INSTITUTO DE INFORMÁTICA
Universidade Federal de Goiás

Introdução a Python

Altino Dantas

INSTITUTO DE INFORMÁTICA
UFG

UFG
UNIVERSIDADE FEDERAL DE GOIÁS

Objetivos

- Explorar as funcionalidades básicas da biblioteca **matplotlib**;
- Tratar informações adicionais como legenda, rótulos e títulos;
- Conhecer os principais tipos de gráficos;

Características

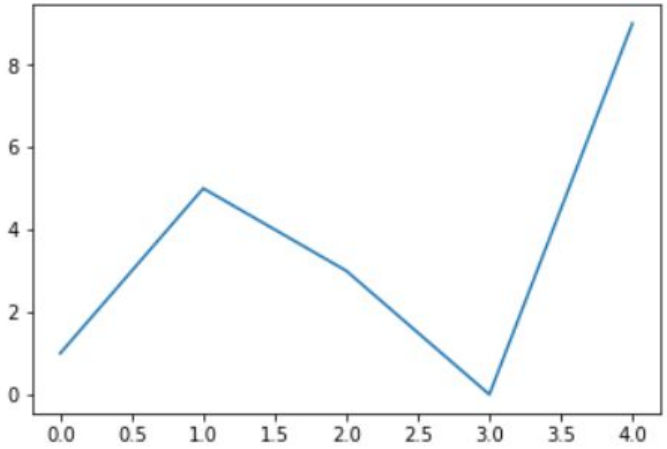
- Fornece uma linguagem de alto nível para construção intuitiva de gráficos;
- Possui diversos formatos de saída para os gráficos;
- “*Matplotlib torna simples tarefas fáceis e possível tarefas difíceis*”;
- Open source;
- Customizável e extensiva;
- Cross-platform.

Primeiros passos

- Uma das vantagens desta biblioteca é a facilidade de gerar gráficos simples;
- Neste caso, os valores do eixo-x são automaticamente definidos:

```
import matplotlib.pyplot as plt
```

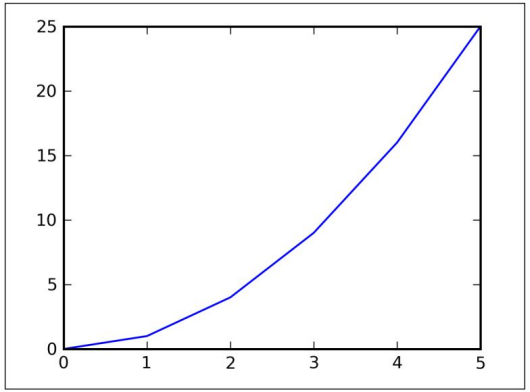
```
plt.plot([1,5,3,0,9])
plt.show()
```



Primeiros passos

- Naturalmente, os valores dos dois eixos podem ser definidas:

```
In [1]: import matplotlib.pyplot as plt
In [2]: x = range(6)
In [3]: plt.plot(x, [xi**2 for xi in x])
Out[3]: [<matplotlib.lines.Line2D object at 0x2408d10>]
In [4]: plt.show()
```

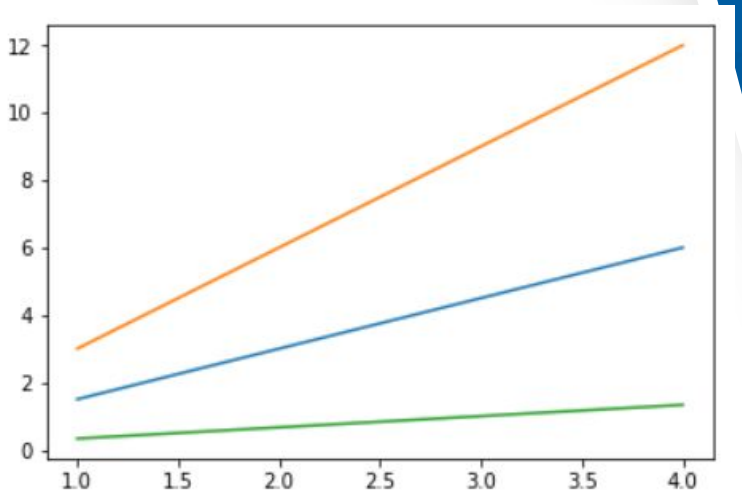


Gráficos com múltiplas linhas

- Outra utilidade seria *plotar* mais de uma linha por gráfico:

```
import matplotlib.pyplot as plt

x = range(1, 5)

plt.plot(x, [xi*1.5 for xi in x])
plt.plot(x, [xi*3.0 for xi in x])
plt.plot(x, [xi/3.0 for xi in x])

plt.show()
```

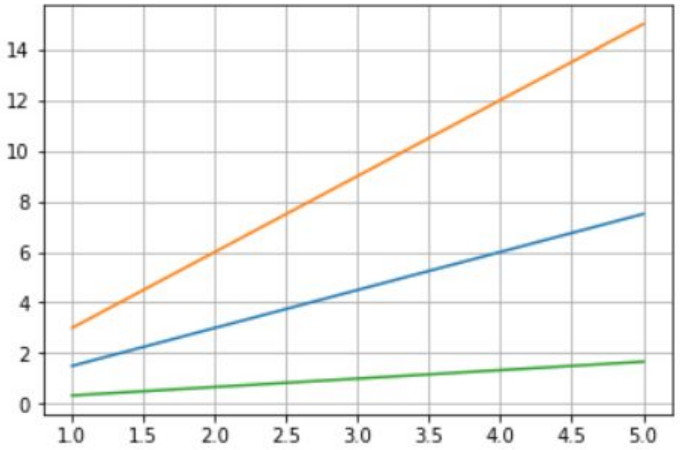
```
import matplotlib.pyplot as plt

x = range(1, 5)

plt.plot(x, [xi*1.5 for xi in x],
         x, [xi*3.0 for xi in x],
         x, [xi/3.0 for xi in x])

plt.show()
```



Grid

- Basta adicionar o método .grid(True) com o parâmetro True;

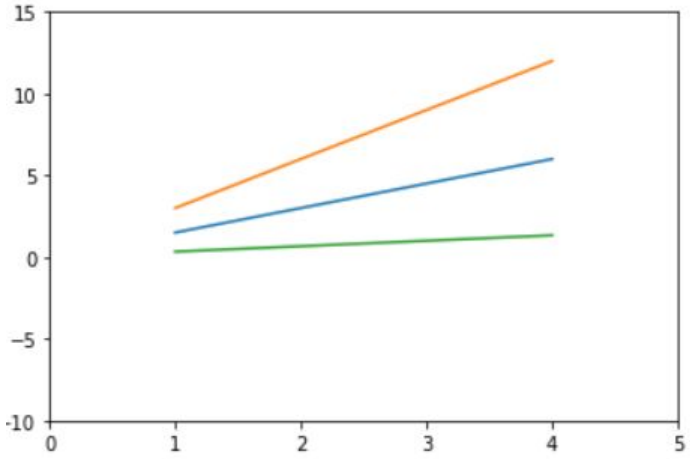
```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(1, 6)

plt.plot(x, [xi*1.5 for xi in x],
         x, [xi*3.0 for xi in x],
         x, [xi/3.0 for xi in x])

plt.grid(True)
plt.show()
```



Eixos

- Por padrão os limites dos eixos são definidos de forma automática, mas podem ser alterados:

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

x = np.arange(1, 5)

plt.plot(x, x*1.5,
         x, x*3.0,
         x, x/3.0)

plt.axis()
plt.axis([0, 5, -10, 15])
plt.show()
```

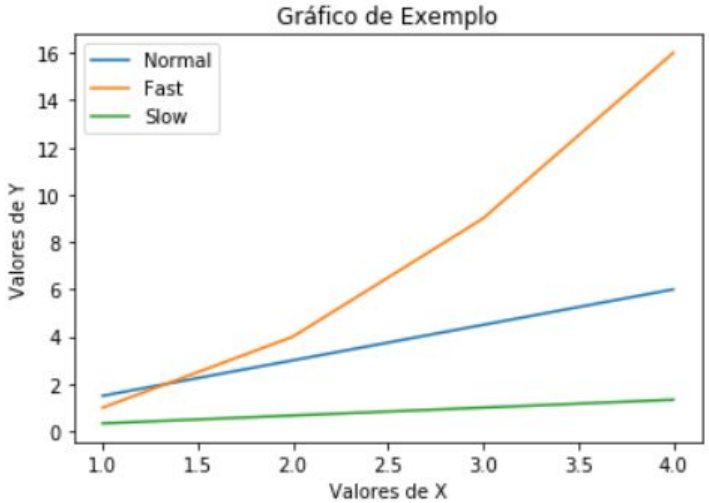


- Outros métodos úteis são: **`xlim([xmin, xmax])`** e **`ylim([ymin,ymax])`.**

Rótulos, Título e Legendas

- Os métodos **.xlabel('texto')** e **.ylabel('texto')** podem ser usados para adicionar rótulos aos eixos;
- **.title('Titulo')** define o título do gráfico;
- **.legend()** habilita uma legenda com *labels* já definidos ou recebe-os por parâmetro

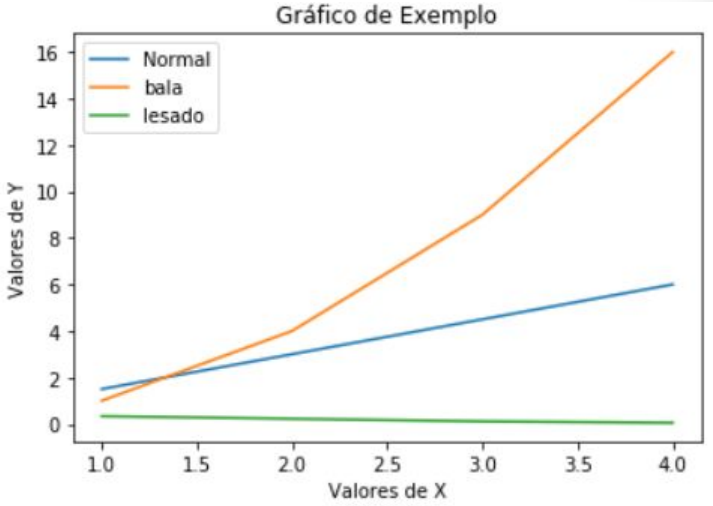
```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(1, 5)

plt.plot(x, x*1.5, label='Normal')
plt.plot(x, x**2, label='Fast')
plt.plot(x, x/3.0, label='Slow')

plt.title('Gráfico de Exemplo') # Define o título
plt.xlabel('Valores de X')      # Define o rótulo do eixo-x
plt.ylabel('Valores de Y')      # Define o rótulo do eixo-y
plt.legend()                    # Habilita a legenda com os "labels"

plt.show()
```



Posicionamento da legenda

- A posição da legenda pode ser definida através do parâmetro loc de **legend**(**loc**=(int[0-10])) ou **legend**(**loc**=(x,y))

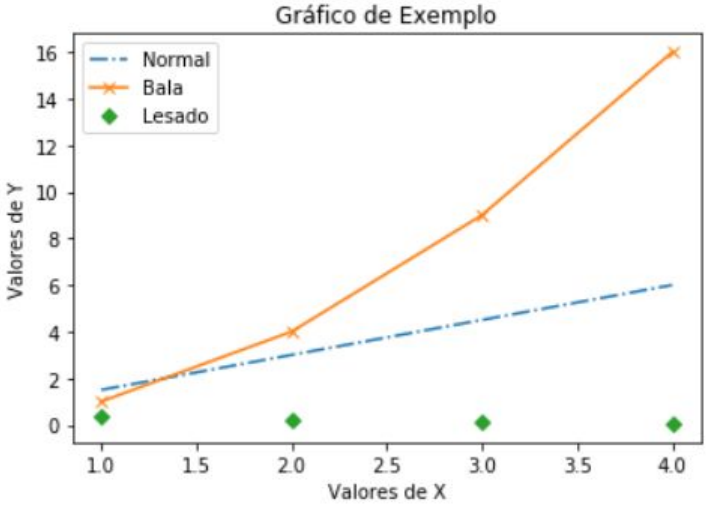
```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(1, 5)

plt.plot(x, x*1.5)
plt.plot(x, x**2)
plt.plot(x, x/3.0**x)

plt.title('Gráfico de Exemplo') # Define o título
plt.xlabel('Valores de X')      # Define o rótulo do eixo-x
plt.ylabel('Valores de Y')      # Define o rótulo do eixo-y
plt.legend(['Normal', 'bala', 'lesado'], loc=(0))

plt.show()
```



Estilo de linhas/marcadores

Style abbreviation	Style
-	solid line
--	dashed line
-.	dash-dot line
:	dotted line

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

x = np.arange(1, 5)

plt.plot(x, x*1.5,'-.')
plt.plot(x, x**2, '-x')
plt.plot(x, x/3.0**x, 'D')

plt.title('Gráfico de Exemplo') # Define o título
plt.xlabel('Valores de X')      # Define o rótulo do eixo-x
plt.ylabel('Valores de Y')      # Define o rótulo do eixo-y
plt.legend(['Normal', 'Bala', 'Lesado'], loc=(0))
```

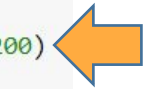


Marker abbreviation	Marker style
v	Triangle down marker
^	Triangle up marker
<	Triangle left marker
>	Triangle right marker
1	Tripod down marker
2	Tripod up marker
3	Tripod left marker
4	Tripod right marker
s	Square marker
p	Pentagon marker
*	Star marker
h	Hexagon marker
H	Rotated hexagon marker
+	Plus marker
x	Cross (x) marker
D	Diamond marker
d	Thin diamond marker
|	Vertical line (vline symbol) marker
_	Horizontal line (hline symbol) marker

Salvando gráficos

- Basta utilizar o método **.savefig()** passando o caminho do arquivo de saída;
- O parâmetro **dpi** pode ser utilizado para definir a resolução da imagem:

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

x = np.arange(1, 5)

plt.plot(x, x*1.5)
plt.plot(x, x**2)
plt.plot(x, x/3.0**x)

plt.title('Gráfico de Exemplo') # Define o título
plt.xlabel('Valores de X')      # Define o rótulo do eixo-x
plt.ylabel('Valores de Y')      # Define o rótulo do eixo-y
plt.legend(['Normal', 'bala', 'lesado'], loc=(0))

plt.savefig('C:\\Users\\Altino\\Desktop\\Notebooks fasam\\fig.png', dpi=200)
```



Tipos de gráficos

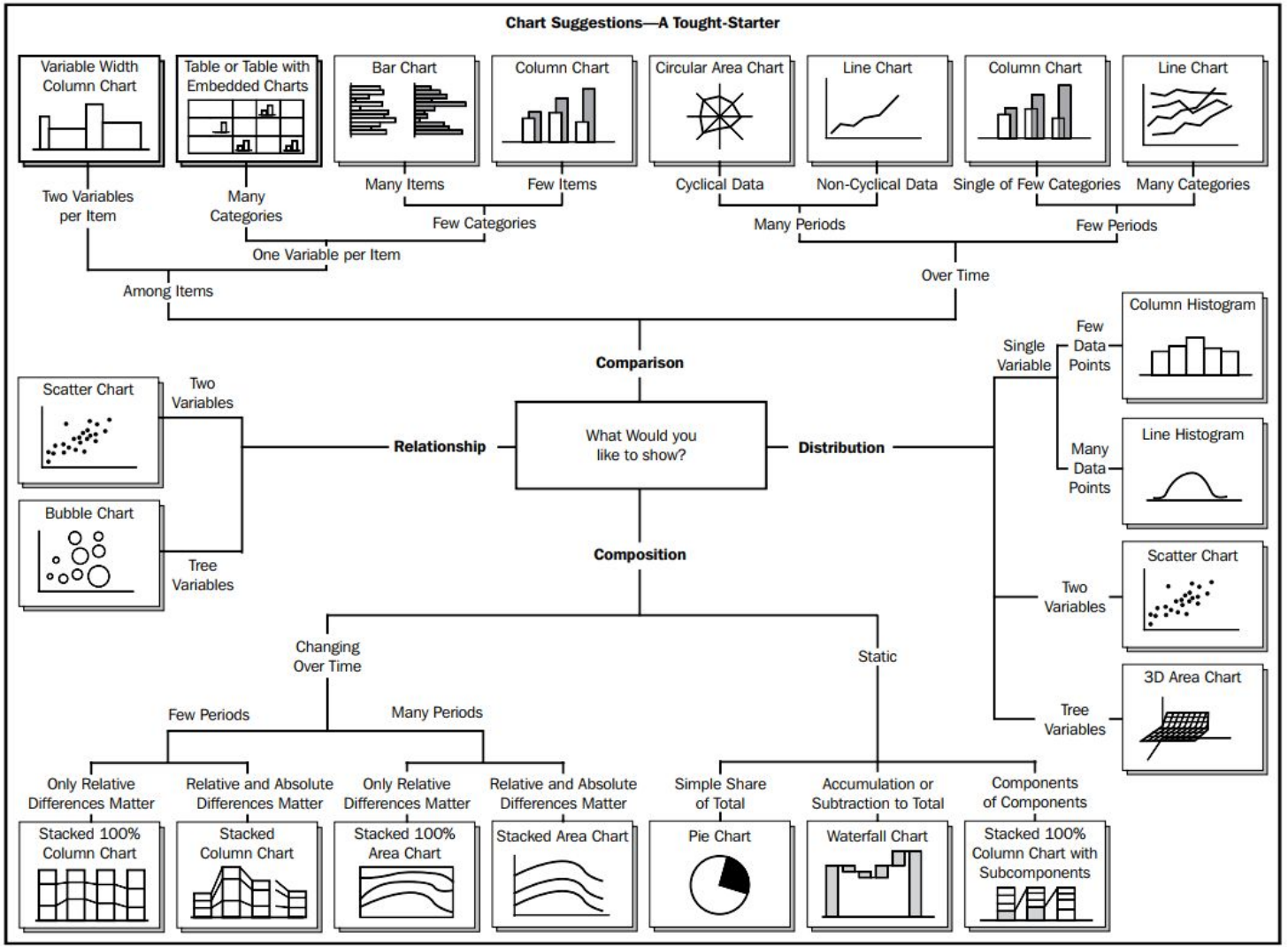


Chart Suggestions—A Tought-Starter
Variable Width Column Chart
Table or Table with Embedded Charts
Bar Chart
Column Chart
Circular Area Chart
Line Chart
Column Chart
Line Chart
Two Variables per Item
Many Categories
Many Items
Few Items
Few Categories
Cyclical Data
Non-Cyclical Data
Many Periods
Single of Few Categories
Many Categories
Few Periods
One Variable per Item
Among Items
Over Time
Comparison
Scatter Chart
Two Variables
Relationship
What Would you like to show?
Distribution
Bubble Chart
Tree Variables
Composition
Single Variable
Few Data Points
Column Histogram
Many Data Points
Line Histogram
Two Variables
Scatter Chart
Tree Variables
3D Area Chart
Changing Over Time
Static
Few Periods
Many Periods
Only Relative Differences Matter
Relative and Absolute Differences Matter
Only Relative Differences Matter
Relative and Absolute Differences Matter
Simple Share of Total
Accumulation or Subtraction to Total
Components of Components
Stacked 100% Column Chart
Stacked Column Chart
Stacked 100% Area Chart
Stacked Area Chart
Pie Chart
Waterfall Chart
Stacked 100% Column Chart with Subcomponents

Histograma

- *Plota* a frequência de dados em x categorias, 10 por padrão;

```
y = np.random.randn(1000)
plt.title('Histograma')
plt.hist(y, 25)
```

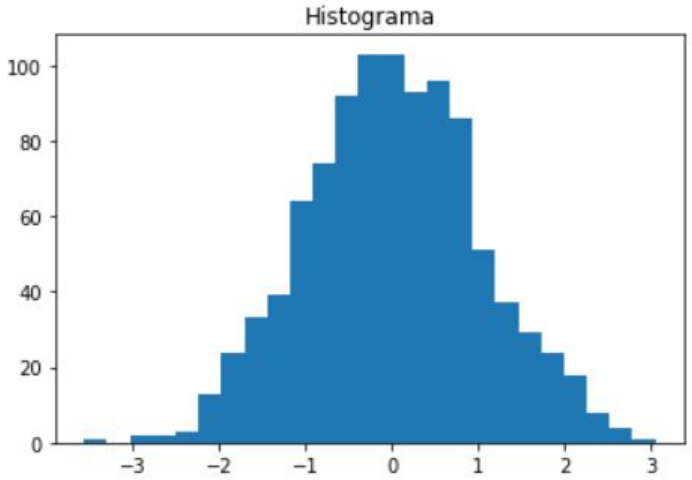
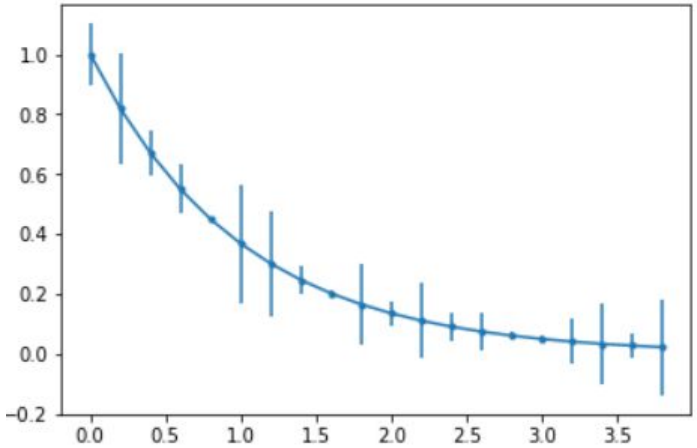


Gráfico de Erros

- Supõe uma lista de valores de erro para cada valor de y:

```
x = np.arange(0, 4, 0.2)

y = np.exp(-x)

e1 = 0.1 * np.abs(np.random.randn(len(y)))

plt.errorbar(x, y, yerr=e1, fmt='.-')
```



- Ou ainda uma lista para –y e outra para +y

```
plt.errorbar(x, y, yerr=[e1, e2], fmt='.-')
```

Gráfico de Barras

- O método .bar() espera os índices e valores da cada barra relacionada:

```
plt.bar(['1', '2', '3'], [3, 2, 5])
plt.title('Gráfico de Barras')
plt.show()
```

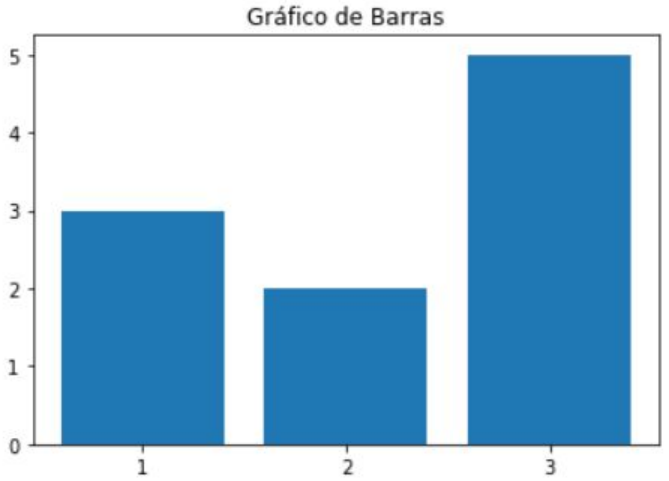


Gráfico de Barras

- Para barras distintas em cores, é necessário um *plot* para cada:

```
dict = {'A': 40, 'B': 70, 'C': 30, 'D': 85}
```

```
for i, key in enumerate(dict):
    plt.bar(i, dict[key])

plt.xticks(np.arange(4), dict.keys())
plt.yticks(list(dict.values()));
plt.show()
```

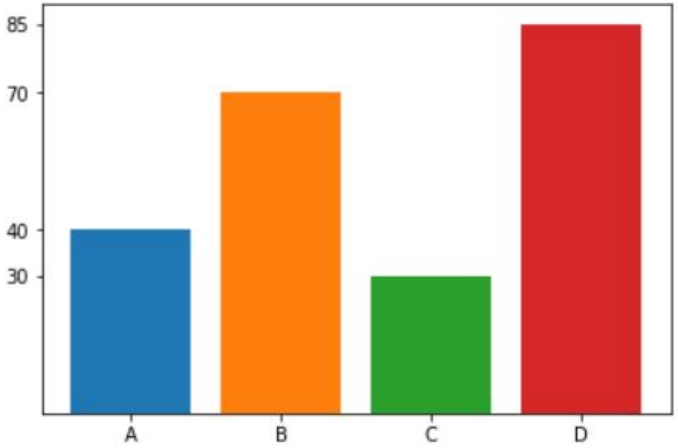


Gráfico de Barras

```
N = 5
menMeans = (20, 35, 30, 35, 27)
womenMeans = (25, 32, 34, 20, 25)
menStd = (2, 3, 4, 1, 2)
womenStd = (3, 5, 2, 3, 3)
ind = np.arange(N)    # Posição dos grupos em x
width = 0.35          # largura das barras

p1 = plt.bar(ind, menMeans, width, yerr=menStd)
p2 = plt.bar(ind, womenMeans, width,
             bottom=menMeans, yerr=womenStd)

plt.ylabel('Escores')
plt.title('Escore por grupos e sexo')
plt.xticks(ind, ('G1', 'G2', 'G3', 'G4', 'G5'))
plt.yticks(np.arange(0, 81, 10))
plt.legend((p1[0], p2[0]), ('Homem', 'Mulher'))

plt.show()
```

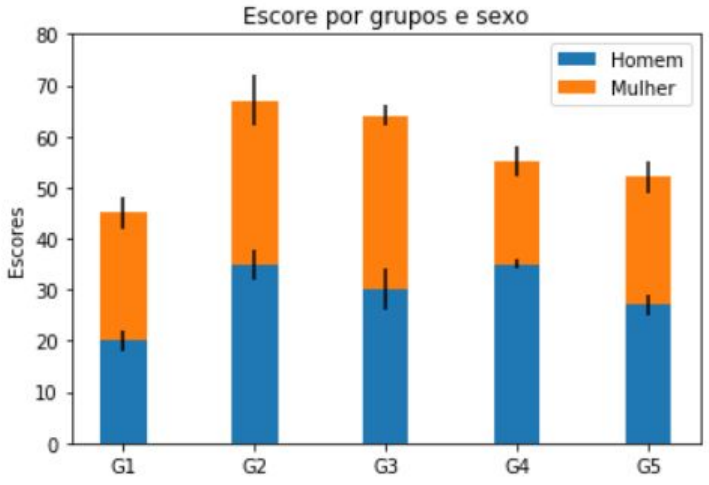


Gráfico de Pizza

- Gráficos de pizza são representações circulares, divididos em setores;
- No Matplotlib, a função pie() gera um gráfico de pizza a partir de um array unidimensional;
- Caso a soma dos valores seja inferior a 1, não haverá normalização e portanto serão usados os valores absolutos

```
x = [.2, .4, .20]
```

```
labels = ['Gatos', 'Cachorros', 'Peixes']
```

```
plt.pie(x, labels=labels);
```

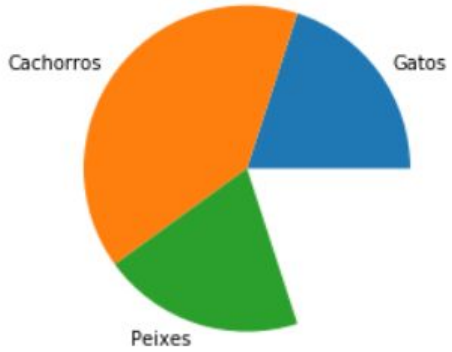
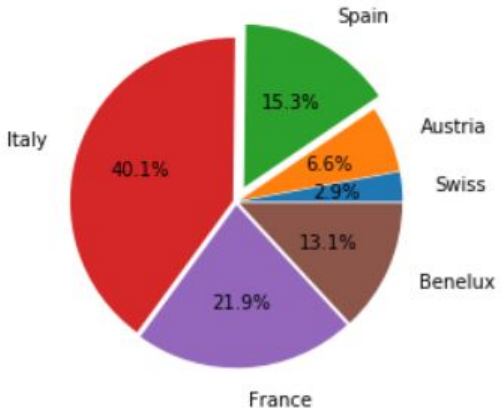


Gráfico de Pizza

- Mais parâmetros:

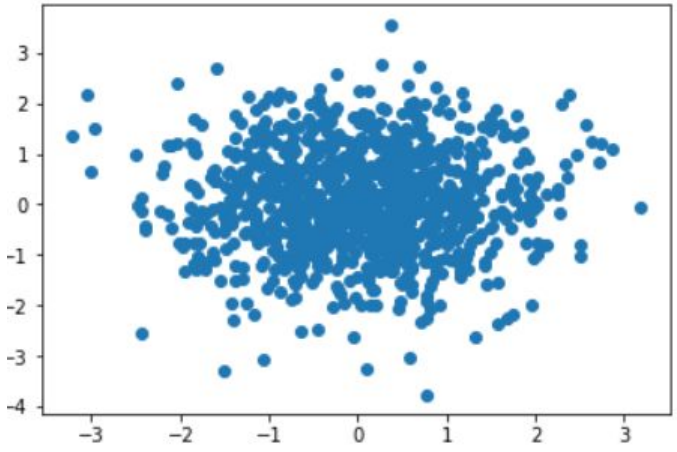
```
plt.pie(x,
        labels=labels,
        explode=[.02,.02,.1,.02,.02,.02], # Deslocamento de cada uma das fatias em relação ao centro
        autopct='%1.1f%%',                # Formato dos valores
        labeldistance=1.2                 # Distância dos rótulos em relação ao centro
       );
```



Gráficos de dispersão

- Exibem valores para dois conjuntos de dados;
- A visualização de dados é feita como uma coleção de pontos desconexos;
- Cada um deles tem suas coordenadas determinadas por (x,y).

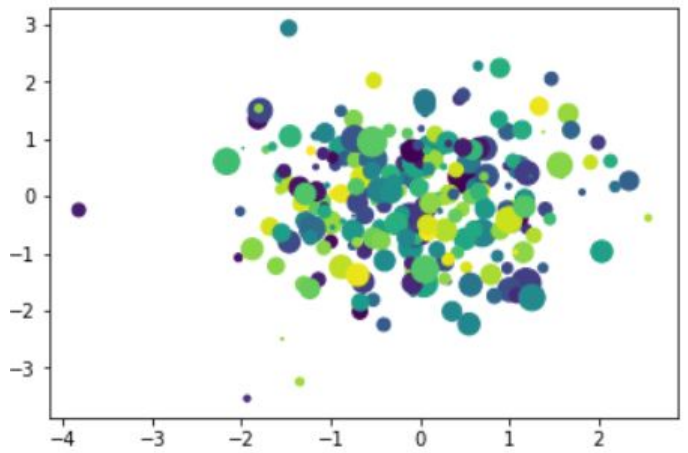
```
x = np.random.randn(1000)

y = np.random.randn(1000)

plt.scatter(x, y);
```



Gráficos de dispersão

- Mais parâmetros:

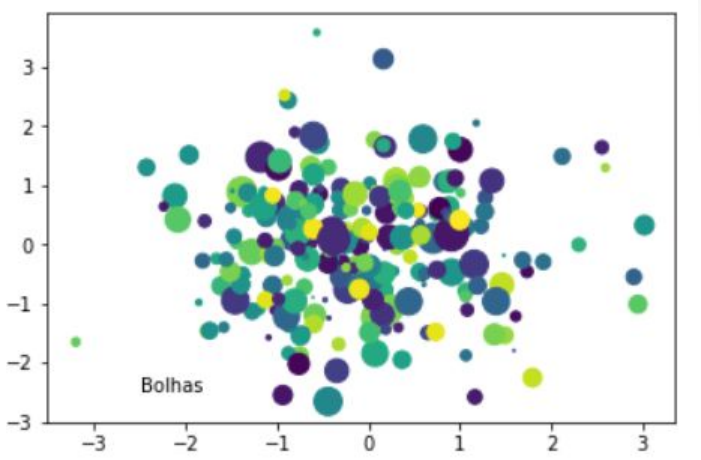
```
x = np.random.randn(500)
y = np.random.randn(500)
size = 100*np.random.randn(500)
colors = np.random.rand(500)
plt.scatter(x, y, s=size, c=colors, marker='o');
```



Textos e anotações

- Através do método **.text()** é possível definir uma texto curto mais a posição em que deve aparecer no gráfico:

```
x = np.random.randn(500)
y = np.random.randn(500)
size = 100*np.random.randn(500)
colors = np.random.rand(500)

plt.text(-2.5, -2.5, 'Bolhas');

plt.scatter(x, y, s=size, c=colors, marker='o');
```



Biblioteca construída sobre a Matplolib

https://seaborn.pydata.org

Obrigado

altinobasilio@inf.ufg.br

Dúvidas ou sugestões?