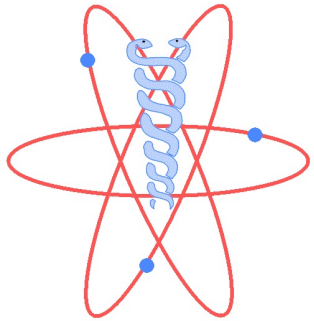
Fisica Computacional

Introdução ao Python
Aula 3

FFCLRP
EXPERIENTIA FIDES NOSTRA - 1964
UNIVERSIDADE DE SÃO PAULO



Prof. Dr. Fernando Fagundes Ferreira.

Estagiário: BSc. Victor Camargo.

10 de abril de 2020

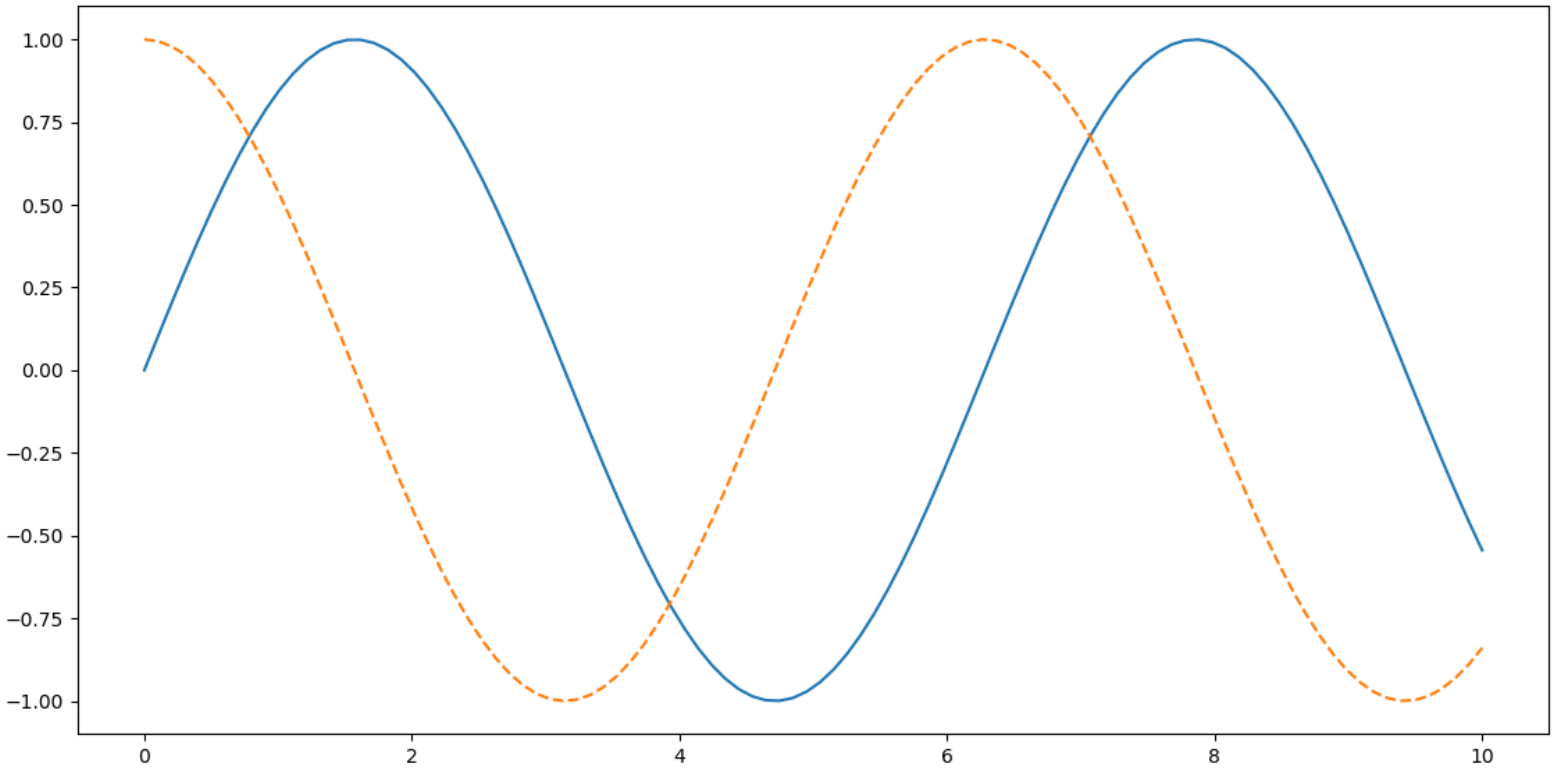
Gráficos: **matplotlib.pyplot Module**

- O programa a seguir, plota as funções seno e cosseno, ilustra a criação de um gráfico xy simples. (A linha de comando *matplotlib inline* é necessária apenas no jupyter-notebook).

```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
x = np.linspace(0, 10, 100)
y = np.sin(x)
w = np.cos(x)
fig = plt.figure()
plt.plot(x, y, '-')
plt.plot(x, w, '--')
plt.show()
```

Gráficos: **matplotlib.pyplot Module**

- A execução do programa produz o seguinte gráfico:



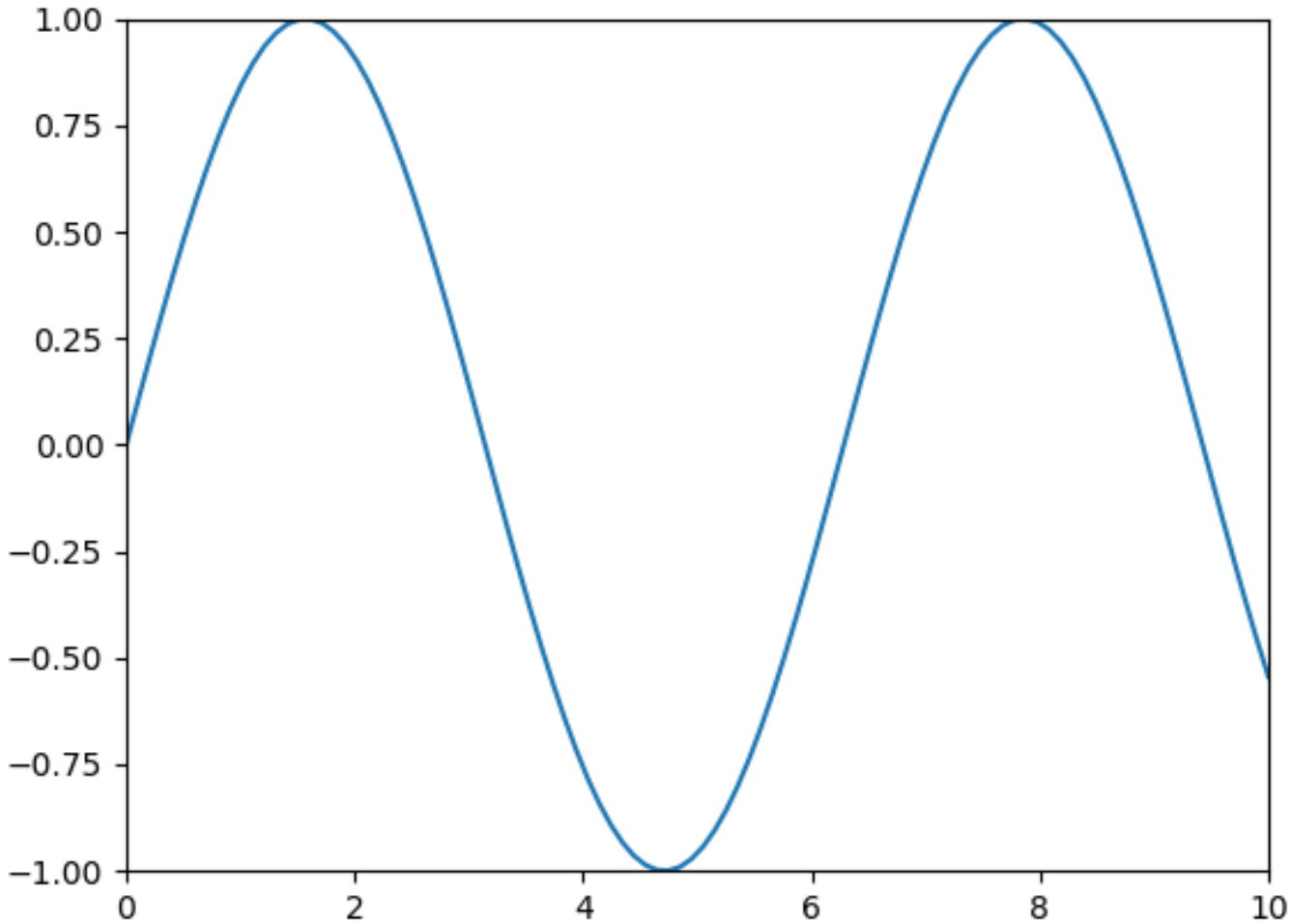
Ajustando o gráfico: limites dos eixos

- A maneira mais básica de ajustar os limites do eixo é usar os métodos plt.xlim() e plt.ylim().

```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
x = np.linspace(0, 10, 100)
fig = plt.figure()
plt.plot(x, np.sin(x))
plt.xlim(0, 10)
plt.ylim(-1, 1)
plt.savefig('plot.png') #salvar o gráfico
plt.show()
```

Ajustando o gráfico: limites dos eixos

- A execução do programa produz o seguinte gráfico:



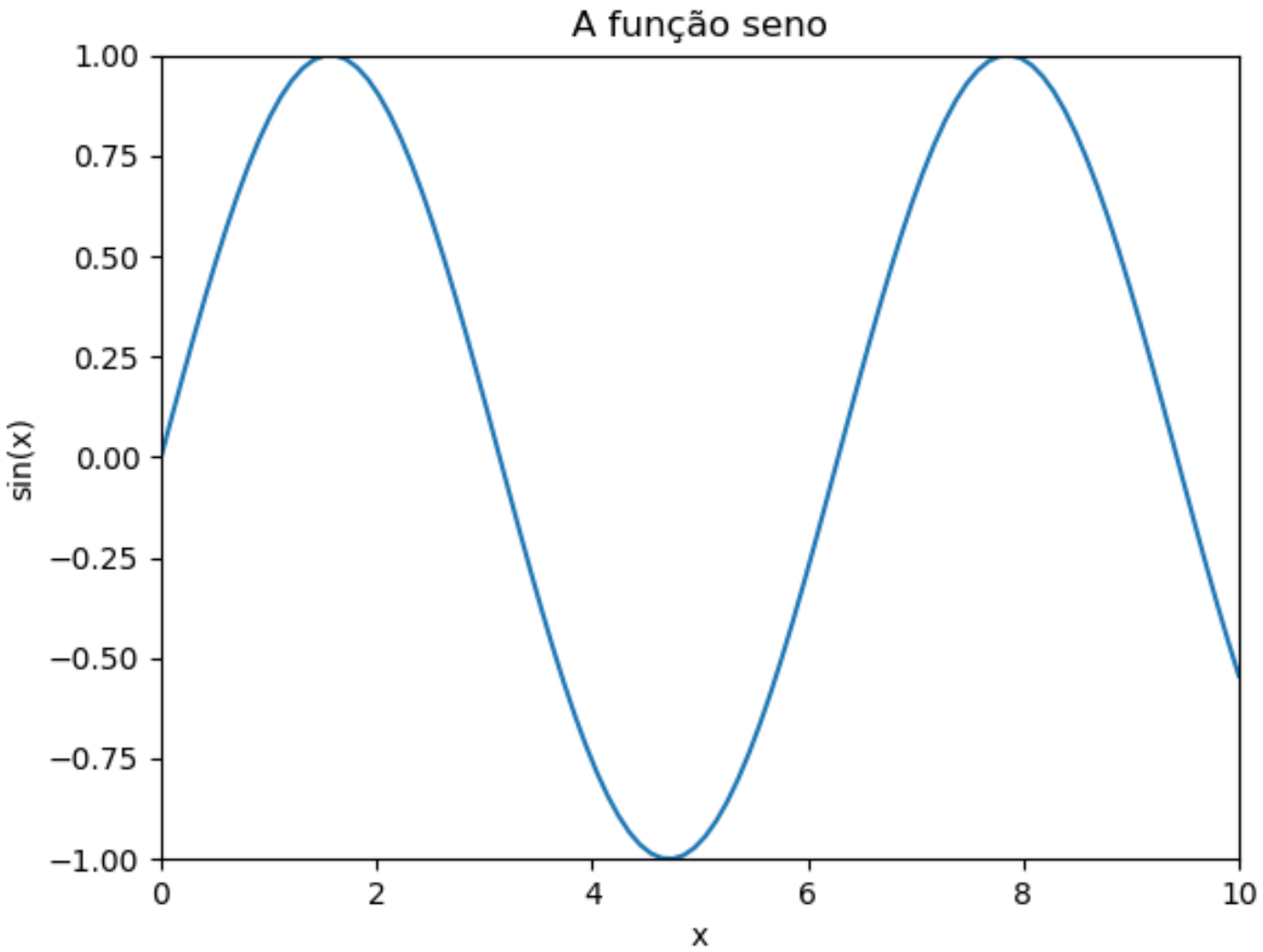
Ajustando o gráfico: Rótulos

- É possível colocar títulos e nomes nos eixos do gráfico usando os métodos label().

```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
x = np.linspace(0, 10, 100)
plt.plot(x, np.sin(x))
plt.title("A função seno")
plt.xlabel("x")
plt.ylabel("sin(x)")
plt.xlim(0, 10)
plt.ylim(-1, 1)
plt.savefig('plot.png') #salvar o gráfico
plt.show()
```

Ajustando o gráfico: Rótulos

- A execução do programa produz o seguinte gráfico:



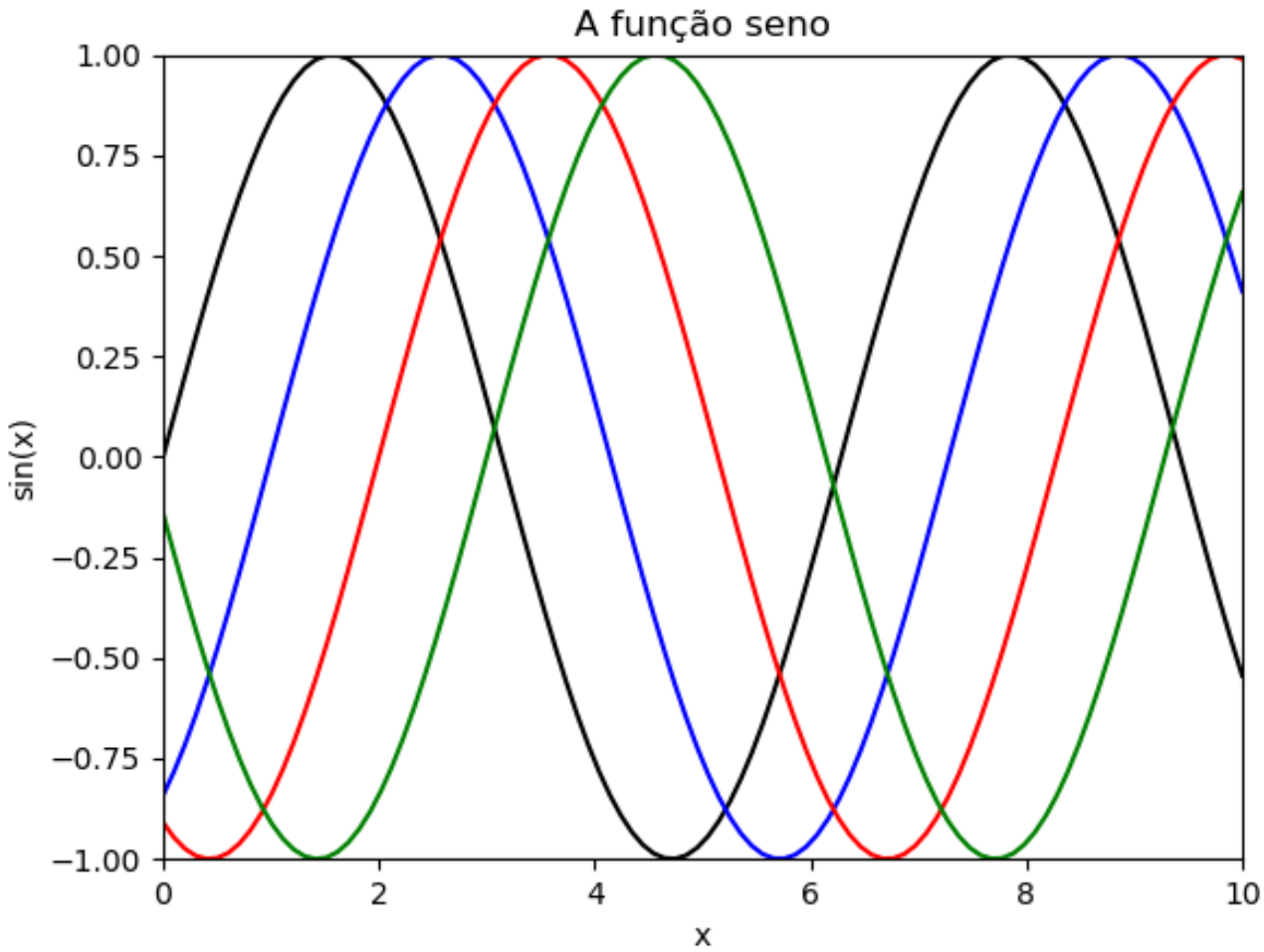
Ajustando o gráfico: Cores e estilos de linha

- A função plt.plot() recebe argumentos adicionais que podem ser usados para especificá-los.

```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
x = np.linspace(0, 10, 100)
plt.plot(x,np.sin(x-0),color='black')
plt.plot(x,np.sin(x-1),color='blue')
plt.plot(x,np.sin(x-2),color='red')
plt.plot(x,np.sin(x-3),color='green')
plt.title("A função seno")
plt.xlabel("x")
plt.ylabel("sin(x)")
plt.xlim(0, 10)
plt.ylim(-1, 1)
plt.show()
```

Ajustando o gráfico: Cores e estilos de linha

- A execução do programa produz o seguinte gráfico:



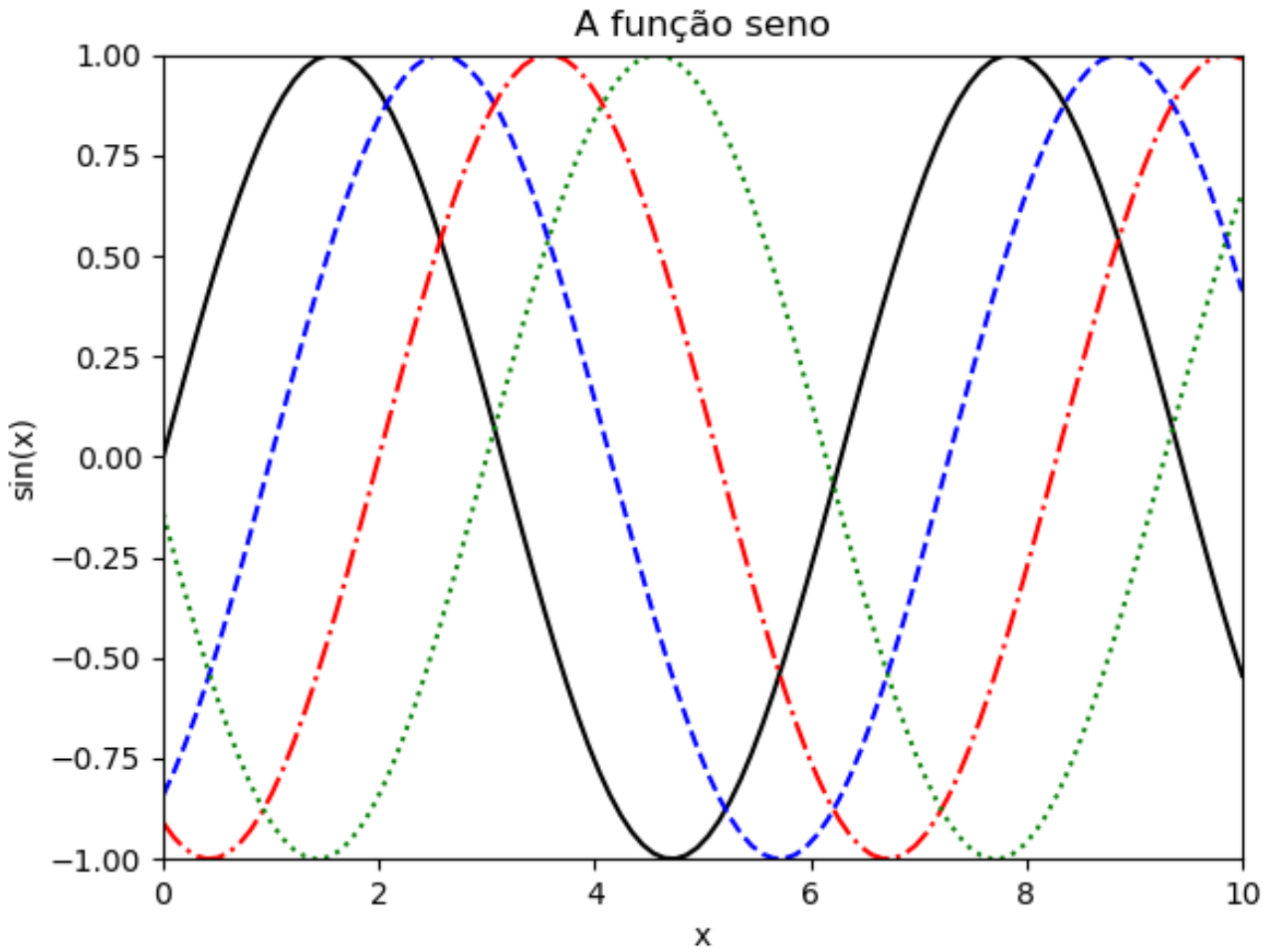
Ajustando o gráfico: Cores e estilos de linha

- Da mesma forma, você pode ajustar o estilo da linha usando a palavra-chave linestyle.

```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
x = np.linspace(0, 10, 100)
plt.plot(x,np.sin(x-0),color='black',linestyle='solid')
plt.plot(x,np.sin(x-1),color='blue',linestyle='dashed')
plt.plot(x,np.sin(x-2),color='red',linestyle='dashdot')
plt.plot(x,np.sin(x-3),color='green',linestyle='dotted')
plt.title("A função seno")
plt.xlabel("x")
plt.ylabel("sin(x)")
plt.xlim(0, 10)
plt.ylim(-1, 1)
plt.show()
```

Ajustando o gráfico: Cores e estilos de linha

- A execução do programa produz o seguinte gráfico:



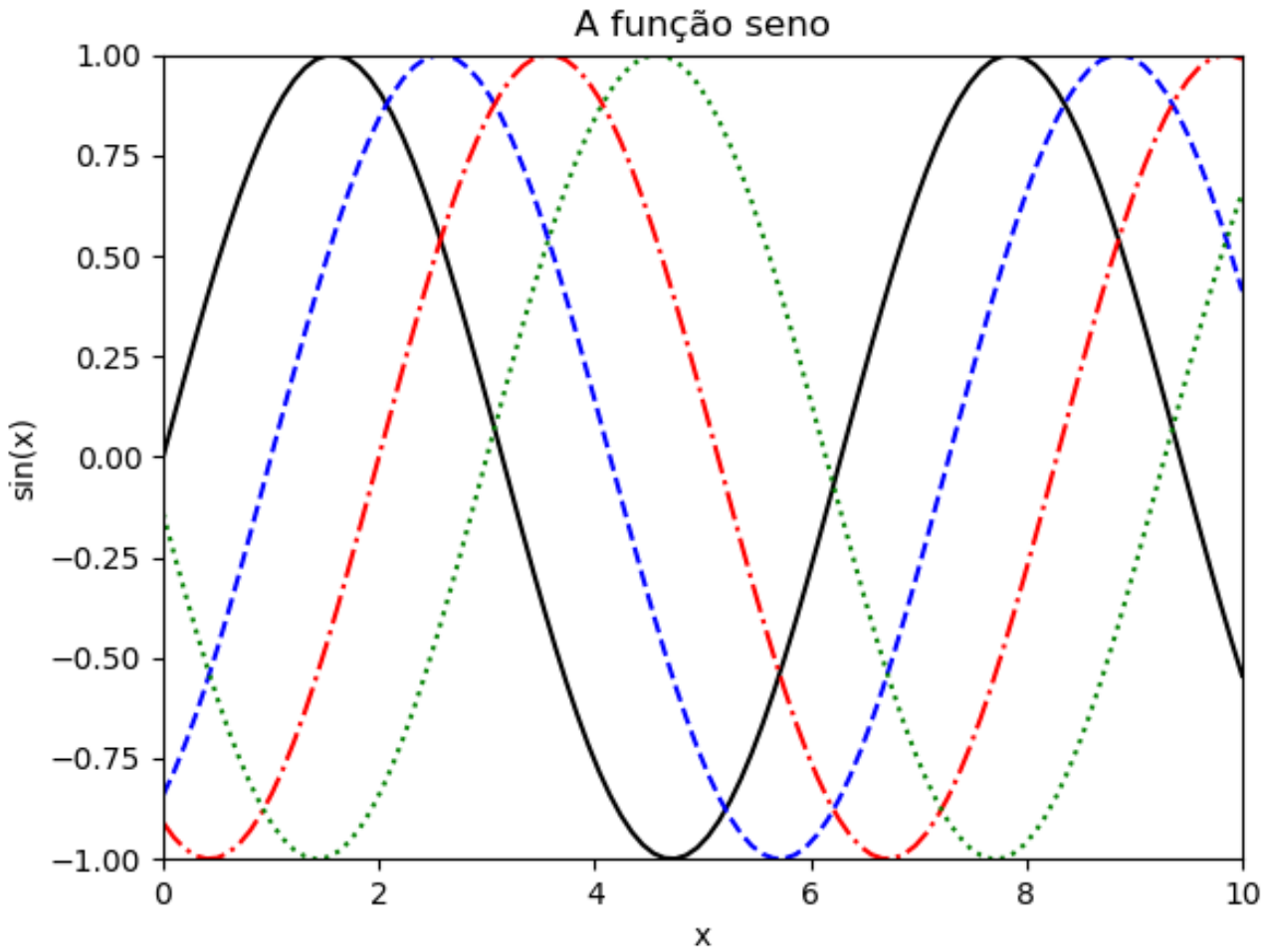
Ajustando o gráfico: Cores e estilos de linha

- Esses comandos linestyle e color podem ser combinados em um único argumento sem palavra-chave para a função plt.plot().

```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
x = np.linspace(0, 10, 100)
plt.plot(x,np.sin(x-0),'-k')
plt.plot(x,np.sin(x-1),'--b')
plt.plot(x,np.sin(x-2),'-.r')
plt.plot(x,np.sin(x-3),':g')
plt.title("A função seno")
plt.xlabel("x")
plt.ylabel("sin(x)")
plt.xlim(0, 10)
plt.ylim(-1, 1)
plt.show()
```

Ajustando o gráfico: Cores e estilos de linha

- A execução do programa produz o seguinte gráfico:



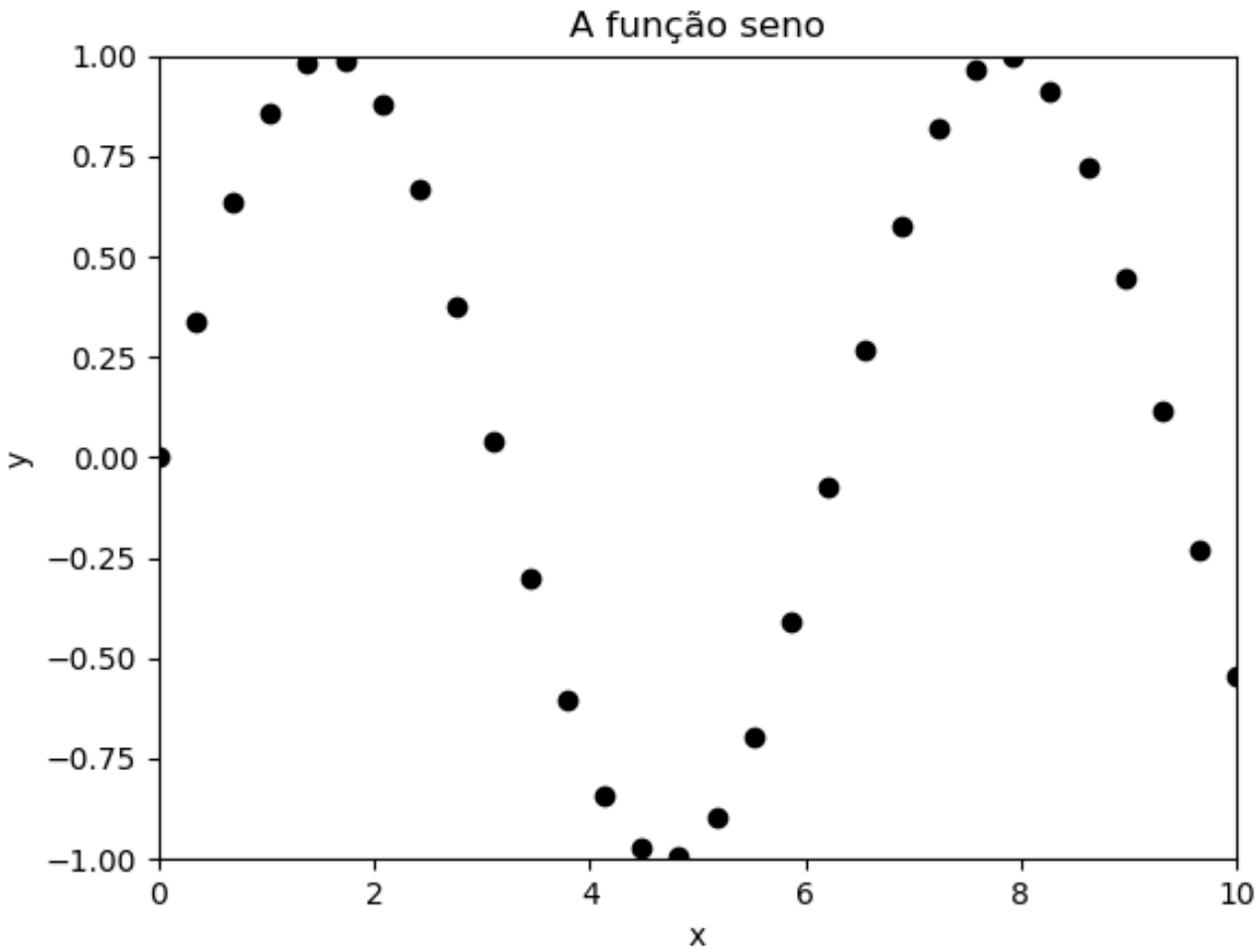
Gráficos de dispersão simples

- Usando a função plt.plot () é possível fazer um diagrama de dispersão simples. Em vez de os pontos serem unidos por segmentos de linha, aqui os pontos são representados individualmente com um ponto, círculo ou outra forma.

```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
x = np.linspace(0, 10, 30)
y = np.sin(x)
plt.plot(x, y, 'o', color='black');
plt.title("A função seno")
plt.xlabel("x")
plt.ylabel("y")
plt.xlim(0,10)
plt.ylim(-1,1)
plt.show()
```

Gráficos de dispersão simples

- A execução do programa produz o seguinte gráfico:



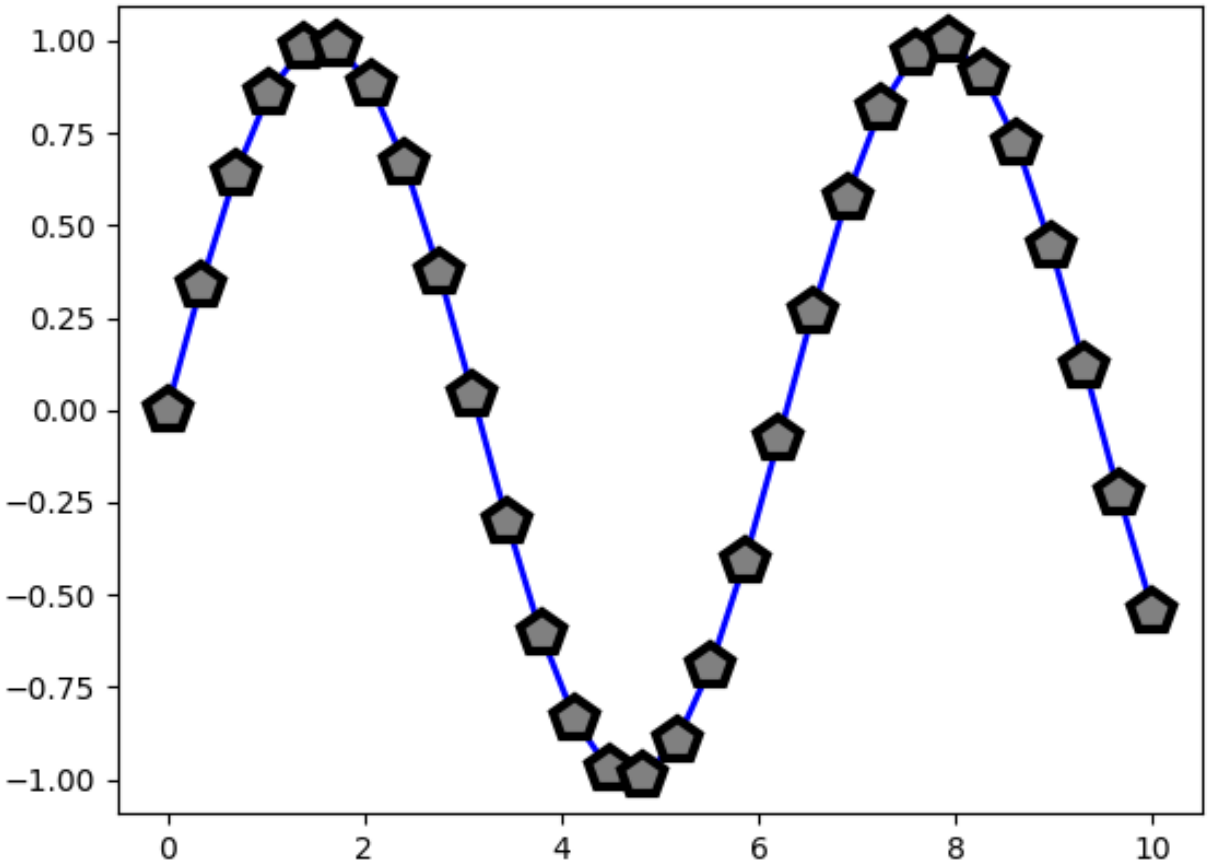
Gráficos de dispersão simples

- Exemplo: cores, ponto em forma de polígono e outras características.

```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
x = np.linspace(0, 10, 30)
y = np.sin(x)
plt.plot(x, y, '-p', color='blue',
         markersize=15, linewidth=2,
         markerfacecolor='gray',
         markeredgecolor='black',
         markeredgewidth=3)
plt.show()
```

Gráficos de dispersão simples

- A execução do programa produz o seguinte gráfico:



Gráficos de dispersão simples

- Os estilos de linha e marcador são especificados pelos caracteres da sequência mostrados na tabela a seguir (apenas alguns dos caracteres disponíveis são mostrados):

'-'	Solid line
'--'	Dashed line
'-.'	Dash-dot line
':'	Dotted line
'o'	Circle marker
'^'	Triangle marker
's'	Square marker
'h'	Hexagon marker
'x'	x marker

Gráficos de dispersão simples

- Alguns dos códigos de localização (loc) para posicionamento da legenda são:

0	”Best” location
1	Upper right
2	Upper left
3	Lower left
4	Lower right

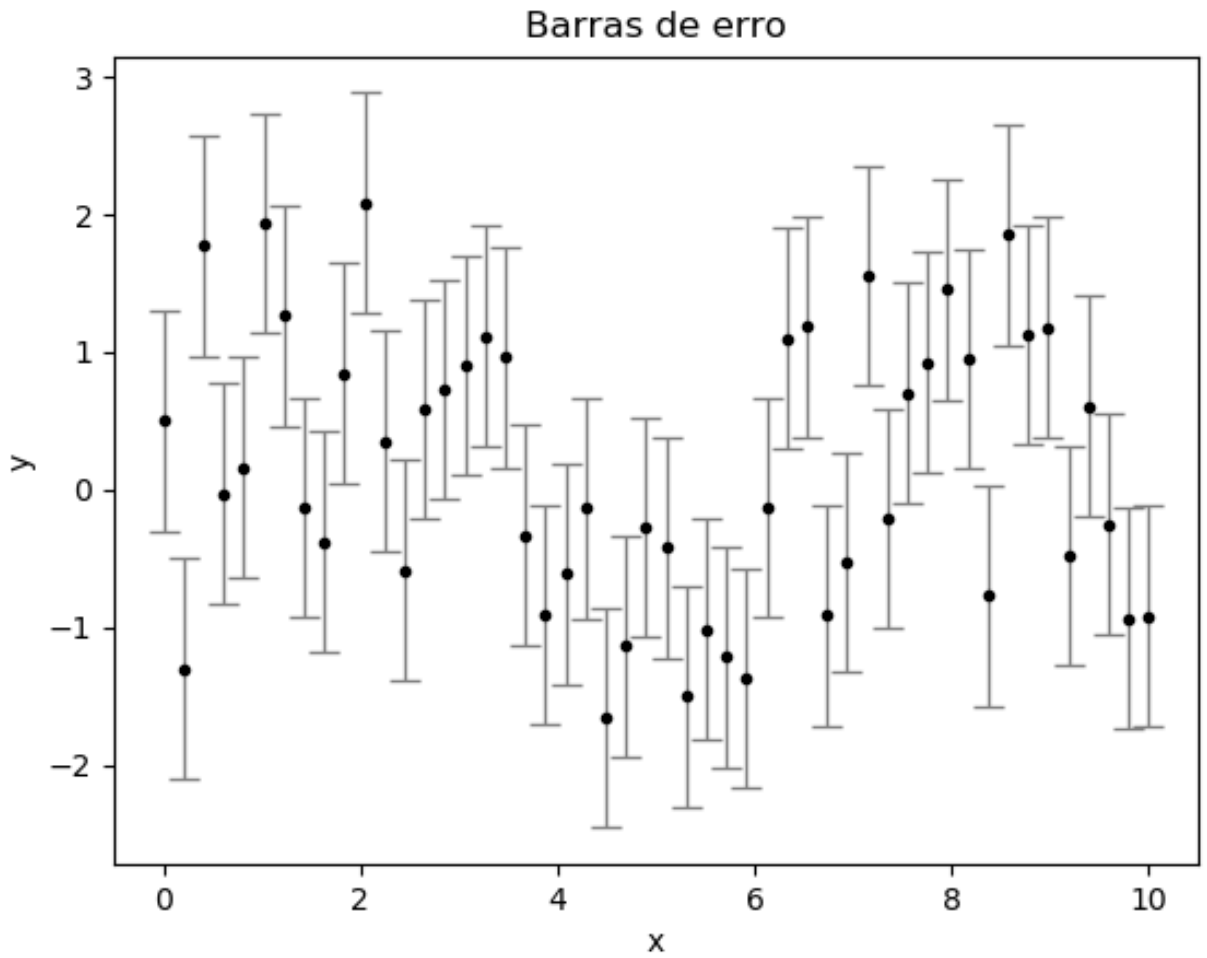
Barras de erro básicas

- Uma barra de erros básica pode ser criada com uma única chamada de função Matplotlib.

```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
x = np.linspace(0, 10, 50)
dy = 0.8
y = np.sin(x) + dy * np.random.randn(50)
plt.errorbar(x, y, yerr=dy, fmt='.k',
             ecolor='gray', elinewidth=1, capsize=5);
plt.title("Barras de erro")
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```

Barras de erro básicas

- A execução do programa produz o seguinte gráfico:



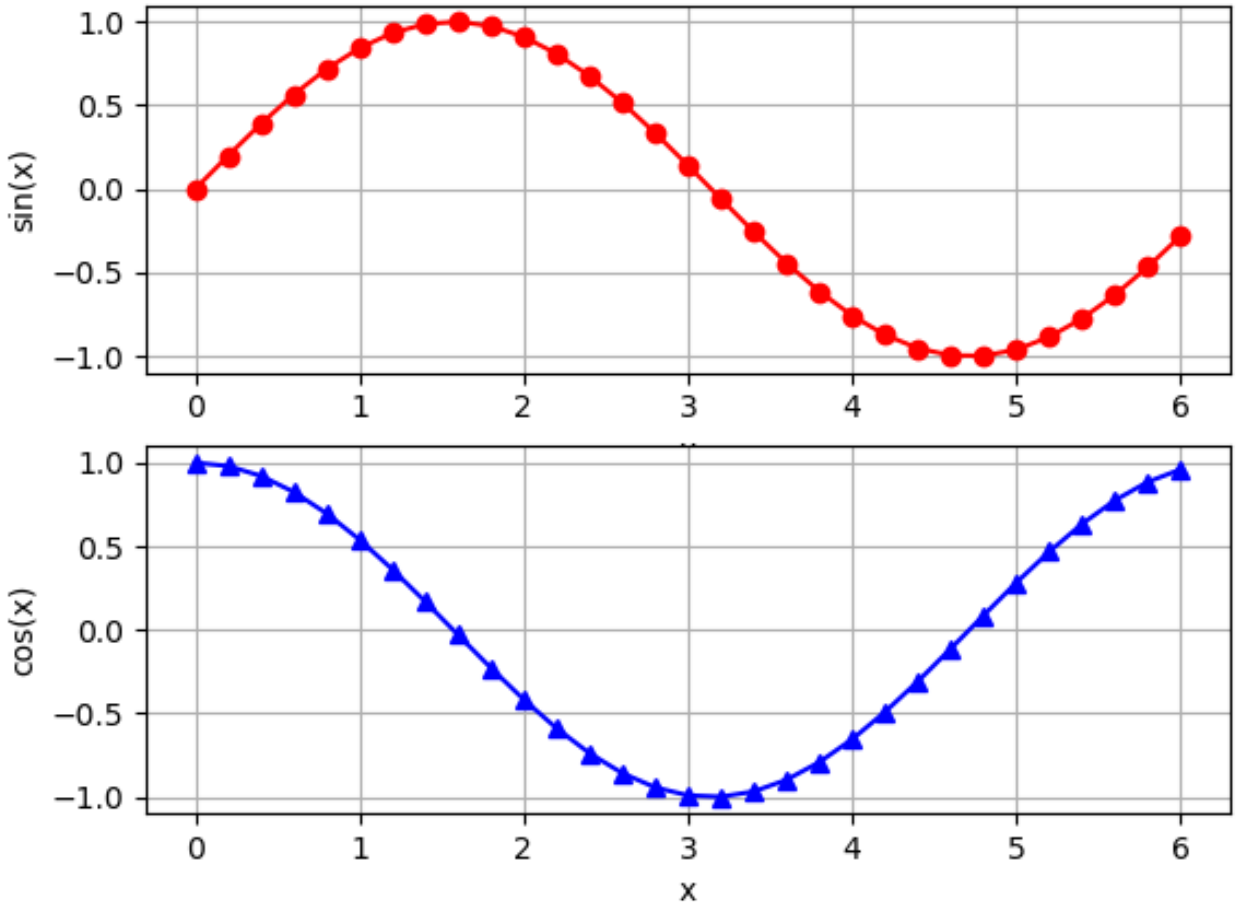
Subplots

- É possível ter mais de um gráfico em uma figura, graças à função subplot(linha,colunas,número do plot).

```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
x = np.arange(0.0,6.2,0.2)
plt.subplot(2,1,1) # primeiro subplot
plt.plot(x,np.sin(x),'o-',c='r')
plt.xlabel('x'); plt.ylabel('sin(x)')
plt.grid(True) # grade de coordenadas
plt.subplot(2,1,2) # segundo subplot
plt.plot(x,np.cos(x),'^-',c='b')
plt.xlabel('x'); plt.ylabel('cos(x)')
plt.grid(True)
plt.show()
```

Subplots

- Gráfico gerado:



Histograma básico

- Um histograma simples é um ótimo primeiro passo para entender um conjunto de dados.

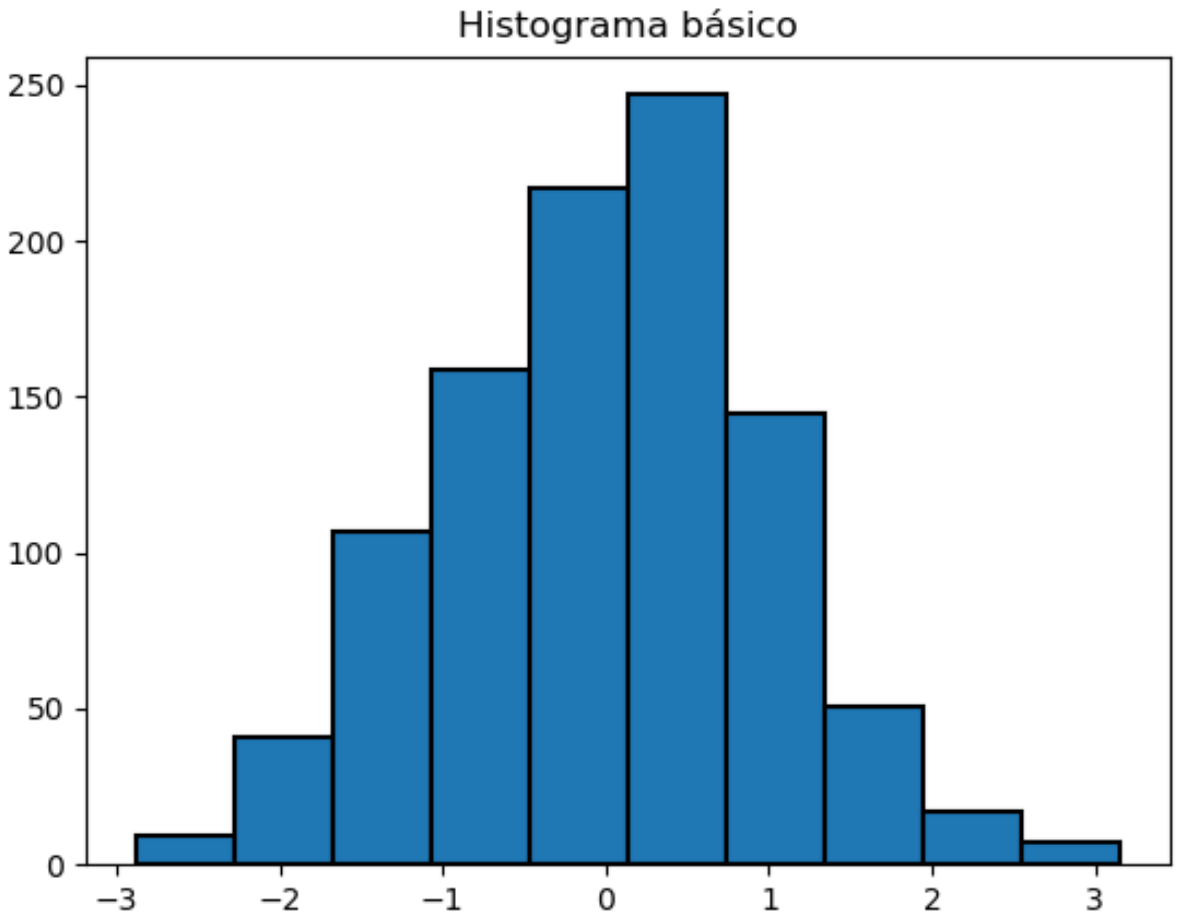
```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline

x = np.random.randn(1000)

plt.hist(x,edgecolor='black', linewidth=1.5)
plt.title('Histograma básico')
plt.show()
```

Histograma básico

- Gráfico gerado:



Gráficos de densidade e contorno

- Às vezes, é útil exibir dados tridimensionais em duas dimensões usando contornos ou regiões codificadas por cores. Existem três funções do Matplotlib que podem ser úteis para esta tarefa:
  - plt.contour() plotagens de contorno
  - plt.contourf() plotagens de contorno preenchidas
  - plt.imshow() exibir imagens

Gráficos de densidade e contorno

- Exemplo:

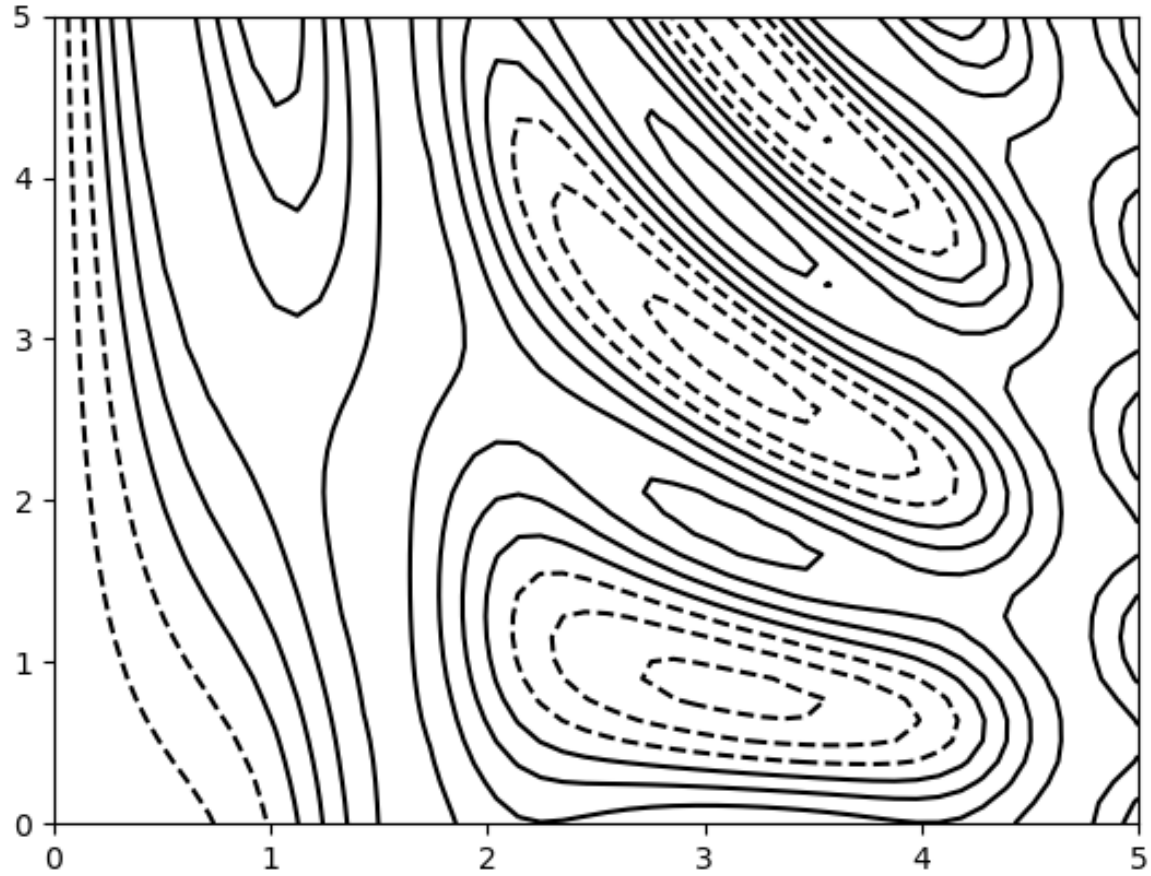
```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline

def f(x,y):
        f = np.sin(x)**10 + np.cos(10+y*x)*np.cos(x)
        return f

x = np.linspace(0,5,50)
y = np.linspace(0,5,40)
X,Y = np.meshgrid(x,y) # níveis de contorno
Z = f(X,Y)
plt.contour(X,Y,Z,colors='black')
plt.show()
```

Gráficos de densidade e contorno

- Gráfico gerado:



Gráficos de densidade e contorno

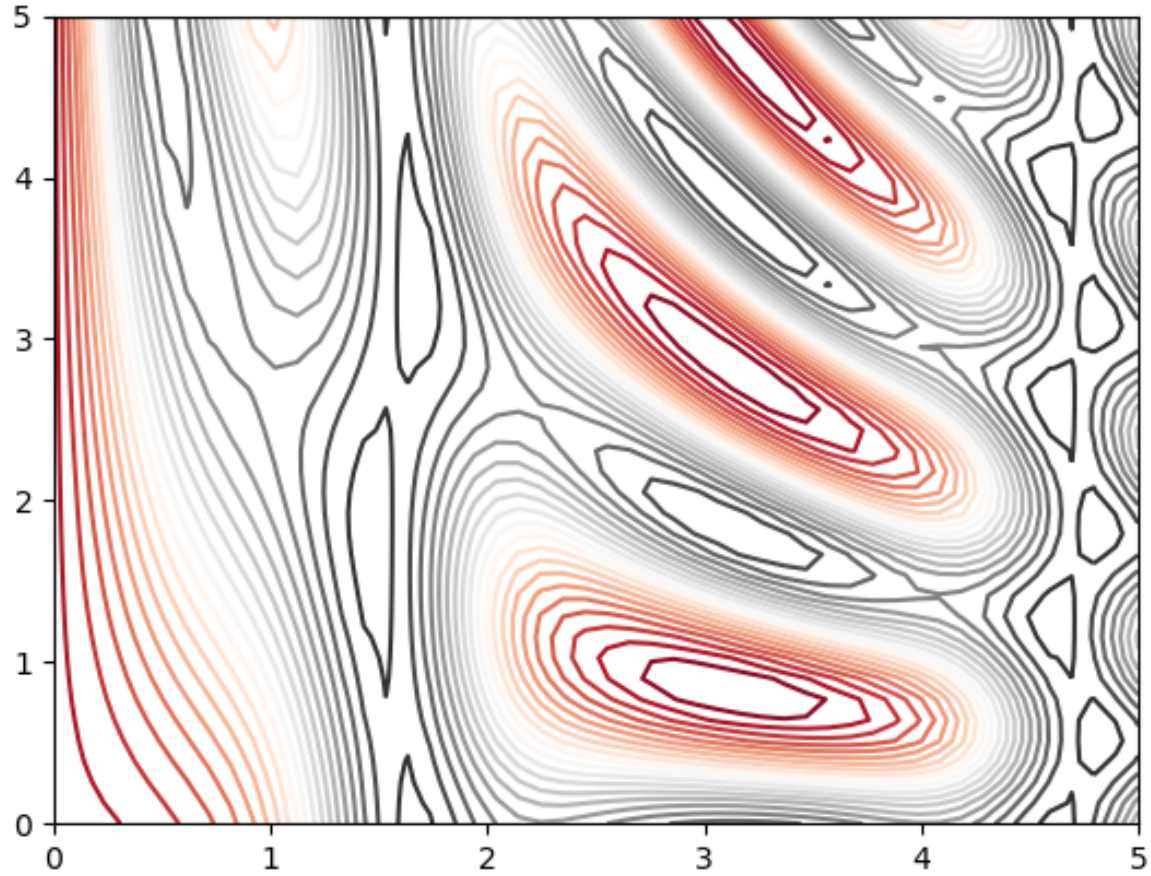
- É possível codificar em cores as linhas especificando um mapa de cores com o argumento cmap.

```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
def f(x,y):
        f = np.sin(x)**10 + np.cos(10+y*x)*np.cos(x)
        return f

x = np.linspace(0,5,50)
y = np.linspace(0,5,40)
X,Y = np.meshgrid(x,y) # níveis de contorno
Z = f(X,Y)
plt.contour(X,Y,Z,20,cmap='RdGy')
plt.show()
```

Gráficos de densidade e contorno

- Gráfico gerado:



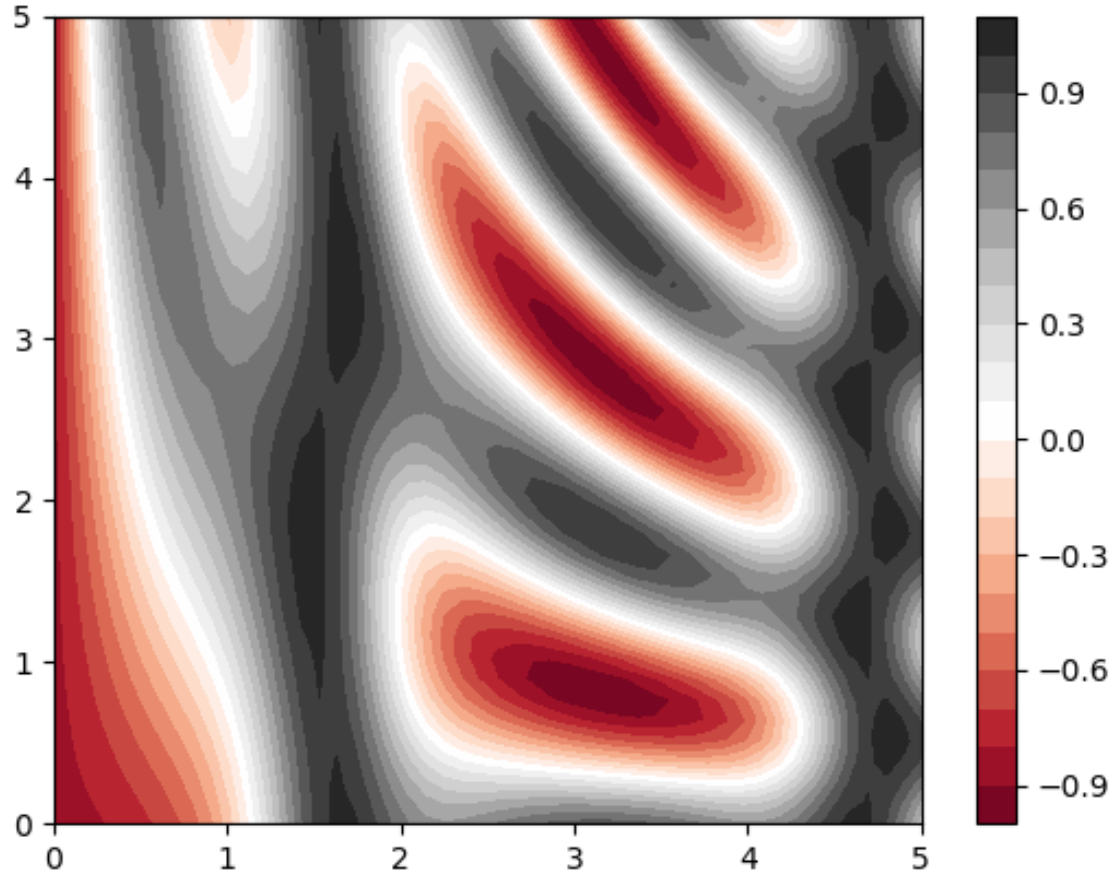
Gráficos de densidade e contorno

- Além disso, adicionaremos o comando plt.colorbar(), que cria um eixo adicional com informações das cores rotuladas para o gráfico.

```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
def f(x,y):
        f = np.sin(x)**10 + np.cos(10+y*x)*np.cos(x)
        return f
x = np.linspace(0,5,50)
y = np.linspace(0,5,40)
X,Y = np.meshgrid(x,y) # níveis de contorno
Z = f(X,Y)
plt.contourf(X,Y,Z,20,cmap='RdGy')
plt.colorbar()
plt.show()
```

Gráficos de densidade e contorno

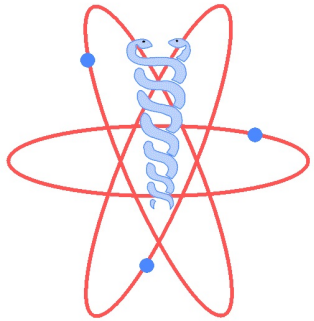
- Gráfico gerado:



Fisica Computacional

Introdução ao Python
Aula 3

FFCLRP
EXPERIENTIA FIDES NOSTRA - 1964
UNIVERSIDADE DE SÃO PAULO



Prof. Dr. Fernando Fagundes Ferreira.

Estagiário: BSc. Victor Camargo.

10 de abril de 2020