ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN
Vietnam - Korea University of Information and Communication Technology

Regular Expressions

Natural Language Processing

http://vku.udn.vn/

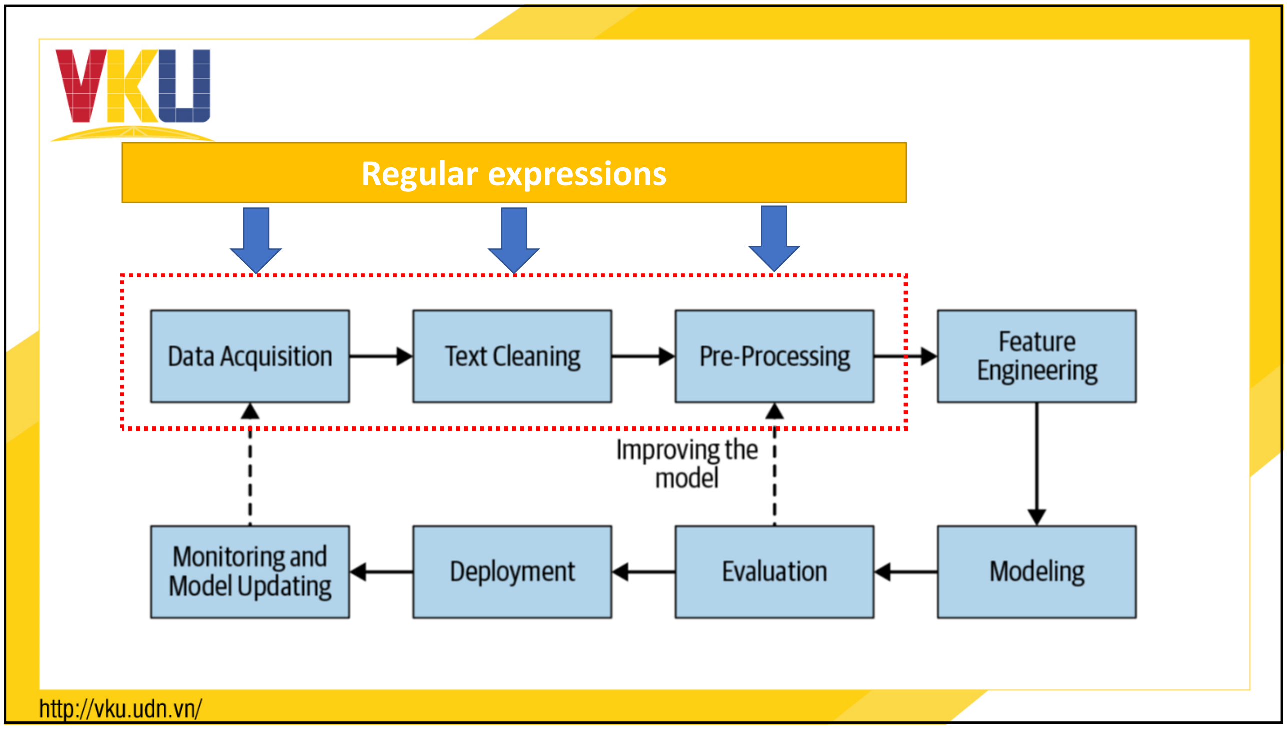
1

```
(http|ftp|https)://([\w_-]+(?:(?:\.[\w_-]+)+))([\w.,@?^=%&:/~+#-]*[\w@?^=%&/~+#-])?
```

HOW TO REGEX
STEP 1: OPEN YOUR FAVORITE EDITOR
@GARABATOKID
STEP 2: LET YOUR CAT PLAY ON YOUR KEYBOARD
/^([A-Z0-9_\.-

http://vku.udn.vn/

2



Regular expressions

Data Acquisition

Text Cleaning

Pre-Processing

Feature Engineering

Modeling

Evaluation

Deployment

Monitoring and Model Updating

Improving the model

http://vku.udn.vn/

3

Regular expressions

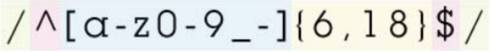
- How can we search for any of these?
  - Covid-19
  - covid-19
  - Covid 19
  - covid 19
  - covid19
  - ~~Covid in 19th century~~

http://vku.udn.vn/

4

What exactly are regular expressions?

- Regular expressions:
  - **strings** with a **special syntax**
  - allows us to **match patterns and find other strings**

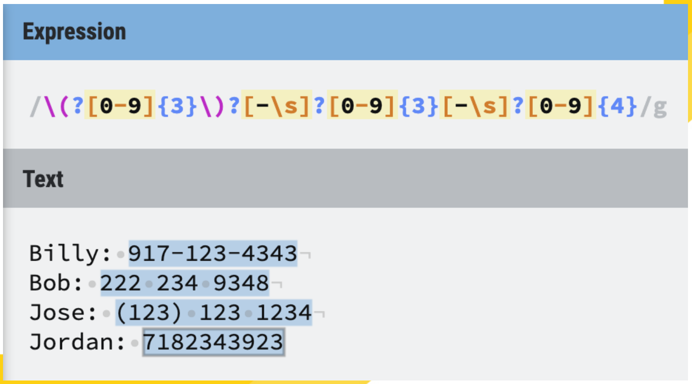


```
/^[a-z0-9_-]{6,18}$/
```

- Find all **web links** in a document
- Parse **email addresses**

  phone numbers
- Remove/replace

  unwanted characters



```
/\(?[0-9]{3}\)?[-\s]?[0-9]{3}[-\s]?[0-9]{4}/g
```

```
Billy: 917-123-4343
Bob: 222 234 9348
Jose: (123) 123 1234
Jordan: 7182343923
```

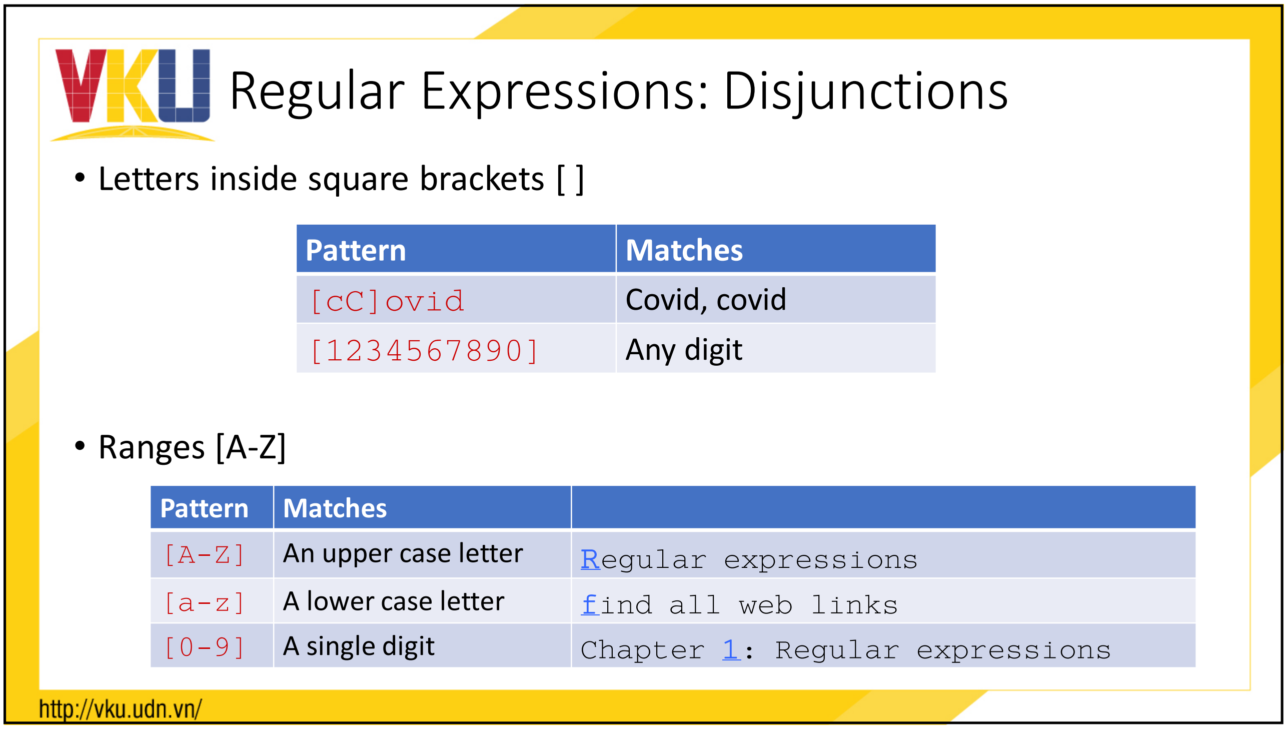
Python library

- \`re\` library → **import re**
- Match a substring by using the **re.match** method

```
if re.match('abc', 'abcdef’):
       print("String contains substring 'abc'”)
Output:  String contains substring 'abc’
```

- Match a **word (\w+)** (the first word in the string)

```
word_regex = '\w+’
re.match(word_regex, 'hi there!’).group()
Output: hi
```



Regular Expressions: Disjunctions

- Letters inside square brackets [ ]

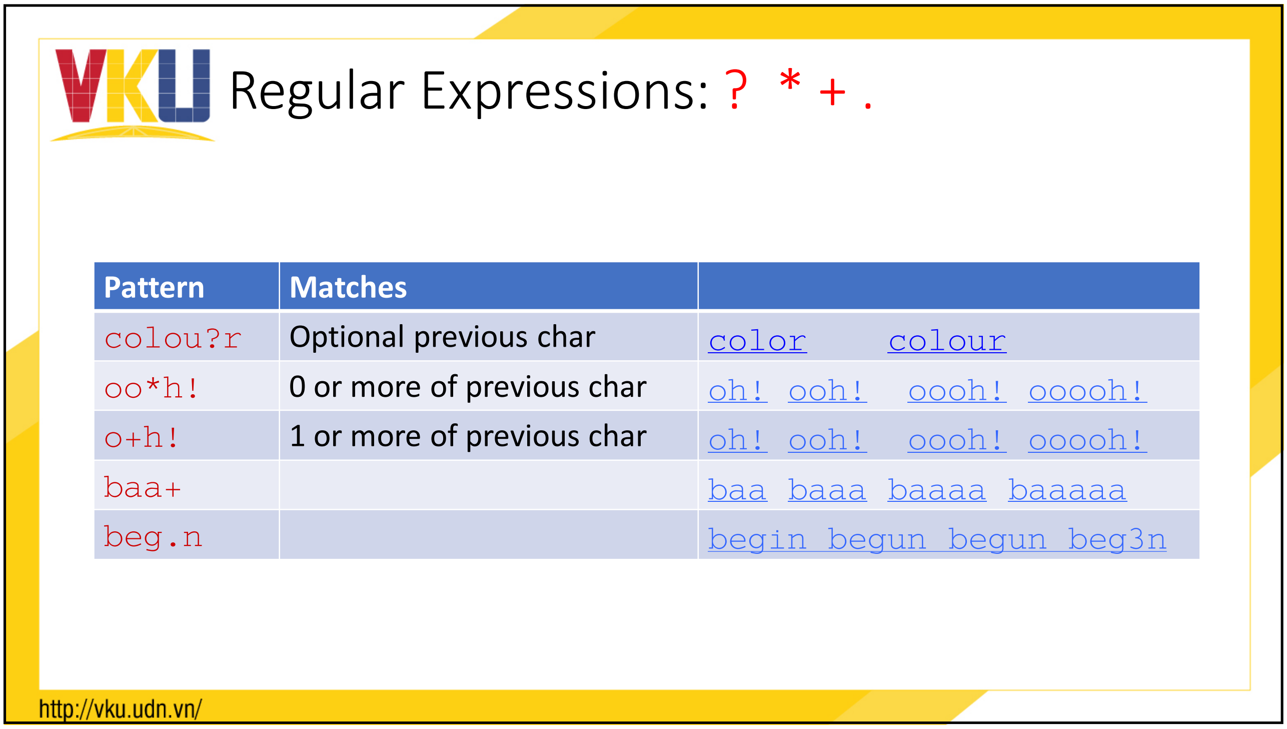
Pattern	Matches
[cC]ovid	Covid, covid
[1234567890]	Any digit

- Ranges [A-Z]

Pattern	Matches	
[A-Z]	An upper case letter	Regular expressions
[a-z]	A lower case letter	find all web links
[0-9]	A single digit	Chapter 1: Regular expressions

http://vku.udn.vn/

7

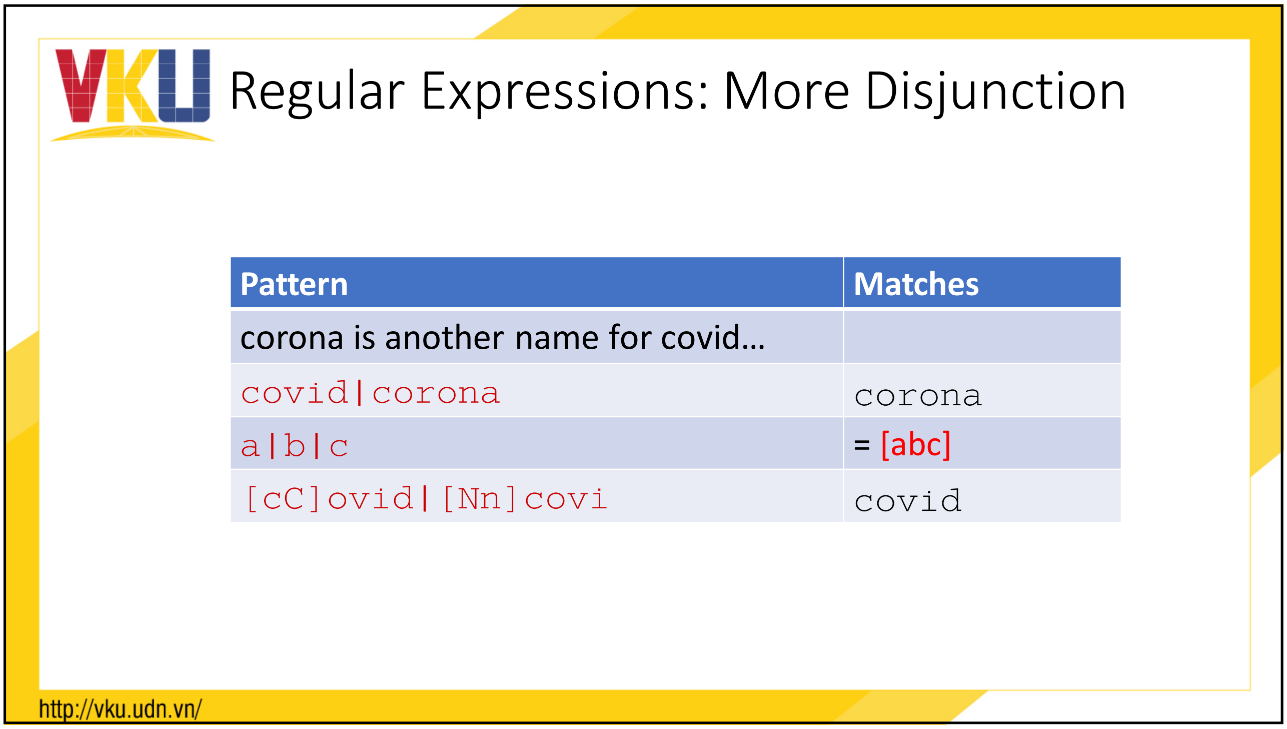


Regular Expressions: ? * + .

Pattern	Matches	
colou?r	Optional previous char	color colour
oo*h!	0 or more of previous char	oh! ooh! oooh! ooooh!
o+h!	1 or more of previous char	oh! ooh! oooh! ooooh!
baa+		baa baaa baaaa baaaaa
beg.n		begin begun begun beg3n

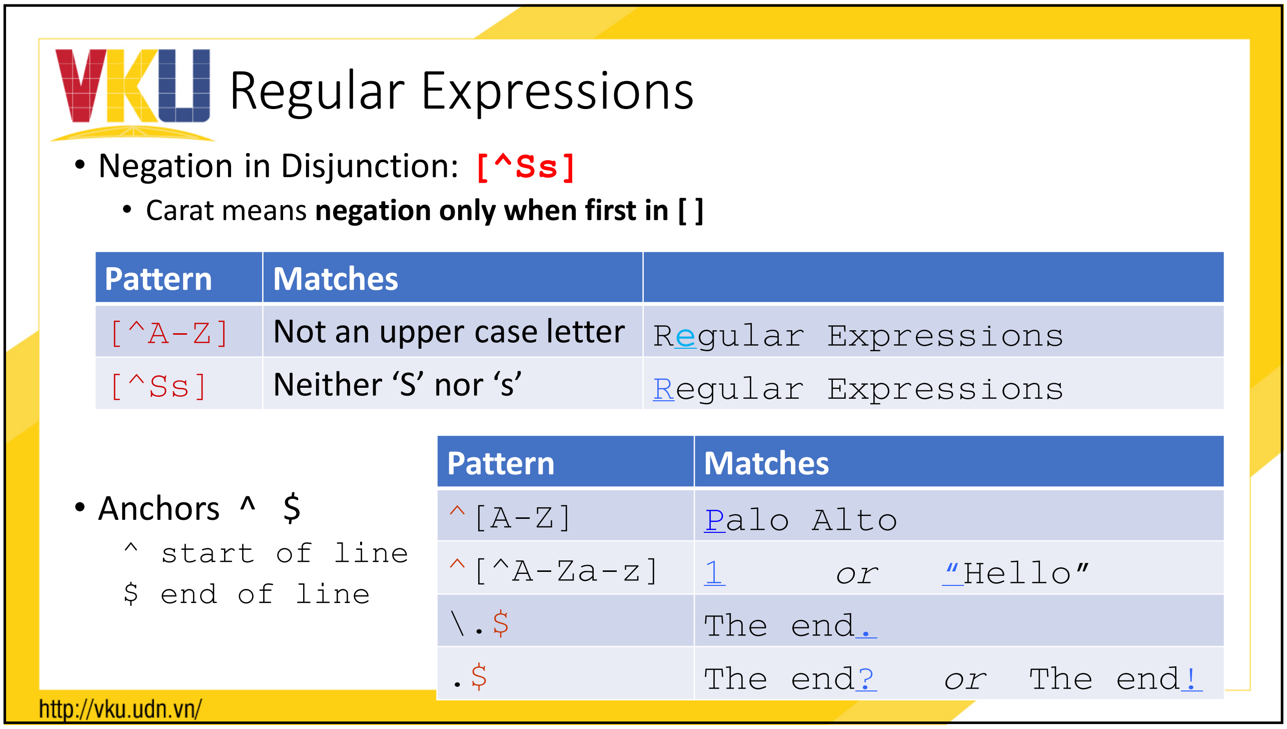
http://vku.udn.vn/

8



Regular Expressions: More Disjunction

Pattern	Matches
corona is another name for covid…	
covid|corona	corona
a|b|c	= [abc]
[cC]ovid|[Nn]covi	covid



Regular Expressions

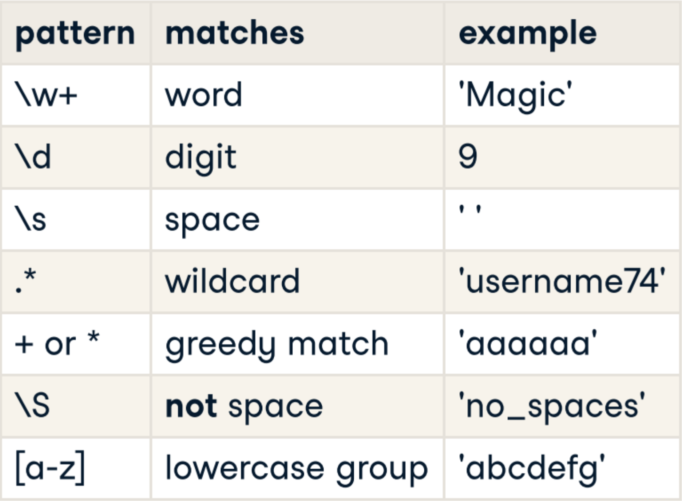
- Negation in Disjunction: **[^Ss]**
  - Carat means **negation only when first in [ ]**

Pattern	Matches	
[^A-Z]	Not an upper case letter	Regular Expressions
[^Ss]	Neither 'S' nor 's'	Regular Expressions

- Anchors ^ $
  - ^ start of line
  - $ end of line

Pattern	Matches
^[A-Z]	Palo Alto
^[^A-Za-z]	1 *or* "Hello"
\.$	The end.
.$	The end? *or* The end!

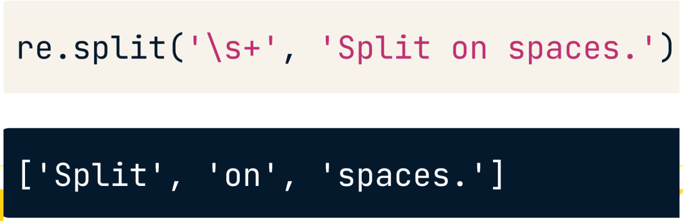
Common regex patterns



pattern	matches	example
\w+	word	'Magic'
\d	digit	9
\s	space	' '
.*	wildcard	'username74'
+ or *	greedy match	'aaaaaa'
\S	**not** space	'no_spaces'
[a-z]	lowercase group	'abcdefg'

Python's **re module**

- **re** module
- **split** : split a string on regex
- **findall** : find all patterns in a string
- **search** : search for a pattern
- **match** : match an entire string or substring based on a pattern
- **sub**: replace the matches with the text of your choice
- May return an iterator, string, or match object



```
re.split('\s+', 'Split on spaces.')
```

```
['Split', 'on', 'spaces.']
```

References

- https://docs.python.org/3/library/re.html
- https://www.w3schools.com/python/python_regex.asp

http://vku.udn.vn/

13

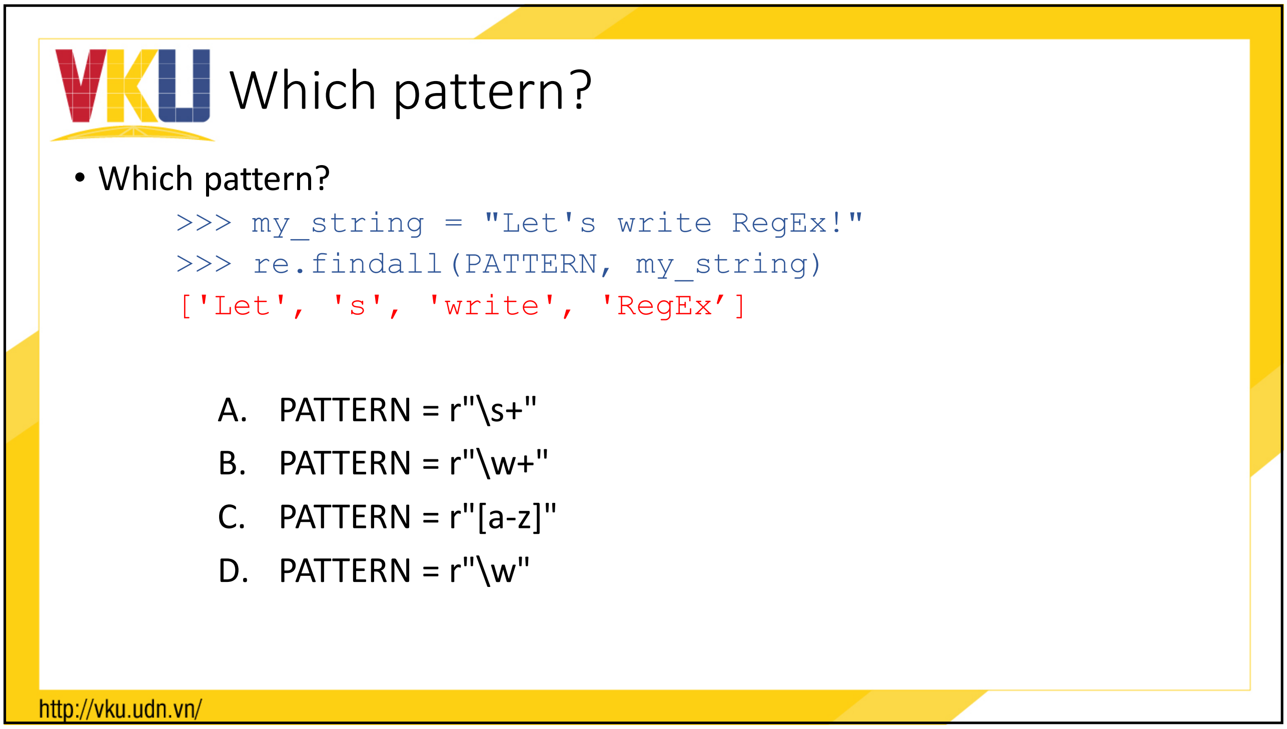
ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN
Vietnam - Korea University of Information and Communication Technology

Let's practice!

Regular Expressions

http://vku.udn.vn/

14



Which pattern?

- Which pattern?

```
>>> my_string = "Let's write RegEx!"
>>> re.findall(PATTERN, my_string)
['Let', 's', 'write', 'RegEx’]
```

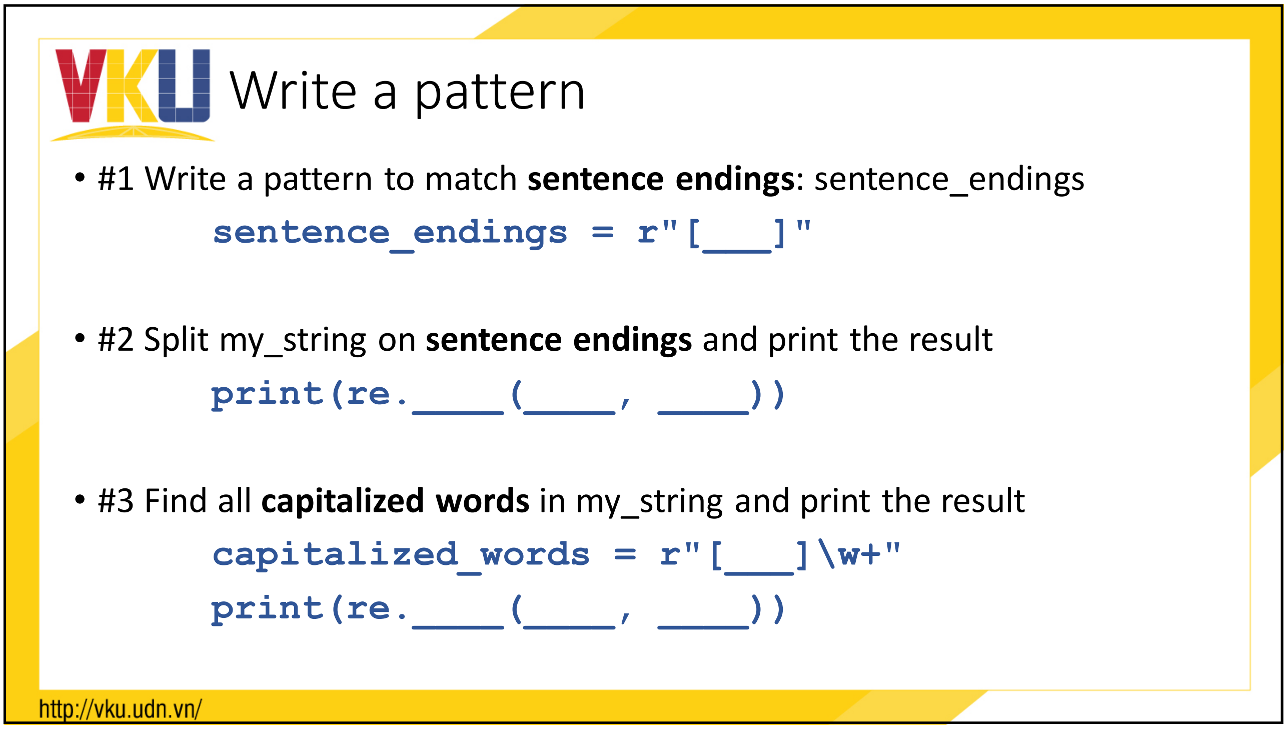
A. PATTERN = r"\s+"

B. PATTERN = r"\w+"

C. PATTERN = r"[a-z]"

D. PATTERN = r"\w"

15



Write a pattern

- #1 Write a pattern to match **sentence endings**: sentence_endings

```
sentence_endings = r"[___]"
```

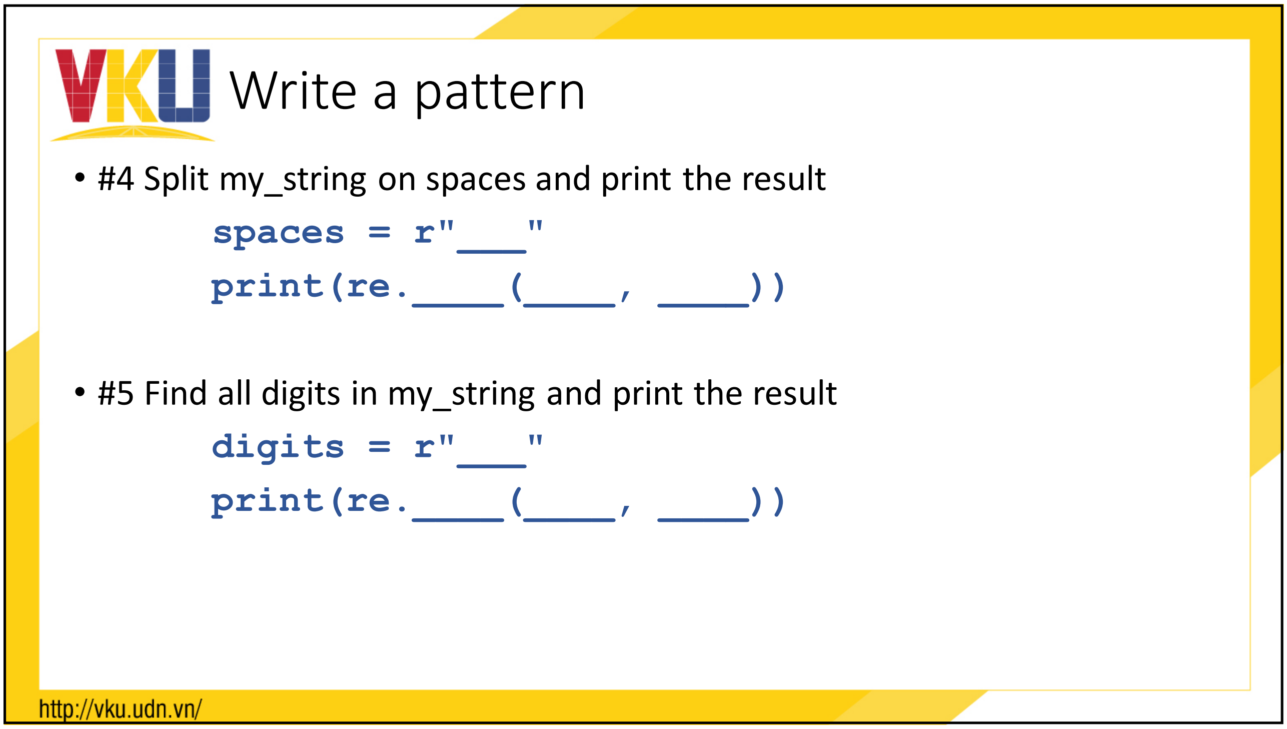
- #2 Split my_string on **sentence endings** and print the result

```
print(re.____(____, ____))
```

- #3 Find all **capitalized words** in my_string and print the result

```
capitalized_words = r"[___]\w+"
print(re.____(____, ____))
```

16



Write a pattern

- #4 Split my_string on spaces and print the result

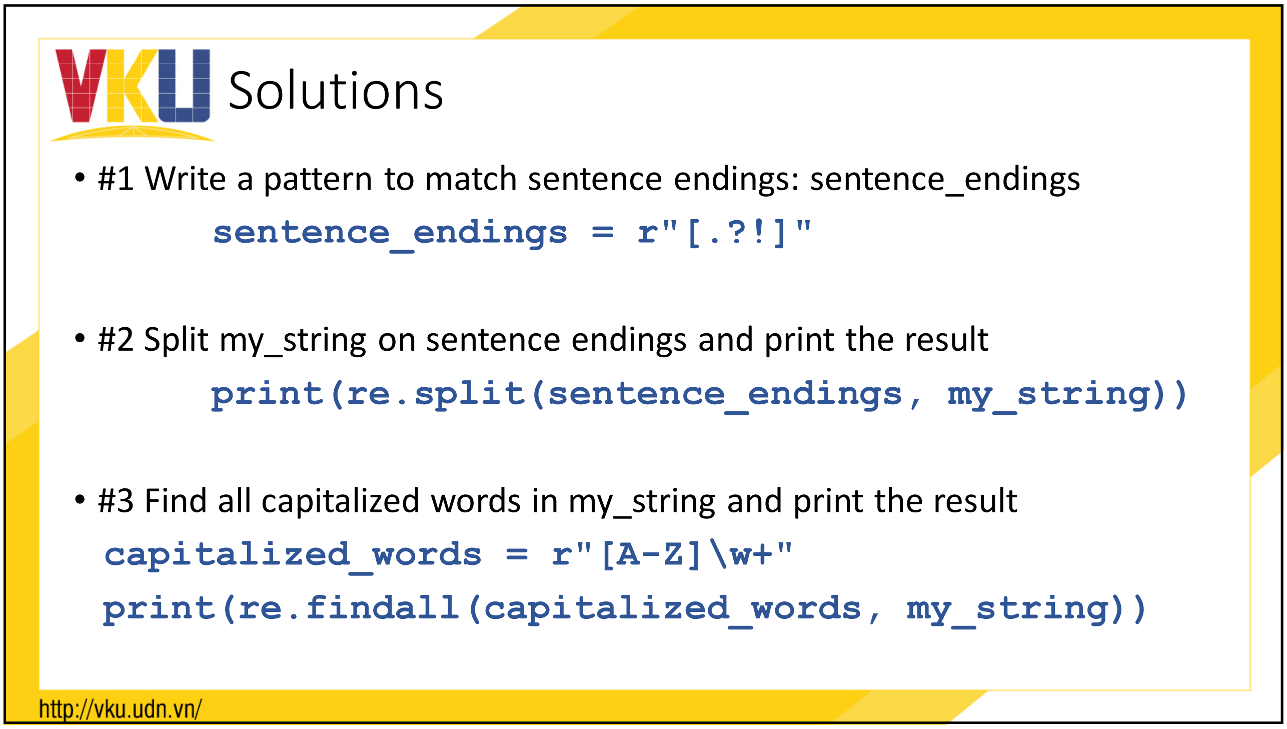
```
spaces = r"___"
print(re.____(____, ____))
```

- #5 Find all digits in my_string and print the result

```
digits = r"___"
print(re.____(____, ____))
```

http://vku.udn.vn/

17



Solutions

- #1 Write a pattern to match sentence endings: sentence_endings

```
sentence_endings = r"[.?!]"
```

- #2 Split my_string on sentence endings and print the result

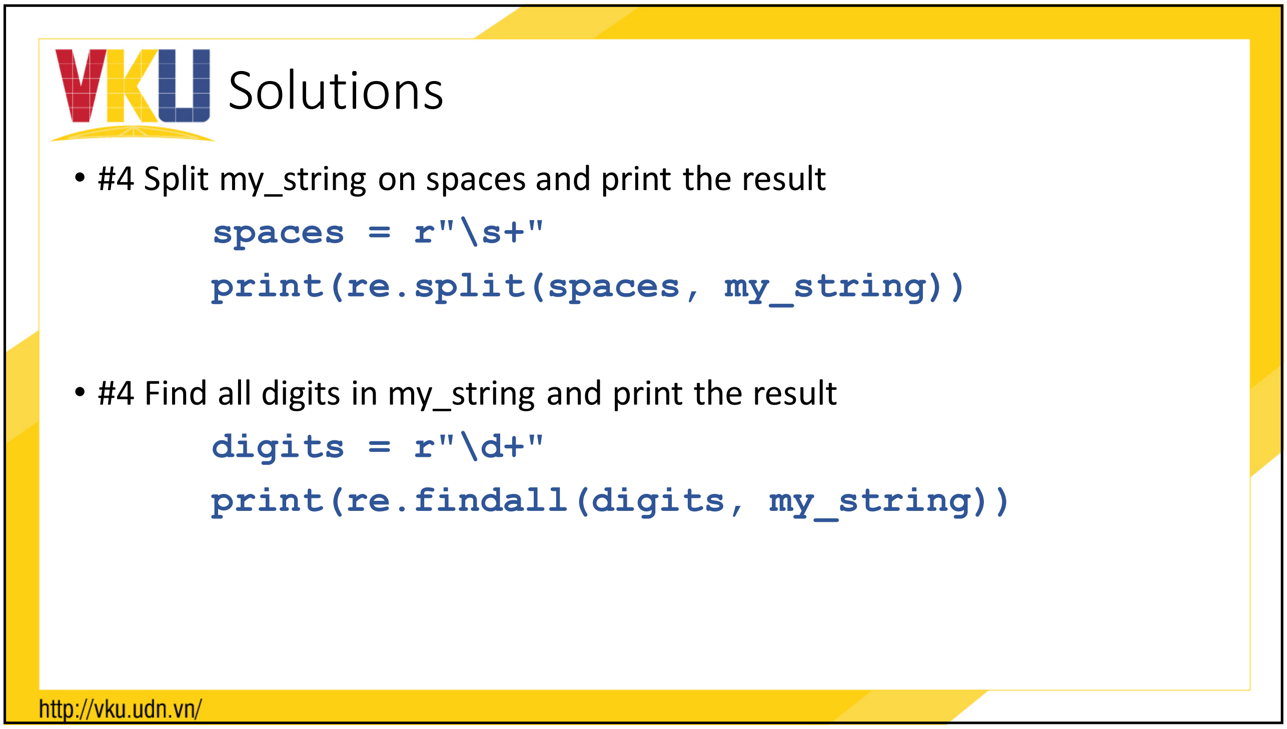
```
print(re.split(sentence_endings, my_string))
```

- #3 Find all capitalized words in my_string and print the result

```
capitalized_words = r"[A-Z]\w+"
print(re.findall(capitalized_words, my_string))
```

http://vku.udn.vn/

18



Solutions

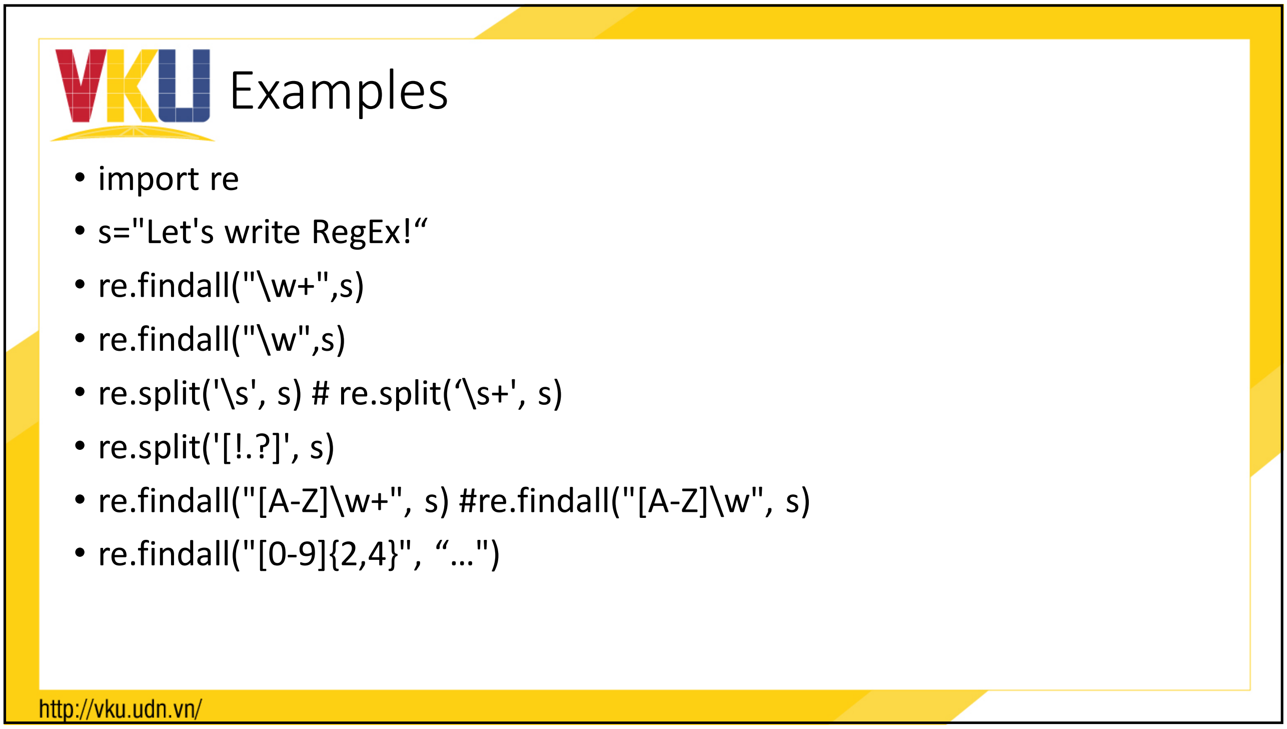
- #4 Split my_string on spaces and print the result

```
spaces = r"\s+"
print(re.split(spaces, my_string))
```

- #4 Find all digits in my_string and print the result

```
digits = r"\d+"
print(re.findall(digits, my_string))
```

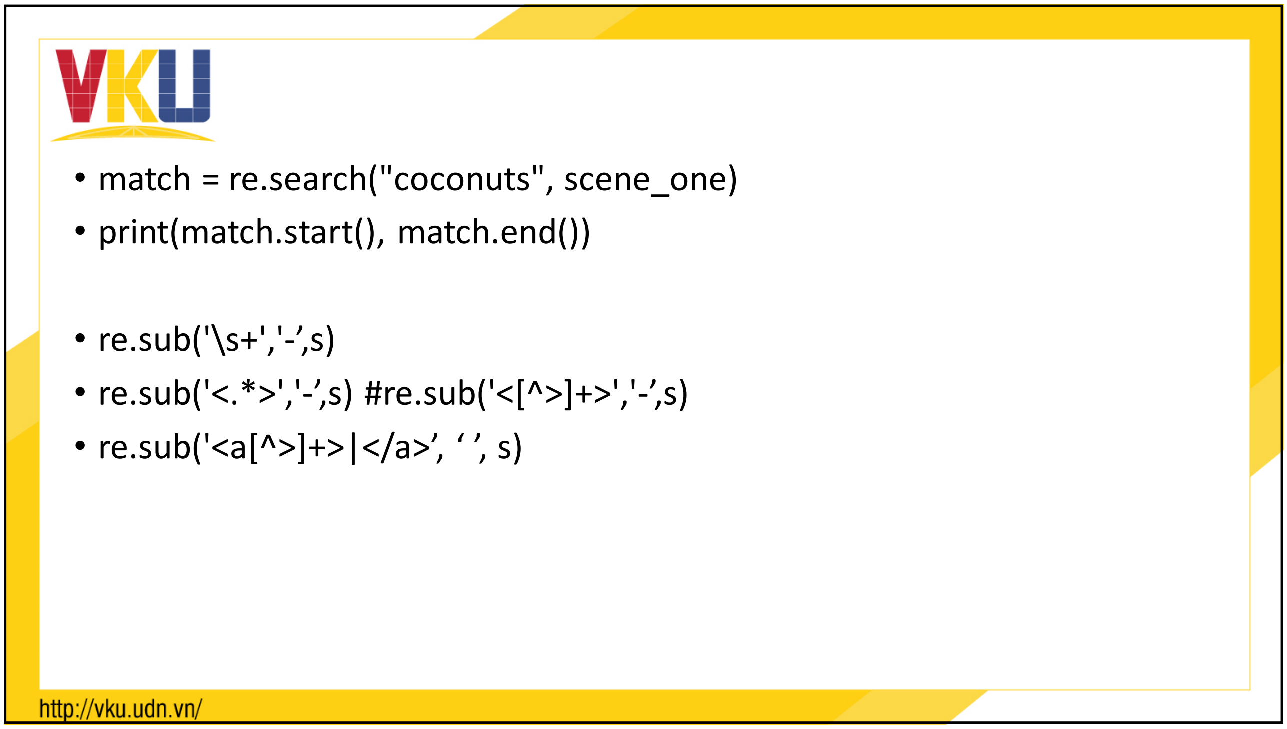
19



Examples

- import re
- s="Let's write RegEx!“
- re.findall("\w+",s)
- re.findall("\w",s)
- re.split('\s', s) # re.split(‘\s+', s)
- re.split('[!.?]', s)
- re.findall("[A-Z]\w+", s) #re.findall("[A-Z]\w", s)
- re.findall("[0-9]{2,4}", “…")

20

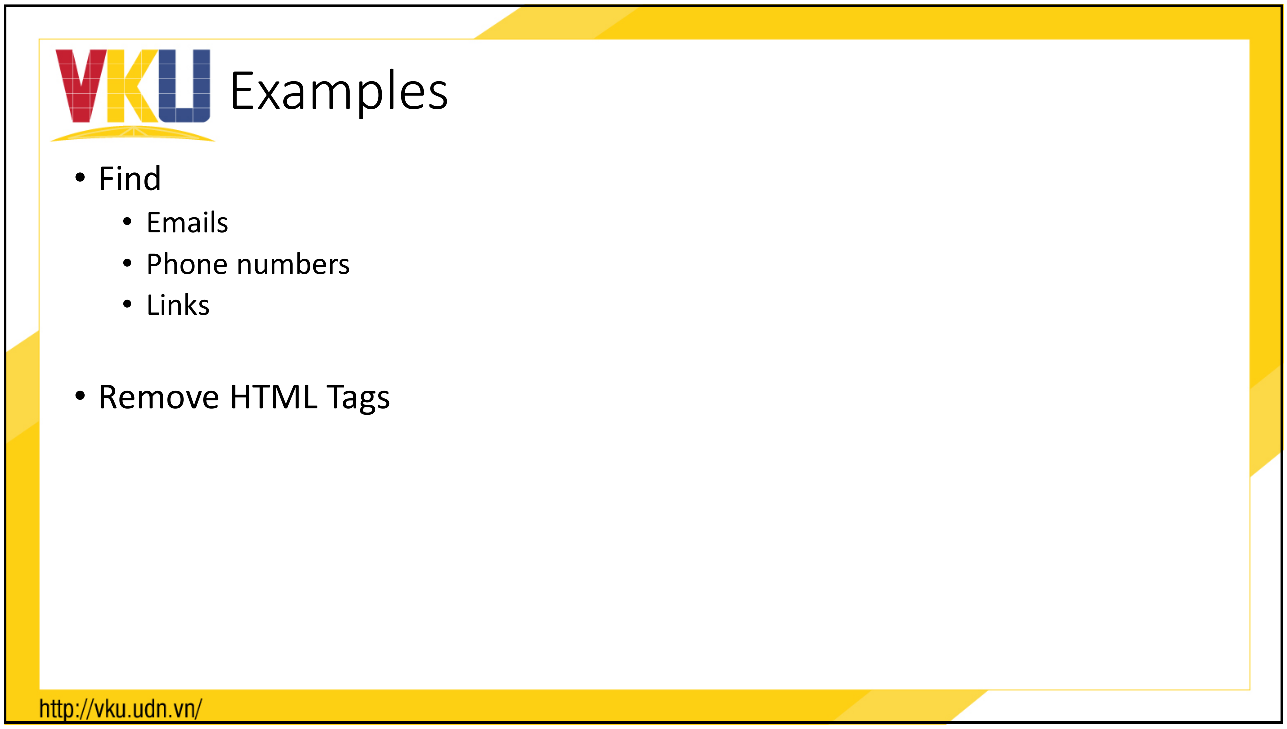


- match = re.search("coconuts", scene_one)
- print(match.start(), match.end())

- re.sub('\s+','-’,s)
- re.sub('<.*>','-’,s) #re.sub('<[^>]+>','-’,s)
- re.sub('<a[^>]+>|</a>’, ‘ ’, s)

http://vku.udn.vn/

21



Examples

- Find
  - Emails
  - Phone numbers
  - Links

- Remove HTML Tags

http://vku.udn.vn/

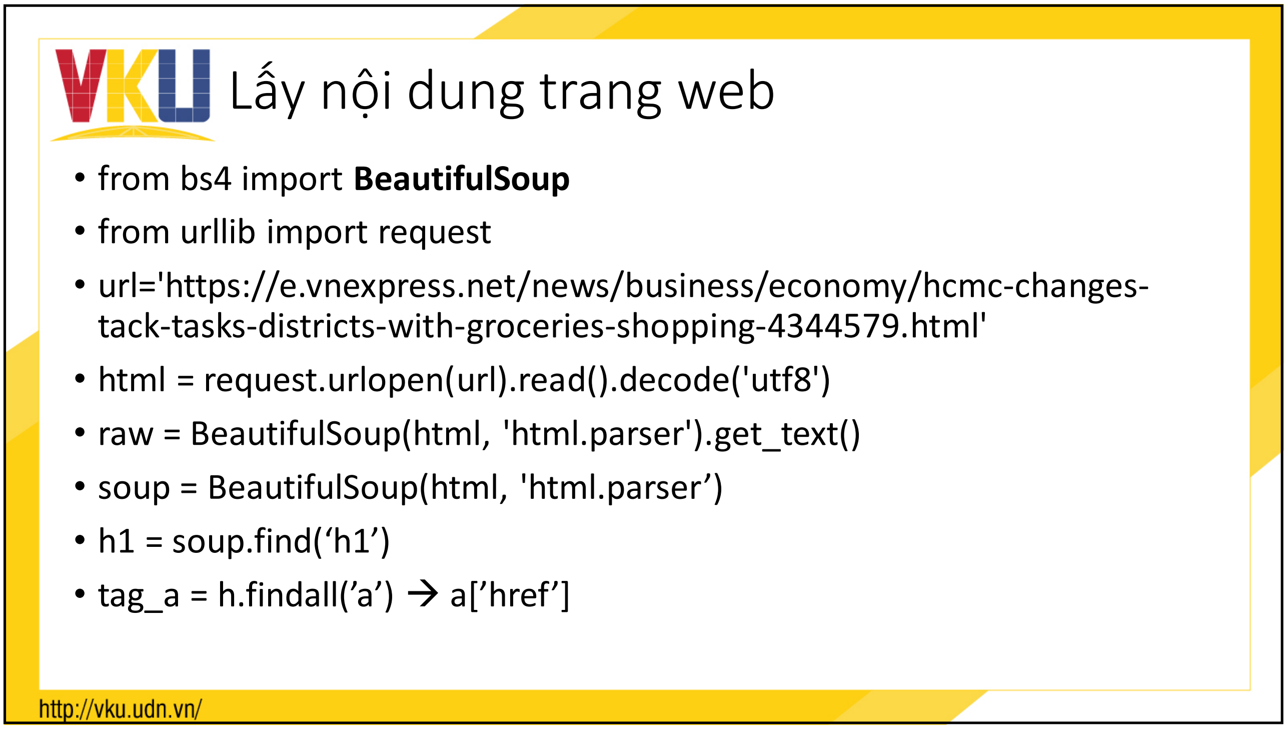
22

- (http|ftp|https)://([\w_-]+(?:(?:\.[\w_-]+)+))([\w.,@?^=%&:/~+#-]*[\w@?^=%&/~+#-])?
- re.findall(r'(https?://[^\s"]+)', s)
- re.findall('<a[^>]*href="([^"]+)"[^>]*>’,s)
- re.findall(r'[\w.+-]+@[\w-]+\.[\w.-]+', x)

- re.search('(http|ftp|https)://([\w_-]+(?:(?:\.[\w_-]+)+))([\w.,@?^=%&:/~+#-]*[\w@?^=%&/~+#-])?', s).group()

Hướng dẫn Bài tập Tuần 2

- Nội dung:
  - Thu thập dữ liệu từ trang web
  - Thực hiện các bước tiền xử lý
  - Đếm tần số xuất hiện của mỗi từ
  - Tách câu trong văn bản

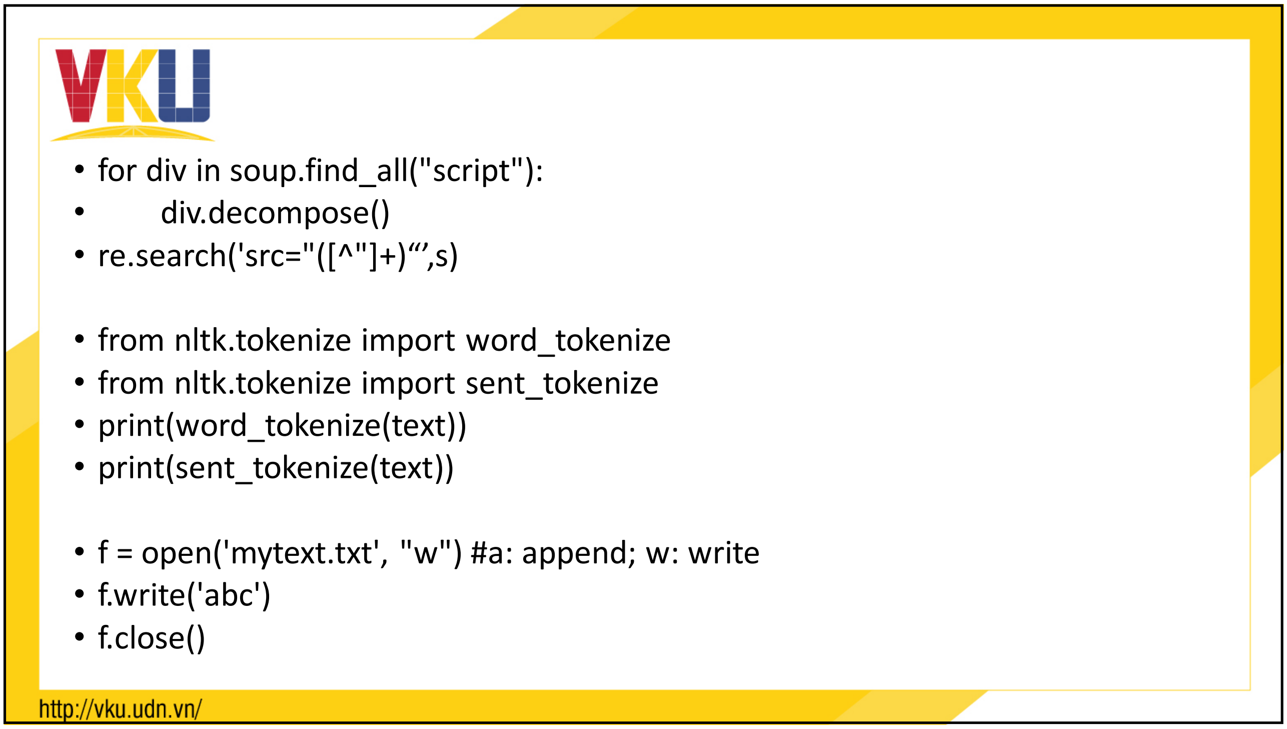


Lấy nội dung trang web

- from bs4 import **BeautifulSoup**
- from urllib import request
- url='https://e.vnexpress.net/news/business/economy/hcmc-changes-tack-tasks-districts-with-groceries-shopping-4344579.html'
- html = request.urlopen(url).read().decode('utf8')
- raw = BeautifulSoup(html, 'html.parser').get_text()
- soup = BeautifulSoup(html, 'html.parser’)
- h1 = soup.find(‘h1’)
- tag_a = h.findall(’a’) → a[’href’]

http://vku.udn.vn/

25



- for div in soup.find_all("script"):
-         div.decompose()
- re.search('src="([^"]+)“’,s)

- from nltk.tokenize import word_tokenize
- from nltk.tokenize import sent_tokenize
- print(word_tokenize(text))
- print(sent_tokenize(text))

- f = open('mytext.txt', "w") #a: append; w: write
- f.write('abc')
- f.close()

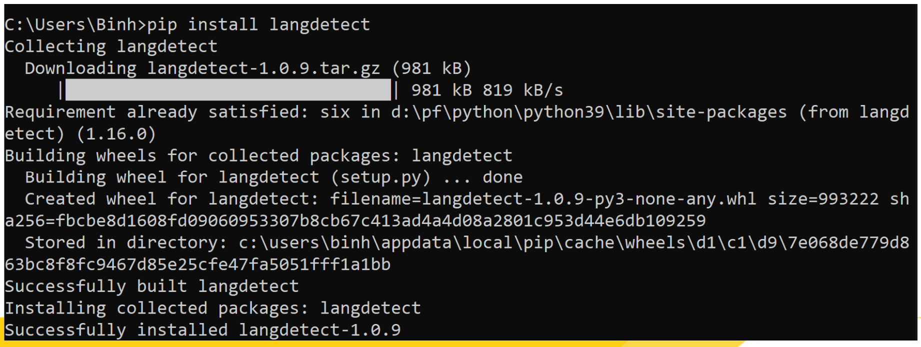
http://vku.udn.vn/

26

NLTK and language detection

- pip install langdetect

- from langdetect import detect
- detect("Chào các bạn")



```
C:\Users\Binh>pip install langdetect
Collecting langdetect
  Downloading langdetect-1.0.9.tar.gz (981 kB)
     |████████████████████████████████| 981 kB 819 kB/s
Requirement already satisfied: six in d:\pf\python\python39\lib\site-packages (from langd
etect) (1.16.0)
Building wheels for collected packages: langdetect
  Building wheel for langdetect (setup.py) ... done
  Created wheel for langdetect: filename=langdetect-1.0.9-py3-none-any.whl size=993222 sh
a256=fbcbe8d1608fd09060953307b8cb67c413ad4a4d08a2801c953d44e6db109259
  Stored in directory: c:\users\binh\appdata\local\pip\cache\wheels\d1\c1\d9\7e068de779d8
63bc8f8fc9467d85e25cfe47fa5051fff1a1bb
Successfully built langdetect
Installing collected packages: langdetect
Successfully installed langdetect-1.0.9
```

http://vku.udn.vn/

27

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN
Vietnam - Korea University of Information and Communication Technology

Vietnamese Natural Language Process Toolkit

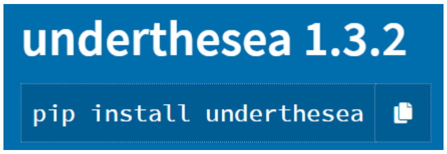
NLP

http://vku.udn.vn/

28

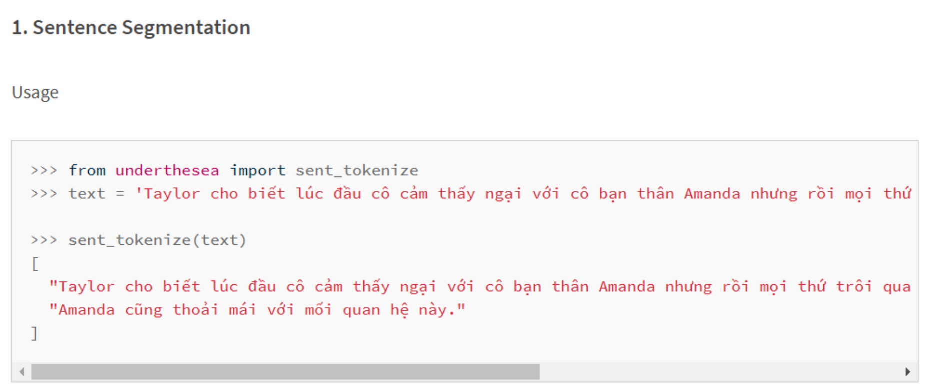
underthesea 1.3.2

- https://pypi.org/project/underthesea/



underthesea 1.3.2

```
pip install underthesea
```



1. Sentence Segmentation

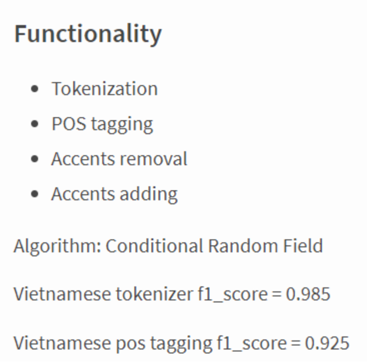
Usage

```
>>> from underthesea import sent_tokenize
>>> text = 'Taylor cho biết lúc đầu cô cảm thấy ngại với cô bạn thân Amanda nhưng rồi mọi thứ

>>> sent_tokenize(text)
[
  "Taylor cho biết lúc đầu cô cảm thấy ngại với cô bạn thân Amanda nhưng rồi mọi thứ trôi qua
  "Amanda cũng thoải mái với mối quan hệ này."
]
```

pyvi 0.1.1

- https://pypi.org/project/pyvi/



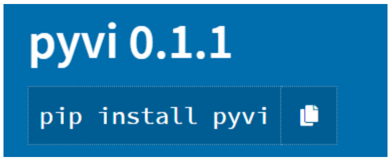
Functionality

- Tokenization
- POS tagging
- Accents removal
- Accents adding

Algorithm: Conditional Random Field

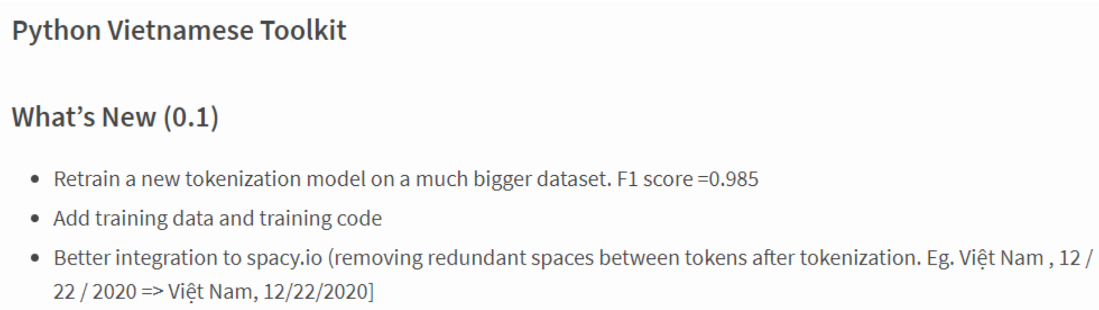
Vietnamese tokenizer f1_score = 0.985

Vietnamese pos tagging f1_score = 0.925



pyvi 0.1.1

```
pip install pyvi
```



Python Vietnamese Toolkit

What's New (0.1)

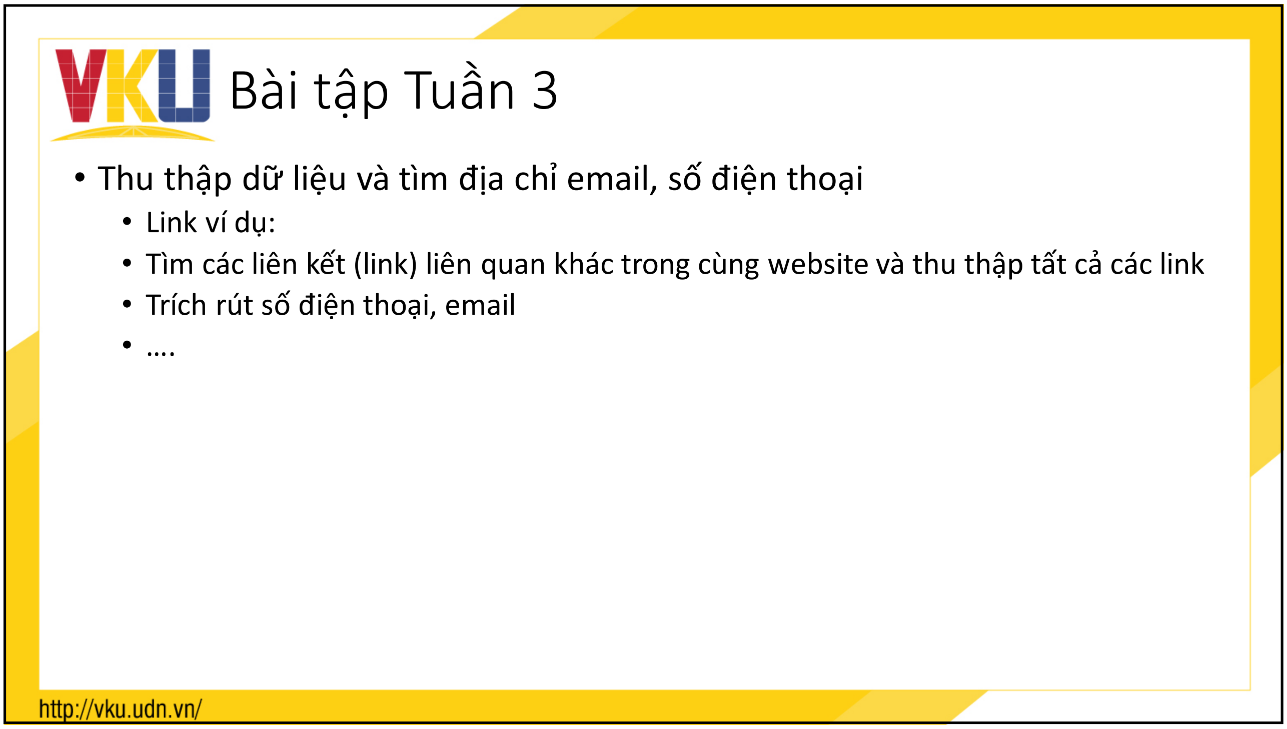
- Retrain a new tokenization model on a much bigger dataset. F1 score =0.985
- Add training data and training code
- Better integration to spacy.io (removing redundant spaces between tokens after tokenization. Eg. Việt Nam , 12 / 22 / 2020 => Việt Nam, 12/22/2020]

- https://vlsp.org.vn/wiki
- https://vlsp.org.vn/resources

- https://vlsp.hpda.vn/demo/?page=resources

http://vku.udn.vn/

31



Bài tập Tuần 3

- Thu thập dữ liệu và tìm địa chỉ email, số điện thoại
  - Link ví dụ:
  - Tìm các liên kết (link) liên quan khác trong cùng website và thu thập tất cả các link
  - Trích rút số điện thoại, email
  - ….

http://vku.udn.vn/

32

33

34