01

None, empty e o if do Python

Transcrição

[00:00] Sejam bem-vindos a mais essa aula do nosso curso. Espero que estejam gostando e que estejam fazendo os exercícios. Vamos começar.

[00:06] Quero começar a construir a classe que vai ser responsável por extrair os argumentos de dentro da nossa string. É bom sempre ter uma classe, para organizar melhor o código, você cria métodos bem definidos, que fazem coisas bem definidas. Você pode reutilizar essas classes e métodos em algum lugar depois, caso você queira fazer manutenção no seu código é mais fácil, porque vai estar tudo organizado.

[00:36] Chega de papo e vamos criar a classe. Vamos criar um arquivo dentro da mesma pasta com o nome ExtratorArgumentosUrl. E vamos criar uma classe dentro desse arquivo com o mesmo nome: class ExtratorArgumentosUrl

[00:57] Em que momento queremos receber a url dentro de algum lugar? Dentro do nosso código main, no caso, e futuramente dentro da nossa aplicação web do Byte Bank. É bom receber o valor no momento em que o objeto é criado, no momento em que estanciamos essa classe.

[01:15] Precisamos de um construtor para receber esse valor: def **init**(self, url):

[01:26] Até aqui tudo certo. Posso simplesmente criar um self.url = url. Vai funcionar naturalmente. Mas, e se der algum bug dentro do sistema do Byte Bank ou se o usuário resolver passar algum valor vazio para nossa classe? Temos que deixar nossa classe pronta para lidar com esse tipo de problema.

[01:57] Antes, quero mostrar um conceito através de uma imagem. Existe diferença entre uma coisa vazia e uma coisa nula. Uma coisa vazia de fato existe, só que está vazio. E uma coisa nula nem sequer existe. Para situar melhor, null no Python é igual a none.

[02:25] Vamos fazer alguns testes no código. Vou criar a variável: string = None. Vou dar um print (string is None). Estou perguntando para o Python. Ele diz que é verdade.

[03:07] Mas se eu colocar uma coisa vazia: string = “”. Ele vai dizer que é falso, porque essa pergunta é muito específica. Eu estou perguntando só se é nome. Não perguntei se é vazio ou nome. A boa notícia é que em geral o if espera um booleano, mas no Python, quando você não passa um booleano para o if, ele também consegue tratar essa informação que foi passada da seguinte maneira: if string: print (“Tem algo Aqui”) else: print(“Não tem nada Aqui”)

[04:04] Se eu coloco string = “Rodrigo”, ele me diz que dentro da minha string existe alguma coisa. Mas se eu colocar string = None, ele me diz que não existe nada. Foi para a segunda condição. E se eu não passar nada: string = “”, ele me diz que não tem nada.

[04:39] O Python sabe diferenciar quando uma string é vazia usando o if. Vamos fazer um teste colocando string = 12. Ele indica que tem algo. Se eu passar string = 0, ele diz que não tem nada.

[04:58] A nossa ideia agora é conseguir jogar essa lógica de alguma forma para dentro da nossa classe. É sempre bom criar métodos para organizar bem o código e não embolar muita coisa dentro de um método só. def urlEhValida (self, url): if url: return True else: return False

[05:50] Posso usar isso dentro do meu construtor. Antes de eu atribuir alguma coisa para o meu atributo URL da minha classe, vou fazer essa verificação: If self.urlEhValida(url): self.url = url else: raise LookupError ("Url inválida!!!!!!")

[06:25] Podemos fazer muita coisa aqui. Podemos printar alguma coisa na tela, podemos dizer ao usuário que ele passou alguma coisa errada. Mas eu prefiro lançar uma exceção, até para aprendermos algo novo, diferente.

[06:40] O raise significa que vou jogar alguma coisa para o meu usuário. Vamos voltar para o código main e tentar usar essa classe.

[07:02] A primeira coisa que temos que fazer é importar ela: from ExtratorArgumentosUrl import ExtratorArgumentosUrl. Quero criar uma instância: url = "https://www.bytebank.com.br/cambio?moedaorigem=real&moedadestino=dolar&valor=700" (https://www.bytebank.com.br/cambio?moedaorigem=real&moedadestino=dolar&valor=700%E2%80%9D) argumento = ExtratorArgumentosUrl (url) print (argumento)

[07:52] Rodando isso, já tenho o objeto do tipo ExtratorArgumentosUrl alocado em algum lugar da memória do meu computador. E se eu tentar passar outra coisa? url = None

[08:10] Ele procurou o erro e lançou a exceção que nós criamos, url inválida. Mas se eu colocar alguma coisa vazia a mesma coisa acontece.

[08:23] Nossa classe está pronta para trabalhar com URLs que sejam none, que não existam, ou com URLs vazias. Percebam uma coisa interessante no método que acabamos de definir: ele tem o self, mas nós não usamos para nada, certo? Nós usamos ele também no init e precisamos dele para criar nosso atributo. Qual a diferença de uma coisa para a outra?

[08:50] Significa que para usar o init preciso de algum objeto, alguma instância do main, ou de algum outro lugar qualquer. Já o método urlEhValida não precisa dessa instância, porque é um método estático, enquanto o outro é um método de instância.

[09:05] Vou provar para vocês que consigo fazer uma modificação nesse método, chamá-lo diretamente do main. Mas antes, precisamos dizer para o Python que esse método é de fato estático. Para isso, usamos uma coisa bem bacana chamada decorador. São formas de modificar o comportamento de funções ou classes sem precisar modificar o código.

[09:36] Um decorador é identificado através do @. Se começa com @, é um decorador. O decorador que queremos é o @staticmethod. Eu vou dizer de fato que esse método é estático.

[09:52] Percebe que o PyCharm até me disse que isso aqui não é mais necessário, que estou com o self, mas não tenho ele e não preciso ter ele. Antes, ele estava com outra cor. E minha classe vai continuar funcionando normalmente.

[10:13] Se eu passar o url de novo, vai continuar rodando normalmente. E agora não preciso fazer argumento igual alguma coisa. Posso simplesmente printar e chamar o método diretamente da classe. Funciona perfeitamente. Antes eu não conseguiria, daria erro, porque ele não sabe que é uma classe estática.

[11:10] Até aqui tudo tranquilo, nossa classe está funcionando normalmente. Na próxima aula vamos continuar criando esses métodos que serão responsáveis por extrair os argumentos da url.