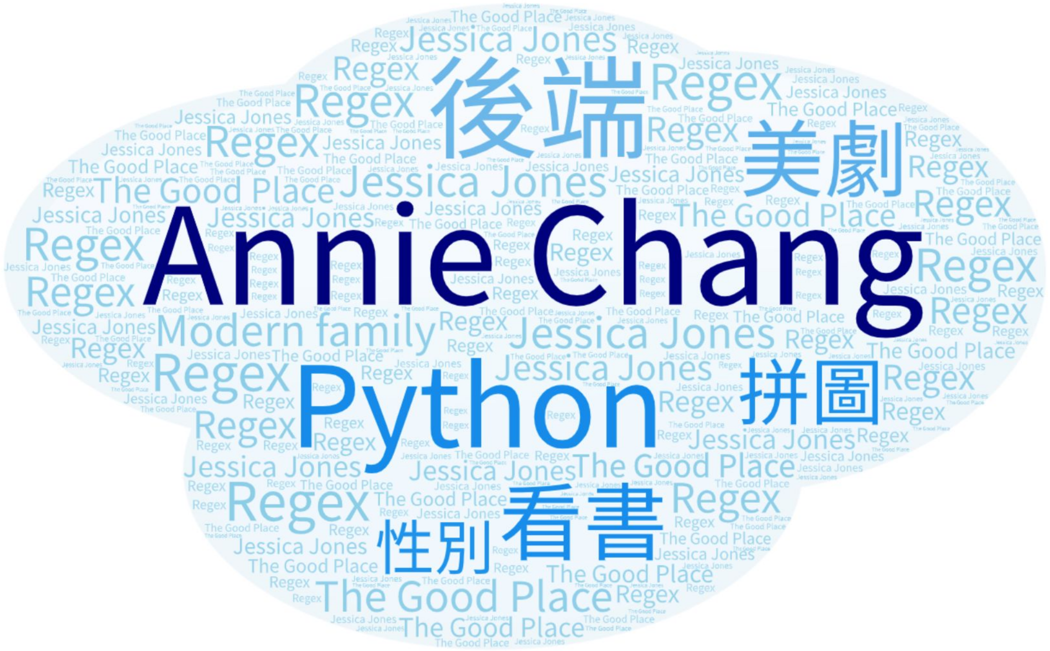
Regular Expression Workshop

Annie 2021.12.19

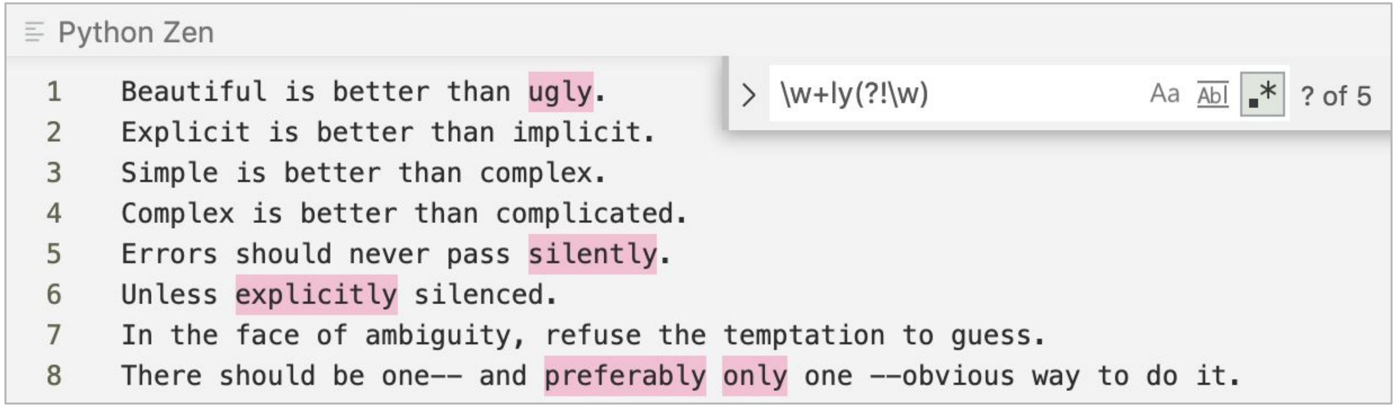
Download Slides : **http://shorturl.at/puyDR**

About Me


Annie Chang
Python
後端
美劇
拼圖
看書
性別
Modern family
Jessica Jones
The Good Place
Regex

What is RE

Regular Expression, RegExp, regex, 正則表達式, 正規表達式...



Python Zen

\w+ly(?!\w)   Aa   Abl   .*   ? of 5

```
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
```

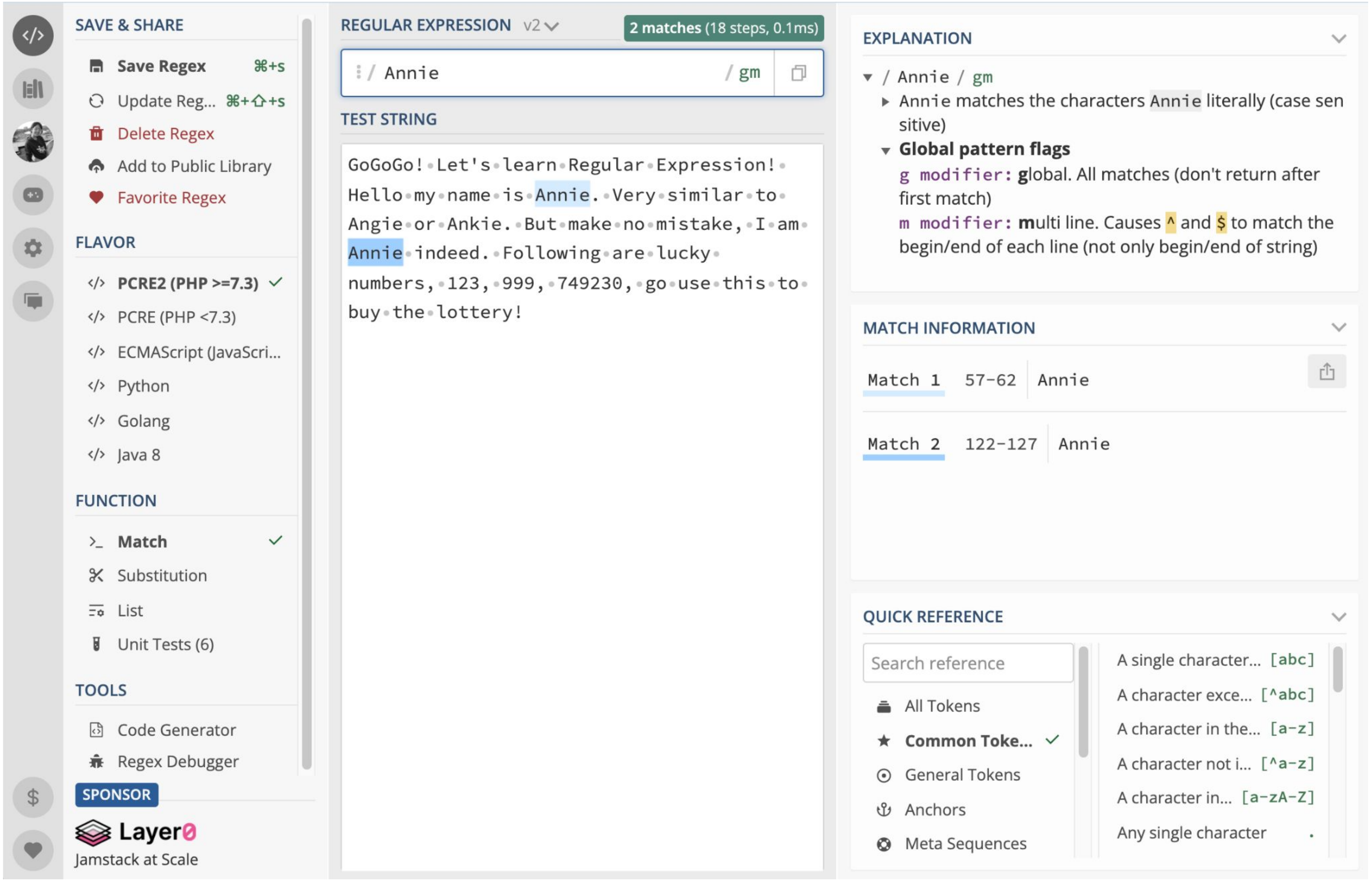
What is not RE

```
annie_chang@11:38~/Music$ ls iTunes*
  Album Artwork                iTunes Library Genius.itdb   iTunes Media
  iTunes Library Extras.itdb   iTunes Library.itl           sentinel
```

Agenda

1. **Regex - Basic**
2. Regex - Advanced
3. **Python library `re`**
4. grep

Regex - Basic



REGULAR EXPRESSION v2 — 2 matches (18 steps, 0.1ms)

```
/ Annie / gm
```

TEST STRING

```
GoGoGo! Let's learn Regular Expression! Hello my name is Annie. Very similar to Angie or Ankie. But make no mistake, I am Annie indeed. Following are lucky numbers, 123, 999, 749230, go use this to buy the lottery!
```

EXPLANATION

- / Annie / gm
  - Annie matches the characters Annie literally (case sensitive)
  - **Global pattern flags**
    - g modifier: **g**lobal. All matches (don't return after first match)
    - m modifier: **m**ulti line. Causes ^ and $ to match the begin/end of each line (not only begin/end of string)

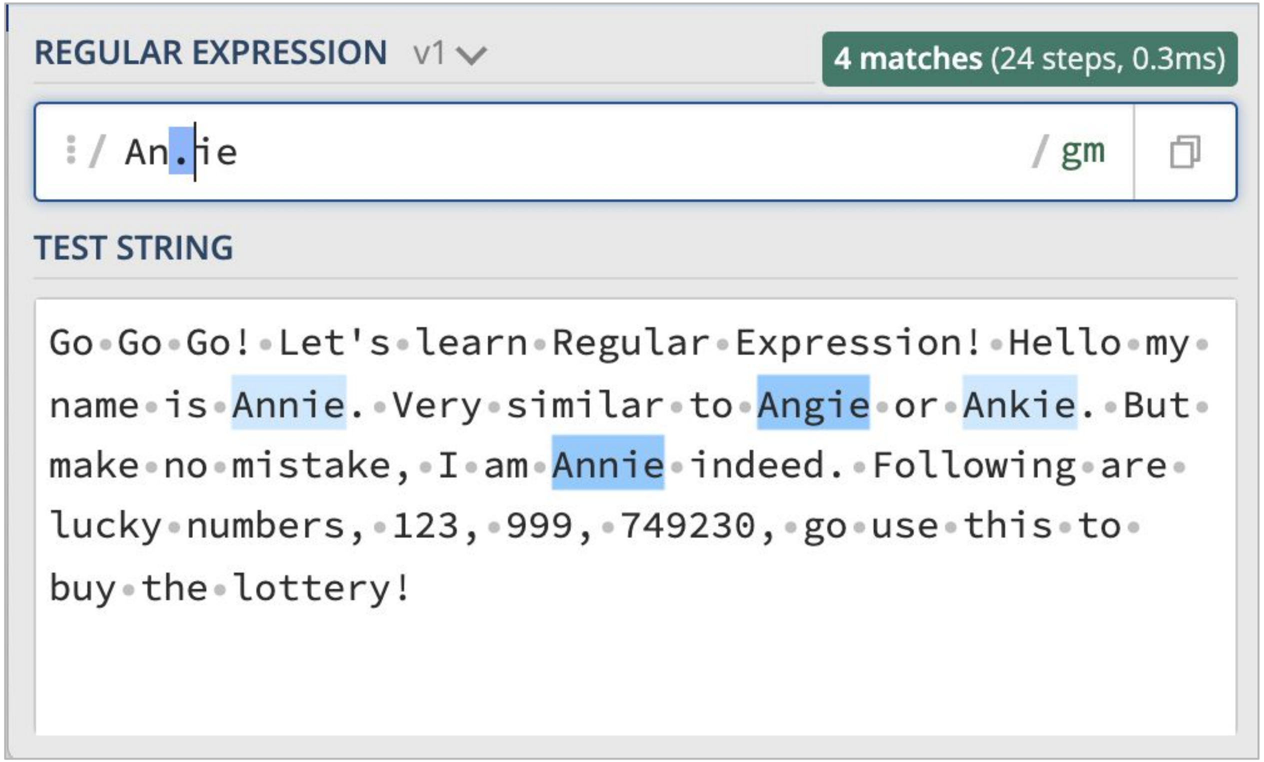
MATCH INFORMATION

Match 1	57-62	Annie
Match 2	122-127	Annie

QUICK REFERENCE

A single character...	[abc]
A character exce...	[^abc]
A character in the...	[a-z]
A character not i...	[^a-z]
A character in...	[a-zA-Z]
Any single character	.

Dot . (Matches Any Single Character)


REGULAR EXPRESSION v1

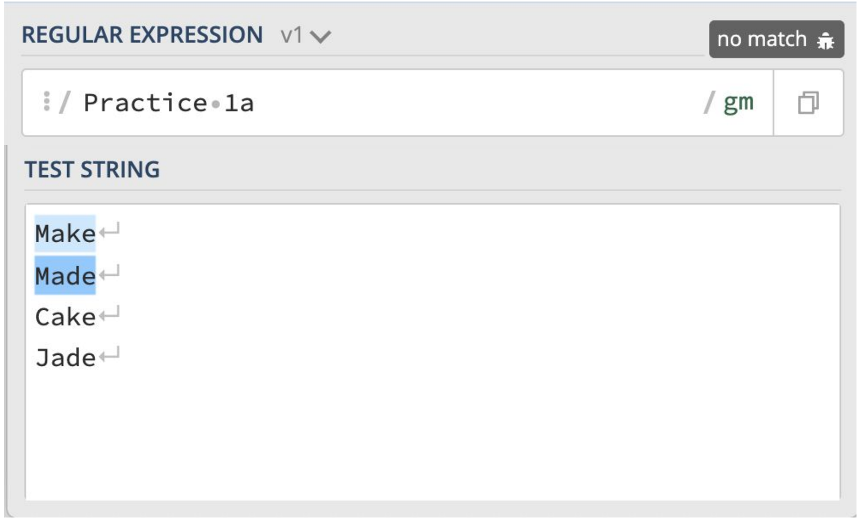
4 matches (24 steps, 0.3ms)

```
/ An.ie / gm
```

TEST STRING

```
Go Go Go! Let's learn Regular Expression! Hello my
name is Annie. Very similar to Angie or Ankie. But
make no mistake, I am Annie indeed. Following are
lucky numbers, 123, 999, 749230, go use this to
buy the lottery!
```

Practice Time (2 Problems)



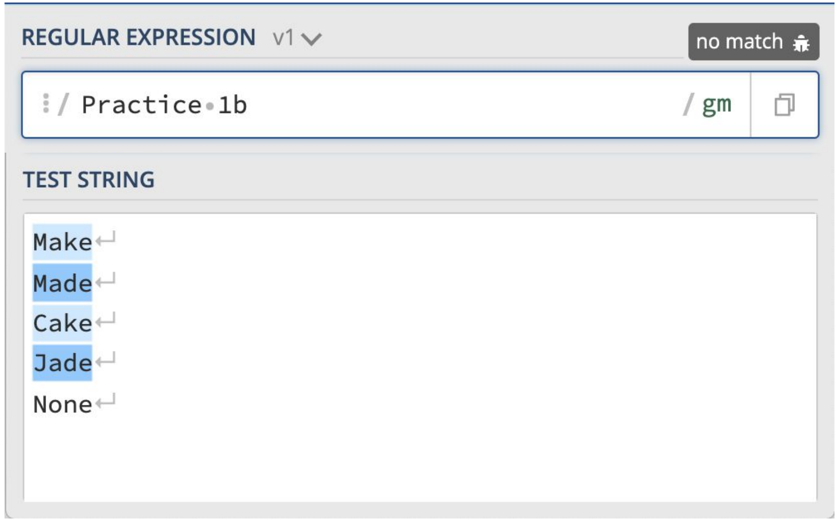
REGULAR EXPRESSION v1

no match

```
/ Practice 1a / gm
```

TEST STRING

```
Make
Made
Cake
Jade
```



REGULAR EXPRESSION v1

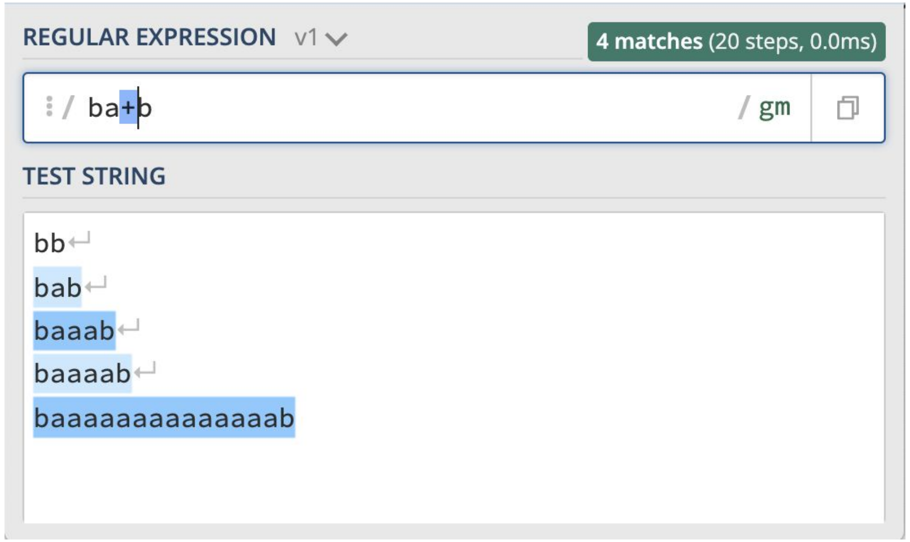
no match

```
/ Practice 1b / gm
```

TEST STRING

```
Make
Made
Cake
Jade
None
```

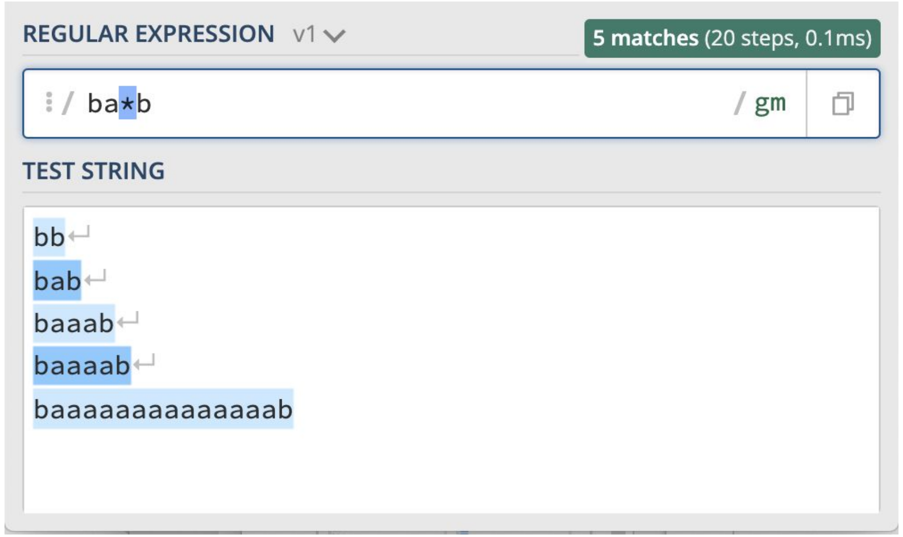
Plus Sign + (>=1 times)
Star Sign * (>=0 times)


REGULAR EXPRESSION v1 — 4 matches (20 steps, 0.0ms)

```
/ ba+b / gm
```

TEST STRING

```
bb
bab
baaab
baaaab
baaaaaaaaaaaaab
```


REGULAR EXPRESSION v1 — 5 matches (20 steps, 0.1ms)

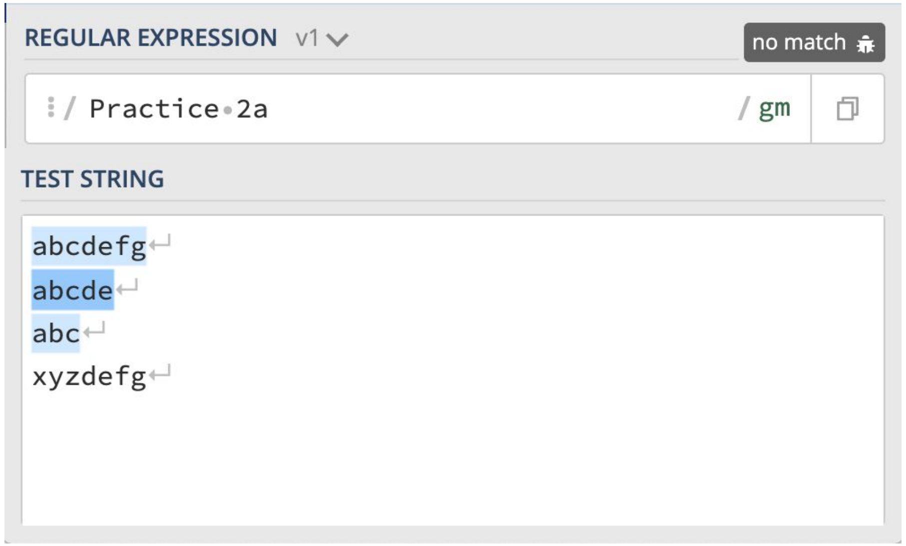
```
/ ba*b / gm
```

TEST STRING

```
bb
bab
baaab
baaaab
baaaaaaaaaaaaab
```

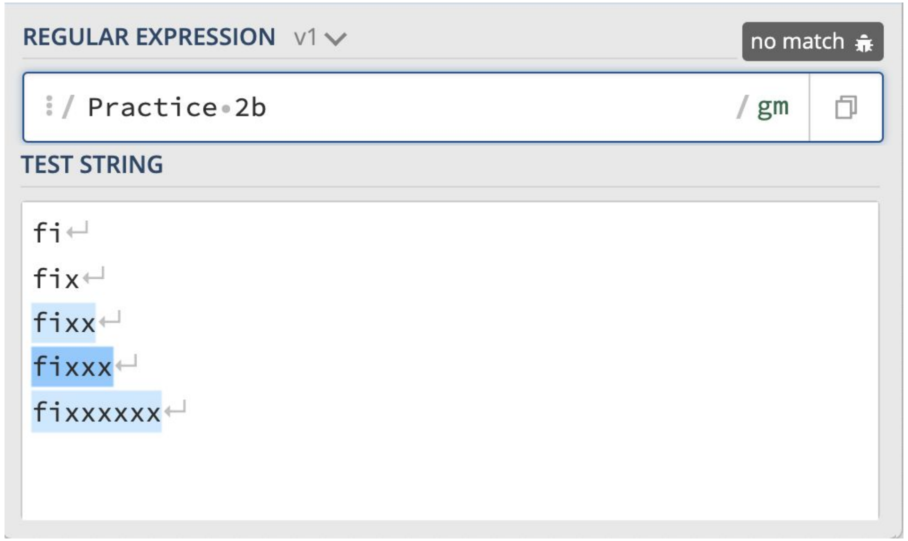
- /a*a/
- /a+a/
- /a*/ will match empty string

Practice Time (3 Problems)



```
/ Practice 2a / gm
```

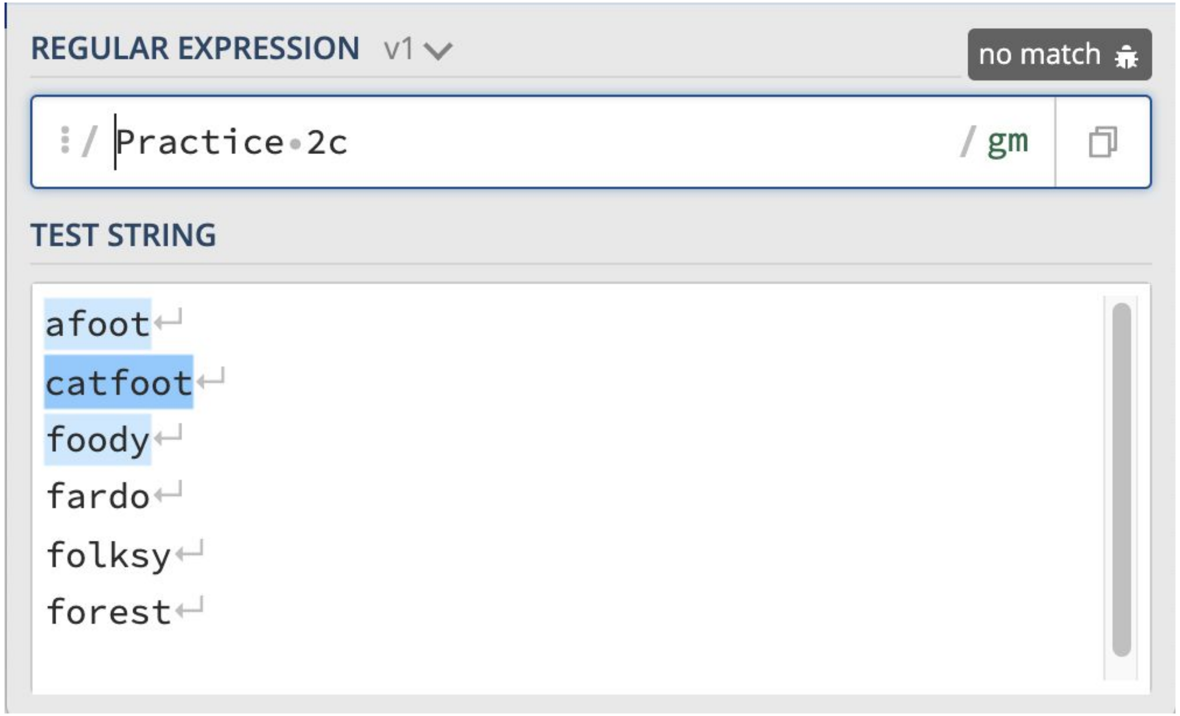
```
abcdefg
abcde
abc
xyzdefg
```



```
/ Practice 2b / gm
```

```
fi
fix
fixx
fixxx
fixxxxxx
```

Practice Time (3 Problems)



REGULAR EXPRESSION v1

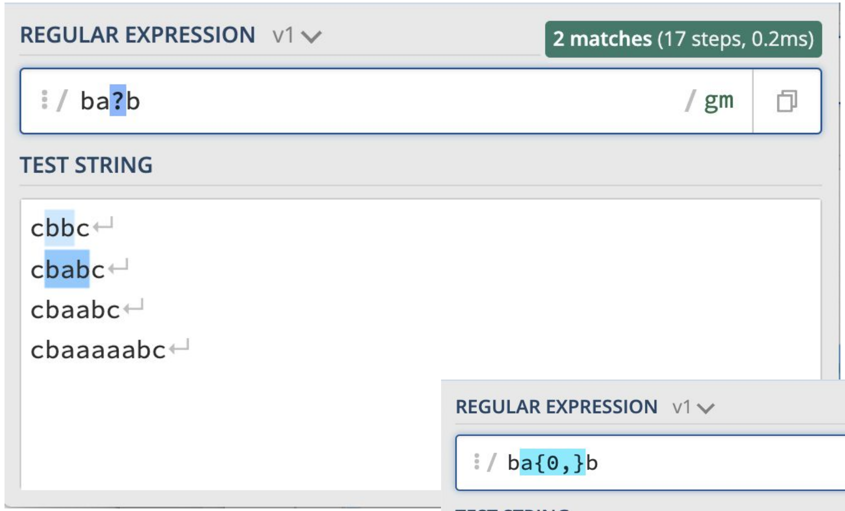
no match

/ Practice 2c / gm

TEST STRING

```
afoot
catfoot
foody
fardo
folksy
forest
```

Question Mark ? (0 or 1 time)
Curly brackets {m, n} (m~n times)

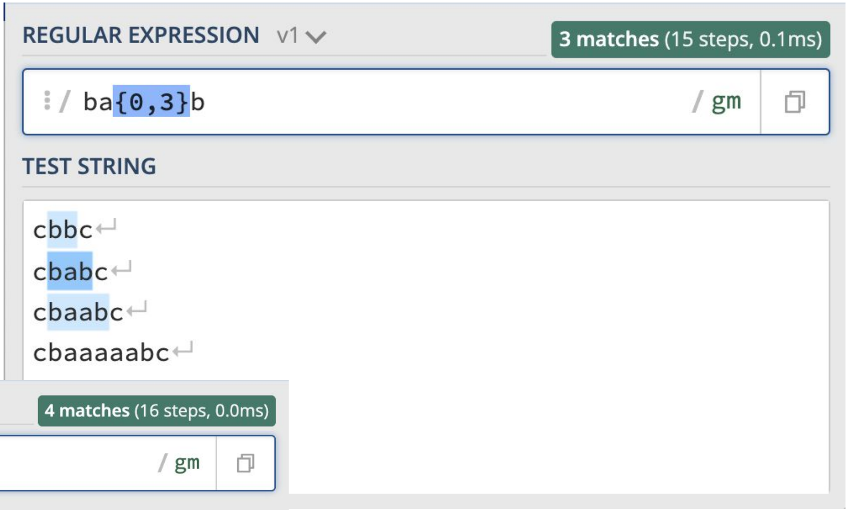

REGULAR EXPRESSION v1

2 matches (17 steps, 0.2ms)

```
/ ba?b / gm
```

TEST STRING

```
cbbc
cbabc
cbaabc
cbaaaaabc
```

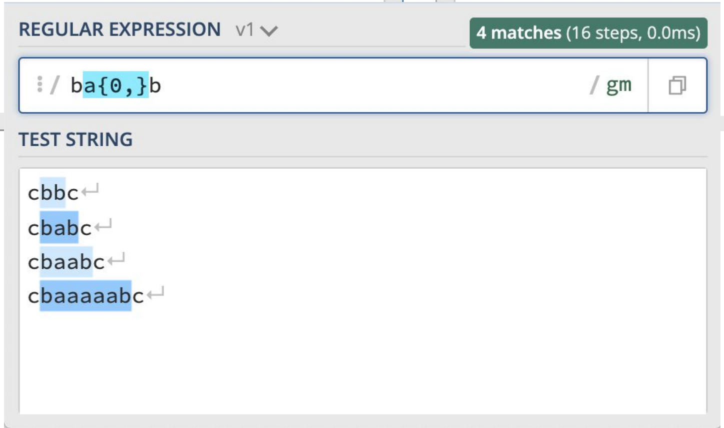

REGULAR EXPRESSION v1

3 matches (15 steps, 0.1ms)

```
/ ba{0,3}b / gm
```

TEST STRING

```
cbbc
cbabc
cbaabc
cbaaaaabc
```

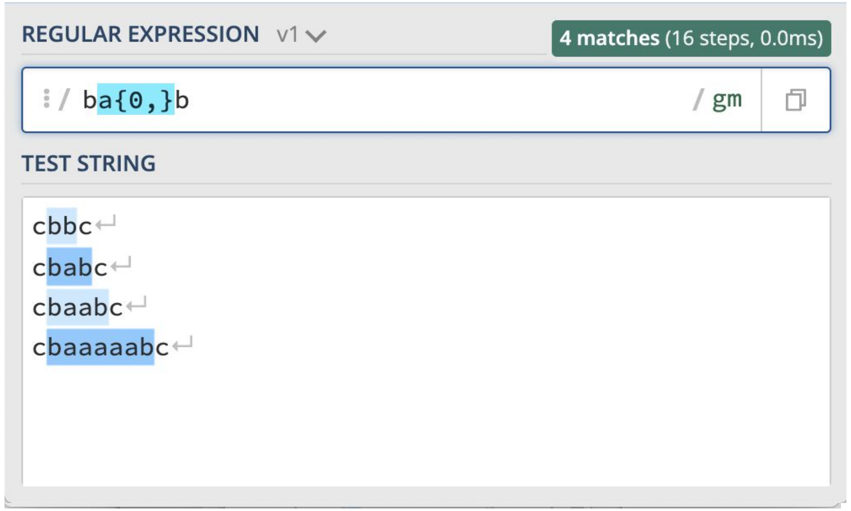

REGULAR EXPRESSION v1

4 matches (16 steps, 0.0ms)

```
/ ba{0,}b / gm
```

TEST STRING

```
cbbc
cbabc
cbaabc
cbaaaaabc
```


REGULAR EXPRESSION v1

4 matches (16 steps, 0.0ms)

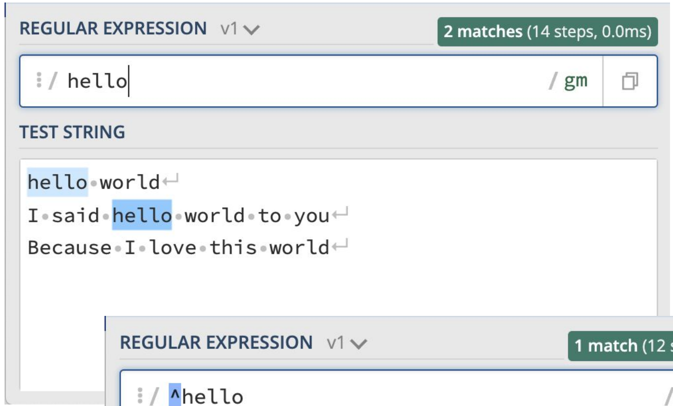
```
/ ba{0,}b / gm
```

TEST STRING

```
cbbc
cbabc
cbaabc
cbaaaaabc
```

- `/a*b/ = /a{_,_}b/`
- `/a+b/ = /a{_,_}b/`
- `/a?b/ = /a{_,_}b/`
- `/a?ab/ = /a{_,_}b/`

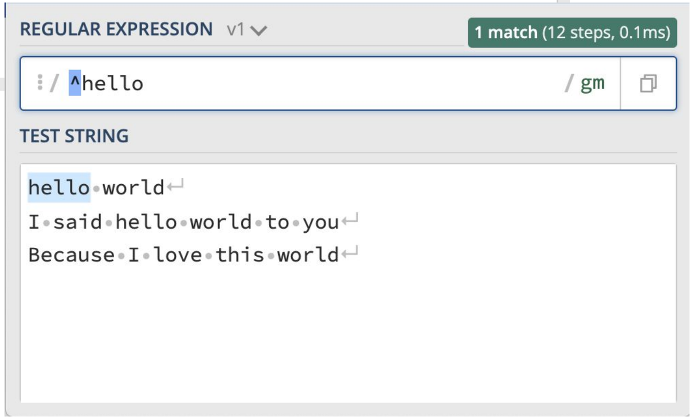
Caret Symbol ^ (Start of line)
Dollar Sign $ (End of line)


REGULAR EXPRESSION v1 — 2 matches (14 steps, 0.0ms)

```
/ hello / gm
```

TEST STRING

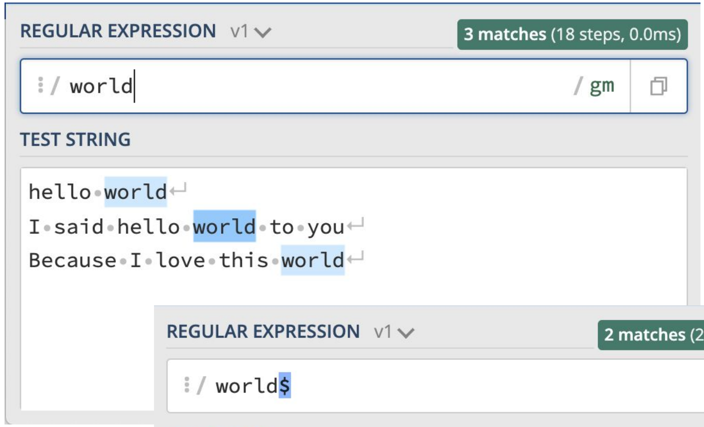
```
hello world
I said hello world to you
Because I love this world
```


REGULAR EXPRESSION v1 — 1 match (12 steps, 0.1ms)

```
/ ^hello / gm
```

TEST STRING

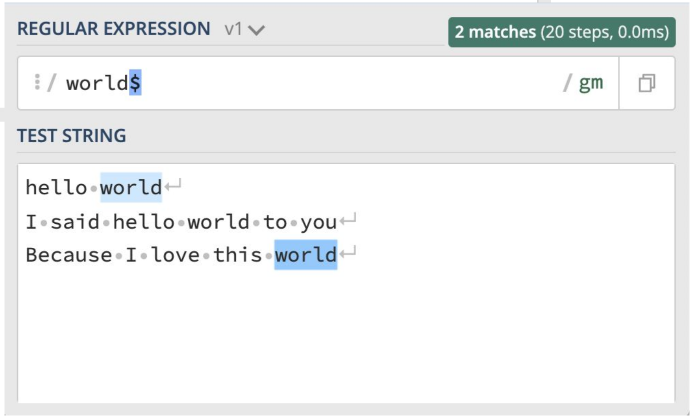
```
hello world
I said hello world to you
Because I love this world
```


REGULAR EXPRESSION v1 — 3 matches (18 steps, 0.0ms)

```
/ world / gm
```

TEST STRING

```
hello world
I said hello world to you
Because I love this world
```


REGULAR EXPRESSION v1 — 2 matches (20 steps, 0.0ms)

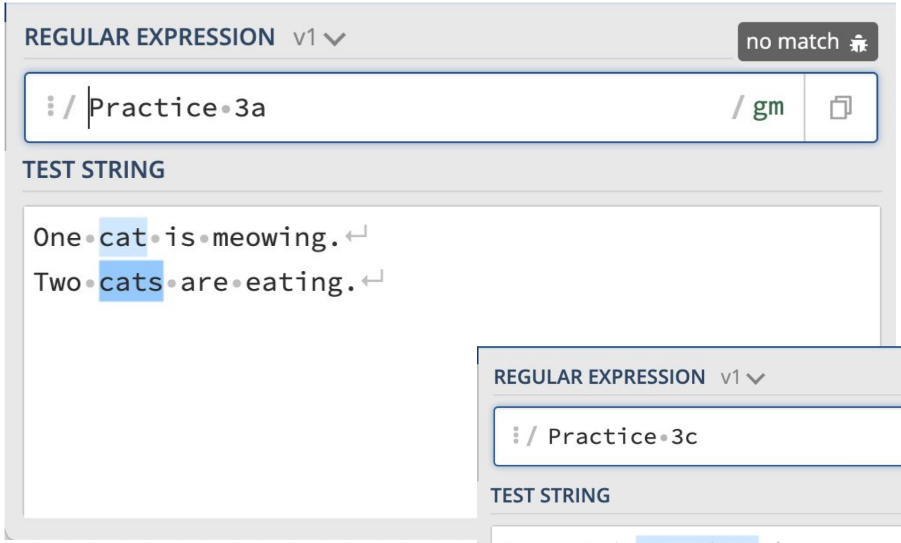
```
/ world$ / gm
```

TEST STRING

```
hello world
I said hello world to you
Because I love this world
```

- /ab^cd/

Practice Time (3 Problems)

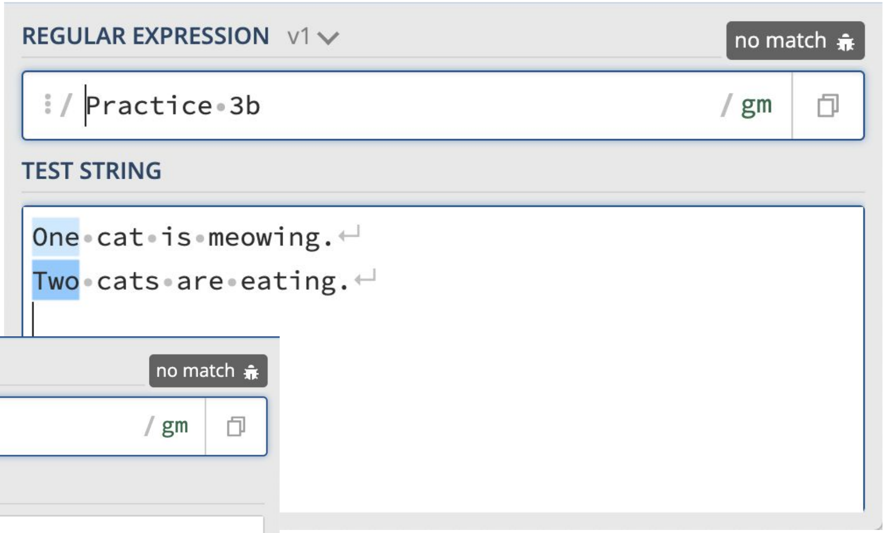


REGULAR EXPRESSION v1
no match

```
/ Practice 3a / gm
```

TEST STRING

```
One cat is meowing.
Two cats are eating.
```

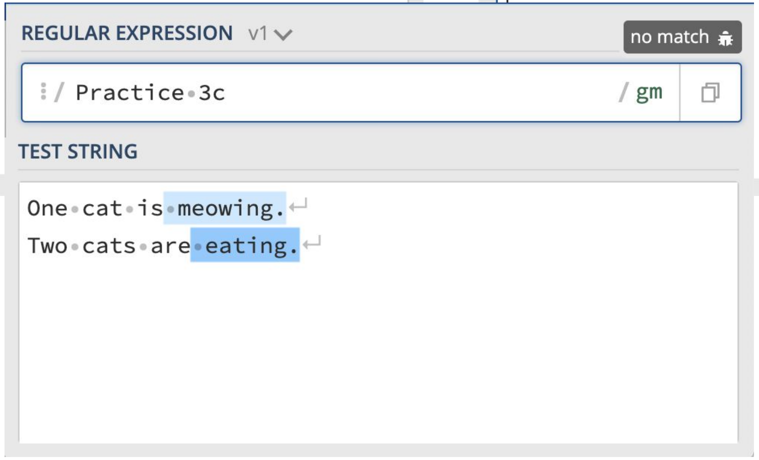


REGULAR EXPRESSION v1
no match

```
/ Practice 3b / gm
```

TEST STRING

```
One cat is meowing.
Two cats are eating.
```



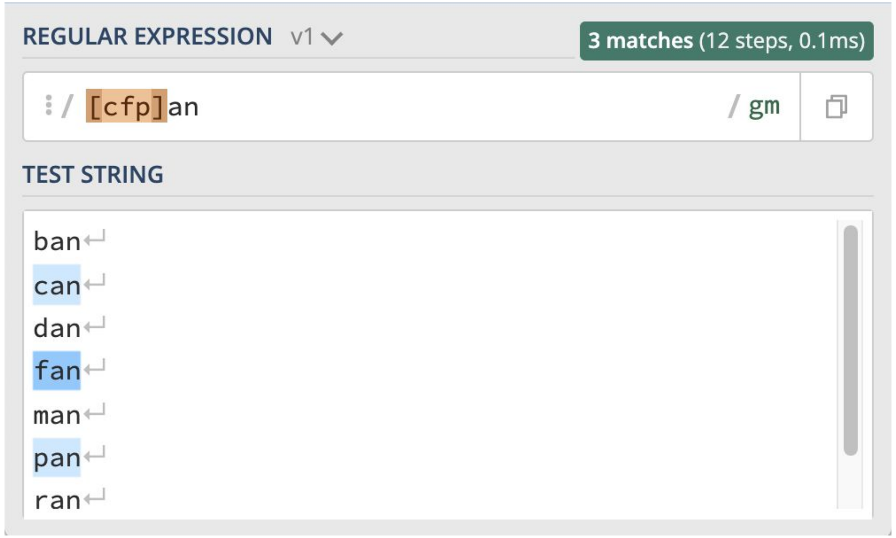
REGULAR EXPRESSION v1
no match

```
/ Practice 3c / gm
```

TEST STRING

```
One cat is meowing.
Two cats are eating.
```

Square Bracket [a-z] (Character Class)



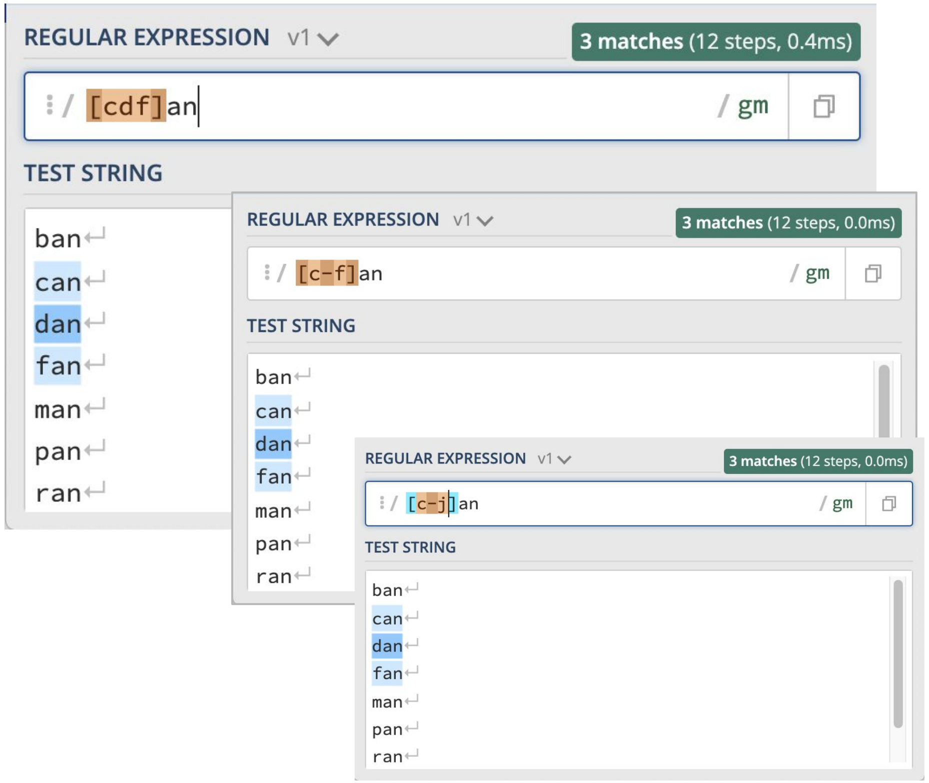
REGULAR EXPRESSION v1

3 matches (12 steps, 0.1ms)

```
/ [cfp]an / gm
```

TEST STRING

```
ban
can
dan
fan
man
pan
ran
```



REGULAR EXPRESSION v1

3 matches (12 steps, 0.4ms)

```
/ [cdf]an / gm
```

TEST STRING

```
ban
can
dan
fan
man
pan
ran
```

REGULAR EXPRESSION v1

3 matches (12 steps, 0.0ms)

```
/ [c-f]an / gm
```

TEST STRING

```
ban
can
dan
fan
man
pan
ran
```

REGULAR EXPRESSION v1

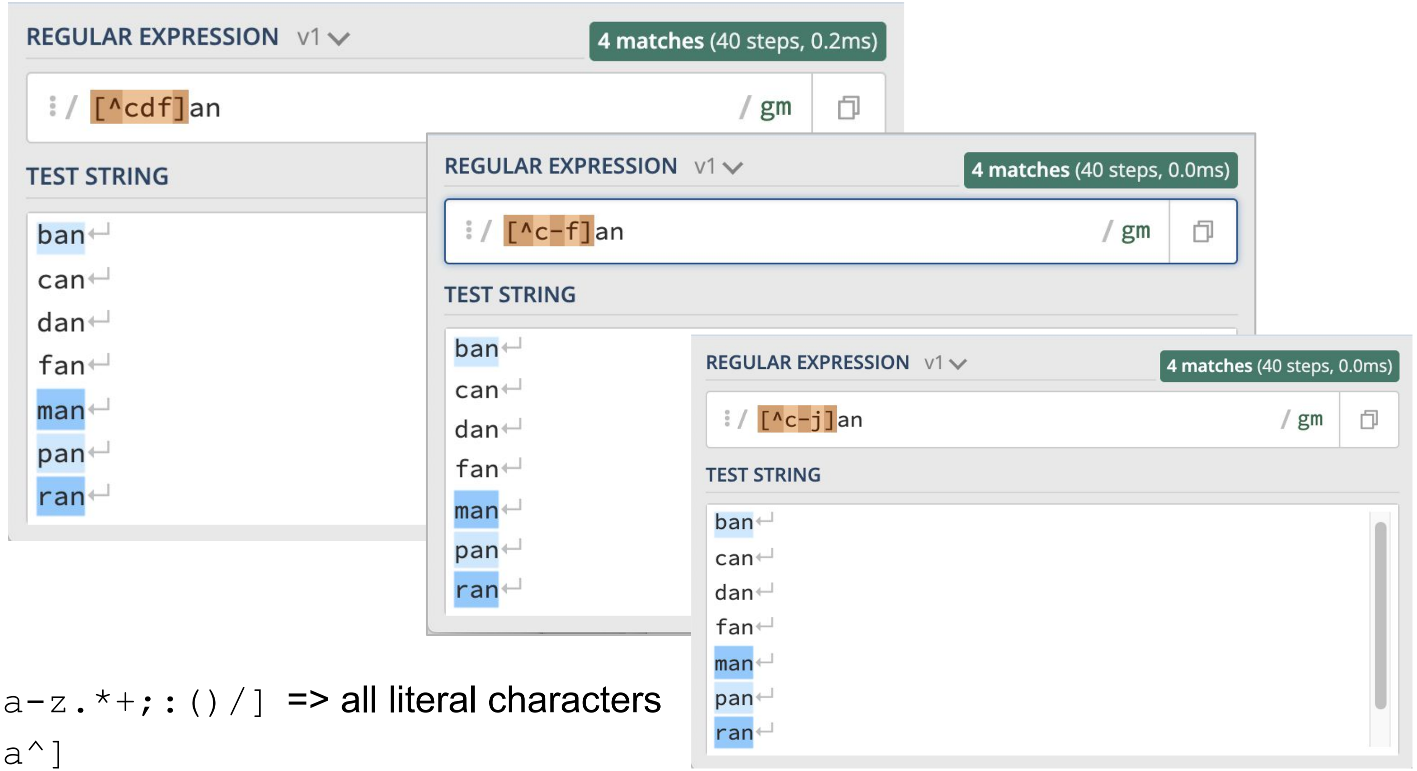
3 matches (12 steps, 0.0ms)

```
/ [c-j]an / gm
```

TEST STRING

```
ban
can
dan
fan
man
pan
ran
```

Character Class - Negative Match



REGULAR EXPRESSION v1 — 4 matches (40 steps, 0.2ms)

```
/ [^cdf]an / gm
```

TEST STRING

```
ban
can
dan
fan
man
pan
ran
```

REGULAR EXPRESSION v1 — 4 matches (40 steps, 0.0ms)

```
/ [^c-f]an / gm
```

TEST STRING

```
ban
can
dan
fan
man
pan
ran
```

REGULAR EXPRESSION v1 — 4 matches (40 steps, 0.0ms)

```
/ [^c-j]an / gm
```

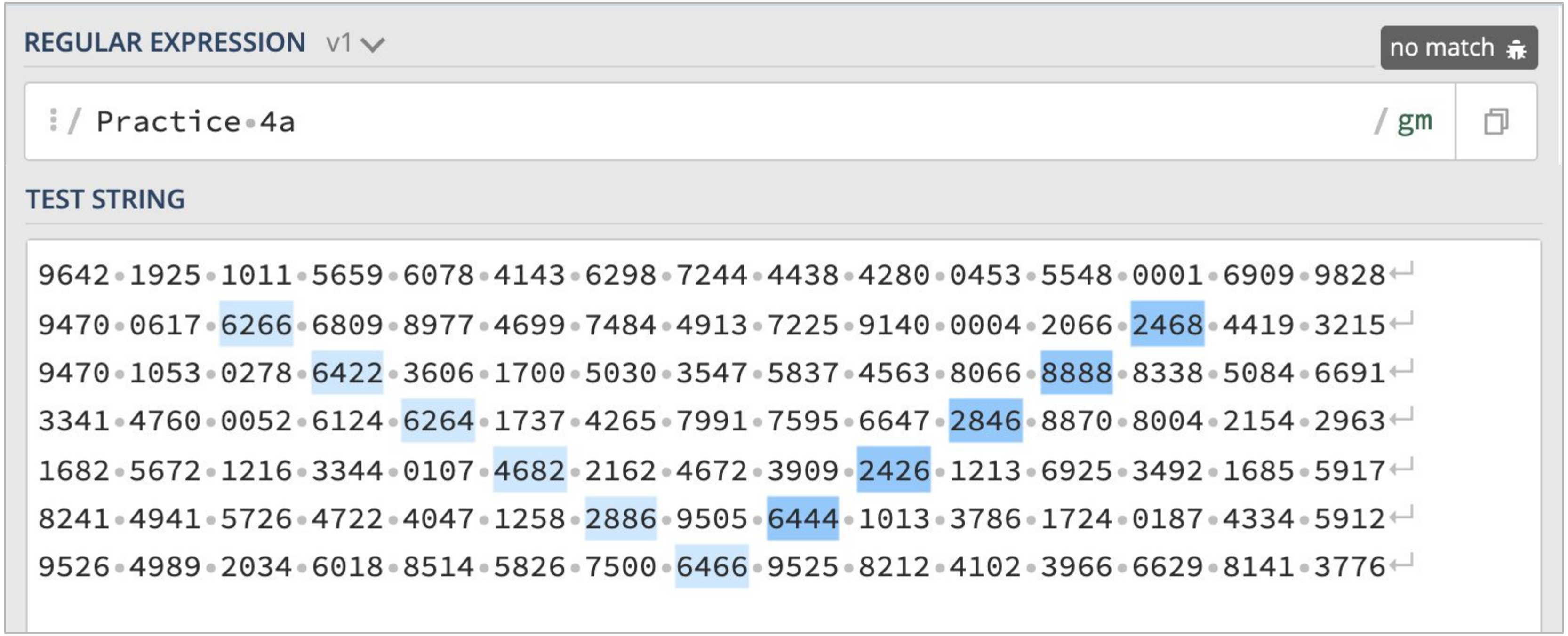
TEST STRING

```
ban
can
dan
fan
man
pan
ran
```

- `[a-z.*+;:()/]` => all literal characters
- `[a^]`
- `[-a]`

Practice Time (3 Problems)

Match numbers that only consist of digits 2, 4, 6, 8



REGULAR EXPRESSION v1

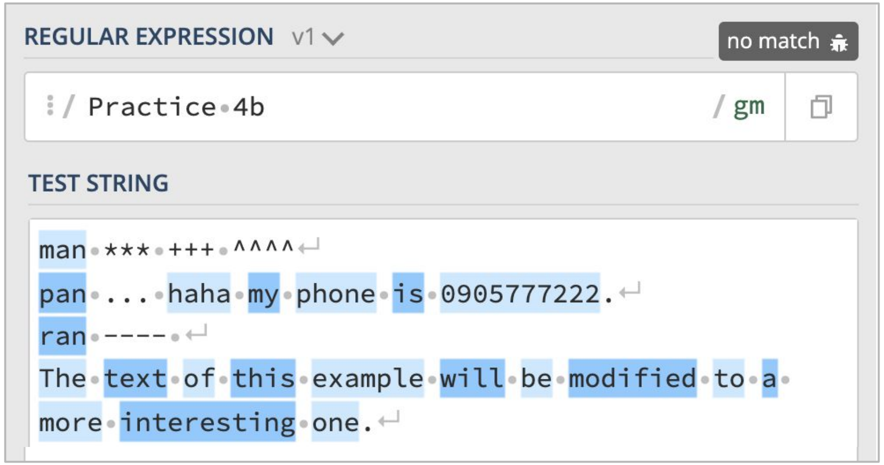
no match

```
/ Practice 4a / gm
```

TEST STRING

```
9642 1925 1011 5659 6078 4143 6298 7244 4438 4280 0453 5548 0001 6909 9828
9470 0617 6266 6809 8977 4699 7484 4913 7225 9140 0004 2066 2468 4419 3215
9470 1053 0278 6422 3606 1700 5030 3547 5837 4563 8066 8888 8338 5084 6691
3341 4760 0052 6124 6264 1737 4265 7991 7595 6647 2846 8870 8004 2154 2963
1682 5672 1216 3344 0107 4682 2162 4672 3909 2426 1213 6925 3492 1685 5917
8241 4941 5726 4722 4047 1258 2886 9505 6444 1013 3786 1724 0187 4334 5912
9526 4989 2034 6018 8514 5826 7500 6466 9525 8212 4102 3966 6629 8141 3776
```

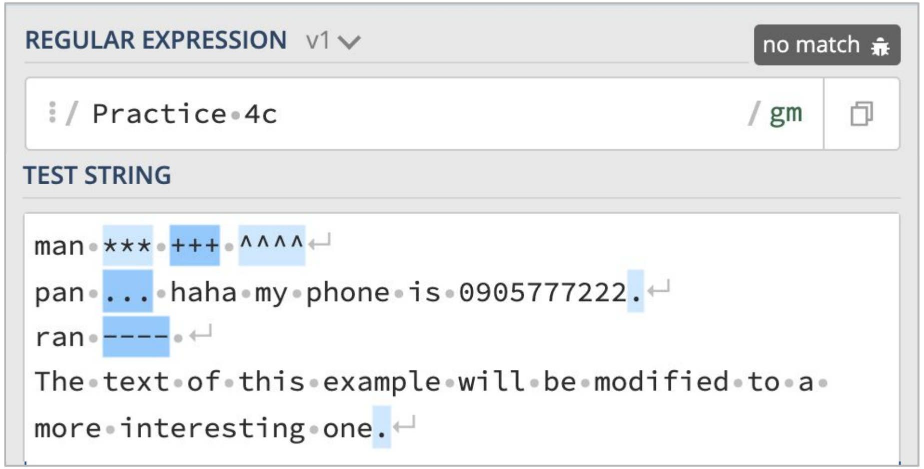
Practice Time (3 Problems)


REGULAR EXPRESSION v1 | no match

```
/ Practice 4b / gm
```

TEST STRING

```
man *** +++ ^^^^
pan ... haha my phone is 0905777222.
ran ---- 
The text of this example will be modified to a 
more interesting one.
```

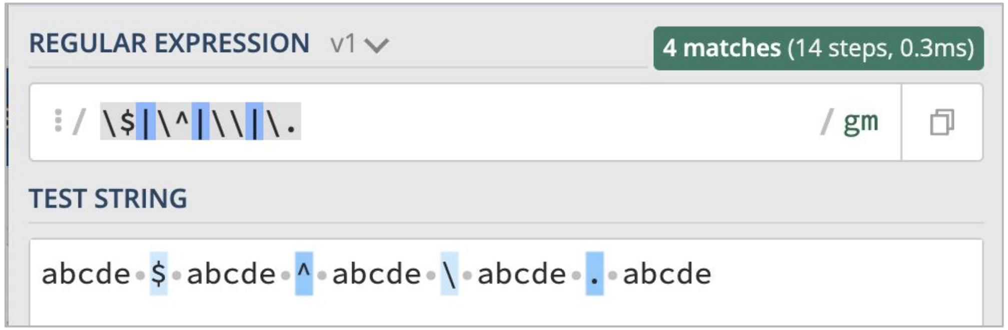

REGULAR EXPRESSION v1 | no match

```
/ Practice 4c / gm
```

TEST STRING

```
man *** +++ ^^^^
pan ... haha my phone is 0905777222.
ran ---- 
The text of this example will be modified to a 
more interesting one.
```

Escape Special Characters



```
/ \$|\^|\\|\. / gm
```

```
abcde $ abcde ^ abcde \ abcde . abcde
```

Shorthand

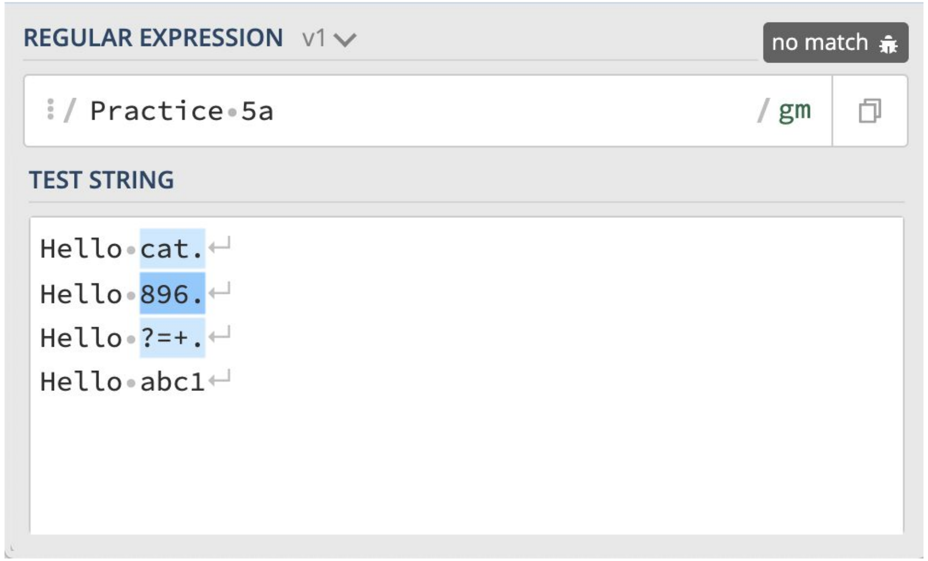
\d		[[:digit:]]	
\w		[[:alpha:]]	
\D		[[:alnum:]]	
\W		[[:lower:]]	

Cheat Sheet

.
+
*
?
{m, n}
^
$
[acd], [^acd]

\d
\w
\D
\W
[[:digit:]]
[[:alpha:]]
[[:alnum:]]
[[:lower:]]

Practice Time (3 Problems)



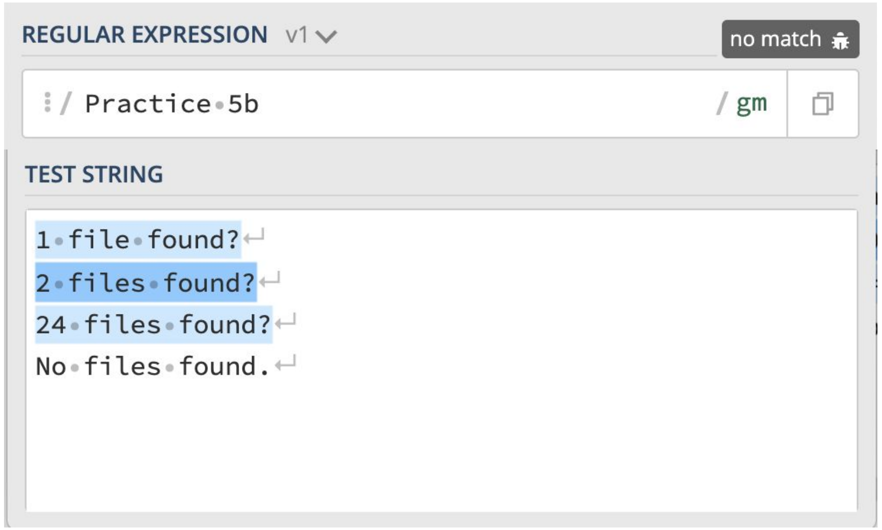
REGULAR EXPRESSION v1

no match

```
/ Practice 5a / gm
```

TEST STRING

```
Hello cat.
Hello 896.
Hello ?=+.
Hello abc1
```



REGULAR EXPRESSION v1

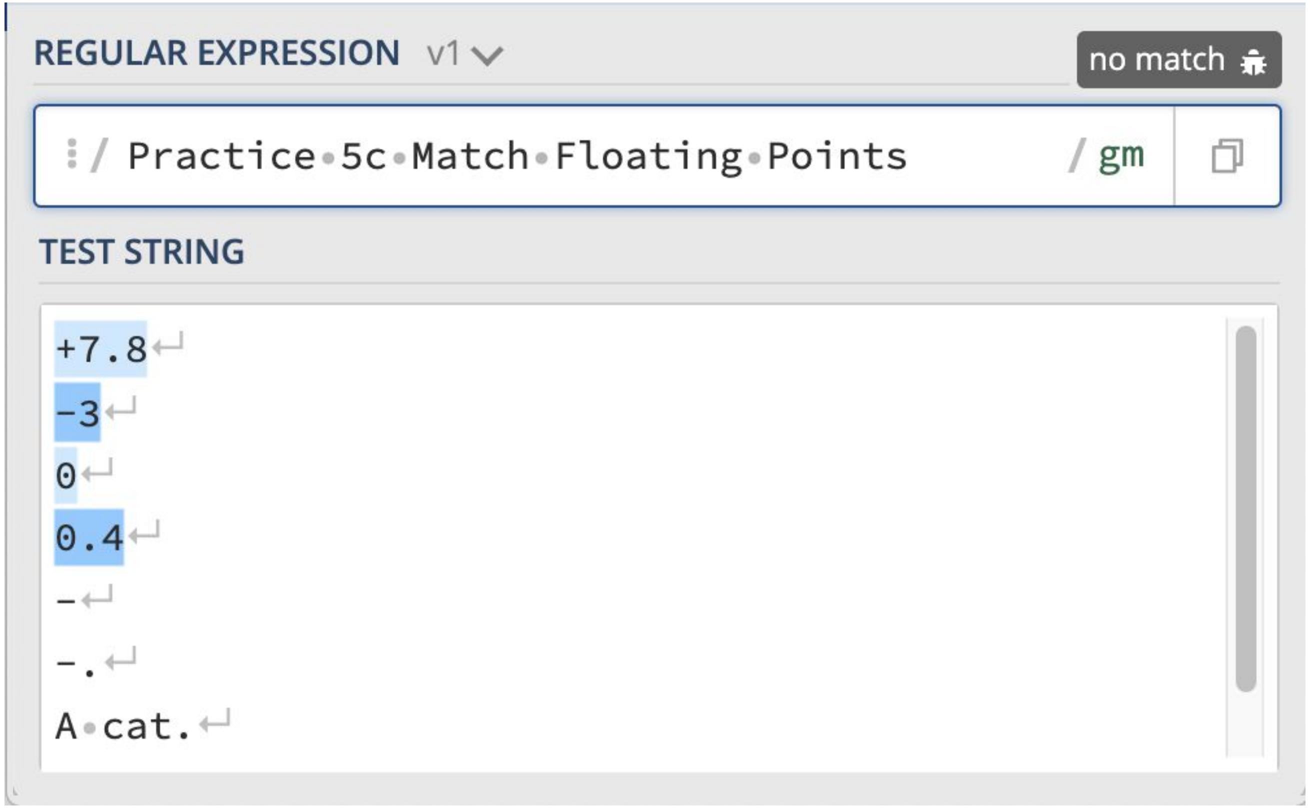
no match

```
/ Practice 5b / gm
```

TEST STRING

```
1 file found?
2 files found?
24 files found?
No files found.
```

Practice Time (3 Problems)



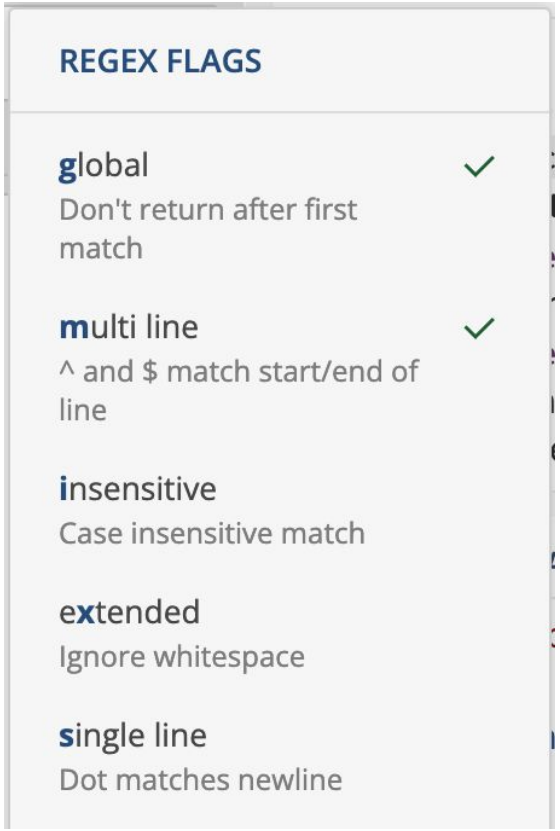
```
/ Practice 5c Match Floating Points / gm
```

TEST STRING

```
+7.8
-3
0
0.4
-
-.
A cat.
```

Regex - Advanced

Mode


REGEX FLAGS

global ✓
Don't return after first match

multi line ✓
^ and $ match start/end of line

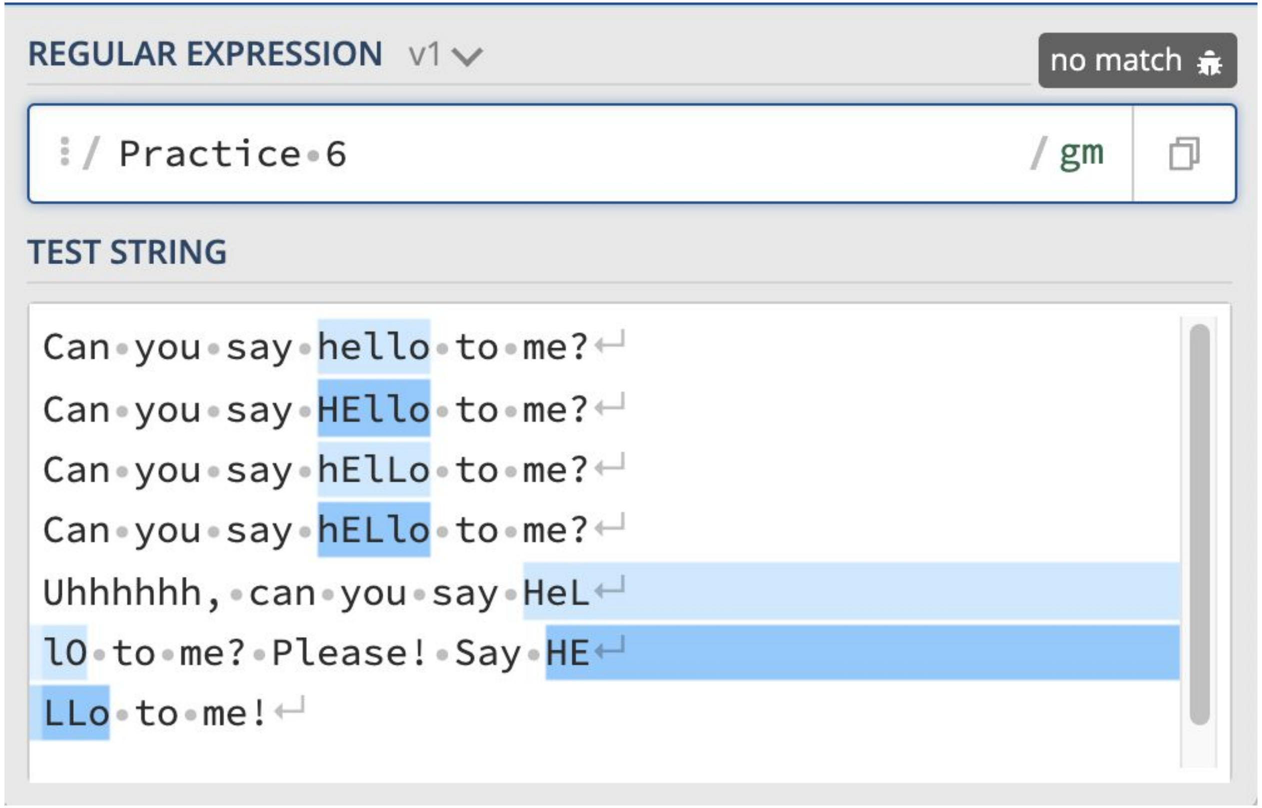
insensitive
Case insensitive match

extended
Ignore whitespace

single line
Dot matches newline

- `/(?i)test/`
- `/te(?i:st)/`
- `/(?i)te(?-i)st/`
- `/(?i)te(?^)st/`
- `/(?s)^.+$/`
    - dot matches line break
    - whole file start/end

Practice Time (1 Problem)



REGULAR EXPRESSION v1

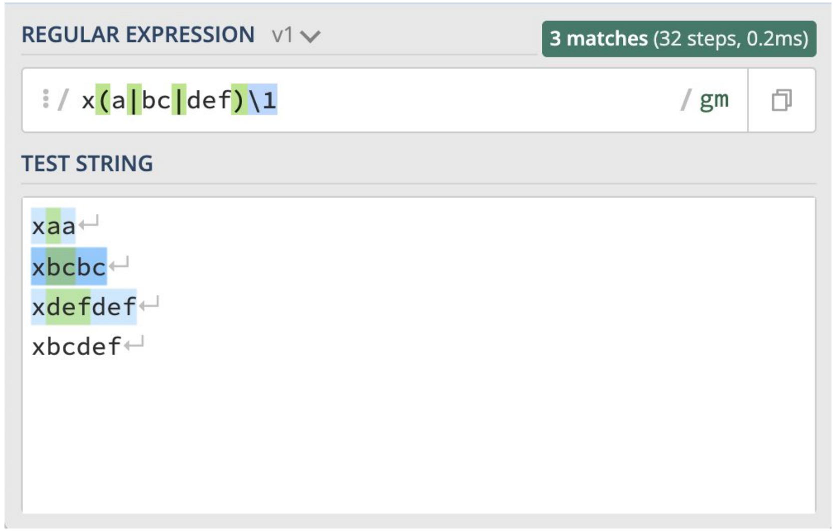
no match

```
/ Practice 6 / gm
```

TEST STRING

```
Can you say hello to me?
Can you say HEllo to me?
Can you say hElLo to me?
Can you say hELlo to me?
Uhhhhhh, can you say HeL
lO to me? Please! Say HE
LLo to me!
```

Capturing Groups



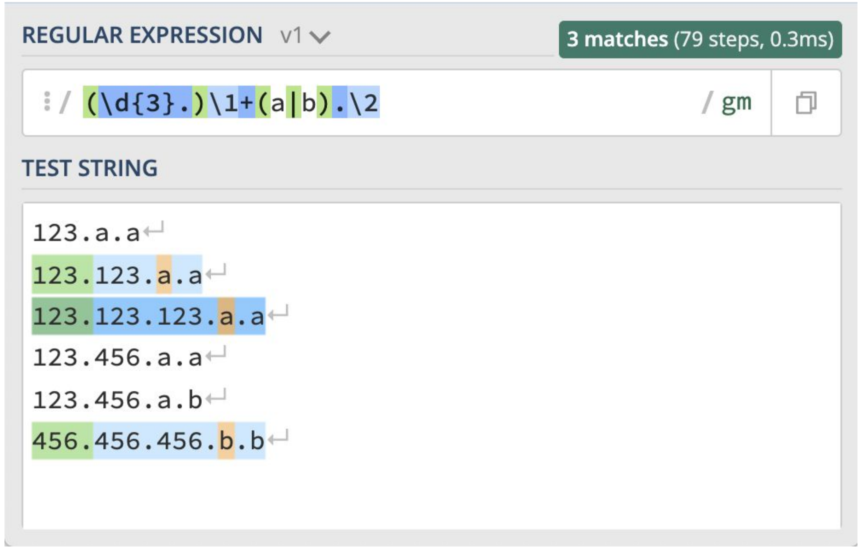
REGULAR EXPRESSION v1

3 matches (32 steps, 0.2ms)

```
/ x(a|bc|def)\1 / gm
```

TEST STRING

```
xaa
xbcbc
xdefdef
xbcdef
```



REGULAR EXPRESSION v1

3 matches (79 steps, 0.3ms)

```
/ (\d{3}.)\1+(a|b).\2 / gm
```

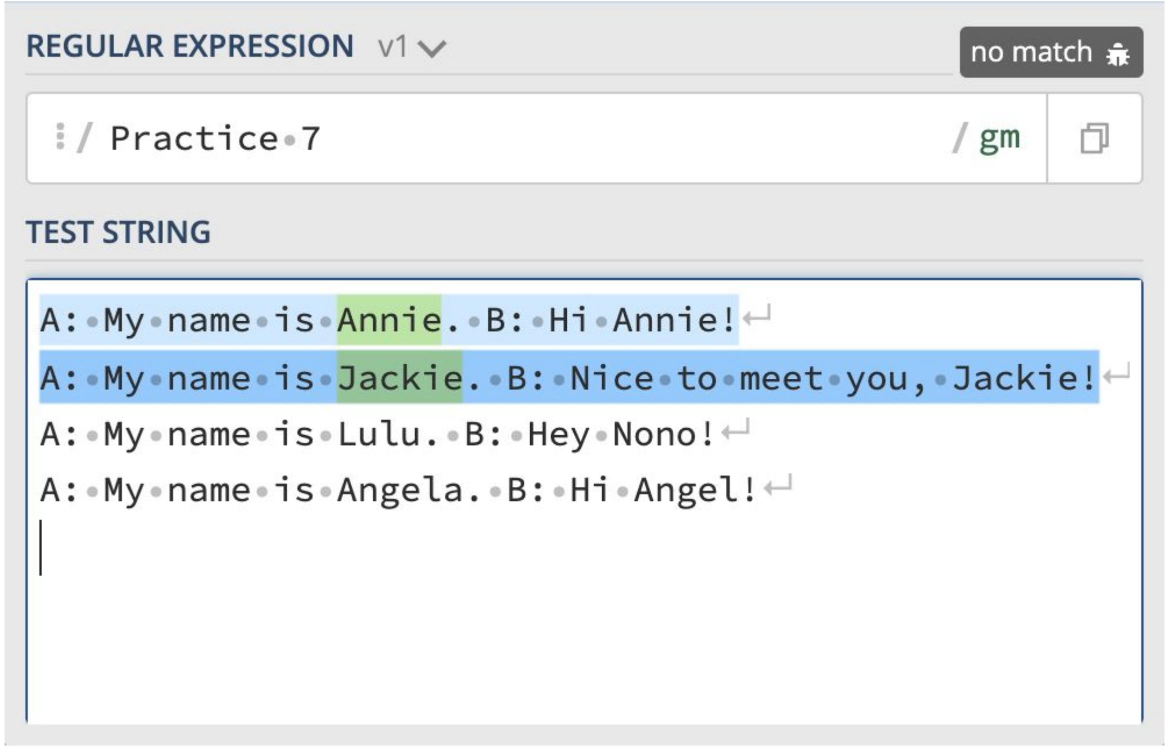
TEST STRING

```
123.a.a
123.123.a.a
123.123.123.a.a
123.456.a.a
123.456.a.b
456.456.456.b.b
```

Capturing Groups

- Some other languages/applications
  - `(abc|def)=\k<1>`
  - `(abc|def)=\g1`
  - `(abc|def)=\g{1}`
  - `(abc|def)=(?P=1)`

Practice Time (1 Problem)



REGULAR EXPRESSION v1

no match

```
/ Practice 7 / gm
```

TEST STRING

```
A: My name is Annie. B: Hi Annie!
A: My name is Jackie. B: Nice to meet you, Jackie!
A: My name is Lulu. B: Hey Nono!
A: My name is Angela. B: Hi Angel!
```

Non Capturing Groups

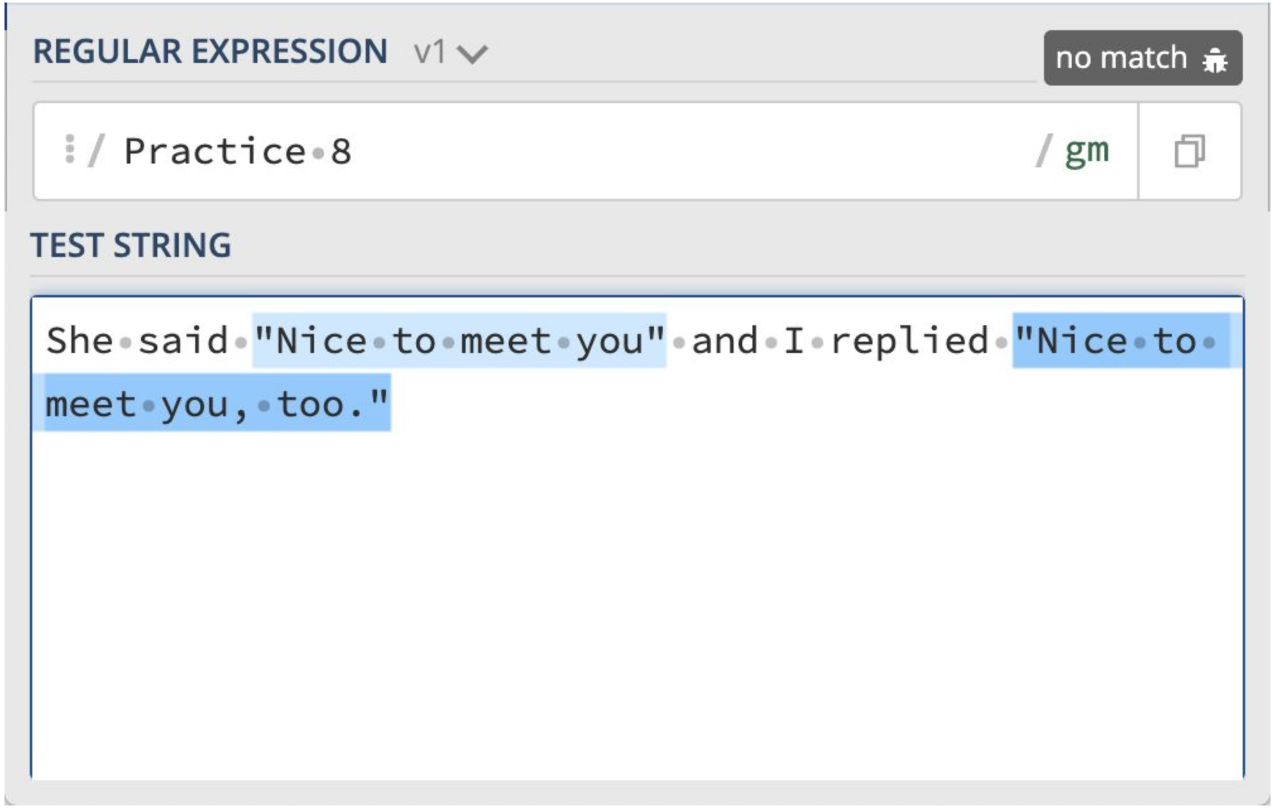
- `?:`

Greedy Match vs Lazy Match

*	*?
+	+?
?	??
a{1,3}	a{1,3}?

- `string = "baaaaaaaaaaa"`
  - `/ba*/ /ba*?/`
  - `/ba+/ /ba+?/`
  - `/ba?/ /ba??/`
  - `/ba{4,6}/ /ba{4,6}?/`

Practice Time (1 Problem)



REGULAR EXPRESSION v1

no match

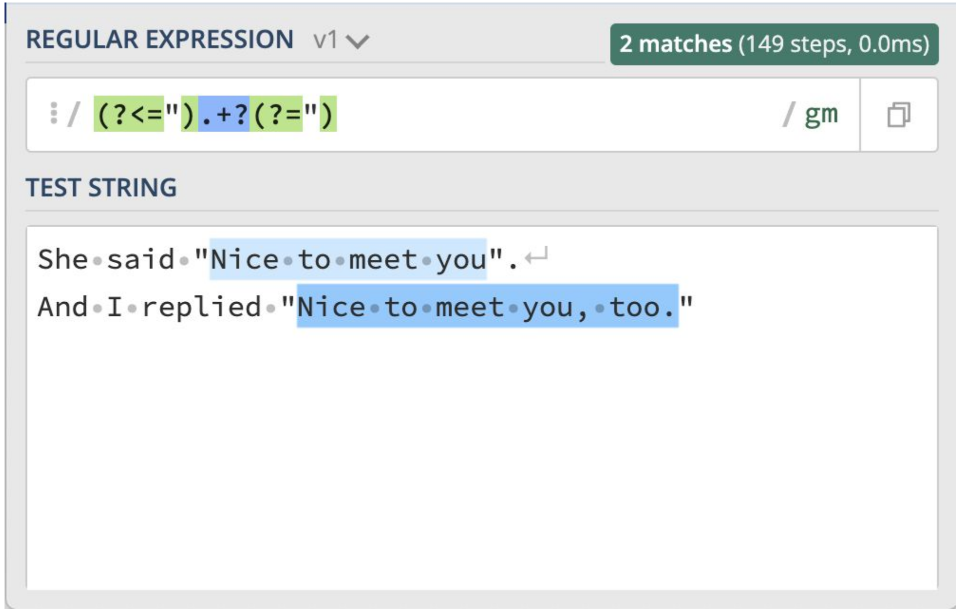
```
/ Practice 8 / gm
```

TEST STRING

```
She said "Nice to meet you" and I replied "Nice to meet you, too."
```

Positive Lookbehind / Lookahead
Negative Lookbehind / Lookahead

(?<=regex)	(?=regex)
(?<!regex)	(?!regex)



REGULAR EXPRESSION v1

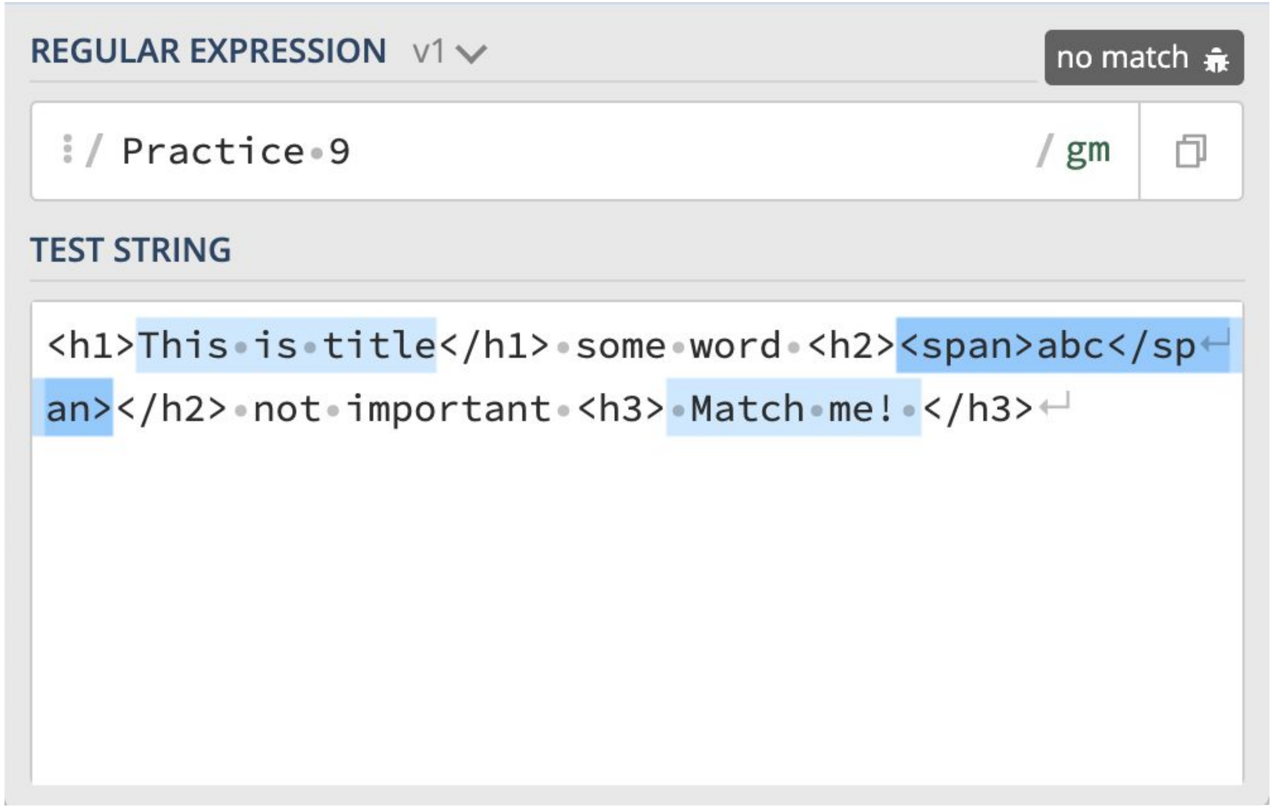
2 matches (149 steps, 0.0ms)

```
/ (?<=").+?(?=") / gm
```

TEST STRING

```
She said "Nice to meet you".
And I replied "Nice to meet you, too."
```

Practice Time (1 Problem)



```
/ Practice 9 / gm
```

```
<h1>This is title</h1> some word <h2><span>abc</span></h2> not important <h3> Match me! </h3>
```

Cheat Sheet

.
+
*
?
{m, n}
^
$
[acd], [^acd]

\d
\w
\D
\W
[[:digit:]]
[[:alpha:]]
[[:alnum:]]
[[:lower:]]

*?
+?
??
a{1,3}?
(?<=regex)
(?=regex)
(?<!regex)
(?!regex)

Python

Methods

	import **re** **regex** = **r**'[a-z]+' **text** = '<vera_wang@gmail.com>' **p** = **re.compile**(regex)	import **re** **regex** = **r**'[a-z]+' **text** = '<vera_wang@gmail.com>'

re.match()	>>> **p.match**(text)
	>>>
	>>> **p.match**(text[1:])
	<re.Match object; span=(0, 4), match='vera'>
	>>> **p.match**(text[1:])**.group**()
	'vera'
		>>> **re.match**(**regex**, text)
		>>>
		>>> **re.match**(**regex**, text[1:])
		<re.Match object; span=(0, 4), match='vera'>
		>>> **re.match**(**regex**, text[1:])**.group**()
		'vera'

re.search()	>>> **p.search**(text)**.group**() 'vera'	>>> **re.search**(**regex**, text)**.group**() 'vera'
re.findall()	>>> **p.findall**(text) ['vera', 'wang', 'gmail', 'com']	>>> **re.findall**(**regex**, text) ['vera', 'wang', 'gmail', 'com']
re.sub()	>>> **p.sub**('**xx**', text) '<xx_xx@xx.xx>'	>>> **re.sub**(**regex**, '**xx**', text) '<xx_xx@xx.xx>'
re.split()	>>> **p.split**(text) ['<', '_', '@', '.', '>']	>>> **re.split**(**regex**, text) ['<', '_', '@', '.', '>']

Methods

	`import re`
	`regex = r'[a-z]+'`
	`text = '<vera_wang@gmail.com>'`
	`p = re.compile(regex)`
		`import re`
		`regex = r'[a-z]+'`
		`text = '<vera_wang@gmail.com>'`

re.match()	`>>> p.match(text)`
	`>>>`
	`>>> p.match(text[1:])`
	`<re.Match object; span=(0`
	`>>> p.match(text[1:]).group()`
	`'vera'`
		`match(regex, text)`
		
		`match(regex, text[1:])`
		`ch object; span=(0, 4), match='vera'>`
		`>>> re.match(regex, text[1:]).group()`

re.search()	`>>> p.search(text).group(`
	`'vera'`
		`search(regex, text).group()`

re.findall()	`>>> p.findall(text)`
	`['vera', 'wang', 'gmail', 'com']`
		`>>> re.findall(regex, text)`
		`['vera', 'wang', 'gmail', 'com']`

re.sub()	`>>> p.sub('xx', text)`
	`'<xx_xx@xx.xx>'`
		`>>> re.sub(regex, 'xx', text)`
		`'<xx_xx@xx.xx>'`

re.split()	`>>> p.split(text)`
	`['<', '_', '@', '.', '>']`
		`>>> re.split(regex, text)`
		`['<', '_', '@', '.', '>']`

- 只能找在開頭的
- 只能找一組
- 要用group()提取

- 只能找一組
- 要用group()提取

Raw String Notation

```
>>> s = '\1'
>>> print(s)

>>> len(s)
1
>>> [ord(c) for c in s]
[1]
>>> re.findall('(.)=\1', 'a=a, b=a')
[]
```

```
>>> s = r'\1'
>>> print(s)
\1
>>> len(s)
2
>>> [ord(c) for c in s]
[92, 49]
>>> re.findall(r'(.)=\1', 'a=a, b=a')
['a']
```

```
>>> s = '\b'
>>> print(s)

>>> len(s)
1
>>> [ord(c) for c in s]
[8]
>>> text = '<vera_wang@$$gmail.##com>'
>>> re.findall('\b.', text)
[]
>>> re.findall('\w+', text)
['vera_wang', 'gmail', 'com']
```

```
>>> s = r'\b'
>>> print(s)
\b
>>> len(s)
2
>>> [ord(c) for c in s]
[92, 98]
>>> text = '<vera_wang@$$gmail.##com>'
>>> re.findall(r'\b.', text)
['v', '@', 'g', '.', 'c', '>']
>>> re.findall(r'\w+', text)
['vera_wang', 'gmail', 'com']
```

```
>>> s = '\w'
>>> len(s)
2
```

Options

re.S	^是全文開頭， $是全文結尾
re.M re.MULTILINE	^是單行開頭， $是單行結尾
re.DOTALL	.能代表換行符號
re.I re.IGNORECASE	忽略字母大小寫

```
>>> text = 'aAaAAa bbBBbb\nAAaaaAa BBBbbb'
>>> print(text)
aAaAAa bbBBbb
AAaaaAa BBBbbb
>>> re.findall(r'b+$', text, flags=re.M|re.I)
['bbBBbb', 'BBBbbb']
```

Python RE library Cheat Sheet

	`import re`
	`regex = r'[a-z]+'`
	`text = '<aaa@bbb.com>\n <xxx@yyy.com>'`
	`p = re.compile(regex)`
		`import re`
		`regex = r'[a-z]+'`
		`text = '<aaa@bbb.com>\n <xxx@yyy.com>'`

re.findall()	`>>> p.findall(text)`
	`['aaa', 'bbb', 'com', 'xxx', 'yyy', 'com']`
		`>>> re.findall(regex, text)`
		`['aaa', 'bbb', 'com', 'xxx', 'yyy', 'com']`
		`>>> re.findall(r'^.....', text, flags=re.M)`
		`['<aaa@', ' <xxx']`
		`>>> re.findall(r'^.....', text, flags=re.S)`
		`['<aaa@']`
		`>>> re.findall(r'^.....', text)  # default: re.S`
		`['<aaa@']`
		`>>> re.findall(r'([a-z]+)@([a-z]+).com', text)`
		`[('aaa', 'bbb'), ('xxx', 'yyy')]`

re.sub()	`>>> p.sub('xx', text)`
	`'<xx@xx.xx>\n <xx@xx.xx>'`
		`>>> re.sub(regex, 'xx', text)`
		`'<xx@xx.xx>\n <xx@xx.xx>'`

re.split()	`>>> p.split(text)`
	`['<', '@', '.', '>\n <', '@', '.', '>']`
		`>>> re.split(regex, text)`
		`['<', '@', '.', '>\n <', '@', '.', '>']`

Ref: https://docs.python.org/3/library/re.html

Practice Time

- Go to Colab
    - Login a google account
    - Execute the 1st block (`cmd+enter` for MacOS / `ctrl+enter` for Windows)
    - Fill in all the ...

grep

Variations

- grep
- egrep / grep -E
- fgrep / grep -F
- pcregrep
- pgrep

```
annie_chang$ echo "aa bbbb+" | grep --color "b+"
aa bbbb+
annie_chang$ echo "aa bbbb+" | grep --color "b\+"
aa bbbb+

annie_chang$ echo "aa bbbb+" | egrep --color "b+"
aa bbbb+
annie_chang$ echo "aa bbbb+" | egrep --color "(?<=aa )b"
egrep: repetition-operator operand invalid

annie_chang$ echo "aa bbbb+" | pcregrep --color "(?<=aa )b"
aa bbbb+

annie_chang$ echo "aa bbbb+" | fgrep --color "b+"
aa bbbb+

annie_chang$ pgrep vim
52286
annie_chang$ ps | grep vim
52286 ttys004    0:00.19 vim /Users/annie_chang/Documents/a.py
75888 ttys048    0:00.00 grep vim
```

Useful Options

- -i, --ignore-case
- -o, --only-matching
- -v, --invert-match
- -n, —line-number
- --color
- -A 3
- -B 3

```
annie_chang$ cat sample.txt
Causes the resulting RE to match.
Repetitions of the preceding RE.
Many repetitions as are possible.
Previous RE should be matched.
ab* will match 'a', 'ab'.

annie_chang$ pcregrep -n --color 'RE' sample.txt
1:Causes the resulting RE to match.
2:Repetitions of the preceding RE.
4:Previous RE should be matched.

annie_chang$ pcregrep -nv 'RE' sample.txt
3:Many repetitions as are possible.
5:ab* will match 'a', 'ab'.
```

Resources & Games

Useful Resources

Powerful Website

- https://www.regular-expressions.info/refrepeat.html
- grep manual

Regex Games

- https://regexone.com/lesson/letters_and_digits
- https://alf.nu/RegexGolf
- http://play.inginf.units.it/#/level/1

Feedback:

https://forms.gle/AQ53Vukv2xF3Ksj89