💜

Text Manipulation: Regex (Regular Expressions)

Documentação: https://docs.python.org/3/library/re.html

Cheat Sheet: https://www.rexegg.com/regex-quickstart.php

Definition:

→ Regex (ou *expressão regular*) é uma sequência de caracteres que define um padrão de busca em texto. Ele é amplamente utilizado na análise de dados para manipulação, extração e validação de informações em grandes conjuntos de dados.

📌 Para que serve na análise de dados?

1. **Limpeza de Dados**:
    - Remover espaços extras, caracteres especiais indesejados, ou padrões inconsistentes.

Patterns and Character Classes

```
grades="ACAAAABCBCBAA"

re.findall("B",grades) #Outpu

# If we wanted to count the n
# all A's followed immediatel
re.findall("[AB]",grades) #Ou

# This is called the set oper
# alphanumerically. For insta
# a simple regex to parse out
re.findall("[A][B-C]",grades)

# Notice how the [AB] pattern
# [A][B-C] pattern denoted tw
# this pattern by using the p
re.findall("AB|AC",grades) #O

# We can use the caret with t
# the grades which were not A
re.findall("[^A]",grades) #Ou
```

- Exemplo: Remover símbolos de uma coluna com valores monetários (`R$ 1.000,00 → 1000.00`).

2. **Extração de Informações**:
   - Buscar padrões específicos dentro de textos, como e-mails, CPFs, datas e números de telefone.
   - Exemplo: Extrair códigos postais (`\d{5}-\d{3}` para CEPs no Brasil).
3. **Transformação de Dados**:
   - Modificar strings aplicando substituições ou reformatando dados.
   - Exemplo: Converter datas no formato `DD/MM/AAAA` para `AAAA-MM-DD`.
4. **Validação de Dados**:
   - Garantir que entradas sigam um formato correto, como um e-mail válido (`^[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$`).
5. **Filtragem de Dados**:
   - Selecionar apenas linhas que contenham determinado padrão.
   - Exemplo: Filtrar logs de erro que começam com a palavra `"ERROR"`.

Quantifiers:

```
# Quantifiers are the number
# quantifier is expressed as
# number of times you want it

# Let's use these grades as a
re.findall("A{2,10}",grades)
#Output: ['AAAA', 'AA']

# Oh, and if you just have on
re.findall("A{2}",grades) #Ou

# Using this, we could find a
re.findall("A{1,10}B{1,10}C{1
```

Há um exemplo bom de como utilizar isso na procura de títulos em uma página de wikipedia no arquivo da aula sobre Regex.

É uma ótima maneira de entender como fazer **web scrapping**, por exemplo, e coletar mais dados.

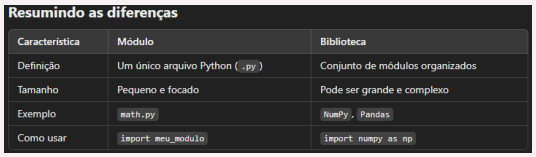
Groups

```
# To this point we have been
# pattern which is matched. B
# and then refer to the group
# pretty natural. Lets rewrit
re.findall("([\w ]*)(\[edit\]
#Output: [('Overview', '[edit
```

Importation:

```
#Importação do módulo "re", q
#regular expressions

import re
```



Toda biblioteca em Python é composta por módulos, mas nem todo módulo é uma biblioteca

Functions:

- **Busca**:
  - `re. search( )`

```
#Função search( ):
text = "This is a good day."
if re.search("good", text):
    print("Wonderful!")
else:
    print("Alas :(")
```

- **Tokenization**: Process where the string is separeted into substrings based on patters.
  - `Findall( )`

```
import re

texto = "O preço do produto A

padrao = r'R\$\d+'  # Express

# Iterar sobre todas as corre
for match in re.finditer(padr
    print(f"Encontrado: {matc
```

- `re.findall()` retorna uma **lista** com todas as correspondências.
- `re.finditer()` retorna um **iterador** que gera objetos `Match` conforme a iteração, economizando memória.

→ A função `.groupdict()` em um objeto `Match` do módulo `re` é usada para retornar um **dicionário** contendo os grupos nomeados da correspondência. Isso é útil quando queremos extrair partes específicas do texto usando grupos nomeados

```
import re

texto = "Os eventos ocorrerão

# Expressão regular com grupo
padrao = r"(?P<dia>\d{2})/(?P

# Encontrando todas as datas
for match in re.finditer(padr
    dados = match.groupdict()
    print(f"Dia: {dados['dia'
```

- `split( )`

```
text = "Amy works diligently.

# This is a bit of a fabricat
re.split("Amy", text)
#Output: ['', ' works diligen

# You'll notice that split ha
# elements of a list. If we w
re.findall("Amy", text)
#Output: ['Amy', 'Amy', 'Amy'
```

The regex specification standard defines a markup language to describes patterns in text!

Anchors

→ Anchors specify the start and/or the end of the string that you are trying to match. The caret character ^ means start and the dollar sign character $ means end. If you put ^ before a string, it means that the text the regex processor retrieves must start with the string you specify. For ending, you have to put the $ character after the string, it means that the text Regex retrieves must end with the string you specify.

```
# Dia: 12, Mês: 03, Ano: 2024
# Dia: 25, Mês: 12, Ano: 2023
```

Look-ahead and Look-behind:

```
# One more concept to be fami
# pattern being given to the
# isolate. For example, in ou
# we actually don't care abou
# we want to use them to matc
# instead with ?= syntax
for item in re.finditer("(?P<
    # What this regex says is
    # of whitespace or regula
    # want this edit put in o
    print(item)

#<_sre.SRE_Match object; span
#<_sre.SRE_Match object; span
#<_sre.SRE_Match object; span
```

```
text = "Amy works diligently.
re.search("^Amy",text)
#Output: <_sre.SRE_Match obje
```