PYTHON

ALGUMAS BIBLIOTECAS

INSTALANDO BIBLIOTECAS

```
$ [sudo] python setup.py install [--user]
```

```
$ [sudo] pip install my_library [--upgrade]
```

```
$ [sudo] apt-get search my_package
$ [sudo] apt-get install my_package
```

Entrando:

1. Via ipython: digite ipython no terminal
2. Diretamente: digite python no terminal

IMPORTANDO AS BIBLIOTECAS (MÓDULOS)

Importando o numpy:

- import numpy

Criando um alias para o módulo:

- import numpy as np

Podemos importar o que nos interessa:

- from numpy import pi
- print(pi)

OPERAÇÕES BÁSICAS

Operador	Símbolo	Exemplo	Saída
soma	+	5+0.5	5.5
subtração	-	5-0.5	4.5
produto	*	5*5	25
exponenciação	**	5**2	25
resto	%	5 % 2	1

OPERAÇÕES BÁSICAS

Operador Condicionais	Símbolo	Exemplo	Saída
maior	>	5 >6	False
maior igual	>=	5>=5.1	False
menor	<	5<6	True
menor igual	<=	5<=4	False
igualdade	==	5==5	True
e lógico	and	(5 and 6) <7	True
ou lógico	or	(5 or 7) <6	True

TIPOS DE VARIÁVEIS

Tipo	Símbolo	Exemplo	Saída
texto	str	str('olá')	'olá'
inteiro	int	int(5.1)	5
real	float	float(6)	6.0
complexo	complex	complex(2,3)	2+3j

Say “Hello World!”

```
>>> print “Hello World”
Hello World
```

Assign a variable

```
>>> x = 2
>>> print x
2
```

Say “Hello World!”

```
>>> print “Hello World”
Hello World
```

Assign a variable

```
>>> x = 2
>>> print x
2
```

Integers

```
>>> 5 + 4
9

>>> 5 - 4
1

>>> 5 * 4
20

>>> 5 / 4
1
```

Say “Hello World!”

```
>>> print “Hello World”
Hello World
```

Assign a variable

```
>>> x = 2
>>> print x
2
```

Integers

```
>>> 5 + 4
9

>>> 5 - 4
1

>>> 5 * 4
20

>>> 5 / 4
1
```

Float/Double

```
>>> 2.0 + 3.0
5.0

>>> 2.0 - 3
-1.0

>>> 2. * 3
6.0

>>> 2 / 3.
0.666666...
```

More on operators
(like +, -, /, *, **, >, <)

Say “Hello World!”

```
>>> print “Hello World”
Hello World
```

Assign a variable

```
>>> x = 2
>>> print x
2
```

Integers

```
>>> 5 + 4
9

>>> 5 - 4
1

>>> 5 * 4
20

>>> 5 / 4
1
```

Float/Double

```
>>> 2.0 + 3.0
5.0

>>> 2.0 - 3
-1.0

>>> 2. * 3
6.0

>>> 2 / 3.
0.666666...
```

More on operators
(like +, -, /, *, **, >, <)

Complex

```
>>> (2 + 3j) + (5 - 4j)
(7 - 1j)

>>> (2 + 3j) - (5 - 4j)
(-3 + 7j)

>>> (2 + 3j) * (5 - 4j)
(22 + 7j)

>>> (2 + 3j) / (5 - 4j)
(-0.049 + 0.561j)
```

LISTAS E VETORES

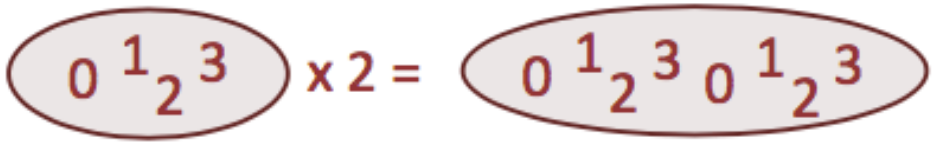
Lists vs NDArrays

```
>>> x = [1,2,3]
>>> print 2 * x
[1, 2, 3, 1, 2, 3]
```

```
>>> x = numpy.array([1,2,3])
>>> print 2 * x
[2, 4, 6]
>>> print 2.0 * x
[2.0, 4.0, 6.0]
```

For more of NumPy methods, check the NumPy Documentation

range(4)




```
>>> x = range(4)
>>> x.__class__
<type 'list'>
>>> x
[0, 1, 2, 3, 4]
>>> 2 * x
[0, 1, 2, 3, 0, 1, 2, 3]
```

LISTAS E VETORES

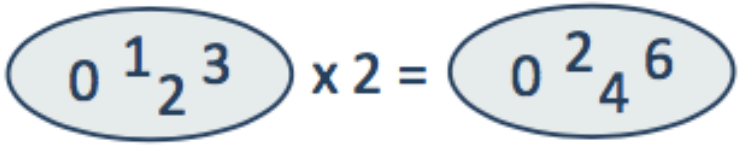
Lists vs NDArrays

```
>>> x = [1,2,3]
>>> print 2 * x
[1, 2, 3, 1, 2, 3]
```

```
>>> x = numpy.array([1,2,3])
>>> print 2 * x
[2, 4, 6]
>>> print 2.0 * x
[2.0, 4.0, 6.0]
```

For more of NumPy methods, check the NumPy Documentation

numpy.arange(4)




```
>>> import numpy
>>> x = nunmpy.range(5)
>>> x.__class__
<type 'ndarray'>
>>> x
array([0, 1, 2, 3, 4])
>>> 2 * x
Array([0, 2, 4, 6, 8])
```

VETORES

```
>>> import numpy
```

NDArrays are faster and easier to use!

```
>>> x = numpy.array([1,2,3])
```

Add, subtract, multiply, divide, power all elements at once!

```
>>> 2 + x
array([3, 4, 5])
```

+

```
>>> 2 * x
array([2, 4, 6])
```

*

```
>>> 2 ** x
array([2, 4, 8])
```

**

```
>>> 2 - x
array([-1, 0, 1])
```

-

```
>>> 2.0 / x
array([2.0, 1.0, 1.33…])
```

/

```
>>> 2 % x
array([0, 0, 2])
```

%

OPERAÇÕES

- Exemplos de operações:
  1. b = np.arange(10)
  2. a=np.copy(b)
  3. a=np.median(b)
  4. a=np.gradient(b)
  5. len(a)
  6. a=np.sort(b)
  7. a.max(), a.min()
  8. np.shape(a)
  9. np.size(a)

INPUT DE DADOS

Input de dados:

```
nome = raw_input("\nQual o seu nome:")
idade = int(raw_input("\nQual a sua idade:"))
peso = float(raw_input("\nQual a sua massa:"))
```

IF / ELIF / ELSE

```
fruit = 'banana'
if fruit is 'apple':
    eat_it()
elif fruit is 'orange':
    make_a_juice()
else:
    leave_it()
```

```
fruit = 'banana'
if fruit is 'apple':
....eat_it()
elif fruit is 'orange':
....make_a_juice()
else:
....leave_it()
```

FOR

```
for i in [1,2,3]:
    print(i)
```

```
1
2
3
```

```
for i in range(10):
    if i > 5:
        print(i)
```

```
6
7
8
9
```

```
for i in range(10):
    if i > 5:
        print(i)
    else:
        print('i lower than five')
```

```
i lower than five
i lower than five
i lower than five
i lower than five
i lower than five
i lower than five
6
7
8
9
```

FOR

```
my_list = [‘a’, ‘b’ ‘c’]

for index in range(len(my_list)):
....print index, my_item[index]

0 a
1 b
2 c
```

WHILE

Enquanto a condição for verdadeira...

```
while < condição:
    do something
```

Contagem regressiva ...

```
x = 10
while x>=1:
    print(x)
    x = x-1
```

DEFININDO FUNÇÕES

Positional Arguments

```
def my_method(x, y, z=2, w=3):
    """Add here some documentation"""
    k = 2 * x - y / z + w
    return k
```

BA DUM TSSS

Positional arguments, their position matters.

```
>>> my_method(1,2)
>>> my_method(1,2,1,0)
```

LER E ESCREVER ARQUIVOS

Criando uma tabela:

```
f = open ("myfile.txt", "w")   # No modo write
random = np.random.random(100)
for i in xrange(len(random)):
    f.write(str(random[i]) + "\n")
f.close()
```

Lendo os dados de uma tabela:

```
f = open ("myfile.txt", "r")   # No modo read
lines = f.readlines()   # salva linhas na variável lines
print(lines)   # imprime as linhas
f.close()
```

TABELA COM 3 COLUNAS

```
import numpy as np
f = open ("myfile.txt", "w")   # No modo write
x = np.random.random(100)
y = np.random.random(100)
z = np.random.random(100)
for i in range(len(random)):
    f.write(str(x[i]) + "\t" + str(y[i]) +
            "\t" + str(z[i]) + "\n")
f.close()
```

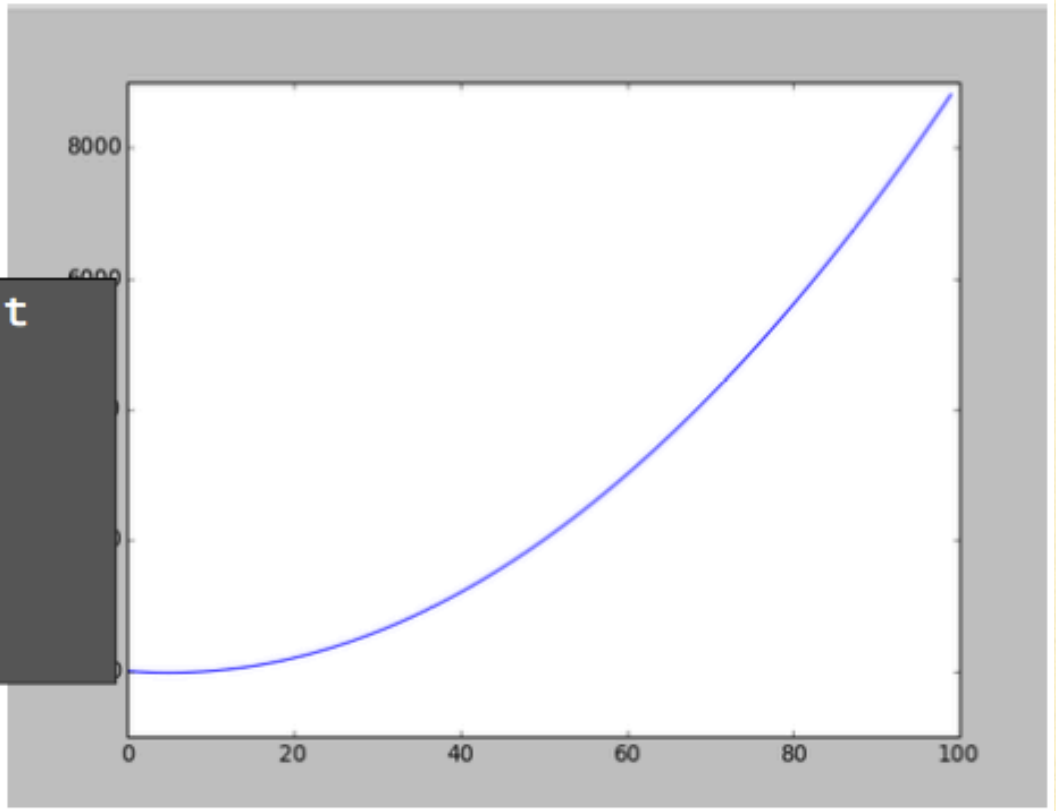
```
tabela = np.loadtxt("myfile.txt")
x = tabela[:, 0]
y = tabela[:, 1]
z = tabela[:, 2]
```

MUITO MAIS FÁCIL...

```
# lendo dados de um arquivo
tabela = np.loadtxt(table, skiprows=1)

# Definindo as variaveis das colunas
lbd = tabela[:, 0]
flux = tabela[:, 1]
error = tabela[:, 2]
```

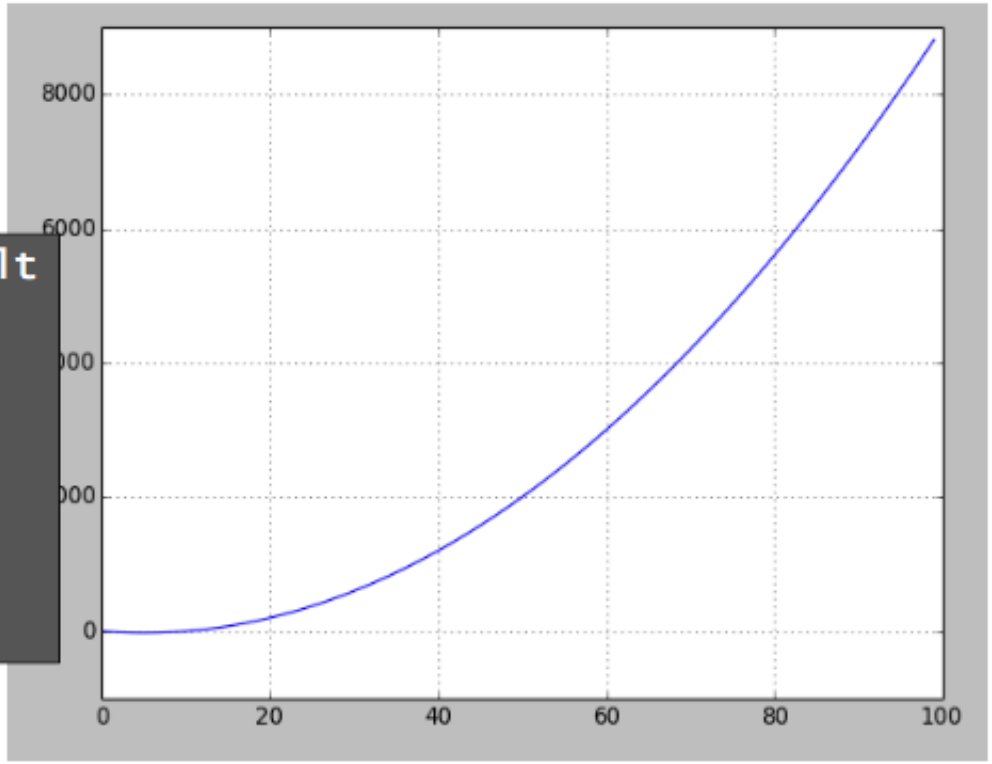
GRÁFICOS

```
>>> import matplotlib.pyplot as plt
>>> import numpy

>>> x = numpy.arange(100)
>>> y = x ** 2 - 10 * x + 2

>>> plt.plot(x, y)
>>> plt.show()
```



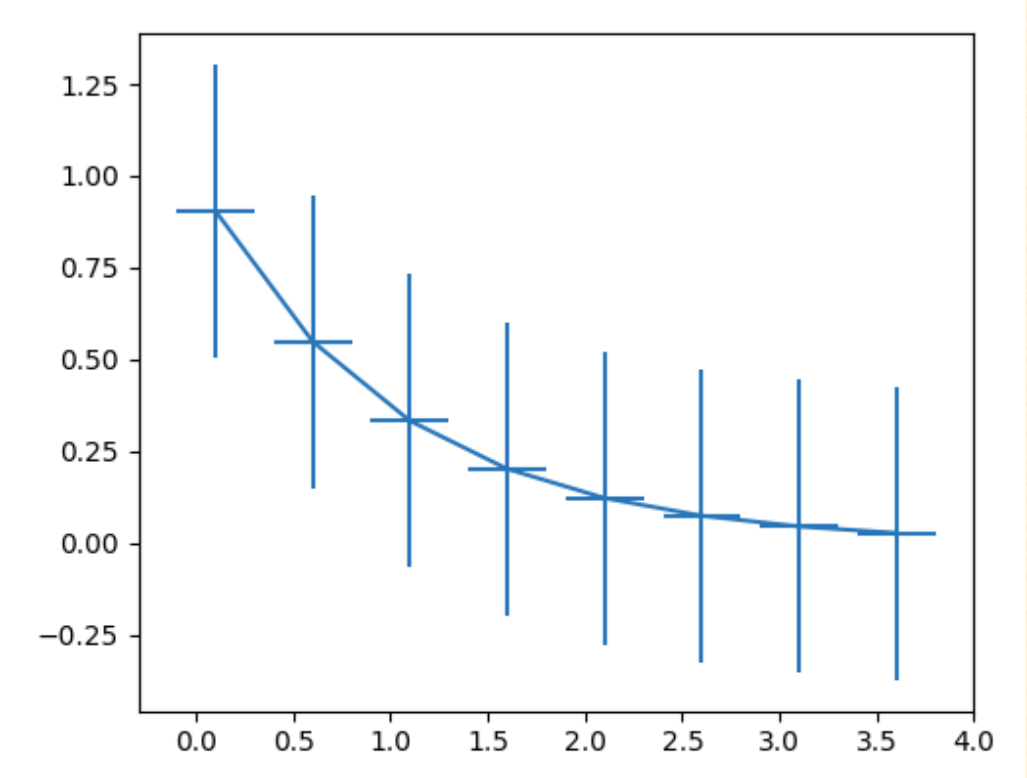
GRÁFICOS

```
>>> import matplotlib.pyplot as plt
>>> import numpy

>>> x = numpy.arange(100)
>>> y = x ** 2 - 10 * x + 2

>>> plt.plot(x, y)
>>> plt.show()
>>> plt.grid()
```



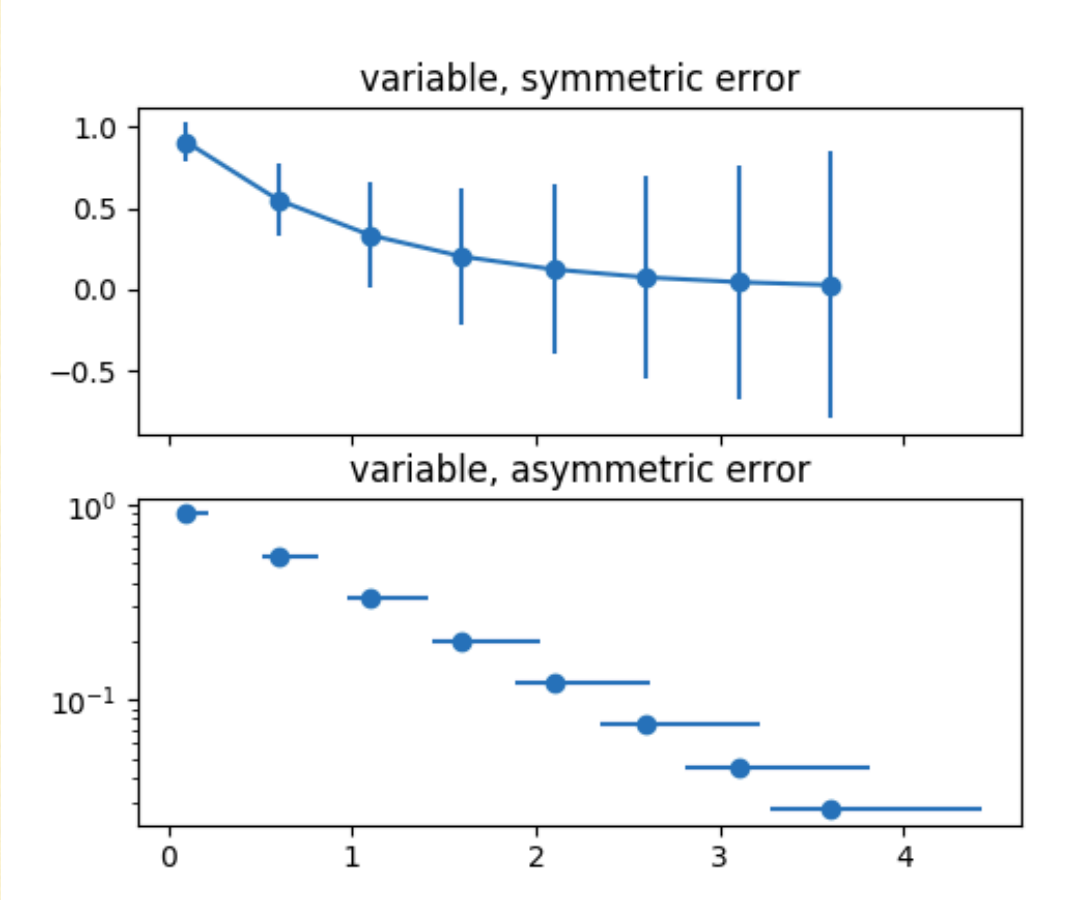
GRÁFICOS

```
import numpy as np
import matplotlib.pyplot as plt

# example data
x = np.arange(0.1, 4, 0.5)
y = np.exp(-x)

fig, ax = plt.subplots()
ax.errorbar(x, y, xerr=0.2, yerr=0.4)
plt.show()
```



GRÁFICOS

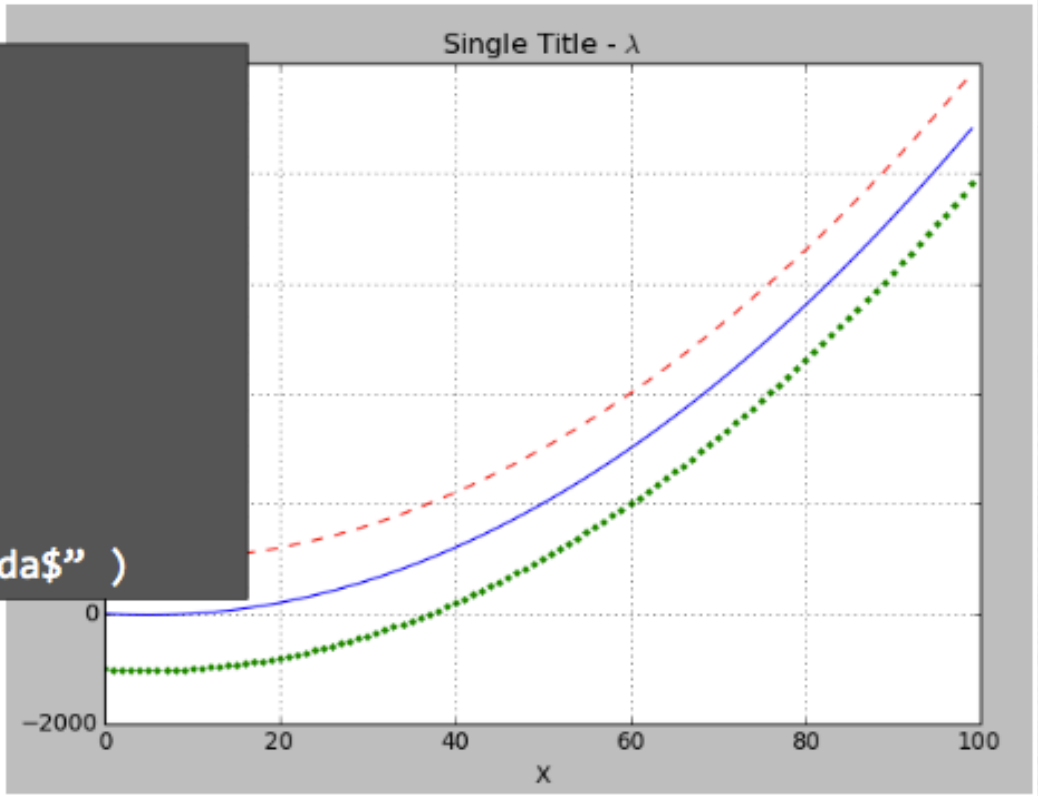
```python
import numpy as np
import matplotlib.pyplot as plt

# example data
x = np.arange(0.1, 4, 0.5)
y = np.exp(-x)

# example error bar values that vary with x-position
error = 0.1 + 0.2 * x

fig, (ax0, ax1) = plt.subplots(nrows=2, sharex=True)
ax0.errorbar(x, y, yerr=error, fmt='-o')
ax0.set_title('variable, symmetric error')

# error bar values w/ different -/+ errors that
# also vary with the x-position
lower_error = 0.4 * error
upper_error = error
asymmetric_error = [lower_error, upper_error]

ax1.errorbar(x, y, xerr=asymmetric_error, fmt='o')
ax1.set_title('variable, asymmetric error')
ax1.set_yscale('log')
plt.show()
```



GRÁFICOS

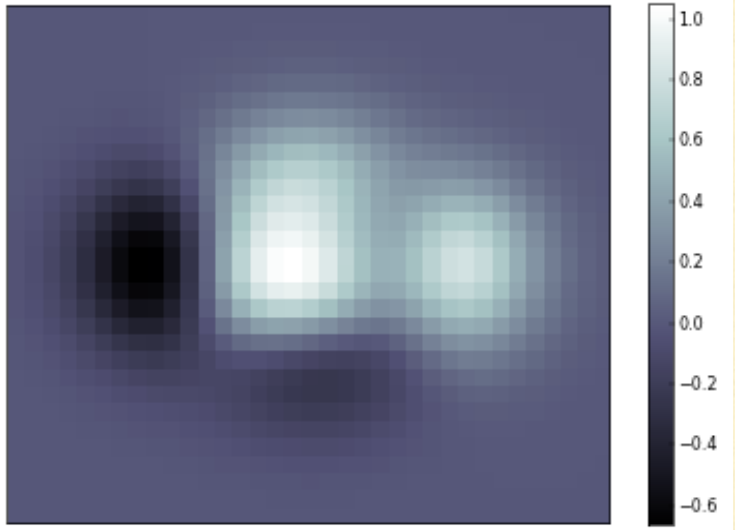
```
>>> import matplotlib.pyplot as plt
>>> import numpy
>>> x = numpy.arange(100)
>>> y = x ** 2 - 10 * x + 2
>>> plt.plot(x, y)
>>> plt.show()
>>> plt.grid()
>>> plt.xlabel("X")
>>> plt.plot(x, y + 1000, 'r--')
>>> plt.plot(x, y - 1000, 'g.')
>>> plt.title( u"Single title -$\lambda$" )
```

For more plotting styles, check out:
matplotlib.pyplot.plot



GRÁFICOS

```
# -----------------------------------------------------------------------------
# Copyright (c) 2015, Nicolas P. Rougier. All Rights Reserved.
# Distributed under the (new) BSD License. See LICENSE.txt for more info.
# -----------------------------------------------------------------------------
import numpy as np
import matplotlib.pyplot as plt

def f(x,y):
    return (1-x/2+x**5+y**3)*np.exp(-x**2-y**2)

n = 10
x = np.linspace(-3,3,3.5*n)
y = np.linspace(-3,3,3.0*n)
X,Y = np.meshgrid(x,y)
Z = f(X,Y)

plt.axes([0.025,0.025,0.95,0.95])
plt.imshow(Z,interpolation='nearest', cmap='bone', origin='lower')
plt.colorbar(shrink=.92)

plt.xticks([]), plt.yticks([])
# savefig('../figures/imshow_ex.png', dpi=48)
plt.show()
```



https://www.labri.fr/perso/nrougier/teaching/matplotlib/#imshow

COMO ACHAR OS ERROS DOS PARÂMETROS

1) Use a rotina stat_basics para aprender como calcular o delta_chi2 correspondente à margem de erro procurada
2) Calcule o delta_chi2 para o número de pontos da sua tabela
3) Grafique este delta_chi2 como uma linha horizontal nos gráficos de chi2 vs. T1 e T2