**Expressões regulares**

Nos últimos anos, houve uma dramática mudança no uso do propósito geral de linguagens de programação para Data Science e Machine Learning. Esse nunca foi o caso – Caso isso acontecesse um tempo atrás iria atrair muita atenção!

Isso significa que mais pessoas/empresas estão usando ferramentas como Python/Javascript para alcançar essas necessidades. Isso é onde expressões regulares se tornam muito eficientes. Expressões regulares normalmente é o jeito padrão de manuseio de dados na maioria dessas ferramentas.

Dado a sua aplicabilidade, fica mais visível de como conhecer e como usar apropriadamente.

O que é expressão regular e como é usada?

Em termos gerais, expressão regular é uma sequência de caracteres principalmente usados para encontrar e substituir padrões em uma variável *string* ou em um arquivo. Elas são usadas pela maioria de linguagens de programação como Python, Perl, R, java e entre outras. Então, aprendendo como usá-las ajuda de várias maneiras. Expressões regulares usa dois tipos de caracteres.

1. Meta caracteres: Como o nome sugere, esses caracteres tem um significado especial, parecido com \* em *wild card*.
2. Literais (como a,b,1,2...)

Em Python, nós modulamos ‘**re**’ que irá ajudar com expressões regulares. Então você precisa importar a biblioteca **re** antes de você usar expressões regulares em Python.

**Use esse código:**

**Import re**

Os tipos de usos mais comuns de expressão regular são:

* Procurar um texto (Procurar e igualar)
* Encontrar um texto (Encontrar todos os textos)
* Quebrar um texto em um sub texto (fragmentar)
* Substituir parte de um texto (sub)

Vamos olhar os métodos que a biblioteca ‘**re**’ providencia para usarmos essas funções.

Quais são os vários métodos de expressões regulares?

A biblioteca ‘re’ providencia múltiplos métodos para tratar em uma variável *string*. Eis os métodos mais usados:

1. re.march()
2. re.search()
3. re.findall()
4. re.split()
5. re.sub()
6. re.compile()

Vamos olhar uma por uma.

**re.match(pattern, string):**

Esse método encontra caracteres se estiver no começo da *string*. Por exemplo, colocando *match()* na string ‘AV Analytics AV’ e procurando pelos caracteres ‘AV’ vai encontrar. Entretanto, se procurarmos somente por *Analytics*, o padrão não vai ser encontrado. Vamos fazer em python agora.

**import re**

**result = re.match(r’AV’, ‘AV Analtics Vidhya AV’)**

**print result**

**Output:**

**<\_sre.SRE\_Match object at 0x0000000009BE4370>**

Acima, é mostrado que os caracteres foram encontrados. Para mostrar o caractere encontrado vamos usar é método *group* (retorna os caracteres encontrados). Use “r” no começo da *string*, indica que é uma *string* padrão em Python.

**result = re.match(r’AV’, ‘AV Analytics Vidhya AV’)**

**print(result.group(0))**

**Output:**

**AV**

Vamos agora encontrar ‘Analytics’ na *string* dada. Podemos ver que a *string* não começa com ‘AV’ então não deve encontrar. Vamos ver o que acontece:

**result = re.match(r'Analytics', 'AV Analytics Vidhya AV')**

**print(result)**

**Output:**

**None**

Existem métodos como *start()* e *end()* para saber a posição inicial e final de uma *string*.

**result = re.match(r'AV', 'AV Analytics Vidhya AV')**

**print(result.start())**

**print(result.end())**

**Output:**

**0**

**2**

Acima você consegue ver a posição inicial e final da *string* ‘AV’ na *string* procurada e as vezes ajuda na performance de manipulação com *string*.

**Nota do editor.:**

*match()* serve para encontrar uma *string* no começo da *string* procurada, só irá achar se essa *string* estiver no começo, isso pode ser testado no exemplo abaixo:

**string = re.compile('tempo')**

**p = string.match('av tempo')**

**if p:**

**print(p.group(0))**

**else:**

**print('None found')**

**Output:**

**None found**

[Python Doc Match](https://docs.python.org/3/howto/regex.html#matching-characters)

**re.search(pattern, string)**

É igual ao *match()* mas não nos restrita a achar caracteres somente no começo da *string*. Ao contrário do método anterior, se procurarmos pela palavra ‘Analytics’ vai dar como encontrado.

**result = re.search(r’Analytics’, ‘AV Analytics Vidhya AV’)**

**print(result.group(0))**

**Output:**

**Analytics**

Aqui você consegue ver que *search()* é um método apropriável para achar um padrão em qualquer posição de uma *string* mas retorna somente o primeiro resultado da busca.

**re.findall(pattern, string)**

Ajuda a obter uma lista de todos os padrões encontrados. Não tem importância se a procura começar do começo ou do final. Se vamos usar o método *findall* para procurar ‘AV’ na *string* dada, irá retornar as duas vezes em que aparece AV. Enquanto é procurado em uma *string*