01

O método replace

Transcrição

[00:00] Sejam muito bem-vindos a mais essa aula. Espero que estejam aproveitando o conteúdo, fazendo os exercícios e tirando suas dúvidas.

[00:07] Quero propor a vocês agora o seguinte problema. Supondo que a nossa url que chega na classe venha com algum bug do sistema ou algum erro de digitação do usuário e o valor da moeda origem, ao invés de vir real, peso ou dólar, venha com a seguinte string: moeda destino. Vou colocar no código para vocês verem como vai ficar: url = "https://bytebank.com/cambio?moedaorigem=moedadestino&moedadestino=dólar" (https://bytebank.com/cambio?moedaorigem=moedadestino&moedadestino=d%C3%B3lar%E2%80%9D)

[00:40] Será que isso vai dar algum problema na nossa classe? A minha moeda origem já viria com problema com certeza, porque o valor dela está incorreto. Mas tem mais um bug. A minha moeda destino também foi retornada com bug. Por quê?

[00:52] Lembra que o find para na primeira ocorrência que ele encontra? No momento em que ele encontra a ocorrência da moeda destino, ele para e retorna todo o resto, de acordo com o método que definimos.

[01:11] A solução que proponho para esse primeiro problema mais simples é: ao invés de buscar somente o nome do argumento, busquemos o nome do argumento com o sinal de igual. Se rodarmos, passa bem perto de vir certo, ele me dá "olar" ao invés de dólar.

[01:33] Esse problema acontece porque somamos mais um para ele pular o sinal de igual. Como consideramos o sinal de igual agora, não precisamos do + 1. Cortando-o, vem tudo certo.

[01:47] Nós acabamos de resolver metade do problema. Temos que resolver agora o problema do bug mesmo, da entrada incorreta da nossa classe da moeda origem. Para isso, teríamos que ir na nossa url e trocar o valor de "moeda destino" para algum padrão qualquer.

[02:09] Como o ByteBank é BR podemos trocar para real. O Python possui um método específico que busca alguma coisa dentro de uma string e substitui por outro valor qualquer, é o método replace. Vamos ver como ele funciona no main. Depois jogamos essa lógica dentro da classe.

[02:29] Vou criar a string bytebank. Quero que a minha string nova receba um banco novo, e quero procurar por bank e substituir por Rodrigo: stringNova = string.replace("byte", "rodrigo")

[03:02] Foi. Mas se eu tiver mais um byte, o que vai acontecer? Será que o replace substitui tudo? Vamos testar: string = bytebankbytebyte

[03:32] Sim, ele substitui tudo. Ficaria rodrigobankrodrigorodrigo. Na nossa url temos um caso parecido com isso, porque possuímos o moeda destino, o bug que veio, e o nome do argumento moeda destino, que eu não quero perder. Mas, para nossa sorte, o replace do Python possui mais um argumento que ele pode receber: a quantidade de substituições que quero fazer: stringNova = string.replace("byte", "rodrigo", 1)

[04:00] Se eu coloco 1, eu só substituo o primeiro índice que eu encontrar. Se eu colocar 2, ele substitui os dois primeiros que encontrar. O último não é substituído. O que vamos usar na nossa classe vai ser para substituir um. O que temos que fazer é jogar essa lógica para dentro da classe.

[04:15] Vamos abrir nosso arquivo e vamos criar um método novo. O que eu quero fazer é sobrescrever o nosso atributo url por esse atributo novo, que vai ter um valor certo. def trocaMoedaOrigem(self): self.url = self.url.replace (“moedadestino”, “real”, 1)

[05:10] Além disso, preciso encaixar esse método em algum lugar. Preciso chamar esse método no momento em que encontro minha moeda origem. Caso essa moeda origem seja igual a moeda destino, eu chamo esse método. É um if. if moedaOrigem == “moedadestino”: self.trocaMoedaOrugem()

[05:41] Vou tratar primeiro da origem. Depois trato da moeda destino.

[05:51] Ok, eu descobri que a moeda origem é igual moeda destino, o valor dela, e já retornei, já substituí minha url pela url correta. Vamos testar isso: print (self.url)

[06:28] Rodando isso, ele fez a substituição, mas o valor da minha moeda origem ainda não foi trocado. Ainda está como moeda destino, então preciso depois que faço essa troca encontrar a moeda origem novamente.

[06:45] Caso essa moeda origem seja igual moeda destino, fiz a troca, e então chamo tudo aquilo de novo dentro do meu if.

[06:58] Testando novamente, deu certo. Nós substituímos o valor da moeda origem e depois reescrevemos a variável que retornamos para o método main.

[07:20] Estamos deixando nossa classe bem forte, bem robusta. Ela está pronta para receber mais erros vindos do sistema, ou um erro de digitação do usuário. Na próxima aula, vamos continuar melhorando nossa classe e deixando ela mais robusta para esse tipo de coisa. Ela já está retornando moeda origem e moeda destino. Agora, precisamos retornar o valor.

[07:46] Na próxima aula continuamos criando-a então. Obrigado por assistir, e até lá.