

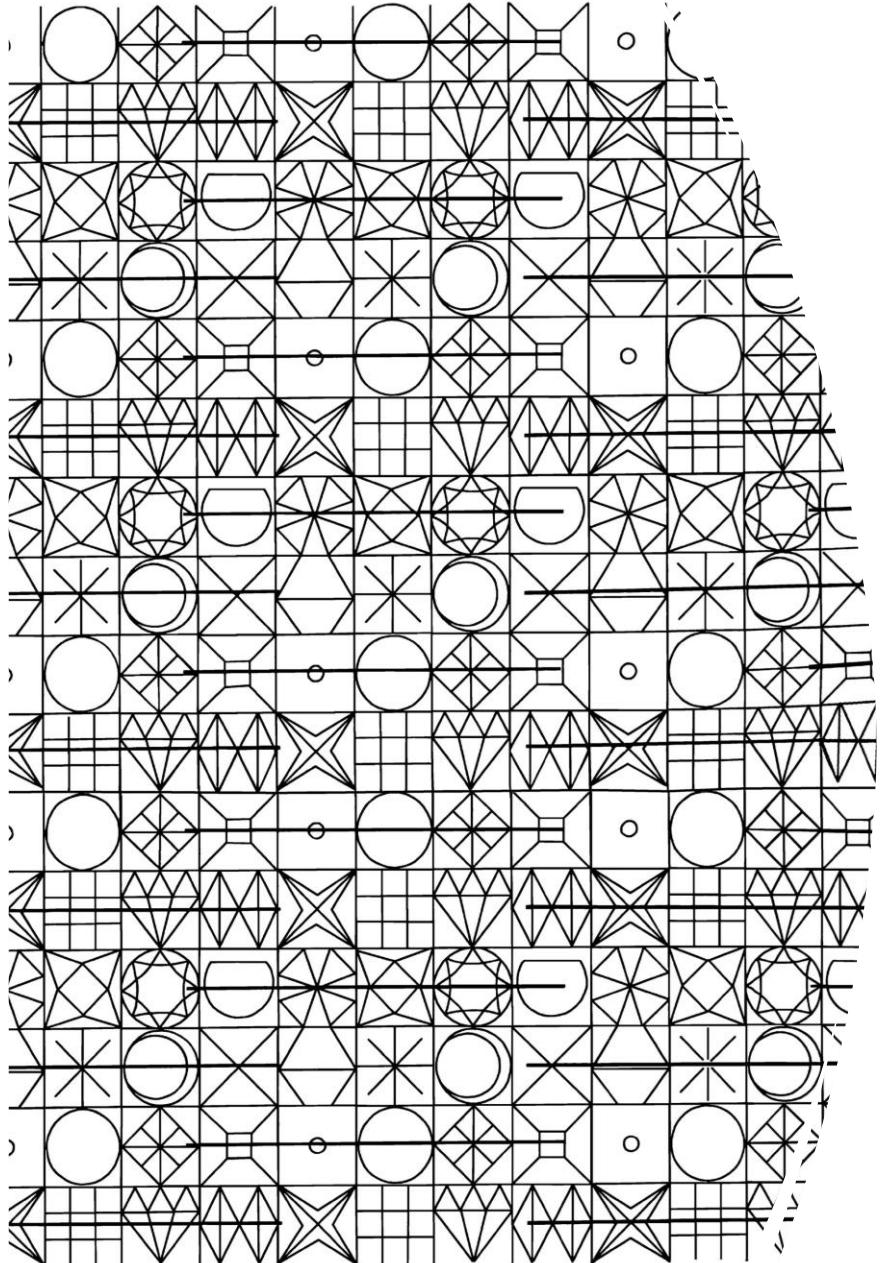
String Matching dengan Regular Expression

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 T yang bersesuaian dengan *pattern* P .

Contoh 1: Exact Matching

The screenshot shows the regexpal.com interface. The URL bar has "jurafsky martin reg" with a search icon. The main header says "regexpal 0.1.4 — a JavaScript regular expression tester". Below it are three checkboxes: "Case insensitive (i)", "\$ match at line breaks (m)", and "Dot matches all (s; via XRegExp)". To the right are "Regex book", "Version History", and "Feedback" links. Further right are "Options" and "Quick Refer" buttons. A yellow bar contains the text "lulusan Informatika ITB". Below this is a text area with Indonesian text. The first sentence contains the phrase "lulusan Teknik Informatika ITB", which is highlighted with a red rectangle. The second sentence contains the phrase "lulusan Informatika ITB", which is highlighted with a yellow rectangle. The third sentence contains the phrase "lulusan teknik informatika itb", which is highlighted with a red rectangle.

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 a web browser window for [regexpal.com](https://www.regexpal.com). The search bar contains the query "jurafsky martin reg". Below the search bar, the page title is "regexpal 0.1.4 — a JavaScript regular expression tester". A navigation bar includes links for "Regex book", "Version History", and "Feedback". Below the navigation bar are three checkboxes: "Case insensitive (i)", "Match at line breaks (m)", and "Dot matches all (s; via [XRegExp](#))". To the right of these checkboxes are "Options" and "Quick Refer" buttons. The main content area displays a text input field containing the regex pattern "lulusan .*[ii]nformatika [Ii][Tt][Bb]". Below this, there are two text snippets. The first snippet is: "Sebagai seorang lulusan Teknik Informatika ITB, saya malu sistem registrasi di ITB seperti ini. Apa bedanya dengan tahun 1990an?". The second snippet 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.". Both snippets have portions highlighted in yellow, specifically the word "lulusan" and the entire regex pattern from the input field.

lulusan .*[ii]nformatika [Ii][Tt][Bb]

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 a web browser window with the URL `regexpal.com` in the address bar. The search query is `jurafsky martin reg`. The results page for `jurafsky martin reg` is displayed, featuring the `regexpal 0.1.4 – a JavaScript regular expression tester` logo. The search results area contains three snippets of text:

- `lulusan .*[Ii]nformatika [Ii][Tt][Bb]`
- `Sebagai seorang lulusan Teknik Informatika ITB, saya malu bedanya dengan tahun 1990an?`
- `Pernah nemu ada yg ngaku lulusan Informatika ITB. Gw tanpa ngelampirin sertifikat training HMIF ITB.`
- `Siapa kak ? Siapa ? "@niaputrinia: Umurnya 26 tahun, lulusan fotografer dan kontributornya National"`

The sidebar on the right contains a "Quick Reference" table for regular expression symbols:

.	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](#)

Basic Regular Expression Patterns

Brackets [] : disjunction

RE	Match	Example Patterns
/ [wW] oodchuck /	Woodchuck or woodchuck	“ <u>Woodchuck</u> ”
/ [abc] /	‘a’, ‘b’, or ‘c’	“In uomini, in soldati”
/ [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>Drenched Blossoms</u> ’”
/ [a-z] /	a lowercase letter	“ <u>my</u> beans were impatient to be hoed!”
/ [0-9] /	a single digit	“Chapter <u>1</u> : Down the Rabbit Hole”

Basic Regular Expression Patterns

- caret \wedge : negasi

RE	Match (single characters)	Example Patterns Matched
$[\wedge A-Z]$	not an uppercase letter	“Oyfn pripetchik”
$[\wedge Ss]$	neither ‘S’ nor ‘s’	“I have no exquisite reason for’t”
$[\wedge \cdot]$	not a period	“our resident Djinn”
$[e^\wedge]$	either ‘e’ or ‘^’	“look up <u>^</u> now”
$a^\wedge b$	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 perdari 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 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&thinsp;723&thinsp;0293 - 650-723-0293;
test.txt - 650-723-0293 - 650-723-0293;
```

Any character e
A period (and s
The start of the
The end of the
A digit, word ch:
Anything except
Character a, b,
a through z.
Any character e
Either aa or bb.
Zero or one of t
Zero or more of
One or more of
Exactly n of the
n or more of the
Between *m* and
??,*?+,?
{*n*} ?, etc.
(expr) Capture expr fo
(?:expr) Non-capturing g
(?:=expr) Followed by exp
(?!expr) Not followed by

Contoh 4: Regex for Phone Number

The screenshot shows the regexpal.com website, version 0.1.4, which is a JavaScript regular expression tester. The URL bar at the top contains the site's name. Below the header, there is a logo of a cartoon character and the text "regexpal 0.1.4 – a JavaScript regular expression tester". There are three checkboxes with labels: "Case insensitive (i)", "^\\$ match at line breaks (m)", and "Dot matches all (s; via XRegExp)". The main input field contains the regular expression: `(\(? \d{3} \)? [-])+\d{4}`. Below the input field is a large text area containing several lines of test data from a file named "test.txt". The regular expression has highlighted matches in yellow and blue, indicating successful matches against the test data. The test data includes various email addresses and phone numbers.

```
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-Zhw6e5eriwMNUAG0D1iwL-eVmVMS2XQoc/edit?usp=sharing>

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

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

Regex di dalam Bahasa Java

The screenshot shows a Java development environment with two main panes. The left pane displays a regular expression tester titled "regexpal 0.1.4 – a JavaScript regular expression tester". It has a search bar containing "\d{2}.\.\d{2}" and a list of matches from a tweet stream. The right pane shows a Java code editor with a class containing methods for extraction and main. A red box highlights the regex pattern in the main method's argument. Below the code editor is a terminal window showing the execution of the code and its output.

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

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 Agustus 1945 - 2022")
```

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

\d{4}: digit characters exactly 4 characters

Options		Quick Reference
19	t t	ntr
.	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
1

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 them
1

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

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.

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

Welcome to

EEEEEE	LL	III	ZZZZZ	AAAAA
EE	LL	II	ZZ	AA AA
EEEEEE	LL	II	ZZZ	AAAAAAA
EE	LL	II	ZZ	AA AA
EEEEEE	LLLLL	III	ZZZZZ	AA AA

Eliza is a mock Rogerian psychotherapist.

The original program was described by Joseph Weizenbaum in 1966.

This implementation by Norbert Landsteiner 2005.

ELIZA: Is something troubling you ?

YOU: Men are all alike.

ELIZA: What is the connection, do you suppose ?

YOU: They're always bugging us about something or other.

ELIZA: Can you think of a specific example ?

YOU: Well, my boyfriend made me come here.

ELIZA: Is it important to you that your boyfriend made you come here ?

YOU: He says I'm depressed much of the time.

ELIZA: I am sorry to hear that you are depressed.

YOU: It's true. I am unhappy.

ELIZA: Can you explain what made you unhappy ?

YOU:

Eliza-like Chatbot using Regex

```
import re

def findAnswer(tekst):
    t=tekst.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
```