Python Aplicado a Métodos Numéricos

Plotando Gráficos de Alta Qualidade Utilizando o Módulo Matplotlib

Prof. Daniel Saad Nogueira Nunes

Instituto Federal de Brasília, Câmpus Taguatinga

Sumário

1. Introdução
2. Matplotlib
3. CSV

Introdução

- Este é o terceiro módulo da oficina **"Python Aplicado a Métodos Numéricos".**
- Até o momento, sabemos:
  - Utilizar as estruturas básicas da Linguagem Python.
  - Utilizar os módulos específicos da linguagem, como o *scipy* e *numpy*, para resolver problemas complexos.

Introdução

- Hoje veremos como expor os resultados obtidos na resolução deste problemas de uma maneira gráfica e fácil.
- Também veremos como importar e exportar dados diretamente de planilhas, automatizando uma série de tarefas.
- Ferramentas utilizada: módulos `matplotlib` e `csv`.

Bibliografia

- TOSI, Sandro. **Matplotlib for Python developers**. Packt Publishing Ltd, 2009.
- `https://matplotlib.org/`

Sumário

2 Matplotlib

Matplotlib

- O módulo `Matplotlib` tem como objetivo gerar gráficos de alta qualidade para publicações científicas.
- Visa simplicidade.
- De acordo John Hunter, criador e líder do projeto:

  Matplotlib tries to make easy things easy and hard things possible.

Matplotlib

- Por que `Matplotlib`?

Matplotlib

- O Python é utilizado como ferramenta para obtenção e tratamento dos dados.
- Por que utilizar uma outra ferramenta como `Matlab` ou `Gnuplot` para plotar os seus gráficos?
- `Matplotlib` possibilita fazer tudo utilizando uma única linguagem.
- Aumento da capacidade de escrita e produtividade.

Matplotlib

Facilidades do Matplotlib

- Utiliza Python, uma linguagem amplamente utilizada no mundo científico.
- É *open source*, não é necessário adquirir uma licença.
- É completo: é possível utilizar uma única linguagem de programação durante todo o *workflow*.
- Integrado ao LaTeX.
- Multiplataforma e portável.

Sumário

2 Matplotlib
- Primeiros passos
- Tipos de gráficos
- Histogramas
- Gráficos de barra
- Gráficos de pizza
- Gráficos de dispersão

Primeiros passos

- Em vez de tentar apresentar todos os detalhes intrínsecos ao módulo, utilizaremos exemplos para entender o funcionamento.
- Estes exemplos podem ser alterados para fazer coisas mais complexas.

Primeiros passos

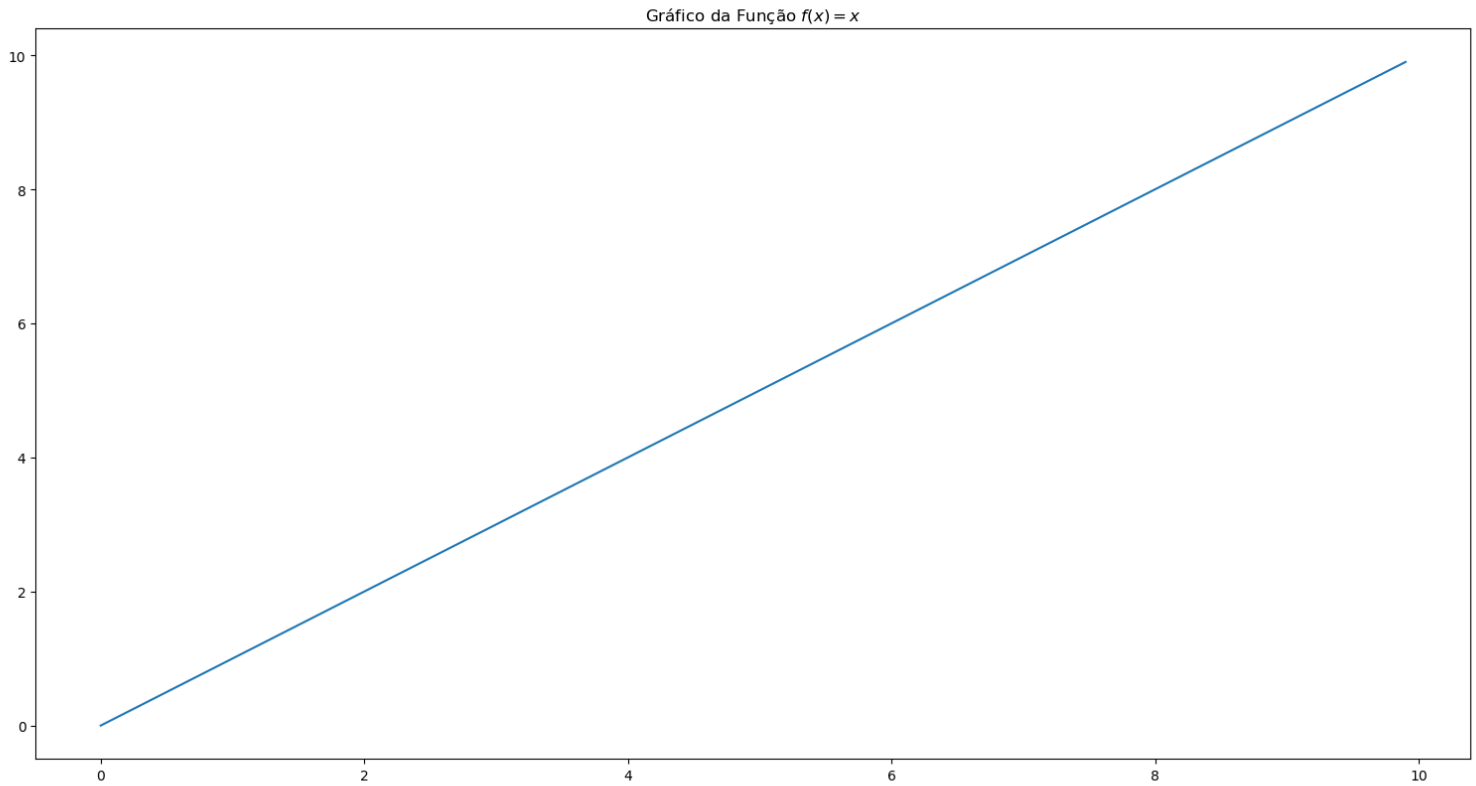
- Começaremos com uma coisa simples.
- Plotar a função identidade:

$$f : \mathbb{R} \to \mathbb{R}$$
$$x \mapsto x$$

Primeiros passos

```
"""
    Autor: Daniel Saad Nogueira Nunes
    Comentários: Exemplo minimalista da plotagem da função f(x) = x utilizando
    a biblioteca Matplotlib
"""

import matplotlib.pyplot as plt
import numpy as np

#   cria um objeto figura e o coloca em fig
fig = plt.figure()
#   da figura criada, é alocada uma matriz 1x1 de gráficos
#   ax corresponde a matriz 1
ax = fig.add_subplot(111)
#   gera o dominio da função y = x
#   domínio da função [0,10) com amostras de tamanho 0.1
x  = np.arange(0,10,0.1)
#   Gera a imagem da função y = x
y = np.array([ a for a in x ])
#   Plota a imagem
l = plt.plot(x,y)
#   Coloca um título correspondente na figura
t = ax.set_title("Gráfico da Função " r"$f(x) = x$")
#   Mostra o resultado na tela
plt.show()
```

Primeiros passos



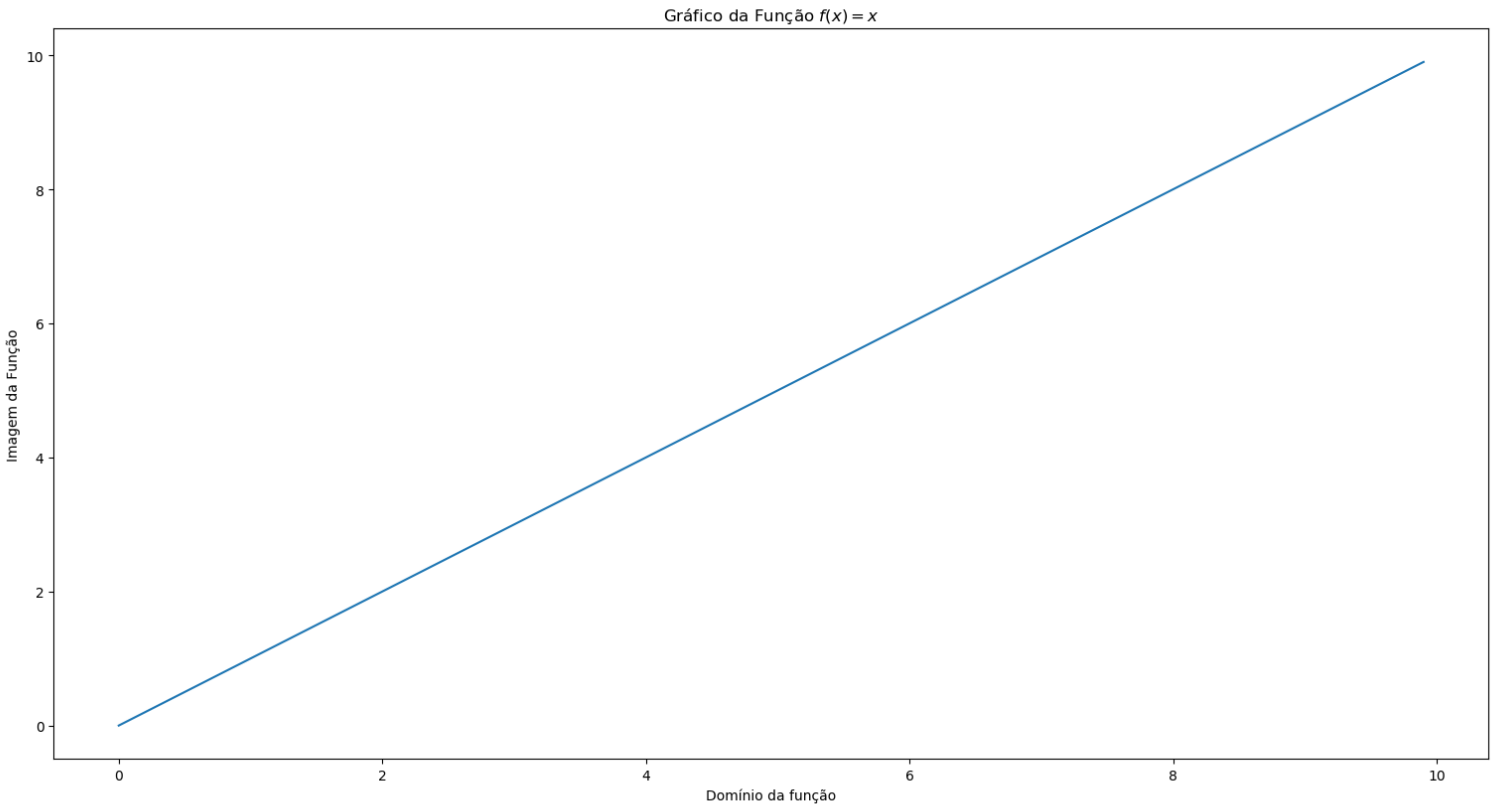
Adicionando eixos

```
"""
    Autor: Daniel Saad Nogueira Nunes
    Comentários: Exemplo da plotagem da função f(x) = x utilizando
    a biblioteca Matplotlib. Neste caso são adicionadas informações nos
    eixos x e y do gráfico.
"""

import matplotlib.pyplot as plt
import numpy as np

#   Cria um objeto figura e o coloca em fig
fig = plt.figure()
#   Da figura criada, é alocada uma matriz 1x1 de gráficos
#   ax corresponde a matriz 1
ax = fig.add_subplot(111);
```

Adicionando eixos

```
#   Gera o dominio da função y = x
x  = np.arange(0,10,0.1) # domínio da função [0,10] com amostras de tamanho 0.1
#   Gera a imagem da função y = x
y = [ a for a in x ]
#   Plota a imagem
l = plt.plot(x,y)
#   Coloca um título correspondente na figura
t = ax.set_title("Gráfico da Função " r"$f(x) = x$")
#   Insere título no eixo x
t_x = ax.set_xlabel("Domínio da função")
#   Insere título no eixo y
t_y = ax.set_ylabel("Imagem da Função")
#   Mostra o resultado na tela
plt.show()
```

Adicionando eixos



Adicionando a legenda

- Para adicionar a legenda, invocamos o método `legend()` do objeto `ax`.

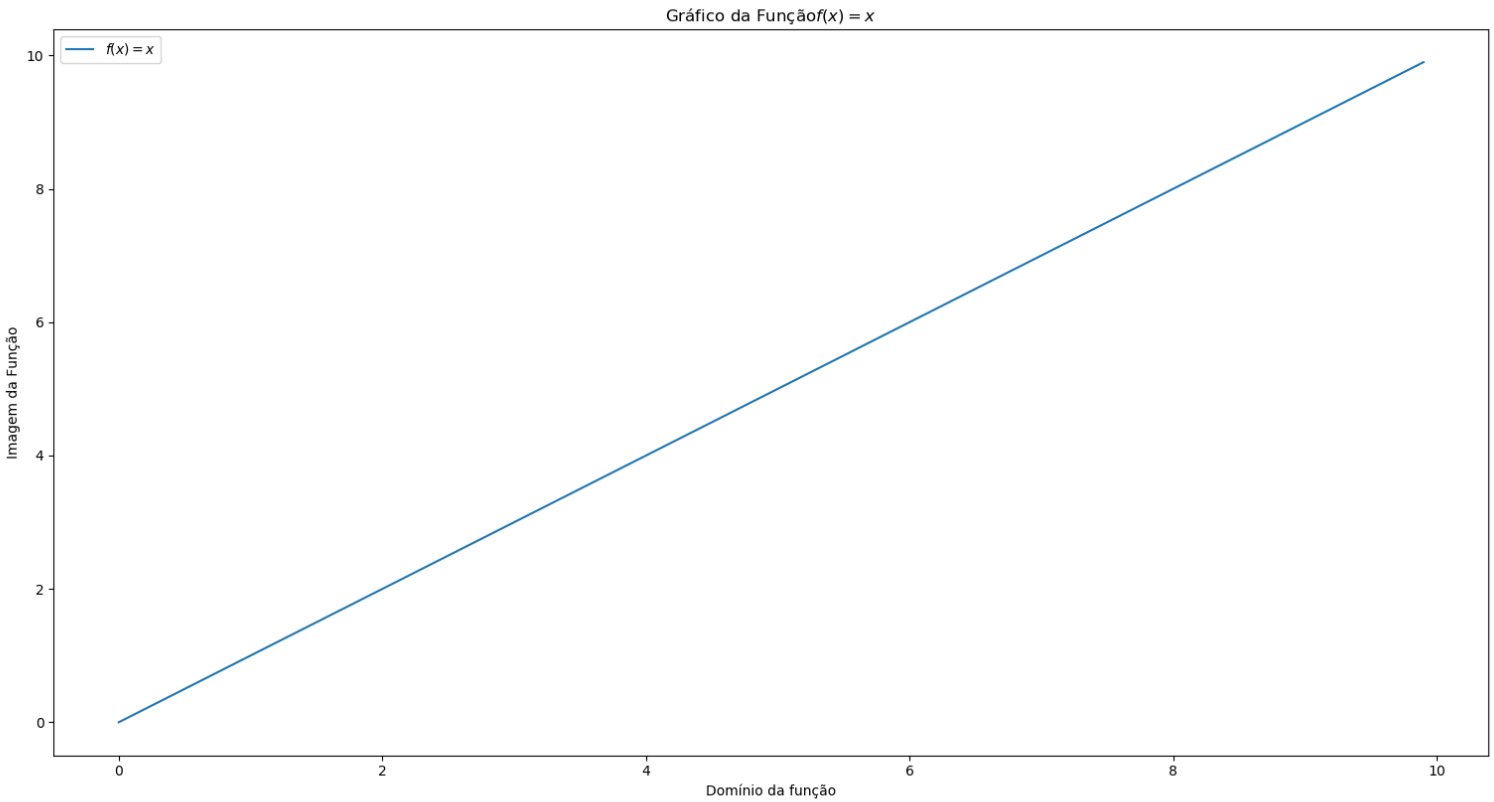
Adicionando a legenda

```
"""
    Autor: Daniel Saad Nogueira Nunes
    Comentários: Exemplo da plotagem da função £f(x) = x£ utilizando
    a biblioteca Matplotlib. Neste caso são adicionadas informações nos
    eixos x e y do gráfico e a legenda correspondente.
"""

import matplotlib.pyplot as plt
import numpy as np

#   Cria um objeto figura e o coloca em fig
fig = plt.figure()
#   Da figura criada, é alocada uma matriz 1x1 de gráficos
#   ax corresponde a matriz 1
ax = fig.add_subplot(111);
```

Adicionando a legenda

```
#   Gera o dominio da função y = x
x  = np.arange(0,10,0.1) # domínio da função [0,10] com amostras de tamanho 0.1
#   Gera a imagem da função y = x
y = [ a for a in x ]
#   Plota a imagem e anota a legenda do gráfico
l = plt.plot(x,y,label=r"$f(x)=x$")
#   Coloca um título correspondente na figura
t = ax.set_title("Gráfico da Função"+r"$f(x) = x$")
#   Insere título no eixo x
t_x = ax.set_xlabel("Domínio da função")
#   Insere título no eixo y
t_y = ax.set_ylabel("Imagem da Função")
#   Insere  a legenda
l = ax.legend()
#   Mostra o resultado na tela
plt.show()
```

Adicionando a legenda



Posicionando a legenda

- Caso a posição da legenda não tenha ficado adequada, é possível ajustá-la.
- É preciso indicar qual a posição.
- Utilizamos novamente o método `legend()`. Mas dessa vez passamos um parâmetro para ele.
- `legend('<parâmetro>')`.

Posicionando a legenda

Tabela: Possíveis posições da legenda.

Parâmetro	Descrição
'best'	posicionamento automático
'upper right'	canto superior direito
'upper left'	canto superior esquerdo
'lower left'	canto inferior esquerdo
'lower right'	canto inferior direito
'right'	direita
'center left'	centralizado à esquerda
'center right'	centralizado à direita
'lower center'	centralizado abaixo
'upper center'	centralizado acima
'center'	centralizado

Posicionando a legenda

- Vamos testar alguns parâmetros?

Melhorando a visualização.

- Para melhorar a visualização, é possível inserir um *grid*.
- Basta invocar o método `grid` do objeto `ax` com parâmetro `True`.

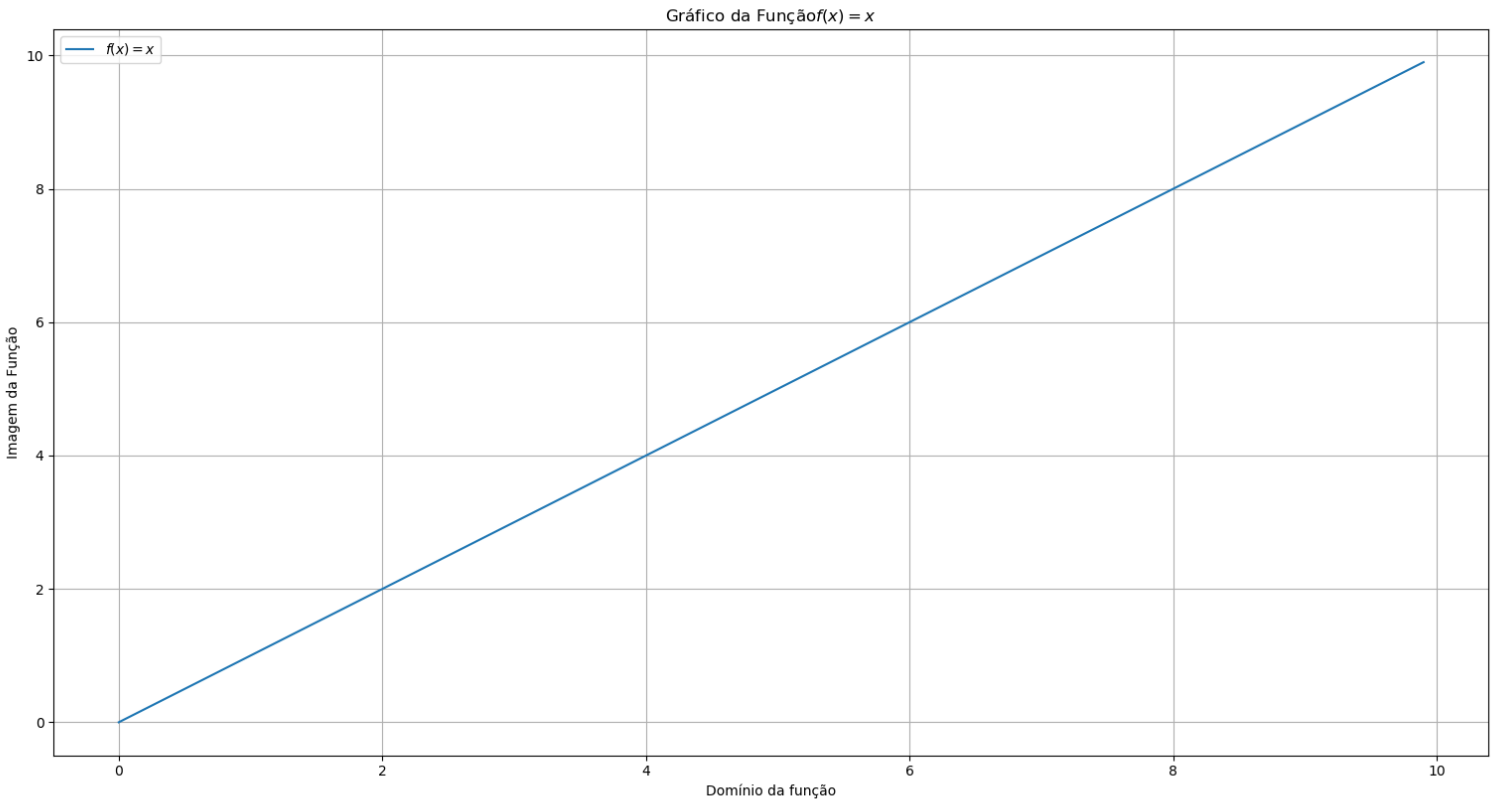
Adicionando a legenda

```
"""
    Autor: Daniel Saad Nogueira Nunes
    Comentários: Exemplo da plotagem da função £f(x) = x£ utilizando
    a biblioteca Matplotlib. Neste caso são adicionadas informações nos
    eixos x e y do gráfico e a legenda correspondente. Além disso, o grid
    é inserido para melhor visualização do gráfico.
"""

import matplotlib.pyplot as plt
import numpy as np

#   Cria um objeto figura e o coloca em fig
fig = plt.figure()
#   Da figura criada, é alocada uma matriz 1x1 de gráficos
#   ax corresponde a matriz 1
```

Adicionando a legenda

```
ax = fig.add_subplot(111);
#   Gera o dominio da função y = x
x  = np.arange(0,10,0.1) # domínio da função [0,10] com amostras de tamanho 0.1
#   Gera a imagem da função y = x
y = [ a for a in x ]
#   Plota a imagem e anota a legenda do gráfico
l = plt.plot(x,y,label=r"$f(x)=x$")
#   Coloca um título correspondente na figura
t = ax.set_title("Gráfico da Função"+r"$f(x) = x$")
#   Insere título no eixo x
t_x = ax.set_xlabel("Domínio da função")
#   Insere título no eixo y
t_y = ax.set_ylabel("Imagem da Função")
#   Insere  a legenda
l = ax.legend(loc='best')
#   Seta a visualização do Grid como verdadeira
ax.grid(True)
plt.show()
```

Adicionando a legenda



Marcadores nos eixos x e y

- É possível melhorar a precisão do *grid* ao definir o espaçamento entre os marcados dos eixos x e y.
- Basta indicar o intervalo dos "palitinhos" com os métodos `set_xticks` e `set_yticks`.
- Exemplo:
  - `ax.set_xtick(np.arange(0,10.5,0.5))`: os marcadores do eixo x avançam de 0.5 em 0.5.
  - `ax.set_ytick(np.arange(0,10.5,1))`: os marcadores do eixo x avançam de 1 em 1.

Marcadores nos eixos x e y

```
"""
    Autor: Daniel Saad Nogueira Nunes
    Comentários: Exemplo da plotagem da função £f(x) = x£ utilizando
    a biblioteca Matplotlib. Neste caso são adicionadas informações nos
    eixos x e y do gráfico e a legenda correspondente. Além disso, o grid
    é inserido para melhor visualização do gráfico. Cuidadosamente, os rótulos
    dos marcadores dos eixos x e y são configurados neste exemplo.
"""

import matplotlib.pyplot as plt
import numpy as np

#   Cria um objeto figura e o coloca em fig
fig = plt.figure()
#   Da figura criada, é alocada uma matriz 1x1 de gráficos
```

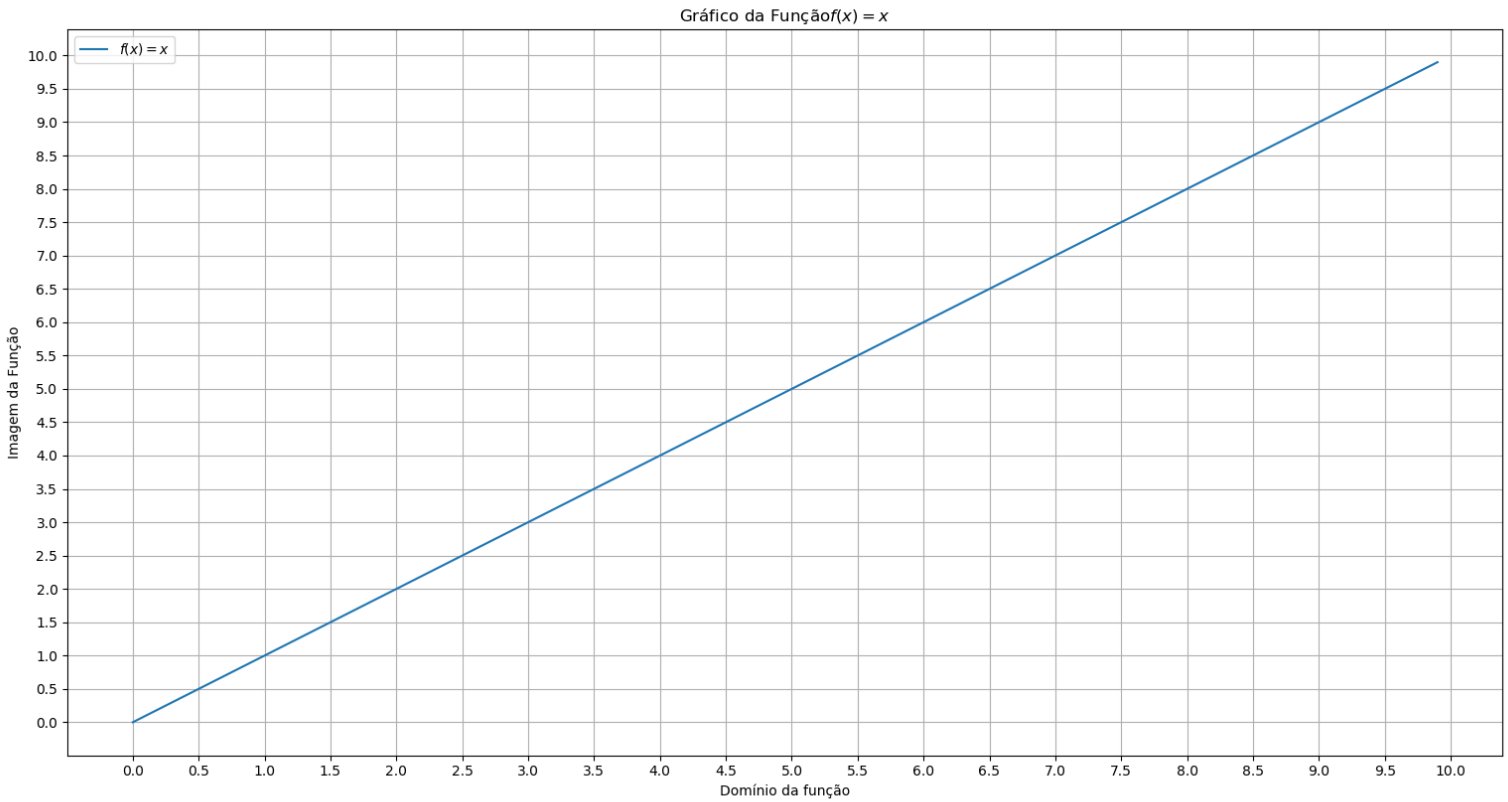
Marcadores nos eixos x e y

```
#   ax corresponde a matriz 1
ax = fig.add_subplot(111);
#   Gera o dominio da função y = x
x  = np.arange(0,10,0.1) # domínio da função [0,10] com amostras de tamanho 0.1
#   Gera a imagem da função y = x
y = [ a for a in x ]
#   Plota a imagem e anota a legenda do gráfico
l = plt.plot(x,y,label=r"$f(x)=x$")
#   Coloca um título correspondente na figura
t = ax.set_title("Gráfico da Função"+r"$f(x) = x$")
#   Insere título no eixo x
t_x = ax.set_xlabel("Domínio da função")
#   Insere título no eixo y
t_y = ax.set_ylabel("Imagem da Função")
```

Marcadores nos eixos x e y

```
#   Insere  a legenda
l = ax.legend(loc='best')
#   Seta a visualização do Grid como verdadeira
ax.grid(True)
#   Seta os marcadores de 0 a 10 com espaçamento de 0.5 no eixo x
ax.set_xticks(np.arange(0,10.5,0.5))
#   Seta os marcadores de 0 a 10 com espaçamento de 0.5 no eixo y
ax.set_yticks(np.arange(0,10.5,0.5))
#   Mostra a figura fig resultante
plt.show()
```

Adicionando a legenda



Plotando múltiplas curvas

- Também é possível em um mesmo gráfico, termos diversas curvas.
- Para isto, o método `plot()` é utilizado várias vezes de modo que cada vez contém a especificação de uma curva.
- Vamos plotar as funções:
  - $f(x) = x$;
  - $f(x) = \lceil x \rceil$;
  - $f(x) = \lfloor x \rfloor$.

Plotando múltiplas curvas

```
"""
    Autor: Daniel Saad Nogueira Nunes
    Comentários: Exemplo de múltiplas funções sob a mesma figura.
"""

import matplotlib.pyplot as plt
import numpy as np

#   Cria um objeto figura e o coloca em fig
fig = plt.figure()
#   Da figura criada, é alocada uma matriz 1x1 de gráficos
#   ax corresponde a matriz 1
ax = fig.add_subplot(111);

#   Gera o conjunto dominio [0,10) com amostras de tamanho 0.01
```

Plotando múltiplas curvas

```
x1 = x2 = x3  = np.arange(0,10,0.01)
#   Gera a imagem da função y = x
y1 = [ a for a in x1 ]
#   Gera a imagem da função y = \lceil x \rceil
y2 = [ np.ceil(a) for a in x2 ]
#   Gera a imagem da função y = \lfloor x \floor
y3 = [ np.floor(a) for a in x3 ]
#   Plota a imagem e anota a legenda do gráfico
l1 = plt.plot(x1,y1,label=r"$f(x) = x$")
l2 = plt.plot(x2,y2,label=r"$g(x)=\lceil x\rceil$")
l3 = plt.plot(x3,y3,label=r"$h(x)=\lfloor x \rfloor$")
#   Coloca um título correspondente na figura
t = ax.set_title("Comportamento das funções teto e piso")
#   Insere título no eixo x
```

Plotando múltiplas curvas

```
t_x = ax.set_xlabel("Domínio da função")
#   Insere título no eixo y
t_y = ax.set_ylabel("Imagem da Função")
#   Insere  a legenda
l = ax.legend(loc='best')
#   Seta a visualização do Grid como verdadeira
ax.grid(True)
#   Seta os marcadores de 0 a 10 com espaçamento de 0.5 no eixo x
ax.set_xticks(np.arange(0,10.5,0.5))
#   Seta os marcadores de 0 a 10 com espaçamento de 0.5 no eixo y
ax.set_yticks(np.arange(0,10.5,0.5))
#   Mostra a figura fig resultante
plt.show()
```

Personalizando as curvas.

- É possível personalizar as curvas plotadas, escolhendo as cores da mesma e o estilo (tracejado, pontos, diamantes,*etc*).
- Na especificação da função `plot()`, podemos especificar a cor e o estilo.
- Exemplo:

```
plt.plot(x3,y3,label=r"$h(x)=\lfloor x \rfloor$",color='blue',linestyle='--')
```

Modificando o estilo de curvas

```
"""
    Autor: Daniel Saad Nogueira Nunes
    Comentários: Exemplo de múltiplas funções sob a mesma figura.
    Manualmente escolhemos a cor e os estilos de cada função.
"""

import matplotlib.pyplot as plt
import numpy as np

#   Cria um objeto figura e o coloca em fig
fig = plt.figure()
#   Da figura criada, é alocada uma matriz 1x1 de gráficos
#   ax corresponde a matriz 1
ax = fig.add_subplot(111);
```

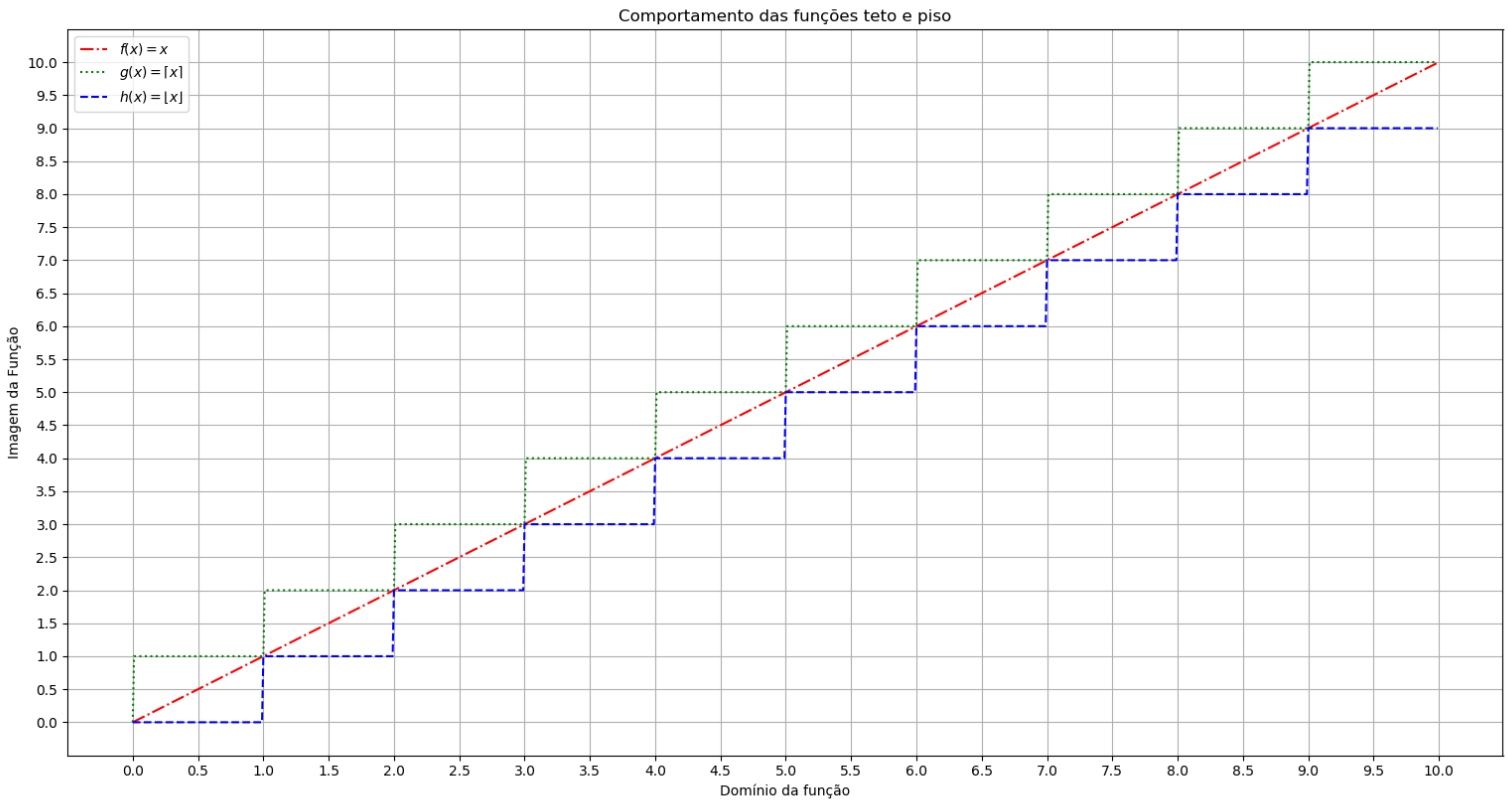
Modificando o estilo de curvas

```
#   Gera o conjunto dominio [0,10) com amostras de tamanho 0.01
x1 = x2 = x3  = np.arange(0,10,0.01)
#   Gera a imagem da função y = x
y1 = [ a for a in x1 ]
#   Gera a imagem da função y = \lceil x \rceil
y2 = [ np.ceil(a) for a in x2 ]
#   Gera a imagem da função y = \lfloor x \floor
y3 = [ np.floor(a) for a in x3 ]
#   Plota a imagem e anota a legenda do gráfico bem como as cores e estilos
#   de linha a serem utilizados
l1 = plt.plot(x1,y1,label=r"$f(x) = x$",color='red',linestyle='-.')
l2 = plt.plot(x2,y2,label=r"$g(x)=\lceil x\rceil$",color='green',linestyle=':')
l3 = plt.plot(x3,y3,label=r"$h(x)=\lfloor x \rfloor$",color='blue',linestyle='--')
#   Coloca um título correspondente na figura
```

Modificando o estilo de curvas

```
t = ax.set_title("Comportamento das funções teto e piso")
#   Insere título no eixo x
t_x = ax.set_xlabel("Domínio da função")
#   Insere título no eixo y
t_y = ax.set_ylabel("Imagem da Função")
#   Insere  a legenda
l = ax.legend(loc='best')
#   Seta a visualização do Grid como verdadeira
ax.grid(True)
#   Seta os marcadores de 0 a 10 com espaçamento de 0.5 no eixo x
ax.set_xticks(np.arange(0,10.5,0.5))
#   Seta os marcadores de 0 a 10 com espaçamento de 0.5 no eixo y
ax.set_yticks(np.arange(0,10.5,0.5))
#   Mostra a figura fig resultante
plt.show()
```

Alterando o estilo de curvas



Alterando os marcadores

- Também é possível alterar os marcadores das amostras.
- Da mesma forma, do exemplo anterior, na especificação da função `plot()`.
- Utilizamos `marker='<parâmetro>'`.

Modificando o estilo de curvas

```
"""
    Autor: Daniel Saad Nogueira Nunes
    Comentários: Exemplo de múltiplas funções sob a mesma figura.
    Manualmente escolhemos a cor e os estilos de cada função.
"""

import matplotlib.pyplot as plt
import numpy as np

#   Cria um objeto figura e o coloca em fig
fig = plt.figure()
#   Da figura criada, é alocada uma matriz 1x1 de gráficos
#   ax corresponde a matriz 1
ax = fig.add_subplot(111);
```

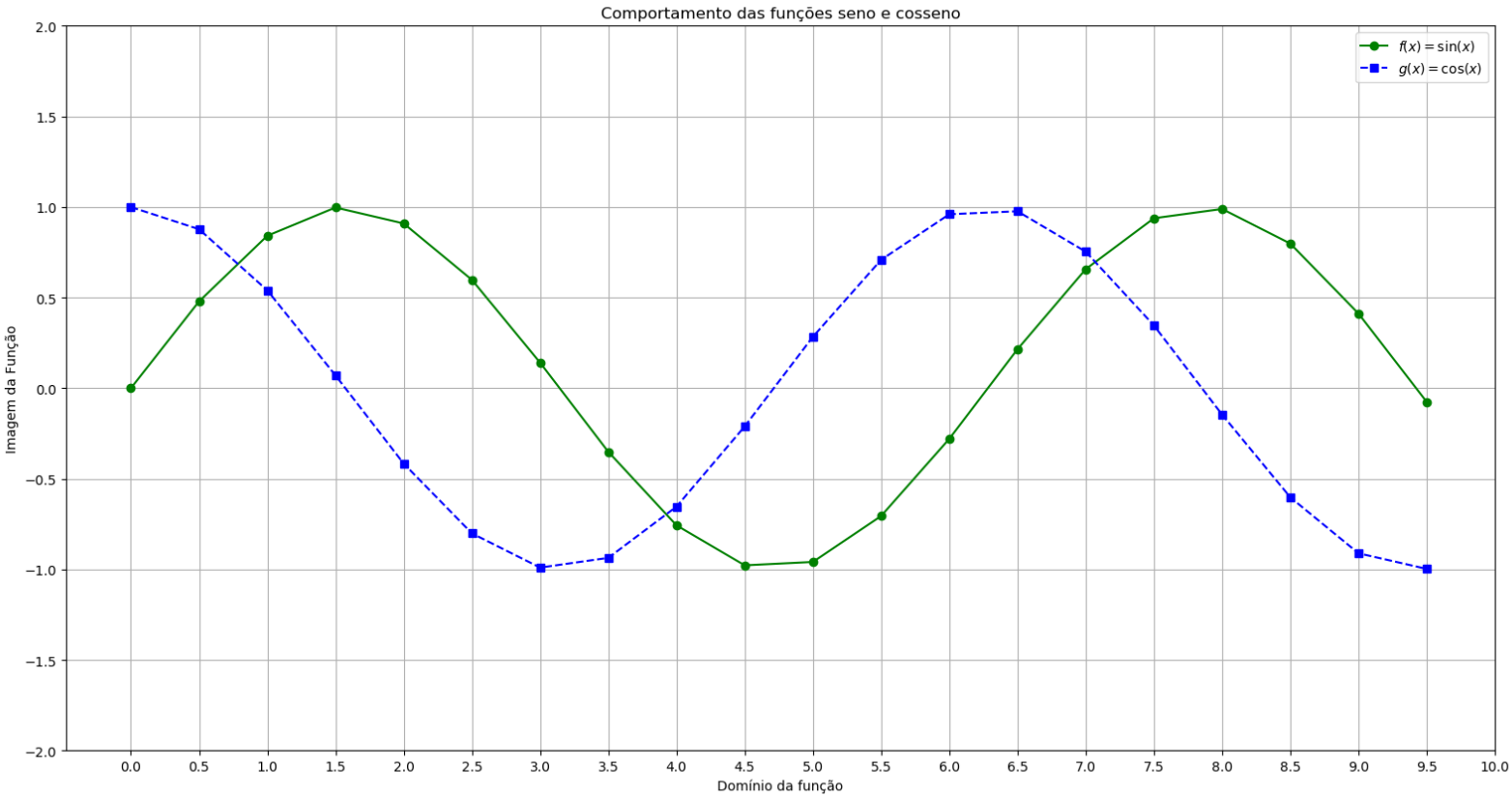
Modificando o estilo de curvas

```
#   Gera o conjunto dominio [0,10) com amostras de tamanho 0.5
x1 = x2   = np.arange(0,10,0.5)
#   Gera a imagem da função y = sin(x)
y1 = [ np.sin(a) for a in x1 ]
#   Gera a imagem da função y = cos(x)
y2 = [ np.cos(a) for a in x2 ]

#   Plota a imagem e anota a legenda do gráfico bem como as cores e estilos
#   de linha a serem utilizados
l1 = plt.plot(x1,y1,label=r"$f(x)=\sin(x)$",color='green',linestyle='-',marker='o')
l2 = plt.plot(x2,y2,label=r"$g(x)=\cos(x)$",color='blue',linestyle='--',marker='s')
#   Coloca um título correspondente na figura
t = ax.set_title("Comportamento das funções seno e cosseno")
#   Insere título no eixo x
```

Modificando o estilo de curvas

```
t_x = ax.set_xlabel("Domínio da função")
#   Insere título no eixo y
t_y = ax.set_ylabel("Imagem da Função")
#   Insere  a legenda
l = ax.legend(loc='best')
#   Seta a visualização do Grid como verdadeira
ax.grid(True)
#   Seta os marcadores de 0 a 10 com espaçamento de 0.5 no eixo x
ax.set_xticks(np.arange(0,10.5,0.5))
#   Seta os marcadores de 0 a 10 com espaçamento de 0.5 no eixo y
ax.set_yticks(np.arange(-2,2.5,0.5))
#   Mostra a figura fig resultante
plt.show()
```

Alterando o estilo de curvas



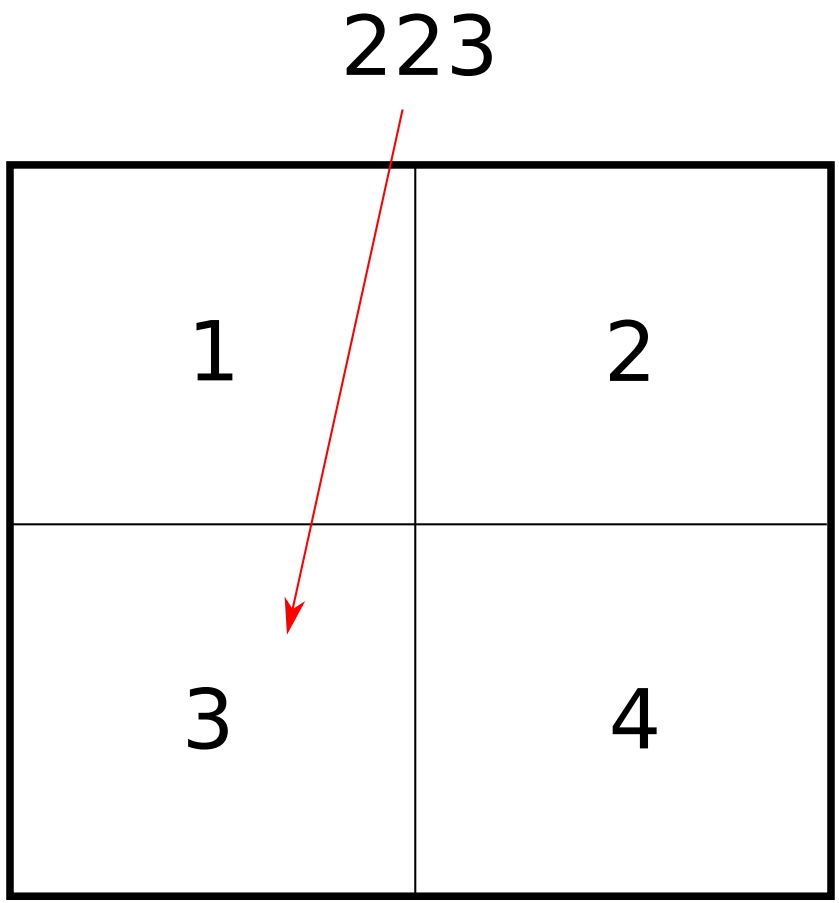
Múltiplos gráficos em uma única imagem

- Também é possível gerar diferentes gráficos em uma única imagem.
- Na definição dos eixos podemos ter o seguinte:

```
ax1 = fig.add_subplot(211)
ax2 = fig.add_subplot(212)
```

- O que significam esses números?
  - Número de linhas.
  - Número de colunas.
  - Índice da célula.

Múltiplos gráficos em uma única imagem



Múltiplos gráficos em uma única imagem

- Vamos alterar o exemplo do $\sin(x)$ e $\cos(x)$ de modo que cada um ocupe um gráfico individual na mesma figura.

Múltiplos gráficos em uma única imagem

```
"""
    Autor: Daniel Saad Nogueira Nunes
    Comentários: Exemplo de múltiplas funções sob a mesma figura.
    Manualmente escolhemos a cor e os estilos de cada função.
"""

import matplotlib.pyplot as plt
import numpy as np

#   Cria um objeto figura e o coloca em fig
fig = plt.figure()
#   Da figura criada, é alocada uma matriz 1x1 de gráficos
#   ax corresponde a matriz 1
ax1 = fig.add_subplot(211);
ax2 = fig.add_subplot(212)
```

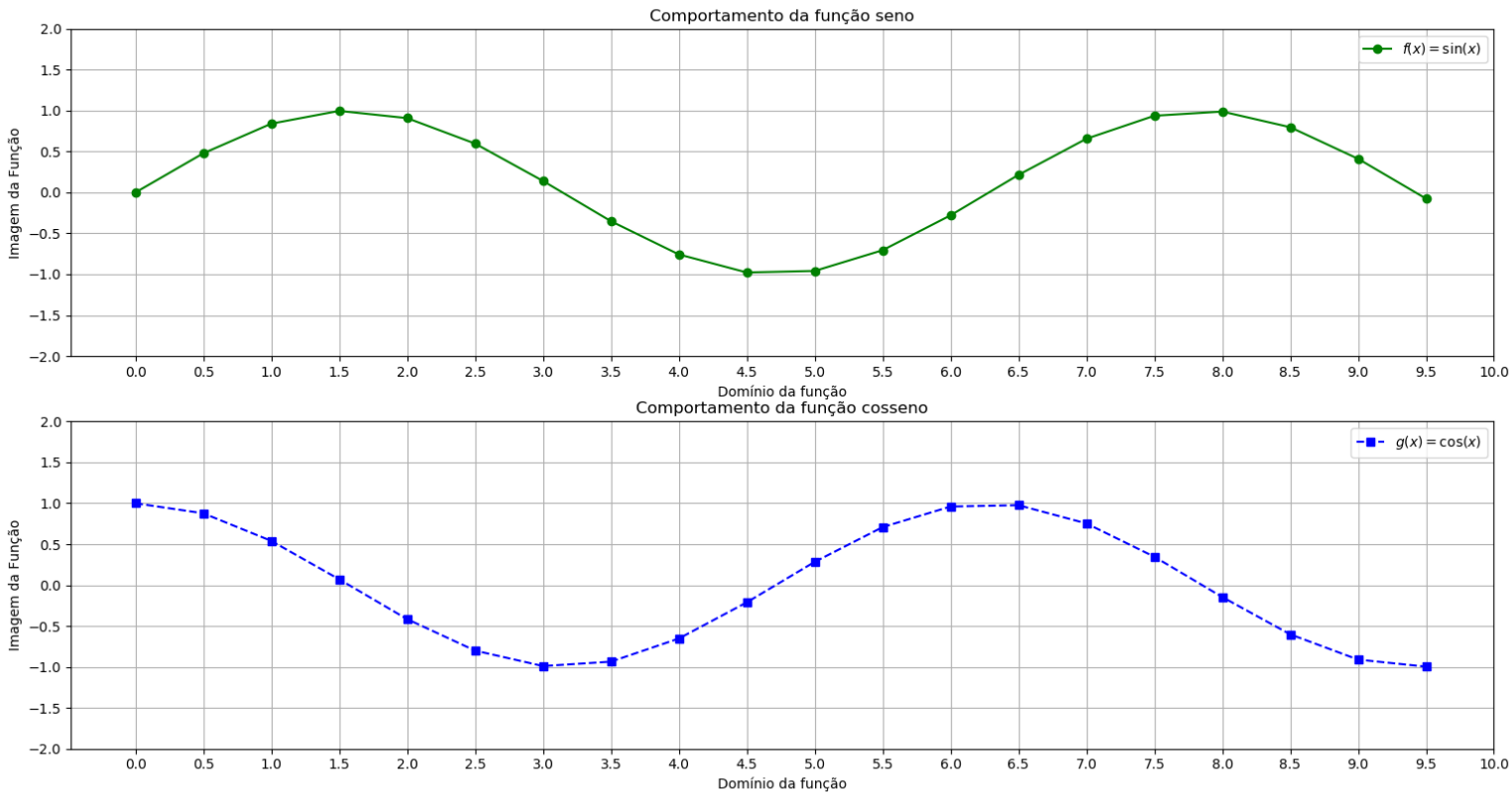
Múltiplos gráficos em uma única imagem

```
#   Gera o conjunto dominio [0,10) com amostras de tamanho 0.5
x1 = x2  = np.arange(0,10,0.5)
#   Gera a imagem da função f(x) = sin(x)
y1 = [ np.sin(a) for a in x1 ]
#   Gera a imagem da função g(x) = cos(x)
y2 = [ np.cos(a) for a in x2 ]

#   Plota a imagem e anota a legenda do gráfico bem como as cores e estilos
#   de linha a serem utilizados
l1 = ax1.plot(x1,y1,label=r"$f(x)=\sin(x)$",color='green',linestyle='-',marker='o')
l2 = ax2.plot(x2,y2,label=r"$g(x)=\cos(x)$",color='blue',linestyle='--',marker='s')
#   Coloca um título correspondente na figura
t1 = ax1.set_title("Comportamento da função seno")
t2 = ax2.set_title("Comportamento da função cosseno")
```

Múltiplos gráficos em uma única imagem

```
t_x1 = ax1.set_xlabel("Domínio da função")
#   Insere título no eixo y
t_y1 = ax1.set_ylabel("Imagem da Função")

t_x2 = ax2.set_xlabel("Domínio da função")
#   Insere título no eixo y
t_y2 = ax2.set_ylabel("Imagem da Função")
#   Insere  a legenda
l1 = ax1.legend(loc='best')
l2 = ax2.legend(loc='best')
#   Seta a visualização do Grid como verdadeira
ax1.grid(True)
ax2.grid(True);
#   Seta os marcadores de 0 a 10 com espaçamento de 0.5 no eixo x
ax1.set_xticks(np.arange(0,10.5,0.5))
ax2.set_xticks(np.arange(0,10.5,0.5))
#   Seta os marcadores de 0 a 10 com espaçamento de 0.5 no eixo y
ax1.set_yticks(np.arange(-2,2.5,0.5))
ax2.set_yticks(np.arange(-2,2.5,0.5))
#   Mostra a figura fig resultante
plt.show()
```

Múltiplos gráficos em uma única imagem



Salvando figuras

- Podemos salvar as figuras utilizando o método `savefig()` do objeto figura.
- Suporta vários formatos:
  - PNG;
  - PDF;
  - SVG;
  - *etc*.

Salvando Figuras

```
"""
    Autor: Daniel Saad Nogueira Nunes
    Comentários: Exemplo de múltiplas funções sob a mesma figura.
    Manualmente escolhemos a cor e os estilos de cada função.
"""

import matplotlib.pyplot as plt
import numpy as np

#   Cria um objeto figura e o coloca em fig
fig = plt.figure()
#   Da figura criada, é alocada uma matriz 1x1 de gráficos
#   ax corresponde a matriz 1
ax1 = fig.add_subplot(211);
ax2 = fig.add_subplot(212)
```

Salvando Figuras

```
#   Gera o conjunto dominio [0,10) com amostras de tamanho 0.5
x1 = x2   = np.arange(0,10,0.5)
#   Gera a imagem da função f(x) = sin(x)
y1 = [ np.sin(a) for a in x1 ]
#   Gera a imagem da função g(x) = cos(x)
y2 = [ np.cos(a) for a in x2 ]

#   Plota a imagem e anota a legenda do gráfico bem como as cores e estilos
#   de linha a serem utilizados
l1 = ax1.plot(x1,y1,label=r"$f(x)=\sin(x)$",color='green',linestyle='-',marker='o')
l2 = ax2.plot(x2,y2,label=r"$g(x)=\cos(x)$",color='blue',linestyle='--',marker='s')
#   Coloca um título correspondente na figura
t1 = ax1.set_title("Comportamento da função seno")
t2 = ax2.set_title("Comportamento da função cosseno")
```

Salvando Figuras

```
t_x1 = ax1.set_xlabel("Domínio da função")
#   Insere título no eixo y
t_y1 = ax1.set_ylabel("Imagem da Função")

t_x2 = ax2.set_xlabel("Domínio da função")
#   Insere título no eixo y
t_y2 = ax2.set_ylabel("Imagem da Função")
#   Insere  a legenda
l1 = ax1.legend(loc='best')
l2 = ax2.legend(loc='best')
#   Seta a visualização do Grid como verdadeira
ax1.grid(True)
ax2.grid(True);
#   Seta os marcadores de 0 a 10 com espaçamento de 0.5 no eixo x
ax1.set_xticks(np.arange(0,10.5,0.5))
ax2.set_xticks(np.arange(0,10.5,0.5))
#   Seta os marcadores de 0 a 10 com espaçamento de 0.5 no eixo y
ax1.set_yticks(np.arange(-2,2.5,0.5))
ax2.set_yticks(np.arange(-2,2.5,0.5))
#   Mostra a figura fig resultante
fig.show()
fig.savefig('seno_cosseno.png',dpi=600)
fig.savefig('seno_cosseno.pdf')
fig.savefig('seno_cosseno.svg')
```

Sumário

2 Matplotlib
- Primeiros passos
- Tipos de gráficos
- Histogramas
- Gráficos de barra
- Gráficos de pizza
- Gráficos de dispersão

Tipos de gráficos

- O `Matplotlib` não se resume a gerar apenas gráficos de linha.
- Podemos gerar diferentes tipos de gráfico, dependendo da aplicação.
- Vamos ver algum deles?

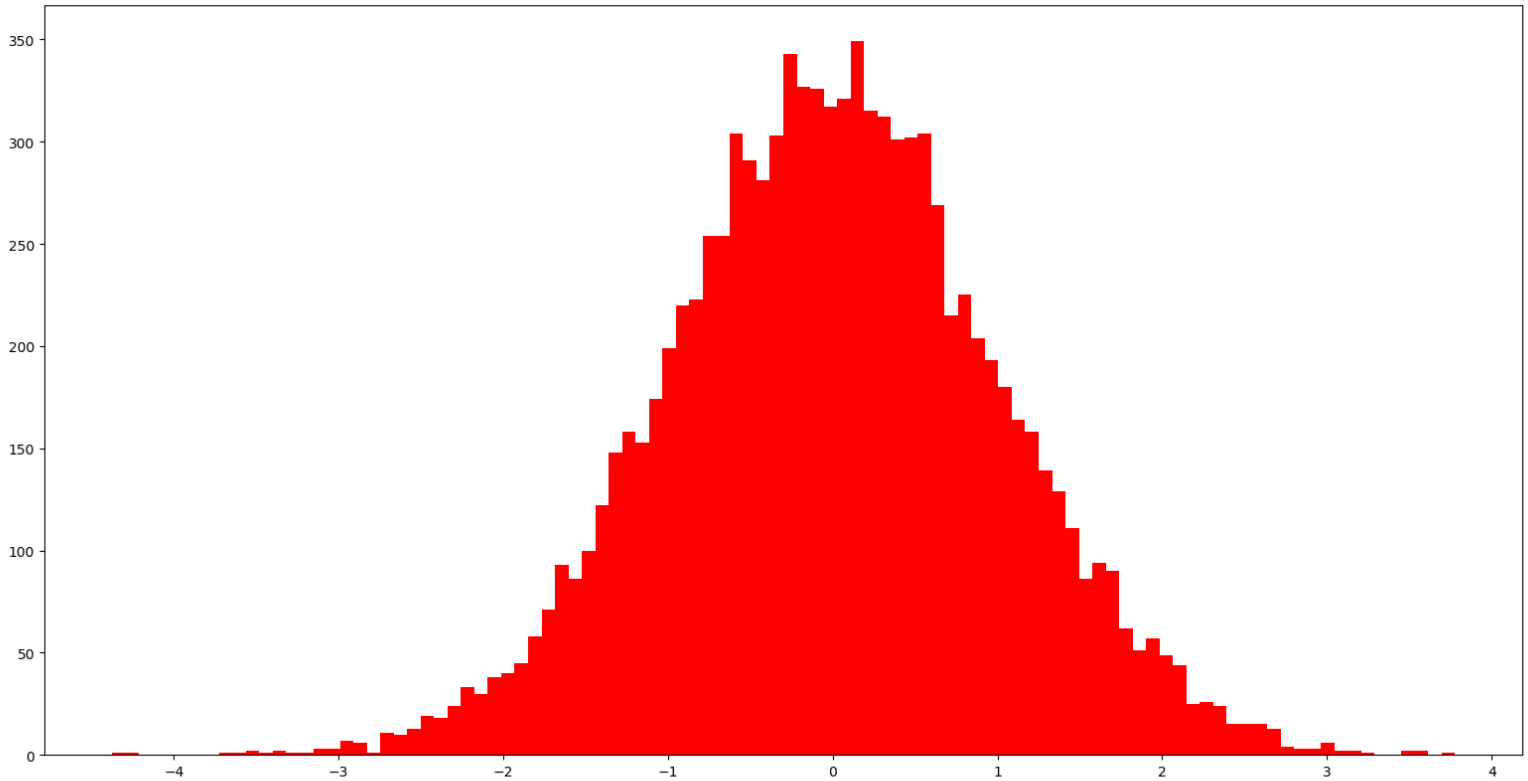
Sumário

2 Matplotlib
- Primeiros passos
- Tipos de gráficos
- Histogramas
- Gráficos de barra
- Gráficos de pizza
- Gráficos de dispersão

Histogramas

- Objetivo: classificar uma única variável em diversas categorias.

Histogramas



Histogramas

```
import matplotlib.pyplot as plt
import numpy as np
#   cria uma figura
fig = plt.figure()
#   cria um objeto axis na posição 111
ax = fig.add_subplot(111)
#   Gera de acordo com a distribuição normal de média 0 e variância 1
#   100 amostras
values = np.random.randn(10000)
#   Classifica estas 100000 amostras em 100 categorias
ax.hist(values,100,color='red')
#   Mostra o gráfico
plt.show()
```

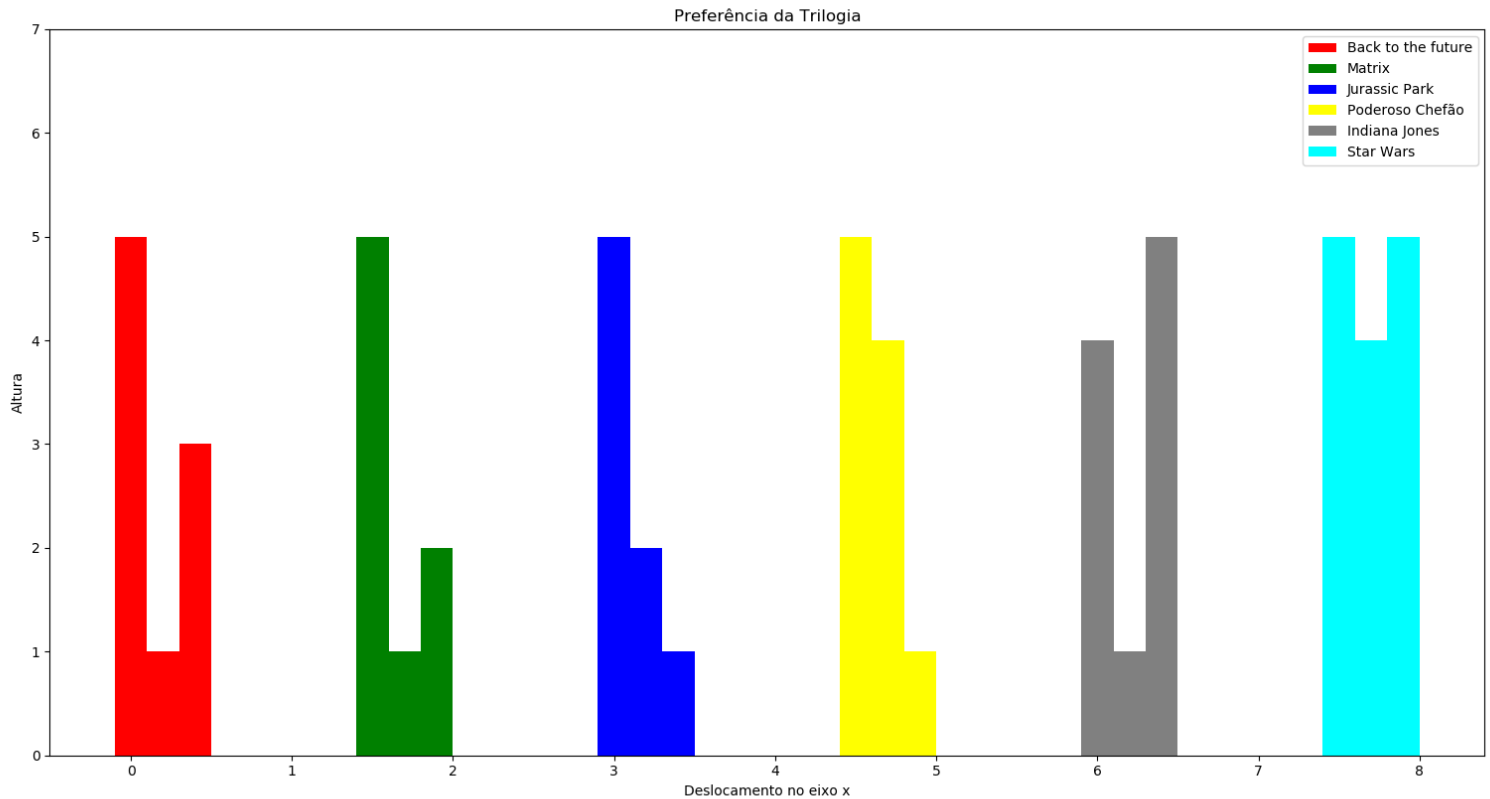
Sumário

2 Matplotlib
- Primeiros passos
- Tipos de gráficos
- Histogramas
- Gráficos de barra
- Gráficos de pizza
- Gráficos de dispersão

Gráficos de barra

- Gráficos em barra representam categorias em cada barra de modo que a altura de cada uma representa o valor associado a mesma.

Gráficos de barra



Gráficos de barra

- Os gráficos em barra exigem um pouco mais de cuidado.
- Para cada barra, precisamos especificar duas coisas:
  - Altura.
  - Deslocamento na horizontal.

Gráficos de barra

```
import matplotlib.pyplot as plt
import numpy as np
#   cria uma figura
fig = plt.figure()
#   cria um objeto axis na posição 111
ax = fig.add_subplot(111)


# Preferências do De Volta para o Futuro
```

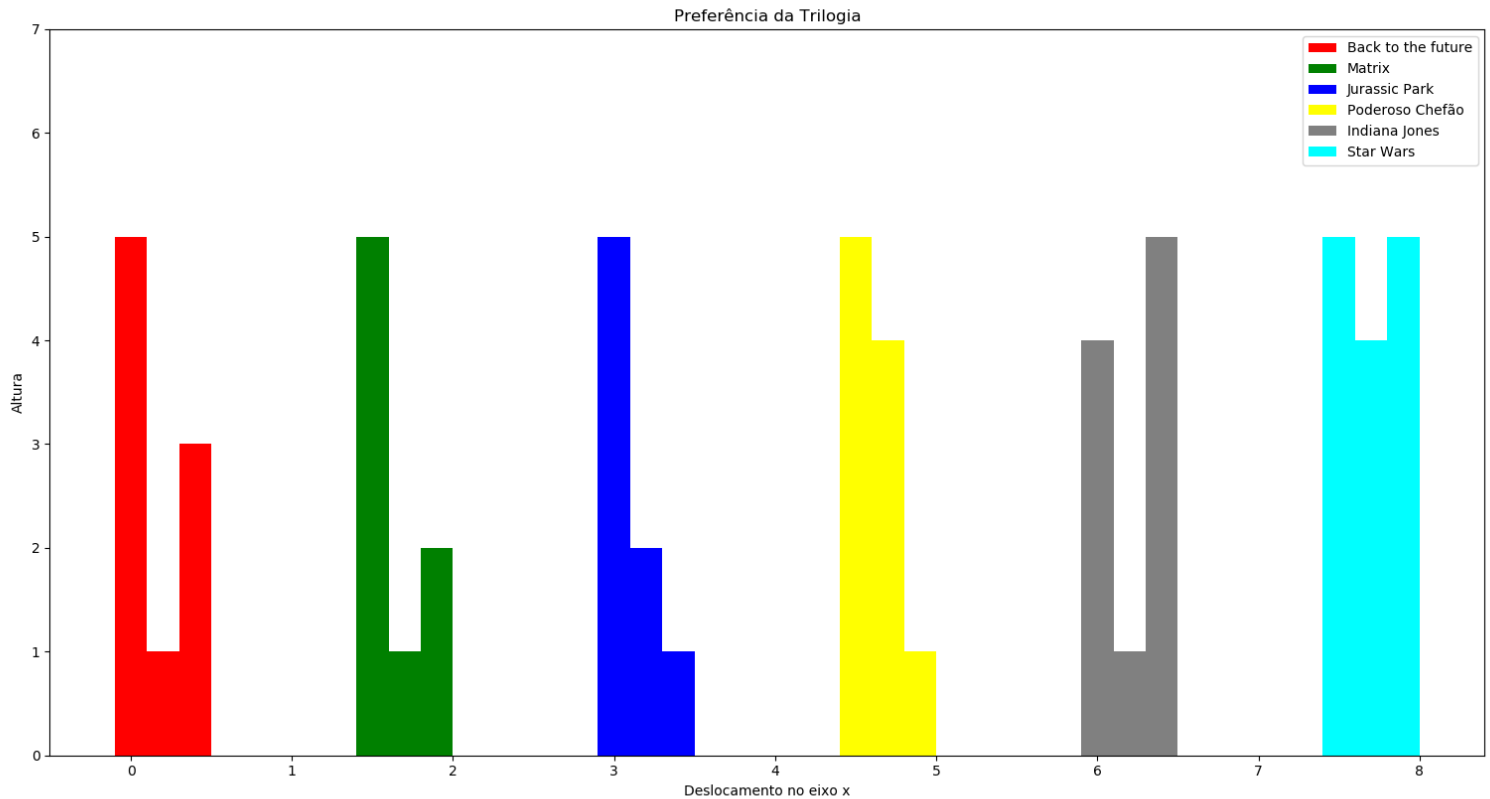
Gráficos de barra

```
b2tf = [5, 1, 3]
# Preferências do Matrix
matrix = [5, 1, 2]
# Preferências do Jurassic Park
jp = [5, 2, 1]
# Preferências do Poderoso Chefão
godfather = [5, 4, 1]
# Preferências do Indiana Jones
indiana_jones = [4, 1, 5]
# Preferências do Star Wars
sw = [5, 4, 5]
# Faz uma lista das preferências de cada trilogia
movies = [b2tf, matrix, jp, godfather, indiana_jones, sw]
# Especifica as cores para cada trilogia
```

Gráficos de barra

```
colors = ['red', 'green', 'blue', 'yellow', 'gray', 'cyan']
# Especifica as legendas para cada trilogia
labels = ['Back to the future', 'Matrix', 'Jurassic Park',
          'Poderoso Chefão', 'Indiana Jones', 'Star Wars']
# Largura das barras
width = 0.2
# Espaço entre barras de diferentes trilogias
gap = .5
# Para cada trilogia, plota um conjunto de três barras.
for i, movie in enumerate(movies):
    locs = [i+i*gap+x*width for x in range(len(movie))]
    ax.bar(locs, movie, width=width, color=colors[i], label=labels[i])
ax.legend(loc='best')
ax.set_yticks(np.arange(0, 8, 1))
ax.set_title("Preferência da Trilogia")
ax.set_xlabel("Deslocamento no eixo x")
ax.set_ylabel("Altura")
plt.show()
```

Gráficos de barra



Sumário

2 Matplotlib
- Primeiros passos
- Tipos de gráficos
- Histogramas
- Gráficos de barra
- Gráficos de pizza
- Gráficos de dispersão

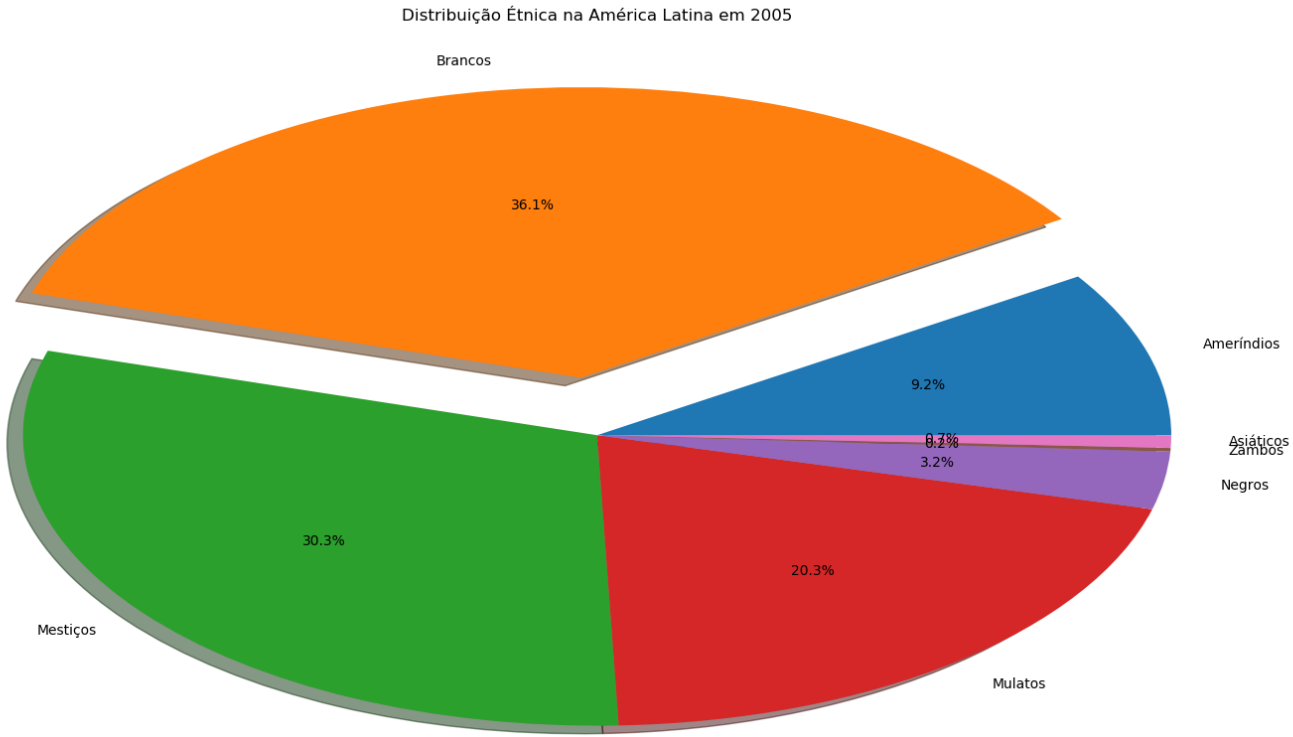
Gráficos de pizza

- Gráficos em pizza podem ser utilizados para representar proporções ou porcentagens.

Gráficos de pizza

```
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_subplot(111)
porcentagem = [9.2, 36.1, 30.3, 20.3, 3.2, 0.2, 0.7]
labels = ['Ameríndios', 'Brancos', 'Mestiços',
          'Mulatos', 'Negros', 'Zambos', 'Asiáticos']
explode = [0, 0.2, 0, 0, 0, 0, 0]
ax.set_title("Distribuição Étnica na América Latina em 2005", y=1.06)
ax.pie(porcentagem, labels=labels, explode=explode, shadow=True,
       autopct='%1.1f%%')
plt.show()
```

Gráficos em pizza



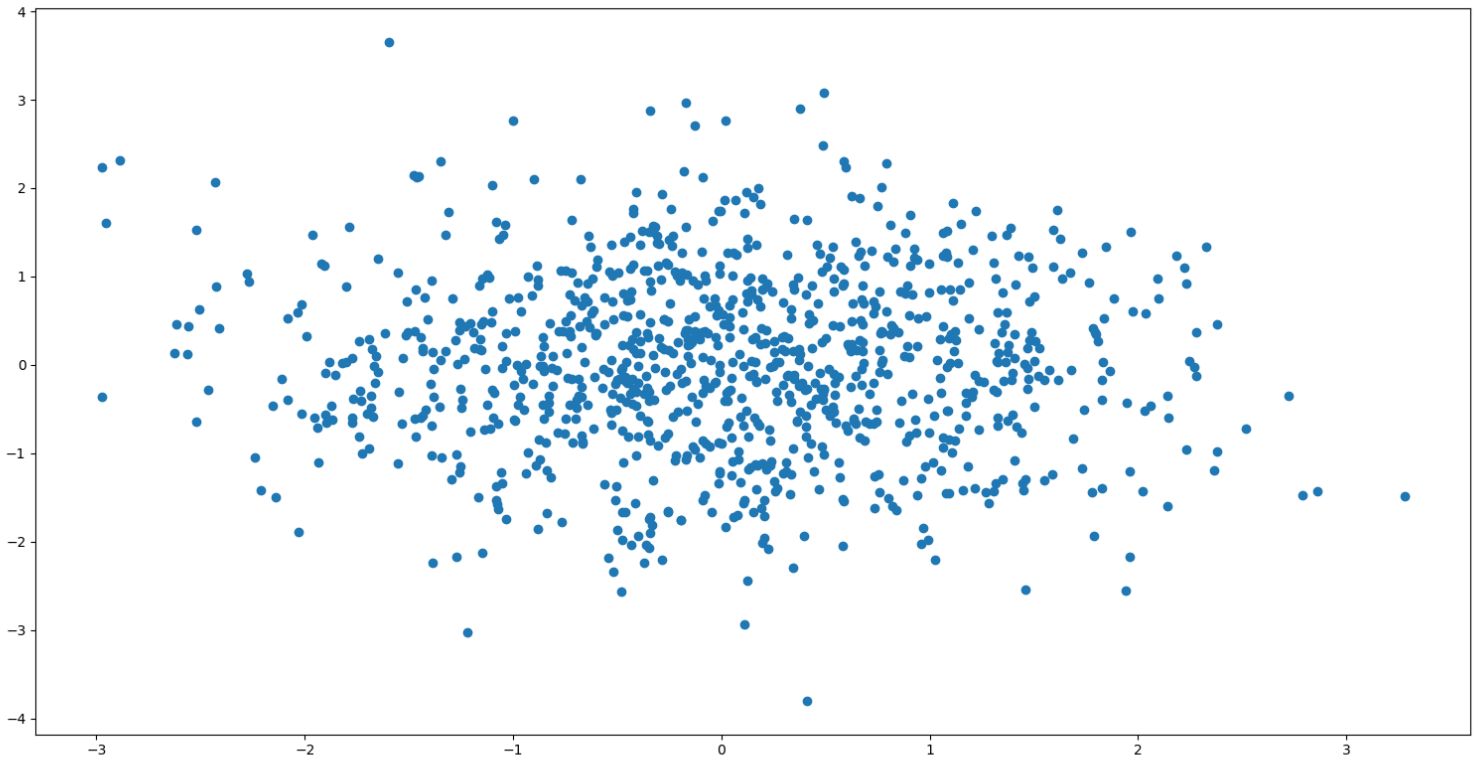
Gráficos de dispersão

- Utilizados para verificar relação de causa e efeito entre duas variáveis.

Gráficos de dispersão

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

fig = plt.figure()
ax = fig.add_subplot(111)
x = np.random.randn(1000)
y = np.random.randn(1000)
ax.scatter(x,y)
plt.show()
```

Gráficos de dispersão



Sumário

3 CSV

CSV

- Até agora vimos como plotar dados de maneira manual.
- Isto é, estamos digitando os valores para que o gráfico seja gerado.
- Há como automatizar esse processo.
- Basta ler os dados de um arquivo, como uma planilha.
- Formato portável: csv.

CSV

- CSV significa *comma-separated-values*.
- Arquivos .csv são arquivos texto cujos valores estão separados por vírgulas (ou outro separador).
- São arquivos portáveis.
- Podem ser abertos e editados utilizando softwares de visualização e manipulação de planilhas.

CSV

- Tome o seguinte exemplo: temos um arquivo `.csv` que possui o nome de pessoas, o time que elas torcem e a idade delas.
- Vamos verificar como isto está representado em modo texto.

CSV

```
Nome,Time,Idade
Astrogildo,Flamengo,25
Epaminondas,Vasco,38
Godofreda,Botafogo,85
Hortolina,Íbis,47
Rogergilce,Paysandu,10
```

CSV

- O usuário pode editar esse arquivo utilizando um editor de texto comum ou um software de manipulação de planilhas (Excel, Libreoffice).

CSV

	A	B	C
1	Nome	Time	Idade
2	Astrogildo	Flamengo	25
3	Epaminondas	Vasco	38
4	Godofreda	Botafogo	85
5	Hortolina	Íbis	47
6	Rogergilce	Paysandu	10

CSV

- O módulo `csv` do Python nos permite manipular arquivos *csv* com extrema facilidade.
- Podemos rodar experimentos e gravar os resultados em arquivos *csv*.
- Podemos ler arquivos *csv* para plotar nossos dados!

Leitura CSV

```
import csv
# abre o arquivo 'arquivo.csv' em modo leitura

nome = []
time = []
idade = []
with open('arquivo.csv') as csvfile:
    # cria um leitor csv do arquivo
    leitor_csv = csv.reader(csvfile,delimiter=',')
    # ignora o cabeçalho
    next(leitor_csv,None)
    # para cada linha deste arquivo, extrai o campo
    for linha in leitor_csv:
        nome.append(linha[0])
        time.append(linha[1])
        idade.append(linha[2])
```

Leitura CSV

```
print("Imprimindo os dados")
for i in range(len(nome)):
    print("Nome = ",nome[i],"Time = ",time[i],"Idade = ",idade[i])
```

Escrita CSV

```
import csv

# Valores iniciais da sequência de Fibonacci
f1 = 1
f2 = 1
# abre um arquivo no modo de escrita

with open('fib.csv', 'w') as csvfile:
    # cria um escritor csv do arquivo
    escritor_csv = csv.writer(csvfile, delimiter=',')
    # Escreve o cabeçalho
    escritor_csv.writerow(['Índice', 'Número de Fibonacci', 'Razão de Ouro'])
    # Escreve os dois primeiros números de Fibonacci e a razão entre eles
    escritor_csv.writerow(['1', str(f1),''])
    escritor_csv.writerow(['2', str(f2),'1'])
    # Faça 1000 iterações e preencha os números de Fibonacci no arquivo csv
```

Escrita CSV

```
for i in range(3,1000):
    # Escreve em uma string o índice, o próximo número de fibonacci e a razão
    string = "{} {} {}".format(i,f1+f2,(f1+f2)/float(f2))
    # Transforma a string em uma lista
    linha = string.split(' ')
    # Escreve a linha no arquivo csv
    escritor_csv.writerow(linha)
    # Calcula o próximo número de Fibonacci
    temp = f1
    f1 = f2
    f2 = temp +f2
```