Regular Expressions

Alfan Farizki Wicaksono
Universitas Indonesia

Source: https://docs.python.org/3/howto/regex.html

Regular Expressions

- Sebuah “bahasa pemrograman kecil” yang ada di dalam python.
- Biasanya digunakan untuk proses matching string dengan pola-pola tertentu:
  - Does this string match the pattern?
  - Is there a match for the pattern anywhere in this string?
  - How to split a string with a delimiter that matches a pattern?
- Bermanfaat untuk kegiatan data science yang melibatkan data tekstual (text mining & NLP).

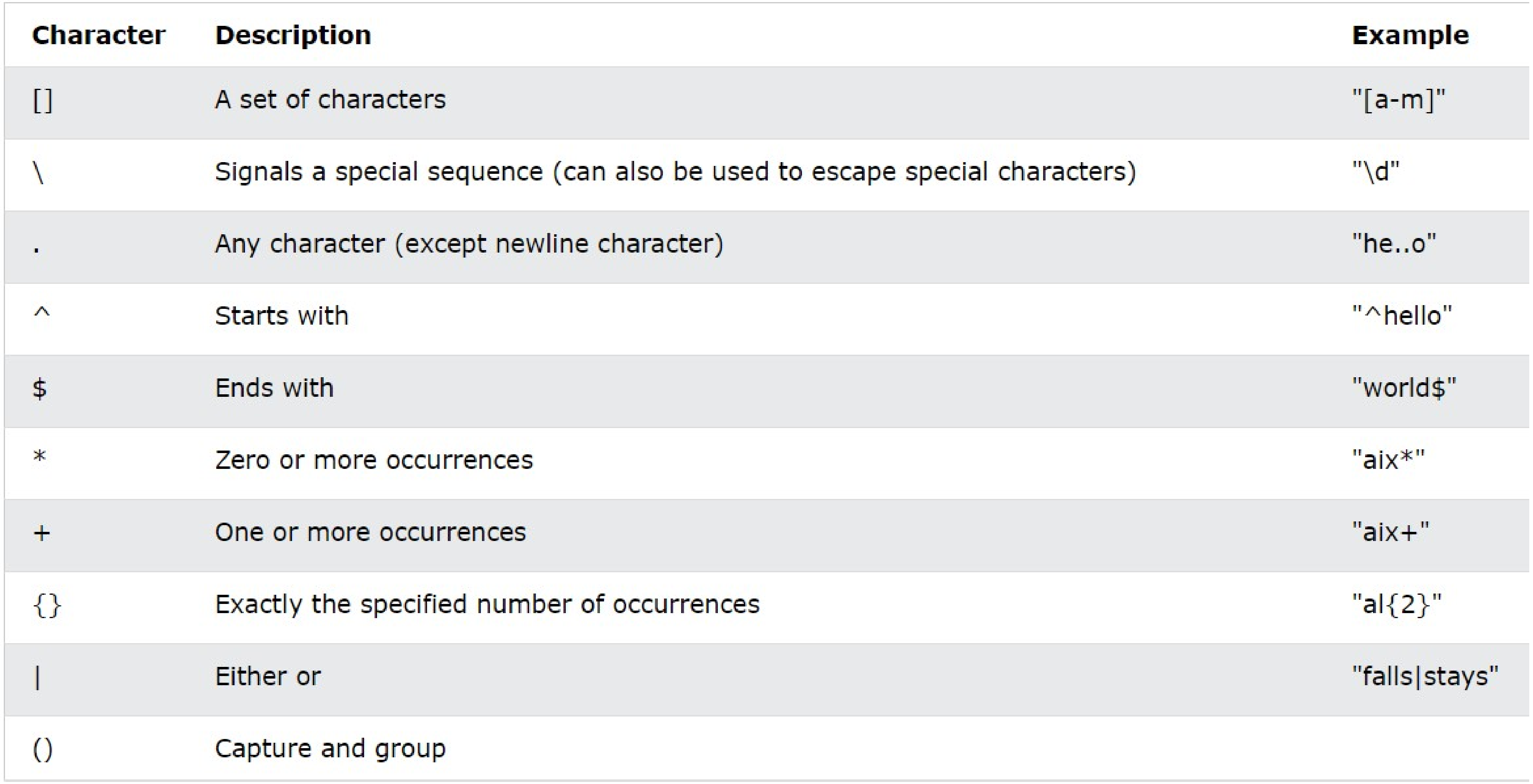
Matching Characters

```
import re


text = "proklamasi Indonesia tahun 1945"
match = re.search("masi Ind", text)


if match:
  print("pattern found")
else:
  print("pattern not found")
```

Yang akan dicetak

Credit: https://www.w3schools.com/python/python_regex.asp

Metacharacters



Character	Description	Example
[]	A set of characters	"[a-m]"
\	Signals a special sequence (can also be used to escape special characters)	"\d"
.	Any character (except newline character)	"he..o"
^	Starts with	"^hello"
$	Ends with	"world$"
*	Zero or more occurrences	"aix*"
+	One or more occurrences	"aix+"
{}	Exactly the specified number of occurrences	"al{2}"
|	Either or	"falls|stays"
()	Capture and group	

Contoh penggunaan metacharacters

```
match = re.search("sia$", "indonesia dan malaysia")
# found, span=(19, 22), match='sia

match = re.search("^malay", "indonesia dan malaysia")
# not found

match = re.search("^indo", "indonesia dan malaysia")
# found, span=(0, 4), match='indo'

match = re.search("dan..", "indonesia dan malaysia")
# found, span=(10, 15), match='dan m'
```

Contoh penggunaan metacharacters

```
match = re.search("[a-d]", "indonesia dan malaysia")
# found, span=(2, 3), match='d'

match = re.search("[abcd]", "indonesia dan malaysia")
# found, span=(2, 3), match='d'

match = re.search("sia*", "indonesiaaa")
# found, span=(6, 11), match='siaaa'

match = re.search("six*", "indonesiaaa")
# found, span=(6, 8), match='si'
```

Contoh penggunaan metacharacters

```
match = re.search("six+", "indonesiaaa")
# not found

match = re.search("malay|indone", "indonesiaaa")
# found, span=(0, 6), match='indone'

match = re.search("\d\d\d", "indonesia1945yes")
# found, span=(9, 12), match='194'

match = re.search("@\w+@", "@@@indonesia1945@@@")
# found, span=(2, 17), match='@indonesia1945@'
```

Contoh penggunaan metacharacters

Catatan!

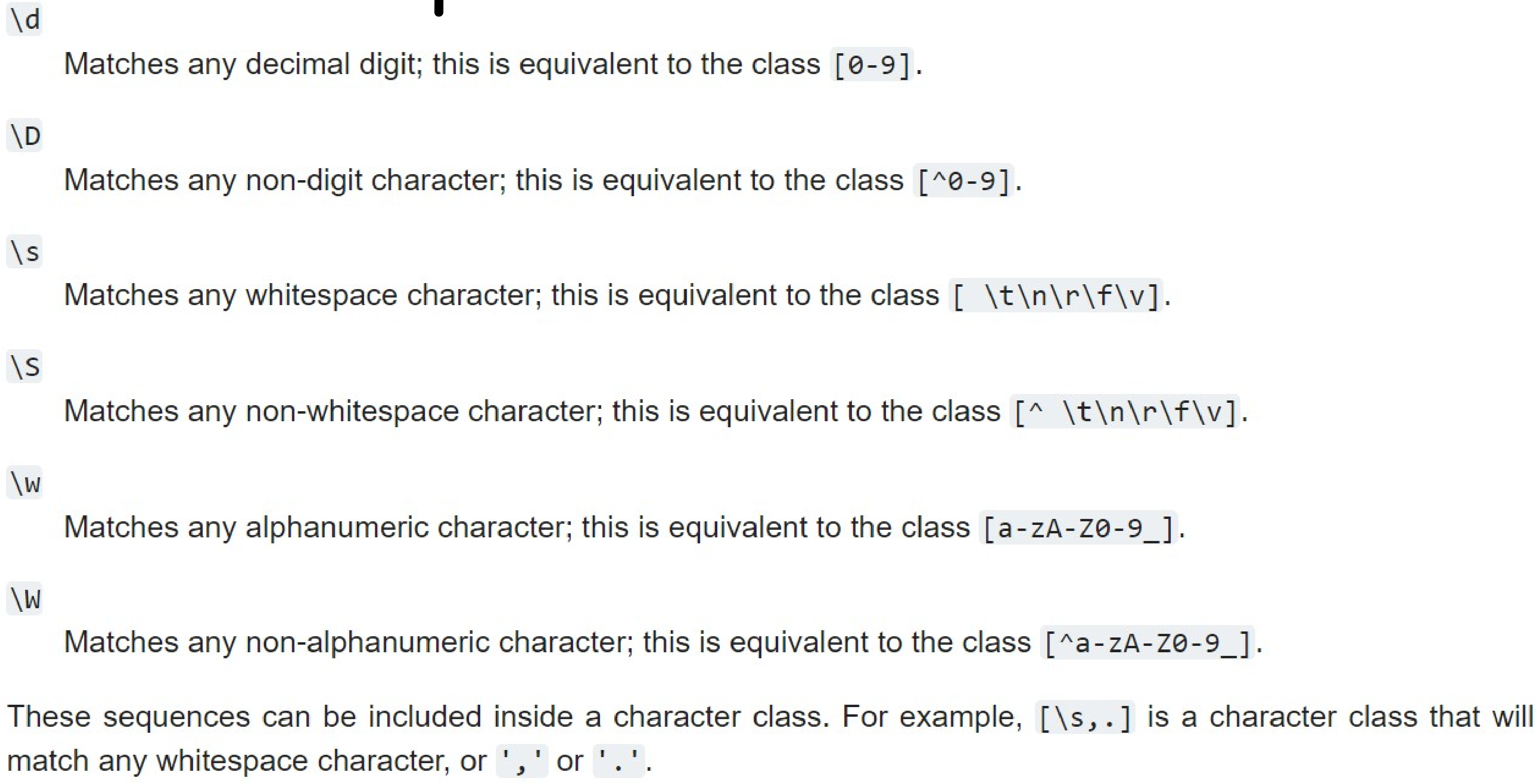
Complementing sets!
Match apapun, selain 9, 4, dan 5

```
match = re.search("[^945]", "indonesia^1945")
# found, span=(0, 1), match='i'
```

Match 9, 4, 5, atau karakter ^

```
match = re.search("[945^]", "indonesia^1945")
# found, span=(9, 10), match='^'
```

Special Characters



`\d`

Matches any decimal digit; this is equivalent to the class `[0-9]`.

`\D`

Matches any non-digit character; this is equivalent to the class `[^0-9]`.

`\s`

Matches any whitespace character; this is equivalent to the class `[ \t\n\r\f\v]`.

`\S`

Matches any non-whitespace character; this is equivalent to the class `[^ \t\n\r\f\v]`.

`\w`

Matches any alphanumeric character; this is equivalent to the class `[a-zA-Z0-9_]`.

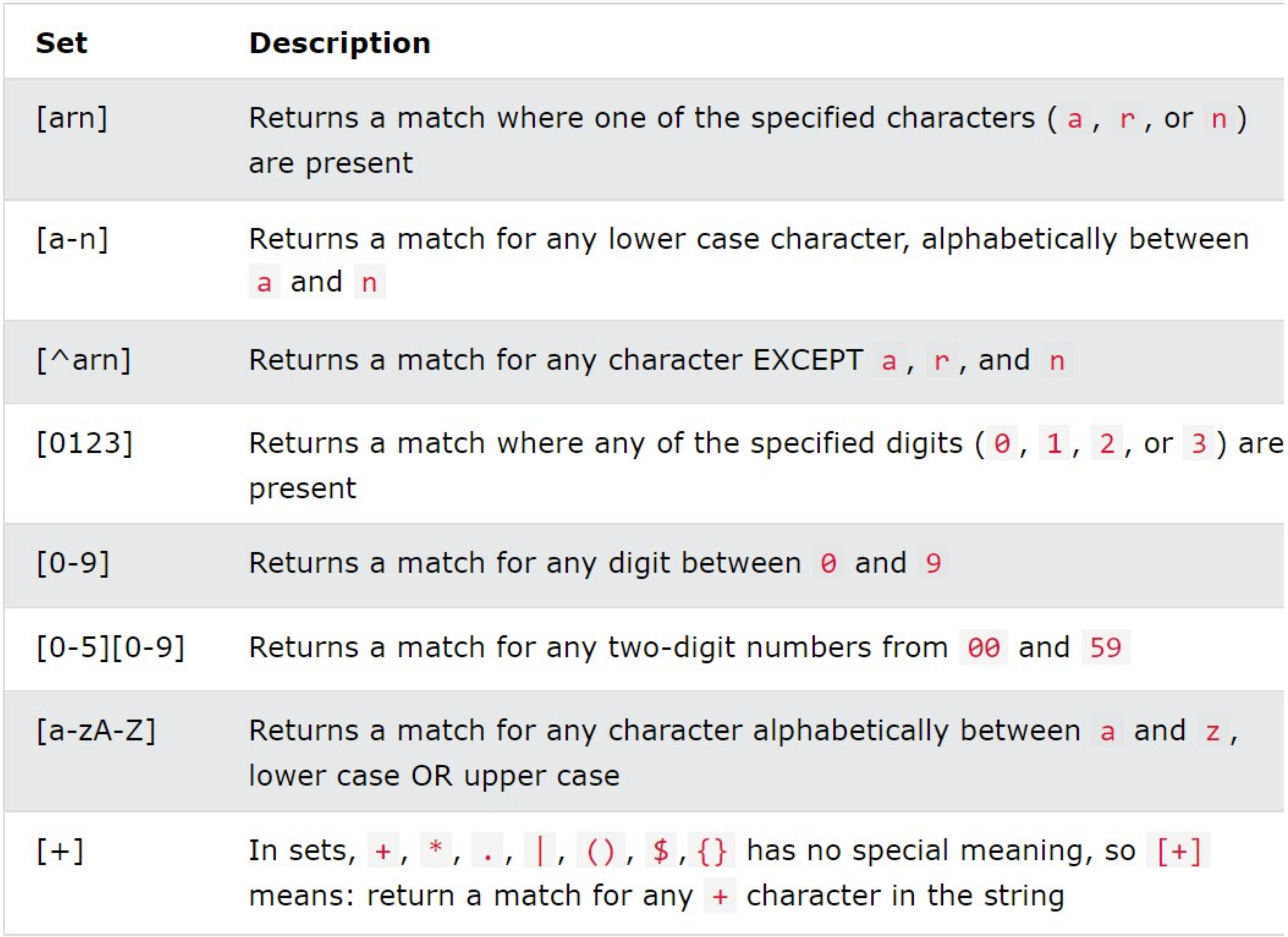
`\W`

Matches any non-alphanumeric character; this is equivalent to the class `[^a-zA-Z0-9_]`.

These sequences can be included inside a character class. For example, `[\s,.]` is a character class that will match any whitespace character, or `','` or `'.'`.

Credit: https://www.w3schools.com/python/python_regex.asp

Sets



Set	Description
[arn]	Returns a match where one of the specified characters (`a`, `r`, or `n`) are present
[a-n]	Returns a match for any lower case character, alphabetically between `a` and `n`
[^arn]	Returns a match for any character EXCEPT `a`, `r`, and `n`
[0123]	Returns a match where any of the specified digits (`0`, `1`, `2`, or `3`) are present
[0-9]	Returns a match for any digit between `0` and `9`
[0-5][0-9]	Returns a match for any two-digit numbers from `00` and `59`
[a-zA-Z]	Returns a match for any character alphabetically between `a` and `z`, lower case OR upper case
[+]	In sets, `+`, `*`, `.`, `|`, `()`, `$`,`{}` has no special meaning, so `[+]` means: return a match for any `+` character in the string

Emails

- ani@indomail.com
- ani.suteja@indomail.co.id
- adi-mantani1983@kaist.ac.kr

Buatlah regex pattern yang cover semua kasus alamat email di atas!

Emails

- ani@indomail.com
- ani.suteja@indomail.co.id
- adi-mantani1983@kaist.ac.kr

```
'\w+@\w+'
```

Ok?

Buatlah regex pattern yang cover semua kasus alamat email di atas!

Emails

- ani@indomail.com
- ani.suteja@indomail.co.id
- adi-mantani1983@kaist.ac.kr

```
'[\w.-]+@[\w.-]+'
```

Ok?

Buatlah regex pattern yang cover semua kasus alamat email di atas!

Group Extraction

Misal, kita ingin extract username & domain dari alamat email

```
import re


str = 'email saya adalah rudi.widodo@cs.ui.ac.id harap disimpan'
match = re.search("([\w.-]+)@([\w.-]+)", str)
if match:
  print(match.group())    ## whole match: rudi.widodo@cs.ui.ac.id
  print(match.group(1))   ## group 1: rudi.widodo
  print(match.group(2))   ## group 2: cs.ui.ac.id
```

Gunakan (dan) untuk grouping

findall()

Extract daftar alamat email dari sebuah text!

```
import re

str = '''
Berikut adalah daftar email yang tidak aktif: ani@mail.co.id, anto@cs.ui.ac.id,
rani.juliana@mail.co.id, dan andi-bhawika@halo.org Kami mohon untuk menghapus
email tersebut dalam daftar anggota.
'''


match = re.findall("[\w.-]+@[\w.-]+", str)

for email in match:
  print(match)

#ani@mail.co.id
#anto@cs.ui.ac.id
#rani.juliana@mail.co.id
#andi-bhawika@halo.org
```

findall()

Extract daftar alamat email dari sebuah text!

```
import re

str = '''
Berikut adalah daftar email yang tidak aktif: ani@mail.co.id, anto@cs.ui.ac.id,
rani.juliana@mail.co.id, dan andi-bhawika@halo.org. Kami mohon untuk menghapus
email tersebut dalam daftar anggota.
'''

match = re.findall("[\w.-]+@[\w.-]+", str)

for email in match:
  print(match)

#ani@mail.co.id
#anto@cs.ui.ac.id
#rani.juliana@mail.co.id
#andi-bhawika@halo.org.
```

Ada titik

Latihan: apa solusinya?

Titik yang tidak diharapkan

findall() & Groups

```
import re

str = '''
Berikut adalah daftar email yang tidak aktif: ani@mail.co.id, anto@cs.ui.ac.id,
rani.juliana@mail.co.id, dan andi-bhawika@halo.org Kami mohon untuk menghapus
email tersebut dalam daftar anggota.
'''


match_tuples = re.findall("([\w.-]+)@([\w.-]+)", str)

for match_tuple in match_tuples:
  print('username: {} dan host: {}'.format(match_tuple[0], match_tuple[1]))

#username: ani dan host: mail.co.id
#username: anto dan host: cs.ui.ac.id
#username: rani.juliana dan host: mail.co.id
#username: andi-bhawika dan host: halo.org
```

Greedy & Non-Greedy

```
import re

str = "<b>hello</b><b>world</b>"

matches = re.findall("<b>.*</b>", str)
for match in matches:
  print(match)
```

Akan match sepanjang mungkin

Output:

```
<b>hello</b><b>world</b>
```

Greedy & Non-Greedy

```
import re

str = "<b>hello</b><b>world</b>"

matches = re.findall("<b>.*?</b>", str)
for match in matches:
  print(match)
```

Greedy: stop at the first </b>

Output:

```
<b>hello</b>
<b>world</b>
```

Latihan

```
import re

str = "<b>hello</b><b>world</b>"

matches = re.findall("<b>.*?</b>", str)
for match in matches:
  print(match)
```

Latihan
Modifikasi pattern ini sehingga output tanpa <b> dan </b>

Output:
```
hello
world
```

split()

```
import re



str = "lauk seperti telur, rendang, tempe, dan tahu sangat bergizi."
tokens = re.split("[\W]+", str)
print(tokens)
```

Output:

```
['lauk', 'seperti', 'telur', 'rendang', 'tempe', 'dan', 'tahu', 'sangat', 'bergizi', '']
```

split()

```
import re



str = "lauk seperti telur, rendang, tempe, dan tahu sangat bergizi."

tokens = re.split("([\W]+)", str)

print(tokens)
```

Output:

```
['lauk', ' ', 'seperti', ' ', 'telur', ', ', 'rendang', ', ', 'tempe', ', ', 'dan', ' ', 'tahu', ' ',
'sangat', ' ', 'bergizi', '.', '']
```

Sub() - replace the matches

```
import re


str = "blue socks and red shoes"


new_str = re.sub("(blue|white|red)", "color", str)

print(new_str)
```

Output:

```
color socks and color shoes
```

Sub() - replace the matches

```
import re


str = "ani@gmail.com rudi@gmail.com dedy@gmail.com"

new_str = re.sub("([\w\.-]+)@([\w\.-]+)", "\1@cs.ui.ac.id", str)

print(new_str)
```

\1 \2 ... maksudnya me-refer ke group(1), group(2), ...

Output:

```
ani@cs.ui.ac.id rudi@cs.ui.ac.id dedy@cs.ui.ac.id
```