02

Trabalhando com listas

Lista

Vejamos outra situação. Vamos declarar três variáveis que representam convites:

```
>>> convite1 = 'Flavio Almeida'
>>> convite2 = 'Nico Steppat'
>>> convite3 = 'Romulo Henrique'
```

Se precisarmos de mais um convite? Mais uma variável precisará ser criada. Se tivermos 100 convites? Com certeza você perderá algumas calorias digitando todas essas variáveis. Será que existe uma maneira de agruparmos `String`, algo como se fosse uma lista? Sim!

Declaração de uma lista

```
>>> convites = ['Flavio Almeida', 'Nico Steppat', 'Romulo Henrique']
```

Conseguimos numa única variável armazenar três strings, excelente! Porém, como fazemos para acessar cada elemento da lista individualmente. Lembra do operador `slice`?

Acessando elementos de uma lista

Podemos passar a posição do elemento que desejamos obter da lista, lembrando que a posição começa de zero até o tamanho da lista menos um:

```
>>> convites[0]
'Flavio Almeida'
>>> convites[1]
'Nico Steppat'
>>> convites[2]
'Romulo Henrique'
```

Repare que em todas essas situações exibimos a string contida na posição da lista. Porém, o operador slice permite extrair uma lista de elementos da lista existente:

```
>>> convites[0:2]
['Flavio Almeida', 'Nico Steppat']
```

No exemplo acima, uma lista com o primeiro e segundo elemento é retornada. Lembre-se que a posição final do slice é não inclusiva, isto é, não entra na seleção.

Podemos omitir a posição final do slice, fazendo com que seja retornado o elemento da posição inicial em diante:

```
>>> convites[1:]
['Nico Steppat', 'Romulo Henrique']
```

Adicionando elementos

Excelente, mas se quisermos adicionar novos elementos de uma lista já criada? Faz todo sentido, pois um convite pode ter chegado na última hora, não? Para isso, utilizamos a função **append**. Não se preocupe agora com funções, mas saiba que elas guardam um código para podemos chamar depois, em nosso caso, a função **append** está presente em todas as strings que criamos. Mais tarde explicaremos este mistério:

```
>>> convites.append('Vitor Mattos')
>>> convites
['Flavio Almeida', 'Nico Steppat', 'Romulo Henrique','Vitor Mattos']
```

Ainda podemos adicionar um estranho no ninho:

```
>>> convites.append(10)
['Flavio Almeida', 'Nico Steppat', 'Romulo Henrique','Vitor Mattos', 10]
```

Este último exemplo mostra que uma lista do Python pode guardar valores de qualquer tipo! Apesar disso, não queremos poluir nossa lista com diferentes tipos.

Removendo elementos

Como removemos o Number que adicionamos na lista? Além da função `append` utilizada para incluir, existe também a função `remove` que recebe como parâmetro o elemento que desejamos remover:

```
>>> convites.remove(10)
>>> convites
['Flavio Almeida', 'Nico Steppat', 'Romulo Henrique','Vitor Mattos']
```

Excelente, removemos um item de nossa lista. E se quisermos remover 'Romulo Henrique'? Realizamos o mesmo procedimento, passando uma String que corresponde ao elemento que desejamos remover:

```
>>> convites.remove('Romulo Henrique')
>>> convites
['Flavio Almeida', 'Nico Steppat','Vitor Mattos']
```

Adivinha o que está faltando agora? Praticar! Hora de encarar os exercícios!