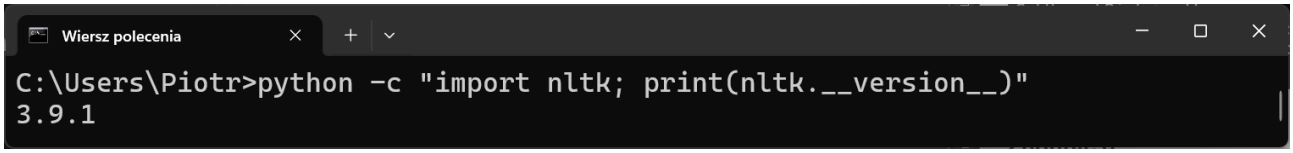
Eksploracja danych internetowych

Laboratorium 2

Prowadzący: pracownik UR
Wykonał: Piotr Rojek, pr125159

Zadanie 1
Zainstaluj pakiet NLTK.



Zadanie 2
Napraw błąd związany z innymi rozmiarami list labelList i textList w jednym z pierwszych przykładów z Wykładu 2.

```
rawData = open("SMSSpamCollection.tsv", encoding="utf-8").read()

parsedData = rawData.replace( _old: '\t',  _new: '\n').split('\n')
labelList = parsedData[0::2]
textList = parsedData[1::2]

print("Długość labelList: ", len(labelList))
print("Długość textList: ", len(textList))

if len(labelList) > len(textList):
    labelList = labelList[:-(len(labelList) - len(textList))]
elif len(textList) > len(labelList):
    textList = textList[:-(len(textList) - len(labelList))]

print("\nDługość labelList: ", len(labelList))
print("Długość textList: ", len(textList))
```

Testy:

```
Długość labelList:  5575
Długość textList:  5574

Długość labelList:  5574
Długość textList:  5574
```

Zadanie 3

Zaprezentuj możliwości wyrażeń regularnych na przykładzie funkcji findall, split, sub, search, match, fullmatch, finditer, escape.

```python
import re

text = """Ala ma kota, a kot ma Ale.
Kot pije mleko.
Kot wszedł na płotek.
Kot spadł z płotka.
Płotek się załamał."""

# Funkcja 'findall' - Poszukuje fragmenty stringa dla danego wzorca.
words_findall = re.findall( pattern: r'\w+', text)
print(words_findall, "\n")

# Funkcja 'split' - Dzieli stringa przy każdym wystąpieniu wzorca.
words_split = re.split( pattern: r'\.\n', text)
print(words_split, "\n")

# Funkcja 'sub' - Zastępuje w stringu wszystkie wystąpienia wzorca.
words_sub = re.sub( pattern: r'Kot',  repl: 'Kotek', text)
print(words_sub, "\n")

# Funkcja 'search' - Skanuje stringa w poszukiwaniu pierwszego miejsca, gdzie występuje wzorzec.
words_search_1 = re.search( pattern: "mleko", text)
words_search_2 = re.search( pattern: "woda", text)
print(words_search_1)
print(words_search_2, "\n")
```

```python
# Funkcja 'match' - Sprawdza, czy początek stringa pasuje do wzorca.
words_match_1 = re.match( pattern: "Ala", text)
words_match_2 = re.match( pattern: "Ania", text)
print(words_match_1)
print(words_match_2, "\n")

# Funkcja 'fullmatch' - Sprawdza, czy cały string pasuje do wzorca.
words_fullmatch = re.fullmatch( pattern: r'[a-zA-Z0-9-\s.]+', text)
print(words_fullmatch, "\n")

# Funkcja 'finditer' - Znajduje dopasowania wzorca w stringu jako iterator
words_finditer = re.finditer( pattern: r'[A-Z]', text)
for iterator in words_finditer:
    print(iterator)

# Funkcja 'escape' - Dodaje ukośniki przed wszystkimi znakami specjalnymi w stringu.
words_escape = re.escape(text)
print("\n", words_escape)
```

Testy:

```
Funkcja 'findall':
['Ala', 'ma', 'kota', 'a', 'kot', 'ma', 'Ale', 'Kot', 'pije', 'mleko', 'Kot', 'wszedł', 'na',
 'płotek', 'Kot', 'spadł', 'z', 'płotka', 'Płotek', 'się', 'załamał']

Funkcja 'split':
['Ala ma kota, a kot ma Ale', 'Kot pije mleko', 'Kot wszedł na płotek', 'Kot spadł z płotka',
 'Płotek się załamał.']

Funkcja 'sub':
Ala ma kota, a kot ma Ale.
Kotek pije mleko.
Kotek wszedł na płotek.
Kotek spadł z płotka.
Płotek się załamał.

Funkcja 'search':
<re.Match object; span=(36, 41), match='mleko'>
None
```

```
Funkcja 'match':
<re.Match object; span=(0, 3), match='Ala'>
None

Funkcja 'fullmatch':
None

Funkcja 'finditer':
<re.Match object; span=(0, 1), match='A'>
<re.Match object; span=(22, 23), match='A'>
<re.Match object; span=(27, 28), match='K'>
<re.Match object; span=(43, 44), match='K'>
<re.Match object; span=(65, 66), match='K'>
<re.Match object; span=(85, 86), match='P'>

Funkcja 'escape':
Ala\ ma\ kota,\ a\ kot\ ma\ Ale\.\
Kot\ pije\ mleko\.\
Kot\ wszedł\ na\ płotek\.\
Kot\ spadł\ z\ płotka\.\
Płotek\ się\ załamał\.
```

Zadanie 4

Na wybranym przez siebie zbiorze tekstowym np.: pobranym z Internetu, ale innym od tego pokazanego na wykładzie, wykonaj:

a) **Usuwanie znaków interpunkcyjnych.**
b) **Tokenizację.**
c) **Usuwanie stopwords.**

```python
import pandas as pd
import string
import re
import nltk.corpus

pd.set_option('display.max_colwidth', 100)
data = pd.read_csv( filepath_or_buffer: 'Organizations.csv', sep=',')
data.drop(data.columns[[0, 1, 8]], axis=1, inplace=True)
print("Początkowe dane:")
print(data.head(), "\n")

# Usuwanie znaków interpunkcyjnych

def remove_punctuation(text):  1 usage
    no_punctuation = text
    if not isinstance(text, int) and not isinstance(text, float):
        no_punctuation = "".join([char for char in text if char not in string.punctuation])
    return no_punctuation

data = data.apply(lambda text: text.apply(remove_punctuation))
print("Usuwanie znaków interpunkcyjnych:")
print(data.head(), "\n")
```

```python
# Tokenizacja

def to_lower(text):  1 usage
    lower = text
    if not isinstance(text, int) and not isinstance(text, float):
        lower = text.lower()
    return lower

def tokenize(text):  1 usage
    tokens = text
    if not isinstance(text, int) and not isinstance(text, float):
        tokens = re.split( pattern: r'\W+', text)
    return tokens

data = data.apply(lambda text: text.apply(to_lower).apply(tokenize))
print("Tokenizacja:")
print(data.head(), "\n")
```

```python
# Usuwanie stopwords

# nltk.download('stopwords')
stopwords = nltk.corpus.stopwords.words("english")

def remove_stopwords(text):  1 usage
    no_stopwords = text
    if not isinstance(text, int) and not isinstance(text, float):
        no_stopwords = [word for word in text if word not in stopwords]
    return no_stopwords

data = data.apply(lambda text : text.apply(remove_stopwords))
print("Usuwanie stopwords:")
print(data.head())
```

Testy:

```
Początkowe dane:
                         Name  ...                Industry
0                Mckinney PLC  ...                   Dairy
1              Cunningham LLC  ...                 Library
2                  Ruiz-Walls  ...  Hospital / Health Care
3  Parrish, Osborne and Clarke  ...             Supermarkets
4       Diaz, Robles and Haley  ...          Nanotechnology

[5 rows x 6 columns]

Usuwanie znaków interpunkcyjnych:
                        Name  ...               Industry
0               Mckinney PLC  ...                  Dairy
1             Cunningham LLC  ...                Library
2                 RuizWalls  ...  Hospital  Health Care
3  Parrish Osborne and Clarke  ...           Supermarkets
4       Diaz Robles and Haley  ...         Nanotechnology

[5 rows x 6 columns]
```

```
Tokenizacja:
                           Name  ...                  Industry
0                 [mckinney, plc]  ...                  [dairy]
1               [cunningham, llc]  ...                [library]
2                     [ruizwalls]  ...  [hospital, health, care]
3  [parrish, osborne, and, clarke]  ...            [supermarkets]
4       [diaz, robles, and, haley]  ...          [nanotechnology]

[5 rows x 6 columns]

Usuwanie stopwords:
                         Name  ...                  Industry
0             [mckinney, plc]  ...                  [dairy]
1           [cunningham, llc]  ...                [library]
2                 [ruizwalls]  ...  [hospital, health, care]
3  [parrish, osborne, clarke]  ...            [supermarkets]
4       [diaz, robles, haley]  ...          [nanotechnology]

[5 rows x 6 columns]
```