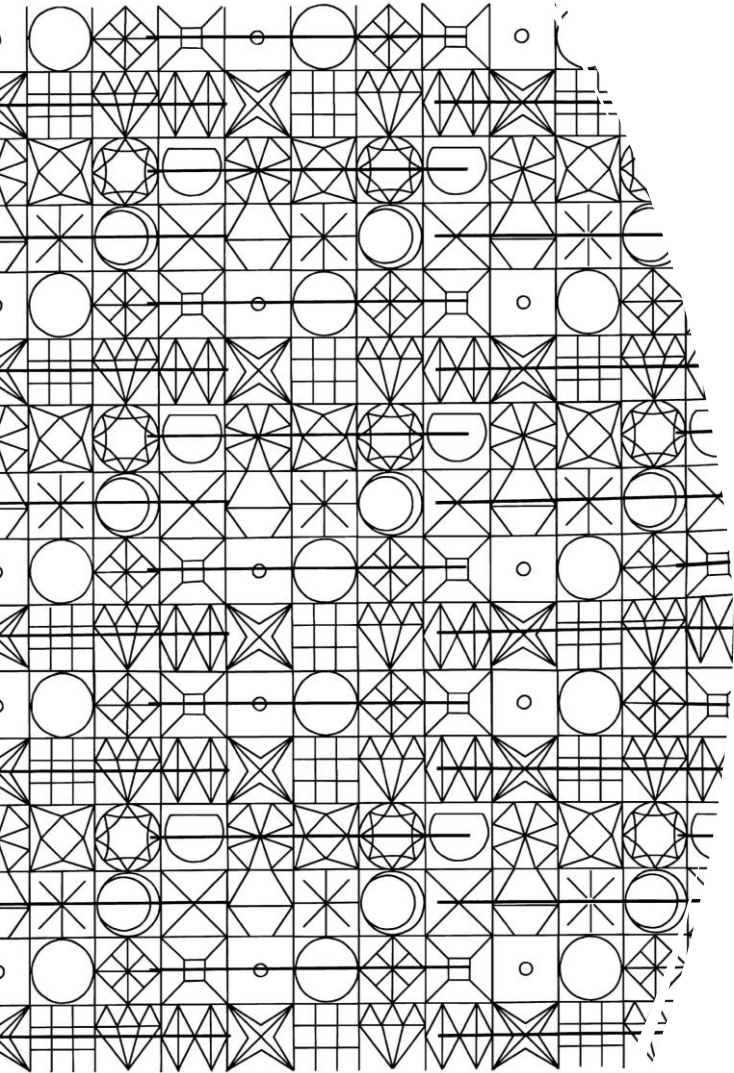


String Matching dengan Regular Expression

Masayu Leylia Khodra

Referensi:

Chapter 2 of *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, by Daniel Jurafsky and James H. Martin
15-211 *Fundamental Data Structures and Algorithms*, by Ananda Gunawardena

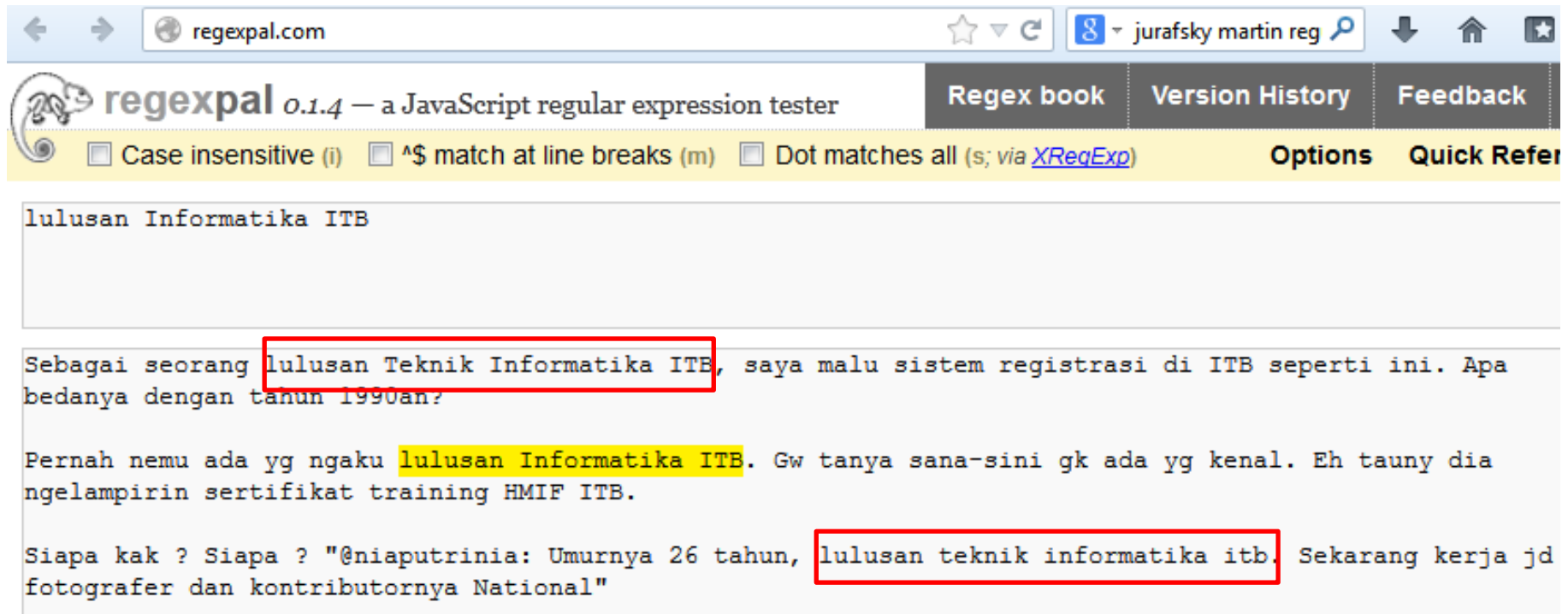


String Matching: Definisi

- Diberikan:
 1. T : teks (*text*), yaitu (*long*) *string* yang panjangnya n karakter
 2. P : *pattern*, yaitu *string* dengan panjang m karakter (asumsi $m \ll n$) yang akan dicari di dalam teks.

Carilah (*find* atau *locate*) di dalam teks yang bersesuaian dengan *pattern*.

Contoh 1: Exact Matching



The screenshot shows the regexpal.com website interface. The browser address bar displays 'regexpal.com'. The page title is 'regexpal 0.1.4 - a JavaScript regular expression tester'. The navigation menu includes 'Regex book', 'Version History', and 'Feedback'. The options bar shows three checked options: 'Case insensitive (i)', '^\$ match at line breaks (m)', and 'Dot matches all (s; via XRegExp)'. The search input field contains the text 'lulusan Informatika ITB'. Below the search field, the search results are displayed in a monospaced font. The first result is 'Sebagai seorang lulusan Teknik Informatika ITB, saya malu sistem registrasi di ITB seperti ini. Apa bedanya dengan tahun 1990an?'. The second result is 'Pernah nemu ada yg ngaku lulusan Informatika ITB. Gw tanya sana-sini gk ada yg kenal. Eh tauny dia ngelampirin sertifikat training HMIF ITB.'. The third result is 'Siapa kak ? Siapa ? "@niaputrinia: Umurnya 26 tahun, lulusan teknik informatika itb. Sekarang kerja jd fotografer dan kontributornya National"'. The search results are highlighted in yellow, and the phrases 'lulusan Teknik Informatika ITB' and 'lulusan teknik informatika itb.' are enclosed in red boxes.

regexpal.com

regexpal 0.1.4 - a JavaScript regular expression tester

Regex book Version History Feedback

Case insensitive (i) ^\$ match at line breaks (m) Dot matches all (s; via XRegExp) Options Quick Refer

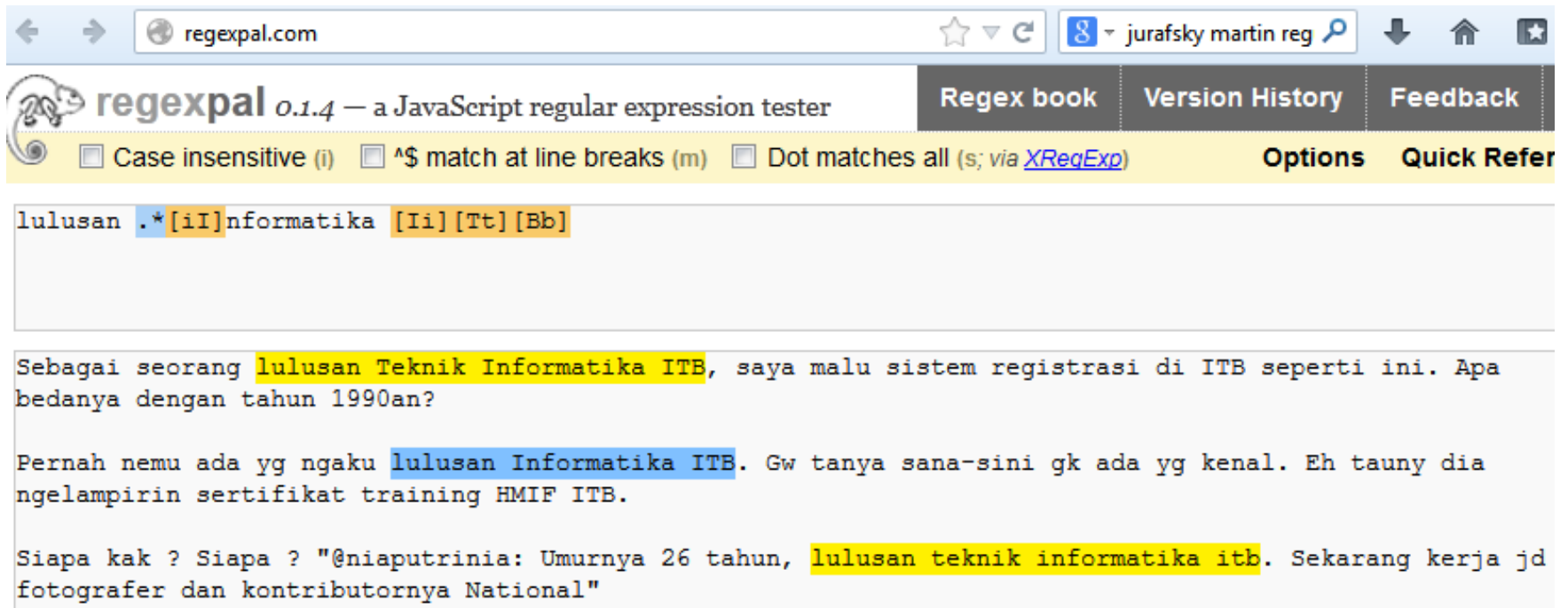
lulusan Informatika ITB

Sebagai seorang lulusan Teknik Informatika ITB, saya malu sistem registrasi di ITB seperti ini. Apa bedanya dengan tahun 1990an?

Pernah nemu ada yg ngaku lulusan Informatika ITB. Gw tanya sana-sini gk ada yg kenal. Eh tauny dia ngelampirin sertifikat training HMIF ITB.

Siapa kak ? Siapa ? "@niaputrinia: Umurnya 26 tahun, lulusan teknik informatika itb. Sekarang kerja jd fotografer dan kontributornya National"

Contoh 2: Regex Matching



The screenshot shows the website `regexpal.com` with the following details:

- Browser address bar: `regexpal.com`
- Page title: `regexpal 0.1.4 - a JavaScript regular expression tester`
- Navigation links: `Regex book`, `Version History`, `Feedback`
- Options: Case insensitive (i), ^\$ match at line breaks (m), Dot matches all (s; via [XRegExp](#))
- Buttons: `Options`, `Quick Refer`
- Input field: `lulusan .* [iI]nformatika [Ii] [Tt] [Bb]`
- Output field:

Sebagai seorang **lulusan Teknik Informatika ITB**, saya malu sistem registrasi di ITB seperti ini. Apa bedanya dengan tahun 1990an?

Pernah nemu ada yg ngaku **lulusan Informatika ITB**. Gw tanya sana-sini gk ada yg kenal. Eh tauny dia ngelampirin sertifikat training HMIF ITB.

Siapa kak ? Siapa ? "@niaputrinia: Umurnya 26 tahun, **lulusan teknik informatika itb**. Sekarang kerja jd fotografer dan kontributornya National"

Notasi Umum Regex

Regex book	Version History	Feedback	Blog
Options		Quick Reference	
.	Any character except newline.		
\.	A period (and so on for *, \ (, \\, etc.)		
^	The start of the string.		
\$	The end of the string.		
\d,\w,\s	A digit, word character [A-Za-z0-9_], or whitespace.		
\D,\W,\S	Anything except a digit, word character, or whitespace.		
[abc]	Character a, b, or c.		
[a-z]	a through z.		
[^abc]	Any character except a, b, or c.		
aa bb	Either aa or bb.		
?	Zero or one of the preceding element.		
*	Zero or more of the preceding element.		
+	One or more of the preceding element.		
{n}	Exactly n of the preceding element.		
{n,}	n or more of the preceding element.		
{m,n}	Between m and n of the preceding element.		
??,*?,+?, {n}?, etc.	Same as above, but as few as possible.		
(expr)	Capture expr for use with \1, etc.		
(?:expr)	Non-capturing group.		
(?=expr)	Followed by expr.		
(?!expr)	Not followed by expr.		

[Near-complete reference](#)

```
/[bcr]at/g
```

Test String

```
bat rat cat
```

```
/[^bcr]at/g
```

Test String

```
bat rat cat hat
```

```
/ke-[1-3]/g
```

Test String

```
Peringkat ke-1 dan ke-5
```

```
/[^a-z]/g
```

Test String

```
Huruf Besar Saja
```

```
/(ade)/g
```

Test String

```
aderay bade
```

```
/[ade]/g
```

Test String

```
aderay bade
```

Contoh 2: Regex

The screenshot shows the website `regexpal.com` with the following content:

Browser address bar: `regexpal.com`

Page title: `regexpal 0.1.4 — a JavaScript regular expression tester`

Navigation links: `Regex book`, `Version History`, `Feedback`

Options: Case insensitive (i) ^\$ match at line breaks (m) Dot matches all (s; via [XRegExp](#))

Input text: `lulusan .*[Ii]nformatika [Ii][Tt][Bb]`

Tested text 1: `Sebagai seorang lulusan Teknik Informatika ITB, saya malu bedaanya dengan tahun 1990an?`

Tested text 2: `Pernah nemu ada yg ngaku lulusan Informatika ITB. Gw tanya ngelampirin sertifikat training HMIF ITB.`

Tested text 3: `Siapa kak ? Siapa ? "@niaputrinia: Umurnya 26 tahun, lulusan fotografer dan kontributornya National"`

Reference table:

<code>.</code>	Any character except newline.
<code>\.</code>	A period (and so on for <code>*</code> , <code>\(</code> , <code>\\</code> , etc.)
<code>^</code>	The start of the string.
<code>\$</code>	The end of the string.
<code>\d,\w,\s</code>	A digit, word character [<code>A-Za-z0-9_</code>], or whitespace.
<code>\D,\W,\S</code>	Anything except a digit, word character, or whitespace.
<code>[abc]</code>	Character a, b, or c.
<code>[a-z]</code>	a through z.
<code>[^abc]</code>	Any character except a, b, or c.
<code>aa bb</code>	Either aa or bb.
<code>?</code>	Zero or one of the preceding element.
<code>*</code>	Zero or more of the preceding element.
<code>+</code>	One or more of the preceding element.
<code>{n}</code>	Exactly <i>n</i> of the preceding element.
<code>{n,}</code>	<i>n</i> or more of the preceding element.
<code>{m,n}</code>	Between <i>m</i> and <i>n</i> of the preceding element.
<code>??,*?,+?</code>	Same as above, but as few as possible.
<code>{n}?</code> , etc.	
<code>(expr)</code>	Capture <i>expr</i> for use with <code>\1</code> , etc.
<code>(?:expr)</code>	Non-capturing group.
<code>(?=expr)</code>	Followed by <i>expr</i> .
<code>(?!expr)</code>	Not followed by <i>expr</i> .

[Near-complete reference](#)

Basic Regular Expression Patterns

brackets [] : disjunction

RE	Match	Example Patterns
/ [wW] oodchuck /	Woodchuck or woodchuck	“ <u>W</u> oodchuck”
/ [abc] /	‘a’, ‘b’, or ‘c’	“In uo <u>m</u> ini, in soldat <u>i</u> ”
/ [1234567890] /	any digit	“plenty of <u>7</u> to 5”

Brackets [] ditambah garis sambung: range

RE	Match	Example Patterns Matched
/ [A-Z] /	an uppercase letter	“we should call it ‘ <u>D</u> renched Blossoms””
/ [a-z] /	a lowercase letter	“ <u>m</u> y beans were impatient to be hoed!”
/ [0-9] /	a single digit	“Chapter <u>1</u> : Down the Rabbit Hole”

Basic Regular Expression Patterns

- caret ^ : negasi

RE	Match (single characters)	Example Patterns Matched
[^A-Z]	not an uppercase letter	“O <u>y</u> fn pripetchik”
[^Ss]	neither ‘S’ nor ‘s’	“ <u>I</u> have no exquisite reason for’t”
[^\.]	not a period	“ <u>o</u> ur resident Djinn”
[e^]	either ‘e’ or ‘^’	“look up <u>_</u> now”
a^b	the pattern ‘a^b’	“look up <u>a</u> <u>b</u> now”

- Tanda tanya ? : bisa ada bisa tidak

RE	Match	Example Patterns Matched
woodchucks?	woodchuck or woodchucks	“ <u>woodchuck</u> ”
colou?r	color or colour	“ <u>colour</u> ”

- Titik: . any character

RE	Match	Example Patterns
/beg.n/	any character between <i>beg</i> and <i>n</i>	<u>begin</u> , <u>beg'n</u> , <u>begun</u>

Regex Kata berawal Huruf Kapital

```
/[A-Z][a-z]*/g
```

Test String

Berkaitan dengan sidang tersebut, Sekretaris Jenderal Partai memastikan bahwa Setya Novanto tidak menghadiri sidang perdar karena sakit. Akibat sakit pula, Ketua Umum Partai Golkar itu pemeriksaan KPK sebagai tersangka kasus e-KTP.

[A-Z][a-z]* : Alfabet huruf besar yang dilanjutkan dengan nol atau banyak huruf kecil

Notasi Regex: Contoh

```
/.udi/g
```

Test String

```
udi Budi rudi yudi udi
```

```
/\.\.\./g
```

Test String

```
Kalimat ini panjang ... dipotong dan selesai
```

Metacharacter titik "." menyatakan karakter apapun (kiri). Gunakanlah backslash "\" untuk metacharacter.

```
/\s/g
```

Test String

```
kata yang dipisahkan spasi
```

```
/rekans?/g
```

Test String

```
halo rek rekan dan rekans
```

```
/[a-z]+\d+/g
```

Test String

```
b24 24b saya123
```

Notasi Regex: Contoh

```
/(ha)+|(he)+|(hi)+/g
```

Test String

```
haha hehehehe hoho hihi
```

```
/(h[aei])+/g
```

Test String

```
haha hehehehe hoho hihi
```

```
/!{2,}/g
```

Test String

```
saya! suka!!!!
```

Contoh 3: Regex for Email

regexpal.com

regexpal 0.1.4 — a JavaScript regular expression tester

Case insensitive (i) ^\$ match at line breaks (m) Dot matches all (s; via XRegExp)


Tentukan regexnya untuk semua email yang diwarnai

```
test.txt - obfuscate('stanford.edu','jurafsky' - jurafsky@stanford.edu;
test.txt - jurafsky@stanford.edu - jurafsky@stanford.edu;
test.txt - jurafsky@stanford.edu - jurafsky@stanford.edu;
test.txt - jurafsky(at)cs.stanford.edu - jurafsky@cs.stanford.edu;
test.txt - jurafsky at csli dot stanford dot edu - jurafsky@csli.stanford.edu;
test.txt - jurafsky@stanford.edu - jurafsky@stanford.edu;
test.txt - jurafsky@cs.stanford.edu - jurafsky@cs.stanford.edu;
test.txt - jurafsky@csli.stanford.edu - jurafsky@csli.stanford.edu;
test.txt - 650-723-0293 - 650-723-0293;
test.txt - (650) 723-0293 - 650-723-0293;
test.txt - 650-723-0293 - 650-723-0293;
test.txt - 650&thinsp;723&thinsp;0293 - 650-723-0293;
test.txt - 650-723-0293 - 650-723-0293;
```

.	Any character
\.	A period
^	The start
\$	The end
\d,\w,\s	A digit, word, or space
\D,\W,\S	Anything but digit, word, or space
[abc]	Character class
[a-z]	a through z
[^abc]	Any character except a, b, or c
aa bb	Either a followed by a or b followed by b
?	Zero or one
*	Zero or more
+	One or more
{n}	Exactly n
{n,}	n or more
{m,n}	Between m and n
??,*?,+?	Same as ?, *, or +, but non-greedy
{n}?, etc.	Same as {n}, but non-greedy
(expr)	Capture the text matched by expr
(?:expr)	Non-capturing group
(?=expr)	Lookahead
(?!expr)	Not followed by

Contoh 4: Regex for Phone Number

regexpal.com

 **regexpal** 0.1.4 — a JavaScript regular expression tester

Case insensitive (i) ^\$ match at line breaks (m) Dot matches all (s; via [XRegExp](#))

```
(\ (?\d{3}\ )?[- ]+\d{4})
```

```
test.txt - obfuscate('stanford.edu','jurafsky' - jurafsky@stanford.edu;
test.txt - jurafsky@stanford.edu - jurafsky@stanford.edu;
test.txt - jurafsky@stanford.edu - jurafsky@stanford.edu;
test.txt - jurafsky(at)cs.stanford.edu - jurafsky@cs.stanford.edu;
test.txt - jurafsky at csli dot stanford dot edu - jurafsky@csli.stanford.edu;
test.txt - jurafsky@stanford.edu - jurafsky@stanford.edu;
test.txt - jurafsky@cs.stanford.edu - jurafsky@cs.stanford.edu;
test.txt - jurafsky@csli.stanford.edu - jurafsky@csli.stanford.edu;
test.txt - 650-723-0293 - 650-723-0293;
test.txt - (650) 723-0293 - 650-723-0293;
test.txt - 650-723-0293 - 650-723-0293;
test.txt - 650&thinsp;723&thinsp;0293 - 650-723-0293;
test.txt - 650-723-0293 - 650-723-0293;
```

Knowledge check Regex

Pelajarilah modul regex: <https://docs.google.com/document/d/1ls6h1A6m-Zhzw6e5eriwMNUAG0D1iwL-eVmVMS2XQoc/edit?usp=sharing>

Kerjakanlah Latihan 1-3 secara mandiri (tidak dikumpulkan).

Untuk Latihan 4, gunakanlah <https://www.regexpal.com/> (tidak dikumpulkan)

Regex di Java

Case insensitive (i) ^\$ match at line breaks (m) Dot matches all (s; via [XRegExp](#))

`\d{2}\.\d{2}`

```
#lalinBDG 09.16 : yg mau ke jln sudirman dsk, jg  
@infobdg: #lalinBDG 08.23 : Macet (lagi) rancaek  
#suaraBDG via @dionmudjenan: Hati-hati jembatan p  
RT @quinsymegamira: leuwipanjang banyak anak jal
```

```
public static void extraction(Pattern myPattern, String str) {  
    String extract;  
  
    Matcher m;  
    m = myPattern.matcher(str);  
    while(m.find()) {  
        extract = m.group();  
        System.out.println(extract);  
    }  
}
```

```
public static void main(String[] args) {  
    resources.extraction(Pattern.compile("\\d{2}\\.\d{2}"), "@infobdg: #
```

it %

Debugger Console x veritransTools (run) x

run:

08.23

BUILD SUCCESSFUL (total time: 0 seconds)

Regex using Python

- `re.compile(pattern, flags=0)`
Compile a regular expression pattern into a [regular expression object](#), which can be used for matching using its [match\(\)](#), [search\(\)](#) and other methods, described below.
- `Pattern.search(string[, pos[, endpos]])`
Scan through *string* looking for the **first location** where this regular expression produces a match, and return a corresponding [match object](#). Return None if no position in the string matches the pattern; note that this is different from finding a zero-length match at some point in the string.
The optional second parameter *pos* gives an index in the string where the search is to start; it defaults to 0. The optional parameter *endpos* limits how far the string will be searched;

Regex using Python

```
import re

#Compile a regular expression pattern into a regular expression object
pattern = re.compile(r"(\d{4})")

#Scan through str looking for 1st loc where this regex produces a match,
#and return a corresponding match object.
pattern.search("17 August 1945 - 2022")
```

```
<re.Match object; span=(11, 15), match='1945'>
```

`\d{4}`: digit characters exactly 4 characters

Regex book	Version History	Feedback	Blog
Options		Quick Reference	
.	Any character except newline.		🔗 ✖
\.	A period (and so on for *, \ (, \\, etc.)		
^	The start of the string.		
\$	The end of the string.		
\d,\w,\s	A digit, word character [A-Za-z0-9_], or whitespace.		
\D,\W,\S	Anything except a digit, word character, or whitespace.		
[abc]	Character a, b, or c.		
[a-z]	a through z.		
[^abc]	Any character except a, b, or c.		
aa bb	Either aa or bb.		
?	Zero or one of the preceding element.		
*	Zero or more of the preceding element.		
+	One or more of the preceding element.		
{n}	Exactly n of the preceding element.		
{n,}	n or more of the preceding element.		
{m,n}	Between m and n of the preceding element.		
??,*?,+?, {n}?, etc.	Same as above, but as few as possible.		
(expr)	Capture expr for use with \1, etc.		
(?:expr)	Non-capturing group.		
(?=expr)	Followed by expr.		
(?!expr)	Not followed by expr.		
Near-complete reference			

Regex using Python

- `Pattern.match(string[, pos[, endpos]])`

If zero or more characters at the **beginning of str** match this regular expression, return a corresponding [match object](#).

Return None if the string does not match the pattern; note that this is different from a zero-length match.

```
pattern = re.compile(r"(\d{4})")
m=pattern.match("1945 - 2022")
m.groups() # The entire match
```

```
('1945',)
```

Regex using Python

- `re.findall(pattern, string, flags=0)`
Return all non-overlapping matches of *pattern* in *string*, as a **list of strings or tuples**. The *string* is scanned left-to-right, and matches are returned in the order found. Empty matches are included in the result.

```
text="17 Agustus 1945 - 2022"  
re.findall(r"(\d{4})", text)
```

```
['1945', '2022']
```

```
#extract adverb  
text = "He was carefully disguised but captured quickly by police."  
re.findall(r"\w+ly\b", text)
```

```
['carefully', 'quickly']
```

Regex using Python

```
In [32]: pattern = re.compile(r"(\d{4})")  
pattern.search("17 Agustus 1945 - 2022") #Scan through string looking for the first location where
```

```
Out[32]: <re.Match object; span=(11, 15), match='1945'>
```

```
In [16]: pattern.match("17 Agustus 1945") #If zero or more characters at the beginning of string match the
```

```
In [34]: m=pattern.match("1945 - 2022")  
m.groups() # The entire match
```

```
Out[34]: ('1945',)
```

```
In [36]: text="17 Agustus 1945 - 2022"  
re.findall(r"(\d{4})", text)
```

```
Out[36]: ['1945', '2022']
```

```
In [35]: #extract adverb  
text = "He was carefully disguised but captured quickly by police."  
re.findall(r"\w+ly\b", text)
```

```
Out[35]: ['carefully', 'quickly']
```

<https://docs.python.org/3/library/re.html>

Baca: <https://www.guru99.com/python-regular-expressions-complete-tutorial.html>

Eliza, simple pattern-based chatbot

ELIZA uses **pattern matching** to recognize phrases like “I need **X**” and translate them into suitable outputs like
“What does wanting **X** have to do with this discussion ?”
“What would it mean to you if you got **X**?”.

Eliza

```
ELIZA: How do you do. Please tell me your problem.
YOU: I am unhappy
ELIZA: Do you think coming here will help you not to be
unhappy?
YOU: I need some help that much seems certain
ELIZA: What does wanting some help have to do with this
discussion?
YOU: Perhaps I could learn to get along with my mother
ELIZA: Your mother?
YOU: Yes, my mother
ELIZA: I understand.
YOU: My mother takes care of me
ELIZA: Tell me more about your family.
YOU: My mother takes care of me
ELIZA: Your mother?
YOU: Yes
ELIZA: I see.
```

Talk

Reset

<https://www.masswerk.at/elizabot/>
<http://psych.fullerton.edu/mbirnbaum/psych101/eliza.htm>

Weizenbaum, J. (1966). ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1), 36-45.

Eliza-like Chatbot using Regex

```
import re

def findAnswer(teks):
    t=teks.upper()
    knowledge_base={".* I'M (DEPRESSED|SAD) .*":"I AM SORRY TO HEAR YOU ARE X",
                    ".* MY (.*)" : "YOUR X",
                    ".* ALL .*" : "IN WHAT WAY",
                    ".* ALWAYS .*" : "CAN YOU THINK OF A SPECIFIC EXAMPLE"
    }
    notFound="Can you repeat your question ?"
    for key in knowledge_base:
        m=re.match(key, t)
        if m:
            answer=knowledge_base[key]
            len_groups = len(m.groups())
            if (len_groups==0):
                return answer
            else:
                X=m.group(1)
                answer=answer.replace("X",X)
                if " ME " in answer:
                    answer=answer.replace(" ME ", " YOU ")
                return answer
    return notFound
```

```
listQ=["Men are all alike",
        "They're always bagging us about something or other",
        "Well, my boyfriend made me come here",
        "He says I'm depressed much of the time."
    ]

for q in listQ:
    print("User: ",q)
    print("Bot: ",findAnswer(q))
```

```
User: Men are all alike
Bot: IN WHAT WAY
User: They're always bagging us about something or other
Bot: CAN YOU THINK OF A SPECIFIC EXAMPLE
User: Well, my boyfriend made me come here
Bot: YOUR BOYFRIEND MADE YOU COME HERE
User: He says I'm depressed much of the time.
Bot: I AM SORRY TO HEAR YOU ARE DEPRESSED
```