Paweekorn Soratyathorn
65070501037

1. Loading data

```
import nltk
# nltk.download('all')
# nltk.download() #for the missing library
f = open('books.txt', encoding='utf-8')
raw = f.read()
tokens = nltk.word_tokenize(raw)
```

```
print(tokens[1000:1100])
```

```
[',', 'eating', 'banquets', 'in', 'the', 'Great', 'Hall', ',', 'sleep\xading', 'in', 'his', 'four-poster', 'bed', 'in', 'the', 'tower',
```

2. Regular expression

```
tokens = [w.lower() for w in tokens]
```

```
import re

[w for w in tokens if re.search('ed$', w)]
```

```
['reserved',
 'published',
 'registered',
 'related',
 'reproduced',
 'stored',
 'transmitted',
 'opened',
 'released',
 'printed',
 'roared',
 'tried',
 'bored',
 'used',
 'snarled',
 'fried',
 'exchanged',
 'tried',
 'drowned',
 'drooped',
 'grinned',
 'turned',
 'gasped',
 'clapped',
 'jumped',
 'thundered',
 'roared',
 'warned',
 'stared',
 'purple-faced',
 'winded',
 'missed',
 'missed',
 'bed',
 'locked',
 'practiced',
 'called',
 'concerned',
 'padlocked',
 'looked',
 'horse-faced',
 'lightning-shaped',
 'survived',
 'feared',
 'died',
 'escaped',
```

```
 'destroyed',
 'failed',
 'killed',
 'treated',
 'rolled',
 'remembered',
 'happened',
 'cleared',
 'looked',
 'cried',
 'rounded',
 'sniffed'
```

What kind of input that matches, and what do not? Also, give a brief explanation.

Ans: This cell matches the words which ended with 'ed'.

```
[w for w in tokens if re.search('^..j..t..$', w)]
```

```
['dejected',
 'adjusted',
 'rejected',
 'rejected',
 'dejected',
 'adjusted',
 'adjusted',
 'adjusted',
 'adjusted']
```

What kind of input that matches, and what do not? Also, give a brief explanation.

Ans: This cell matches with words which has letter 'j' at 3rd position and 't' at 6th position.

```
[w for w in tokens if re.search('^spo.ts?$', w)]
```

```
['sport',
 'sports',
 'sport',
 'sport',
 'sports',
 'sport',
 'sports',
 'sports',
 'sports',
 'sports',
 'sports',
 'sports',
 'sports',
 'spout',
 'sport',
 'sport',
 'sport',
 'sports',
 'spout',
 'sport',
 'sports',
 'spout',
 'spout',
 'sport',
 'sports',
 'sport',
 'sport']
```

What kind of input that matches, and what do not? Also, give a brief explanation.

Ans: Words which first 3 letters are 'spo' follow by any characters and ended with 't' or 's'.

```
[w for w in tokens if re.search('^[b-f]', w)]
```

```
['chamber',
 'by',
 'by',
 'books',
 'for',
 'f.',
 'driver',
 'foul-weather',
 'friend',
 'copyright',
 'by',
```

```
 'by',
 'copyright',
 'bros.',
 'by',
 'division',
 'characters',
 'bros.',
 'be',
 'form',
 'by',
 'electronic',
 'for',
 'department',
 'broadway',
 'congress',
 'cataloging-in-publication',
 'data',
 'chamber',
 'by',
 'cm',
 'chamber',
 'finds',
 'danger',
 'from',
 'dark',
 'been',
 'fiction',
 'fiction',
 'fiction',
 'england',
 'fiction',
 'fic',
 'dc21',
 'first',
 'edition',
 'chapter',
 'birthday',
 'for',
 'first',
 'broken',
 'breakfast',
 'four',
 'drive',
 'dursley',
 'been',
 'early',
 'by',
```

What kind of input that matches, and what do not? Also, give a brief explanation.

Ans: Words started with b to f (b, c, d, e, f).

```
1 [w for w in tokens if re.search('^[bdf][aue][rts]$', w)]
```

```
['but',
 'but',
 'far',
 'but',
 'but',
 'but',
 'fat',
 'but',
 'fat',
 'but',
 'but',
 'but',
 'fat',
 'fat',
 'but',
 'but',
 'but',
 'but',
 'but',
 'but',
 'but',
 'but',
 'but',
 'but',
 'but',
 'bat',
 'but',
 'but',
 'but',
 'but',
```

```
 'but',
 'but',
 'but',
 'but',
 'but',
 'but',
 'but',
 'but',
 'bar',
 'but',
 'but',
 'but',
 'but',
 'but',
 'but',
 'but',
 'but',
 'but',
 'far',
 'but',
 'but',
 'fat',
 'but',
 'but',
 'but',
 'but',
 'but',
 'but',
 'but',
 'but',
```

What kind of input that matches, and what do not? Also, give a brief explanation.

Ans: Words started with 'b' or 'd' or 'f' followed by 'a' or 'u' or 'e' and ended with 'r' or 't' or 's'.

```
1 set([w for w in tokens if re.search('noo+$', w)])
```

```
{'nooo',
 'noooo',
 'nooooo',
 'noooooo',
 'nooooooo',
 'noooooooo',
 'nooooooooooo',
 'noooooooooooooooo'}
```

What kind of input that matches, and what do not? Also, give a brief explanation.

Ans: Words contained no which has o follwed 1 til infinity letters.

```
1 set([w for w in tokens if re.search('noo*$', w)])
```

```
{'albino',
 'dunno',
 'gemino',
 'inferno',
 'n-no',
 'no',
 'nooo',
 'noooo',
 'nooooo',
 'noooooo',
 'nooooooo',
 'noooooooo',
 'nooooooooooo',
 'noooooooooooooooo',
 'oppugno',
 'palomino',
 'patrono',
 'piano',
 'pi\xadano',
 'rhino',
 'xeno'}
```

What kind of input that matches, and what do not? Also, give a brief explanation.

Ans: Words contained 'no' which ended with 0 until infinity letter 'o'.

```
[w for w in tokens if re.search('\.+$', w)]
```

```
['j.',
 'k.',
 'a.',
 'p.',
 'f.',
 '.',
 'j.',
 'k.',
 '.',
 'bros.',
 '.',
 'inc.',
 '.',
 'inc.',
 'bros.',
 'j.',
 'k.',
 '.',
 'inc.',
 '.',
 'j.',
 'k.',
 'j.',
 'k.',
 '.',
 'p.',
 '.',
 '.',
 '.',
 '.',
 '.',
 '.',
 '.',
 '.',
 '.',
 '.',
 '.',
 '.',
 '.',
 '.',
 'u.s.a.',
 '.',
 'mr.',
 '.',
 '.',
 '.',
 '.',
 '.',
 '.',
 'out.',
 '.',
 '.',
 'bacon.',
 '.',
 '.',
 '.',
 '.',
 '.',
 'pan.',
 '.',
```

What kind of input that matches, and what do not? Also, give a brief explanation.

Ans: Words ended with 1 or more dot symbol.

```
[w for w in tokens if re.search('^[A-Z]+\.$', w)]
```

```
[]
```

What kind of input that matches, and what do not? Also, give a brief explanation.

Ans: Words contained 1 or more of the capital letter and ended with dot symbol.

```
[w for w in tokens if re.search('^[0-9]{4}$', w)]
```

```
['1999',
 '1999',
 '1920',
 '1999',
 '1999',
```

```
 '1875',
 '1492',
 '1289',
 '2007',
 '2007',
 '1966',
 '1967',
 '1975',
 '1977',
 '2007',
 '1945',
 '1689',
 '1960',
 '1960',
 '1981',
 '1981',
 '2000',
 '2000',
 '1920',
 '2000',
 '1792',
 '2005',
 '2005',
 '1920',
 '2005',
 '2003',
 '2003',
 '1920',
 '2003',
 '1875',
 '1749',
 '1999',
 '1999',
 '1920',
 '1999',
 '1999',
 '1612',
 '1722',
 '1296',
 '1997',
 '1998',
 '1920',
 '1998',
 '1998',
 '1945',
 '1473',
 '1945',
 '1709',
 '1637']
```

What kind of input that matches, and what do not? Also, give a brief explanation.

Ans: This cell matches the pattern of number with length of 4 also, started and ended with the number.

```
1 set([w for w in tokens if re.search('(ed|ing)$', w)])
```

```
{'conquering',
 'wit\xadnessed',
 'boring',
 'appealed',
 'circling',
 'connected',
 'unre\xadlated',
 'preferred',
 'waddled',
 'bandaged',
 'exhibited',
 'high-ceilinged',
 'cheating',
 'chuckled',
 'sizing',
 'uncorking',
 'hunt¬ing',
 'lion-topped',
 'leav\xading',
 'glitter\xading',
 'overflow\xading',
 'seedy-looking',
 'eve\xadning',
 'winced',
 'flame-loving',
 'belled',
```

```
 'honeyed',
 'fanning',
 'pray\xading',
 'tabooed',
 'olive-skinned',
 'swallowed',
 'distinctive-looking',
 'over\xadpowered',
 'com\xadpared',
 'blushing',
 'pummelled',
 'buoyed',
 'standing',
 'battled',
 'overstretching',
 'motioning',
 'half-concealed',
 'reckoned',
 'lung\xading',
 'crusted',
 'changing',
 'skated',
 'creat\xading',
 'tramped',
 'gabbling',
 'siphoned',
 'dumbfounded',
 'carted',
 'de\xadmanded',
 'electrified',
 'unrav\xadeled',
 'head\xadlined',
```

What kind of input that matches, and what do not? Also, give a brief explanation.

Ans: A words ended with 'ed' or 'ing'.

3. Regular expression tokenizer

3.1 Basic regex tokenizer

```
s = ("Good muffins cost $3.88\nin New York.  Please buy me\n"
      "two of them.\n\nThanks.")
s2 = ("Alas, it has not rained today. When, do you think, "
      "will it rain again?")
s3 = ("<p>Although this is <b>not</b> the case here, we must "
      "not relax our vigilance!</p>")
```

```
s2
```

```
'Alas, it has not rained today. When, do you think, will it rain again?'
```

```
nltk.regexp_tokenize(s2, r'[,\.\?!"]\s*', gaps=True)
```

```
['Alas',
 'it has not rained today',
 'When',
 'do you think',
 'will it rain again']
```

What happened after applying the above RegEx tokenizer to the input string?

Ans: It splits a sentence into a list of words by the white space and some special character (, . ? !).

```
s3
```

```
'<p>Although this is <b>not</b> the case here, we must not relax our vigilance!</p>'
```

```
nltk.regexp_tokenize(s3, r'</?.>', gaps=False)
```

```
['<p>', '<b>', '</b>', '</p>']
```

What happened after applying the above RegEx tokenizer to the input string?

Ans: This RegEx extract the tags of each sentences.

```
nltk.regexp_tokenize(s3, r'</?.>', gaps=True)
```

```
['Although this is ',
 'not',
 ' the case here, we must not relax our vigilance!']
```

What happened after applying the above RegEx tokenizer to the input string?

Ans: This RegEX extracts the content of each sentences with difference from previous cell by the hyperparameter *gaps*.

Lab part

```
def pattern_matching(regex='.', string=''):
    pattern = re.compile(regex)
    matches = pattern.finditer(string)

    for match in matches:
        print(match)
```

1. Stock Indices

ref: https://en.wikipedia.org/wiki/List_of_stock_market_indices

```
stock_index = '''
S&P Asia 50
S&P/ASX 20
S&P Europe 350
STOXX Europe 600
FTSE Developed Europe Index.
BSE DCI
EGX 70 Index
SSE Composite Index
CSC X
CASPI
KSE-30 Index
MVIS® Australia Small-Cap Dividend Payers Index
'''
```

a. Why you want to discover those patterns?
Ans. I am interested in stock price right now because I want to try stock trade later as soon as I have enough money and knowledge for it.

b. Why are they interesting to you?
Ans. The stock price of each one will increase or decrease relates with the market at that moment. I want to know how the market works or how people know when to trade or sell their stock to make profit from it so, the true reason why I'm interested in this topics is learning how market works through stock price.

c. Does this RegEx works as you expected?
Ans. Yes, but I don't think the pattern would be this plain. T_T

2. Product's Model

- In this topic, I'll use *ExxonMobil* lubricant product's name as an example.

ref: https://www.mobil.co.th/th-th/our-products#t=https%3A%2F%2Fwww.mobil.co.th%2Fth-th%2Four-products&sort=relevancy

```
brand = '''
Mobil Delvac Modern™ 15W-40 Full Protection
Mobil Delvac Modern™ 15W-40 Super Defense
Mobil Delvac Legend™ CH-4 15W-40 Heavy Duty
```

```
Mobil Delvac Legend™ 1330

Mobil 1™ Triple Action Power+
Mobil 1™ Turbo Diesel 5W-40
Mobil 1™ ESP 0W-30
Mobil 1™ Racing 4T 10W-40

Mobil Super™ All-In-One Protection
Mobil Super™ All-In-One Protection Turbospeed
Mobil Super™ Friction Fighter
Mobil Super™ Friction Fighter Turbospeed
Mobil Super™ Everyday Protection Turbospeed

Mobil Super Moto™ 20W-50
Mobil Super Moto™ Scooter 10W-40
Mobil Super Moto™ Scooter Gear Oil
'''
```

```
regex = r'Mobil.((1|Super|Delvac).?).([a-zA-Z]+.?).((\w+)-?(\w+)).((\w+.)?\w+)?'
pattern_matching(regex, brand)
```

```
<re.Match object; span=(1, 44), match='Mobil Delvac Modern™ 15W-40 Full Protection'>
<re.Match object; span=(45, 86), match='Mobil Delvac Modern™ 15W-40 Super Defense'>
<re.Match object; span=(87, 119), match='Mobil Delvac Legend™ CH-4 15W-40'>
<re.Match object; span=(131, 156), match='Mobil Delvac Legend™ 1330'>
<re.Match object; span=(158, 186), match='Mobil 1™ Triple Action Power'>
<re.Match object; span=(188, 215), match='Mobil 1™ Turbo Diesel 5W-40'>
<re.Match object; span=(216, 234), match='Mobil 1™ ESP 0W-30'>
<re.Match object; span=(235, 260), match='Mobil 1™ Racing 4T 10W-40'>
<re.Match object; span=(262, 296), match='Mobil Super™ All-In-One Protection'>
<re.Match object; span=(297, 342), match='Mobil Super™ All-In-One Protection Turbospeed'>
<re.Match object; span=(343, 372), match='Mobil Super™ Friction Fighter'>
<re.Match object; span=(373, 413), match='Mobil Super™ Friction Fighter Turbospeed'>
<re.Match object; span=(414, 457), match='Mobil Super™ Everyday Protection Turbospeed'>
<re.Match object; span=(459, 483), match='Mobil Super Moto™ 20W-50'>
<re.Match object; span=(484, 516), match='Mobil Super Moto™ Scooter 10W-40'>
<re.Match object; span=(517, 551), match='Mobil Super Moto™ Scooter Gear Oil'>
```

a. Why you want to discover those patterns?
Ans. I think this findings will be helpful when doing an sentiment analysis in term of collecting the users' opinion from websites such as pantip, facebook, instagram, e.g.

b. Why are they interesting to you?
Ans. For the regex part, it is quite hard for me when I have to due with the superscript so, I think it is challenging to do this one. In context part, I think it will be interesting if we can find the product which is limited or less manufacture to people and go deeper like why it is limited.

c. Does this RegEx works as you expected?
Ans. Yes, sure!

Note: This example included only passenger vechicle. If you want to detect all product model the second group of regex should be \w+ for flexible with all product model name.

3. Transaction ID

- I'll use my 7-11 online transaction to be example for this topics

```
transID = '''
TID#20240820000000002790365064
TID#20240819000000002787577518
TID#20240815000000002776969741
TID#20240805000000002749615183
TID#20240803000000002744049940
TID#20240724000000002718083563

TID#08031240728857327204727
TID#08031240815421010192537
TID#08031240807568636204714
'''
```

```
regex = r'TID#((2024\d{4})(0{7})27\d{8}|(08031240)\d{15})'
pattern_matching(regex, transID)
```

```
<re.Match object; span=(1, 30), match='TID#20240820000000002790365064'>
<re.Match object; span=(31, 60), match='TID#20240819000000002787577518'>
<re.Match object; span=(61, 90), match='TID#20240815000000002776969741'>
<re.Match object; span=(91, 120), match='TID#20240805000000002749615183'>
<re.Match object; span=(121, 150), match='TID#20240803000000002744049940'>
<re.Match object; span=(151, 180), match='TID#20240724000000002718083563'>
<re.Match object; span=(182, 209), match='TID#08031240728857327204727'>
<re.Match object; span=(210, 237), match='TID#08031240815421010192537'>
<re.Match object; span=(238, 265), match='TID#08031240807568636204714'>
```

a. Why you want to discover those patterns?

Ans. I found that this transaction numbers help in Fraud detection by linking the suspicious transaction together so, I have a thought that finding a pattern from these unique numbers can be useful.

b. Why are they interesting to you?

Ans. I am curious what's the benefit of serial number so, I search for it on google and I found that it is used for tracking and identifying the parcel. I didn't found the complex pattern from the serial number so, I continued on searching for ID which can used for tracking as well and I found this topic!

c. Does this RegEx works as you expected?

Ans. Exactly

Note: I found some insight from my examples. Transaction that amount is more than 100 baht will start with 0803124. Other than this case, the set of number will start with present data in format YYYYMMDD.

```
Start coding or generate with AI.
```