

Algoritmo e Programação Matemática

LAÇOS E LISTAS

Renato Dourado Maia

Instituto de Ciências Agrárias

Universidade Federal de Minas Gerais



A Função `list`

- A função `list` pode ser utilizada para realizar a **conversão** de uma *string* numa lista.
 - Isso é interessante, pois um elemento de uma lista pode ser **modificado individualmente**, enquanto os de uma *string*, não.
 - A transformação **inversa** pode ser realizada pelo método `join` (estudaremos métodos depois).

```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> Lista = list('Acordem!')
>>> Lista
['A', 'c', 'o', 'r', 'd', 'e', 'm', '!']
>>> Lista[1] = 'Aaaaaaa'
>>> Lista
['A', 'Aaaaaaa', 'o', 'r', 'd', 'e', 'm', '!']
>>> ''.join(Lista)
'AAaaaaaaordem!'
```



Variáveis do Tipo `list`

- Uma variável do tipo `list` na verdade contém uma **referência** para um valor do tipo `list`.
 - **Atribuir** uma variável a outra cria uma **nova referência**, mas **não uma nova lista!**
 - Para se criar um **novo valor**, pode-se utilizar uma **expressão que retorne o valor desejado**.
 - Para saber se duas variáveis se referem a um **mesmo valor**, pode-se utilizar o operador `is`.



Variáveis do Tipo `list`

```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> a = b = [1, 2, 3]
>>> c = a
>>> d = c[:]
>>> a is b
True
>>> c is b
True
>>> d is c
False
>>> a[1] = 5
>>> b
[1, 5, 3]
>>> d
[1, 2, 3]
```



A Classe `list`

- Uma lista é um **objeto** de uma **classe** chamada `list`:
 - Não estudamos **programação orientada a objetos**, mas alguns aspectos precisam ser **adiantados**.
- Listas possuem **métodos** que poder ser a elas aplicados.
- Um método é **semelhante** a uma função e é invocado da seguinte maneira: `objeto.método(argumentos)`.
 - `Lista.reverse()` inverte a ordem dos elementos de Lista.
- Para saber **todos os métodos** da classe `list`:
 - `help(list)`.



Alguns Métodos da Classe `list`

- `append(elemento)`:
 - **Acrésceta** elemento no **final da lista**.
 - Note que a operação **altera a lista** e não apenas retorna uma lista modificada!

```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> Lista = [1, 2]
>>> Lista.append(3)
>>> Lista
[1, 2, 3]
>>> Lista.append([4, 5])
>>> Lista
[1, 2, 3, [4, 5]]
Ln: 10 Col: 4
```



Alguns Métodos da Classe `list`

- `count(elemento)`:
 - **Retorna quantas vezes** o elemento aparece na lista.
- `extend(Lista2)`:
 - **Acrescenta** os elementos de `Listas2` no **final da lista**.
 - A lista é **alterada!**

```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> [1, 2, 3, 4, 1, 2, 3, 4].count(1)
2
>>> Lista = [1, 2]
>>> Lista.extend([3, 4])
>>> Lista
[1, 2, 3, 4]
```



Alguns Métodos da Classe `list`

- `index` (elemento):
 - **Retorna** o índice da **primeira ocorrência** de elemento na lista.
 - Se a lista não contiver o elemento, ocorre um **erro**.

```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> Lista = [9, 8, 33, 12]
>>> Lista.index(33)
2
>>> Lista.index(7)

Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    Lista.index(7)
ValueError: 7 is not in list
>>> 7 in Lista
False
Ln: 14 Col: 4
```



Alguns Métodos da Classe `list`

- `insert` (índice, elemento):
 - **Inserer** elemento na posição índice da lista.
 - Assim como o método `extend`, **altera a lista!**
 - Atribuições a fatias, como vimos na aula passada, realizam a mesma operação, mas são **menos legíveis**.

```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> Lista = [0, 1, 2, 3]
>>> Lista.insert(1, 'dois')
>>> Lista
[0, 'dois', 1, 2, 3]
>>> Lista = [1, 2, 3]
>>> Lista[1:1] = ['dois']
>>> Lista
[1, 'dois', 2, 3]
>>> Retorno = Lista.insert(3, '3')
>>> type(Retorno)
<type 'NoneType'>
```



Alguns Métodos da Classe `list`

- `pop` (índice):
 - **Remove** o elemento da posição índice.
 - Caso o índice **não seja informado**, assume-se o **último**.

```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> Lista = ['O', 'pessoal', 'desanimou', '?']
>>> Lista.pop()
'?'
>>> Lista
['O', 'pessoal', 'desanimou']
>>> Lista.pop(1)
'pessoal'
>>> Lista
['O', 'desanimou']
```



Alguns Métodos da Classe list

- remove (elemento):
 - Remove o **primeiro elemento igual a elemento**.
 - Caso elemento **não exista** na lista, um **erro** é gerado.

```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> Lista = ['Ei', 'Oi', 'Hi']
>>> Lista.remove('Oi')
>>> Lista
['Ei', 'Hi']
>>> Lista.remove('Inexistente')

Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    Lista.remove('Inexistente')
ValueError: list.remove(x): x not in list
Ln: 13 Col: 4
```



Alguns Métodos da Classe `list`

- `reverse()`:
 - **Inverte** a ordem dos elementos da lista.
 - A lista é **alterada!**

```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> Lista = [1, 2, 3]
>>> Lista2 = Lista[::-1]
>>> Lista
[1, 2, 3]
>>> Lista2
[3, 2, 1]
>>> Lista.reverse()
>>> Lista
[3, 2, 1]
```



Alguns Métodos da Classe `list`

- `sort(cmp=None, key=None, reverse=False)`:
 - **Ordena** a lista.
 - Os argumentos são **opcionais** e, por **default**, a lista é ordenada em **ordem crescente**.

```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> Lista = [9, 8, 7, 1, 4, 2]
>>> Lista.sort()
>>> Lista
[1, 2, 4, 7, 8, 9]
>>> Lista = [9, 8, 7, 1, 4, 2]
>>> Lista.sort(reverse = True)
>>> Lista
[9, 8, 7, 4, 2, 1]
>>> Lista = [9, 8, 7, 1, 4, 2]
>>> Lista.sort(None, None, True)
>>> Lista
[9, 8, 7, 4, 2, 1]
```



Alguns Métodos da Classe `list`

- `sort(cmp=None, key=None, reverse=False)`:
 - O argumento `cmp` especifica uma **função de comparação**.
 - ✓ Essa função é chamada pelo `sort` para definir se um elemento é **anterior** ou **posterior** a outro.
 - ✓ A forma dessa função é `comp(Elem1, Elem2)` e ela deve retornar um inteiro **negativo** caso `Elem1` seja anterior a `Elem2`, **positivo** caso `Elem2` seja anterior a `Elem1` e **zero** se tanto faz.

```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> def Compara(Elem1, Elem2):
        return Elem1%10 - Elem2%10

>>> Compara(100, 22)
-2
>>> Lista = [100, 22, 303, 104]
>>> Lista.sort(Compara)
>>> Lista
[100, 22, 303, 104]
```



Alguns Métodos da Classe `list`

- `sort(cmp=None, key=None, reverse=False)`:
 - O argumento `key` **especifica uma função** a ser aplicada em **cada elemento**, de forma que, se for passada uma função f , em vez de os elementos serem ordenados pelos valores v , serão ordenados pelos valores $f(v)$.

```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> Lista = ['abc', 'de', 'fghi']
>>> Lista.sort(key = len)
>>> Lista
['de', 'abc', 'fghi']
Ln: 7 Col: 4
```



Matrizes

- Listas podem ser utilizadas para **armazenar matrizes**.
- Como?
 - Lista de listas!

```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> Matriz = [[1, 2, 3], [4, 5, 6], [7, 8, 9],]
>>> Matriz[0][0]
1
>>> Matriz[2][2]
9
>>> Matriz[2][1]
8
>>>
```

Vejam o programa [MatrizNumpy.py...](#)



Tuplas

- Tuplas são listas que não podem ser modificadas.
- Uma tupla é **inicializada** separando-se os elementos com uma **vírgula** e utilizando-se **parênteses** (ou não):

```
T1 = (-91, 'uma string', 7.2, 0)
```

```
T1 = -91, 'uma string', 7.2, 0
```



Tuplas

```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> T = (1, 2, 3, 4, 5)
>>> T.append(6)

Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    T.append(6)
AttributeError: 'tuple' object has no attribute 'append'
>>> del T[0]

Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    del T[0]
TypeError: 'tuple' object doesn't support item deletion
>>> T[0] = 1

Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    T[0] = 1
TypeError: 'tuple' object does not support item assignment
Ln: 22 Col: 4
```

Estudaremos mais sobre tuplas depois...



Exercício

1. Escreva um programa para computar o produto entre duas matrizes $m1$ e $m2$ (assuma que as dimensões são compatíveis e que as variáveis $m1$ e $m2$ serão inicializadas no início do programa).



That's All Folks!



SEGUNDA LISTA DE EXERCÍCIOS!

