@Odemir Depieri Jr

Trabalhando com

Expressões Regulares

REGEX

O que é Expressões Regulares?

Um aviso antes 

Trabalhar com Regex pode parecer um pouco estranho no começo devido a sintaxe.
Talvez não aprenda de primeira, mas com o tempo e treino irá pegar a manha.

Expressões Regulares (Regex)

De forma simples, **expressão regular** é uma **sequência** de caracteres usados principalmente para encontrar e substituir padrões numa string ou em um arquivo.

Os usos mais comuns de regex são:

- Buscar uma string (Texto)
- Achar uma string (Texto)
- Quebrar uma string em sub strings
- Substituir parte de uma string

Artigo para leitura

https://medium.com/data-hackers/6-dicas-sobre-express%C3%B5es-regulares-em-python-238bed9ccdad

@Odemir Depieri Jr

Operadores da Biblioteca RE

Operadores	Descrição
.	Corresponde a qualquer caractere único, exceto a nova linha “\ n”.
?	Corresponde a 0 ou uma ocorrência do padrão, encontrada à esquerda
+	Corresponde a uma ou mais ocorrências do padrão, encontradas à esquerda
*	Corresponde a 0 ou mais ocorrências do padrão, encontradas à esquerda
\w	Corresponde a um caracter alfanumérico
\W	Corresponde a um caracter não-alfanumérico
\d	Encontra dígitos [0-9]
\D	Encontra não-dígitos
\s	Corresponde com caracter único de espaço em branco (espaço, nova linha, retorno, tab, from)
\S	Corresponde a qualquer caracter que não espaço em branco
\b	Fronteira entre palavra e não-palavra
\B	Oposto de \b
[..]	Corresponde com qualquer caracter único nos colchetes e [^…] corresponde a qualquer caracter único fora dos colchetes
[^…]	Corresponde a qualquer caracter único fora dos colchetes
\	Usado para caracteres de significado especial como \. para corresponder a um período ou \+ para sinal +
^ e $	^ e $ correspondem ao início e final da string, respectivamente
{n, m}	Encontra pelo menos n e no máximo m ocorrências da expressão precedente, se escrevermos com {,m} então irá retornar pelo menos qualquer mínima ocorrência até no máximo m da expressão precedente
a | b	Corresponde a a ou b
()	Agrupa expressões regulares e retorna o texto correspondente
\t, \n, \r	Corresponde a tab, nova linha, retorno

Mão na Massa

Estou usando o Google Colab para compilar o script

```
[167] # Importando a Lib para operar com Expressões Regulares
      import re
```

```
[168] # --- Vamos usar essa frase para nosso tutorial ---
      Frase = 'Vamos entender como funciona Expressões Regulares em 2021 !'
```

```
[169] # Vamos retornar palavras de uma Frase

      # Função findall --> Retorna uma lista contendo todas as correspondências
      Retornar_01 = re.findall('Expressões Regulares', Frase)
      Retornar_02 = re.findall('Vamos', Frase)
      Retornar_03 = re.findall('2021', Frase)

      print( Retornar_01, '\n' )
      print( Retornar_02, '\n' )
      print( Retornar_03 )
```

```
['Expressões Regulares']

['Vamos']

['2021']
```

```
[170] # Parametro [a-z]
      # Vamos retornar uma lista passando o alfabeto como parametro
      # Nesse caso a função irá retornar as letras de A a E
      Retornar_Alfabeto = re.findall('[a-d]', Frase)
      print( Retornar_Alfabeto )
```

```
['a', 'd', 'c', 'c', 'a', 'a']
```

```
[171] # Parametro ...
      # Vamos retonrar uma sequência mas não especificando todas as letras
      # Vamos passar o '...' (pontinhos) para informar a quantidade
      Retornar_Sequencia_01 = re.findall('ente...r', Frase)
      Retornar_Sequencia_02 = re.findall('Ex.......s', Frase)
      Retornar_Sequencia_03 = re.findall('.egulares', Frase)

      print( Retornar_Sequencia_01, '\n' )
      print( Retornar_Sequencia_02, '\n' )
      print( Retornar_Sequencia_03 )
```

```
['entender']

['Expressões']

['Regulares']
```

```
[172] # Parametro ^
      # Vamos retornar algo que começe com algum parametro
      # Nesse caso será retornando a palavra caso localizado
      Retornar_Comeco_01 = re.findall('^Vamos', Frase)
      Retornar_Comeco_02 = re.findall('^entender', Frase)

      print(Retornar_Comeco_01, '\n')
      # nesse caso sera vazia pq a frase não começa 'entender'
      print(Retornar_Comeco_02)
```

```
['Vamos']

[]
```

```
[173] # Parametro $
      # Vamos retornar algo que termine com algum parametro
      # Nesse caso será retornando a palavra caso localizado
      Retornar_Final_01 = re.findall('Vamos$', Frase)
      Retornar_Final_02 = re.findall('!$', Frase)

      # nesse caso sera vazia pq a frase não termina com '!'
      print(Retornar_Final_01, '\n')
      print(Retornar_Final_02)
```

```
[]

['!']
```

```
[174] # Parametro {}
      # Retornando com numero exato de ocorrência
      # Irá verificar se a palavra é seguida de 2 s
      # Nesse caso é verificado se a ultima letra se repete
      # Caso positivo irá retornar o valor
      Retornar_Exato = re.findall('es{2}', Frase)
      print( Retornar_Exato )
```

```
['ess']
```

```
[175] # Parametro |
      # Retornando uma palavra com uma condição 'OU'
      Retornar_Condicao_01 = re.findall('como|talvez', Frase)
      Retornar_Condicao_02 = re.findall('Não|entender', Frase)

      print( Retornar_Condicao_01, '\n' )
      print( Retornar_Condicao_02, '\n' )
```

```
['como']

['entender']
```

```
[176] # Parametro:  \A
      # Retorna uma correspondência se os caracteres especificados
      # estiverem no início da string
      Retornando_Especificado_Inicio_01 = re.findall('\AVamos', Frase)
      Retornando_Especificado_Inicio_02 = re.findall('\Aentender', Frase)

      print(Retornando_Especificado_Inicio_01, '\n')
      print(Retornando_Especificado_Inicio_02)
```

```
['Vamos']

[]
```

```
[177] # Parametro: \b
      # Retorna uma correspondência onde os caracteres especificados
      # estão no início ou no final de uma palavra

      # Passando o \b como inicio
      Retornando_Especificado_Inicio_Fim_01 = re.findall(r'\bmos', Frase)
      # Passando o \b como final
      Retornando_Especificado_Inicio_Fim_02 = re.findall(r'mos\b', Frase)

      print(Retornando_Especificado_Inicio_Fim_01, '\n')
      print(Retornando_Especificado_Inicio_Fim_02 )
```

```
[]

['mos']
```

```
[178] # Parametro: \B
      # Retorna uma correspondência onde os caracteres especificados estão presentes,
      # mas NÃO no início (ou no final) de uma palavra

      # Passando o \B como inicio
      Retornando_Especificado_Inicio_Fim_01 = re.findall(r'\Bona', Frase)
      # Passando o \B como final
      Retornando_Especificado_Inicio_Fim_02 = re.findall(r'ona\B', Frase)

      print(Retornando_Especificado_Inicio_Fim_01, '\n')
      print(Retornando_Especificado_Inicio_Fim_02 )
```

```
['ona']

[]
```

```
[179] # Parametro \d
      # Vamos retornar apenas valores numericos
      Retornar_Apenas_Numeros = re.findall('\d', Frase)
      print( Retornar_Apenas_Numeros )
```

```
['2', '0', '2', '1']
```

```
[180] # Parametro \D
      # Retorna uma correspondência onde a string NÃO contém dígitos
      Retornando_Sem_Numeros = re.findall('\D', Frase)
      print(Retornando_Sem_Numeros)
```

```
['V', 'a', 'm', 'o', 's', ' ', 'e', 'n', 't', 'e', 'n', 'd', 'e', 'r',
```

```
[181] # Parametro \s
      # Retorna uma correspondência onde a string contém um caractere de espaço em branco
      Retornando_Espacos = re.findall('\s', Frase)
      print( Retornando_Espacos )
```

```
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
```

```
[182] # Parametro \S
      # Retorna uma correspondência onde a string NÃO contém um caractere de espaço em branco
      Retornando_Espacos = re.findall('\S', Frase)
      print( Retornando_Espacos )
```

```
['V', 'a', 'm', 'o', 's', 'e', 'n', 't', 'e', 'n', 'd', 'e', 'r', 'c', 'o', 'm', 'o', 'f
```

```
[183] # Parametro \w
      # Retorna uma correspondência em que a string contém quaisquer caracteres de palavra
      # (caracteres de a a Z, dígitos de 0-9 e o caractere sublinhado _)
      Retornando_Letras_Digitos = re.findall('\w', Frase)
      print( Retornando_Letras_Digitos )
```

```
['V', 'a', 'm', 'o', 's', 'e', 'n', 't', 'e', 'n', 'd', 'e', 'r', 'c', 'o', 'm', 'o', 'f
```

```
[184] # Parametro \W
      # Retorna uma correspondência em que a string NÃO contém nenhum caractere de palavra
      Retornando_Caracter_Espcial = re.findall('\W', Frase)
      print( Retornando_Caracter_Espcial )
```

```
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '!']
```

```
[185] # Parametro [xxx]
      # Retorna uma correspondência onde qualquer um dos dígitos especificados
      Retorno_Numeros = re.findall('[202]', Frase)
      print( Retorno_Numeros )
```

```
['2', '0', '2']
```

```
[186] # Parametro [x-x]
      # Retorna uma correspondência para qualquer dígito entre 0 e 9
      Retorno_Numeros_Range = re.findall('[0-1]', Frase)
      print( Retorno_Numeros_Range )
```

```
['0', '1']
```

[187]
```
# Vamos pesquisar palavras na Frase

# Função findall --> Retorna um objeto com a posição se houver uma
#correspondência em qualquer lugar da string

# Procurando um espaço
Procurar_01 = re.search('\s', Frase)

# Procurando a palavra 'como'
Procurar_02 = re.search('como', Frase)

# Procurando a !('Exclamação')
Procurar_03 = re.search('!', Frase)

print( 'O "espaço" esta na posição: ', Procurar_01.start(), '\n' )
print( 'A palavra "como" esta na posição:', Procurar_02.start(), '\n')
print( 'A "Exclamaçao" esta na posição:', Procurar_03.start() )
```

```
O "espaço" esta na posição:  5

A palavra "como" esta na posição: 15

A "Exclamaçao" esta na posição: 58
```

[188]
```
# Quebrando/Separando uma string

# Função Split --> Retorna uma lista onde a string foi dividida em cada partida

# Separando a frase por um espaço
Quebrando_01 = re.split('\s', Frase)

# Separando a frase pela palavra 'como'
Quebrando_02 = re.split('como', Frase)

# Separando a frase pela palavra 'em'
Quebrando_03 = re.split('em', Frase)

print( Quebrando_01, '\n')
print( Quebrando_02, '\n')
print( Quebrando_03)
```

```
['Vamos', 'entender', 'como', 'funciona', 'Expressões', 'Regulares', 'em', '2021'

['Vamos entender ', ' funciona Expressões Regulares em 2021 !']

['Vamos entender como funciona Expressões Regulares ', ' 2021 !']
```

```
# Função Sub --> Substitui uma ou mais correspondências por uma string

# Substituindo o espaço pelo anderline
Substituindo_01 = re.sub('\s', '_|_', Frase)

# Substituido a palavra 'como' por 'COMO'
Substituindo_02 = re.sub('como', 'COMO', Frase)

# Substituido a palavra 'em' por uma frase
Substituindo_03 = re.sub('em', 'NO ANO de', Frase)

print( Substituindo_01, '\n')
print( Substituindo_02, '\n')
print( Substituindo_03)
```

```
Vamos_|_entender_|_como_|_funciona_|_Expressões_|_Regulares_|_em_|_2021_|_!

Vamos entender COMO funciona Expressões Regulares em 2021 !

Vamos entender como funciona Expressões Regulares NO ANO de 2021 !
```

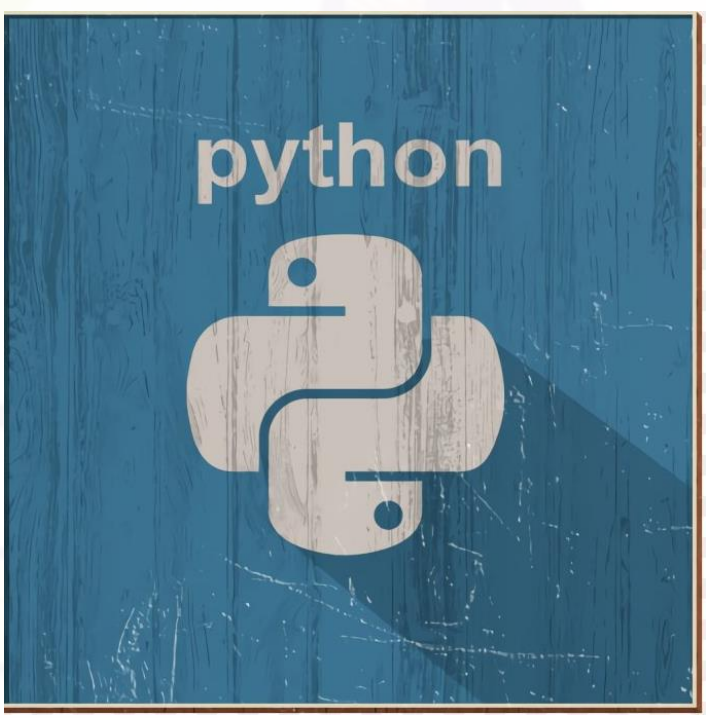
```
[190] # ---- Exemplo do dia a dia ---- #

      # Email, SMS, Mensagem com informe para pagamento

      Mensagem = ( 'Olá Odemir, Segue o numero do codigo de barra para' +
                   'pagamento 222 333 4444 no valor de R$1999,00.' +
                   'com vencimento em 01/01/2021. Obrigado' )

      Filtrar_Numero_Boleto = re.findall(r'\d{3} \d{3} \d{4}', Mensagem)
      Filtrar_Valor = re.findall('R\W\d{4}\W\d{2}', Mensagem)
      Filtrar_Data_Vencimento = re.findall('\d{2}\W\d{2}\W\d{4}', Mensagem)

      print('Codigo de Barra:', Filtrar_Numero_Boleto[0] )
      print('Valor para Pagamento:', Filtrar_Valor[0] )
      print('Data Vencimento:', Filtrar_Data_Vencimento[0] )
```

```
Codigo de Barra: 222 333 4444
Valor para Pagamento: R$1999,00
Data Vencimento: 01/01/2021
```

Final

Esse guia é sobre uso da biblioteca RE e suas utilidades.

Guia da documentação das bibliotecas caso queira mais informação
https://docs.python.org/3/library/re.html



Odemir Depieri Jr

Software Engineer Sr
Tech Lead
Specialization AI