Computational Thinking With Python

Prof. Gilberto Alexandre das Neves
profgilberto.neves@fiap.com.br

FIAP

Lists

List

O objeto **List** em *Python* é utilizado para armazenar múltiplos valores em uma única variável.

O **List** em *Python* é um dos 4 tipos de variáveis que são utilizados para armazenar coleções de dados (outros tipos veremos futuramente).

Criamos um **List** e inicializamos seus valores usamos o **colchetes** [ ].
Exemplo:

```
1  frutas = ["maçã","uva","tomate"]
2  print(frutas)
```

List

O **List** em *Python* ordena seus valores por **índices** (começando em **0**). Não possui um tamanho fixo (seu tamanho muda de acordo com os elementos que possui).

Os valores de um **List** podem ser duplicados, já que o acesso a eles é pelo seu índice. Para verificar o tamanho de um **List** utilizamos a função **len()**. Para adicionar novos valores em um **List** já existente utilizamos a função **append()**.

Podemos ter **Lists** de tipos diversos de dados e inclusive **List** com elementos dos mais variados tipos.

```
1 frutas = ["maçã","uva","tomate"]
2 numeros = [21, 45, 73, 12, 45, 1, 45]
3 logicos = [True, True, False, True]
4 misto = ["Lero-lero", True, 23, "Gerônimo", 4.77]
5 print(frutas, numeros, logicos, misto)
```

Exemplo 1

```
1 frutas = ["maçã","uva","tomate"]
2 print(frutas)
3 print(frutas[2])
4 print(len(frutas))
5 frutas.append("cereja")
6 frutas.append("acerola")
7 print(frutas)
8 print(len(frutas))
9 print(frutas[-1])
```

List

Podemos acessar uma faixa de valores indicando o índice inicial (incluso), dois pontos e o índice final (não incluso).

```
1 frutas = ["maçã","uva","kiwi","cereja","pêra","manga"]
2 print(frutas[2:5])
```

Outra opção é usar o dois pontos e um índice final (não incluso), assim ele vai acessar desde o índice inicial até um elemento antes do índice final.

```
1 frutas = ["maçã","uva","kiwi","cereja","pêra","manga"]
2 print(frutas[:4])
```

List

FIAP

Outra opção é indicar um índice inicial (incluso) e o dois pontos, assim ele vai acessar do índice inicial até o fim da **List**.

```
1 frutas = ["maçã","uva","kiwi","cereja","pêra","manga"]
2 print(frutas[3:])
```

Outra opção é usar índices negativos, indicando um índice “final” (incluso), dois pontos e o índice “inicial” (não incluso).

```
1 frutas = ["maçã","uva","kiwi","cereja","pêra","manga"]
2 print(frutas[-4:-1])
```

Podemos verificar se um elemento existe em um **List** com o comando **if** em conjunto com o comando **in**.

```
1 frutas = ["maçã","uva","kiwi","cereja","pêra","manga"]
2 if "cereja" in frutas :
3     print("Sim, nós temos cerejas!")
```

List

Podemos substituir de um elemento da **List** por outro pelo seu índice.

```
1 frutas = ["maçã","uva","kiwi"]
2 frutas[1] = "manga"
```

Podemos substituir valores de um **List** indicando um índice inicial (incluso), dois pontos e um índice final (não incluso).

```
1 frutas = ["maçã","uva","kiwi"]
2 frutas[1:3] = ["manga","caqui"]
```

Podemos inserir um novo elemento no **List** em uma posição específica com a função **insert()** e o índice onde desejamos incluir o elemento.

```
1 frutas = ["maçã","uva","kiwi"]
2 frutas.insert(1,"banana")
```

List

Com a função **extend()** podemos incluir um outro **List** em nosso **List** original.

```
1 frutas = ["maçã","pêra","banana"]
2 citricos = ["limão","laranja"]
3 frutas.extend(citricos)
4 print(frutas)
```

Com a função **remove()** podemos remover um elemento específico do **List**.

```
1 frutas = ["maçã","pêra","banana","kiwi"]
2 frutas.remove("banana")
3 print(frutas)
```

List

Outra forma de remover um elemento de um **List** é utilizando a função **pop()** e indicar seu índice.

```
1 frutas = ["maçã","pêra","banana","kiwi"]
2 frutas.pop(2)
3 print(frutas)
```

Se utilizar a função **pop()** sem indicar um índice ele irá remover o último elemento do **List**.

```
1 frutas = ["maçã","pêra","banana","kiwi"]
2 frutas.pop()
3 print(frutas)
```

List

Podemos navegar pelos elementos de um **List** com o comando **for**:

```
1 frutas = ["maçã","pêra","banana","kiwi"]
2 for i in frutas :
3     print(i)
```

Outra opção é passar seu **índice** para navegar entre seus elementos n comando **for**:

```
1 frutas = ["maçã","pêra","banana","kiwi"]
2 for i in range(len(frutas)) :
3     print(frutas[i])
```

List

Podemos ordenar os elementos de um **List** com a função **sort()**

```
1 frutas = ["maçã","pêra","banana","kiwi"]
2 frutas.sort()
3 print(frutas)
```

A função **sort()** também funciona com **List** numérico

```
1 numeros = [50,10,33,40,15,60,20]
2 numeros.sort()
3 print(numeros)
```

List

Para ordenar de forma decrescente os elementos de um **List** com a função **sort()** usamos o argumento **reverse** (indicando seu valor como **True**).

```
1 frutas = ["maçã","pêra","banana","kiwi"]
2 frutas.sort(reverse = True)
3 print(frutas)
4
5 numeros = [50,10,33,40,15,60,20]
6 numeros.sort(reverse = True)
7 print(numeros)
```

List

Ao ordenar um **List** de ***strings*** com palavra maiúscula e minúsculas podemos obter um resultado incomum.

```
1 frutas = ["maçã","Pêra","banana","Kiwi"]
2 frutas.sort()
3 print(frutas)
```

Para solucionar este problema podemos utilizar o parâmetro **key** na função **sort()**.

```
1 frutas = ["maçã","Pêra","banana","Kiwi"]
2 frutas.sort(key = str.lower)
3 print(frutas)
```

List

Não é possível copiar um **List** para outro **List** apenas usando o símbolo de atribuição **=** (feito desta maneira, alterações no **List** original também afetam a cópia).

```
1 frutas = ["maçã","pêra","banana"]
2 copiafrutas = frutas
3 frutas.append("kiwi")
4 print(frutas)
5 print(copiafrutas)
```

List

A forma indicada para se copiar um **List** para outro **List** é usando a função **copy()** ou a função **list()**.

```
1 frutas = ["maçã","pêra","banana"]
2 frutas2 = frutas.copy()
3 frutas3 = list(frutas)
4 frutas.append("kiwi")
5 print(frutas)
6 print(frutas2)
7 print(frutas3)
```

FIAP

Exercícios

Exercícios

FIAP

1. FUPQ peça para o usuário digitar uma lista de 15 números inteiros. Ao final exiba a somatória de todos os valores da lista.

2. FUPQ peça para o usuário digitar uma lista de 20 números reais. Ao final exiba a média dos valores digitados contidos na lista.

3. FUPQ peça para o usuário digitar números inteiros (e armazene estes valores em uma lista), ele deve continuar digitando até digitar 0 para encerrar a digitação. Ao final exiba: o tamanho da lista, a lista completa em ordem crescente, a quantidade de números negativos da lista e a quantidade de números primos positivos da lista.

Referências

Introdução à programação com Python. Nilo Menezes. Novatec, 2019.

Curso Intensivo de Python: Uma introdução prática e baseada em projetos à programação. Eric Matthes. Novatec, 2016.

Até breve!