03

Quantificadores com Intervalos

Transcrição

[00:00] Mantendo o padrão de na dúvida olhar a documentação do Python, caso eu queira encontrar algum número que seja de celular ou de telefone, quero dizer que antes do hífen, posso ter quatro ou cinco dígitos. Vamos ver se tem alguma coisa que pode nos ajudar com isso.

[00:20] Vamos procurar algo parecido com o que queremos. {m, n}, ele vai encontrar de m a n da expressão regular que antecede ele. Ou seja, se eu colocar 4,5 dentro do nosso código, vou poder encontrar ou um, ou outro. Vamos testar: [0-9]{4, 5}. Funcionou, mesmo com cinco números.

[01:18] Essa sintaxe da expressão regular, que são as chaves, posso usar para buscar intervalos. São expressões regulares com quantificadores e com intervalos. Agora, outra proposição para vocês. Supondo que eu tenha três celulares, e esteja mandando um e-mail para o meu colega de trabalho com todos os três celulares. Esse método não suporta esse tipo de coisa.

[01:50] Se eu testar, ele me retorna somente a primeira ocorrência do padrão que ele encontrou. Novamente, vamos dar uma olhada na documentação e ver se tem algum método sem ser o search que pode nos ajudar.

[03:15] O findall parece promissor. Encontre todos. Ele vai retornar todas as vezes que encontrar o padrão. O próprio nome já explicou o que ele faz. Vamos tentar. Ao invés de search, vamos colocar findall. Vou tirar o group porque quero algo bem específico.

[03:49] Ele retornou uma lista com todas as ocorrências desse padrão encontrado dentro do texto. Vou colocar mais um com quatro dígitos antes do hífen. Ele funciona também. E me retorna na ordem em que encontrou. Vamos tentar com mais um. Continua funcionando. O problema é que eu tinha escrito 45678, mas ele me retornou 4567. Ou seja, os caracteres que ele encontrou antes desse 8. Parece que não faz muito sentido isso.

[04:43] Eu posso colocar então de {4, 5}, ele vai me retornar o 8 daí. Mas, no caso do Brasil, números de celular tem cinco números antes do hífen e quatro depois. Então não quero isso acontecendo mesmo.

[04:58] O que quero agora é que meu padrão esteja pronto para receber o hífen ou não. Olhando a documentação novamente, vamos pesquisar.

[05:39] Parece que encontramos algo que funciona para nós, o asterisco. Vamos testar: [0-9] {4, 5} [-] * [0-9] {4}. E vou também tirar o hífen de cima. Ele continua me retornando, mesmo assim. Colocando o asterisco, ele funciona normalmente. Se eu tirar o hífen de todos, ele vai continuar funcionando.

[06:00] Agora, nossa expressão regular consegue pegar número de celular ou de telefone e consegue identificar tendo o hífen ou não. Vocês já sabem bastante sobre expressões regulares, vocês já aprenderam muitos métodos de string que já deram muitas ferramentas para a caixa de ferramentas de desenvolvedores de vocês.

[06:41] Na próxima aula, vamos aprender o que são métodos especiais no Python. Com isso, vamos aprender bastante sobre a arquitetura, o funcionamento da linguagem Python. Vejo vocês lá.