/99135

jurafsky@stanford.edu
jurafsky(at)cs.stanford.edu
jurafsky(at)cs.stanford.edu dot pk
jurafsky at csli dot stanford dot edu
jurafsky at csli dot stanford dot edu
jurafsky (at) csli dot stanford dot edu
jurafsky@stanford.edu OR jurafsky@cs.stanford.edu OR jurafsky@csli.stanford.edu as appropriate.

LATIHAN 3

Berdasarkan Tweet dengan hashtag #nowlistening, cobalah buat regex untuk mengekstrak judul dan penyanyi.



Berdasarkan Tweet dengan keyword “macet”, cobalah buat regex untuk mengekstrak lokasi kemacetan (asumsikan tweet sudah bersih dari spam dan tweet yang tidak relevan).

Latihan 4:
Buatlah regex yang sedapat mungkin dapat mengekstrak lokasi kemacetan berdasarkan tweet berikut (idealnya hanya satu regex untuk semua tweet):

- Yang mau lewat **gunung batu** mendingan cari jalan lain. **Macet** parah. @infobdg
- **Cikampek** macet pic.twitter.com/TWRGfR7FRR
- @e100ss macet di **depan kebun raya Purwodadi arah Surabaya**, cuaca panas pic.twitter.com/Lv9267aoWs
- **Jalan Raya Brangkal Mojokerto arah surabaya** **macet** total.. Belum tau kenapa @e100ss
- **Gerbang tol kejapanan** macet kenapa ya 🤔.. @e100ss

- **Arah grahadi** ditutup, jalanan macet
- Jangan lewat **Betro**, macet, kalian tak akan kuat. #macetpangkalkere #macetjalannyaemosiwarganya
- FYI mau kasih tau aja dari **bundaran HI sampe semanggi** macet banget jangan bawa mobil dah asli .
- Truk2 ini luar biasa memang , kemarin dari **Malang ke Sby** kena macet bbrp kali truk 4 biji mogok...ini balik ke Malang kena macet parah lg di **Porong** kena 2 truk mogok

Referensi:

1. Lesson: Regular Expressions https://docs.oracle.com/javase/tutorial/essential/regex/
2. Chapter 2 of An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, by Daniel Jurafsky and James H. Martin