Regular Expression

SCHOOL OF INFOCOMM

Text

- Building blocks of written language are the symbols such as the alphabets
- A text can be viewed as an arrangement of symbols organized in an intentional and meaningful way
  - E.g R, O, M, T, S ➔ ROMTS ???
  - ➔ STORM ✓✓✓

Structure in Text

- Entities such as company names or names of people
- Date formats
- Email message structure - header followed by subject followed by body
- Visual structure such as tables
- Markup such as HTML or XML

What are Regular Expressions (RegEx)?

- To understand text, we need mechanism for analysing its contents, markups and format
- Regular expressions is the starting point
- A regular expression a specialist notation used to describe string patterns that we want to match
- Example : `/[\w._%+-]+@[\w.-]+\.[a-zA-Z]{2,4}/`

Use Cases for RegEx

- Parsing documents with structured layout

  - HTML, find all text within and tags

- Handling personal identifiable information (PII)

  - Is there something in the text that looks like NRIC?

- Find logical "cut point" of huge corpus of document

  - *Find start and end of sentences*

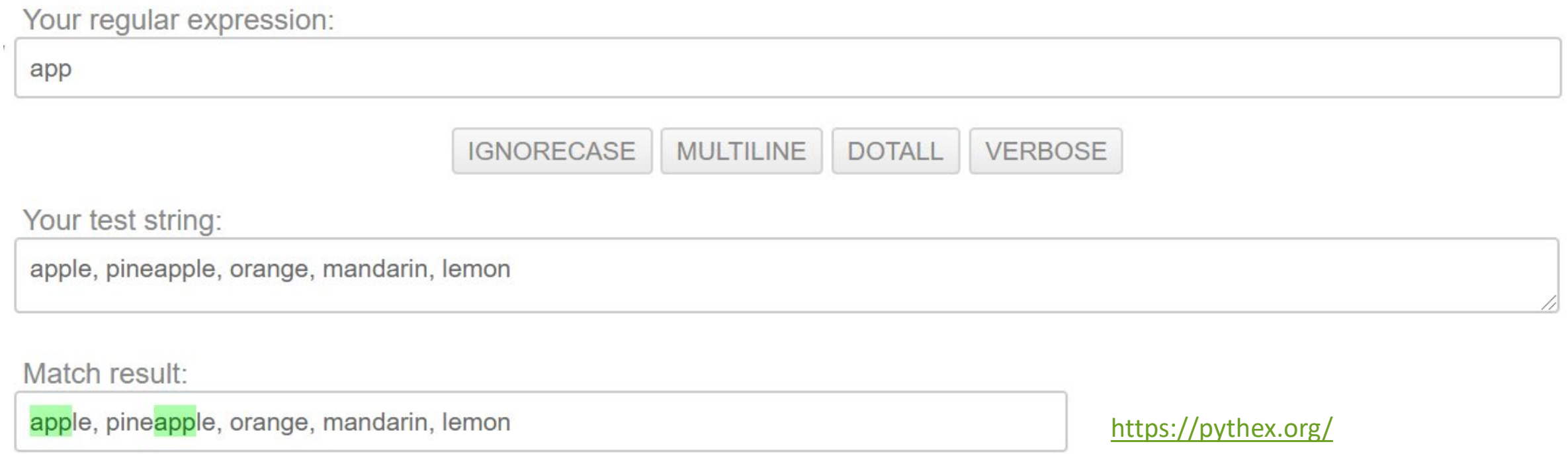
- Replace unwanted characters or hidden text encoding

  - *Line breaks*

Your First Regular Expression

https://pythex.org/

Matching fixed strings



https://pythex.org/

Special Sequence / Symbols

\d	Digits in 0123456789
\D	Non-digits
\w	Letters, digits and _
\W	Non-words
\t	tab
\n	new line
\r	Return
\s	White spaces, including tabs, new lines , return

** The backslash is an escape character*

Finding numbers in text (Attempt #1)

Your regular expression:

\d

IGNORECASE MULTILINE DOTALL VERBOSE

Your test string:

Singtel is a Singaporean multinational telecommunications conglomerate and one of the four major telcos operating in the country, the others being StarHub, M1 and TPG Telecom. The company is the largest mobile network operator in Singapore with 4.1 million subscribers and through subsidiaries, has a combined mobile subscriber base of 640 million customers at the end of financial year

Match result:

Singtel is a Singaporean multinational telecommunications conglomerate and one of the four major telcos operating in the country, the others being StarHub, M1 and TPG Telecom. The company is the largest mobile network operator in Singapore with 4.1 million subscribers and through subsidiaries, has a combined mobile subscriber base of 640 million customers at the end of financial year

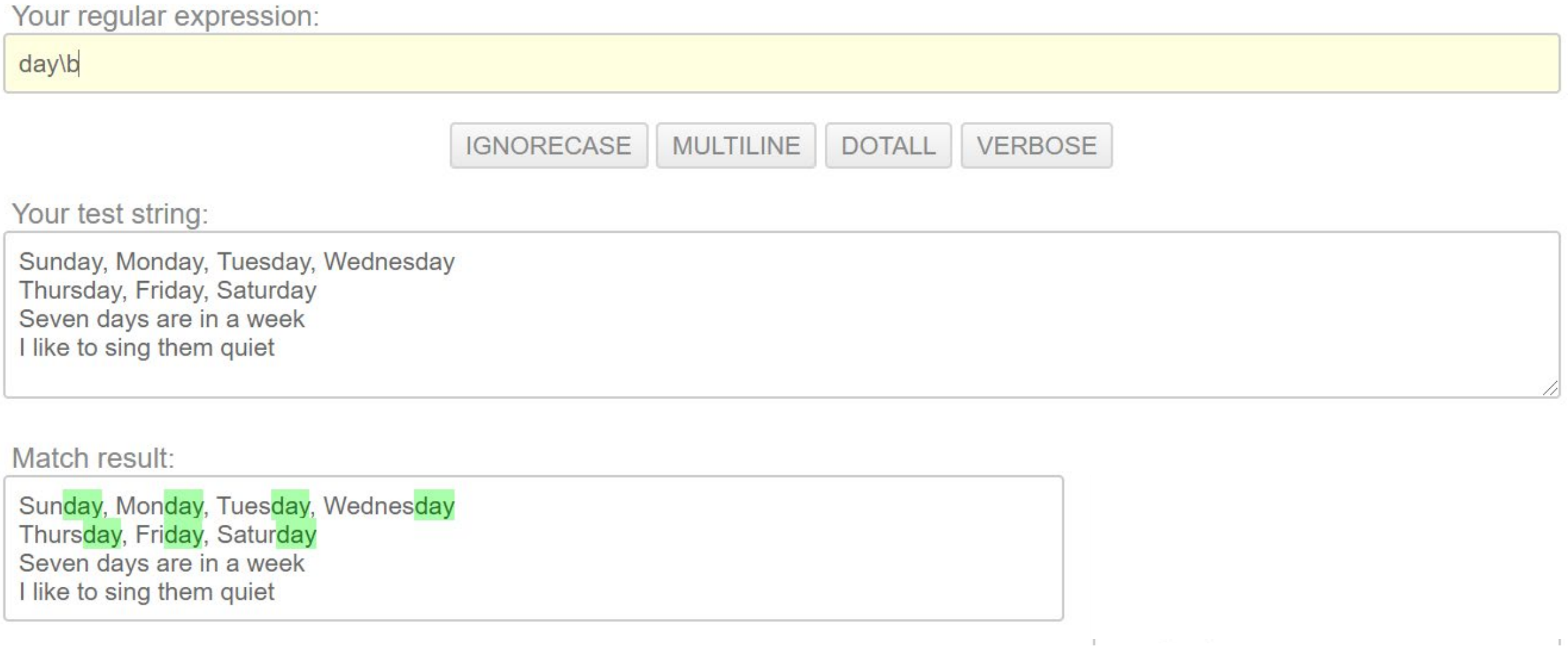
Boundaries

Boundaries characters are used to ‘anchor’ a pattern to some edges

\b	Word boundaries
\B	Non-word boundaries
^	Beginning of line
$	End of line

Matching fixed strings patterns found at a word boundary

"day\b" – find word breaks (\b) ending with **day**



Qualifiers (1)

X*	0 or more repetitions of pattern X
X+	1 or more repetitions of pattern X
X?	0 or 1 instances of pattern X

Finding numbers in text (Attempt #2)

Your regular expression:

```
\b\d+\b
```

IGNORECASE | MULTILINE | DOTALL | VERBOSE

Your test string:

Singtel is a Singaporean multinational telecommunications conglomerate and one of the four major telcos operating in the country, the others being StarHub, M1 and TPG Telecom. The company is the largest mobile network operator in Singapore with 4.1 million subscribers and through subsidiaries, has a combined mobile subscriber base of 640 million customers at the end of financial year .

Match result:

Singtel is a Singaporean multinational telecommunications conglomerate and one of the four major telcos operating in the country, the others being StarHub, M1 and TPG Telecom. The company is the largest mobile network operator in Singapore with 4 . 1 million subscribers and through subsidiaries, has a combined mobile subscriber base of 640 million customers at the end of financial year .

Finding numbers in text (Attempt #3)

Your regular expression:

```
\b\d+.?\d+\b
```

IGNORECASE | MULTILINE | DOTALL | VERBOSE

Your test string:

Singtel is a Singaporean multinational telecommunications conglomerate and one of the four major telcos operating in the country, the others being StarHub, M1 and TPG Telecom. The company is the largest mobile network operator in Singapore with 4.1 million subscribers and through subsidiaries, has a combined mobile subscriber base of 640 million customers at the end of financial year .

Match result:

Singtel is a Singaporean multinational telecommunications conglomerate and one of the four major telcos operating in the country, the others being StarHub, M1 and TPG Telecom. The company is the largest mobile network operator in Singapore with 4.1 million subscribers and through subsidiaries, has a combined mobile subscriber base of 640 million customers at the end of financial year .

Qualifiers (2)

X{m}	Exactly m instances of X
X{m, n}	Between m and n instances of X
X{m, }	At least m instances of X

Finding numbers with certain size

Your regular expression:

\d{8,8}

IGNORECASE MULTILINE DOTALL VERBOSE

Your test string:

Please pick me up at postal code 12345.
Please call me at 90012304 or 23410099.

Match result:

Please pick me up at postal code 12345.
Please call me at 90012304 or 23410099.

Characters Class

[……..] Use to express a set of choices between individual characters

[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
[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

Finding numbers with certain size AND some pattern

Your regular expression:

```
[896]\d{7,7}
```

IGNORECASE | MULTILINE | DOTALL | VERBOSE

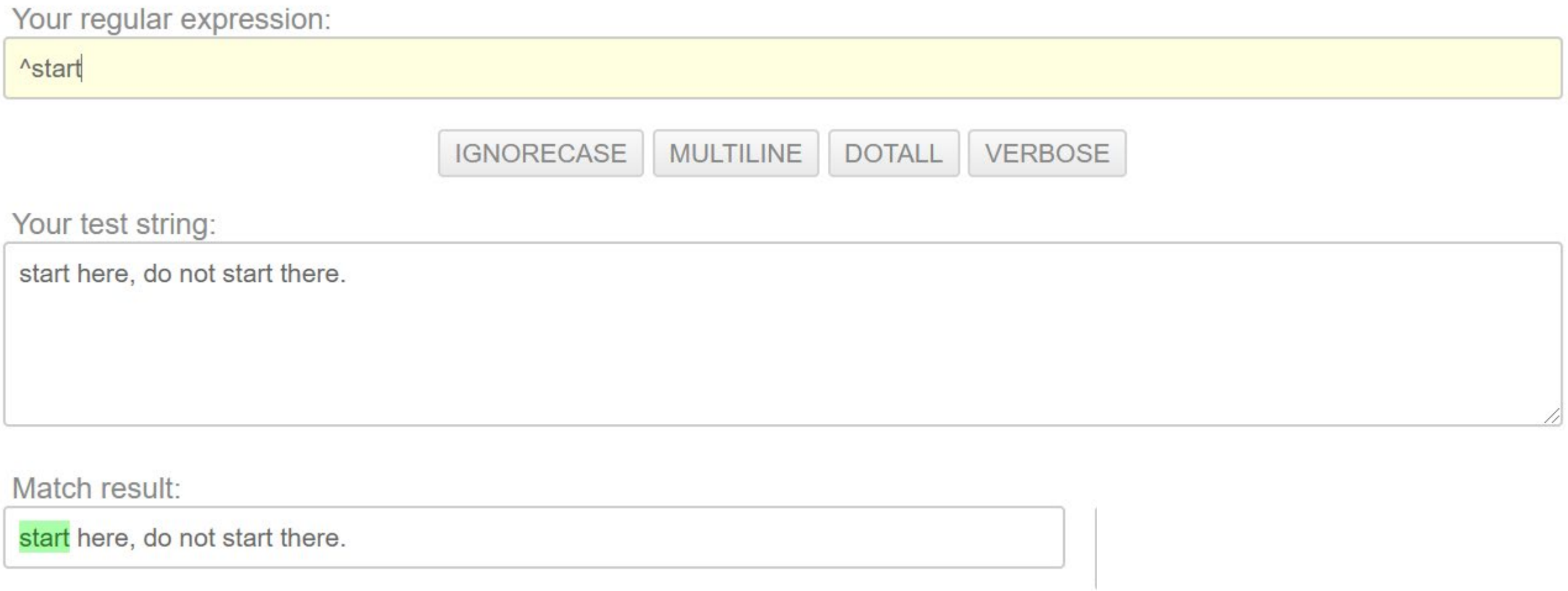
Your test string:

```
Please pick me up at postal code 12345.
Please call me at 90012304 or 23410099.
```

Match result:

```
Please pick me up at postal code 12345.
Please call me at 90012304 or 23410099.
```

Find a sentence starting with a pattern



Find a sentence ending with a pattern

Your regular expression:

end$

IGNORECASE MULTILINE DOTALL VERBOSE

Your test string:

Here is not the end but there is the end

Match result:

Here is not the end but there is the end

Scoping / Group

(....) Use to express a sequence of symbols that must be together

(abc)	Returns a match that contains ‘abc’
(abc)+	Returns a match that contains abc or abcabc or abcabcabc

Conditions (alternatives)

- | Use to express matches with two or more regular expression

X | Y	Returns a match that contains pattern X or pattern Y
X | Y | Z	Returns a match that contains pattern X or pattern Y or pattern Z

Find some mentions of names

Your regular expression:

(Singtel) | (TPG) | (M1)

IGNORECASE MULTILINE DOTALL VERBOSE

Your test string:

Singtel is a Singaporean multinational telecommunications conglomerate and one of the four major telcos operating in the country, the others being StarHub, M1 and TPG Telecom. The company is the largest mobile network operator in Singapore with 4.1 million subscribers and through subsidiaries, has a combined mobile subscriber base of 640 million customers at the end of financial year

Match result:

Singtel is a Singaporean multinational telecommunications conglomerate and one of the four major telcos operating in the country, the others being StarHub, M1 and TPG Telecom. The company is the largest mobile network operator in Singapore with 4.1 million subscribers and through subsidiaries, has a combined mobile subscriber base of 640 million customers at the end of financial year

Exercise

Appreciation of RegEx syntax

https://regexone.com/

Lesson 1: An Introduction, and the ABCs
Lesson 1½: The 123s
Lesson 2: The Dot
Lesson 3: Matching specific characters
Lesson 4: Excluding specific characters
Lesson 5: Character ranges
Lesson 6: Catching some zzz's
Lesson 7: Mr. Kleene, Mr. Kleene
Lesson 8: Characters optional
Lesson 9: All this whitespace
Lesson 10: Starting and ending
Lesson 11: Match groups

Problem 1: Matching a decimal numbers
Problem 2: Matching phone numbers
Problem 3: Matching emails

Instructions:
Complete Lesson 1 to Lesson 12.
Complete Problem 1 to Problem 3.

Check the solution only if you must

Regex Related Tasks

Many languages allow programmers to define regex and then use them to:

- **Validate** that a piece of text (or a portion of that text) matches some pattern
- **Find** fragments of some text that match some pattern
- **Extract** fragments of some text
- **Replace** fragments of text with other text

Regular Expression in Python

- Module **re** – part of the standard Python library that provides text process services
- Both patterns and strings to be searched can be Unicode (strings) as well as 8-bit strings (bytes)
- Backslash character ('\') to indicate special forms or to allow special characters to be used without invoking their special meaning

Search Method

`re.search(pattern, string):`

Scan through ***string*** looking for the **first location** where the regular expression ***pattern*** produces a match and return a corresponding ***match object***

The **match object** contains information about the search and the result.
`.span()` returns a tuple containing the start-, and end positions of the match.
`.group()` returns the part of the string where there was a match
.string returns the string passed into the function

Example 1A

```
1  pattern = "dividend"
2
3  string = """A special dividend is a non-recurring distribution of company assets,
4  usually in the form of cash, to shareholders. A special dividend is usually larger
5  compared to normal dividends paid out by the company and often tied to a
6  specific event like an asset sale or other windfall event."""
7
8  result = re.search(pattern, string)
```

```
1  print(result.span())
2  print(result.group())
```

```
(10, 18)
dividend
```

Refer to notebook : re-example-1

Example 1B

```
1  pattern = "DIVIDEND"
2
3  string = """A special dividend is a non-recurring distribution of company assets,
4  usually in the form of cash, to shareholders. A special dividend is usually larger
5  compared to normal dividends paid out by the company and often tied to a
6  specific event like an asset sale or other windfall event."""
7
8  result = re.search(pattern, string)
```

```
1  print (result)
```

```
None
```

In this example, the match object is None because _____

Example 1C

```
1  pattern = "DIVIDEND"
2
3  string = """A special dividend is a non-recurring distribution of company assets,
4  usually in the form of cash, to shareholders. A special dividend is usually larger
5  compared to normal dividends paid out by the company and often tied to a
6  specific event like an asset sale or other windfall event."""
7
8  result = re.search(pattern, string, flags=re.IGNORECASE)
```

```
1  print (result)
```

```
<_sre.SRE_Match object; span=(10, 18), match='dividend'>
```

In this example, the flag _____ is passed into the .search() method to tell RE to ignore case differences

Findall Method

```
re.findall(pattern, string):
```

Scan through ***string*** looking for all **matches of the** regular expression ***pattern*** and return ***a list of text fragments*** that match the pattern

Example 2

```
1  string = """Including the interim dividend of 10 cents per share paid
2  on 3 December 2013, the proposed final dividend of 11 cents per share
3  and the proposed special dividend of 25 cents per share to be paid on
4  14 August 2014, the total dividend for the 2013-14 financial year will
5  be 46 cents per share. """
6
7  pattern = "\d+ cents per share"
8
9  result = re.findall(pattern, string, flags=re.IGNORECASE)
```

```
1  print (result)
```

```
['10 cents per share', '11 cents per share', '25 cents per share', '46 cents per share']
```

Refer to notebook : re-example-2

Compiling Regular Expression

- A regex can be compiled into an object that can be used for faster validation, extraction, and replacing.

- The text to be processed is then passed into the compiled object by by calling its associated methods such as `findall(), search()` etc

Example 3A

```
1  pattern = "\d+ cents per share"
2  compiled_re = re.compile(pattern)
```

```
1  string = """Including the interim dividend of
2  10 cents per share paid on 3 December 2013, the proposed final dividend
3  of 11 cents per share and the proposed special dividend of 25 cents per share
4  to be paid on 14 August 2014, the total dividend for the
5  2013-14 financial year will be 46 cents per share. """
```

```
1  result = compiled_re.findall(string)
```

```
1  print (result)
```

```
['10 cents per share', '11 cents per share', '25 cents per share', '46 cents per share']
```

Refer to notebook : re-example-3

Example 3B

Compiling a regular expression is useful when processing a huge corpus of text where the same regular expression is applied across multiple documents.

```
pattern = "\d+ cents per share"
compiled_re = re.compile(pattern)
```

```
import pandas as pd
df = pd.read_csv("data/dividend_statements.csv")
```

```
df["dividend"] = "empty"
```

```
for ind in df.index:
    df["dividend"][ind] = compiled_re.findall(df["dividend_text"][ind])
```

```
df.head()
```

	company	dividend_text	dividend
0	ABC	Including the interim dividend of 10 cents per...	[10 cents per share, 11 cents per share, 25 ce...
1	XYZ	Together with the interim dividend of 12 cents...	[12 cents per share, 10 cents per share]
2	PQR	Including the interim dividend of 34 cents per...	[34 cents per share, 11 cents per share, 25 ce...

Substitute Method

`re.sub(pattern, replace_string, string)`:

Use to replace occurrences of a particular sub-string (***pattern***) with another sub-string (***replace_string***). It returns the new string.

Use case: mask out sensitive information from text

Example 4

```
1  pattern = "[S|T]\d{5,8}[A-Z]"
2  mask = "<SECRET>"
```

```
1  string = """This medical report for medical insurance claims
2  for patient PETER HO with NRIC S6712098J. Please treat this report as confidential."""
```

```
1  result = re.sub(pattern, mask, string)
2  print (result)
```

```
This medical report for medical insurance claims
for patient PETER HO with NRIC <SECRET>. Please treat this report as confidential.
```

Refer to notebook : re-example-4

Split Method

`re.split(pattern, string)`:

Use to split ***string*** by the occurrences of ***pattern***. It returns a list of the various text segments of the original string.

Use case: break huge amount of text into smaller text segments

Example 5

```
1 string = """Mr. Smith bought cheapsite.com for 1.5 million dollars.
2 He paid a lot for it. Did he mind? Adam Jones thinks he didn't.
3 In any case, this isn't true... Well, with a probability of .9, it isn't! Do you agree?"""
```

```
1 pattern = "(?<!\w\.\w.)(?<![A-Z][a-z]\.)(?<=\.|\?)\s"
```

```
1 result = re.split(pattern, string)
2 print(result)
```

```
['Mr. Smith bought cheapsite.com for 1.5 million dollars.', '\nHe paid a lot for it.', 'Did h
e mind?', "Adam Jones thinks he didn't.", "\nIn any case, this isn't true...", "Well, with a
probability of .9, it isn't! Do you agree?"]
```

How do you remove the \n from the result?

Refer to notebook : re-example-5

Common Problem – meta characters in text

```
It is a smilie :) Many people are too lazy to type it.
Multiple "))))" do not mean laughing, it's just a more
friendly smile. :))
```

The text above carries a special meta character **)**.
How do we search for the smilie?

If we are not careful in writing the regular expression, it can be interpreted as the close of a parameter or perform special function.

Use **** to inhibit the "specialness" of a character. If you are unsure if a character has special meaning, you can put a slash in front of it, such as **\)**, to make sure it is treated just as a character.

Example 6

```
string = """It is a smilie :) Many people are too lazy to type it.
Multiple "))))" do not mean laughing, it's just a more friendly smile. :))
"""
```

```
good_pattern = ":\)+"
result = re.findall(good_pattern, string)
print(result)
```

With Escape \

```
[':)', ':))']
```

```
poor_pattern = ":)+"
result = re.findall(poor_pattern, string)
print(result)
```

Without Escape \

```
---------------------------------------------------------------------------
error                                     Traceback (most recent call last)
```

Refer to notebook : re-example-6

Exercise

Use regular expression to gain preliminary insights into a large corpus of data

Proceed with Jupyter Notebook

re-ex-sherlockhomes.ipynb

Optional Video

Regular Expressions Lecture — [NLP || Dan Jurafsky || Stanford University] - https://youtu.be/EyzTQ0OKeNw

Regular Expressions in Practical NLP - https://youtu.be/k242_PpMEsQ

References

- Python RE module documentation, https://docs.python.org/3/library/re.html
- Regular expression in Python, https://www.w3schools.com/python/python_regex.asp
- Python Regular Expressions, https://developers.google.com/edu/python/regular-expressions