_____ / _____ / _____

Listas em Python

Uma lista em Python é uma coleção de coisas. Os elementos individuais podem ser acessados pelo seu índice dentro da lista. Em outras linguagens (C, Java, C++, APL, PHP, etc) este conceito é conhecido como vetor, e nelas todos os elementos têm que ter o mesmo tipo. Essa restrição não existe em Python, mostra de sua modernidade. Em C, uma lista de inteiros só pode ter inteiros. Em Python uma lista pode ter inteiros, nomes, caracteres, todos misturados.

Uma lista em Python é caracterizada pela presença dos colchetes delimitando os elementos que são separados por uma vírgula. Por exemplo `A=[1, 5, 7, 90]`. Aqui, a lista de nome `A` tem 4 elementos, a saber `1`, `5`, `7` e `90`.

A lista pode conter zero elementos, o que se conhece como lista vazia, e ela é criada assim a partir da definição ou seja `A=[]`. Novos elementos podem ser adicionados ao final de uma lista já existente, usando-se a operação `append`. Acompanhe o exemplo:

```
>>> A = []
>>> A.append(5)
>>> A
[5]
>>> A.append('viva')
>>> A
[5, 'viva']
```

Indexação

Os elementos individuais de uma lista podem ser acessados (tanto para leitura quanto para gravação) usando-se um índice. Indices positivos vão da esquerda para a direita e negativos da direita para a esquerda. O primeiro elemento da lista à esquerda é o elemento zero. O segundo é o 1, o terceiro o 2 e assim por diante até o $n-1$-ésimo que o último elemento de uma lista de n elementos.

Já no sentido oposto, o último elemento é o -1, o penúltimo é o -2 e assim por diante até o $-n$ que é o primeiro elemento de uma lista de n elementos.

Acompanhe o que se disse no exemplo:

```
    LA  = [  5, 33, 21, -4, 126.7, 'oba', 18]
índice >= 0:  0   1   2   3     4      5    6
índice <0:   -7  -6  -5  -4    -3     -2   -1
```

A lista pode ser consultada, como em `>>>LA[3]` que é igual a `-4` como pode ser alterada, fazendo-se `LA[1]=100`, o que vai substituir o valor 33 pelo valor 100 dentro da lista.

Fatiamento

É uma operaçao muito comum em Python envolvendo partes (fatias) de listas e que é fortemente baseada na indexação de listas. O formato de um fatiamento é

```
inicio:final:incremento
```

Os 3 valores podem ser omitidos, e quando o incremento é omitido, omite-se também o segundo `:`. O `início` indica qual o elemento inicial da fatia, e o `final` indica o seguinte ao último elemento da fatia. Quando `início` é omitido, o Python assume o primeiro elemento da lista, quando o `final` é omitido, assume-se o último e quando o `incremento` é omitido assume-se o valor 1. Acompanhe e procure entender cada um dos exemplos a seguir.

```
>>> LB=[5, 17, 33, 80, 91]
>>> LB[:-1]
[5, 17, 33, 80]
>>> LB[1:-1]
[17, 33, 80]
>>> LB[::2]
[5, 33, 91]
>>> LB[::-1]
[91, 80, 33, 17, 5]
>>> LB[::-2]
[91, 33, 5]
>>> LB[1:3]
[17, 33]
>>> LB[-3:-1]
[33, 80]
>>> LB[:3]
[5, 17, 33]
>>> LB[3:]
[80, 91]
>>> LB[:]
[5, 17, 33, 80, 91]
```

Para operar este conceito é interessante desenhar um esquema identificando e numerando os limites de cada elemento. Faça isso marcando os colchetes e as vírgulas da lista, positivos da esquerda para a direita começando em zero e negativos da direita para a esquerda. Agora imagine o número fornecido como uma faca cortando uma torta no local indicado. Veja:

```
LB = [   5,  17,  33,  80, 91 ]
     |    |   |   |    |    |
     0    1   2   3    4    5
    -5   -4  -3  -2   -1
```

Outras operações

lista.insert(índice, elemento): Insere o elemento dado na lista citada, após o índice especificado. Esta operação não retorna nada.

```
>>> LC={5, 33, 9.5, 'oba', 'além']
>>> LC.insert(3, 123)
>>> LC
[5, 33, 9.5, 123, 'oba', 'além']
```

lista.remove(elemento): Remove o primeiro elemento citado. Esta operação não retorna nada.

```
>>> LC=[5, 33, 9.5, 'oba', 'além']
>>> LC.remove(9.5)
>>> LC
[5, 33, 'oba', 'além']
```

lista.pop([índice]): Remove e DEVOLVE o elemento citado. Como o índice é opcional, se omitido, remove e devolve o último elemento.

```
>>> LC=[5, 33, 9.5, 'oba', 'além']
>>> LC.pop()
'além'
>>> LC
[5, 33, 9.5, 'oba']
```

lista.sort(): Ordena a lista no local.
lista.reverse(): Reverte a lista no local
len(lista): Retorna o tamanho da lista
min(lista): Retorna o valor mínimo da lista. Só funciona em listas homogêneas contendo números. Senão dá erro.
max(lista): Retorna o valor máximo da lista. Só funciona em listas homogêneas contendo números. Senão dá erro.
sum(lista): Retorna a soma da lista. Só funciona em listas homogêneas contendo números. Senão dá erro.

```
>>> LD=[1, 4, 6.8, 9]
>>> len(LD)
4
>>> min(LD)
1
>>> max(LD)
9
>>> sum(LD)
20.8
```

Estas são ALGUMAS das operações envolvendo listas. Para ver tudo, consulte a documentação do Python.

Exemplo

A seguir uma instância feita e correta para os exercícios.

```
1. Considere GO=[3, 8, 11, 12, 26, 27]
Efetue todos os fatiamentos.
Ao final, some todos os elementos.
 a) GO[-6:5:2]
 b) GO[1:-2:2]
 c) GO[2:-3:2]
 d) GO[1:6:3]
 e) GO[3:4:2]

2. Considere OH=[10, 15, 21, 22, 23, 28]
Efetue todos os fatiamentos.
Ao final, some todos os elementos.
 a) OH[-6:4]
 b) OH[1:4:2]
 c) OH[-4:6:3]
 d) OH[3:5]
 e) OH[2:-3:2]

3. Considere AN=[1, 5, 9, 14, 18, 24, 27, 29]
Efetue todos os fatiamentos.
Ao final, some todos os elementos.
 a) AN[-6:8:3]
 b) AN[:6:2]
 c) AN[1:-2:2]
 d) AN[3:-3:2]
 e) AN[-8:6:2]

4. Considere TX=[1, 2, 4, 14, 15, 23, 25, 26]
Efetue todos os fatiamentos.
Ao final, some todos os elementos.
 a) TX[:8:2]
 b) TX[:7:2]
 c) TX[2:-2:2]
 d) TX[1:7]
 e) TX[-6:-1:2]

5. Considere
   CF=[3, 6, 8, 10, 15, 19, 23, 25, 27]
Efetue todos os fatiamentos.
Ao final, some todos os elementos.
 a) CF[:9]
 b) CF[:7]
 c) CF[1:8:2]
 d) CF[2:-2:2]
 e) CF[3:-2]

Respostas deste exemplo: 117 220 146 236 393
```

☞ Para você fazer

```
1. Considere XM=[6, 8, 17, 21, 22, 24]
Efetue todos os fatiamentos.
Ao final, some todos os elementos.
 a) XM[3:-1:2]
 b) XM[:5:2]
 c) XM[-4:-2:2]
 d) XM[:5:3]
 e) XM[3:5:2]

2. Considere ZK=[3, 4, 6, 11, 12, 16, 18, 22]
Efetue todos os fatiamentos.
Ao final, some todos os elementos.
 a) ZK[-6:6]
 b) ZK[:6]
 c) ZK[:6:-2]
 d) ZK[3:8:2]
 e) ZK[-6:-2]

3. Considere IY=[2, 4, 5, 13, 21, 22]
Efetue todos os fatiamentos.
Ao final, some todos os elementos.
 a) IY[2:-1]
 b) IY[:4:-2]
 c) IY[-4:5]
 d) IY[:4:2]
 e) IY[:6:2]

4. Considere XV=[7, 19, 20, 24, 27, 28]
Efetue todos os fatiamentos.
Ao final, some todos os elementos.
 a) XV[-4:-1:2]
 b) XV[2:6:2]
 c) XV[1:5:2]
 d) XV[:4:-2]
 e) XV[2:4]

5. Considere HR=[7, 10, 13, 15, 16, 18, 21, 22]
Efetue todos os fatiamentos.
Ao final, some todos os elementos.
 a) HR[:6]
 b) HR[:8]
 c) HR[:7:3]
 d) HR[2:6:2]
 e) HR[:6:3]
```

Responda aqui:

1	2	3	4	5