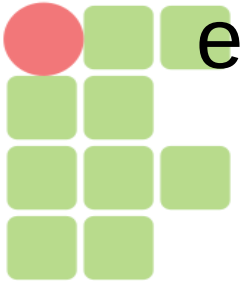
Vetores

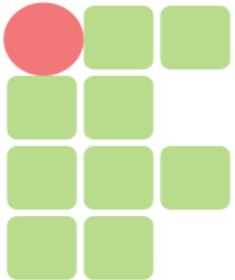
- Vetores são uma estrutura de dados composta disponível em diversas linguagens de programação para **agrupamento de dados**
- Até agora, em um programa que necessitava de **n** entradas, utilizávamos várias variáveis para representar tais entradas
- Com uma estrutura que permite o agrupamento, é possível armazenar todos esses valores em uma só variável



Vetores / Listas

- Em Python, utilizaremos inicialmente as listas como o tipo de dados para agrupamento de valores
  - Uma lista de números inteiros

```
>>> l = [10, 20, 30, 40, 50]
>>> len(l)
5
>>> for elem in l:
...     print elem,
10 20 30 40 50
```



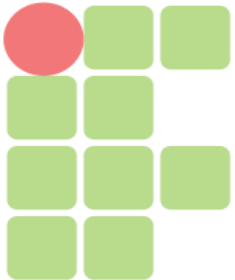
Listas

- Listas podem conter os mais variados tipos de dados
  - Uma lista de *Strings*

```
>>> l = ['jose', 'maria', 'mario']
>>> for elem in l:
...     print elem,
jose maria mario
```

  - Uma lista mista

```
>>> l = ['jose', 7.2, 'A', 12]
>>> for elem in l:
...     print elem,
jose 7.2 A 12
```



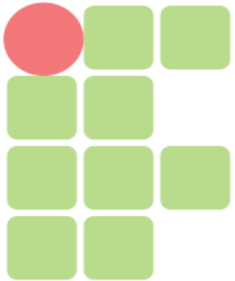
Listas

- Outra lista mista

```
>>> l = ['jose', [7, 6, 8, 7], 7, 'A', 12]
>>> for elem in l:
...     print elem
jose
[7, 6, 8, 7]
7
A
12
```

- Lista vazia

```
>>> l = []
>>> for x in l:
...     print x,
>>>
```

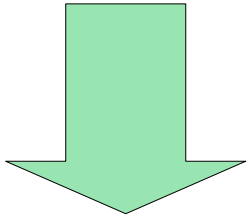


Listas

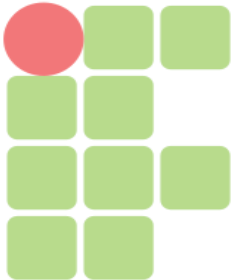
- Uma lista de números inteiros em sequência

```
>>> l = range(0, 10)
>>> for i in l:
...     print i,
0 1 2 3 4 5 6 7 8 9
```

```
for i in range(0, 10):
    print i,
```



```
i = 0
while i < 10:
    print i,
    i = i + 1
```



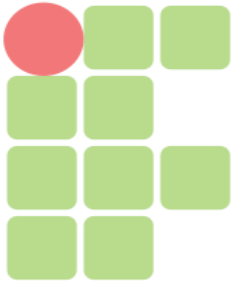
Listas

- A função range() pode receber um argumento adicional: o **passo**

```
>>> range(0, 10, 2)
[0, 2, 4, 6, 8]
```

- Também pode-se usar com apenas um argumento

```
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```



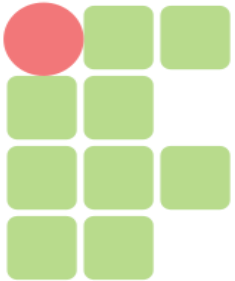
Listas

- Acesso a elementos da lista
  - Operador [ ] → colchetes

```
>>> lis = ['jose, 'maria', 'mario']
>>> print lis[1]
maria
```

  - Listas são modificáveis (diferentemente da *String*)

```
>>> lis = ['jose, 'maria', 'mario']
>>> lis[1] = 'paulo'
>>> for nome in lis:
...     print nome,
jose paulo mario
```

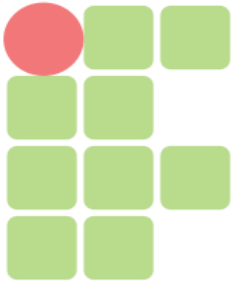


Listas

- Somente posições existentes podem ser acessadas/modificadas

```
>>> lis = ['jose, 'maria', 'mario']
>>> lis[4] = 'paulo'
IndexError: list assignment index out of range
```

- Como adicionar um novo elemento em uma lista já existente?
  - Função append()
  - Operador concatenação **+**
    - Concatena elementos do mesmo tipo
      - Lista com lista

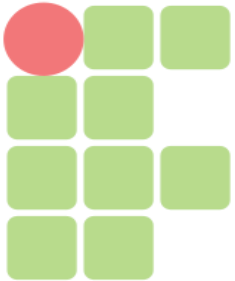


Listas

- Pertinência em uma lista
  - Operador **in**

```
>>> lis = ['jose, 'maria', 'mario']
>>> 'maria' in lis
True
>>> 'marcos' in lis
False
```

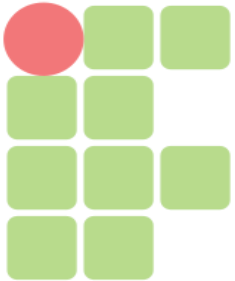
```
lis = ['jose, 'maria', 'mario']
nome = raw_input('digite o nome do aluno: ')
if nome in lis:
   print 'O aluno está matriculado.'
else:
      print 'aluno desconhecido.'
```



Listas

- Fatiamento de listas
  - Operador :

```
>>> lista = ['a', 'b', 'c', 'd', 'e', 'f']
>>> lista[1:3]
['b', 'c']
>>> lista[:4]
['a', 'b', 'c', 'd']
>>> lista[3:]
['d', 'e', 'f']
>>> lista[:]
['a', 'b', 'c', 'd', 'e', 'f']
```



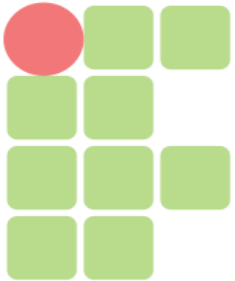
Listas

- Removendo um elemento

```
>>> a = ['um', 'dois', 'tres']
>>> del a[1]
>>> a
['um', 'tres']
```

- Removendo vários elementos

```
>>> lista = ['a', 'b', 'c', 'd', 'e', 'f']
>>> del lista[1:5]
>>> print lista
['a', 'f']
```



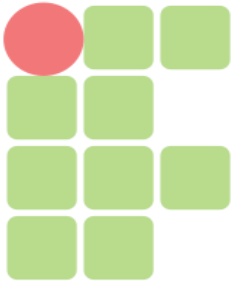
Listas

- Clonando listas
  - Necessário quando precisamos alterar uma lista e manter os valores antigos

```
>>> a = [1, 2, 3]
>>> b = a[:]
>>> print b
[1, 2, 3]
```

  - Por que não **`b = a`** ?

Porque isso apenas cria um apelido!



Listas

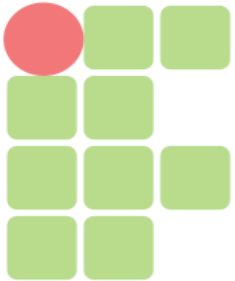
- Exemplos:

```
>>> a = [1, 2, 3]
>>> b = a
>>> a.append(10)
>>> print a
[1, 2, 3, 10]
>>> print b
[1, 2, 3, 10]
```

```
>>> a = [1, 2, 3]
>>> b = a[:]
>>> a.append(10)
>>> print a
[1, 2, 3, 10]
>>> print b
[1, 2, 3]
```

```
>>> a = [1, 2, 3]
>>> b = a
>>> b.append(10)
>>> print a
[1, 2, 3, 10]
>>> print b
[1, 2, 3, 10]
```

```
>>> a = [1, 2, 3]
>>> b = a[:]
>>> b.append(10)
>>> print a
[1, 2, 3]
>>> print b
[1, 2, 3, 10]
```



Listas

- Listas aninhadas

```
lista = [[10, 20], [100, 200]]
print lista[1][0]
```

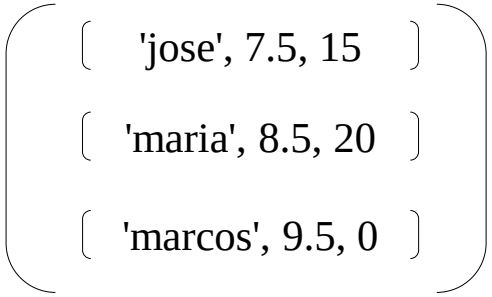
$$\begin{pmatrix} 10 & 20 \\ 100 & 200 \end{pmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```
>>> matriz = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Listas

- Listas aninhadas



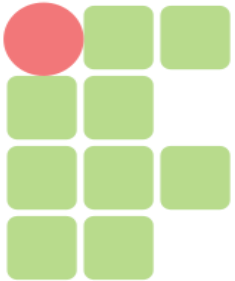

```
L = [ ['jose', 7.5, 15], ['maria', 8.5, 20], ['marcos', 9.5, 0]]
```

Listas

- A função split() das *strings*...
  - Quebra uma string em strings menores e retorna estas como uma lista de strings separadas

```
s = 'Meus parabens para voce!'
l = s.split('pa')
print l
```

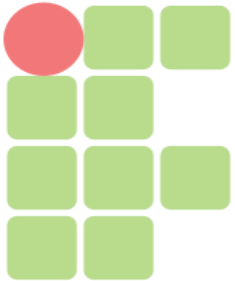
```
['Meus ', 'rabens ', 'ra voce!']
```



Listas

- Função split()

```
data = raw_input('Digite uma data separada por / :')
data_l = data.split('/')
dia = int(data_l[0])
mes = int(data_l[1])
ano = int(data_l[2])
```

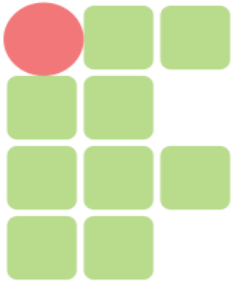


Listas

- Função string.join()
  - Une os elementos de uma lista em uma String

```
import string
L = ['08', '01', '1983']
data = string.join(L, '/')
print data
```

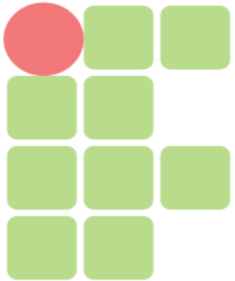
```
08/01/1983
```



Exercícios

- Faça um programa que leia 5 números inteiros e, após a leitura de todos, imprima estes na ordem de leitura.

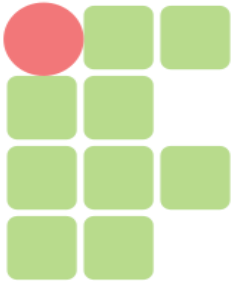
```
l = []
i = 0
while i < 5:
    n = input('digite um numero: ')
    l.append(n)
    i = i + 1

for e in l:
    print e
```



Exercícios

- Um programa que leia 10 números reais e imprima estes na ordem inversa de leitura

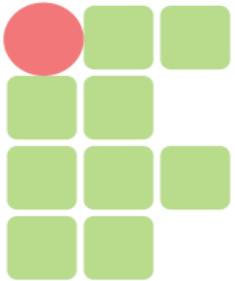
```
l = []
i = 0
while i < 10:
    n = input('digite um numero: ')
    l.append(n)
    i = i + 1

for e in reversed(l):
    print e
```



Exercícios

- Faça um programa que leia 4 notas, mostre as notas e a média na tela.

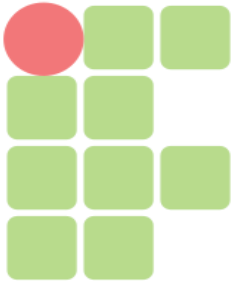
```
l = []
i = 0
while i < 4:
    n = input('digite um numero: ')
    l.append(n)
    i = i + 1

soma = 0
for e in l:
    soma = soma + e
media = float(soma)/len(l)
print media
```



Exercícios

- Faça um programa que peça as quatro notas de 10 alunos, calcule e armazene numa lista a média de cada aluno e imprima o número de alunos com média maior ou igual a 7.0.

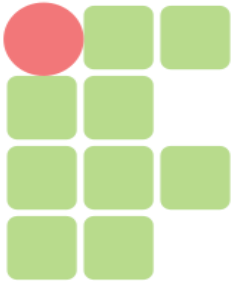
```
medias = []
for i in range(0, 3):
    media = 0
    for j in range(0, 4):
        n = input('digite uma nota do aluno' + str(i) + ': ')
        media = media + n
    media = media / 4
    medias.append(media)
cont = 0
for media in medias:
    if media >= 7:
        cont = cont + 1
print cont
```



Exercícios

- Faça um programa que leia uma lista de 5 números inteiros, mostre a soma, a multiplicação e os números.

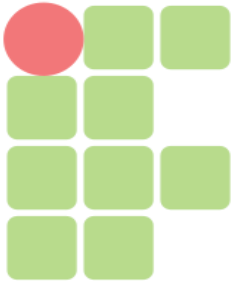
```
lis = []
for i in range(0, 5):
    n = input('digite um numero: ')
    lis.append(n)

soma = 0
mult = 1
for e in lis:
    soma += e
    mult *= e
    print e,
print '\nSoma: ', soma
print 'Multiplicacao: ', mult
```



Exercícios

- Faça um programa que leia dois vetores com 10 elementos cada. Gere um terceiro vetor de 20 elementos, cujos valores deverão ser compostos pelos elementos intercalados dos dois outros vetores.

```
lis1 = []
lis2 = []
for i in range(0, 10):
    n = input('digite um numero: ')
    lis1.append(n)
for i in range(0, 10):
    n = input('digite um numero: ')
    lis2.append(n)

lis3 = []
for i in range(0, 10):
    lis3.append(lis1[i])
    lis3.append(lis2[i])
for e in lis3:
    print e,
```



Exercícios

- Faça um programa que armazene em uma lista de listas as seguintes informações sobre os sócios de um clube: nome, endereço, telefone e situação (OK, Débito, Inativo). Tal clube possui um número indefinido de sócios.

```
socios = [['Jose da Silva', 'Rua das Flores, 18', '5555-5555', 'O'],
          ['Fulano', 'Centro', '', 'D']]

nome = raw_input('Digite o nome: ')
end = raw_input('Digite o endereco: ')
fone = raw_input('Digite o telefone: ')
sit = raw_input('Digite a situacao: ')
socios.append([nome, end, fone, sit])
for socio in socios:
    print 'nome:     ', socio[0]
    print 'endereco: ', socio[1]
    print 'telefone: ', socio[2]
    print 'situacao: ', socio[3]
    print '--------------------------------'
```