Listas

Prof. Alberto Costa Neto

Programação em Python

Uma Lista é um tipo de Coleção

- Uma coleção permite colocar vários valores em um única "variável"
- Coleções são práticas porque permitem carregar muitos valores empacotados de forma conveniente pelo programa

```
amigos = [ 'Jose', 'Maria', 'Pedro' ]
bagagem = [ 'meia', 'camisa', 'perfume' ]
```

O que não é uma “Coleção”?

A maioria de nossas variáveis tem um valor apenas – quando um novo valor é colocado nela, o antigo é sobrescrito

```
$ python
>>> x = 2
>>> x = 4
>>> print(x)
4
```

Constantes List

- Listas são circundadas por colchetes e seus elementos são separados entre si por vírgula
- Um elemento de uma lista pode ser qualquer objeto de Python – até mesmo outra lista
- Uma lista pode estar vazia

```
>>> print([1, 24, 76])
[1, 24, 76]
>>> print(['red', 'yellow', 'blue'])
['red', 'yellow', 'blue']
>>> print(['red', 24, 98.6])
['red', 24, 98.6]
>>> print([ 1, [5, 6], 7])
[1, [5, 6], 7]
>>> print([])
[]
```

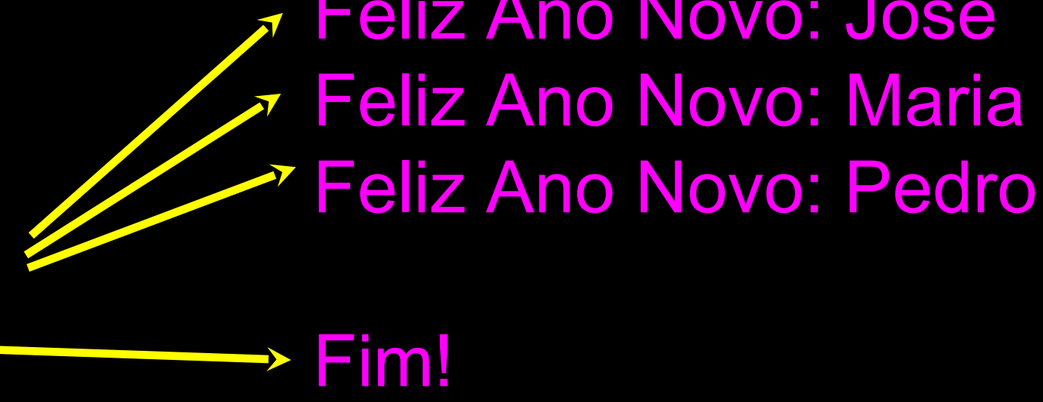
Um Laço Definido Simples

```
for i in [5, 4, 3, 2, 1] :
    print(i)
print('Fim!')
```

```
5
4
3
2
1
Fim!
```

Um Laço Definido com Strings

```
amigos = ['Jose', 'Maria', 'Pedro']
for amigo in amigos :
    print('Feliz Ano Novo:', amigo)
print('Fim!')
```



Detalhes internos das Listas

Da mesma forma que as strings, nós podemos acessar qualquer elemento na lista usando seu índice entre colchetes

Jose	Maria	Pedro
0	1	2

```
>>> amigos = ['Jose', 'Maria', 'Pedro']
>>> print(amigos[1])
Maria
```

Listas são Mutáveis

- Strings são “imutáveis” - não podemos mudar o conteúdo de uma string, mas podemos criar uma nova string com as mudanças desejadas
- Listas são “mutáveis’ - podemos modificar qualquer elemento de uma lista. Para isso basta utilizar seu índice para indicar a posição e atribuir um novo valor

```
>>> fruta = 'Banana'
>>> fruta[0] = 'b'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not
support item assignment
>>> f = fruta.lower()
>>> print(f)
banana
>>> idades = [2, 14, 26, 41, 63]
>>> print(idades[2], idades)
26 [2, 14, 26, 41, 63]
>>> idades[2] = idades[2] + 2
>>> print(idades)
[2, 14, 28, 41, 63]
```

Qual é o Comprimento da Lista?

- A função len() pode receber uma lista como parâmetro e retorna o número de *elementos* na lista.
- Na verdade len() informa o número de elementos de qualquer conjunto ou sequência (já vimos que funciona para string...)

```
>>> msg = 'Ola Bob'
>>> print(len(msg))
7
>>> x = [1, 2, 'jose', 99]
>>> print(len(x))
4
>>>
```

Usando a função range

- A função range retorna uma lista de números que variam de zero até o antecessor do parâmetro
- Assim fica fácil construir um laço for e dispor de índices

```
>>> print(range(4))
[0, 1, 2, 3]
>>> amigos = ['Jose','Maria','Pedro']
>>> print(len(amigos))
3
>>> print(range(len(amigos)))
[0, 1, 2]
>>>
```

Laços para todos os gostos...

```
amigos = ['Jose', 'Maria', 'Pedro']


for amigo in amigos :
  print('Feliz Ano Novo:', amigo)


for i in range(len(amigos)) :
  print(i+1, '- Feliz Ano Novo:', amigos[i])
```

```
Feliz Ano Novo: Jose
Feliz Ano Novo: Maria
Feliz Ano Novo: Pedro

1 - Feliz Ano Novo: Jose
2 - Feliz Ano Novo: Maria
3 - Feliz Ano Novo: Pedro
```

Concatenando listas usando +

- Podemos criar uma nova lista a partir da junção dos elementos de 2 listas existentes

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> c = a + b
>>> print(c)
[1, 2, 3, 4, 5, 6]
>>> print(a)
[1, 2, 3]
```

Há um certo valor na Lista?

- Python provê dois operadores (in e not in) que permitem checar, respectivamente, se um item está ou não em uma lista
- Eles são operadores lógicos (retornam True ou False e não modificam a lista)

```
>>> nums = [1, 9, 21, 10, 16]
>>> 9 in nums
True
>>> 15 in nums
False
>>> 20 not in nums
True
>>>
```

Criando uma lista do início

- Podemos criar uma lista vazia e então adicionar os elementos usando o método append
- A lista permanece na ordem de inserção. Novos elementos são adicionados no final da lista

```
>>> lista = list()
>>> lista.append('livro')
>>> lista.append(99)
>>> print(lista)
['livro', 99]
>>> lista.append('biscoito')
>>> print(lista)
['livro', 99, 'biscoito']
```

Listas podem ser particionadas (sliced) usando :

```
>>> t = [9, 41, 12, 3, 74, 15]
>>> t[1:3]
[41,12]
>>> t[:4]
[9, 41, 12, 3]
>>> t[3:]
[3, 74, 15]
>>> t[:]
[9, 41, 12, 3, 74, 15]
```

Lembrete: *Da mesma forma que em strings*, o segundo número é “até, mas não o incluindo”