Funções para **for**

- enumerate

Exemplo

```
>>> L1 = ['Dumbledore','Hagrid','Dobby','sabao']
>>> L2 = [1,2,3,4]
>>> for k,w in enumerate(L1):
...:     print(k,w)

0 Dumbledore
1 Hagrid
2 Dobby
3 sabao
```

Funções para **for**

- zip

Exemplo

```
>>> L1 = ['Dumbledore','Hagrid','Dobby','sabao']
>>> L2 = [1,2,3,4]
>>> for k in zip(L1,L2):
...:     print(k)

('Dumbledore', 1)
('Hagrid', 2)
('Dobby', 3)
('sabao', 4)
```

Matrizes em Python

Exemplo

```
>>> M = [[1,2,3],[4,5,6],[7,8,9]]
>>> print(M)

[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
>>> M[0]
[1,2,3]
>>> M[2][1]
8
```

List Comprehension

Exemplo

```
>>> L = [[] for k in range(5)]
>>> print(L)

[[], [], [], [], []]
>>> L = [k for k in range(5) if k%2==0]
>>> print(L)

[0, 2, 4]
```

Funções anônimas: **lambda**

Exemplo

```
>>> f1 = lambda x: x**2
>>> f1(34)
1156
>>> f2 = lambda x,y: x*y
>>> f2(5,8)
40
>>> f3 = lambda L: [k for k in L if k%2!=0]
>>> f3([1,2,3,4,5,6,7,8,9])
[1, 3, 5, 7, 9]
```

Funções de ordem maior:

- **map**
- **filter**

Exemplos

```
>>> L = [1,2,3,4,5,6,7,8,9,10]
>>> f1 = lambda s: str(s)
>>> list(map(f1,L))
[’1’, ’2’, ’3’, ’4’, ’5’, ’6’, ’7’, ’8’, ’9’, ’10’]
>>> f2 = lambda x: x%2==0
>>> list(filter(f2,L))
[2, 4, 6, 8, 10]
```

Importando bibliotecas

```
# biblioteca.funcao() para acessar suas funcoes
import biblioteca

# bib.funcao() para acessar suas funcoes
import biblioteca as bib

# Importa a funcao f1 diretamente
from biblioteca import f1

# Importa todas as funcoes de biblioteca diretamente
from biblioteca import *
```

Biblioteca **math**

Exemplo

```
>>> import math
>>> math. #tab para ver as funcoes disponiveis
>>> math.sin(math.pi)
1.2246467991473532e-16
>>> math.factorial(20)
2432902008176640000
>>> math.sqrt(42)
6.48074069840786
>>> math.log(27,3)
4.0
```

Biblioteca numérica para manipulação de vetores multidimencionais: **numpy**

array()

```
>>> import numpy as np
>>> a = np.array([2,5,8,4,9,3])
>>> a
array([2, 5, 8, 4, 9, 3])
>>> L = [2,5,7,9,4,0]
>>> a = np.array(L)
>>> a
array([2, 5, 7, 9, 4, 0])
```

zeros() e ones()

```
>>> np.zeros(5)
array([ 0.,  0.,  0.,  0.,  0.])
>>> np.zeros((3,3))
array([[ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.]])
>>> np.ones(5)
array([ 1.,  1.,  1.,  1.,  1.])
>>> np.ones((3,3))
array([[ 1.,  1.,  1.],
       [ 1.,  1.,  1.],
       [ 1.,  1.,  1.]])
```

identity()

```
>>> np.identity(5)
array([[ 1.,  0.,  0.,  0.,  0.],
       [ 0.,  1.,  0.,  0.,  0.],
       [ 0.,  0.,  1.,  0.,  0.],
       [ 0.,  0.,  0.,  1.,  0.],
       [ 0.,  0.,  0.,  0.,  1.]])
```

random

```
>>> np.random.random(3)
array([ 0.20256883,  0.20576955,  0.73657194])
>>> np.random.random((3,3))*(100-50)+50
array([[ 99.06180386,  76.3900278 ,  78.57222814],
       [ 89.99773629,  91.02358742,  91.79590092],
       [ 85.49658424,  60.47626305,  88.01781855]])
>>> np.random.randint(100)
7
>>> np.random.randint(100,size=(3,3))
array([[10, 27, 87],
       [49, 71, 71],
       [93, 84,  2]])
```

Funções

```
>>> a1 = np.ones((5,6))
>>> np.size(a1)   # ou a1.size
30
>>> np.ndim(a1)   # ou a1.ndim
2
>>> np.shape(a1)  # ou a1.shape
(5, 6)
>>> np.arange(0,1,0.2)   # Início, fim, passo
array([ 0. ,  0.2,  0.4,  0.6,  0.8])
>>> np.arange(0,1+0.2,0.2)
array([ 0. ,  0.2,  0.4,  0.6,  0.8,  1.])
>>> np.linspace(0,1,5)   # Início, fim, quantos
array([ 0.  ,  0.25,  0.5 ,  0.75,  1.  ])
>>> np.linspace(0,1,5,endpoint=False)
array([ 0. ,  0.2,  0.4,  0.6,  0.8])
```

Acesso

```
>>> a1 = np.random.randint(100,size=(3,3))
>>> a1
array([[17,  9, 27],
       [46, 21, 54],
       [10, 62, 28]])
>>> a1[1,2]
54
>>> a1[:,1]
array([ 9, 21, 62])
>>> a1[2,:]
array([10, 62, 28])
```

Acesso

```
>>> a2 = np.random.randint(100,size=(5,6))
>>> a2
array([[98, 73, 40, 44, 34, 24],
       [78, 25, 34, 28, 54, 66],
       [ 4,  5, 13, 54, 37, 63],
       [24, 71, 83, 87, 22,  0],
       [40, 22, 96, 47, 96, 63]])
>>> a2[1:3,:]
array([[78, 25, 34, 28, 54, 66],
       [ 4,  5, 13, 54, 37, 63]])
>>> a2[::2,::3]
array([[98, 44],
       [ 4, 54],
       [40, 47]])
```

Operações

```
>>> a = np.array([1,2,3,4,5])
>>> a + 3
array([4, 5, 6, 7, 8])
>>> a - 3
array([-2, -1,  0,  1,  2])
>>> a * 5
array([ 5, 10, 15, 20, 25])
>>> a / 2
array([ 0.5,  1. ,  1.5,  2. ,  2.5])
>>> a // 2
array([0, 1, 1, 2, 2])
>>> a ** 7
array([    1,   128,  2187, 16384, 78125])
```

Operações

```
>>> a1, a2 = np.array([1,2,3,]), np.array([4,5,6])
>>> a1 + a2
array([5, 7, 9])
>>> a1 - a2
array([-3, -3, -3])
>>> a1 * a2
array([ 4, 10, 18])
>>> a1 / a2
array([ 0.25,  0.4 ,  0.5 ])
>>> a1 // a2
array([0, 0, 0])
>>> a1 ** a2
array([  1,  32, 729])
```

Produtos entre vetores

```
>>> a1, a2 = np.array([1,2,3,]), np.array([4,5,6])
>>> np.dot(a1,a2)     # Escalar
32
>>> np.cross(a1,a2)   # Vetorial
array([-3,  6, -3])
>>> np.outer(a1,a2)   # Tensorial
array([[ 4,  5,  6],
       [ 8, 10, 12],
       [12, 15, 18]])
```

Vectorization - aplicando funções a vetores

```
>>> a = np.array([[1,2,3],[4,5,6],[7,8,9]]) # matriz
>>> np.max(a)
9
>>> np.min(a)
1
>>> np.mean(a)
5.0
>>> np.std(a)      # Desvio padrão
2.581988897471611
>>> np.var(a)      # Variância
6.666666666666667
>>> np.trace(a)    # Traço
15
```

Vectorization - aplicando funções a vetores

```
>>> f = lambda x: x**3
>>> a = np.array([1,2,3,4,5])
>>> f(a)
array([  1,   8,  27,  64, 125])
```

matrix()

```
>>> M = np.matrix([[1,2,3],[4,5,6],[7,8,9]])
>>> M
>>> matrix([[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]])
>>> a = np.array([[1,2,3],[4,5,6],[7,8,9]])
>>> M = np.matrix(a)
>>> M
>>> matrix([[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]])
```

Métodos úteis

```
>>> M = np.matrix([[1,2,3],[1,0,2],[0,2,3]])
>>> M.I     # inversa
matrix([[ 1.  ,  0.  , -1.  ],
        [ 0.75, -0.75, -0.25],
        [-0.5 ,  0.5 ,  0.5 ]])
>>> M.T     # transposta
matrix([[1, 1, 0],
        [2, 0, 2],
        [3, 2, 3]])
>>> M.A     # transforma matrix em array
array([[1, 2, 3],
       [1, 0, 2],
       [0, 2, 3]])
```

Operações matriz-vetor

```
>>> M1 = np.matrix([[1,2,3],[1,0,2],[0,2,3]])
>>> M2 = np.matrix([3,6,1])
>>> M1*M2.T
matrix([[18],
        [ 5],
        [15]])
>>> M2*M1
matrix([[ 9,  8, 24]])
>>> M1.I*M2.T     # resolvendo sistema linear
matrix([[ 2. ],
        [-2.5],
        [ 2. ]])
```

Biblioteca de álgebra linear: **linalg**

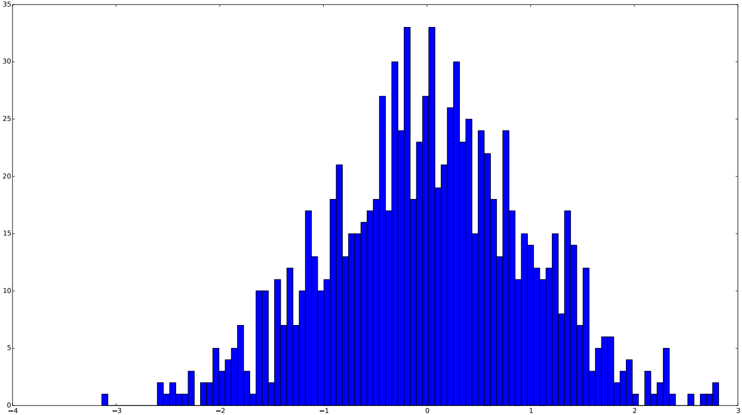
Exemplos

```
>>> M = np.matrix([[1,2,3],[1,0,2],[0,2,3]])
>>> b = np.array([1,2,3])
>>> np.linalg.norm(M)    # norma
5.656854249492380б
>>> np.linalg.det(M)     # determinante
-4.0
>>> np.linalg.eigvals(M)     # auto valores
array([-1.21018413,  0.73928861,  4.47089552])
>>> np.linalg.solve(M,b)    # resolve sistema linear
array([-2. , -1.5,  2. ]
```

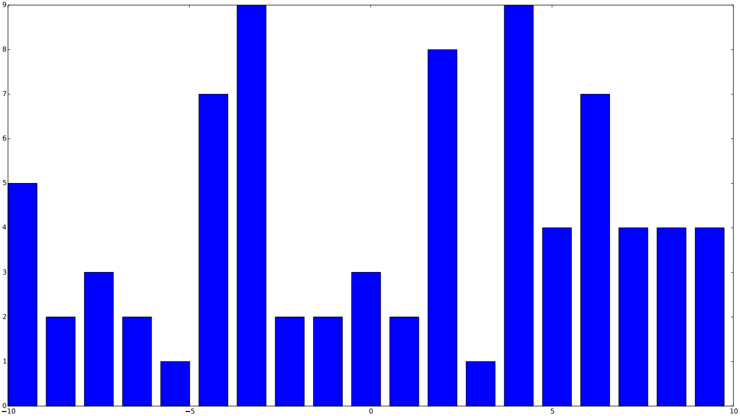
Biblioteca gráfica: **matplotlib.pyplot**

```python
# Importando bibliotecas
import numpy as np
import matplotlib.pyplot as plt
```

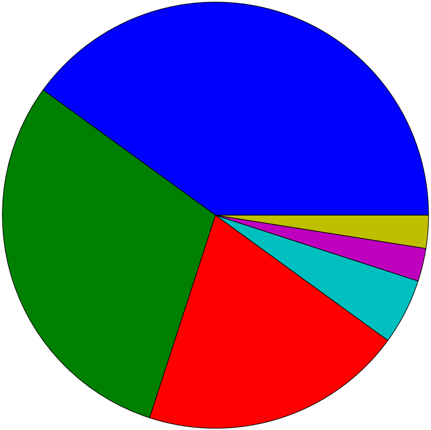
```
# Distribuicao aleatoria normal
y = np.random.randn(1000)
# Histograma, bins-> quantidade de colunas
plt.hist(y,bins=100)
plt.show()
```



```
x = np.linspace(-10,10,20)
y = np.random.randint(10,size=20)
plt.bar(x,y)
plt.show()
```



```
p = [0.4,0.3,0.2,0.05,0.025,0.025]
plt.pie(p)
```

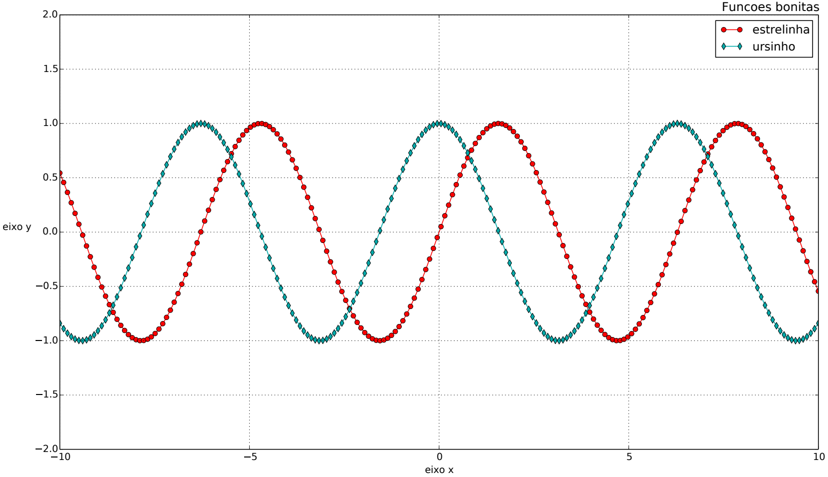


```
f_estrelinha = lambda x: np.sin(x)
f_ursinho = lambda x: np.cos(x)

x = np.linspace(-10,10,200)
y1 = f_estrelinha(x)
y2 = f_ursinho(x)

plt.plot(x,y1,’ro-’,label=’estrelinha’)
plt.plot(x,y2,’d-’,color=’#00AAAA’,label=’ursinho’)

plt.grid(linestyle=’:’,solid_capstyle=’butt’)
plt.axis([-10,10,-2,2])
plt.xlabel(’eixo x’,fontsize=12)
plt.ylabel(’eixo y’,rotation=0,fontsize=12,
           horizontalalignment=’right’)
plt.title(’Funcoes bonitas’,fontsize=15,loc=’right’)
plt.legend(loc=’best’)
plt.show()
```


Funcoes bonitas
estrelinha
ursinho
eixo y
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−10
−5
0
5
10
eixo x