02

Regex com quantificadores

Transcrição

[00:00] Agora já vimos a força que as expressões regulares têm, sabemos melhor o que elas são e sabemos como usá-las dentro do Python. Um problema que estamos enfrentando agora é que esse padrão está enorme, ocupando nossa tela toda. Queremos saber se existe uma forma de reduzir isso.

[00:22] Vamos olhar a documentação do Python e procurar onde encontramos expressões regulares ali. Vou abrir o Google e simplesmente digitar Python doc. Já é logo o primeiro link. Nele temos as versões do Python, e aí vemos as documentações dessas versões específicas. A última é 3.7. A 3.8 está em desenvolvimento.

[00:56] A documentação é dividida em algumas partes. A primeira é "o que tem de novo no Python?", temos um tutorial, e depois temos referências. Vamos abrir o library reference.

[01:25] Ele descreve bem a sintaxe e a semântica da linguagem Python. Built-in é tudo que vem junto com a instalação do Python. Quando você instala, se alguma coisa vem junto com a instalação do Python, é built-in. Por exemplo, lembram que fizemos o import re? Eu não precisei instalar nada a mais do que o Python, então posso dizer que essa é uma biblioteca built-in do Python.

[01:58] Será que essa biblioteca re está aqui dentro? Está. Faz sentido. Expressões regulares ficam dentro de textos, portanto aqui tenho tudo sobre expressões regulares.

[02:16] Se você quiser ler toda essa documentação, você com certeza vai saber tudo sobre expressões regulares do Python. Vou procurar aqui caracteres especiais, os colchetes, porque estamos usando eles.

[02:38] Ele diz que os caracteres dentro dos colchetes podem ser listados individualmente, que é exatamente o que fizemos. O nosso padrão está pronto para encontrar qualquer elemento dentro dos colchetes.

[03:08] Já se parece um pouco com o que queremos. Se dermos dois caracteres separados com um hífen, ele vai encontrar todos os itens entre aqueles caracteres. Então, se eu colocar [0-9], vou encontrar todos as coisas que estão entre esses dois números.

[03:45] Vamos abrir nosso código e testar isso. Vou começar mudando o primeiro: [0-9]. Caso não dê certo, nós nem tentamos nos outros. Mas deu. Então, vou apagar todo o resto e substituir por esse range. Testando, ele continua funcionando.

[04:29] Vimos a documentação e encontramos uma forma de melhorar nosso código. Mas será que não existe alguma forma de eu dizer que estou procurando a mesma coisa quatro vezes? Se fosse um inteiro, eu poderia colocar [0-9]*4 [0-9]*4. Não funcionou. Talvez esse asterisco nem seja um caractere especial da minha biblioteca re.

[05:06] Vamos olhar na documentação se tem alguma coisa. Ele me diz que o {m} específica o número de cópias da expressão regular. Por exemplo, a{6} vai ser seis caracteres "a". Ou seja, minha expressão regular vai buscar seis caracteres especificamente.

[05:38] Parece que isso resolve nosso problema. Vamos testar: [0-9]{4}[0-9]{4}. Funcionou. Olhando a documentação, dando uma lida, você acha.

[06:10] Aprendemos agora a olhar a documentação do Python. Sempre que você tiver uma dúvida, jogue no Google. Se você achar rápido, ok. Caso não ache sua dúvida resolvida de forma rápida, o lugar certo para procurar e onde com certeza terá

tudo que você precisa, é na documentação.

[06:34] Na aula que vem vamos continuar melhorando a expressão regular e deixando ela mais pronta para pegar um celular que tem cinco dígitos no começo, ao invés de quatro.