01

O que são expressões regulares

Transcrição

[00:00] Fala, pessoal! Sejam bem-vindos à mais essa aula do nosso curso.

[00:02] Como eu dei o spoiler na semana passada, vamos começar a falar sobre expressões regulares. Vou começar criando algumas variáveis para vermos em que situação uma expressão regular pode ser bem aplicada.

[00:16] O nome da minha variável vai ser: email1 = “Meu numero é 1234-1234”

[00:26] Isso aqui representa o que seria um e-mail para informar um número de celular. email2 = “Fale comigo em 1234-1234 esse é meu telefone” email3 = “1234-1234 é o meu celular”

[00:52] Perceba que aqui a gente não tem um lugar específico para podermos usar o método find. Não temos um nome de argumento que precede o meu número de celular. Ele pode estar no final do texto, no meio do texto ou no começo do texto. Mas, meu número de celular possui uma estrutura bem padronizada. Ele tem cinco dígitos antes e quatro dígitos depois. No caso, usei quatro dígitos para ser um número de telefone, um telefone fixo. Não vai ser celular.

[01:27] Como podemos representar esse padrão com uma forma muito bem definida? Usando a variável “padrão”, que concorda que o número pode variar de 0 a 9. Então: padrao = “[0123456789]”.

[01:47] Isso pode representar o que vem a ser um número dentro do meu número de telefone. Mas eu preciso de quatro. Então, vou dar ctrl+c ctrl+v, ficando então: padrao = “[0123456789] [0123456789] [0123456789] [0123456789] [-] [0123456789] [0123456789] [0123456789] [0123456789]”.

[02:20] Agora sim. Tenho meu padrão bem representado dentro do Python.

[02:26] Agora, o que precisamos fazer é importar a biblioteca que é responsável por trabalhar com expressões regulares. Lembra que aqui a gente tem o extrator de argumentos do URL? A gente criou a classe e depois importamos dentro do main, com from, extratorArgumentosUrl import e a nossa classe.

[02:46] Aqui, nós podemos simplesmente dar o import. “re” é a biblioteca que já vem junto com o Python que é responsável por lidar com expressões regulares. Já montamos o padrão. Nós temos o e-mail. O que precisamos fazer agora é acessar o método específico dentro dessa biblioteca que é responsável por fazer essa busca.

[03:14] Então, vou criar uma variável com o nome “retorno”. O método “search” recebe alguns argumentos. Ele recebe o padrão que eu estou procurando e recebe o texto em que quero buscar isso. Vou buscar dentro do email1 para começar. retorno = re.search(padrao,email1)

[03:38] E vou printar isso: print (retorno)

[03:51] Ele já encontrou alguma coisa. Só que isso não significa nada para a gente. Está aqui o nosso match. O método que o retorno possui é o “group”. print (retorno.group())

[04:04] Agora sim. Usando esse “.group” eu consigo retornar só o que eu quero, só os resultados que foram encontrados. E encontrei dentro do email1, mas vamos tentar agora com o email2. retorno = re.search(padrao,email2)

[04:18] Continua funcionando normalmente, sendo no meio do texto. Vamos tentar com o email3. retorno = re.search(padrao,email3)

[04:25] Funciona também. Agora, vou criar o email4, só para vermos se está funcionando. email4 = “lalalalala 93543-1254 hsiahsahrueuhdiaeu”

[04:43] Aqui, vou substituir por email4: retorno = re.search(padrao,email4)

[04:47] Vamos rodar novamente. Perceba que eu estou dentro de uma string totalmente aleatória, usei um valor diferente, 1234, 1234, e continuou funcionando. Mas é porque esse padrão representa qualquer dígito, qualquer valor de 0 a 9. E se eu encontrar quatro dígitos seguidos, seguidos de um hífen, e seguidos de mais quatro dígitos, a minha regex vai identificar com o search e vai retornar para mim. E só para printar de maneira bonitinha eu coloquei o “.group”. Está aqui como essa biblioteca é poderosa e como ela possui métodos interessante para nós.

[05:29] Na aula que vem, o que vamos fazer é reduzir um pouco esse padrão. Então, vejo vocês lá.