Introdu??o Python PET El?trica

November 25, 2016

1 Introdução à linguagem Python

1.0.1 25 anos PET Elétrica - 25/11/2016

1.1 Tipos de variáveis

```
In [14]: from math import pi
         inteiro = 10
         print inteiro

         real = 2 * pi * inteiro ** 2
         print real

         cadeia_de_caracteres = "PET Elétrica"
         print "25 anos " + cadeia_de_caracteres

         booleano_verdadeiro = True # 1, [1, 2, 3], (4, 5, 6) etc
         booleano_falso = False # 0 [], (), None etc
         print str(booleano_falso) + " is not " + str(booleano_verdadeiro)

         variavel_que_nao_conheco = 3
         print type(variavel_que_nao_conheco)
```

```
10
628.318530718
25 anos PET Elétrica
False is not True
<type 'int'>
```

2 Iteráveis !!!!

2.1 listas

```
In [290]: listas = [1, 2, 3, 4, 5]
          carrinho_de_compra = ["kindle", "calculadora", "celular"]
          lista_complexa = [[1, 3, 4], "PET", -1]
          print lista_complexa
          print "kindle" in carrinho_de_compra
          print "tablet" in carrinho_de_compra

          print "Tamanho da lista"
          print len(carrinho_de_compra)
```

```
[[1, 3, 4], 'PET', -1]
True
False
Tamanho da lista
3
```

2.2 tuplas

```
In [92]: imutavel = (102.23, 43, carrinho_de_compra, cadeia_de_caracteres)
         print imutavel
```

```
(102.23, 43, ['kindle', 'calculadora', 'celular'], 'PET El\xc3\xa9trica')
```

2.3 Dicionários

```
In [100]: dicionario = {'elemento 1': [1, 2, 4, 5],'elemento 2': carrinho_de_compra, 'elemento 3': imuta
          print dicionario

          print dicionario.keys()
          print dicionario.values()

          segundo_dicionario = dict(chave1=[1, 2, 3, 4], chave2='valor2')
          print segundo_dicionario
```

```
{'elemento 1': [1, 2, 4, 5], 'elemento 3': (102.23, 43, ['kindle', 'calculadora', 'celular'], 'PET El\xc
['elemento 1', 'elemento 3', 'elemento 2']
[[1, 2, 4, 5], (102.23, 43, ['kindle', 'calculadora', 'celular'], 'PET El\xc3\xa9trica'), ['kindle', 'ca
{'chave1': [1, 2, 3, 4], 'chave2': 'valor2'}
```

2.3.1 Indexação - [0]

2.4 Conjuntos

```
In [213]: conjunto_a = {1, 40, 4, 10}
          conjunto_b = {3, 5, 10, 4, "a", 4}
          print conjunto_b
          print conjunto_a - conjunto_b # Diferença
          print conjunto_b | conjunto_a # União
          print conjunto_a & conjunto_b # Interseção
```

```
set(['a', 10, 3, 4, 5])
set([40, 1])
set(['a', 1, 3, 4, 5, 40, 10])
set([10, 4])
```

```
In [218]: #Start, Stop, Step
          lista_de_numeros = range(10)

          # elemento na primeira posicao
          lista_de_numeros[0]

          # elemento na terceira posicao
          lista_de_numeros[2]

          # elementos da segunda até a quinta posição
          lista_de_numeros[1:5]  # inclusivo e exclusivo
```

```
# do primeiro ao quarto elemento
lista_de_numeros[0:4]
lista_de_numeros[:4]
print lista_de_numeros[0:4] == lista_de_numeros[:4]

# do quinto até o último elemento
lista_de_numeros[4:len(lista_de_numeros)]
lista_de_numeros[4:]
print lista_de_numeros[4:10] == lista_de_numeros[4:]

# Step
# do primeiro até o último com passo de 2
lista_de_numeros[0:10:2]
lista_de_numeros[::2]
print lista_de_numeros[0:10:2] == lista_de_numeros[::2]

# Strings também são iteráveis
cidade = 'Universidade Federal de Juiz de Fora'
# Primeiro elemento
print cidade[0]

# Último elemento
print cidade[-1]

# Inverso
cidade[::-1]

# Assertiva In
"Juiz de Fora" in cidade
```

```
True
True
True
U
a
```

Out[218]: True

3 Tudo em Python é um objeto

3.0.1 Tudo possui atributos e métodos

In [127]:
```
# STRINGS

arquivo_de_texto = "linha1\r\nlinha2\r\nlinha3"
# alguns métodos de strings
#arquivo_de_texto.split('\r\n')
print arquivo_de_texto.upper()

# LISTAS

minha_lista = [4, 3, 2, 1]
#minha_lista[4] = 10
minha_lista.append(10)
```

```
        minha_lista.extend([0, -1, -2])
        # minha_lista.append([0, -1, -2])
        print minha_lista
        minha_lista.sort()
        print minha_lista


        #DICIONARIOS

        materia_da_prova = {'Prova1': ('pagina20-30', 'trabalho extra'),
                            'Prova2': ('capitulos10-15', 'artigo cientifico')}
        materia_da_prova.update({'Prova3': ('apenas trabalho')})
        print materia_da_prova
```

```
LINHA1
LINHA2
LINHA3
[4, 3, 2, 1, 10, 0, -1, -2]
[-2, -1, 0, 1, 2, 3, 4, 10]
{'Prova1': ('pagina20-30', 'trabalho extra'), 'Prova2': ('capitulos10-15', 'artigo cientifico'), 'Prova3
```

4 Nome de variáveis

4.0.1 Começar por número?

```
In [18]: 25anos = 25
```

```
  File "<ipython-input-18-57821f29f044>", line 1
    25anos = 25
        ^
SyntaxError: invalid syntax
```

4.0.2 Começar por underscore?

```
In [76]: _significado_da_vida_do_universo_e_tudo_mais = 42
         # Nao tenham medo de colocar nomes grandes
```

4.0.3 Começar por símbolos esquisitos?

```
In [24]: $anos = 30
```

```
  File "<ipython-input-24-296e1492c816>", line 1
    $anos = 30
    ^
SyntaxError: invalid syntax
```

4.0.4 camelCase

```
In [25]: algunsGostamAssim = "tudo bem"
```

5 Controle de fluxo

6 If, else, elif

```
In [128]: sol = []
          if sol:
              print "vou a praia"
          else:
              print "vou estudar eletromag :("
```

```
vou estudar eletromag :(
```

```
In [29]: nota = 5
         if nota <= 4:
             print "Reprovado"
         elif nota > 4 and nota < 6:
             print "Fazer trabalho extra"
         else:
             print "aprovado"
```

```
Fazer trabalho extra
```

```
In [39]: sol = True
         cerveja = []


         if sol or cerveja:
         print "churrasco"
         else:
             print "bar"
```

```
          File "<ipython-input-39-69989d0bcc70>", line 6
        print "churrasco"
            ^
    IndentationError: expected an indented block
```

6.1 for

```
In [43]: iteravel = ["1", "3", 4, True, [3, 2, "1"]]
         for objeto in iteravel:
             print objeto
```

```
1
3
4
True
[3, 2, '1']
```

6.2 while

```
In [75]: import random
```

```
numero_aleatorio = random.random()
numero_iteracoes = 0

while numero_aleatorio < 0.8:
    numero_aleatorio = random.random()
    numero_iteracoes += 1
    print numero_iteracoes
```

```
1
2
```

7 Iterando sobre iteráveis !!!

7.0.1 List comprehension

```
In [135]: number_list = []
          for k in range(10):
              number_list.append(k)
          print number_list

          same_list = [k for k in range(10)]
          print number_list == same_list

          filtered_list = [k for k in range(10) if not k % 2]
          print filtered_list
```

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

7.0.2 Old style

```
In [138]: prices = [30.4, 45.6, 120.4]
          products = ['book', 'game', 'hard drive']
          counter = 0
          for prod in products:
              print prod + " $" + str(prices[counter])
              counter += 1
```

```
book $30.4
game $30.4
hard drive $30.4
```

7.0.3 Python Style

```
In [139]: for index, prod in enumerate(products):
              print prod + " $" + str(prices[index])
```

```
book $30.4
game $45.6
hard drive $120.4
```

7.0.4 Iterando em dicionários

```
In [143]: for key, value in materia_da_prova.items():
              print key, value
```

```
Prova1 ('pagina20-30', 'trabalho extra')
Prova2 ('capitulos10-15', 'artigo cientifico')
Prova3 apenas trabalho
```

8 Funções

```
In [197]: def power2(argument):
              return argument ** 2

          print power2(4)
          print power2(10)
```

```
16
100
```

```
In [201]: def custom_power(argument, power):
              return argument ** power

          print custom_power(4, 2)
          print custom_power(4, 3)
```

```
16
64
```

```
In [80]: def list_filter(elements_list):
             new_list = []
             for element in elements_list:
                 if element > 0:
                     new_list.append(element)
                     # Se element for maior que 0?

             return new_list


         money_amount = [100.23, 4503.23, -120.23, 90.56]
         positive_money_amount = list_filter(money_amount)
         print positive_money_amount
```

```
[100.23, 4503.23, 90.56]
```

```
In [86]: def flexible_list_filter(elements_list, criteria=0):
             new_list = []
             for element in elements_list:
                 if element > criteria:
                     new_list.append(element)
                     # Se element for maior que 0?

             return new_list

         positive_money_amount = flexible_list_filter(money_amount, criteria=200)
         print positive_money_amount
```

```
[4503.23]
```

9 Numpy, Scipy, Matplotlib e Pandas

10 Numpy

https://docs.scipy.org/doc/numpy/

10.1 Criando meus dados

```
In [221]: import numpy as np
          array_of_floats = np.array([1, 2, 3, 4])
          type(array_of_floats)
```

```
Out[221]: numpy.ndarray
```

```
In [242]: numpy_list = np.arange(100000)

          %timeit np.cumsum(python_list)
```

```
1000 loops, best of 3: 982 µs per loop
```

```
In [243]: def custom_cumsum(values):
              result = 0
              output = []
              for val in values:
                  result += val
                  output.append(result)
              return output

          %timeit custom_cumsum(python_list)
```

```
1000 loops, best of 3: 1.02 ms per loop
```

10.1.1 Indexing and slicing

Uma dimensão

```
In [297]: values = np.arange(10)
          print values
          print values[0]
          print values[0:4]
```

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

Mais de uma dimensão

```
In [304]: matrix = np.matrix([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
          print matrix
          print matrix[0]
          print matrix[0, 1]
          print matrix[1, 0:2]
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
[[1 2 3]]
2
[[4 5]]
```

10.1.2 indexação lógica

```
In [308]: values = np.arange(10)
          print values[values > 5]
          print values[values < 6]
```

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

10.1.3 E quando queremos usar indexação lógica combinada???

10.2 Operações básicas

10.2.1 Transposta

```
In [280]: matrix = np.matrix([[1, 2, 3], [4, 5, 6]])
          print matrix
          print matrix.T
```

```
[[1 2 3]
 [4 5 6]]
[[1 4]
 [2 5]
 [3 6]]
```

10.2.2 Multiplicação por um escalar

```
In [281]: scalar = 3
          print scalar * matrix
```

```
[[ 3  6  9]
 [12 15 18]]
```

10.2.3 Benefícios do Numpy sobre as listas do Python

```
In [289]: number_list = [1, 2, 3, 4]
          print number_list * 3
          print number_list + 4
```

```
[1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4]
```

```
        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-289-0e50c8fb1ac0> in <module>()
          1 number_list = [1, 2, 3, 4]
          2 print number_list * 3
    ----> 3 print number_list + 4


        TypeError: can only concatenate list (not "int") to list
```

10.2.4 Soma de matrizes

```
In [ ]: matrix2 = np.ones((2, 3))
        print matrix + matrix2
```

10.2.5 Multiplicação de matrizes

```
In [282]: print "Multiplicação de matrizes"
          print matrix.T * matrix2 # Devemos seguir a álgebra linear
```

```
Multiplicação de matrizes
[[ 5.  5.  5.]
 [ 7.  7.  7.]
 [ 9.  9.  9.]]
```

10.2.6 Inversão

```
In [283]: print "Inversão"
          print np.invert(matrix)
```

```
Inversão
[[-2 -3 -4]
 [-5 -6 -7]]
```

10.2.7 Saber ou mudar a dimensão

```
In [287]: print "Saber a dimensão"
          print matrix2.shape

          print "Mudar a dimensão"
          matrix3 = np.random.randn(10)
          matrix3 = matrix3.reshape(2, 5)
          print matrix3
          print matrix3.shape
```

```
Saber a dimensão
(2, 3)
Mudar a dimensão
[[ 0.5807347  -1.13342708 -1.68574094  1.16625932  0.02954759]
 [ 0.3245368   0.39775207  0.79460561 -0.10508138  1.40892655]]
(2, 5)
```

10.3 Elementwise in arrays

```
In [275]: random_array = np.random.randn(10)
          print np.mean(random_array)
          print random_array.mean()
          print random_array.std()
```

```
-0.0563941650787
-0.0563941650787
0.898380307227
1
```

```
In [294]: from math import sqrt
          list_of_numbers = [1, 2, 3, 4, 5]
          #print sqrt(list_of_numbers)
          #[sqrt(val) for val in list_of_numbers]

          print np.sqrt(list_of_numbers)
```

```
[ 1.          1.41421356  1.73205081  2.          2.23606798]
```

11 Exercícios

Create a vector with values ranging from 10 to 49
Reverse a vector (first element becomes last)
Create a 3x3 matrix with values ranging from 0 to 8
Create a 10x10 array with random values and find the minimum and maximum values

12 Matplotlib

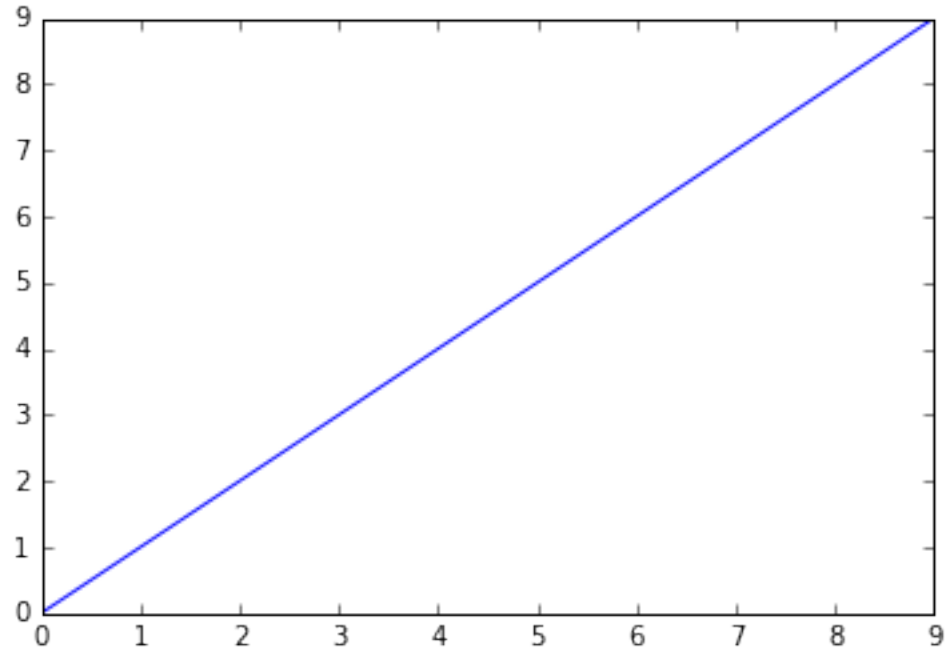
http://matplotlib.org/contents.html

```
In [310]: import matplotlib.pyplot as plt
          %matplotlib inline
```

12.0.1 primeiro plot

```
In [312]: values = np.arange(10)
          plt.plot(values)
```

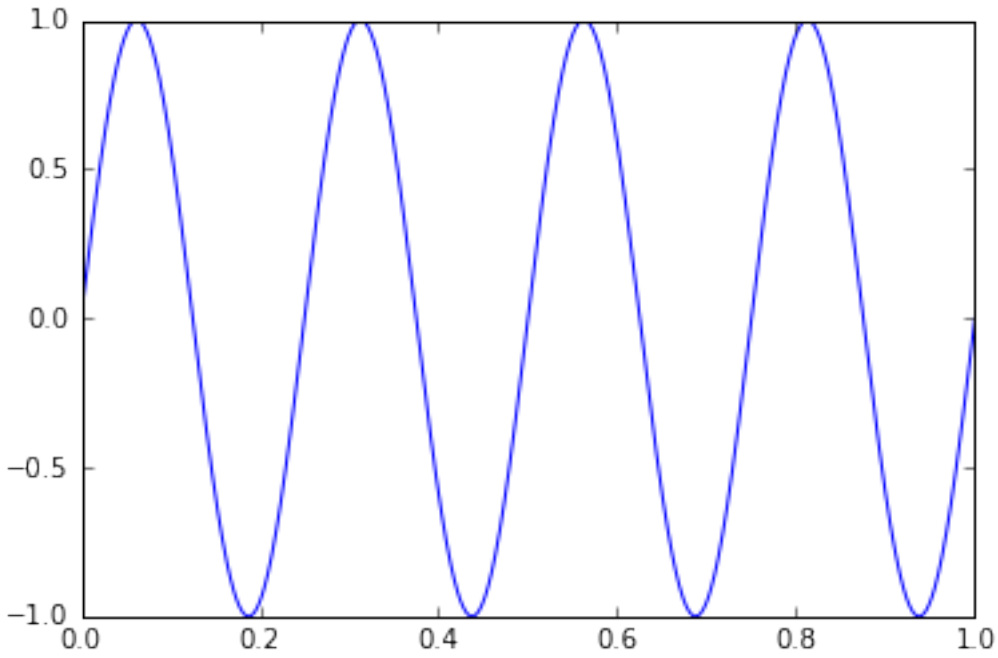
```
Out[312]: [<matplotlib.lines.Line2D at 0x10d46bcd0>]
```



12.0.2 par ordenado

```
In [313]: t = np.linspace(0, 1, 1000)
          sin = np.sin(2 * pi * t * 4  + 0)
          plt.plot(t, sin)
```

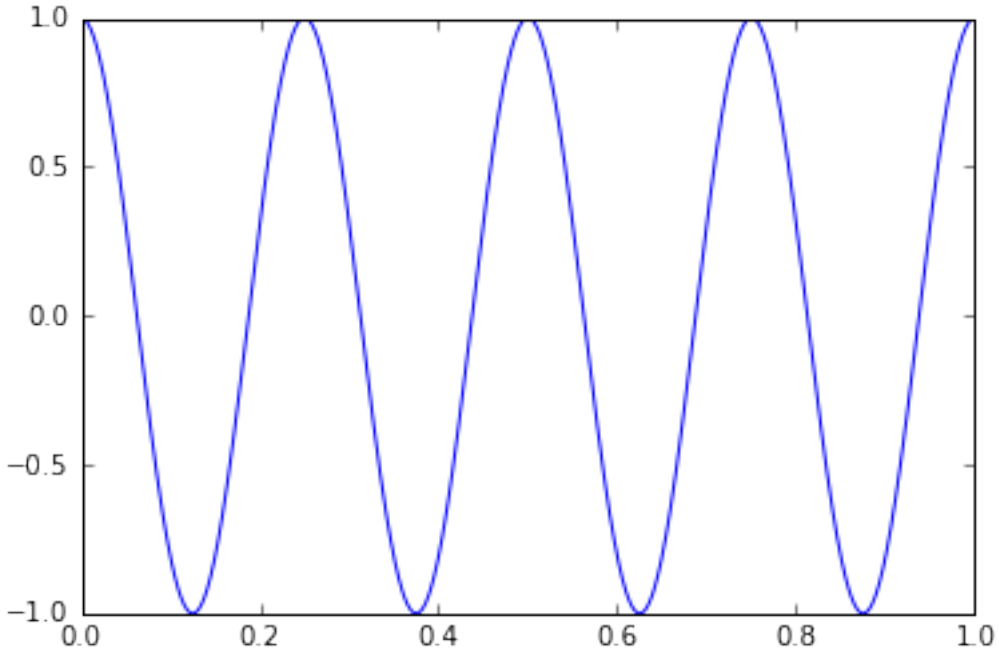
```
Out[313]: [<matplotlib.lines.Line2D at 0x10d591290>]
```



12.0.3 Adicionando Labels

```
In [314]: t = np.linspace(0, 1, 1000)
          cos = np.cos(2 * pi * t * 4  + 0)
          plt.plot(t, cos)
```

```
Out[314]: [<matplotlib.lines.Line2D at 0x10d628c50>]
```



12.1 Vamos para o IPython

12.1.1 Exercícios

Plotar um seno (4 Hz) e um cosseno (2 Hz) juntos (podem escolher as cores)
Fazer um subplot com 2 linhas e 1 coluna: 1 - 10000 valores pseudo-aleatórios 2 - seu histogram

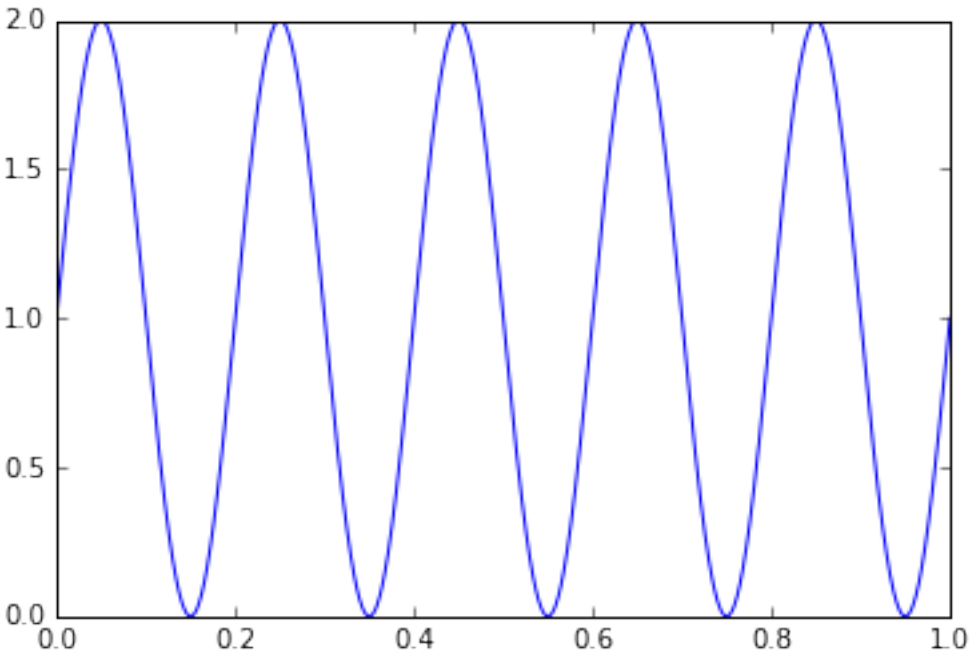
13 Scipy

https://docs.scipy.org/doc/scipy/reference/

13.1 Fitting

In [348]:
```
from scipy.integrate import trapz

t = np.linspace(0, 1, 1000)
sin = np.sin(2 * np.pi * t * 5)
plt.plot(t, sin)

auc = trapz(t, sin)

print "Area sob a curva: "  + str(auc)
```

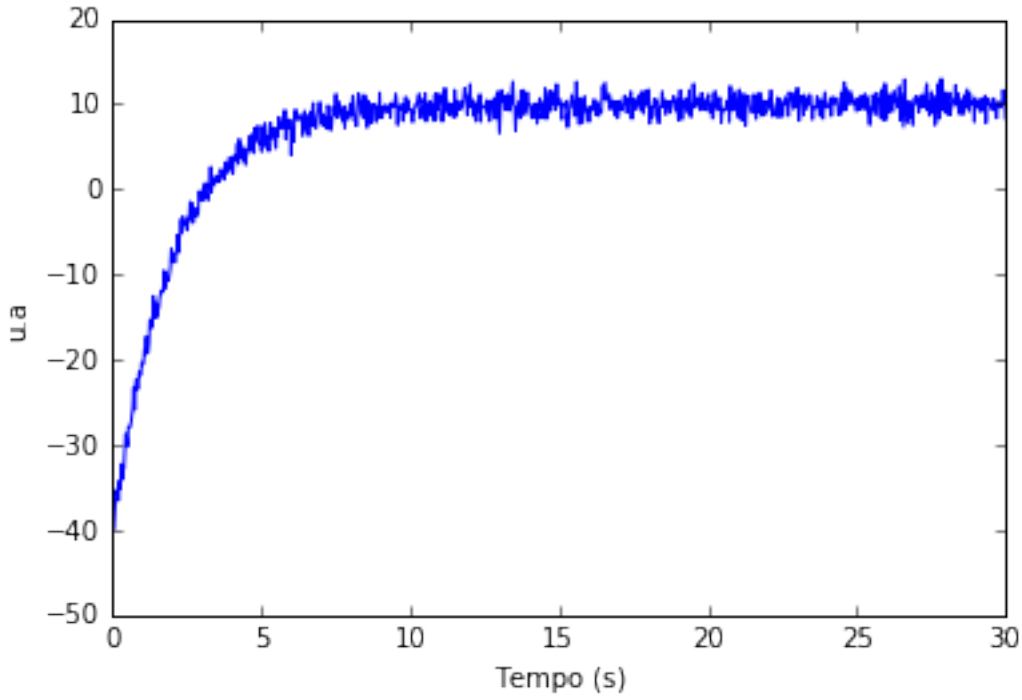
```
Area sob a curva: -1.72778458207e-15
```



In [ ]:
```
auc = trapz
```

In [323]:
```
x = np.linspace(0, 30, 1000)
y = 10 - 50 * np.exp(-x / 2) + np.random.randn(1000)
plt.plot(x, y)
plt.xlabel('Tempo (s)')
plt.ylabel('u.a')
```

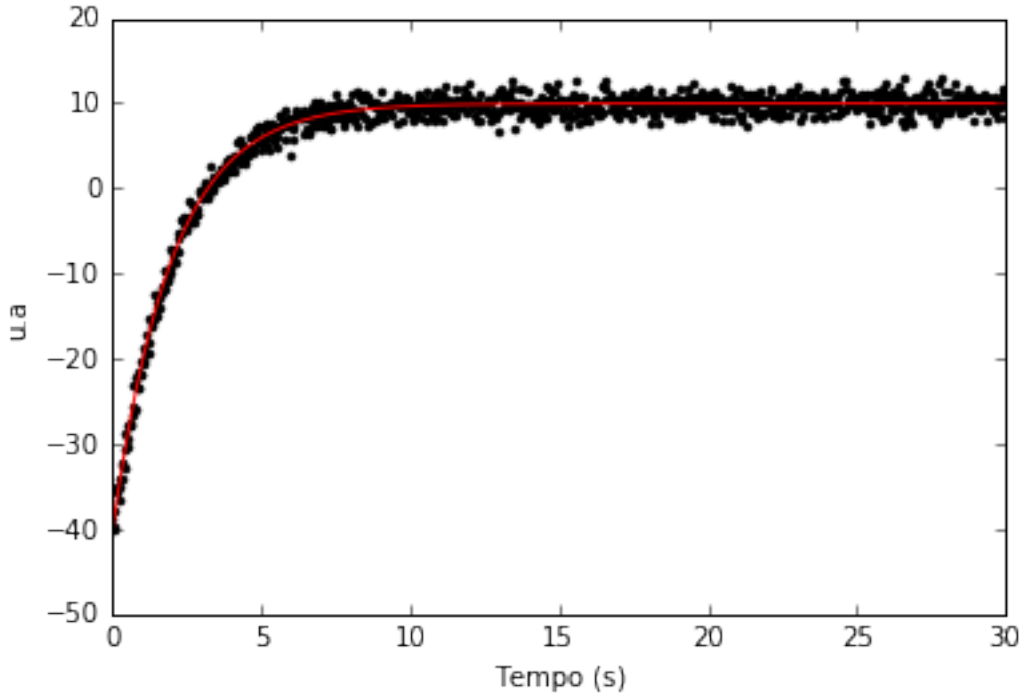
Out[323]: <matplotlib.text.Text at 0x10ddc4c90>



In [328]:
```
from scipy.optimize import leastsq

#Define the exponential model
def my_fun(par, t, y):
    err = y - (par[0] + par[1] * np.exp(-t / par[2]))
    return err

#Function to evaluate the model
def fun_eval(par, t):
    return par[0] + par[1] * np.exp(-t / par[2])

#Initial guesses
p0 = [min(y), max(y) - min(y), x[-1] / 3]

#Nonlinear fit
result = leastsq(my_fun, p0, args=(x, y))


#Plot the curves
plt.plot(x, y, 'k.')
```

```
plt.plot(x, fun_eval(result[0], x), 'r')
plt.xlabel('Tempo (s)')
plt.ylabel('u.a')
```

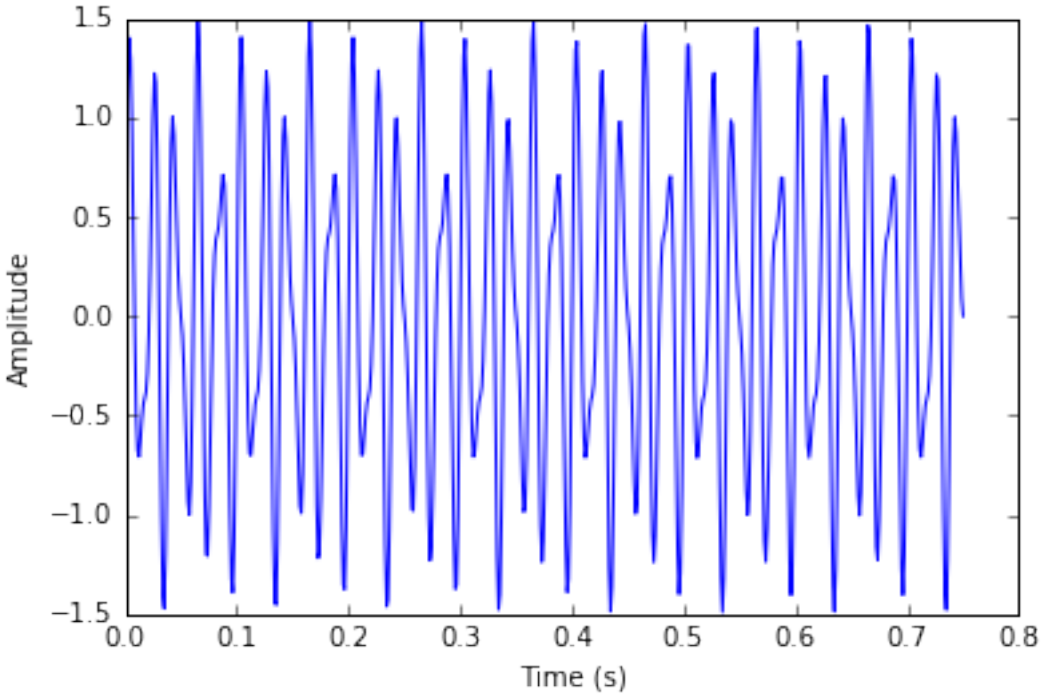
```
Out[328]: <matplotlib.text.Text at 0x10ea55690>
```



13.1.1 Fast Fourier Transform

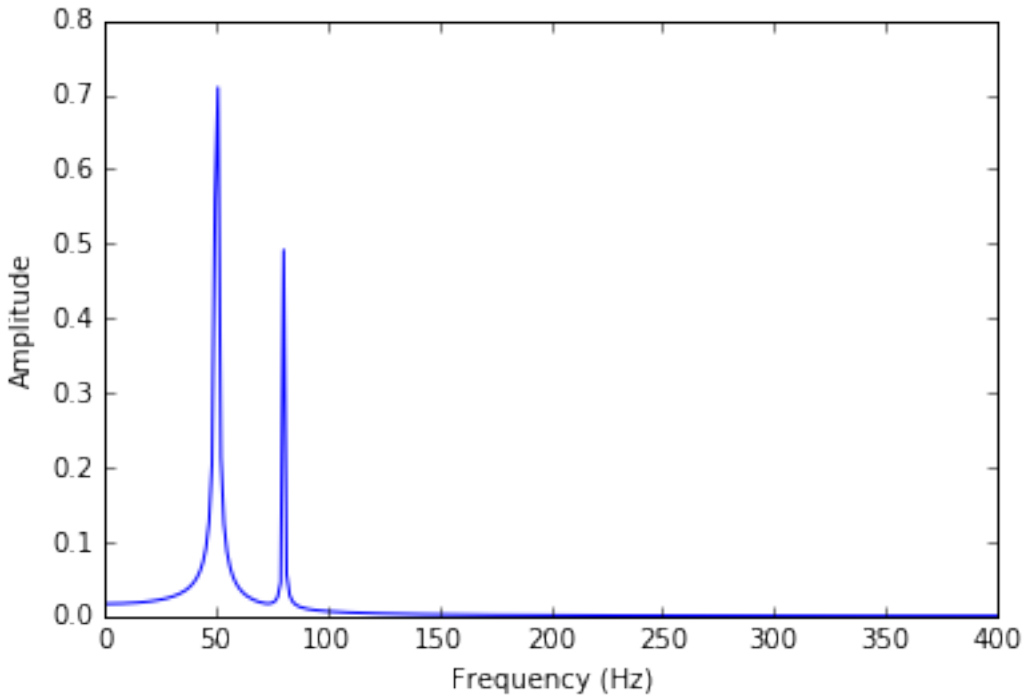
```
In [335]: # Number of samplepoints
          N = 600
          # sample spacing
          T = 1.0 / 800.0
          x = np.linspace(0.0, N * T, N)
          y = np.sin(50.0 * 2.0 * np.pi * x) + 0.5 * np.sin(80.0 * 2.0 * np.pi * x)
          plt.plot(x, y)
          plt.xlabel('Time (s)')
          plt.ylabel('Amplitude')
```

```
Out[335]: <matplotlib.text.Text at 0x10f828850>
```



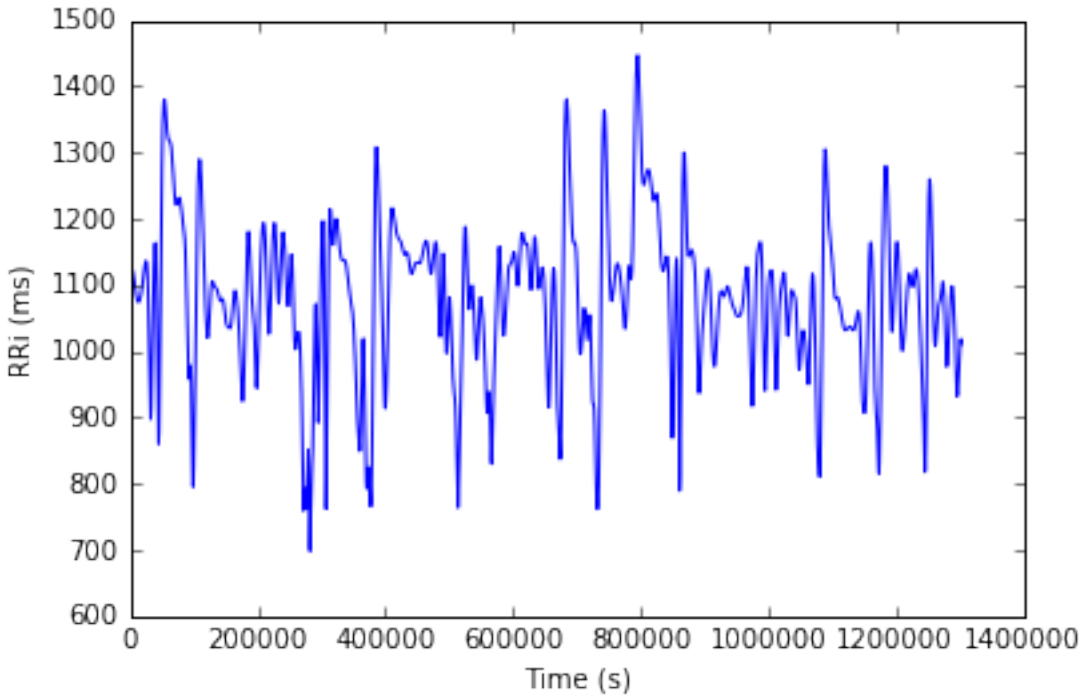
In [338]:
```
from scipy.fftpack import fft

yf = fft(y)
xf = np.linspace(0.0, 1.0/(2.0*T), N/2)
plt.plot(xf, 2.0/N * np.abs(yf[:N//2]))
plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude')
```

Out[338]:
```
<matplotlib.text.Text at 0x10f99b110>
```



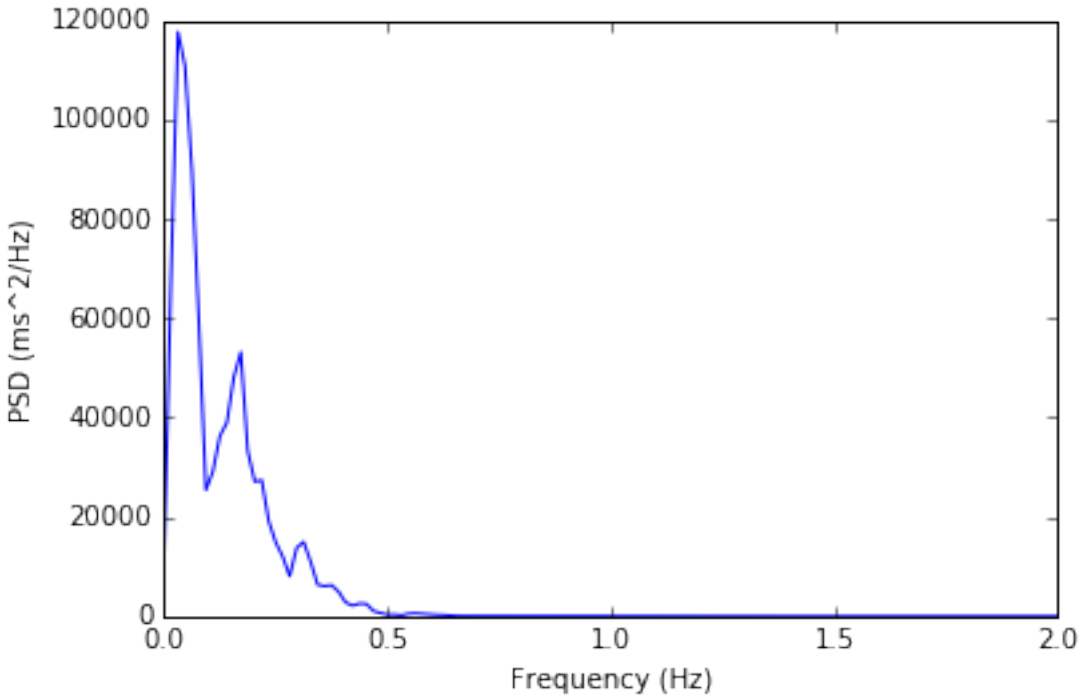
13.1.2 Power spectral density estimation

```
In [331]: rri = np.array([float(rri.strip()) for rri in open('dados/rri.txt')])
          rri_time = np.cumsum(rri)
          rri_time -= rri_time[0]

          plt.plot(rri_time, rri)
          plt.xlabel('Time (s)')
          plt.ylabel('RRi (ms)')

Out[331]: <matplotlib.text.Text at 0x10dbfd610>
```



In [334]:
```
from scipy.signal import welch
#PSD esitmation with hanning window, 256 points each segment with 50% overlap.
Fxx, Pxx = welch(x=rri, fs=4.0, window="hanning",
                 nperseg=256, noverlap=128,
                 detrend="linear")


plt.plot(Fxx, Pxx)
plt.xlabel('Frequency (Hz)')
plt.ylabel('PSD (ms^2/Hz)')
```

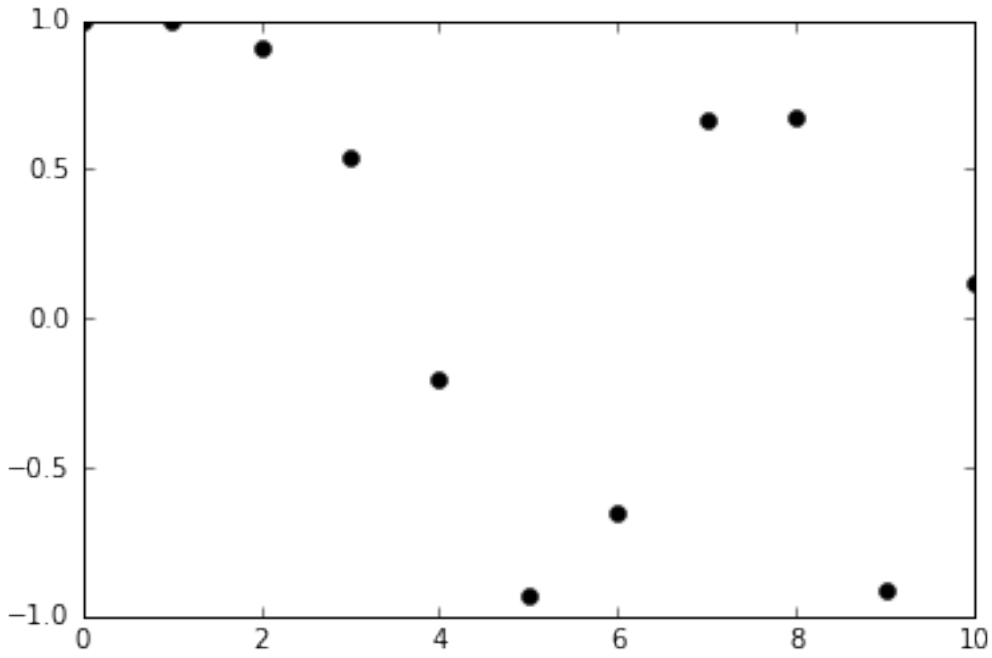
Out[334]:
```
<matplotlib.text.Text at 0x10f724c50>
```



13.1.3 Interpolação

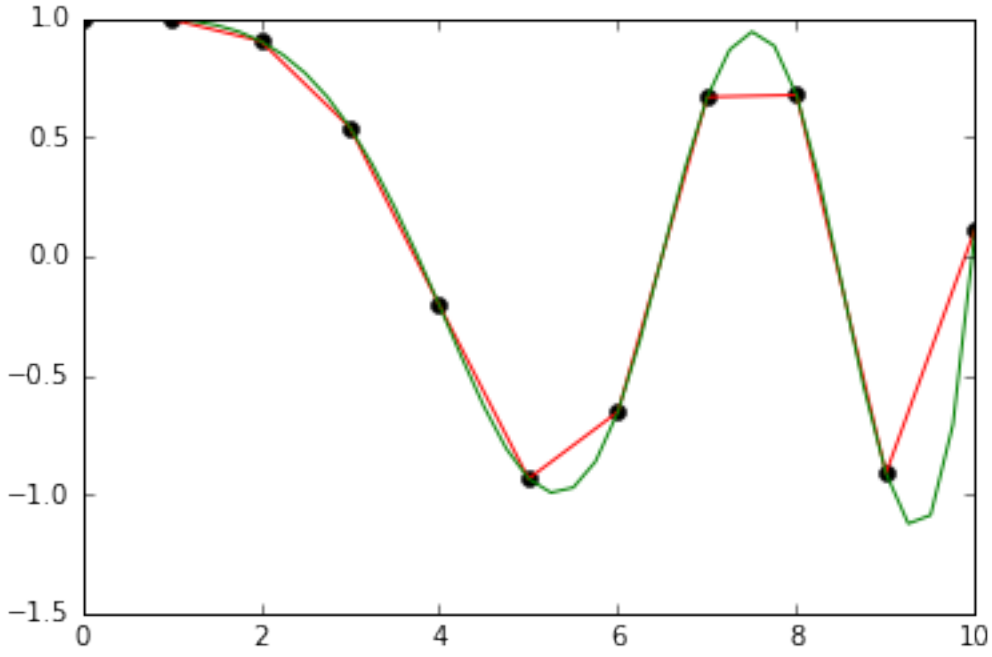
```
In [352]: from scipy.interpolate import interp1d

          t = np.linspace(0, 10, 11)
          cos = np.cos(-t ** 2 / 9.0)


          plt.plot(t, cos, 'ko')
```

```
Out[352]: [<matplotlib.lines.Line2D at 0x110bbcb90>]
```



In [354]:
```
interp_func = interp1d(t, cos)
interp_cubic = interp1d(t, cos, kind='cubic')
t_full = np.linspace(0, 10, 41)
plt.plot(t, cos, 'ko')
plt.plot(t_full, interp_func(t_full), 'r')
plt.plot(t_full, interp_cubic(t_full), 'g')
```

Out[354]: [<matplotlib.lines.Line2D at 0x110673b90>]



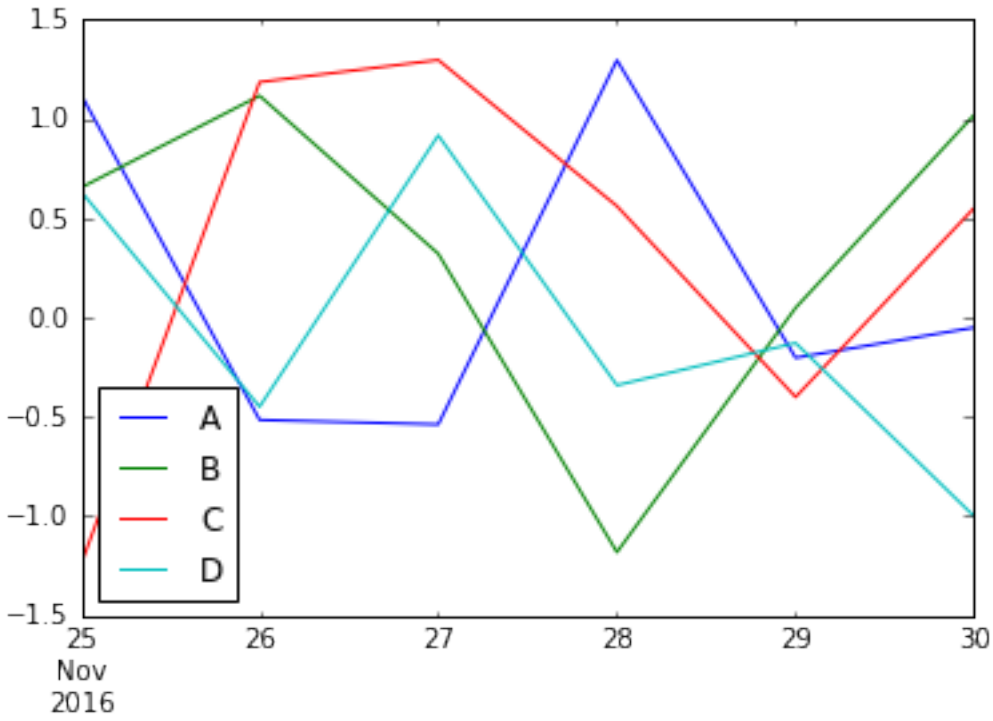
Estatística e muito mais

14 Pandas

http://pandas.pydata.org/

14.0.1 Criando meus dados

```
In [372]: import pandas

          series = pandas.Series([1,3,5,np.nan,6,8])
          series

Out[372]: 0    1.0
          1    3.0
          2    5.0
          3    NaN
          4    6.0
          5    8.0
          dtype: float64

In [371]: dates = pandas.date_range('20161125', periods=6)
          dates

Out[371]: DatetimeIndex(['2016-11-25', '2016-11-26', '2016-11-27', '2016-11-28',
                         '2016-11-29', '2016-11-30'],
                        dtype='datetime64[ns]', freq='D')

In [373]: data_frame = pandas.DataFrame(np.random.randn(6,4),
                                  index=dates, columns=list('ABCD'))
          data_frame

          data_frame.plot()

Out[373]: <matplotlib.axes._subplots.AxesSubplot at 0x1116cd350>
```




```
In [359]: lung_mechanics = pandas.read_csv('dados/LungMech_3.csv', sep=";")
          lung_mechanics.head()
```

Out[359]:

	ANIMAL	GROUP	TIME	Bemerkungen	start-time	\
0	381107	1	1	bas_er_5:48-5:53	20929.688	
1	381107	1	2	ppc_8:10-8:15	29418.728	
2	381107	1	3	ran_9:32-9:37	34354.728	
3	381107	1	4	ven00_10:37-10:42 data till signal volume	38241.288	
4	381107	1	5	ven04_14:29-14:34 data till signal volume	52181.048	

	end-time	duration	nr cycles	VT	lung_mechanics_RP	...	\
0	21229.688	300.0	150	248.571823	2.000268	...	
1	29718.728	300.0	149	248.584459	2.000324	...	
2	34654.728	300.0	149	248.745619	2.000270	...	
3	38541.288	300.0	149	168.818356	2.000324	...	
4	52481.048	300.0	174	218.820910	1.712277	...	

	VT/kg	RR	MV	C	deltaP (VT/C)	Flow(deltaP/R)	\
0	8.018446	30.0	7.457155	29.188826	8.515993	NaN	
1	8.018854	29.8	7.407817	22.215990	11.189439	NaN	
2	8.024052	29.8	7.412619	21.210618	11.727410	NaN	
3	5.445753	29.8	5.030787	15.868591	10.638522	NaN	
4	7.058739	34.8	7.614968	20.230729	10.816264	NaN	

	Energy	power	Unnamed: 57	Unnamed: 58
0	2116.835803	6.227720	NaN	NaN
1	2781.520533	8.128665	NaN	NaN

```
2  2917.141903  8.525002          NaN          NaN
3  1795.977757  5.248533          NaN          NaN
4  2366.824790  8.077297          NaN          NaN

[5 rows x 59 columns]
```

In [374]:
```
lung_mechanics.describe()
```

Out[374]:

	ANIMAL	GROUP	TIME	start-time	end-time
count	240.000000	240.000000	240.000000	2.400000e+02	240.000000
mean	381123.833333	1.916667	5.500000	1.892376e+06	29059.547945
std	10.476779	0.813930	2.878284	2.857085e+07	35815.494473
min	381107.000000	1.000000	1.000000	5.634143e+02	0.390000
25%	381115.750000	1.000000	3.000000	1.915905e+04	13.717500
50%	381121.500000	2.000000	5.500000	3.970691e+04	13276.736000
75%	381132.250000	3.000000	8.000000	7.578188e+04	54426.580574
max	381143.000000	3.000000	10.000000	4.426656e+08	126464.976000

	duration	nr cycles	VT	lung_mechanics_RP
count	240.000000	240.000000	240.000000	240.000000
mean	294.061476	187.995833	230.628440	1.950200
std	38.897026	113.724158	50.242249	0.760423
min	5.672000	6.000000	109.627816	0.606880
25%	300.000000	124.000000	194.839687	1.712218
50%	300.000000	137.000000	226.906422	2.072479
75%	300.000000	175.000000	267.197512	2.400489
max	300.000000	493.000000	368.366875	4.000662

	lung_mechanics_TiTtot	...	VT/kg	RR
count	240.000000	...	240.000000	240.000000
mean	546.492357	...	6.366077	38.624429
std	5969.937082	...	1.219081	22.536272
min	0.161403	...	3.243426	14.800000
25%	0.243810	...	5.810048	24.800000
50%	0.343741	...	5.998466	28.600000
75%	0.493400	...	7.798167	35.000000
max	65535.000000	...	8.702682	98.600000

	MV	C	deltaP (VT/C)	Flow(deltaP/R)	Energy
count	240.000000	240.000000	240.000000	0.0	240.000000
mean	8.400240	21.936341	11.600772	NaN	2778.858276
std	5.031314	7.410927	4.010621	NaN	1298.941126
min	2.930630	9.504305	3.344865	NaN	381.561004
25%	6.098110	16.328140	8.511618	NaN	1708.870528
50%	6.988701	20.219184	11.615712	NaN	2776.173029
75%	8.279947	25.959970	14.630645	NaN	3703.207508
max	66.665552	47.488601	20.726933	NaN	6803.475748

	power	Unnamed: 57	Unnamed: 58
count	240.000000	1.0	1.000000
mean	8.637010	0.0	109381.315268
std	3.137008	NaN	NaN
min	2.492335	0.0	109381.315268
25%	6.641065	0.0	109381.315268
50%	8.335968	0.0	109381.315268

```
          75%     10.220459          0.0  109381.315268
          max     26.902421          0.0  109381.315268

          [8 rows x 58 columns]
```

```
In [378]: lung_mechanics['VT'].head()
          lung_mechanics['VT'].tail()
```

```
Out[378]: 235    245.695793
          236    246.094132
          237    245.705150
          238    245.191123
          239    246.012704
          Name: VT, dtype: float64
```

```
In [384]: lung_mechanics[['VT', 'VT/kg']].head()
```

```
Out[384]:            VT     VT/kg
          0  248.571823  8.018446
          1  248.584459  8.018854
          2  248.745619  8.024052
          3  168.818356  5.445753
          4  218.820910  7.058739
```

14.0.2 Ler dados que estão na web

```
In [385]: user = pandas.read_csv(
              'http://files.grouplens.org/datasets/movielens/ml-100k/u.user')
```

```
In [386]: user.head()
```

```
Out[386]:    1|24|M|technician|85711
          0       2|53|F|other|94043
          1      3|23|M|writer|32067
          2  4|24|M|technician|43537
          3       5|33|F|other|15213
          4   6|42|M|executive|98101
```

```
In [387]: col_names = ['user_id', 'age', 'sex',
                       'occupation', 'zip-code']
```

```
In [390]: user = pandas.read_csv(
              'http://files.grouplens.org/datasets/movielens/ml-100k/u.user',
          names=col_names, sep="|")
```

```
In [392]: user.head()
```

```
Out[392]:    user_id  age sex  occupation zip-code
          0        1   24   M  technician    85711
          1        2   53   F       other    94043
          2        3   23   M      writer    32067
          3        4   24   M  technician    43537
          4        5   33   F       other    15213
```

14.0.3 Indexação Lógica

```
In [410]: user.age[user.occupation == 'writer'].head()
          user.age[user.occupation == 'writer'][:5]

Out[410]: 2     23
          20    26
          21    25
          27    32
          49    21
          Name: age, dtype: int64

In [407]: user.occupation[user['sex'] == 'M'].head(10) #Describe

Out[407]: 0        technician
          2            writer
          3        technician
          5         executive
          6     administrator
          7     administrator
          8           student
          9            lawyer
          12         educator
          13        scientist
          Name: occupation, dtype: object

In [409]: user.occupation[user['sex'] == 'F'].tail(10)

Out[409]: 913           other
          916         student
          919          artist
          920         student
          921    administrator
          924         salesman
          929        scientist
          937       technician
          938          student
          941        librarian
          Name: occupation, dtype: object
```

14.0.4 Exercícios

Mostrar usuários com 40 anos que sejam homens

Mostrar a idade média de mulheres programadoras

15 Desafios

Abrir o arquivo decay.txt (não podem usar bibliotecas) plotar em função do tempo

De posse dos dados do arquivo decay.txt descobrir qual valor associado ao tempo após uma constante de tempo (tau), a amplitude do inicio até tau e de tau até o final

Plotar o (com subplot 2 x 1) sinal e na tela inferior o sinal ajustado (bolinhas pretas e ajuste vermelho)

Marcar no gráfico da função (no eixo x e y) onde ocorreu o tau

```
In [ ]:
```