01

O laço com for

Transcrição

Voltando ao código do nosso jogo de adivinhação, implementamos o loop `while`, no qual temos uma variável `rodada` que começa com o valor 1, e é incrementada dentro do loop, que por sua vez tem uma condição de entrada, que é a `rodada` ser menor ou igual ao total de tentativas, que é 3.

Ou seja, a `rodada` tem um valor inicial, que é 1, e vai até 3. Fazemos um laço começando com um valor inicial, até um valor final, sempre incrementando esse valor a cada iteração.

Em casos como esse, existe um outro loop que simplifica essa ideia de começar com um valor, e incrementá-lo até chegar em um valor final, o loop `for`.

Entendendo o for

Para entender o loop `for`, podemos ir até o console do Python para ver o seu funcionamento. A ideia é nós definirmos o valor inicial e o valor final, que o loop o incrementa automaticamente. Para definir o valor inicial e final, utilizamos a função `range`, passando-os por parâmetro, definindo assim a série de valores. A sintaxe é a seguinte

```
>>> para variável em série de valores:
...     faça algo
```

Isso, em Python, pode ficar assim:

```
>>> for rodada in range(1,10):
...
```

Na primeira iteração, o valor da variável `rodada` será 1, depois 2 e até chegar ao **valor final da função `range` menos 1**, isto é, o segundo parâmetro da função não é inclusivo. No exemplo acima, a série de valores é de 1 a 9. Podemos confirmar isso imprimindo o valor da variável `rodada`:

```
>>> for rodada in range(1,10):
...     print(rodada)
...
1
2
3
4
5
6
7
8
9
```

Com a função `range`, podemos definir um *step*, que é o intervalo entre os elementos, por padrão o *step* é 1. Definimos-o passando um terceiro parâmetro para a função:

```
>>> for rodada in range(1,10,2):
...     print(rodada)
...
1
3
5
7
9
```

Mas não necessariamente precisamos usar a função `range` no `for`, podemos passar os valor manualmente:

```
>>> for rodada in [1,2,3,4,5]:
...     print(rodada)
...
1
2
3
4
5
```

O resultado é o mesmo, mas o código fica mais verboso.

Utilizando o for no jogo

Voltando ao nosso jogo, não vamos mais utilizar o `while`, e sim o `for`, começando no 1 e indo até o total de tentativas. Para isso precisamos remover a declaração da variável `rodada` * e o seu incremento dentro do loop:

```
numero_secreto = 42
total_de_tentativas = 3

for rodada in range(1, total_de_tentativas):
    print("Tentativa {} de {}".format(rodada, total_de_tentativas))
    chute_str = input("Digite o seu número: ")
    print("Você digitou: ", chute_str)
    chute = int(chute_str)

    acertou = numero_secreto == chute
    maior = chute > numero_secreto
    menor = chute < numero_secreto

    if (acertou):
        print("Você acertou!")
    else:
        if (maior):
            print("Você errou! O seu chute foi maior que o número secreto.")
        elif (menor):
            print("Você errou! O seu chute foi menor que o número secreto.")

print("Fim do jogo")
```

É importante saber que o `for` não deve ter parênteses.

Podemos testar e ver que só fizemos 2 tentativas. Isso porque, como foi falado anteriormente, o segundo parâmetro da função `range` não é inclusivo, no caso do nosso jogo, `range(1,3)` irá gerar a série 1 e 2 somente. Logo vamos somar 1 ao total de tentativas dentro da função `range` :

```
numero_secreto = 42
total_de_tentativas = 3

for rodada in range(1, total_de_tentativas + 1):
    print("Tentativa {} de {}".format(rodada, total_de_tentativas))
    chute_str = input("Digite o seu número: ")
    print("Você digitou: ", chute_str)
    chute = int(chute_str)

    acertou = numero_secreto == chute
    maior = chute > numero_secreto
    menor = chute < numero_secreto

    if (acertou):
        print("Você acertou!")
    else:
        if (maior):
            print("Você errou! O seu chute foi maior que o número secreto.")
        elif (menor):
            print("Você errou! O seu chute foi menor que o número secreto.")

print("Fim do jogo")
```

Agora podemos testar novamente o nosso jogo, e ver que tudo está funcionando perfeitamente!