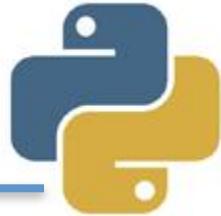


Dados com Numpy e Gráficos com Matplotlib



Márcio Palheta, M.Sc.
marcio.palheta@gmail.com

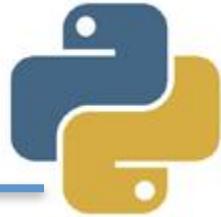


Apresentação

- Programador desde 2000
- Professor de programação desde 2009
- Mestre pelo ICOMP/UFAM – 2013
- Fundador da Buritech – 2014
- Doutorando pelo ICOMP/UFAM
- Pesquisador das áreas: Banco de Dados, Recuperação da Informação, Big Data, Mineração de dados e Aprendizado de Máquina

<https://sites.google.com/site/marciopalheta>

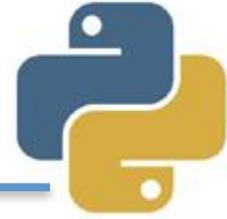




Agenda

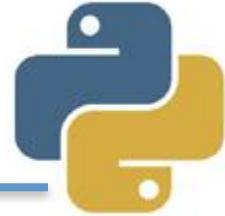
- Relembrando o trabalho com listas
 - De volta às matrizes
 - Instalação e configuração do numpy
 - Representação de dados com numpy
 - Hora dos gráficos
 - Entendendo o matplotlib
-

Revendo o trabalho com listas

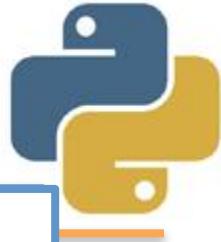


- Listas são **estruturas** muito simples para **armazenamento** de dados
 - Permite **acesso sequencial** ao dados
 - Fácil acesso e **manipulação** de **subconjuntos**
 - Um grande número de **funções PRONTAS**
 - Agora, vamos ao **jupyter notebook**
-

Exercícios de revisão



- Crie funções que recebam uma lista e:
 - devolva o maior elemento
 - devolva o menor elemento
 - devolva o seu somatório
 - Um escalar X e devolva a frequência de X na lista
 - devolva a lista invertida
-

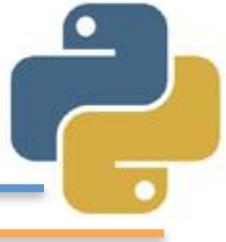


Exercício 1

Criando listas

```
# -*- coding: UTF-8 -*-
# Definição e inicialização
lista = [10, 20, 30, 40, 50, 30, 70, 80]
print type(lista), lista
lista2 = range(10)
print 'De 0 a 10', lista2
lista3 = range(3,10)
print 'De 3 a 10', lista3
lista4 = range(3,10, 2)
print 'De 3 a 10, passo=2', lista4
lista5 = [x*2 for x in range(8)]
print 'list comprehensions', lista5

<type 'list'> [10, 20, 30, 40, 50, 30, 70, 80]
De 0 a 10 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
De 3 a 10 [3, 4, 5, 6, 7, 8, 9]
De 3 a 10, passo=2 [3, 5, 7, 9]
list comprehensions [0, 2, 4, 6, 8, 10, 12, 14]
```



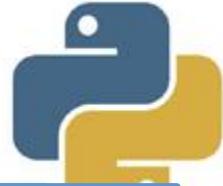
Exercício 2

Percorrendo listas

```
lista = [50, 70, 80, 40, 10, 30, 20, 30]
# Percorrer lista obtendo índice e valor
for indice, valor in enumerate(lista[:3]):
    print "lista[%d]=%d" % (indice, valor),
print '\n',lista
print 'lista invertida com passo do slicing = -1'
print [x for x in lista[::-1]]
print 'lista invertida com reversed()'
print [x for x in reversed(lista)]
print 'lista ordenada\n', [x for x in sorted(lista)]
```

```
lista[0]=50 lista[1]=70 lista[2]=80
[50, 70, 80, 40, 10, 30, 20, 30]
lista invertida com passo do slicing = -1
[30, 20, 30, 10, 40, 80, 70, 50]
lista invertida com reversed()
[30, 20, 30, 10, 40, 80, 70, 50]
lista ordenada
[10, 20, 30, 30, 40, 50, 70, 80]
```

Exercício 3



Trabalhando com referências

```
lista = [10, 20, 30, 40, 50, 60, 70, 80]

# Enviando cópia da referência
lista_ref = lista

print 'Antes da alteração\n', lista, '\n', lista_ref
lista_ref[0] = 300
print 'Depois da alteração\n', lista, '\n', lista_ref
```

Antes da alteração

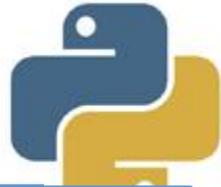
```
[10, 20, 30, 40, 50, 60, 70, 80]
```

```
[10, 20, 30, 40, 50, 60, 70, 80]
```

Depois da alteração

```
[300, 20, 30, 40, 50, 60, 70, 80]
```

```
[300, 20, 30, 40, 50, 60, 70, 80]
```



Exercício 4

Trabalhando com valores

```
lista = [10, 20, 30, 40, 50, 60, 70, 80]

# Enviando cópia dos dados
lista_copia = lista[:]

print 'Antes da alteração\n', lista, '\n', lista_copia
lista_copia[0] = 300
print 'Depois da alteração\n', lista, '\n', lista_copia
```

Antes da alteração

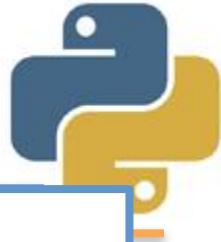
```
[10, 20, 30, 40, 50, 60, 70, 80]
```

```
[10, 20, 30, 40, 50, 60, 70, 80]
```

Depois da alteração

```
[10, 20, 30, 40, 50, 60, 70, 80]
```

```
[300, 20, 30, 40, 50, 60, 70, 80]
```

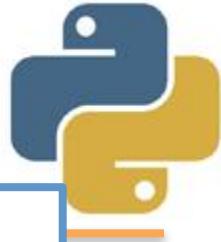


Exercício 5

Listas aleatórias

```
import random
lista = [50, 70, 80, 40, 10, 30, 20, 30]
print('lista original\n', lista)
random.shuffle(lista)
print('lista embaralhada\n', lista)
print('Pegando um número aleatório da lista:', random.choice(lista))
print('Amostra aleatória de 4 elementos')
print(random.sample(lista, 4))
print('Lista gerada com 10 números aleatórios(entre 0 e 100)')
print(random.sample(range(0, 100), 10))
```

```
lista original
[50, 70, 80, 40, 10, 30, 20, 30]
lista embaralhada
[40, 70, 10, 30, 80, 20, 50, 30]
Pegando um número aleatório da lista: 20
Amostra aleatória de 4 elementos
[30, 80, 40, 20]
Lista gerada com 10 números aleatórios(entre 0 e 100)
[66, 68, 37, 21, 17, 85, 11, 95, 25, 4]
```



Exercício 6

Slicing em listas

```
lista = [50, 70, 80, 40, 10, 30, 20, 30]
print 'Coordenada 1:', lista[1]
print 'Elementos dos índices de 1 a 4:', lista[1:5]
print 'Elementos até o índice 4:', lista[:5]
print 'Elementos a partir do índice 5:', lista[5:]
print '5 últimos elementos', lista[-5:]
print 'Começando no índice 1, de 2 em 2:', lista[1::2]
print 'Elementos de índice par', lista[::2]
print 'Elementos de índice ímpar', lista[1::2]
```

Coordenada 1: 70

Elementos dos índices de 1 a 4: [70, 80, 40, 10]

Elementos até o índice 4: [50, 70, 80, 40, 10]

Elementos a partir do índice 5: [30, 20, 30]

5 últimos elementos [40, 10, 30, 20, 30]

Começando no índice 1, de 2 em 2: [70, 40, 30, 30]

Elementos de índice par [50, 80, 10, 20]

Elementos de índice ímpar [70, 40, 30, 30]



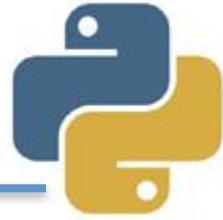
Exercício 7

Listas e funções

```
lista = [50, 70, 80, 40, 10, 30, 20, 30]
print 'somatório:', sum(lista)
print 'frequência de 30:', lista.count(30)
print 'Maior elemento:', max(lista)
print 'Menor elemento:', min(lista)
print 'Juntando duas listas e formando pares de elementos:'
lista = zip(range(0, 5), range(5, 10))
print lista
print 'Separando os elementos de uma lista de forma intercalada:'
lista = range(0, 10)
intercaladas = lista[::2], lista[1::2]
print lista, '\n', intercaladas

print 'Transformando uma lista de strings em uma string CSV:'
lista = ["Curso", "de", "data", "sciense", "em", 'python']
csv_values = ';'.join(lista)
print csv_values
```

Resultado no notebook



```
print 'Transformando uma lista de strings em uma string CSV:'  
lista = ["Curso", "de", "data", "sciense", "em", 'python']  
csv_values = ';' .join(lista)  
print csv_values
```

somatório: 330

frequência de 30: 2

Maior elemento: 80

Menor elemento: 10

Juntando duas listas e formando pares de elementos:

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

Separando os elementos de uma lista de forma intercalada:

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

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

Transformando uma lista de strings em uma string CSV:

```
Curso;de;data;sciense;em;python
```



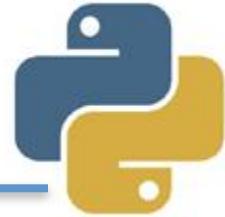
Exercício 8

Aplicando funções a elementos da lista

```
def negativo(valor):
    return valor * -1

lista = range(11)
print 'lista:', lista
print 'dupli:', lista * 2
print 'Lista de nros pares ao quadrado',[x*2 for x in lista if x % 2 == 0]
print 'Gerando uma lista de LISTAS'
print '\t',[[s.capitalize(), s.upper(), len(s)] for s in ['um', 'dois', 'tres']]
print 'Gerando uma lista de TUPLAS'
print '\t',[((s.capitalize(), s.upper(), len(s))) for s in ['um', 'dois', 'tres']]
print 'função:', map(negativo, lista)
print 'lambda:', map(lambda x: x*3, lista)
print 'Filtrando os elementos de uma lista, de acordo com um critério:'
def criterio(x):
    return x >= 0
lista = range(-5, 5)
print 'lista sem filtro, ', lista
print 'lista COM filtro, ', filter(criterio, lista)
```

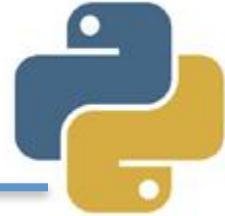
Resultado no console



```
lista = range(-5, 5)
print 'lista sem filtro, ', lista
print 'lista COM filtro, ', filter(criterio, lista)
```

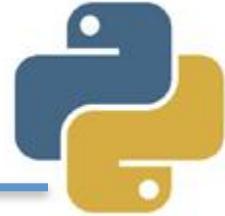
```
lista: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
dupli: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Lista de nros pares ao quadrado [0, 4, 8, 12, 16, 20]
Gerando uma lista de listas
[[['Um', 'UM', 2], ['Dois', 'DOIS', 4], ['Tres', 'TRES', 4]]]
Gerando uma lista de TUPLAS
[('Um', 'UM', 2), ('Dois', 'DOIS', 4), ('Tres', 'TRES', 4)]
função: [0, -1, -2, -3, -4, -5, -6, -7, -8, -9, -10]
lambda: [0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30]
Filtrando os elementos de uma lista, de acordo com um critério:
lista sem filtro, [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4]
lista COM filtro, [0, 1, 2, 3, 4]
```

Exercício 9



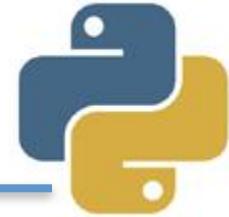
- Crie funções que recebam uma matriz e:
 - devolva o maior elemento
 - devolva o menor elemento
 - devolva o seu somatório
 - Um escalar X e devolva a frequência de X na matriz
-

Exercício 10



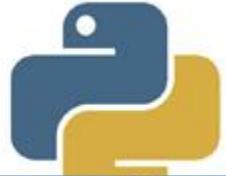
- Crie funções que recebam duas matrizes A e B e um escalar X:
 - Devolva a matriz resultante de $A + B$
 - Devolva a matriz resultante de $A * X$
 - Devolva a transposta de A
 - Devolva uma lista com a 1^a coluna de A
-

Numpy



- Biblioteca para **data science** em python
 - Tem como principal **objetivo**, otimizar o uso de **arrays** multidimensionais
 - Processamento **mais rápido** do que as listas tradicionais
 - Muitas **funções PRONTAS** para uso
-

Exercício 11: Numpy



```
In [13]: # -*- coding: UTF-8 -*-
import numpy as np
```

```
lista = [10, 20, 30, 40, 50, 60]
np_array = np.array(lista)
type(np_array), array
```

```
Out[13]: (numpy.ndarray, array([10, 20, 30, 40, 50, 60]))
```

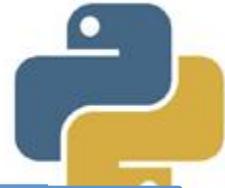
```
In [11]: sum(np_array)
```

```
Out[11]: 210
```

```
In [12]: np_array[2:5]
```

```
Out[12]: array([30, 40, 50])
```

Exercício 12: matrizes



```
mat = numpy.array([[1,2, 3], [4, 5, 6], [7, 8, 9]])  
print type(mat), '\n', mat
```

```
<type 'numpy.ndarray'>  
[[1 2 3]  
 [4 5 6]  
 [7 8 9]]
```

```
# Acessando uma linha e uma célula da matriz  
print 'mat[1]:', mat[1], ' - mat[1][1]:', mat[1][1]
```

```
mat[1]: [4 5 6] - mat[1][1]: 5
```

```
# Acessando uma coluna  
print 'mat[:,0]:', mat[:, 0]
```

```
mat[:,0]: [1 4 7]
```

Exercício 13: Transposta



```
mat = numpy.array([[1,2, 3], [4, 5, 6], [7, 8, 9]])
print mat
# Matriz transposta - Linha vira coluna
print 'Transposta:\n', mat.transpose()
```

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

Transposta:

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



Exercício 14: operações

```
In [42]: m1 = numpy.array([[1, 2], [3, 4]])
print m1
m2 = numpy.array([[5, 6], [7, 8]])
print m2
m1 + m2
```

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

```
Out[42]: array([[ 6,  8],
 [10, 12]])
```

```
In [41]: m1 - m2
```

```
Out[41]: array([[-4, -4],
 [-4, -4]])
```

Exercício 15: funções



```
In [48]: # Soma dos valores do vetor  
m3 = numpy.array([3, 5, 8, 4])  
m3.sum()
```

Out[48]: 20

```
In [49]: # Indice do maior elemento  
m3.argmax()
```

Out[49]: 2

```
In [50]: # Indice do menor elemento  
m3.argmin()
```

Out[50]: 0



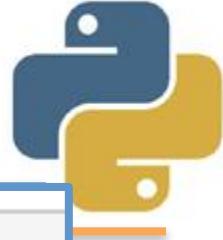
Exercício 16: mais funções

In [53]:

```
import numpy as np
matrix = np.array([[1,2, 3], [4, 5, 6], [7, 8, 9]])
print matrix
# media dos elementos
print 'matrix.mean():',matrix.mean()
# diagonal
print 'matrix.diagonal():',matrix.diagonal()
# dimensao
print 'matrix.ndim:',matrix.ndim
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
matrix.mean(): 5.0
matrix.diagonal(): [1 5 9]
matrix.ndim: 2
```

Exercício 17: slicing



```
In [58]: import numpy as np  
  
lista = [80, 20, 40, 50, 90]  
a = np.array(lista)  
a[1:5]
```

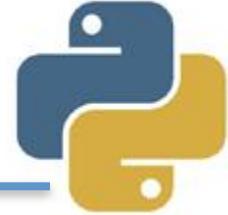
```
Out[58]: array([20, 40, 50, 90])
```

```
In [60]: a[::-2]
```

```
Out[60]: array([80, 40, 90])
```

```
In [64]: a[-3:]
```

```
Out[64]: array([40, 50, 90])
```

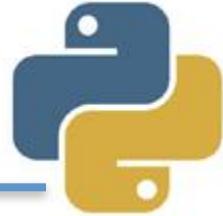


Exercício 18: por referência

```
In [72]: lista = [80, 20, 40, 50, 90]
a = np.array(lista)
b = a[:]
print 'array a\n',a
print 'array b\n',b
print '**** Alterando b[0] = 100'
b[0] = 100
print 'array a\n',a
print 'array b\n',b
```

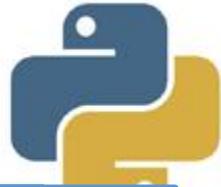
```
array a
[80 20 40 50 90]
array b
[80 20 40 50 90]
**** Alterando b[0] = 100
array a
[100 20 40 50 90]
array b
[100 20 40 50 90]
```

Exercício 19: por valor



```
In [73]: lista = [80, 20, 40, 50, 90]
          a = np.array(lista)
          b = a.copy()
          print 'array a\n',a
          print 'array b\n',b
          print 'Alterando b[0] = 100'
          b[0] = 100
          print 'array a\n',a
          print 'array b\n',b
```

```
array a
[80 20 40 50 90]
array b
[80 20 40 50 90]
Alterando b[0] = 100
array a
[80 20 40 50 90]
array b
[100 20 40 50 90]
```



Exercício 19: dimensões

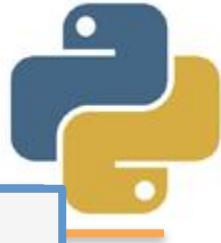
```
In [84]: matriz_base = np.arange(0,15)
print 'Arranjo inicial\n', matriz_base
mat_35 = np.reshape(matriz_base, (3, 5))
print 'Matriz 3x5\n', mat_35
mat_53 = np.reshape(matriz_base, (5, 3))
print 'Matriz 5x3\n', mat_53
```

```
Arranjo inicial
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 ]
Matriz 3x5
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]]
Matriz 5x3
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]
 [12 13 14]]
```



Exercício 20: pontos

```
[90]: import numpy as np  
# Pontos de clientes de um programa de fidelidade  
joao = [20, 30, 40, 15]  
jose = [100, 50, 40, 60]  
maria = [80, 90, 47, 30]  
anna = [22, 40, 30, 27]  
  
pontos = np.array([joao, jose, maria, anna])  
  
print pontos  
print 'pontos[0]:', pontos[0]  
print 'pontos.item(5):', pontos.item(5)  
print 'Pontuação média:', pontos.mean()  
print 'Mediana:', np.median(pontos)  
print 'Maior pontuação:', pontos.max()  
print 'Menor pontuação:', pontos.min()
```

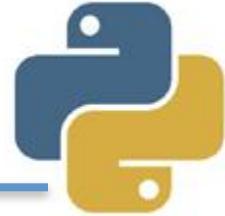


Resultado no notebook

```
print pontos
print 'pontos[0]:', pontos[0]
print 'pontos.item(5):', pontos.item(5)
print 'Pontuação média:', pontos.mean()
print 'Mediana:', np.median(pontos)
print 'Maior pontuação:', pontos.max()
print 'Menor pontuação:', pontos.min()
```

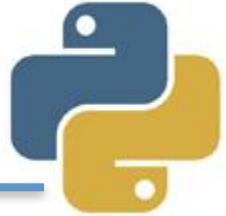
```
[[ 20  30  40  15]
 [100  50  40  60]
 [ 80  90  47  30]
 [ 22  40  30  27]]
pontos[0]: [20 30 40 15]
pontos.item(5): 50
Pontuação média: 45.0625
Mediana: 40.0
Maior pontuação: 100
Menor pontuação: 15
```

Matplotlib



- Biblioteca para **data science** em python, usada para visualização de dados
 - Pacote **muito usado** para a **visualização** de gráficos **2D**
 - **Fácil** de usar, permite que o cientista de dados gere **versões diferentes** de visualização
-

Exercício 21: 1ª Plotagem

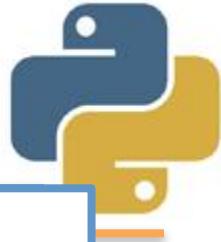


```
# -*- coding: UTF-8 -*-
```

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

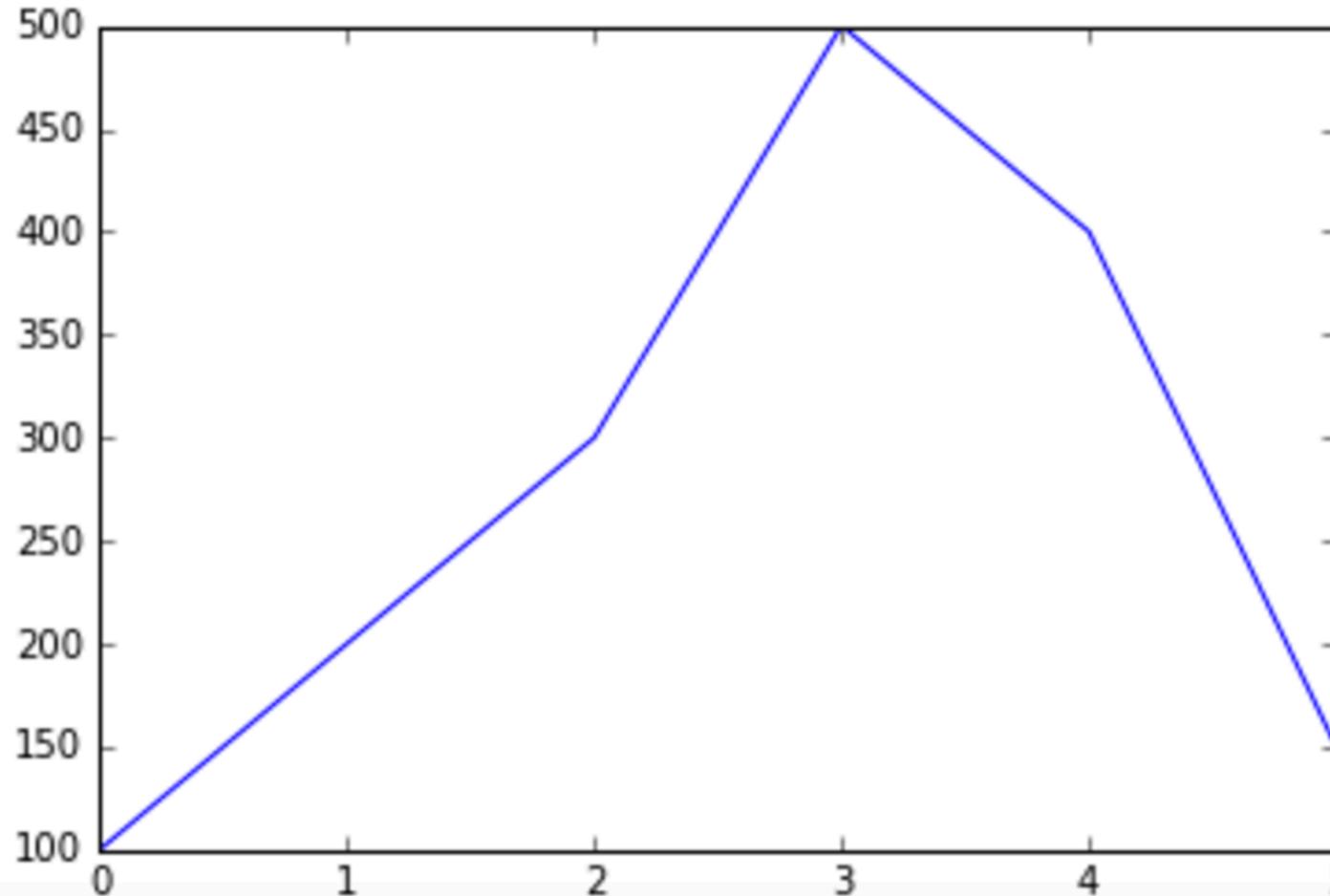
pontuacao = np.array([100, 200, 300, 500, 400, 150])
print(pontuacao)
plt.plot(pontuacao)
```

```
[100 200 300 500 400 150]
```



Resultado no notebook

```
[<matplotlib.lines.Line2D at 0x1167732d0>]
```

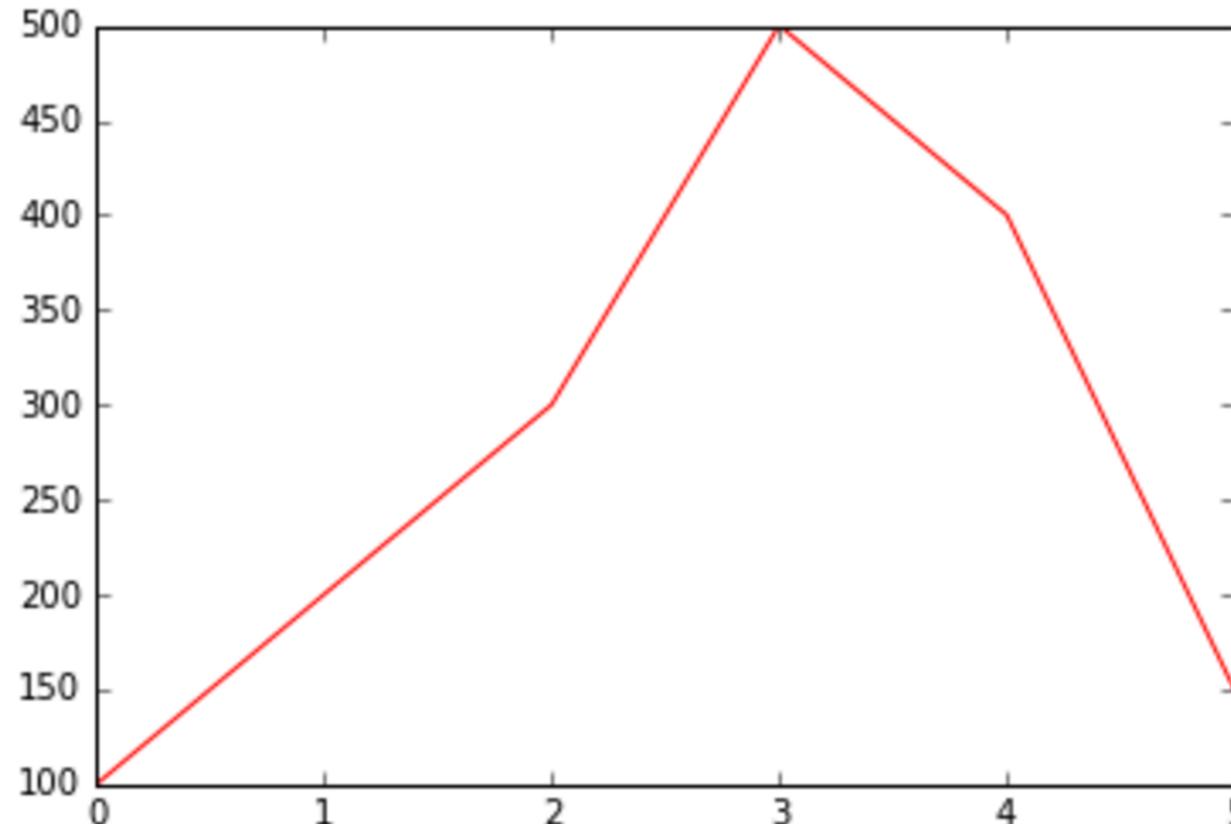


Exercício 22: Cores



```
In [47]: plt.plot(pontuacao, c='red')
```

```
Out[47]: [<matplotlib.lines.Line2D at 0x11668b6d0>]
```

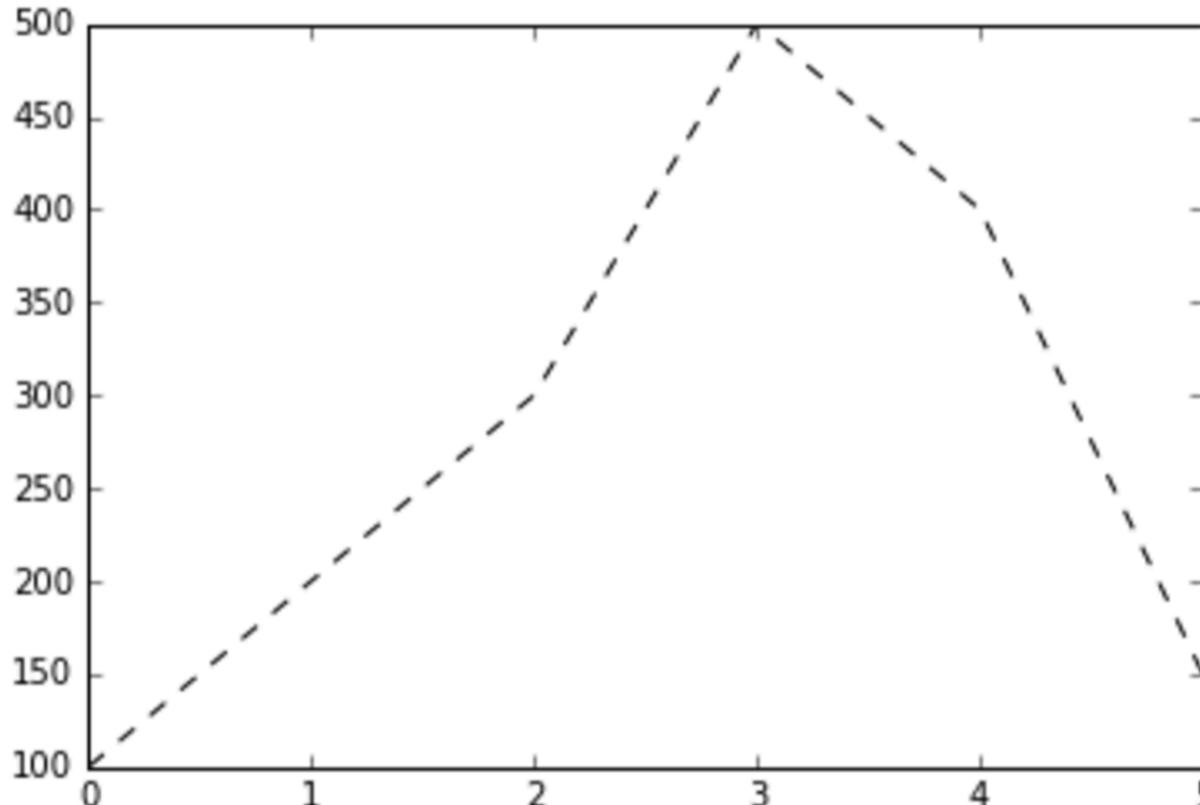


Exercício 23: estilo da linha



```
In [48]: plt.plot(pontuacao, c='black', ls='--')
```

```
Out[48]: [<matplotlib.lines.Line2D at 0x116327e50>]
```

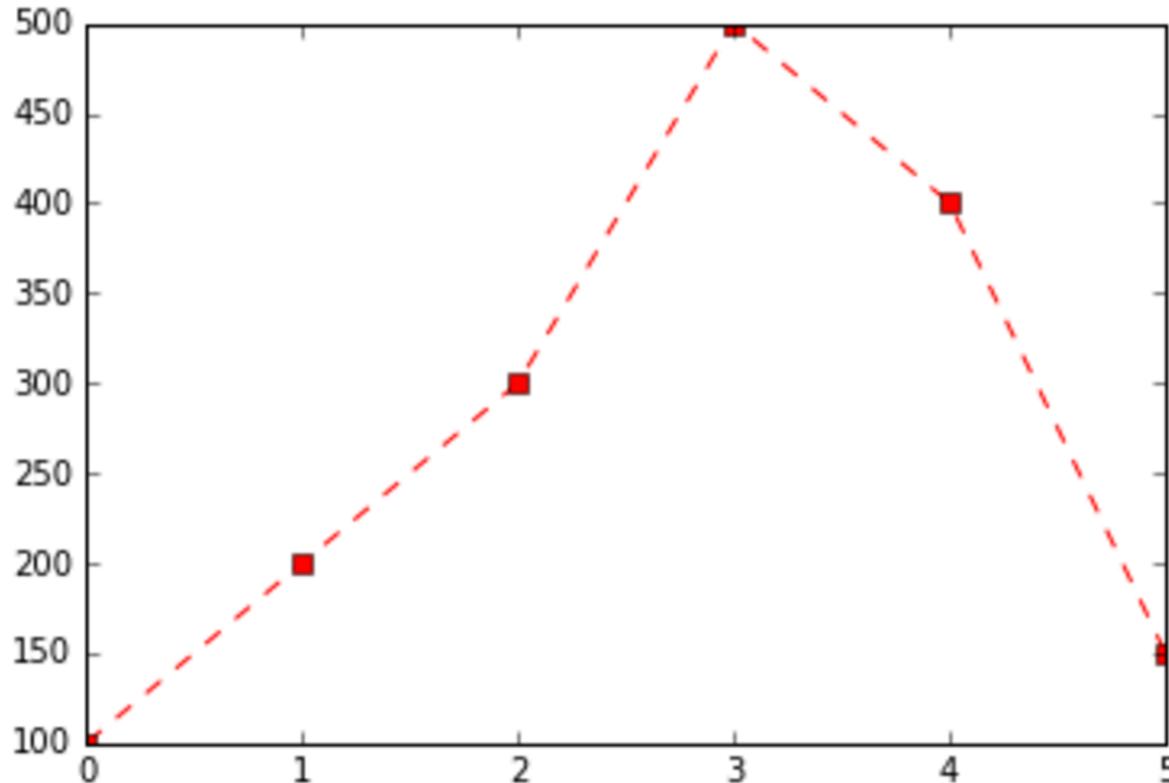


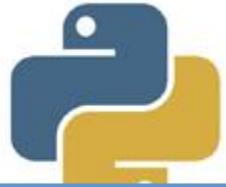


Exercício 24: Marcações

```
In [49]: plt.plot(pontuacao, c='red', ls='--', marker='s')
```

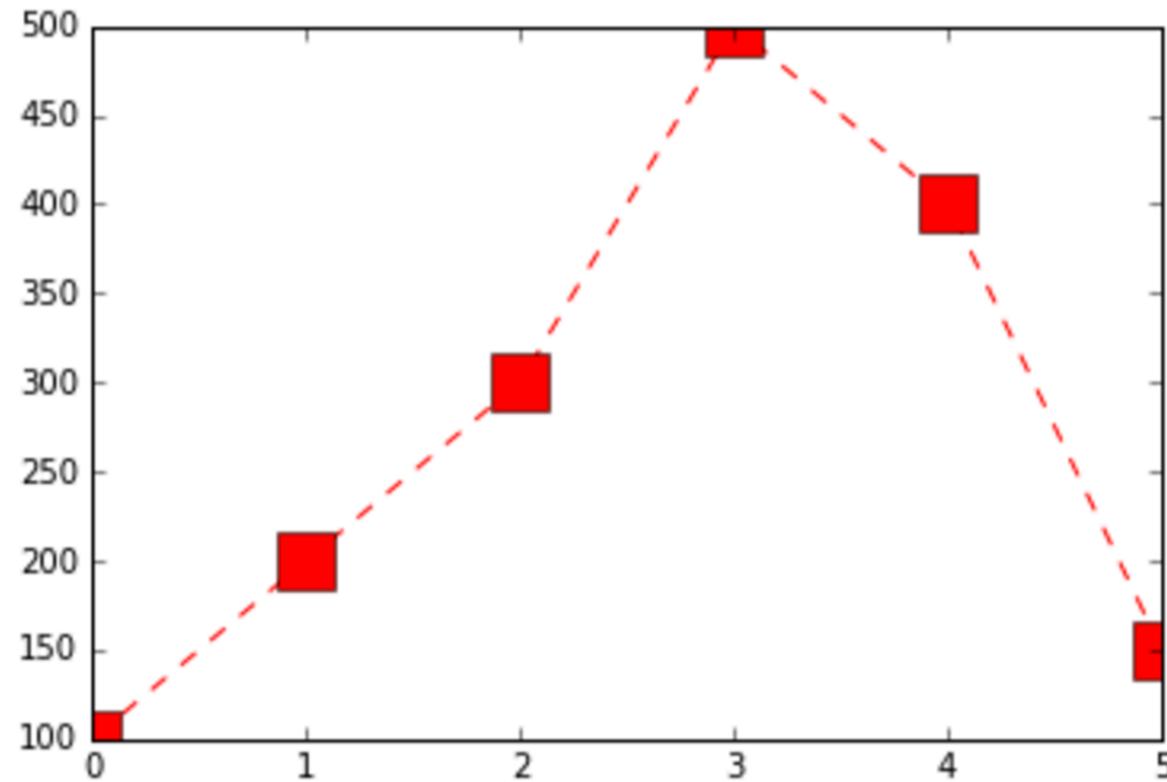
```
Out[49]: [<matplotlib.lines.Line2D at 0x115bcfe10>]
```

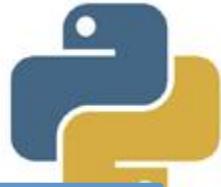




Exercício 25: Pontos maiores

```
In [50]: plt.plot(pontuacao, c='red', ls='--', marker='s', ms='18')  
plt.show()
```

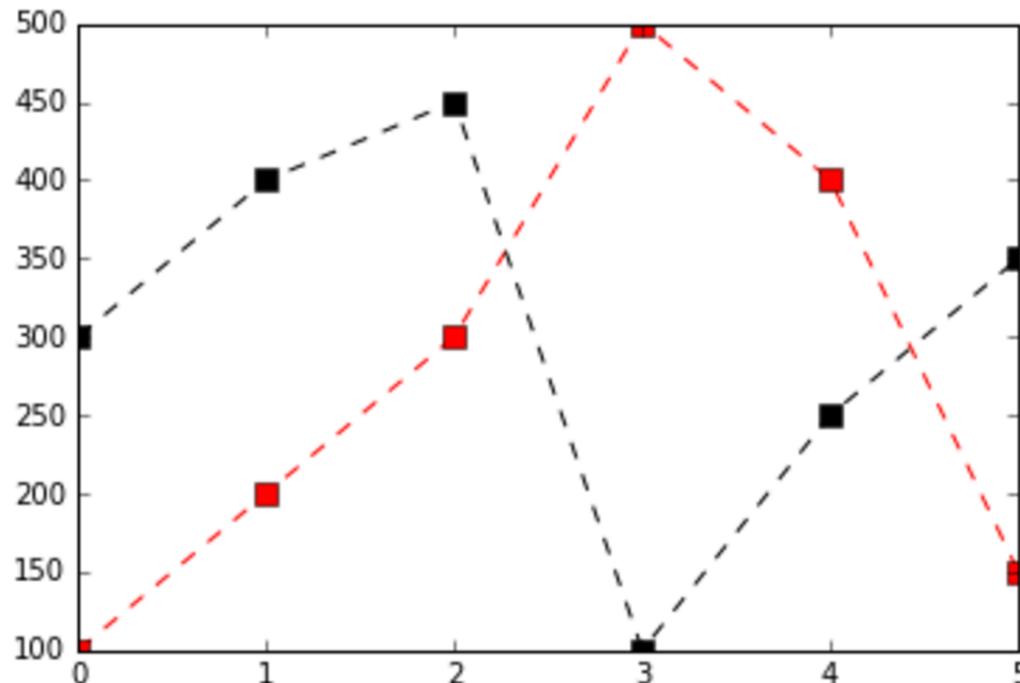


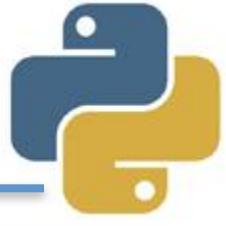


Exercício 26: várias linhas

```
In [51]: pontuacao2 = np.array([300, 400, 450, 100, 250, 350])
plt.plot(pontuacao, c='red', ls='--', marker='s', ms='8')
plt.plot(pontuacao2, c='black', ls='--', marker='s', ms='8')
```

```
Out[51]: [
```





FIM

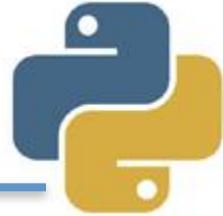


Contatos

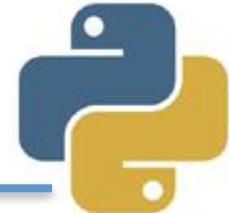


- Márcio Palheta
 - marcio.palheta@gmail.com
 - @marciopalheta
 - <https://sites.google.com/site/marciopalheta/>
-

Bibliografia



- LIVRO: Apress - Beginning Python From Novice to Professional
 - LIVRO: O'Reilly - Learning Python
 - <http://www.python.org>
 - <http://www.python.org.br>
 - Mais exercícios:
 - <http://wiki.python.org.br/ListaDeExercicios>
 - Documentação do python:
 - <https://docs.python.org/2/>
-



Dados com Numpy e Gráficos com Matplotlib



Márcio Palheta, M.Sc.
marcio.palheta@gmail.com