regex

January 23, 2022

[1]:
```
#primeiro importar o módulo RE
import re
```

[5]:
```
#tem várias funções de processamento que pode usar, uma delas é a match() que␣
 ↪busca uma combinação no começo da string e retorna
#um booleano. o search() busca pela combinação em qualquer lugar da string

text = "Este sim é um texto de exemplo"
#agora vamos ver se é um exemplo de texto ou não

if re.search('sim', text): #o primeiro parâmetro é o padrão "sim"
    print('Obrigado ;)')
else:
    print('Que pena :/')
```

```
Obrigado ;)
```

[8]:
```
#checando por condicionais, podemos segmentar uma string. é a tokenização, onde␣
 ↪uma string é separada em
#substrings baseado em padrões. tokenização é uma atividade vital em linguagem␣
 ↪natural
#as funções findall() e split() vão analisar a strign e retornar os pedaços

txt = 'Amy é boa. Amy trabalha bem. Amy é gostosona!!'

#vamos partir a string em cada vez que achar "Amy"
re.split('Amy', txt)

re.findall('Amy', txt)
```

[8]:
```
['Amy', 'Amy', 'Amy']
```

[ ]:
```
#o search() procura por algum padrão na string, o split() vai usar um padrão␣
 ↪para criar listas de substrings
#e o findall() vai encontrar todas as ocorrências desse padrão na string
```

[14]:
```
#as âncoras especificam o início e/ou fim da string que você quer analisar. o ^␣
 ↪indica o início da anlálise
```

```
#e o $ indica o fim. se você coloca o ^ antes de uma string, quer dizer que a␣
 ↪análise começa ali
#e termina quando encontrar o $

#exemplo:
txt = 'Amy é boa. Amy trabalha bem. Amy é gostosona!!'

#vamos ver se começa com "Amy"
if re.search("^Amy", txt):
    print("Sim, começa com 'Amy'")
else:
    print("Não começa com 'Amy'")


re.search("^Amy", txt)
#o re.search() retorna o span --> a localização onde foi encontrado o que␣
 ↪procuramos e o Match = o padrão que procuramos
```

```
Sim, começa com 'Amy'
```

[14]:
```
<re.Match object; span=(0, 3), match='Amy'>
```

1 Padrões e classes de caracteres

[44]:
```
#lista com notas de um aluno durante um semestre
notas = 'ABCAABAACBCBACBCAAAA'

#para descobrir quantos B esse aluno teve, podemos usar o B como parâmetro
re.findall('B', notas)
```

[44]:
```
['B', 'B', 'B', 'B', 'B']
```

[24]:
```
#agora apra saber a quantidade de A ou B não podemos colocar AB, pq daí ia␣
 ↪buscar um A seguido de um B
#coloca então o AB dentro de colchetes, daí ele busca individualmente quantas␣
 ↪vezes aparece

re.findall('[AB]', notas)
```

[24]:
```
['A', 'B', 'A', 'B', 'B', 'A', 'B', 'A']
```

[30]:
```
#ainda da pra buscar por letras minusculas colcoando [a-z]
#construiindo um regex para analisar todas as vezes que o aluno ganhou um A␣
 ↪seguido por B ou C
re.findall("[A][B-C]", notas) #busca ai todos os A que são seguidos por um B ou␣
 ↪por um C

#também pode usar o | para representar o OU
```

```
re.findall('AB|AC', notas)
```

[30]: ['AB', 'AB', 'AC']

[31]:
```
#podemos usar o ^ com o operador para negar o resultado, por exemplo, se␣
 ↪quisermos analisar apenas as notas que NÃO SÃO A:
re.findall("[^A]", notas) #exclui tudo que é A e manda o resto
```

[31]: ['B', 'C', 'B', 'C', 'B', 'C', 'C', 'B', 'C']

2 Quantificadores

[45]:
```
#quantificadores são o número de vezes você quer que um padrão seja combinado/
 ↪correspondido
#O mais básico é expressado por e{m,n} onde: E é a expressão ou caractere que␣
 ↪estamos combinando,
#M é o mínimo de vezez que você quer que ele seja combinado e N é o número␣
 ↪máximo de vezes a ser combinado

#usando as notas como exemplo, quantas vezes esse estudante tirou notas A em␣
 ↪sequencia? ou seja, duas notas A uma após a outra
re.findall('A{2,5}', notas)
#olha, quantas vezes esse aluno tirou notas a em sequência?
#quero que seja no mínimo duas notas e no máximo dez notas A em sequência
```

[45]: ['AA', 'AA', 'AAAA']

[46]:
```
re.findall('A{1,1}A{1,1}', notas)
#a diferença aqui é que o primeiro código procura por no mínimo duas letras A e␣
 ↪máximo de 5 A's em sequência
#no segundo cógido ele procura um A seguido por outro A, nesse caso, 4 A's são␣
 ↪dois A's seguidos por mais dois A's
#ele corta uma sequência de 4 em duas de 2

#essa função não pode ser separada por espaços dentro das {}, o {m,n} tem que␣
 ↪ser colado um no outro, não pode ser {m, n}
#e se não colocar um valor para m ou n, o default sera {1,1}
#se tiver só um número A{2} ele será o m e o n, mínimo e máximo
```

[46]: ['AA', 'AA', 'AA', 'AA']

[48]:
```
#podemos encontrar um padrão decrescente nas notas do aluno, uma piora
re.findall('A{1,10}B{1,10}C{1,10}', notas)
```

[48]: ['ABC']

[49]:
```
#tem outros três quantificadores que são usados de formas mais simples. um␣
 ↪asterisco * para zero
#ou um match, uma interrogação ? para 1 ou 2 combinações ou um sinal de + para␣
 ↪uma ou mais combinações
```

```
with open('ferpa.txt', 'r') as file:
    wiki = file.read()
wiki
```

[49]: 'Overview[edit]\nFERPA gives parents access to their child\'s education records, an opportunity to seek to have the records amended, and some control over the disclosure of information from the records. With several exceptions, schools must have a student\'s consent prior to the disclosure of education records after that student is 18 years old. The law applies only to educational agencies and institutions that receive funds under a program administered by the U.S. Department of Education.\n\nOther regulations under this act, effective starting January 3, 2012, allow for greater disclosures of personal and directory student identifying information and regulate student IDs and e-mail addresses.[2] For example, schools may provide external companies with a student\'s personally identifiable information without the student\'s consent.[2]\n\nExamples of situations affected by FERPA include school employees divulging information to anyone other than the student about the student\'s grades or behavior, and school work posted on a bulletin board with a grade. Generally, schools must have written permission from the parent or eligible student in order to release any information from a student\'s education record.\n\nThis privacy policy also governs how state agencies transmit testing data to federal agencies, such as the Education Data Exchange Network.\n\nThis U.S. federal law also gave students 18 years of age or older, or students of any age if enrolled in any post-secondary educational institution, the right of privacy regarding grades, enrollment, and even billing information unless the school has specific permission from the student to share that specific type of information.\n\nFERPA also permits a school to disclose personally identifiable information from education records of an "eligible student" (a student age 18 or older or enrolled in a postsecondary institution at any age) to his or her parents if the student is a "dependent student" as that term is defined in Section 152 of the Internal Revenue Code. Generally, if either parent has claimed the student as a dependent on the parent\'s most recent income tax statement, the school may non-consensually disclose the student\'s education records to both parents.[3]\n\nThe law allowed students who apply to an educational institution such as graduate school permission to view recommendations submitted by others as part of the application. However, on standard application forms, students are given the option to waive this right.\n\nFERPA specifically excludes employees of an educational institution if they are not students.\n\nThe act is also referred to as the Buckley Amendment, for one of its proponents, Senator James L. Buckley of New York.\n\nAccess to public records[edit]\nThe citing of FERPA to conceal public records that are not "educational" in nature has been widely criticized, including by the act\'s primary Senate sponsor.[4] For example, in the Owasso Independent School District v. Falvo case, an important part of the debate was determining the relationship between peer-grading and "education records" as defined in FERPA. In the Court of Appeals, it was ruled that students placing grades on the work of other students made such work into an "education record." Thus, peer-grading was determined as a violation of FERPA

```
privacy policies because students had access to other students\' academic
performance without full consent.[5] However, when the case went to the Supreme
Court, it was officially ruled that peer-grading was not a violation of FERPA.
This is because a grade written on a student\'s work does not become an
"education record" until the teacher writes the final grade into a grade
book.[6]\n\nStudent medical records[edit]\nLegal experts have debated the issue
of whether student medical records (for example records of therapy sessions with
a therapist at an on-campus counseling center) might be released to the school
administration under certain triggering events, such as when a student sued his
college or university.[7][8]\n\nUsually, student medical treatment records will
remain under the protection of FERPA, not the Health Insurance Portability and
Accountability Act (HIPAA). This is due to the "FERPA Exception" written within
HIPAA.[9]'
```

[52]:
```
#olhando o documento, todos os cabeçalhos começam com [edit] seguidos por um \n
#então se quisermos uma lista de todos os cabeçalhos, podemos usar o findall

re.findall('[a-zA-z]{1,1000}\[edit\]', wiki)
```

[52]:
```
['Overview[edit]', 'records[edit]', 'records[edit]']
```

[56]:
```
#deu certo mas pegou só a última palavra do cabeçalho, queremos o cabeçalho␣
 ↪inteiro
#tem uma opção de usar o \w para dar match em qualquer letra, incluindo dígitos␣
 ↪e números
re.findall('[\w]{1,1000}\[edit\]', wiki)
```

[56]:
```
['Overview[edit]', 'records[edit]', 'records[edit]']
```

[58]:
```
#podemos usar outros três quantificadores para remover as chaves {} do código
#começando pelo asterisco *
re.findall('[\w]*\[edit\]', wiki)
```

[58]:
```
['Overview[edit]', 'records[edit]', 'records[edit]']
```

[62]:
```
#aqui adicionamos espaço usando um espaço após o W
re.findall('[\w ]*\[edit\]', wiki)
```

[62]:
```
['Overview[edit]',
 'Access to public records[edit]',
 'Student medical records[edit]']
```

[67]:
```
#isso nos deu uma lista de títulos numa página da wiki
#agora vamos criar uma lista de titulos
for title in re.findall('[\w ]*\[edit\]', wiki):
    #agora pegamos esse resultado intermediário e separamos no colchete e␣
 ↪pegamos o primeiro resultado
    print(re.split('[\[]', title)[0])
```

```
Overview
Access to public records
Student medical records
```

3 Grupos

[68]:
```
#da pra dar match com padrões diferentes, chamados de grupos
#para agrupar padrões usa-se o parenteses
#é tipo buscar dois padrões de uma vesz só

re.findall('([\w ]*)(\[edit\])', wiki)
```

[68]:
```
[('Overview', '[edit]'),
 ('Access to public records', '[edit]'),
 ('Student medical records', '[edit]')]
```

[71]:
```
#o python quebra o nosso resultado em grupos, mas podemos nos referir aos␣
 ↪grupos por números também
#nesse caso usamos o finditer()

for item in re.finditer('([\w ]*)(\[edit\])', wiki):
    print(item.groups())
```

```
('Overview', '[edit]')
('Access to public records', '[edit]')
('Student medical records', '[edit]')
```

[ ]:
```
#vemos que o método groups() retorna uma tupla contendo os grupos. podemos␣
 ↪pegar um grupo individual usando group(number)
#onde number = número do grupo e group(0) são todas as correspondências
```

[77]:
```
print('Grupo 1 abaixo')
for item in re.finditer('([\w ]*)(\[edit\])', wiki):
    print(item.group(1))
print('-----')
print('Grupo 2 abaixo')
for item in re.finditer('([\w ]*)(\[edit\])', wiki):
    print(item.group(2))
```

```
Grupo 1 abaixo
Overview
Access to public records
Student medical records
-----
Grupo 2 abaixo
[edit]
[edit]
[edit]
```

[92]:
```
#mais uma opção do grupos regex é dar nome aos grupos.par aisso usamos a syntax␣
 ↪(?P<name>)
```

```
#onde os parênteses iniciam os grupos, o ?P indica que é uma extensão de regex␣
 ↪básica e <name> é a chave do dicionário
#que queremos usar envelopada em <>

for item in re.finditer('(?P<title>[\w ]*)(?P<edit_link>\[edit])', wiki):
    #aqui podemos retornar um dicionário para o grupo usando o .groupdict()
    print(item.groupdict()['title'])
    #também podemos imprimir o dicionário completo, mostrando que o [edit]␣
 ↪ainda está lá como um valor da chave "edit_link"
    print(item.groupdict())
    print('----')
```

```
Overview
{'title': 'Overview', 'edit_link': '[edit]'}
----
Access to public records
{'title': 'Access to public records', 'edit_link': '[edit]'}
----
Student medical records
{'title': 'Student medical records', 'edit_link': '[edit]'}
----
```

4 Look-ahead e look-behind matching

[94]:
```
#a gente pega um pedaço do texto mas usa somente o que vem antes ou depois␣
 ↪desse trecho em específico,
#como no caso do [edit], a gente usa ele como referência mas não retorna esse␣
 ↪[edit],
#daí ele é colocado num grupo diferente e o que nós queremos encontrar vai em␣
 ↪outro grupo

for item in re.finditer('(?P<title>[\w ]+)(?=\[edit\])', wiki):
    print(item)
```

```
<re.Match object; span=(0, 8), match='Overview'>
<re.Match object; span=(2715, 2739), match='Access to public records'>
<re.Match object; span=(3692, 3715), match='Student medical records'>
```

[104]:
```
with open('buddhist.txt', 'r') as file:
    wiki = file.read()
wiki
```

[104]:
```
'Buddhist universities and colleges in the United States\nFrom Wikipedia, the
free encyclopedia\nJump to navigationJump to search\n\nThis article needs
additional citations for verification. Please help improve this article by
adding citations to reliable sources. Unsourced material may be challenged and
```

```
removed.\nFind sources: "Buddhist universities and colleges in the United
States"  news ů newspapers ů books ů scholar ů JSTOR (December 2009) (Learn how
and when to remove this template message)\nThere are several Buddhist
universities in the United States. Some of these have existed for decades and
are accredited. Others are relatively new and are either in the process of being
accredited or else have no formal accreditation. The list
includes:\n\nDhammakaya Open University  located in Azusa, California, part of
the Thai Wat Phra Dhammakaya[1]\nDharmakirti College  located in Tucson,
Arizona Now called Awam Tibetan Buddhist Institute
(http://awaminstitute.org/)\nDharma Realm Buddhist University  located in
Ukiah, California\nEwam Buddhist Institute  located in Arlee, Montana\nNaropa
University is located in Boulder, Colorado (Accredited by the Higher Learning
Commission)\nInstitute of Buddhist Studies  located in Berkeley,
California\nMaitripa College  located in Portland, Oregon\nSoka University of
America  located in Aliso Viejo, California\nUniversity of the West  located
in Rosemead, California\nWon Institute of Graduate Studies  located in
Glenside, Pennsylvania\nReferences[edit]\n^ Banchanon, Phongphiphat (3 February
2015).  "" [Getting to know the Dhammakaya network].
Forbes Thailand (in Thai). Retrieved 11 November 2016.\nExternal
links[edit]\nList of Buddhist Universities and Colleges in the world\n'
```

[132]:
```
#a descrição segue um padrão: o nome seguido por um - depois "located in" e por␣
 ↪fim a cidade e o estado
#usaremos o "verbose". com ele você pode escrever regexes em mais de uma linha␣
 ↪e incrementar a leitura. temos que
#indicar explicitamente todos os caracteres de espaço vazio, tanto por colocar␣
 ↪um \ antes dele ou usando
#o valor especial /s. daí podemos escrever o regex mais em forma de código

pattern="""
(?P<title>.*)         #the university title
(\ located\ in\ )    #an indicator of the location
(?P<city>\w*)         #city the university is in
(,\ )                 #separator for the state
(?P<state>\w*)        #the state the city is located in"""

#agora quando chamamos a função finditer() passamos o flag re.VERBOSE como␣
 ↪último parâmetro,
#isso torna muito mais fácil de entender regexes grandes
for item in re.finditer(pattern, wiki, re.VERBOSE):
    #aqui podemos obter o dicionário para o item com .groupdict()
    print(item.groupdict())
```

```
{'title': 'Dhammakaya Open University ', 'city': 'Azusa', 'state': 'California'}
{'title': 'Dharmakirti College ', 'city': 'Tucson', 'state': 'Arizona'}
{'title': 'Dharma Realm Buddhist University ', 'city': 'Ukiah', 'state':
'California'}
{'title': 'Ewam Buddhist Institute ', 'city': 'Arlee', 'state': 'Montana'}
```