Trabalhando com texto em Python

1 - Comprimento de uma string

In [1]:
```
text1 = "Ethics are built right into the ideals and objectives of the United Nations"

len(text1)
```

Out[1]:
```
75
```

2 - Comprimento de uma lista (tokens)

In [2]:
```
text2 = text1.split(' ')
text2
```

Out[2]:
```
['Ethics',
 'are',
 'built',
 'right',
 'into',
 'the',
 'ideals',
 'and',
 'objectives',
 'of',
 'the',
 'United',
 'Nations']
```

In [3]:
```
len(text2)
```

Out[3]:
```
13
```

3 - Encontrado palavras com Lista de Compreensão

In [4]:
```
[w for w in text2 if len(w) > 3]
```

Out[4]:
```
['Ethics',
 'built',
 'right',
 'into',
 'ideals',
 'objectives',
 'United',
 'Nations']
```

4 - Encontrando palavras que iniciam com letra maiúscula

In [5]:
```
[w for w in text2 if w.istitle()]
```

Out[5]:
```
['Ethics', 'United', 'Nations']
```

5 - Encontrando palavras no plural

In [6]:
```
[w for w in text2 if w.endswith('s')]
```

Out[6]:
```
['Ethics', 'ideals', 'objectives', 'Nations']
```

6 - Palavras em conjuntos

In [7]:
```
text3 = 'To be or not to be'

text4 = text3.split(' ')

len(text4)
```

Out[7]:
```
6
```

In [8]:
```
len(set(text4))
```

Out[8]:
```
5
```

In [9]:
```
set(text4)
```

Out[9]:
```
{'To', 'be', 'not', 'or', 'to'}
```

7 - Convertendo para caixa baixa

In [10]:
```
[w.lower() for w in text4]
```

Out[10]:
```
['to', 'be', 'or', 'not', 'to', 'be']
```

In [11]:
```
set([w.lower() for w in text4])
```

Out[11]:
```
{'be', 'not', 'or', 'to'}
```

8 - Encontrando substrings

In [12]:
```
text3.find('not')
```

Out[12]:
```
9
```

In [13]:
```
text3.rfind('be')
```

Out[13]:
```
16
```

9 - Troca de substrings

In [14]:
```
text3.replace('not', 'NOT')
```

Out[14]:
```
'To be or NOT to be'
```

10 - Unir (join) strings

In [15]:
```
text5 = 'ouagadougou'

text6 = text5.split('ou')
```

In [ ]:
```
text6
```

Out[15]:
```
['', 'agad', 'g', '']
```

In [16]:
```
'ou'.join(text6)
```

Out[16]:
```
'ouagadougou'
```

In [17]:
```
text5.split(' ')
```

Out[17]:
```
['ouagadougou']
```

11 - Lista de todos os caracteres

In [18]:
```
[c for c in text5] #list(text5)
```

Out[18]:
```
['o', 'u', 'a', 'g', 'a', 'd', 'o', 'u', 'g', 'o', 'u']
```

12 - Lendo arquivos e linhas

In [33]:
```
f = open('./Data/UNDHR.txt', 'r')
f
```

Out[33]:
```
<_io.TextIOWrapper name='./Data/UNDHR.txt' mode='r' encoding='UTF-8'>
```

In [34]:
```
f.readline()
```

Out[34]:
```
'Universal Declaration of Human Rights - English\n'
```

In [35]:
```
f.readline().rstrip() # funciona para \n, \r\n ou \r
```

Out[35]:
```
'© 1996 - 2009 The Office of the High Commissioner for Human Rights'
```

In [36]:
```
f.seek(0)
f.readline()
```

Out[36]:
```
'Universal Declaration of Human Rights - English\n'
```

13 - Lendo o arquivo inteiro

In [37]:
```
f.seek(0)
text7 = f.read()
```

In [38]:
```
len(text7)
```

Out[38]:
```
12534
```

In [40]:
```
# text7
```

14 - Quebrando o arquivo em linhas

In [41]:
```
text8 = text7.splitlines() # limpa para \n, \r\n ou \r
len(text8)
```

Out[41]:
```
218
```

In [42]:
```
text8[0]
```

Out[42]:
```
'Universal Declaration of Human Rights - English'
```

15 - Hashtags e chamadas (callouts)

In [43]:
```
text9 = '@UN @UN_Women "Ethics are built right into the ideals and objectives of the Unite
#UNSG @ NY Society for Ethical Culture bit.ly/2guVelr'

text10 = text9.split(' ')

text10
```

Out[43]:
```
['@UN',
 '@UN_Women',
 '"Ethics',
 'are',
 'built',
 'right',
 'into',
 'the',
 'ideals',
 'and',
 'objectives',
 'of',
 'the',
 'United',
 'Nations"',
 '#UNSG',
 '@',
 'NY',
 'Society',
 'for',
 'Ethical',
 'Culture',
 'bit.ly/2guVelr']
```

In [44]:
```
[w for w in text10 if w.startswith('#')]
```

Out[44]:
```
['#UNSG']
```

In [45]:
```
[w for w in text10 if w.startswith('@')]
```

Out[45]:
```
['@UN', '@UN_Women', '@']
```

16 - Biblioteca re (regular expressions)

In [46]:
```
import re

[w for w in text10 if re.search('@[A-Za-z0-9_]+', w)]
```

Out[46]: ['@UN', '@UN_Women']

In [47]:
```
[w for w in text10 if re.search('@\w+', w)]
```

Out[47]: ['@UN', '@UN_Women']

17 - Corresponde a qualquer caractere único, exceto o caractere de nova linha

A função group() retorna a string encontrada pelo biblioteca re.

In [48]:
```
re.search(r'Co.k.e', 'Cookie').group()
```

Out[48]: 'Cookie'

In [49]:
```
print('\w')
```

\w

18 - Verifica um ou mais caracteres à esquerda (repetição)

In [50]:
```
re.search(r'Co+kie', 'Cooookie').group()
```

Out[50]: 'Cooookie'

19 - Verifica qualquer ocorrência de a ou o ou ambos na sequência especificada (repetição)

In [51]:
```
re.search(r'Ca*o*kie', 'Ckie').group()
```

Out[51]: 'Ckie'

20 - Encontrar todas as vogais

In [52]:
```
text5
re.findall(r'[aeiou]', text5)
```

Out[52]: ['o', 'u', 'a', 'a', 'o', 'u', 'o', 'u']

21 - Encontrar a negação das vogais

In [53]:
```
re.findall(r'[^aeiou]', text5)
```

Out[53]: ['g', 'd', 'g']

Texto no Pandas

22 - Importando texto

In [54]:
```
import pandas as pd

time_sentences = ["Monday: The doctor's appointment is at 2:45pm.",
                  "Tuesday: The dentist's appointment is at 11:30 am.",
                  "Wednesday: At 7:00pm, there is a basketball game!",
```

```
                  "Thursday: Be back home by 11:15 pm at the latest.",
                  "Friday: Take the train at 08:10 am, arrive at 09:00am."]

df = pd.DataFrame(time_sentences, columns=['text'])
df
```

Out[54]:

	text
0	Monday: The doctor's appointment is at 2:45pm.
1	Tuesday: The dentist's appointment is at 11:30...
2	Wednesday: At 7:00pm, there is a basketball game!
3	Thursday: Be back home by 11:15 pm at the latest.
4	Friday: Take the train at 08:10 am, arrive at ...

23 - Número de caracteres para cada string em df ['text']

In [55]:
```
df['text'].str.len()
```

Out[55]:
```
0    46
1    50
2    49
3    49
4    54
Name: text, dtype: int64
```

24 - Número de tokens para cada string em df ['text']

In [56]:
```
df['text'].str.split().str.len()
```

Out[56]:
```
0     7
1     8
2     8
3    10
4    10
Name: text, dtype: int64
```

25 - Encontre quais entradas contêm a palavra 'appointment'

In [57]:
```
df['text'].str.contains('appointment')
```

Out[57]:
```
0     True
1     True
2    False
3    False
4    False
Name: text, dtype: bool
```

26 - Encontre quantas vezes um numero ocorre

In [58]:
```
df['text'].str.count(r'\d')
```

Out[58]:
```
0    3
1    4
2    3
3    4
4    8
Name: text, dtype: int64
```

27 - Encontre todas as ocorrências dos dígitos

In [59]:
```
df['text'].str.findall(r'\d')
```

Out[59]:
```
0                    [2, 4, 5]
1                 [1, 1, 3, 0]
2                    [7, 0, 0]
3                 [1, 1, 1, 5]
4    [0, 8, 1, 0, 0, 9, 0, 0]
Name: text, dtype: object
```

28 - Agrupe e encontre as horas e os minutos

In [60]:
```
df['text'].str.findall(r'(\d?\d):(\d\d)')
```

Out[60]:
```
0               [(2, 45)]
1              [(11, 30)]
2               [(7, 00)]
3              [(11, 15)]
4    [(08, 10), (09, 00)]
Name: text, dtype: object
```

29 - Substitua os dias da semana por 'DIA'

In [61]:
```
df['text'].str.replace(r'\w+day\b', 'DIA')
```

Out[61]:
```
/var/folders/01/_r7b02r11p15j0s54gb9x0040000gn/T/ipykernel_19811/2228108679.py:1: FutureWa
rning: The default value of regex will change from True to False in a future version.
  df['text'].str.replace(r'\w+day\b', 'DIA')
0          DIA: The doctor's appointment is at 2:45pm.
1       DIA: The dentist's appointment is at 11:30 am.
2          DIA: At 7:00pm, there is a basketball game!
3         DIA: Be back home by 11:15 pm at the latest.
4    DIA: Take the train at 08:10 am, arrive at 09:...
Name: text, dtype: object
```

30 - Substitua os dias da semana por três abreviações de letras

In [62]:
```
df['text'].str.replace(r'(\w+day\b)', lambda x: x.groups()[0][:3])
```

Out[62]:
```
/var/folders/01/_r7b02r11p15j0s54gb9x0040000gn/T/ipykernel_19811/1376844824.py:1: FutureWa
rning: The default value of regex will change from True to False in a future version.
  df['text'].str.replace(r'(\w+day\b)', lambda x: x.groups()[0][:3])
0          Mon: The doctor's appointment is at 2:45pm.
1       Tue: The dentist's appointment is at 11:30 am.
2          Wed: At 7:00pm, there is a basketball game!
3         Thu: Be back home by 11:15 pm at the latest.
4    Fri: Take the train at 08:10 am, arrive at 09:...
Name: text, dtype: object
```

31 - Criar novas colunas a partir da primeira correspondência dos grupos extraídos

In [63]:
```
df['text'].str.extract(r'(\d?\d):(\d\d)')
```

Out[63]:

	0	1
0	2	45
1	11	30
2	7	00

	0	1
3	11	15
4	08	10

32 - Extrair informação de tempo completa

In [64]:
```
df['text'].str.extractall(r'((\d?\d):(\d\d) ?([ap]m))')
```

Out[64]:

		0	1	2	3
	match				
0	**0**	2:45pm	2	45	pm
1	**0**	11:30 am	11	30	am
2	**0**	7:00pm	7	00	pm
3	**0**	11:15 pm	11	15	pm
4	**0**	08:10 am	08	10	am
	1	09:00am	09	00	am

33 - Extrair informação de tempo completa e configurar o nome da coluna do dataframe

In [65]:
```
df['text'].str.extractall(r'(?P<time>(?P<hour>\d?\d):(?P<minute>\d\d) ?(?P<period>[ap]m))'
```

Out[65]:

		time	hour	minute	period
	match				
0	**0**	2:45pm	2	45	pm
1	**0**	11:30 am	11	30	am
2	**0**	7:00pm	7	00	pm
3	**0**	11:15 pm	11	15	pm
4	**0**	08:10 am	08	10	am
	1	09:00am	09	00	am

In [ ]: