02

Dando um nome a nossa classe

Transcrição

[00:00] E se no lugar de printar o len da minha classe, o tamanho que a classe tem, eu quiser printar a minha classe só? Vou dar um print na instância argumentosUrl que foi criada lá em cima.

[00:14] Ele me dá vários caracteres. É alguma coisa específica do meu sistema operacional, mas que para o usuário não faz o menor sentido ver. Para nós que estamos criando já não faz sentido.

[00:35] A boa notícia é que da mesma que tinha um método especial que fazia o len funcionar de forma melhor com a nossa classe, existe também um método especial que faz o print funcionar. Esse método cria uma representação string da nossa classe. O nome dele e é str. Aqui é muito parecido, eu preciso dar um return de alguma coisa. def__str__(self): return “Minha classe”

[01:09] Ele retornou alguma coisa diferente. Retornou “Minha classe”. Não faz muito sentido, porque é muito genérico. Eu quero algo mais específico. Tenho que decidir o que quero que seja mostrado quando printo minha classe. Pode ser a url inteira. def__str__(self): return self.url

[01:30] O que ele me retornou eu podia ter feito a qualquer momento com o get. Acho que não quero criar um método especial str só para isso. Vou começar a criar uma variável que vai ser a representacaoString = self.extraiValor(). Vou retornar essa variável. def__str__(self): representacaoString = self.extraiValor() return representacaoString

[02:01] Rodando o main mais uma vez, ele me retorna 1500. Deu o que eu queria. Mas só isso faz sentido para nós? Acho que não. Vamos concatenar isso. def__str__(self): representacaoString = self.extraiValor() return representacaoString + self.extraiArgumentos()

[02:27] Eu estou tentando passar uma tuple dentro desse meu método especial str, mas não posso. Tenho que passar somente strings. Então vou tentar popular duas variáveis. def__str__(self): moedaOrigem, moedaDestino = self.extraiArgumentos () representacaoString = self.extraiValor() + “ “ + moedaOrigem + “ “ + moedaDestino return representacaoString + self.extraiArgumentos()

[02:59] Já funcionou. Eu peguei 1500 e o real. Eu consegui agora representar de forma bem mais clara para o meu usuário o que tem dentro dessa instância. Mas posso trazer ainda maior. Posso fazer de uma forma que eu tenha textos indicando o que significam esses nomes. Por exemplo: def__str__(self): moedaOrigem, moedaDestino = self.extraiArgumentos () representacaoString = “Valor:” + self.extraiValor() + “ “ + moedaOrigem + “ “ + moedaDestino return representacaoString + self.extraiArgumentos()

[03:48] Existe outra função do Python que me ajuda com isso. É a format. Dentro do format, eu vou indicar o que vai estar dentro de cada uma das chaves. def__str__(self): moedaOrigem, moedaDestino = self.extraiArgumentos () representacaoString2 = “Valor:” + self.extraiValor() + “ “ + moedaOrigem + “ “ + moedaDestino representacaoString = “Valor: {}\n Moeda Origem: {}\n Moeda Destino: {}\n”.format(self.extraiValor(), moedaOrigem, moedaDestino) return representacaoString + self.extraiArgumentos()

[05:03] Agora temos valor igual a 1500, moeda origem igual a real e moeda destino igual a dólar. Simplesmente por printar a minha instância da classe recebo de forma bem mais estruturada tudo que tem ali dentro. Certo que isso não serve para eu fazer uma conversão de alguma coisa, porque isso é string, mas para uma representação do que tem ali dentro, está ótimo.

[05:31] Por enquanto é isso. Na próxima aula vamos conseguir começar a comparar essas instâncias. Um exemplo: se eu criar um argumentosUrl2 e printar, perguntando se o argumentosUrl é igual ao argumentosUrl2, sendo que não mudei nada, ele vai me dizer que é falso. Na próxima aula vamos ver como resolver esse problema.