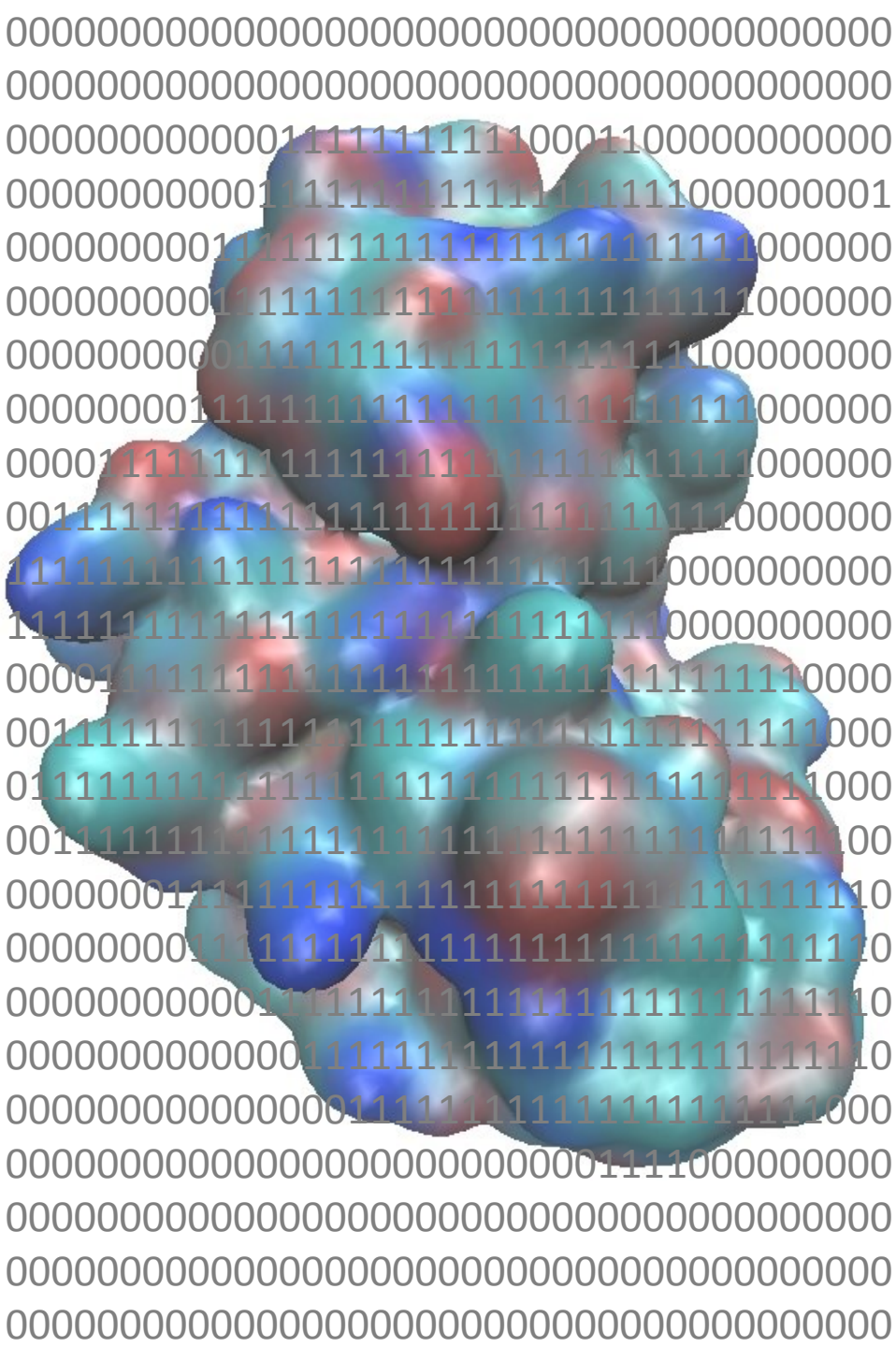
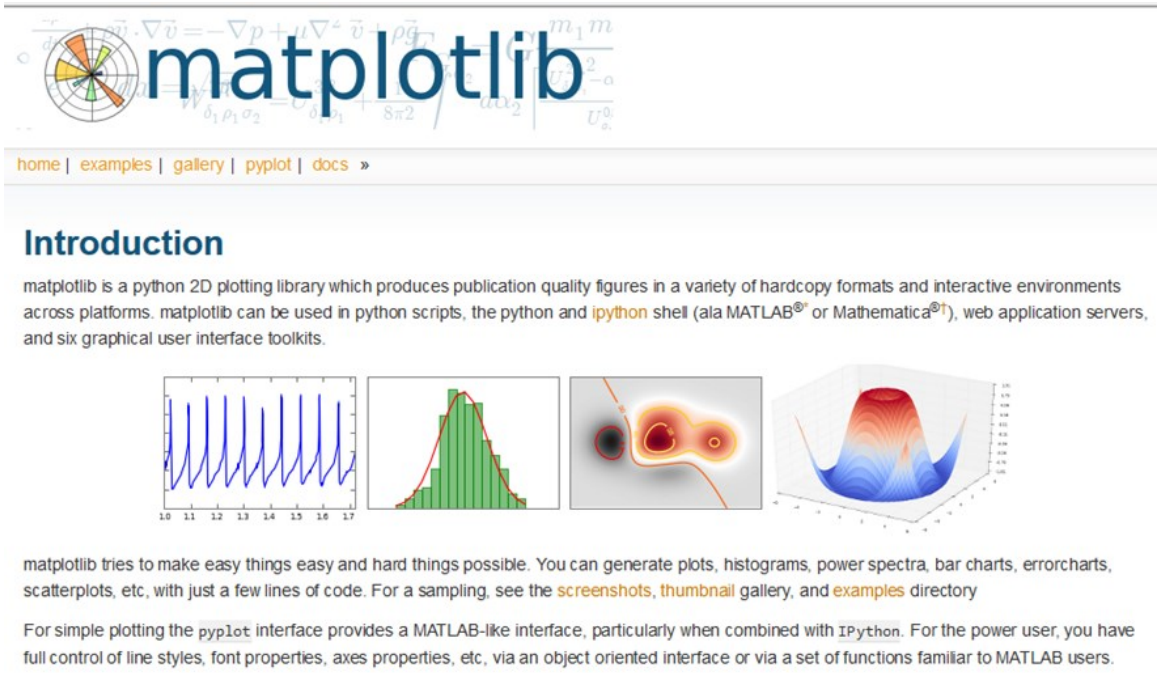


python™
www.python.org

Introdução ao Matplotlib

© 2019 Dr. Walter F. de Azevedo Jr.

Uma das características da linguagem de programação Python, que tem atraído um grande número de fãs na comunidade de computação científica é a possibilidade de uso de bibliotecas gratuitas para gráficos e análise numérica. Nesta texto, introduziremos as bibliotecas *Matplotib*. Ao instalarmos o Python, por meio do *Pyzo*, já temos estas bibliotecas instaladas. Veremos alguns recursos das bibliotecas *Matplotlib* e *NumPy*.



Página de entrada do matplotlig: http://matplotlib.org/
Acesso em: 02 de abril de 2019.

Vejamos o uso das bibliotecas *Matplotlib* e *NumPy*, para gerarmos o gráfico do seno. A primeira linha de código importa a biblioteca *matplotlib* para gráfico, como *plt*. Assim, todos os recursos desta biblioteca podem ser chamados a partir da notação *dot* (.). O mesmo acontece na segunda linha de código, com a biblioteca *numpy*, importada como *np*. O método *.linspace()*, da biblioteca *numpy,* gera um conjunto de números *floats*, entre os intervalos mostrados, no caso, entre 0 e 2*Pi. O valor de Pi pode ser chamado com *numpy*, por meio de *np.py*. O último número, dentro dos parênteses do método *linspace()*, indica o número de pontos gerados, entre os valores máximo e mínimo.

```
import matplotlib.pyplot as plt
import numpy as np
# Defines x range with linspace
x = np.linspace(0, 2*np.pi, 100)
# Defines sine function
sine_func = np.sin(x)
# Creates plot
plt.plot(x, sine_func)
# Shows plot
plt.show()
# Saves plot on png file
plt.savefig('sine1.png')
```

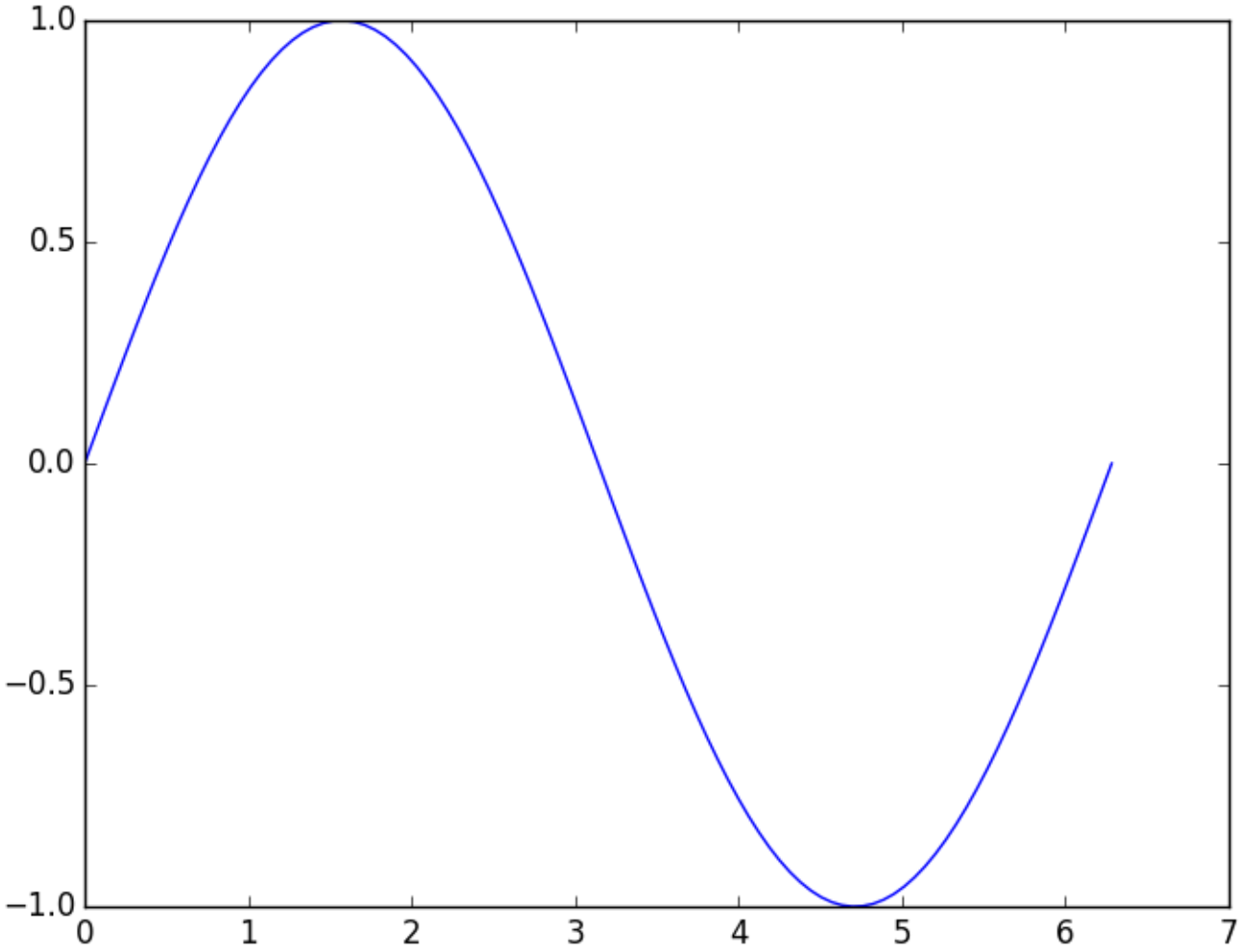
Os números gerados com o método *np.linspace(0,2*np.pi,100)* são atribuídos à variável *x*. Veja que o último número, 100, indica o número de *floats* gerados no intervalo estabelecido, pelos dois primeiros números, dentro dos parênteses.
A linha de comando *sine_func = np.sin(x),* define a função seno, a partir da biblioteca *numpy*. O resultado é atribuído à variável *sine_func*. O método *plt.plot(x,sine_func)*, da biblioteca *matplotlib*, gera o gráfico cartesiano, os valores do eixo x são aqueles atribuídos à variável *x*, e os valores do eixo y, são aqueles atribuídos à variável *sine_func*.

```
import matplotlib.pyplot as plt
import numpy as np
# Defines x range with linspace
x = np.linspace(0, 2*np.pi, 100)
# Defines sine function
sine_func = np.sin(x)
# Creates plot
plt.plot(x, sine_func)
# Shows plot
plt.show()
# Saves plot on png file
plt.savefig('sine1.png')
```

O método *plt.show()* mostra o gráfico na tela. Este método pode ter resultados diferentes, conforme a instalação da biblioteca *Matplotlib* e o sistema operacional que você está utilizando. O método *plt.savefig('sine1.png')* salva o gráfico gerado no arquivo *sine1.png*. Execute o código *sine1.py* e clique na pasta da aula de hoje. Localize o arquivo *sine1.png* e clique duas vezes neste. Veja o gráfico gerado.

```
import matplotlib.pyplot as plt
import numpy as np
# Defines x range with linspace
x = np.linspace(0, 2*np.pi, 100)
# Defines sine function
sine_func = np.sin(x)
# Creates plot
plt.plot(x, sine_func)
# Shows plot
plt.show()
# Saves plot on png file
plt.savefig('sine1.png')
```

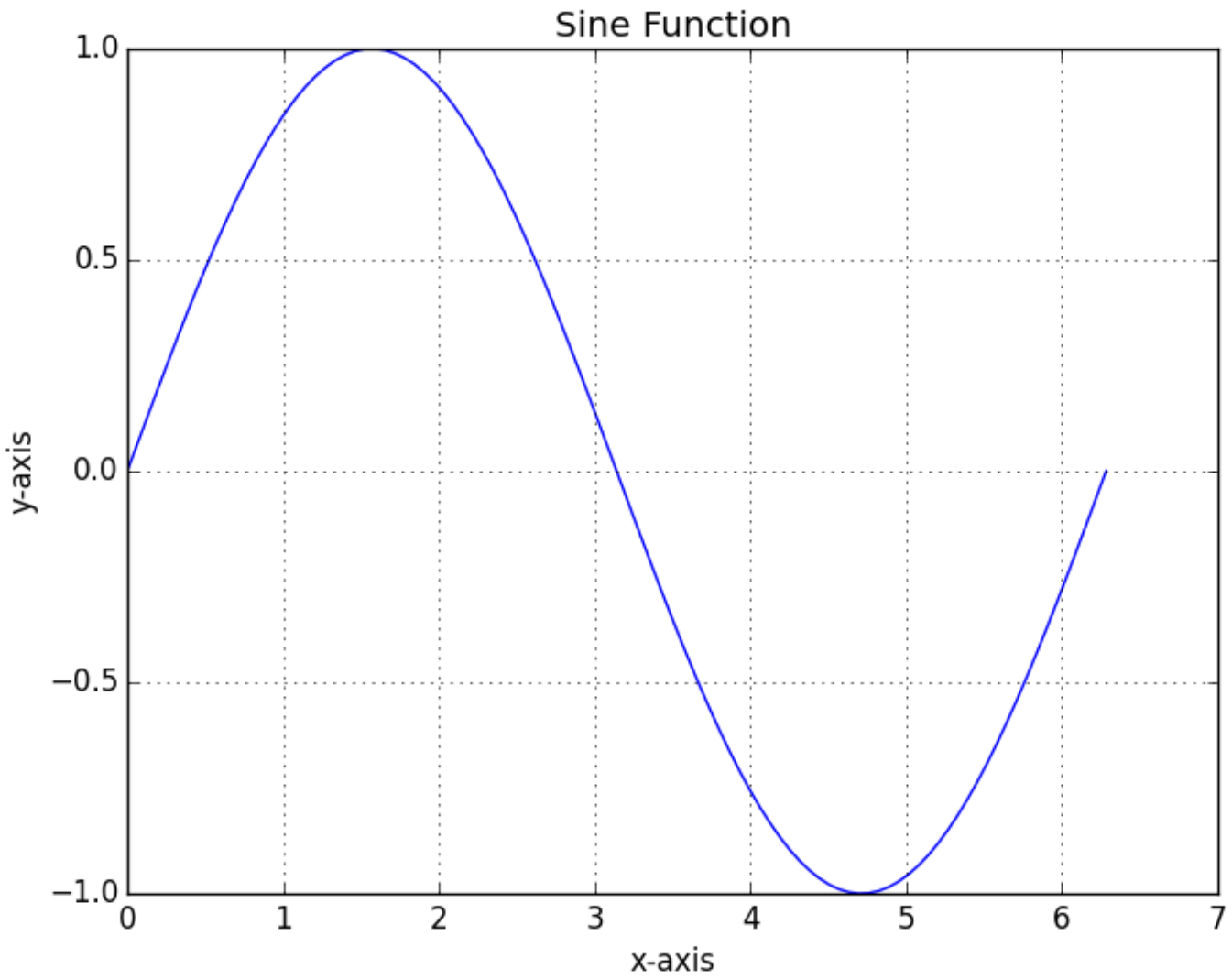
Abaixo temos o gráfico da função seno, gerado pelo programa *sine1.py*.



Como em toda biblioteca gráfica, podemos adicionar os nomes dos eixos, gradeado e título, com os métodos destacados em vermelho no código *sine2.py*. Os métodos *plt.xlabel()*, *plt.ylabel()* e *plt.title()* aceitam variáveis como argumentos, os strings, como no código abaixo.

```
import matplotlib.pyplot as plt
import numpy as np
# Defines x range with linspace
x = np.linspace(0, 2*np.pi, 100)
# Defines sine function
sine_func = np.sin(x)
# Creates plot
plt.plot(x, sine_func)
plt.xlabel('x-axis')          # Adds axis label
plt.ylabel('y-axis')          # Adds axis label
plt.title('Sine Function')    # Adds title
plt.grid(True)                # Adds grid to the plot
# Shows plot
plt.show()
# Saves plot on png file
plt.savefig('sine2.png')
```

Abaixo temos o gráfico da função seno, gerado pelo programa *sine2.py*.



O programa *trig1.py* gera o gráfico para as funções seno e cosseno. Para isto basta adicionarmos a variável *cos_func*, e atribuirmos a ela o resultado do método *np.cos(x)*.

```
import matplotlib.pyplot as plt
import numpy as np
# Defines x range with linspace
x = np.linspace(0, 2*np.pi, 100)
# Defines sine and cosine functions
sine_func = np.sin(x)
cos_func = np.cos(x)
# Creates plots
plt.plot(x, sine_func)
plt.plot(x, cos_func)
plt.xlabel('x-axis')          # Adds axis label
plt.ylabel('y-axis')          # Adds axis label
plt.title('Sine and Cosine Functions')  # Adds title
plt.grid(True)                # Adds grid to the plot
# Shows plot
plt.show()
# Saves plot on png file
plt.savefig('trig1.png')
```

Temos que adicionar o método *plt.plot(x,cos_func)* para o gráfico do cosseno, como indicado abaixo. Mudamos também o título do gráfico.

```
import matplotlib.pyplot as plt
import numpy as np
# Defines x range with linspace
x = np.linspace(0, 2*np.pi, 100)
# Defines sine and cosine functions
sine_func = np.sin(x)
cos_func = np.cos(x)
# Creates plots
plt.plot(x, sine_func)
plt.plot(x, cos_func)
plt.xlabel('x-axis')          # Adds axis label
plt.ylabel('y-axis')          # Adds axis label
plt.title('Sine and Cosine Functions')  # Adds title
plt.grid(True)                # Adds grid to the plot
# Shows plot
plt.show()
# Saves plot on png file
plt.savefig('trig1.png')
```

Abaixo temos o gráfico das funções seno cosseno, geradas pelo programa *trig1.py*. Veja que linha do cosseno é verde.

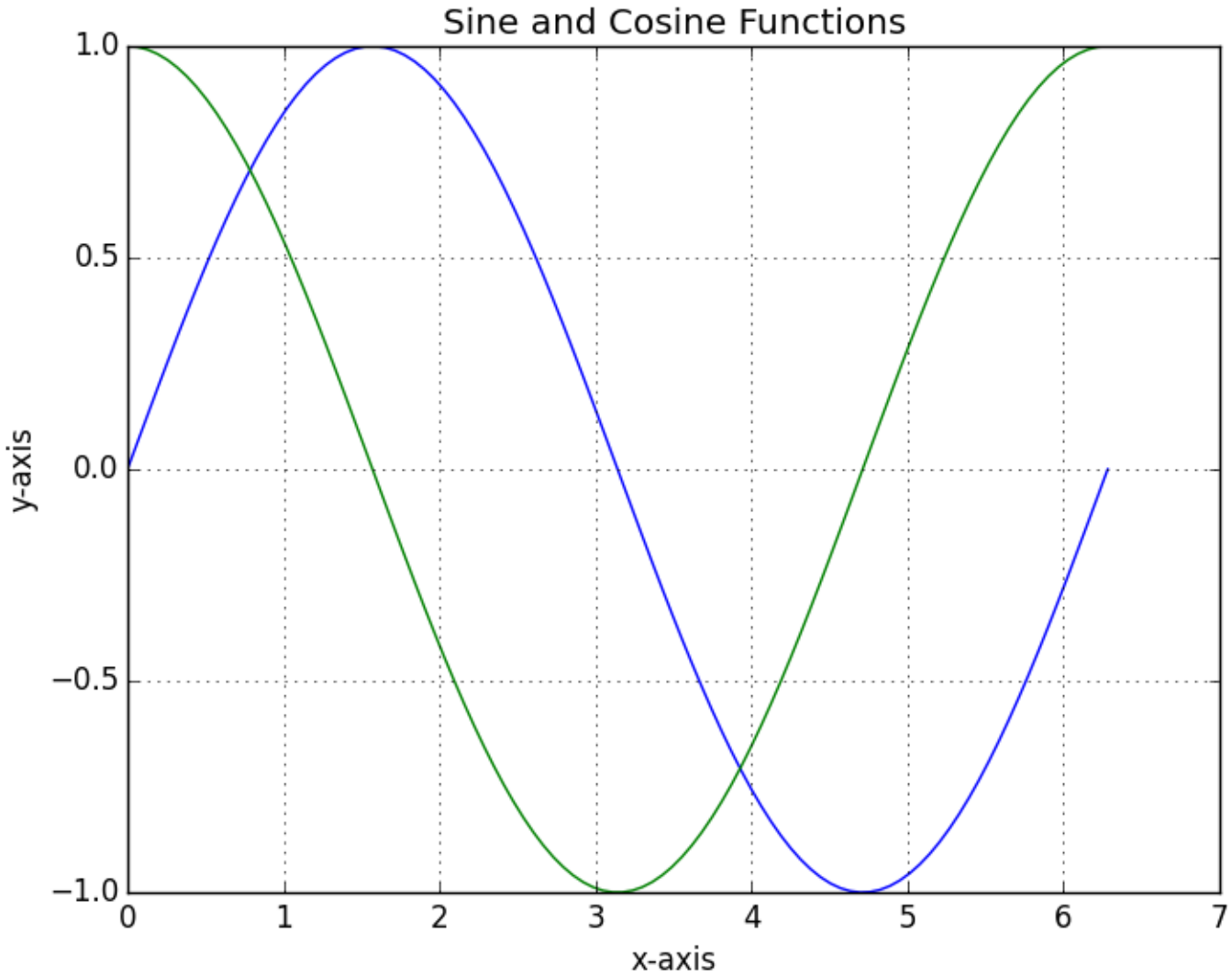


Gráfico das funções seno e cosseno

Programa: *trig2.py*

Resumo
Programa para gerar os gráficos da função seno e cosseno, a partir das bibliotecas *Matplotlib* e *NumPy.* O usuário digita a faixa de valores para o eixo x e os títulos dos eixos x,y, bem como o título do gráfico e o nome do arquivo de saída. Ambas funções são colocadas no mesmo gráfico. O programa gera o arquivo de saída no formato PNG.

-BRESSERT, Eli. SciPy and NumPy. Sebastopol: O'Reilly Media, Inc., 2013. 56 p.

-DAWSON, Michael. **Python Programming, for the absolute beginner.** 3ed. Boston: Course Technology, 2010. 455 p.

-HETLAND, Magnus Lie. **Python Algorithms. Mastering Basic Algorithms in the Python Language.** Nova York: Springer Science+Business Media LLC, 2010. 316 p.

-IDRIS, Ivan. **NumPy 1.5. An action-packed guide dor the easy-to-use, high performance, Python based free open source NumPy mathematical library using real-world examples. Beginner's Guide.** Birmingham: Packt Publishing Ltd., 2011. 212 p.

-KIUSALAAS, Jaan. **Numerical Methods in Engineering with Python.** 2ed. Nova York: Cambridge University Press, 2010. 422 p.

-LANDAU, Rubin H. **A First Course in Scientific Computing: Symbolic, Graphic, and Numeric Modeling Using Maple, Java, Mathematica, and Fortran90**. Princeton: Princeton University Press, 2005. 481p.

-LANDAU, Rubin H., PÁEZ, Manuel José, BORDEIANU, Cristian C. **A Survey of Computational Physics. Introductory Computational Physics.** Princeton: Princeton University Press, 2008. 658 p.

-LUTZ, Mark. **Programming Python.** 4ed. Sebastopol: O'Reilly Media, Inc., 2010. 1584 p.

-MODEL, Mitchell L. **Bioinformatics Programming Using Python**. Sebastopol: O'Reilly Media, Inc., 2011. 1584 p.

-TOSI, Sandro. **Matplotlib for Python Developers.** Birmingham: Packt Publishing Ltd., 2009. 293 p.

Última atualização: 02 de abril de 2019.