



無料電子ブック

学習

matplotlib

Free unaffiliated eBook created from
Stack Overflow contributors.

#matplotlib

.....	1
1: matplotlib	2
.....	2
.....	2
.....	2
Examples.....	2
.....	2
Windows	2
OS X	2
Linux	3
Debian / Ubuntu.....	3
Fedora / Red Hat.....	3
.....	3
matplotlib.....	3
.....	5
2D.....	6
2: 3	8
.....	8
Examples.....	11
3.....	11
3: Figure	13
.....	13
Examples.....	13
pyplot.....	13
plt.closeFigure.....	13
4: LogLog Graphing	14
.....	14
Examples.....	14
LogLog.....	14
5: TeX / LaTeX	17
.....	

Examples.....	17
TeX.....	17
TeX.....	19
6:	21
.....	21
Examples.....	21
FuncAnimation.....	21
gif.....	22
matplotlib.widgets.....	23
matplotlib.....	24
7:	27
Examples.....	27
.....	27
.....	29
.....	31
.....	33
8:	35
Examples.....	35
.....	35
.....	35
.....	36
9:	38
Examples.....	38
.....	38
10:	39
Examples.....	39
.....	39
11:	41
Examples.....	41
Boxplot.....	41

12:	48
Examples	48
.....	48
.....	50
.....	52
.....	54
13:	57
Examples	57
.....	57
.....	57
14:	59
Examples	59
.....	59
.....	59
.....	60
.....	61
.....	61
2	62
.....	63
.....	63
.....	65
.....	66
.....	67
15:	71
.....	71
Examples	72
.....	72
16:	74
Examples	74
.....	74
17:	76

Examples.....	76
.....	76
.....	76
18:	78
.....	78
Examples.....	78
.....	78
/.....	79
gridspec.....	81
x2.....	82
.....	83
.....	91

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [matplotlib](#)

It is an unofficial and free matplotlib ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official matplotlib.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

1: matplotlibをいめる

matplotlibはPythonのプロットライブラリです。アプリケーションにプロットをめむためのオブジェクトのAPIをします。これはMATLABのとにしています。

もともと[JDHunter](#)によってかれたもので、にされています。BSDスタイルのライセンスでされています。

バージョン

バージョン	サポートされているPythonのバージョン		
1.3.1	2.6,2.7,3.x	い	20131010
1.4.3	2.6,2.7,3.x	の	2015-07-14
1.5.3	2.7,3.x	の	2016-01-11
2.x	2.7,3.x	のバージョン	2016-07-25

Examples

インストールとセットアップ

matplotlibのインストールにはいくつかのがありますが、そのうちのいくつかはしているシステムによってなります。かければ、パッケージマネージャをとってmatplotlibモジュールとそのをにインストールすることができます。

Windows

Windowsマシンでは、pipパッケージマネージャをしてmatplotlibをインストールできます。Windowsでのpipについては、[ここ](#)をしてください。

OS X

matplotlibをインストールするには、[pip](#)パッケージマネージャをすることをおめします。あなたのシステムにPythonライブラリのいくつかえは[libfreetype](#)をインストールするがあるなら、[homebrew](#)のをしてください。

らかのでpipをできないは、 [ソース](#)からインストールをみてください。

Linux

には、 `python-matplotlib`パッケージを `pip install matplotlib`か、 `pip install matplotlib`して、`matplotlib`のインストールにシステムパッケージマネージャまたはpipをするのがです。

これがなえば、しているマシンで `sudo`をつていないは、 `--user` オプション `python setup.py install --user`をして [ソース](#)からインストールできます。、これは `~/.local matplotlib`をインストールします。

Debian / Ubuntu

```
sudo apt-get install python-matplotlib
```

Fedora / Red Hat

```
sudo yum install python-matplotlib
```

トラブルシューティング

れたmatplotlibをするについては、 [matplotlibのWebサイト](#)をしてください。

matplotlib プロットのカスタマイズ

```
import pylab as plt
import numpy as np

plt.style.use('ggplot')

fig = plt.figure(1)
ax = plt.gca()

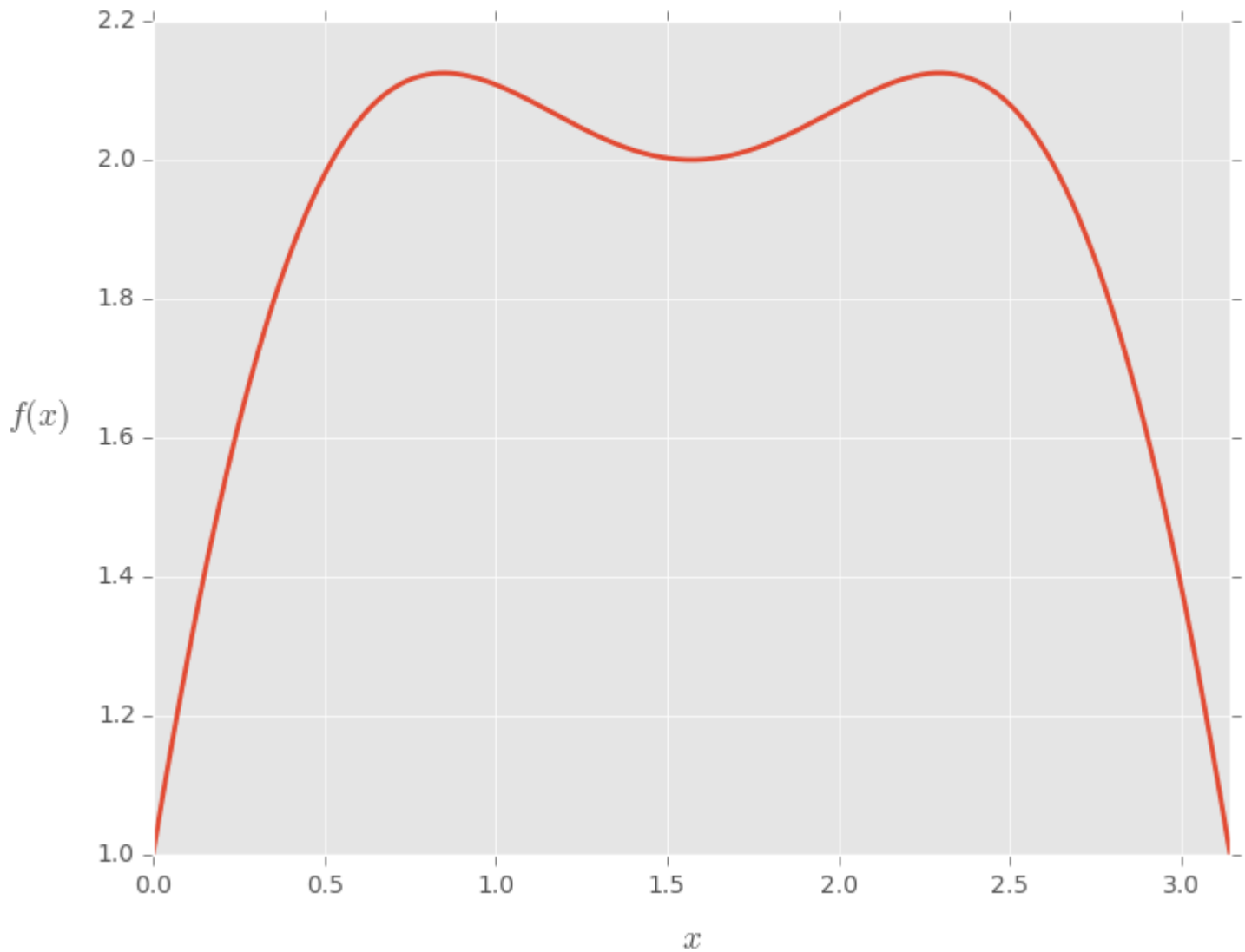
# make some testing data
x = np.linspace( 0, np.pi, 1000 )
test_f = lambda x: np.sin(x)*3 + np.cos(2*x)

# plot the test data
ax.plot( x, test_f(x) , lw = 2)

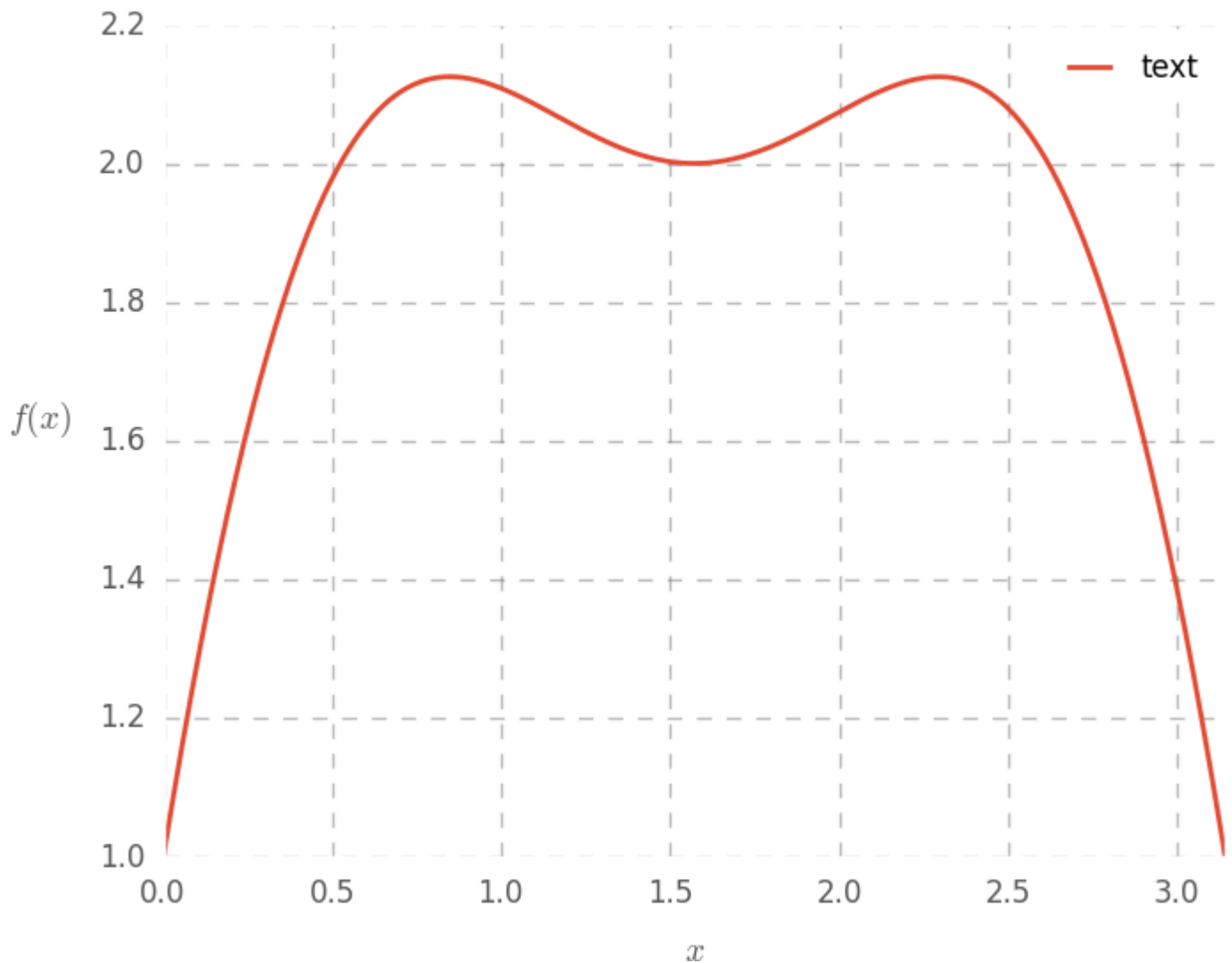
# set the axis labels
ax.set_xlabel(r'$x$', fontsize=14, labelpad=10)
ax.set_ylabel(r'$f(x)$', fontsize=14, labelpad=25, rotation=0)

# set axis limits
ax.set_xlim(0,np.pi)

plt.draw()
```

```
# Customize the plot
ax.grid(1, ls='--', color='#777777', alpha=0.5, lw=1)
ax.tick_params(labelsize=12, length=0)
ax.set_axis_bgcolor('w')
# add a legend
leg = plt.legend( ['text'], loc=1 )
fr = leg.get_frame()
fr.set_facecolor('w')
fr.set_alpha(.7)
plt.draw()
```



オブジェクト

Matplotlibはプロットのためのオブジェクトとのをサポートしています。はにMatlabのににようにされています。

「ステートマシン」ともばれるは、すべてのまたはMatlabなどでするのコマンドをします。、オブジェクトのは、のオブジェクト、などにしてにします。Pythonのでは、ながよりもれているため、オブジェクトのがよりpythonであるとべています。しかし、のは、Matlabのしいツールやさな"いて"プロットスクリプトのにです。は、2つのなるスタイルのです。

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

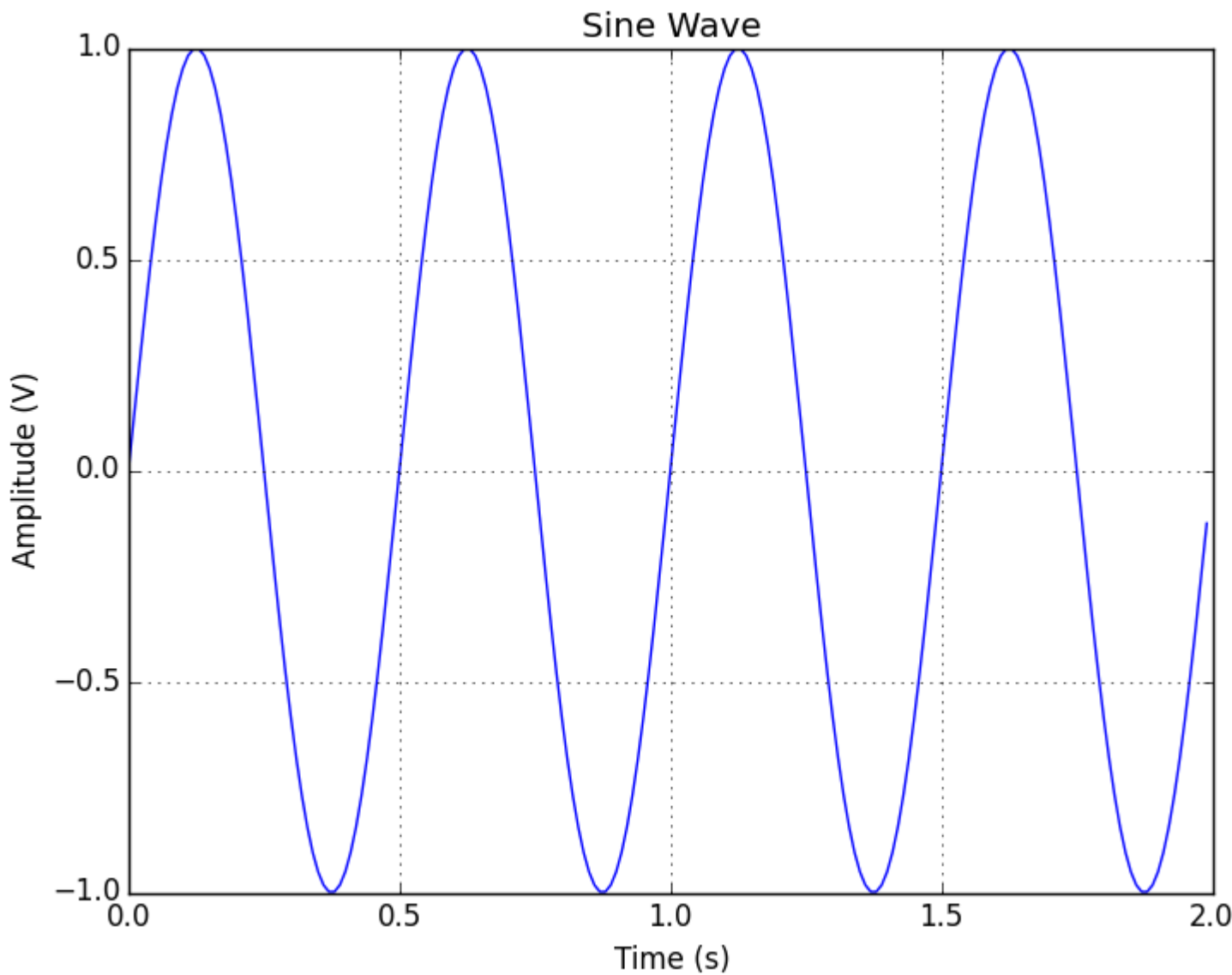
t = np.arange(0, 2, 0.01)
y = np.sin(4 * np.pi * t)

# Imperative syntax
plt.figure(1)
plt.clf()
plt.plot(t, y)
```

```
plt.xlabel('Time (s)')
plt.ylabel('Amplitude (V)')
plt.title('Sine Wave')
plt.grid(True)

# Object oriented syntax
fig = plt.figure(2)
fig.clf()
ax = fig.add_subplot(1,1,1)
ax.plot(t, y)
ax.set_xlabel('Time (s)')
ax.set_ylabel('Amplitude (V)')
ax.set_title('Sine Wave')
ax.grid(True)
```

のとも、にすじプロットをします。

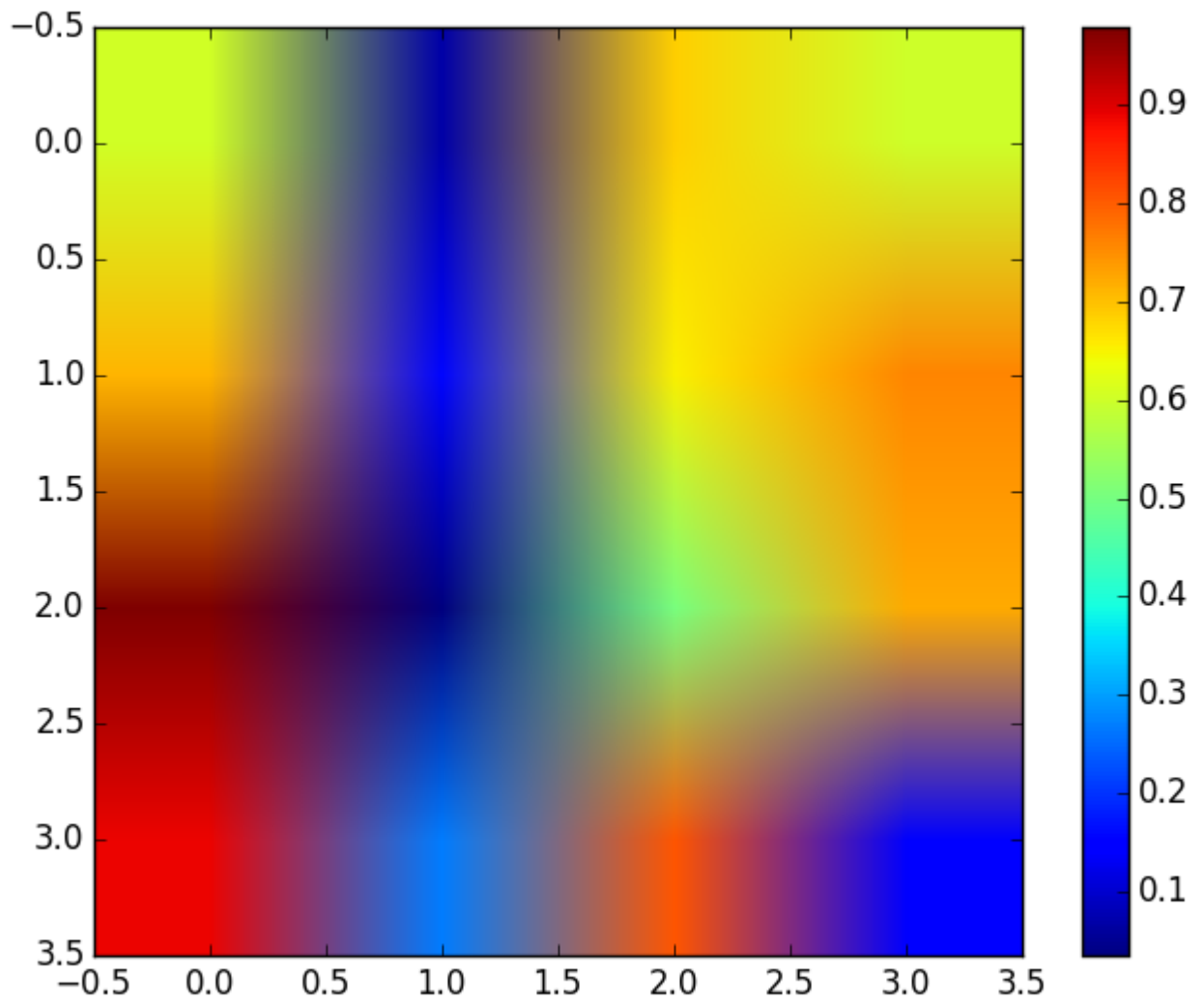


2Dアレイ

に2Dをします。

```
import numpy as np
from matplotlib.pyplot import imshow, show, colorbar

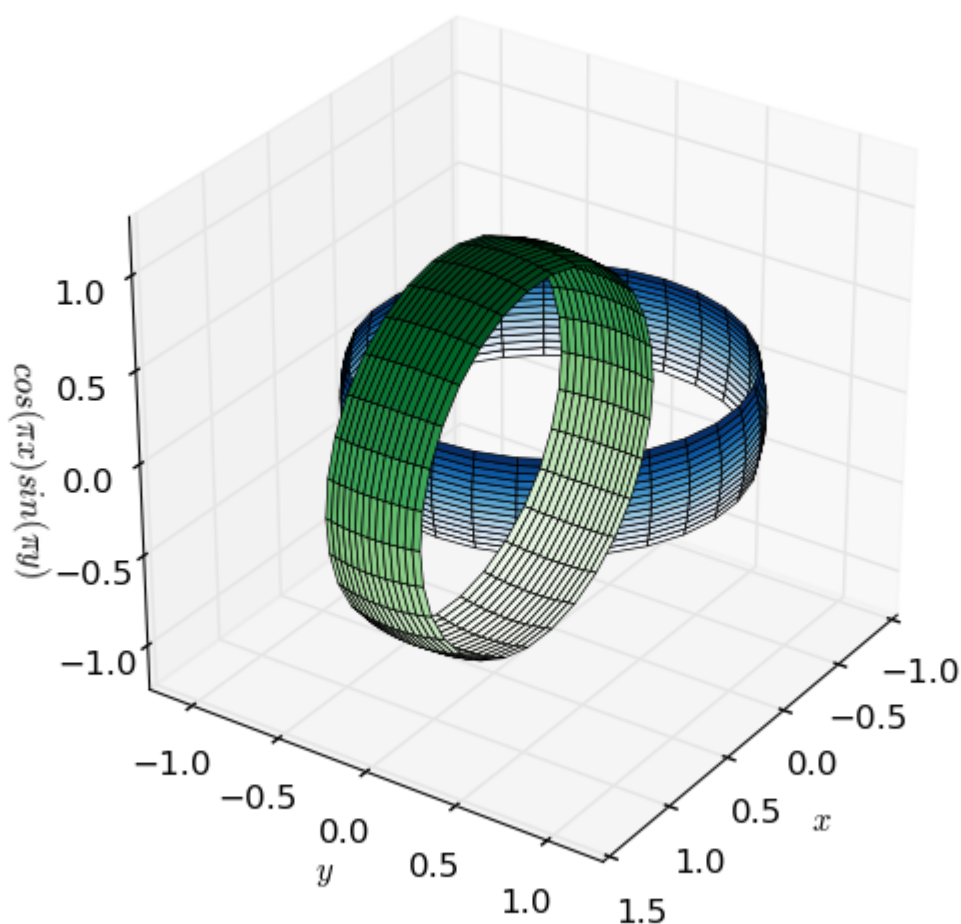
image = np.random.rand(4,4)
imshow(image)
colorbar()
show()
```



オンラインでmatplotlibをいめるをむ <https://riptutorial.com/ja/matplotlib/topic/881/matplotlibをいめる>

2: 3プロット

レンダリングエンジンは2Dなので、matplotlibの3Dプロットにはちょっとしたものでした。1つの2Dチャックをのチャックにプロットすることによって3Dセットアップがレンダリングされるといいます。オブジェクトのたのさにするレンダリングのがしばしばあることをする。こののは、2つのされていないオブジェクトがにろにあるか、またはににあることで、ののように2つのしたリングアニメーションGIFのはクリックのようなアーティファクトにつながります。



しかしこれはすることができます。このアーチファクトは、じプロットにのサーフェスをプロットするにのみします。サーフェスはの2Dシェイプとしてレンダリングされ、ビューのをするパラメータが1つあります。のなサーフェスにもじがないことにづくでしょう。

これをやるは、なブリッジをしてプロットオブジェクトをにすることです。

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
```

```

import numpy as np
from scipy.special import erf

fig = plt.figure()
ax = fig.gca(projection='3d')

X = np.arange(0, 6, 0.25)
Y = np.arange(0, 6, 0.25)
X, Y = np.meshgrid(X, Y)

Z1 = np.empty_like(X)
Z2 = np.empty_like(X)
C1 = np.empty_like(X, dtype=object)
C2 = np.empty_like(X, dtype=object)

for i in range(len(X)):
    for j in range(len(X[0])):
        z1 = 0.5*(erf((X[i,j]+Y[i,j]-4.5)*0.5)+1)
        z2 = 0.5*(erf((-X[i,j]-Y[i,j]+4.5)*0.5)+1)
        Z1[i,j] = z1
        Z2[i,j] = z2

        # If you want to grab a colour from a matplotlib cmap function,
        # you need to give it a number between 0 and 1. z1 and z2 are
        # already in this range, so it just works as is.
        C1[i,j] = plt.get_cmap("Oranges")(z1)
        C2[i,j] = plt.get_cmap("Blues")(z2)

# Create a transparent bridge region
X_bridge = np.vstack([X[-1,:],X[-1,:]])
Y_bridge = np.vstack([Y[-1,:],Y[-1,:]])
Z_bridge = np.vstack([Z1[-1,:],Z2[-1,:]])
color_bridge = np.empty_like(Z_bridge, dtype=object)

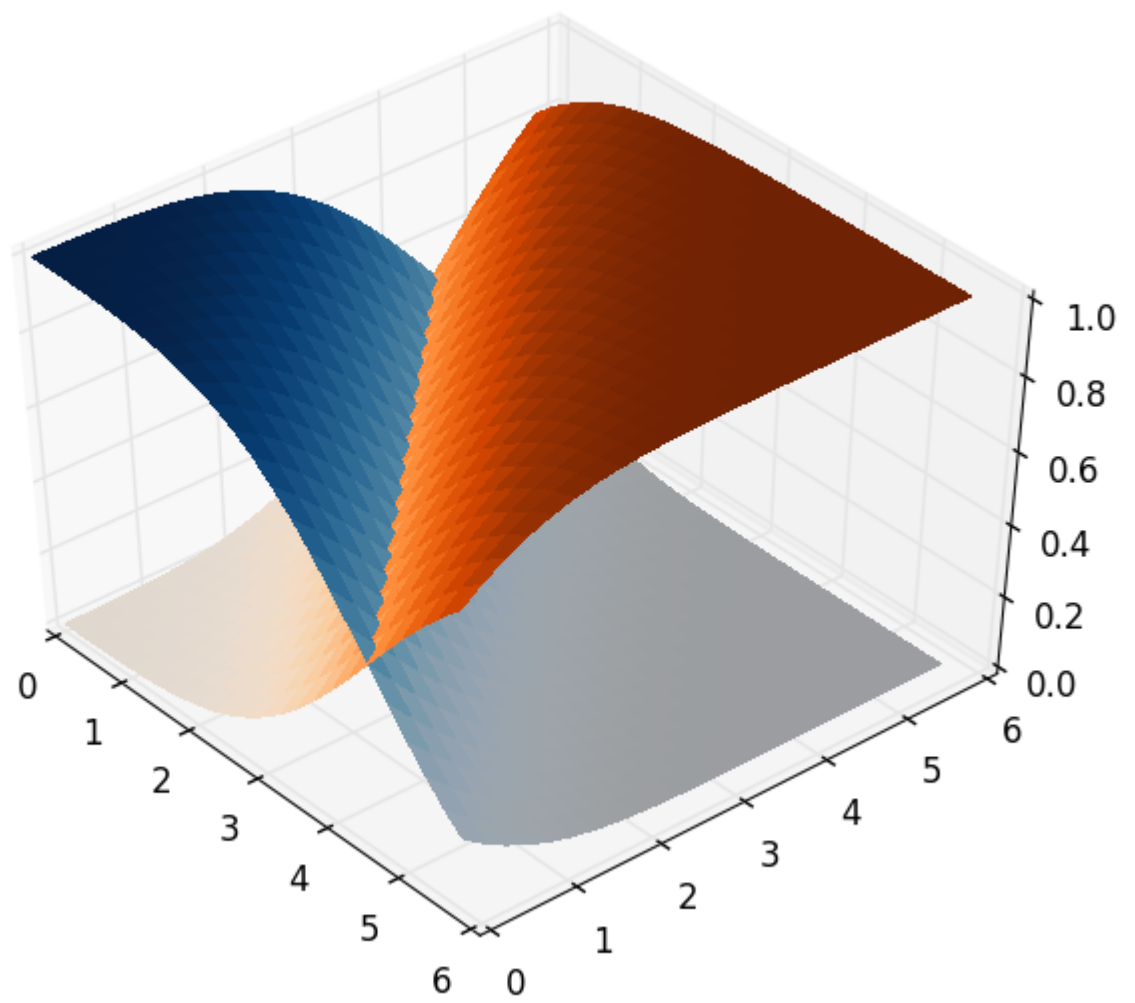
color_bridge.fill((1,1,1,0)) # RGBA colour, onlt the last component matters - it represents
the alpha / opacity.

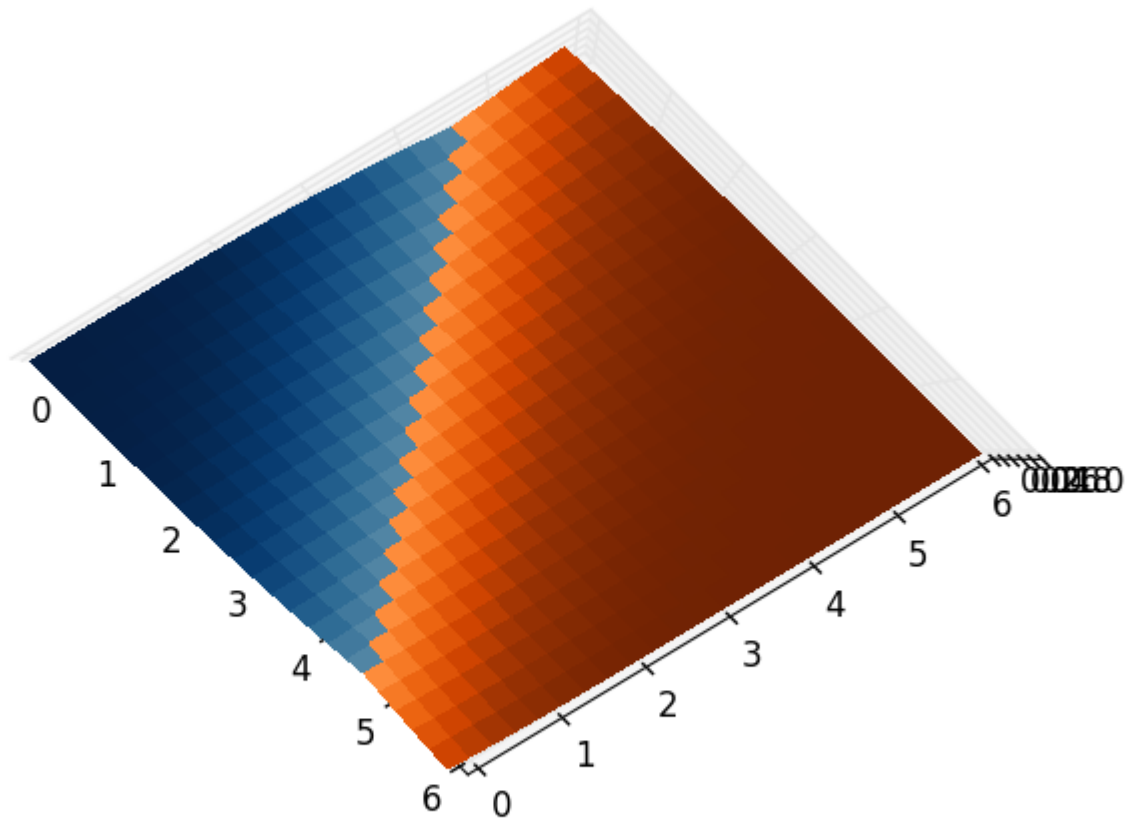
# Join the two surfaces flipping one of them (using also the bridge)
X_full = np.vstack([X, X_bridge, np.flipud(X)])
Y_full = np.vstack([Y, Y_bridge, np.flipud(Y)])
Z_full = np.vstack([Z1, Z_bridge, np.flipud(Z2)])
color_full = np.vstack([C1, color_bridge, np.flipud(C2)])

surf_full = ax.plot_surface(X_full, Y_full, Z_full, rstride=1, cstride=1,
                             facecolors=color_full, linewidth=0,
                             antialiased=False)

plt.show()

```





Examples

3D

Matplotlibはデフォルトで2Dです。3Dのplotをするには、`Axes3D`クラスを`mplot3d`ツールキットからインポートする必要があります。これにより、つまり、`3d`のようになります。

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
```

`lineplot`、`barplot`、`plot`などの2D plotの他に、`ax.plot_surface`などのいくつかの`surface plot`があります。

```
# generate example data
```

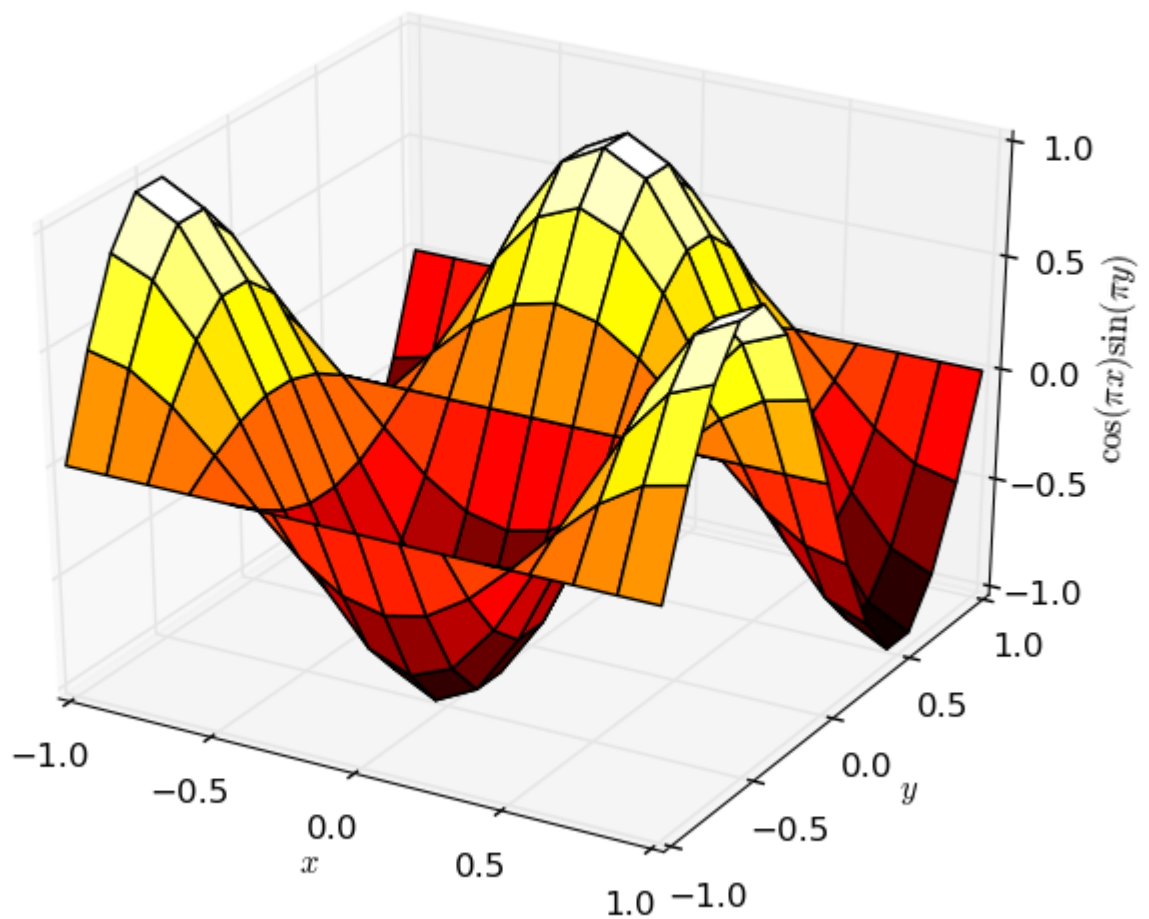


```

import numpy as np
x,y = np.meshgrid(np.linspace(-1,1,15),np.linspace(-1,1,15))
z = np.cos(x*np.pi)*np.sin(y*np.pi)

# actual plotting example
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
# rstride and cstride are row and column stride (step size)
ax.plot_surface(x,y,z,rstride=1,cstride=1,cmap='hot')
ax.set_xlabel(r'$x$')
ax.set_ylabel(r'$y$')
ax.set_zlabel(r'$\cos(\pi x) \sin(\pi y)$')
plt.show()

```



オンラインで3プロットをむ <https://riptutorial.com/ja/matplotlib/topic/1880/3プロット>

3: Figure ウィンドウをじる

- `plt.close` アクティブなFigureをじます。
- `plt.closefig` ハンドル 'fig' でをじます。
- `plt.closenum` をじます。 'num'
- `plt.closetname` 'name' というラベルでをじます。
- `plt.close 'all'` すべてのをじます。

Examples

pyplot をってアクティブなをじる

matplotlib への pyplot インターフェースは、Figure をじるもなかかもしれません。

```
import matplotlib.pyplot as plt
plt.plot([0, 1], [0, 1])
plt.close()
```

plt.close をしてのFigure をじる

のは、そのハンドルをすることによってじることができます

```
import matplotlib.pyplot as plt

fig1 = plt.figure() # create first figure
plt.plot([0, 1], [0, 1])

fig2 = plt.figure() # create second figure
plt.plot([0, 1], [0, 1])

plt.close(fig1) # close first figure although second one is active
```

オンラインでFigureウィンドウをじるをむ <https://riptutorial.com/ja/matplotlib/topic/6628/figureウィンドウをじる>

4: LogLog Graphing

き

LogLogグラフは、をにするがあります。

Examples

LogLog グラフ

えば、 $A = 30$ 、 $a = 3.5$ など、 $y = A \cdot x^a$ とする。 $\ln y = \ln A \cdot x^a = \ln A + \ln x^a = \ln A + a \cdot \ln x$ このようにして、 $+ a \cdot \ln x$ である。したがって、 x と y のについてをつプロットはになります。こののきは $y = A \cdot x^a$ の a であり、 y_0 は A の、 $\ln A = \ln 30 = 3.401$ である。

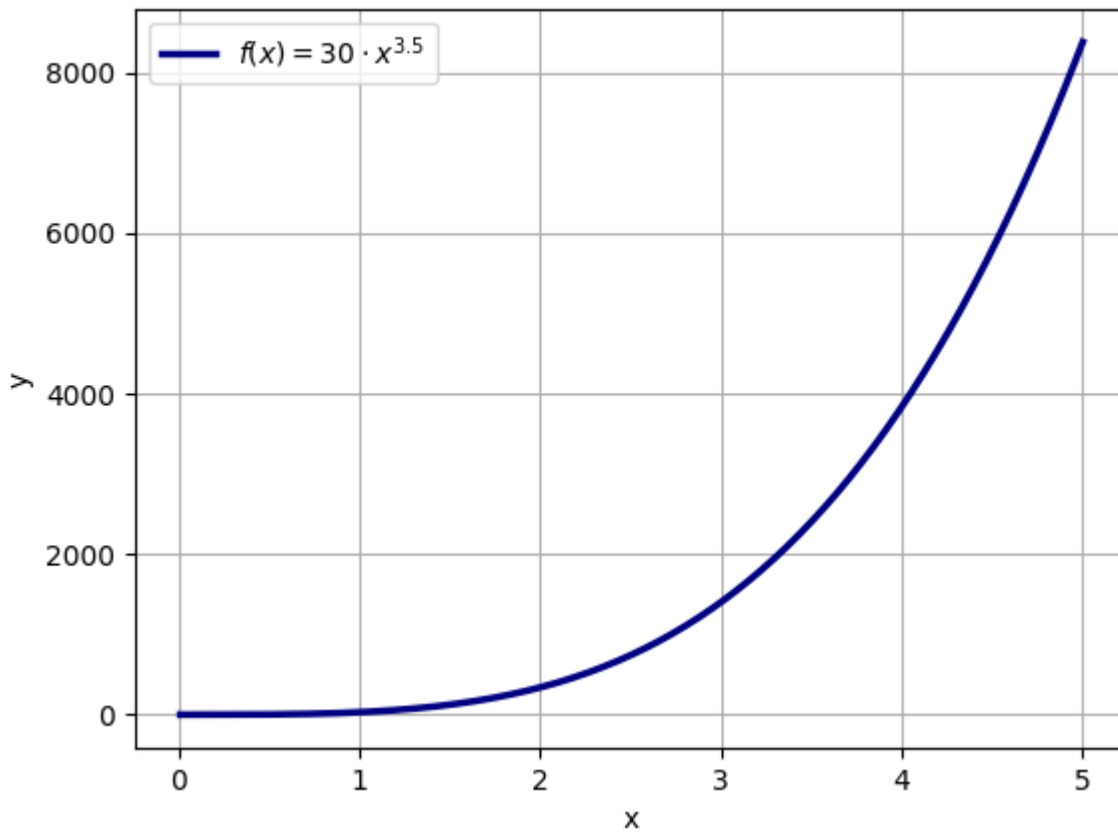
のは、とloglogプロットのをしていますは $y = A \cdot x^a$ 、 $A = 30$ 、 $a = 3.5$ 。

```
import numpy as np
import matplotlib.pyplot as plt
A = 30
a = 3.5
x = np.linspace(0.01, 5, 10000)
y = A * x**a

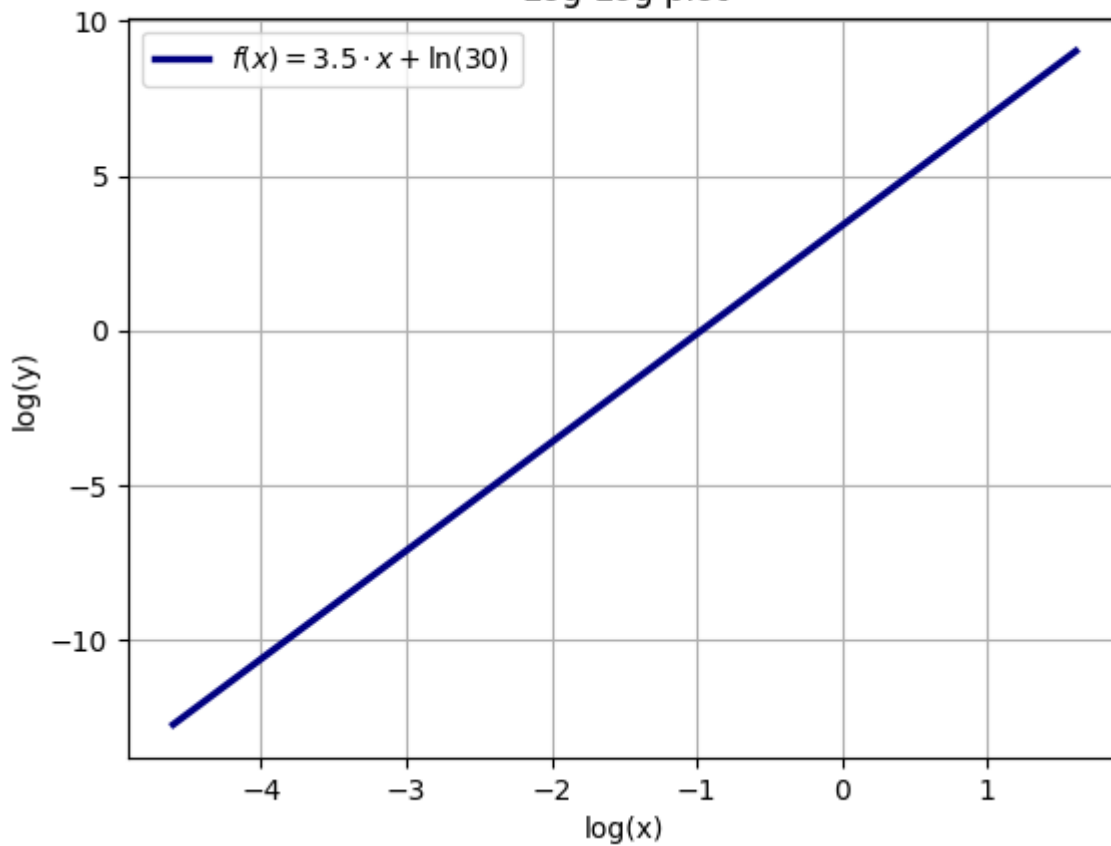
ax = plt.gca()
plt.plot(x, y, linewidth=2.5, color='navy', label=r'$f(x) = 30 \cdot x^{3.5}$')
plt.legend(loc='upper left')
plt.xlabel(r'x')
plt.ylabel(r'y')
ax.grid(True)
plt.title(r'Normal plot')
plt.show()
plt.clf()

xlog = np.log(x)
ylog = np.log(y)
ax = plt.gca()
plt.plot(xlog, ylog, linewidth=2.5, color='navy', label=r'$f(x) = 3.5 \cdot x + \ln(30)$')
plt.legend(loc='best')
plt.xlabel(r'log(x)')
plt.ylabel(r'log(y)')
ax.grid(True)
plt.title(r'Log-Log plot')
plt.show()
plt.clf()
```

Normal plot



Log-Log plot



オンラインでLogLog Graphingをむ <https://riptutorial.com/ja/matplotlib/topic/10145/loglog-graphing>

5: TeX / LaTeXとの

- MatplotlibのLaTeXサポートには、するLaTeXのインストール、dvipngLaTeXインストールにまれるかもしれませんが、GhostscriptGPL Ghostscript 8.60がされますがです。
- Matplotlibのpgfサポートには、TikZ / PGFパッケージTeXLiveなどをむのLaTeXインストールがです.XeLaTeXまたはLuaLaTeXがインストールされていることがましいです。

Examples

プロットにTeXをする

TeXは、`rc`をしてプロットにできます

```
import matplotlib.pyplot as plt
plt.rc(usetex = True)
```

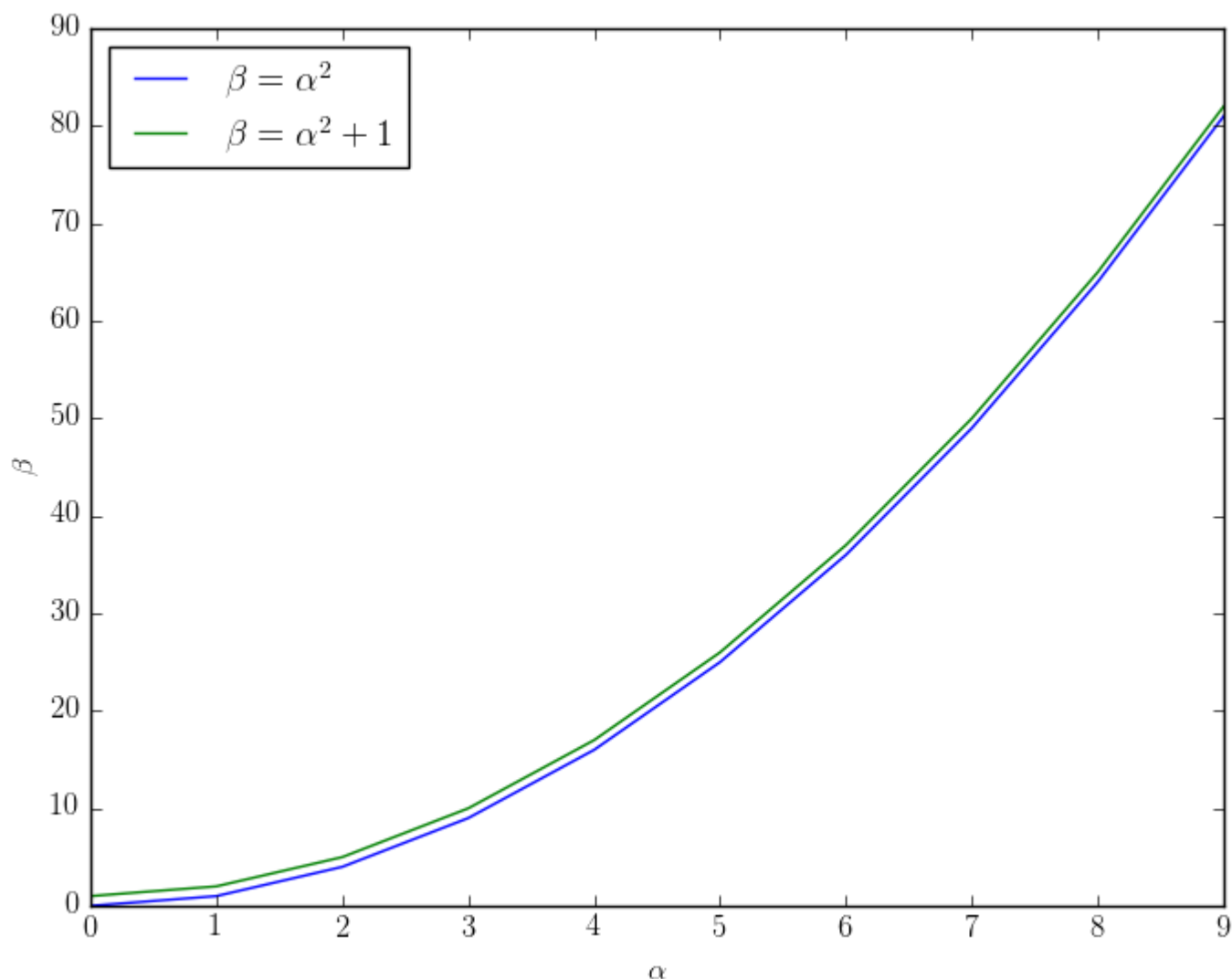
`rcParams`アクセスする

```
import matplotlib.pyplot as plt
params = {'tex.usetex': True}
plt.rcParams.update(params)
```

TeXはコマンドとシンボルにバックスラッシュ\をします。これはPythonのとするがあります。Pythonでリテラルのバックスラッシュをするには、エスケープするか、のにみむがあります。

```
plt.xlabel('\alpha')
plt.xlabel(r'\alpha')
```

のプロット



コードによってすることができます

```
import matplotlib.pyplot as plt
plt.rc(usetex = True)
x = range(0,10)
y = [t**2 for t in x]
z = [t**2+1 for t in x]
plt.plot(x, y, label = r'$\beta=\alpha^2$')
plt.plot(x, z, label = r'$\beta=\alpha^2+1$')
plt.xlabel(r'$\alpha$')
plt.ylabel(r'$\beta$')
plt.legend(loc=0)
plt.show()
```

された `$$...$$` や `\begin{equation}...\end{equation}` はサポートされていません。それにもかかわらず、`\displaystyle` でされるスタイルはです。

ラテックスパッケージをロードするには、`tex.latex.preamble` をします

```
params = {'text.latex.preamble' : [r'\usepackage{siunitx}', r'\usepackage{amsmath}']}
plt.rcParams.update(params)
```

ただし、[matplotlibrcファイル](#)ののののとおりです。

```
#text.latex.preamble : # IMPROPER USE OF THIS FEATURE WILL LEAD TO LATEX FAILURES
                        # AND IS THEREFORE UNSUPPORTED. PLEASE DO NOT ASK FOR HELP
                        # IF THIS FEATURE DOES NOT DO WHAT YOU EXPECT IT TO.
                        # preamble is a comma separated list of LaTeX statements
                        # that are included in the LaTeX document preamble.
                        # An example:
                        # text.latex.preamble : \usepackage{bm},\usepackage{euler}
                        # The following packages are always loaded with usetex, so
                        # beware of package collisions: color, geometry, graphicx,
                        # typelcm, textcomp. Adobe Postscript (PSSNFS) font packages
                        # may also be loaded, depending on your font settings
```

TeXをするプロットのとエクスポート

TeXドキュメントにmatplotlibでしたプロットをめるには、`pdf`または`eps`ファイルとしてするがあります。このようにして、プロットのテキストTeXをむは、ドキュメントのテキストとしてレンダリングされます。

```
import matplotlib.pyplot as plt
plt.rc(usetex=True)
x = range(0, 10)
y = [t**2 for t in x]
z = [t**2+1 for t in x]
plt.plot(x, y, label=r'$\beta=\alpha^2$')
plt.plot(x, z, label=r'$\beta=\alpha^2+1$')
plt.xlabel(r'$\alpha$')
plt.ylabel(r'$\beta$')
plt.legend(loc=0)
plt.savefig('my_pdf_plot.pdf') # Saving plot to pdf file
plt.savefig('my_eps_plot.eps') # Saving plot to eps file
```

matplotlibのプロットは、グラフィックをするために`pgf`マクロパッケージを使ってTeXコードにエクスポートすることができます。

```
import matplotlib.pyplot as plt
plt.rc(usetex=True)
x = range(0, 10)
y = [t**2 for t in x]
z = [t**2+1 for t in x]
plt.plot(x, y, label=r'$\beta=\alpha^2$')
plt.plot(x, z, label=r'$\beta=\alpha^2+1$')
plt.xlabel(r'$\alpha$')
plt.ylabel(r'$\beta$')
plt.legend(loc=0)
plt.savefig('my_pgf_plot.pgf')
```

するTeXエンジンをするには`rc`コマンドをします

```
plt.rc('pgf', texsystem='pdflatex') # or luatex, xelatex...
```

`.pgf`フィギュアをめるには、LaTeXドキュメントにきみます


```
\usepackage{pgf}  
\input{my_pgfg_plot.pgfg}
```

オンラインでTeX / LaTeXとのをむ <https://riptutorial.com/ja/matplotlib/topic/2962/tex---latexとの>

6: アニメーションとインタラクティブなプロット

き

Python matplotlibをすると、アニメーショングラフをにできます。

Examples

FuncAnimationによるアニメーション

[matplotlib.animation](#)パッケージには、アニメーションをするためのクラスがいくつかされています。 [FuncAnimation](#) は、をりしびすことでアニメーションをします。ここでは、サインのグラフの `animate()` をします。

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

TWOPI = 2*np.pi

fig, ax = plt.subplots()

t = np.arange(0.0, TWOPI, 0.001)
s = np.sin(t)
l = plt.plot(t, s)

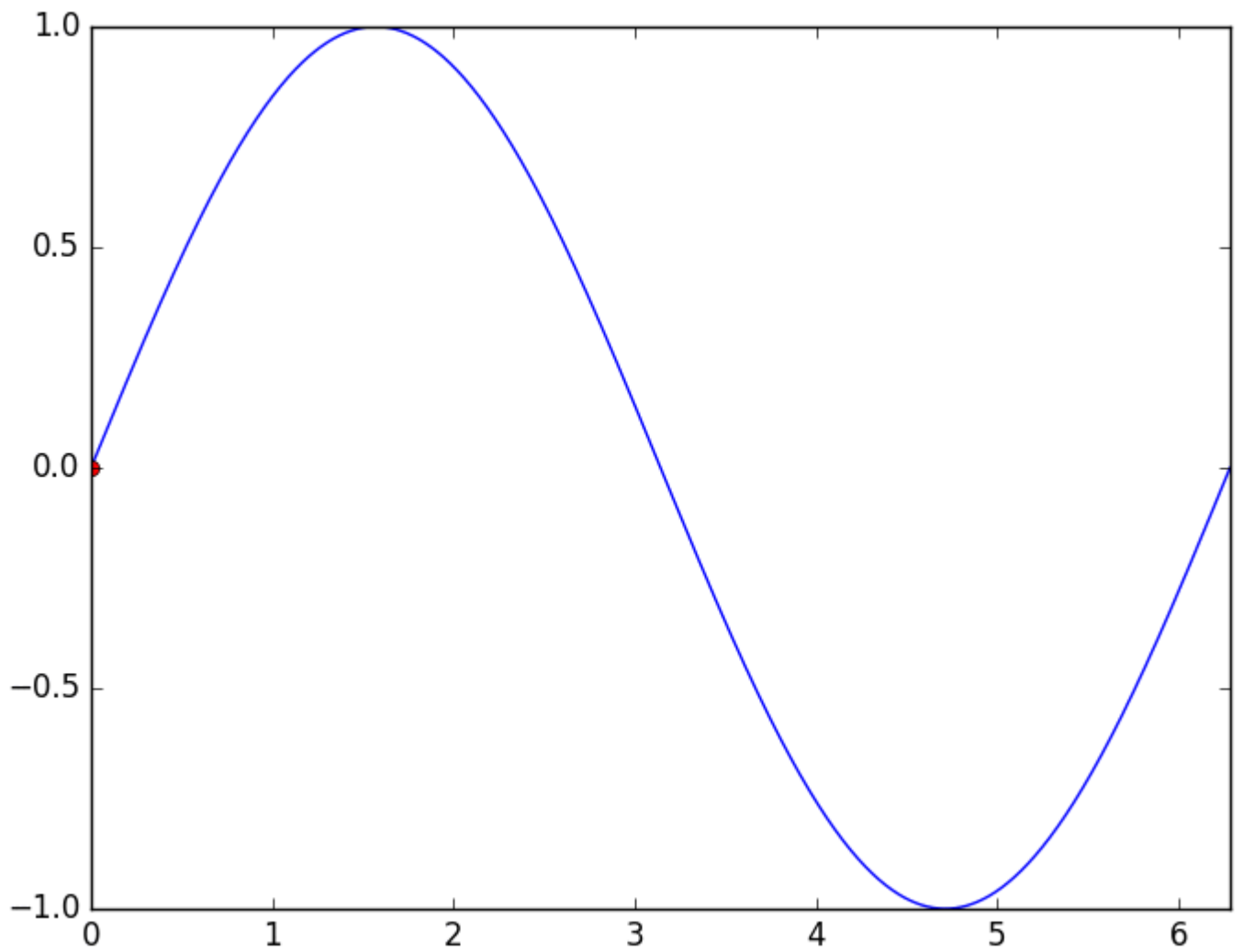
ax = plt.axis([0, TWOPI, -1, 1])

redDot, = plt.plot([0], [np.sin(0)], 'ro')

def animate(i):
    redDot.set_data(i, np.sin(i))
    return redDot,

# create animation using the animate() function
myAnimation = animation.FuncAnimation(fig, animate, frames=np.arange(0.0, TWOPI, 0.1), \
                                     interval=10, blit=True, repeat=True)

plt.show()
```



アニメーションをgifにする

ここでは、`save`メソッドをして、ImageMagickをして`Animation`オブジェクトをします。

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from matplotlib import rcParams

# make sure the full paths for ImageMagick and ffmpeg are configured
rcParams['animation.convert_path'] = r'C:\Program Files\ImageMagick\convert'
rcParams['animation.ffmpeg_path'] = r'C:\Program Files\ffmpeg\bin\ffmpeg.exe'

TWOPI = 2*np.pi

fig, ax = plt.subplots()

t = np.arange(0.0, TWOPI, 0.001)
s = np.sin(t)
l = plt.plot(t, s)
```

```

ax = plt.axis([0,TWOPI,-1,1])

redDot, = plt.plot([0], [np.sin(0)], 'ro')

def animate(i):
    redDot.set_data(i, np.sin(i))
    return redDot,

# create animation using the animate() function with no repeat
myAnimation = animation.FuncAnimation(fig, animate, frames=np.arange(0.0, TWOPI, 0.1), \
                                     interval=10, blit=True, repeat=False)

# save animation at 30 frames per second
myAnimation.save('myAnimation.gif', writer='imagemagick', fps=30)

```

matplotlib.widgetsによるインタラクティブコントロール

プロットとするためにMatplotlibはGUIウィジェットをします。ウィジェットにはmatplotlib.axes.Axesオブジェクトがです。

サインカーブのをするスライダウィジェットのデモがあります。は、スライダのon_changed()イベントによってトリガされます。

```

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from matplotlib.widgets import Slider

TWOPI = 2*np.pi

fig, ax = plt.subplots()

t = np.arange(0.0, TWOPI, 0.001)
initial_amp = .5
s = initial_amp*np.sin(t)
l, = plt.plot(t, s, lw=2)

ax = plt.axis([0,TWOPI,-1,1])

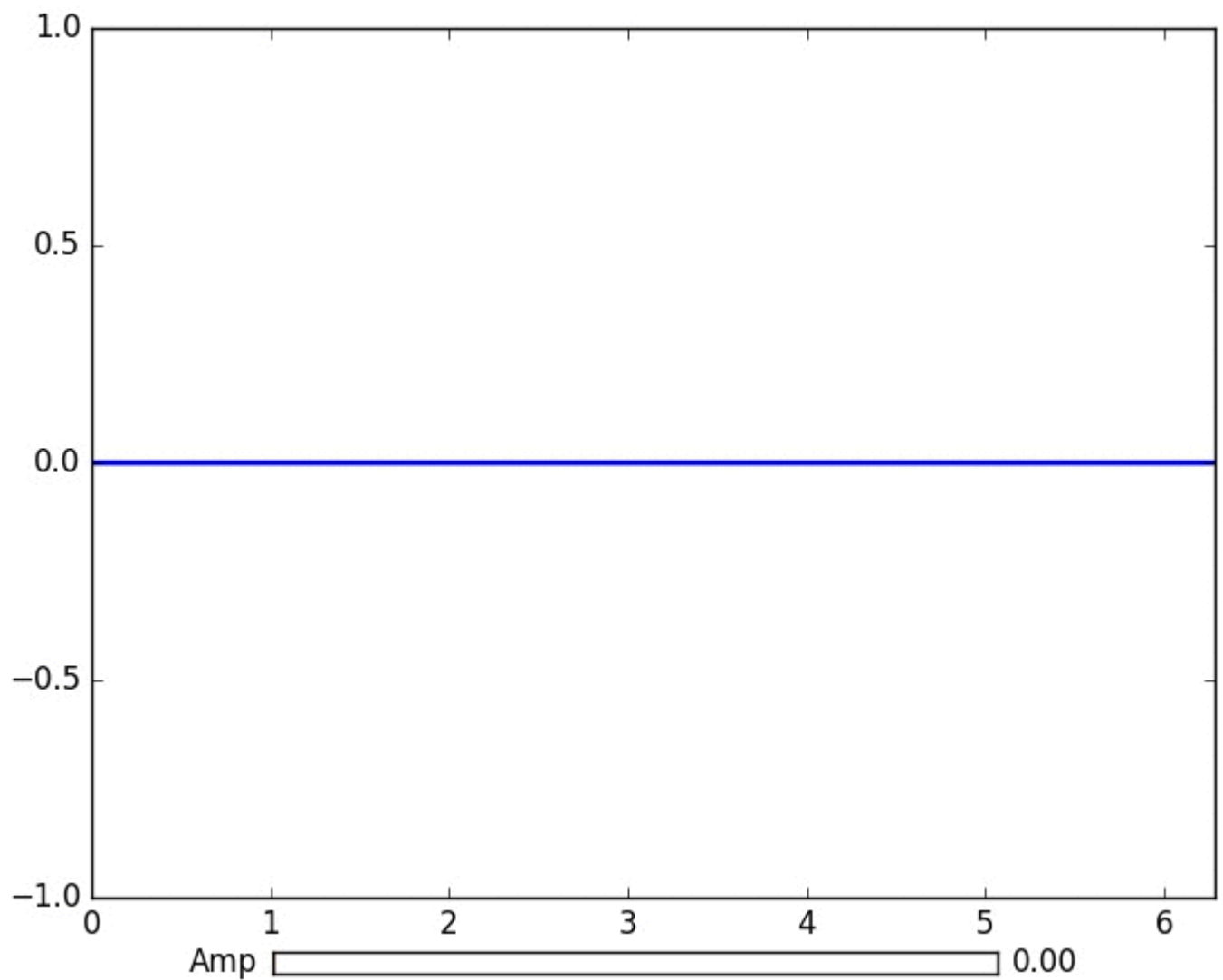
axamp = plt.axes([0.25, .03, 0.50, 0.02])
# Slider
samp = Slider(axamp, 'Amp', 0, 1, valinit=initial_amp)

def update(val):
    # amp is the current value of the slider
    amp = samp.val
    # update curve
    l.set_ydata(amp*np.sin(t))
    # redraw canvas while idle
    fig.canvas.draw_idle()

# call update function on slider value change
samp.on_changed(update)

plt.show()

```



のなウィジェット

- [AxesWidget](#)
- ボタン
- [CheckButtons](#)
- カーソル
- [EllipseSelector](#)
- ラッソ
- [LassoSelector](#)
- [LockDraw](#)
- マルチカーソル
- ラジオボタン
- [RectangleSelector](#)
- [SpanSelector](#)
- [SubplotTool](#)
- [ToolHandles](#)

matplotlibでパイプからライブデータをプロットする

これは、データをリアルタイムでしたいにです。このデータは、例えば、アナログをにサンプリングしているマイクロコントローラからるがあります。

このでは、きパイプFIFOともばれますからデータをします。このでは、パイプのデータはでられたにするがありますが、みにわけてすることができます。

データ

```
100
123.5
1589
```

きパイプの

ライブラリコレクションのデータdequeもします。 dequeオブジェクトはリストのようにかなりします。しかしデキューオブジェクトでは、デキューオブジェクトをにちながらかをするのはです。これにより、にグラフをにさせたりめたりするのではなく、xをのさにつことができます。

dequeオブジェクトの

なバックエンドをすることは、パフォーマンスにとってです。あなたのオペレーティングシステムでバックエンドがするかどうかをし、なものをしてください。にとってはqt4aggとデフォルトのバックエンドだけがしましたが、デフォルトのものはすぎました。 [matplotlibのバックエンドの](#)

このは[ランダムデータをプロットするmatplotlibの](#)についています。

このコードのはされません。

```
import matplotlib
import collections
#selecting the right backend, change qt4agg to your desired backend
matplotlib.use('qt4agg')
import matplotlib.pyplot as plt
import matplotlib.animation as animation

#command to open the pipe
datapipe = open('path to your pipe','r')

#amount of data to be displayed at once, this is the size of the x axis
#increasing this amount also makes plotting slightly slower
data_amount = 1000

#set the size of the deque object
datalist = collections.deque([0]*data_amount,data_amount)

#configure the graph itself
fig, ax = plt.subplots()
line, = ax.plot([0,]*data_amount)

#size of the y axis is set here
ax.set_ylim(0,256)

def update(data):
    line.set_ydata(data)
```

```

        return line,

def data_gen():
    while True:
        """
        We read two data points in at once, to improve speed
        You can read more at once to increase speed
        Or you can read just one at a time for improved animation smoothness
        data from the pipe comes in as a string,
        and is seperated with a newline character,
        which is why we use respectively eval and rstrip.
        """
        datalist.append(eval((datapipe.readline()).rstrip('\n')))
        datalist.append(eval((datapipe.readline()).rstrip('\n')))
        yield datalist

ani = animation.FuncAnimation(fig,update,data_gen,interval=0, blit=True)
plt.show()

```

あなたのプロットがしばらくしてからめめるは、`datalist.append`データをさらにして、フレームをよりくのにみませてください。できるだけバックエンドをしてください。

これはの1.7ghz i3 4005uのパイプからの150hzのデータでした。

オンラインでアニメーションとインタラクティブなプロットをむ

<https://riptutorial.com/ja/matplotlib/topic/6983/>アニメーションとインタラクティブなプロット

7: カラーマップ

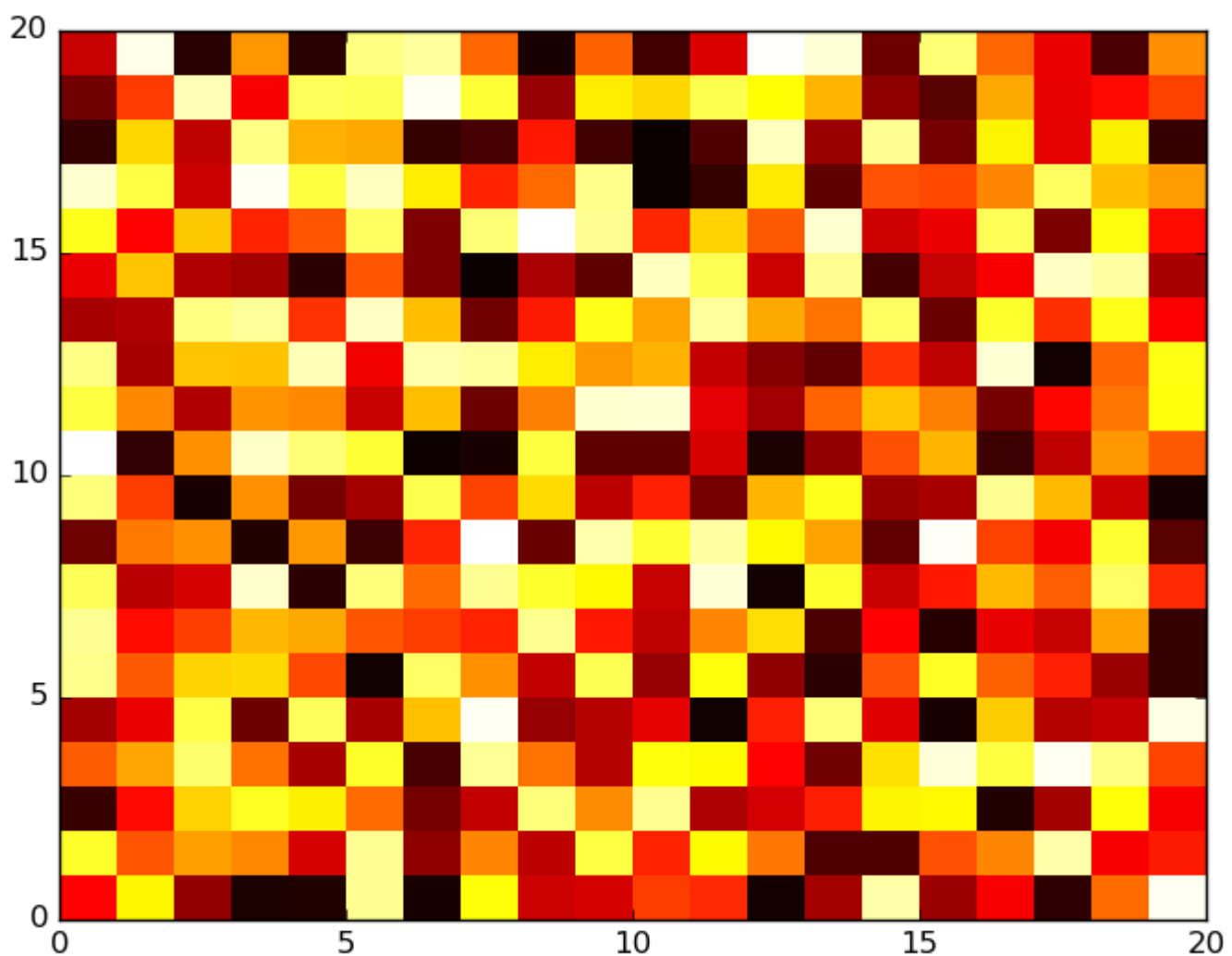
Examples

な

みみのカラーマップをするのは、なカラーマップの [カラーマップにあるように](#) を、は `cmap` キーワードので、プロット `pcolormesh` または `contourf` などにすのとじくらいです。

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

plt.figure()
plt.pcolormesh(np.random.rand(20,20), cmap='hot')
plt.show()
```



カラーマップは、3データを2プロットするににですが、なカラーマップはな3プロットをよりにすることができます。

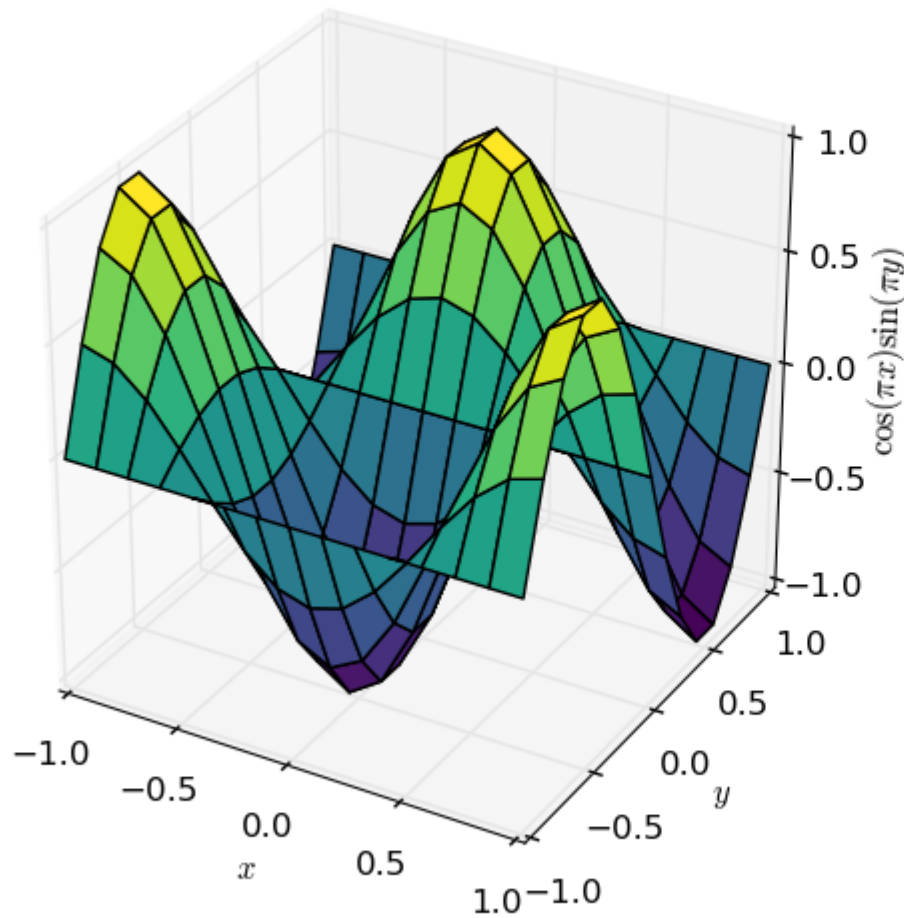
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.ticker import LinearLocator

# generate example data
import numpy as np
x,y = np.meshgrid(np.linspace(-1,1,15),np.linspace(-1,1,15))
z = np.cos(x*np.pi)*np.sin(y*np.pi)

# actual plotting example
fig = plt.figure()
ax1 = fig.add_subplot(121, projection='3d')
ax1.plot_surface(x,y,z,rstride=1,cstride=1,cmap='viridis')
ax2 = fig.add_subplot(122)
cf = ax2.contourf(x,y,z,51,vmin=-1,vmax=1,cmap='viridis')
cbar = fig.colorbar(cf)
cbar.locator = LinearLocator(numticks=11)
cbar.update_ticks()
for ax in {ax1, ax2}:
    ax.set_xlabel(r'$x$')
    ax.set_ylabel(r'$y$')
    ax.set_xlim([-1,1])
    ax.set_ylim([-1,1])
    ax.set_aspect('equal')

ax1.set_zlim([-1,1])
ax1.set_zlabel(r'$\cos(\pi x) \sin(\pi y)$')

plt.show()
```



カスタムカラーマップの

カラーマップでされたみみのカラーマップ およびそれらのマップ、に、`cm` されているとはに、カスタムカラーマップをすることもできます。キーは `matplotlib.cm` モジュールです。

のでは、のをむ `cm.register_cmap` をするになカラーマップをしています。データでにでになのアルファがされています。カラーマップのからたなは、`cm` のインポート、`register_cmap` へのびし、およびカラーマップを `plot_surface` です。

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.cm as cm

# generate data for sphere
from numpy import pi, meshgrid, linspace, sin, cos
th, ph = meshgrid(linspace(0, pi, 25), linspace(0, 2*pi, 51))
x, y, z = sin(th)*cos(ph), sin(th)*sin(ph), cos(th)

# define custom colormap with fixed colour and alpha gradient
# use simple linear interpolation in the entire scale
```

```

cm.register_cmap(name='alpha_gradient',
                 data={'red': [(0.,0,0),
                              (1.,0,0)],

                      'green': [(0.,0.6,0.6),
                                (1.,0.6,0.6)],

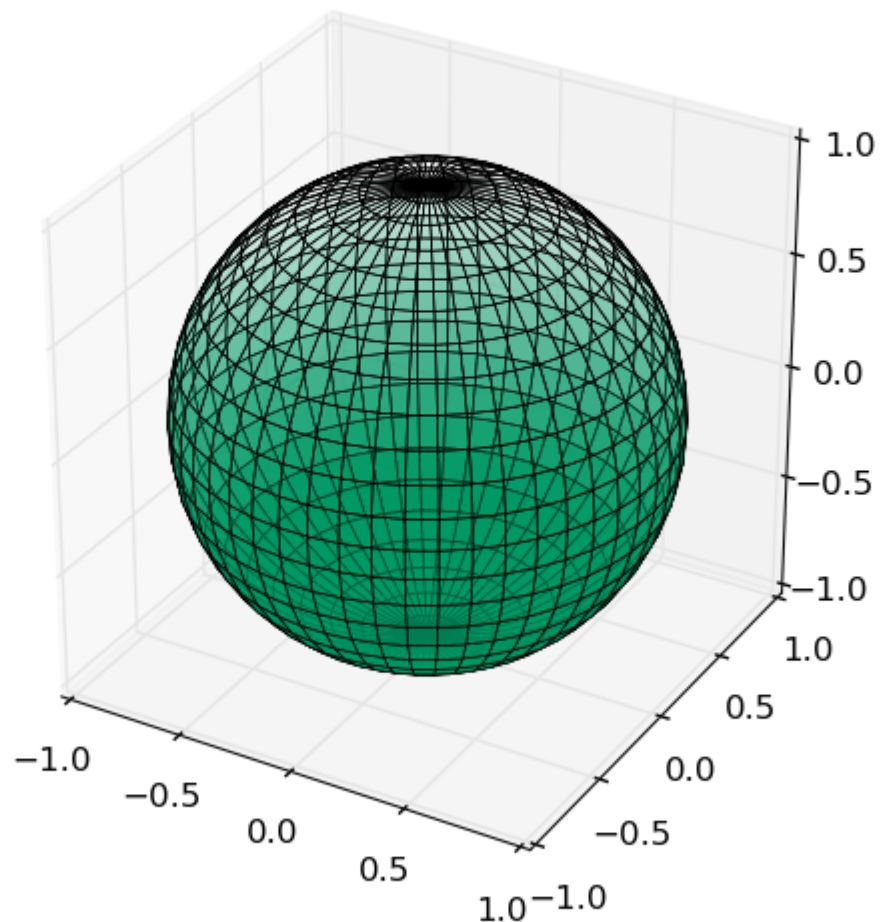
                      'blue': [(0.,0.4,0.4),
                               (1.,0.4,0.4)],

                      'alpha': [(0.,1,1),
                                (1.,0,0)]})

# plot sphere with custom colormap; constrain mapping to between |z|=0.7 for enhanced effect
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x,y,z,cmap='alpha_gradient',vmin=-
0.7,vmax=0.7,rstride=1,cstride=1,linewidth=0.5,edgecolor='b')
ax.set_xlim([-1,1])
ax.set_ylim([-1,1])
ax.set_zlim([-1,1])
ax.set_aspect('equal')

plt.show()

```



よりなシナリオでは、するプロットでされるをするためにmatplotlibがにするR / G / B / Aのリストをすることができます。

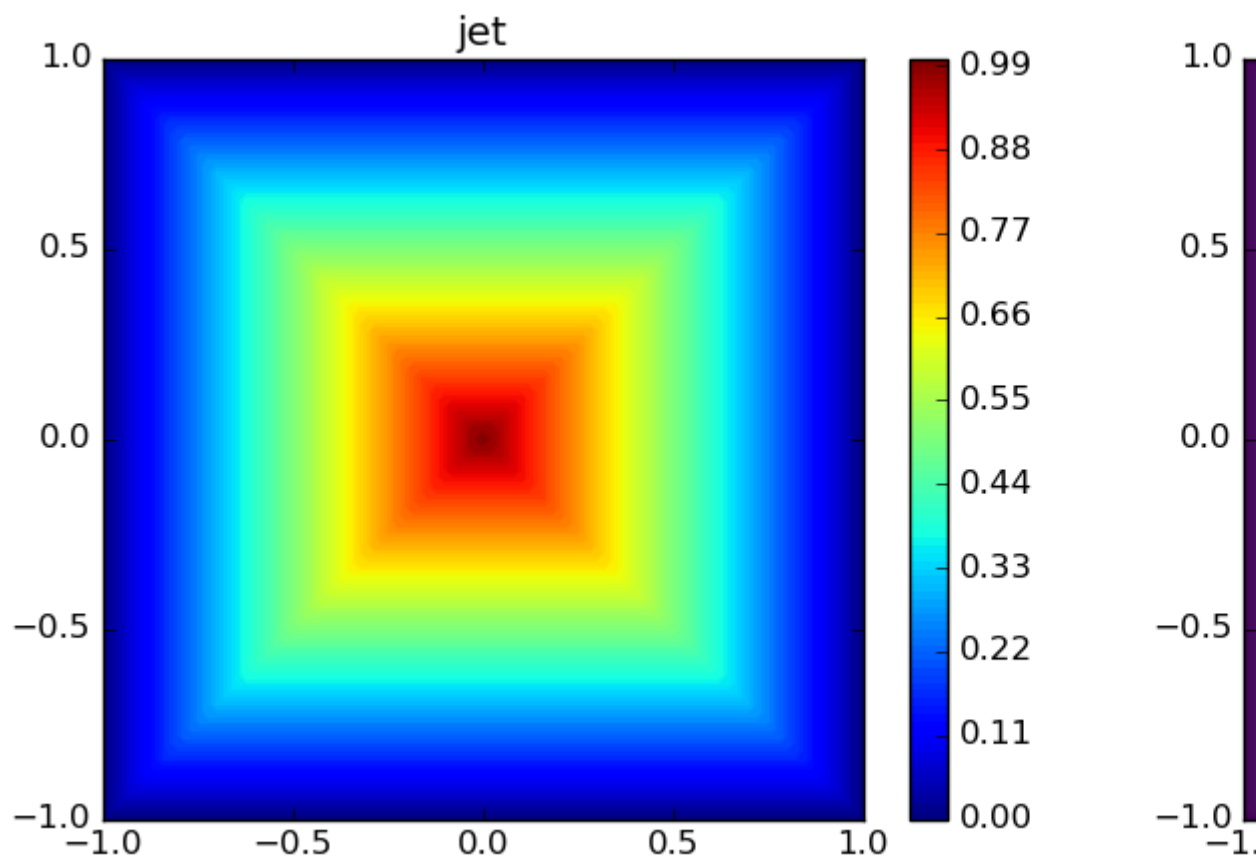
になカラーマップ

jet とばれるMATLABののデフォルトカラーマップR2014bできえられましたは、いコントラストとしみやすさのためにしていますのからmatplotlibのデフォルトでした。そのにもかかわらず、[のカラーマップ](#)には、データをにすることにあることがよくあります。これらのカラーマップのは、データのにしていません。カラーマップをグレースケールにするとえば、プリンタをしてをするなど、がわれるがあります。

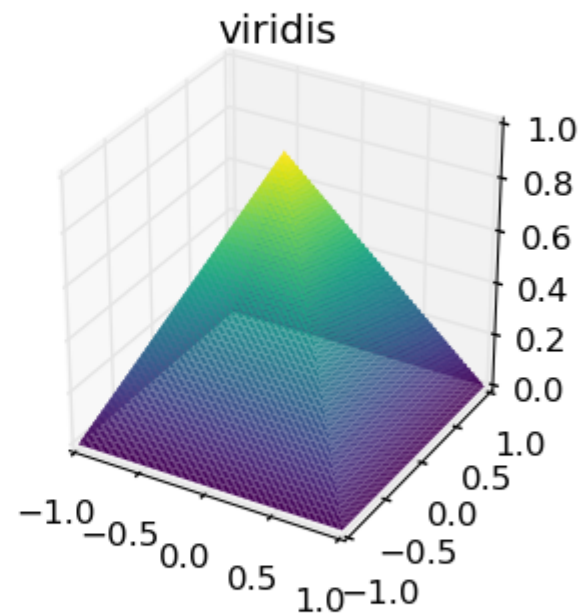
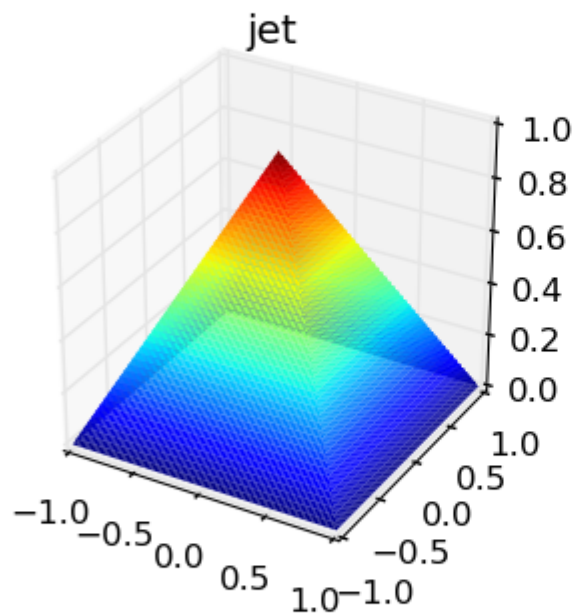
になカラーマップは、データをなりかつアクセスにするためにされました。 Matplotlibはバージョン1.5に[4つのしいにユニークなカラーマップ](#)をし、そのうちの1つ viridis というをバージョン2.0からデフォルトにしました。これらの4つのカラーマップ viridis、inferno、plasma、magma は、のからはすべてです。そうしないとにいがないう、デフォルトでデータにしてください。これらのカラーマップは、できるだけりのないようにしていますするがないはフィーチャをしない

ため、のがしたオーディエンスにしています。

データをにめるとして、ピラミッドオブジェクトのの2つのトップビュープロットをえてみましょう。



どちらがしいピラミッドですかえはもちろんともそうですが、これは`jet` カラーマップをったプロットからははっきりしていません。



これは、のにある。

カスタムのカラーマップ

みのがあり、そののにをするは、カスタムカラーマップをできます。えは

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

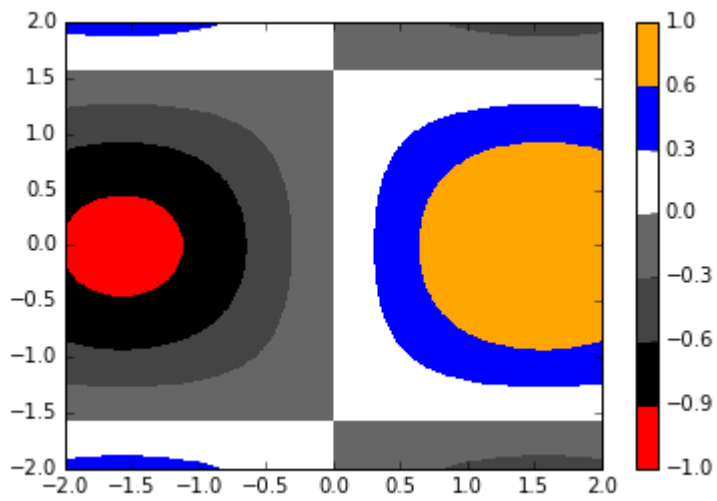
x = np.linspace(-2,2,500)
y = np.linspace(-2,2,500)
XX, YY = np.meshgrid(x, y)
Z = np.sin(XX) * np.cos(YY)

cmap = colors.ListedColormap(['red', '#000000', '#444444', '#666666', '#ffffff', 'blue',
                              'orange'])
boundaries = [-1, -0.9, -0.6, -0.3, 0, 0.3, 0.6, 1]
norm = colors.BoundaryNorm(boundaries, cmap.N, clip=True)

plt.pcolormesh(x,y,Z, cmap=cmap, norm=norm)
```

```
plt.colorbar()  
plt.show()
```

プロデュース



i は i と $i+1$ ののにされます。は 'red'、 'green'、 HTMLコード '#ffaa44'、 '#441188' またはRGB タプル (0.2, 0.9, 0.45) '#441188' (0.2, 0.9, 0.45) でできます。

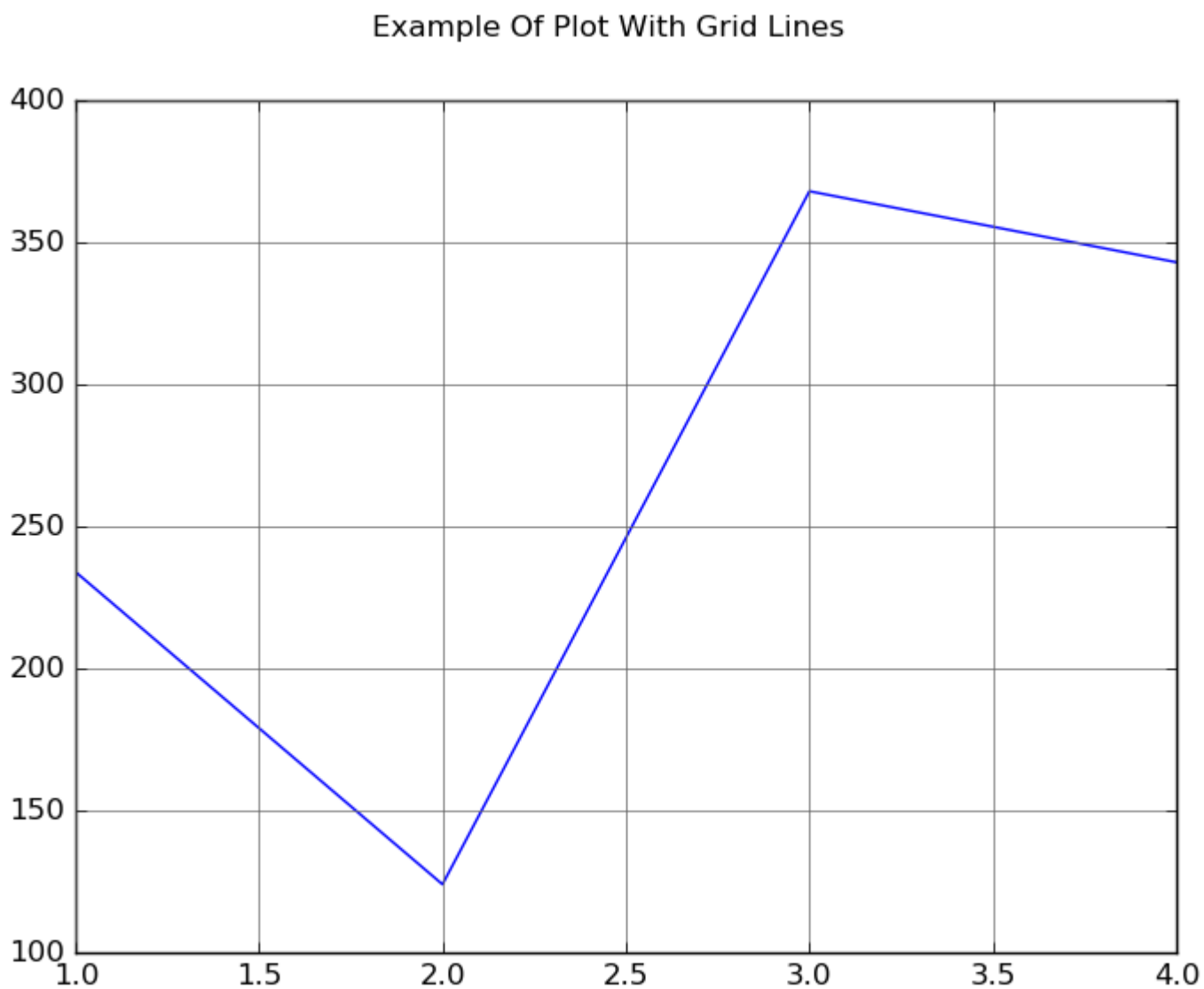
オンラインでカラーマップをむ <https://riptutorial.com/ja/matplotlib/topic/3385/カラーマップ>

8: グリッドとり

Examples

グリッドラインでプロット

グリッドをプロット



```
import matplotlib.pyplot as plt

# The Data
x = [1, 2, 3, 4]
y = [234, 124, 368, 343]

# Create the figure and axes objects
```



```
fig, ax = plt.subplots(1, figsize=(8, 6))
fig.suptitle('Example Of Plot With Grid Lines')

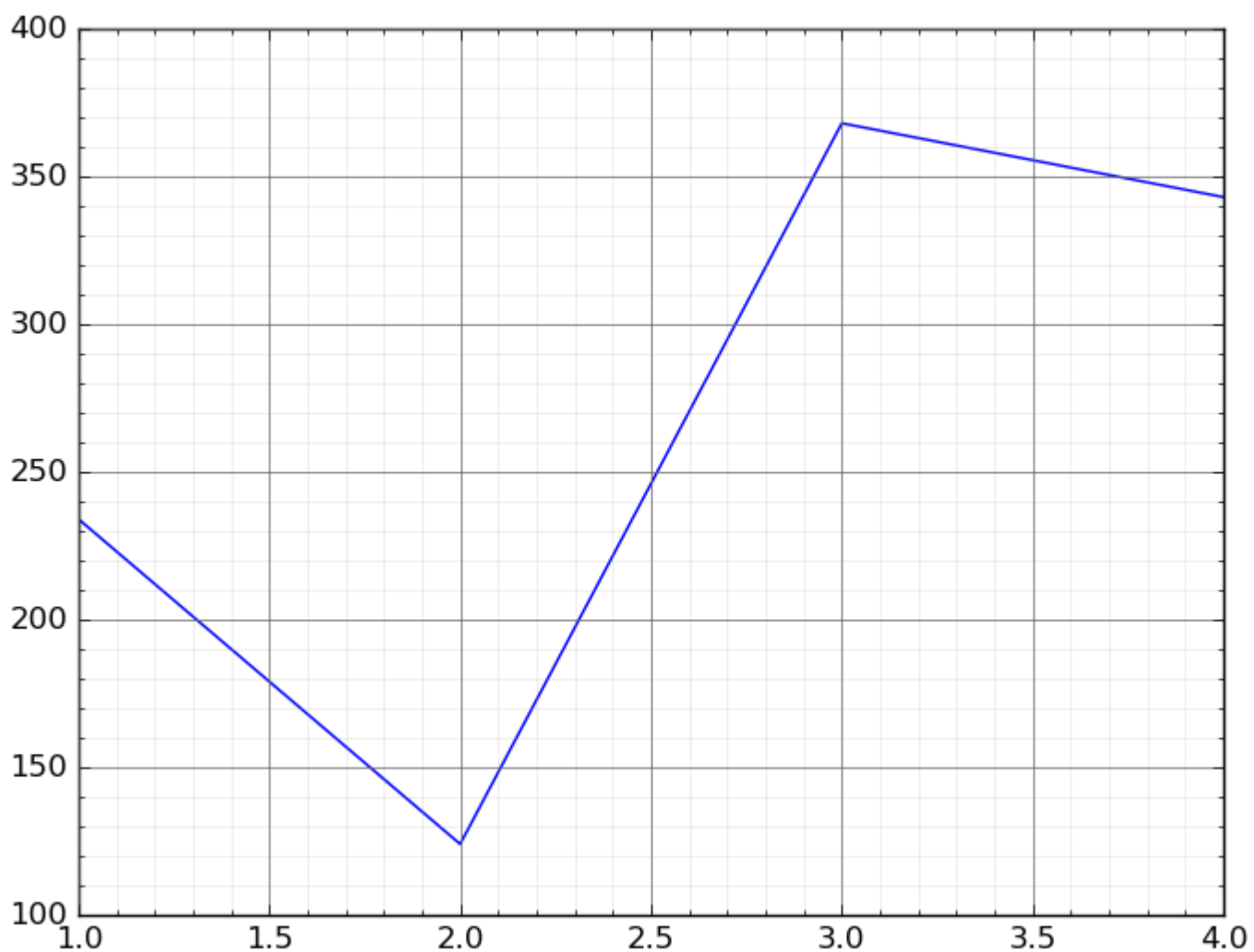
# Plot the data
ax.plot(x,y)

# Show the grid lines as dark grey lines
plt.grid(b=True, which='major', color='#666666', linestyle='-')

plt.show()
```

メジャーとマイナーグリッドラインをプロット

Example Of Plot With Major and Minor Grid Lines



```
import matplotlib.pyplot as plt

# The Data
x = [1, 2, 3, 4]
y = [234, 124, 368, 343]
```

```
# Create the figure and axes objects
fig, ax = plt.subplots(1, figsize=(8, 6))
fig.suptitle('Example Of Plot With Major and Minor Grid Lines')

# Plot the data
ax.plot(x,y)

# Show the major grid lines with dark grey lines
plt.grid(b=True, which='major', color='#666666', linestyle='-')

# Show the minor grid lines with very faint and almost transparent grey lines
plt.minorticks_on()
plt.grid(b=True, which='minor', color='#999999', linestyle='-', alpha=0.2)

plt.show()
```

オンラインでグリッドとりをむ <https://riptutorial.com/ja/matplotlib/topic/4029/グリッドとり>

9: ヒストグラム

Examples

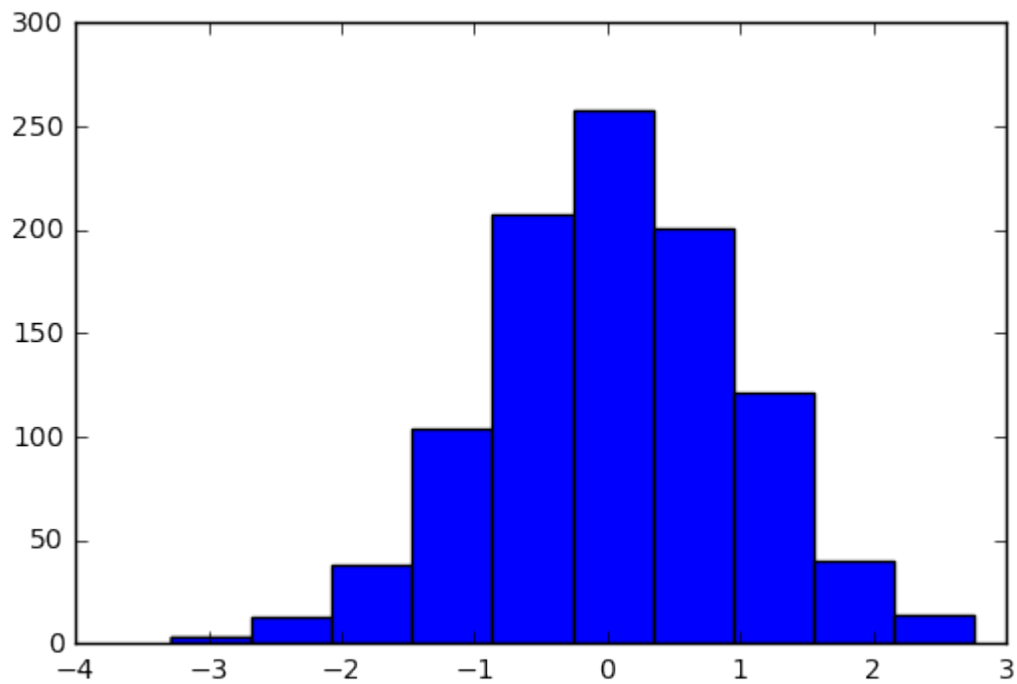
シンプルなヒストグラム

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

# generate 1000 data points with normal distribution
data = np.random.randn(1000)

plt.hist(data)

plt.show()
```



オンラインでヒストグラムをむ <https://riptutorial.com/ja/matplotlib/topic/7329/ヒストグラム>

10: ボックスプロット

Examples

ボックスプロット

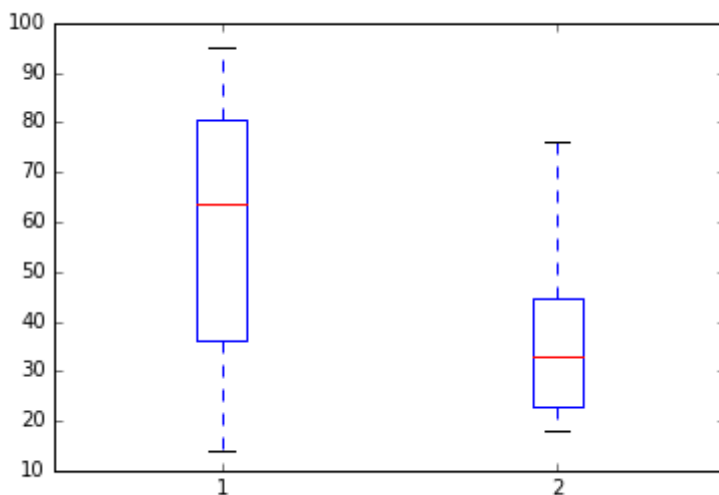
ボックスプロットは、さまざまなデータの値を視覚化するのに役立ちます。それらは値の分布を視覚化するために役立ちます。

matplotlibのboxplotの機能は、データをリストのリストとして提供することで簡単に使用できます

```
import matplotlib as plt

dataline1 = [43,76,34,63,56,82,87,55,64,87,95,23,14,65,67,25,23,85]
dataline2 = [34,45,34,23,43,76,26,18,24,74,23,56,23,23,34,56,32,23]
data = [ dataline1, dataline2 ]

plt.boxplot( data )
```

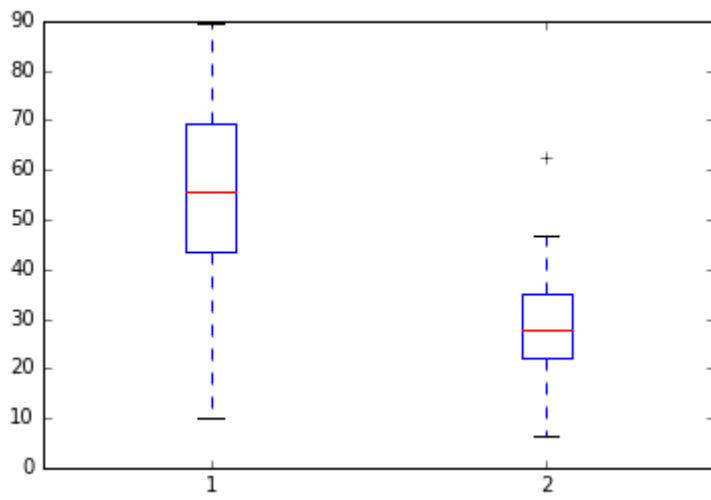


しかし、このようにすることがないので、numpyをプロットのパラメータとして使用する必要があります。これは次のように行うことができます。

```
import numpy as np
import matplotlib as plt

np.random.seed(123)
dataline1 = np.random.normal( loc=50, scale=20, size=18 )
dataline2 = np.random.normal( loc=30, scale=10, size=18 )
data = np.stack( [ dataline1, dataline2 ], axis=1 )

plt.boxplot( data )
```



オンラインでボックスプロットをむ <https://riptutorial.com/ja/matplotlib/topic/6086/ボックスプロット>

11: ボックスプロット

Examples

Boxplot

`matplotlib`のは、のがあるひげを。こののするは、デフォルトで`boxplot`が50をいでしていることです。ボックスはQ1とQ3パーセンタイル25と75をし、ウィスカーはデータのをしますQ1 - 1.5 IQR、Q3 + 1.5 IQR、IQRはですが、がします。また、このをえるサンプルはマーカーとしてされますこれらはチラシです。

`boxplot`のすべてののがじルールにうわけではありません。おそらく、もな`boxplot`ダイアグラムでは、ひげをとってとをしていますしない。また、このプロットはボックスアンドウィスカープロット、ボックスアンドウィスカーダイアグラムともばれます。

のレシピは、の`matplotlib`の`boxplot`のでできることをしています。

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

X1 = np.random.normal(0, 1, 500)
X2 = np.random.normal(0.3, 1, 500)

# The most simple boxplot
plt.boxplot(X1)
plt.show()

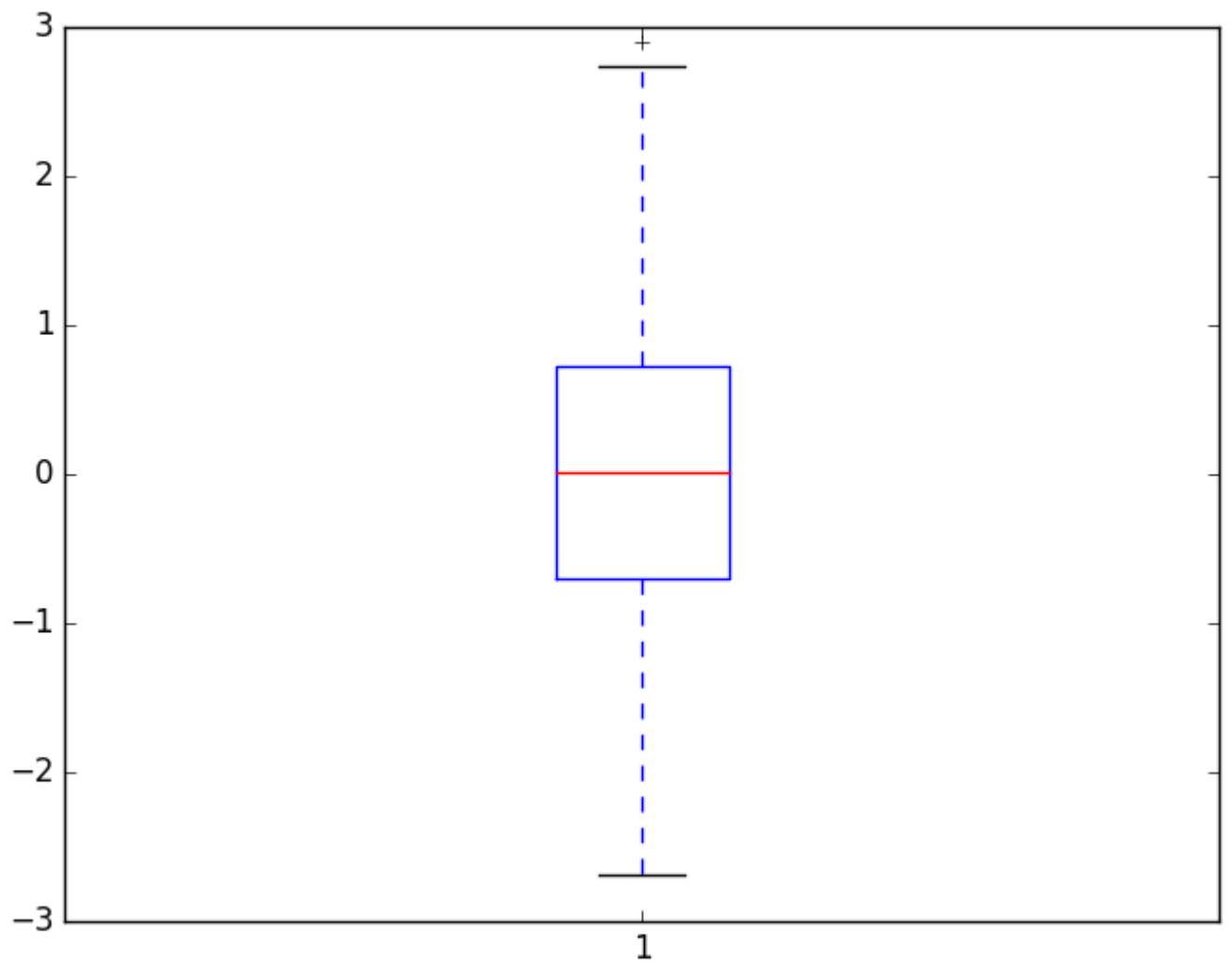
# Changing some of its features
plt.boxplot(X1, notch=True, sym="o") # Use sym="" to shown no fliers; also showfliers=False
plt.show()

# Showing multiple boxplots on the same window
plt.boxplot((X1, X2), notch=True, sym="o", labels=["Set 1", "Set 2"])
plt.show()

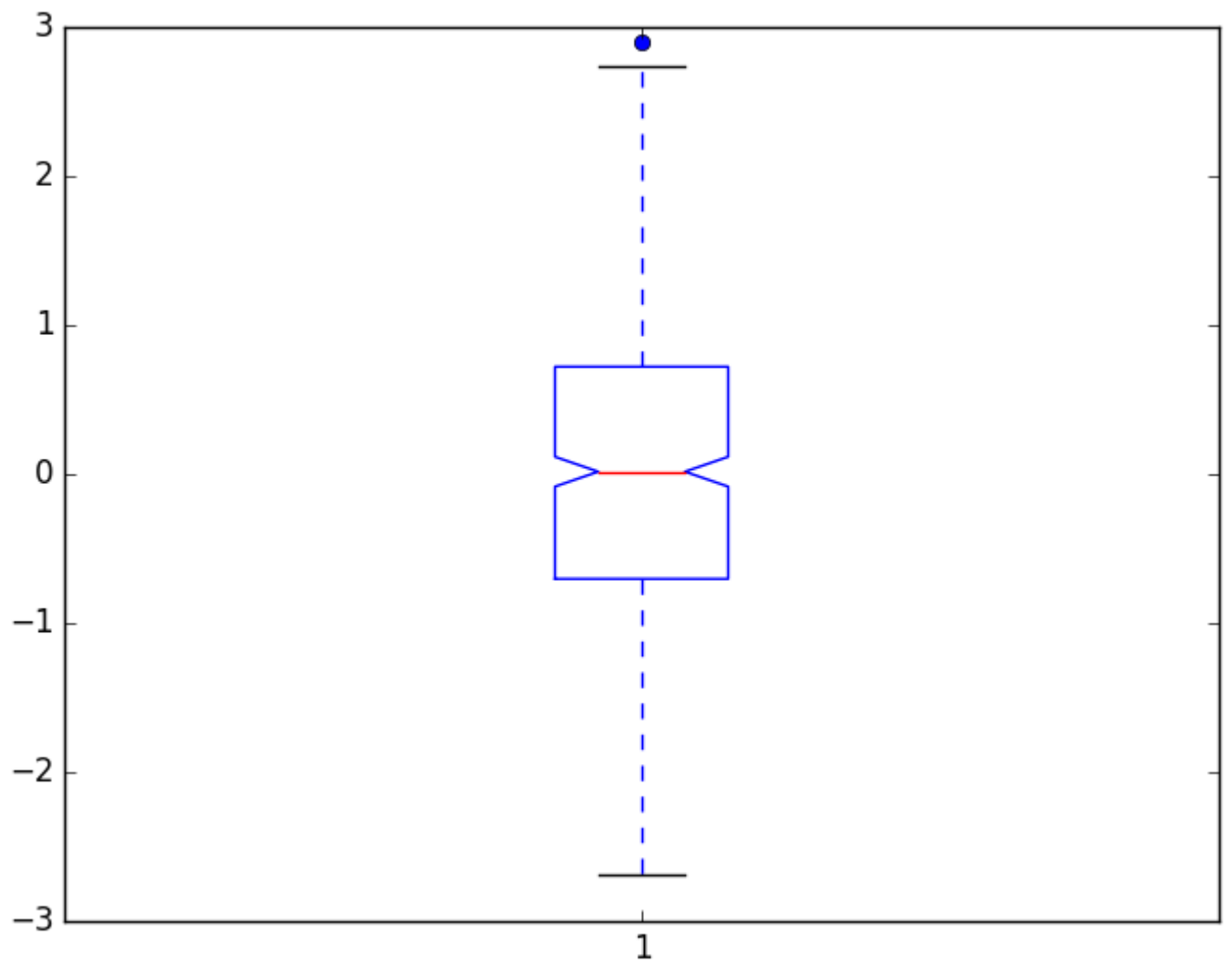
# Hidding features of the boxplot
plt.boxplot(X2, notch=False, showfliers=False, showbox=False, showcaps=False, positions=[4],
labels=["Set 2"])
plt.show()

# Advanced customization of the boxplot
line_props = dict(color="r", alpha=0.3)
bbox_props = dict(color="g", alpha=0.9, linestyle="dashdot")
flier_props = dict(marker="o", markersize=17)
plt.boxplot(X1, notch=True, whiskerprops=line_props, boxprops=bbox_props,
flierprops=flier_props)
plt.show()
```

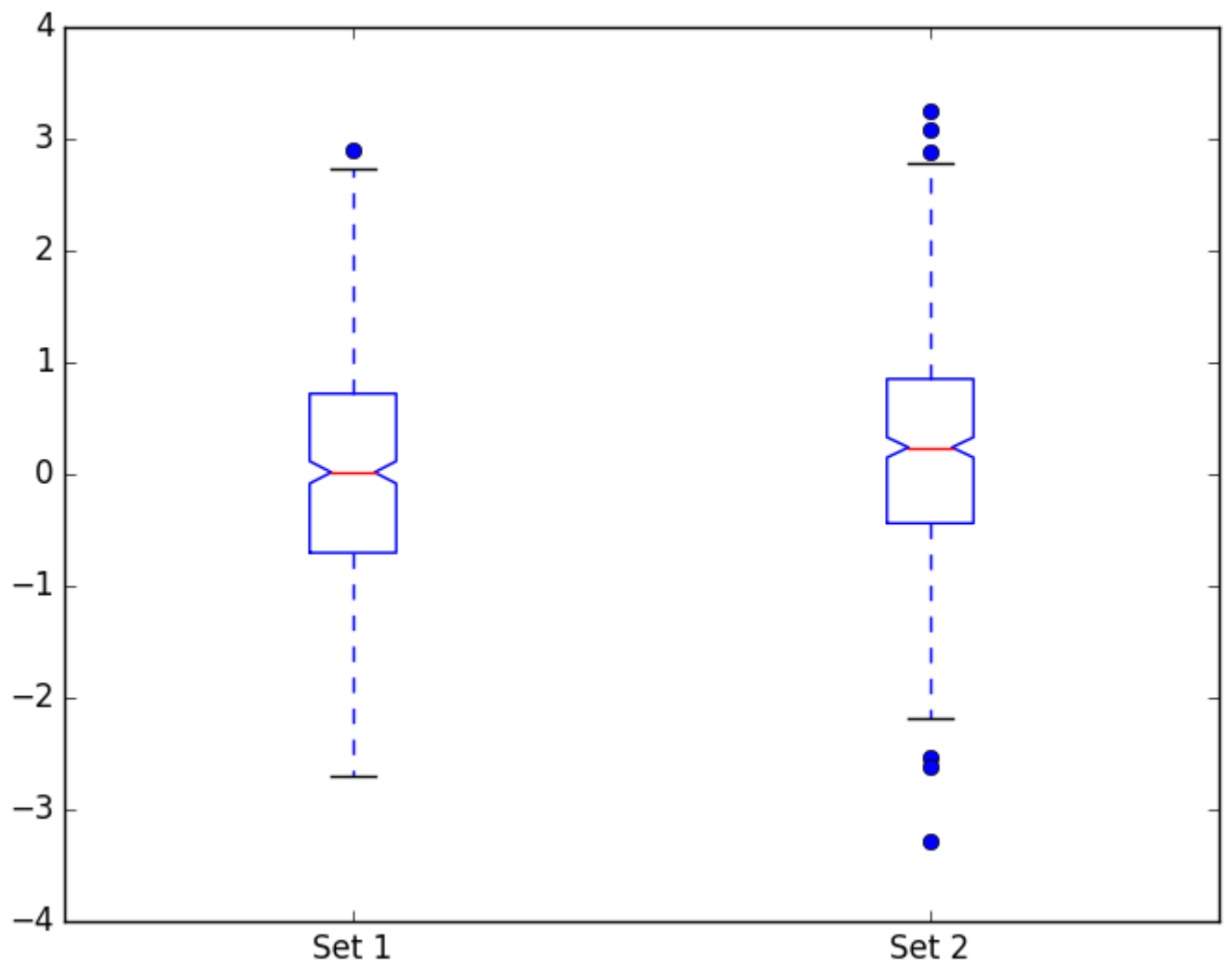
これにより、のプロットがられます。



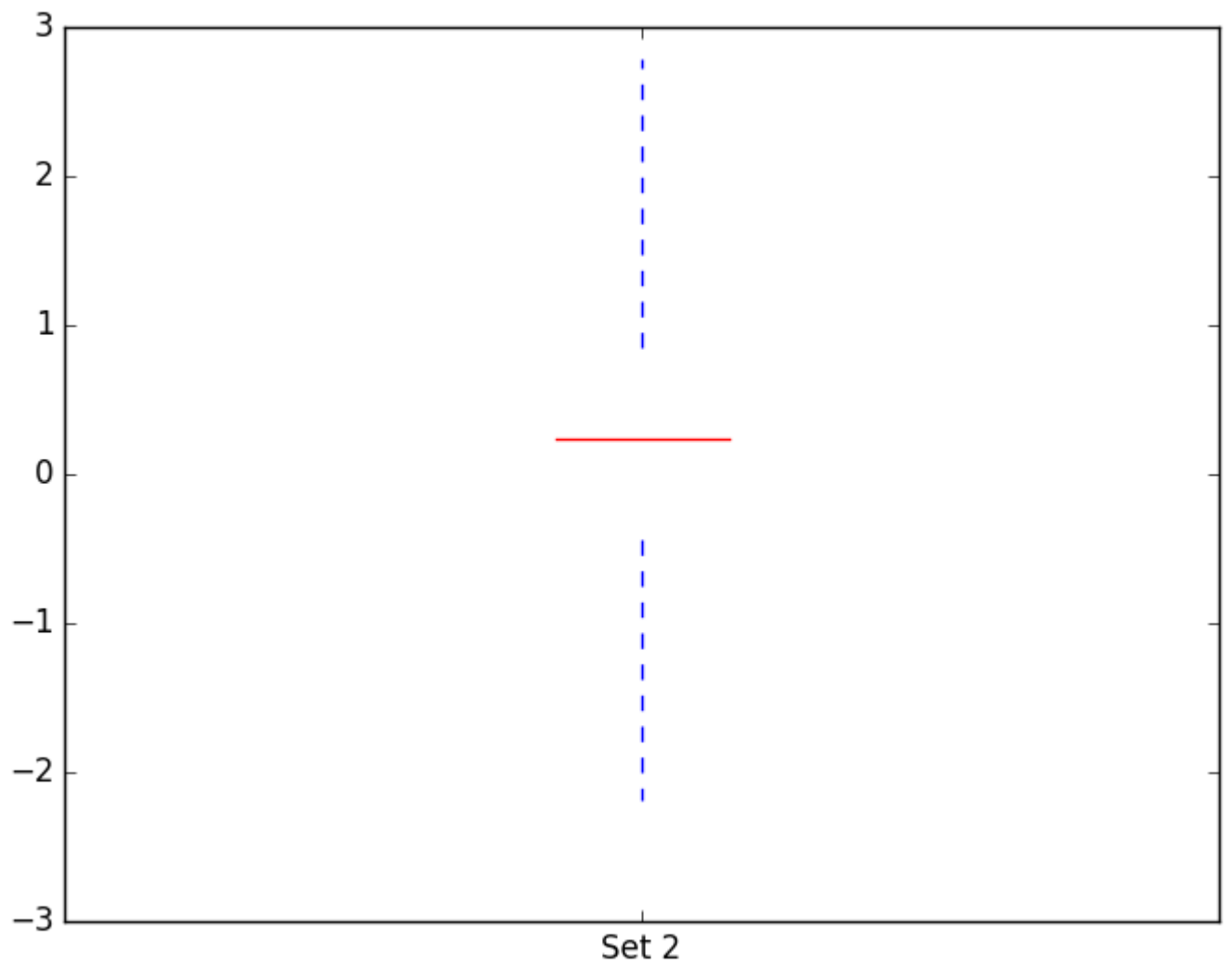
1. デフォルトの`matplotlib boxplot`



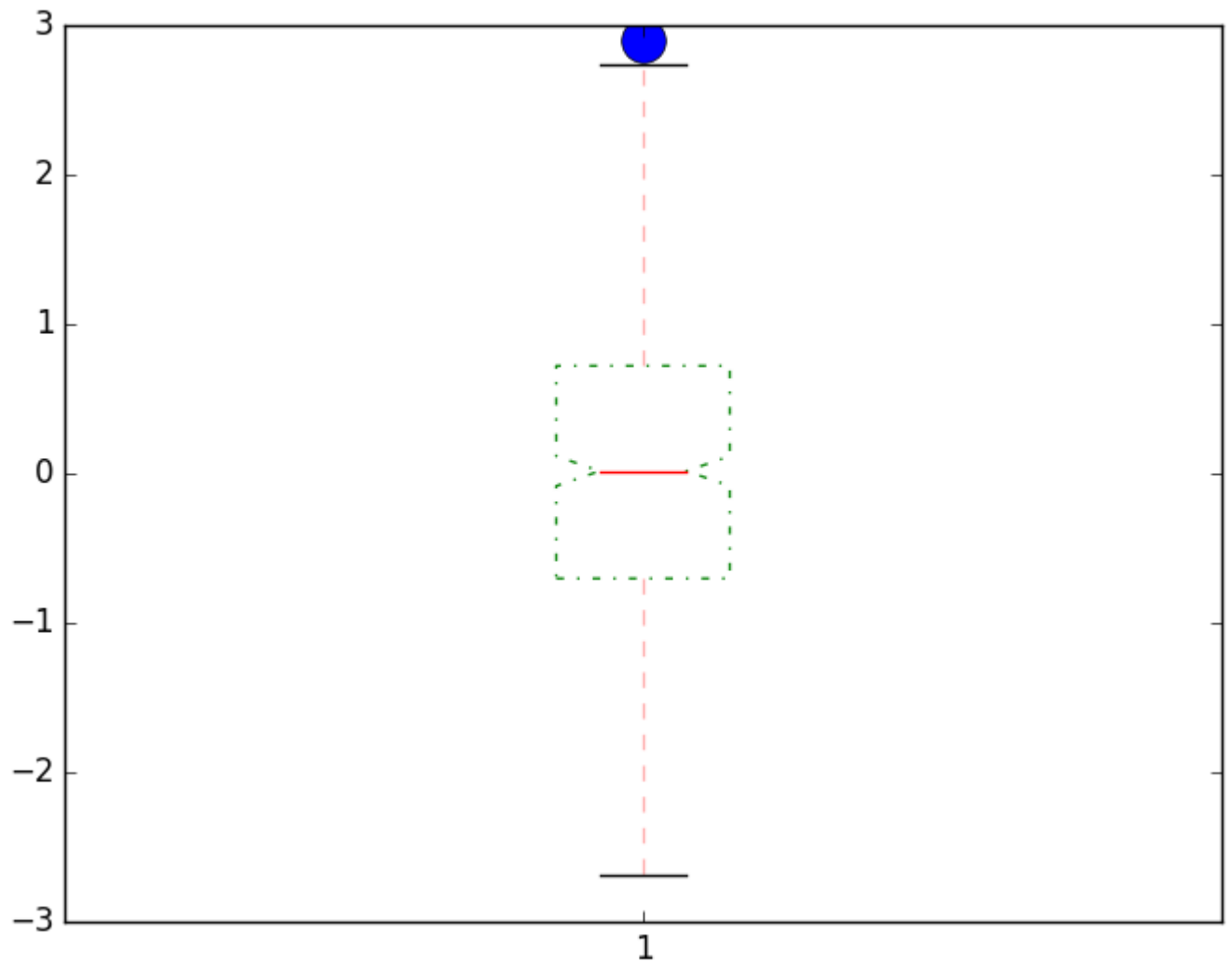
2. のをって *boxplot* のいくつかのをする



3. ジプロットウィンドウののボックスプロット



4. ボックスプロットのいくつかのをす



5. をったボックスプロットのなカスタマイズ

あなたのひげのいくつかのなカスタマイズをうつもりならば、あなたはそれをとておくべきは、あなたがたとえはの

```
line_props = dict(color="r", alpha=0.3)
bbox_props = dict(color="g", alpha=0.9, linestyle="dashdot")
flier_props = dict(marker="o", markersize=17)
plt.boxplot(X1, notch=True, whiskerprops=line_props, boxprops=bbox_props,
            flierprops=flier_props)
plt.show()
```

...ほとんどすべてをLine2Dオブジェクトにします。これは、そのクラスでなだけがであることをします。 whiskerprops、 boxprops、 flierprops、 およびcapprops boxpropsなどのキーワードのにつくでしょう。これらは、それをさらにカスタマイズするためのをするためになです。

このをしたボックスプロットのカスタマイズをさらににするがあります。いくつかのでは、 [パッチ](#)などののmatplotlibホームをしてボックスプロットをすることがで

ありるえば、ボックスにかなりのがえられる。

オンラインでボックスプロットをむ <https://riptutorial.com/ja/matplotlib/topic/6368/ボックスプロット>

12:

Examples

な

じプロットにのがあり、それぞれがなるをち、がをしているかをえるをりたいとします。 `plot()` をびすと、にラベルをすことでこれをうことができます。たとえば、のに *"My Line 1"* というラベルがかけられます。

```
ax.plot(x, y1, color="red", label="My Line 1")
```

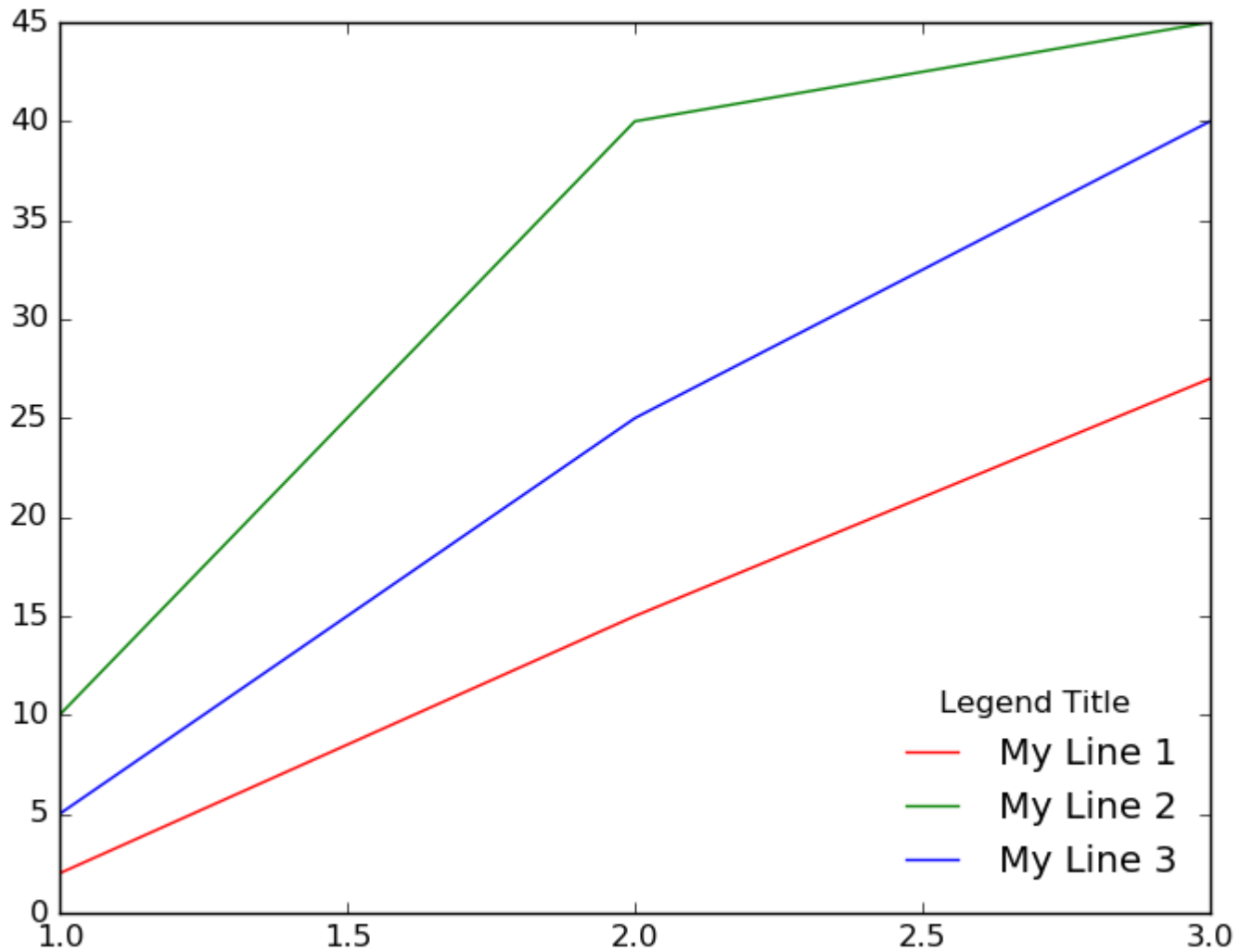
これは、そののにされるテキストをします。のをするために、 `ax.legend()` びすことができます。

デフォルトでは、プロットののボックスにがされます。 `legend()` にをして、それをカスタマイズすることができます。たとえば、フレームボックスをむようににし、のをびしてのタイトルをすることができます。

```
ax.legend(loc="lower right", title="Legend Title", frameon=False)
```

はそうです

Simple Legend Example



```
import matplotlib.pyplot as plt

# The data
x = [1, 2, 3]
y1 = [2, 15, 27]
y2 = [10, 40, 45]
y3 = [5, 25, 40]

# Initialize the figure and axes
fig, ax = plt.subplots(1, figsize=(8, 6))

# Set the title for the figure
fig.suptitle('Simple Legend Example ', fontsize=15)

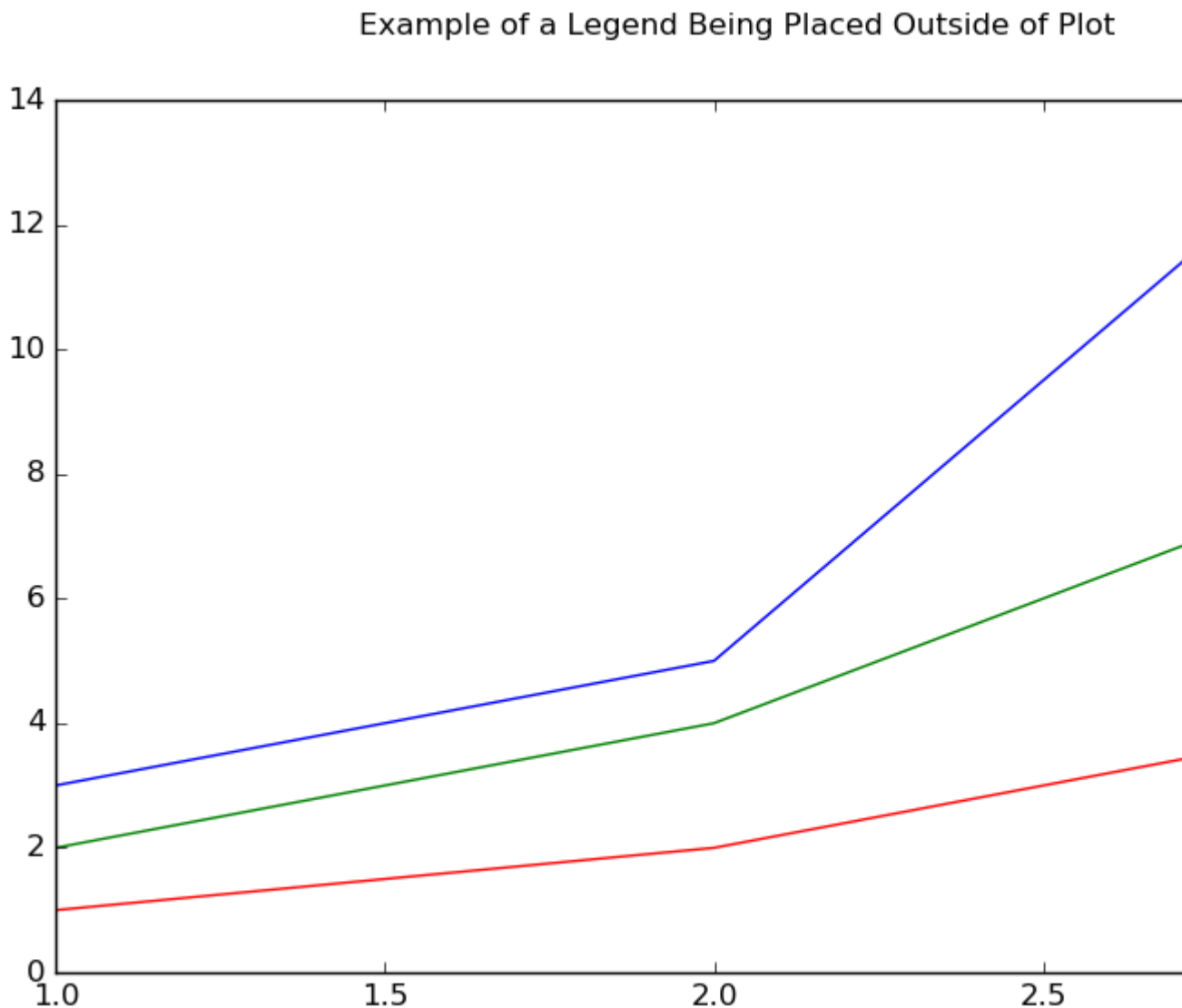
# Draw all the lines in the same plot, assigning a label for each one to be
# shown in the legend
ax.plot(x, y1, color="red", label="My Line 1")
ax.plot(x, y2, color="green", label="My Line 2")
ax.plot(x, y3, color="blue", label="My Line 3")

# Add a legend with title, position it on the lower right (loc) with no box framing (frameon)
ax.legend(loc="lower right", title="Legend Title", frameon=False)
```

```
# Show the plot  
plt.show()
```

プロットのにかれた

をプロットのにするがあるか、またはましいがあります。のコードは、それをうをしています。



```
import matplotlib.pyplot as plt  
fig, ax = plt.subplots(1, 1, figsize=(10,6)) # make the figure with the size 10 x 6 inches  
fig.suptitle('Example of a Legend Being Placed Outside of Plot')  
  
# The data  
x = [1, 2, 3]  
y1 = [1, 2, 4]  
y2 = [2, 4, 8]  
y3 = [3, 5, 14]  
  
# Labels to use for each line  
line_labels = ["Item A", "Item B", "Item C"]
```

```

# Create the lines, assigning different colors for each one.
# Also store the created line objects
l1 = ax.plot(x, y1, color="red")[0]
l2 = ax.plot(x, y2, color="green")[0]
l3 = ax.plot(x, y3, color="blue")[0]

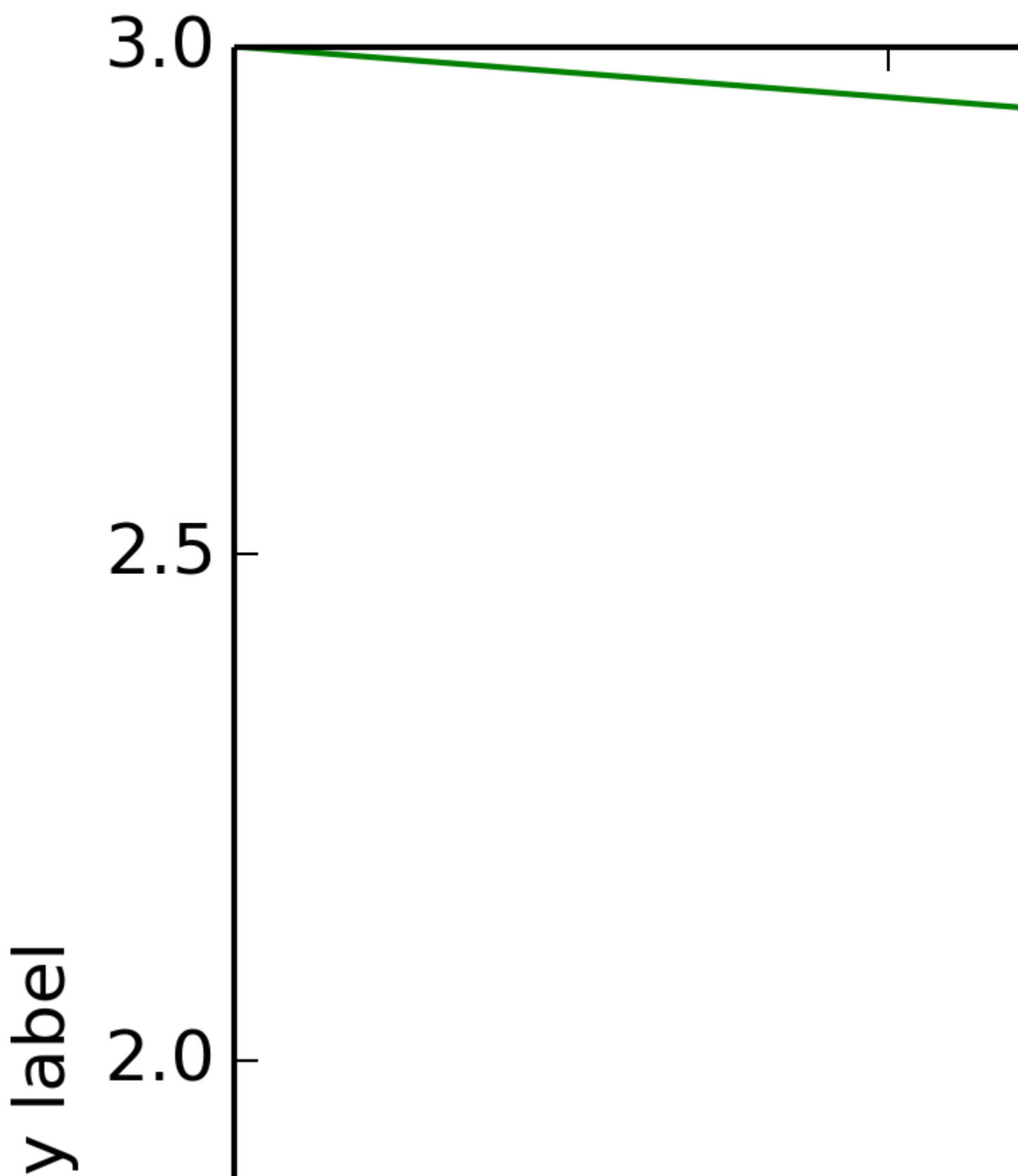
fig.legend([l1, l2, l3],          # List of the line objects
           labels= line_labels,   # The labels for each line
           loc="center right",    # Position of the legend
           borderaxespad=0.1,     # Add little spacing around the legend box
           title="Legend Title")  # Title for the legend

# Adjust the scaling factor to fit your legend text completely outside the plot
# (smaller value results in more space being made for the legend)
plt.subplots_adjust(right=0.85)

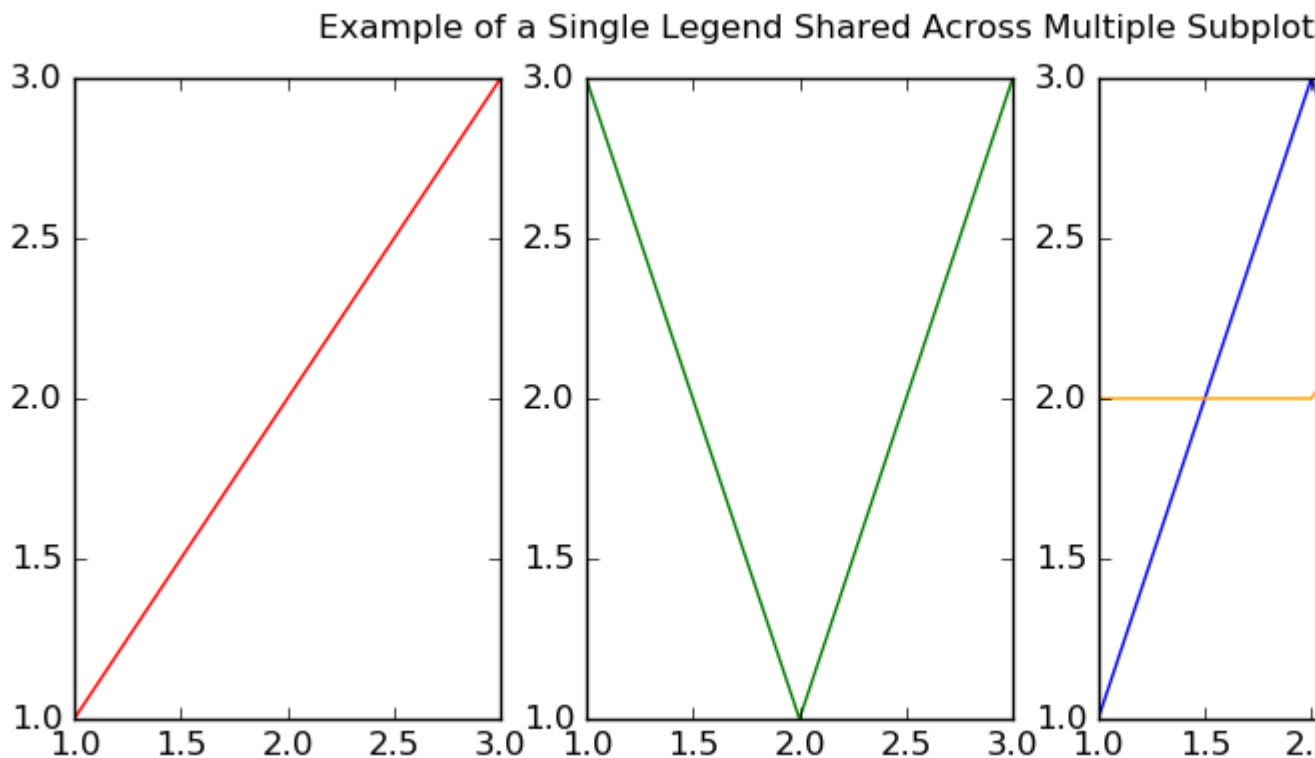
plt.show()

```

をプロットのにするもうつのは、のにすように、 `bbox_to_anchor + bbox_extra_artists +`
`bbox_inches='tight'` をすること `bbox_to_anchor`。



したいがあります。



これをうには、レベルでををするのではなく、のグローバルをするがありますサブプロットごとにのがされます。これは、のコードのコードでられるように、`fig.legend()`をびすことによってされます。

```
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(10,4))
fig.suptitle('Example of a Single Legend Shared Across Multiple Subplots')

# The data
x = [1, 2, 3]
y1 = [1, 2, 3]
y2 = [3, 1, 3]
y3 = [1, 3, 1]
y4 = [2, 2, 3]

# Labels to use in the legend for each line
line_labels = ["Line A", "Line B", "Line C", "Line D"]

# Create the sub-plots, assigning a different color for each line.
# Also store the line objects created
l1 = ax1.plot(x, y1, color="red")[0]
l2 = ax2.plot(x, y2, color="green")[0]
l3 = ax3.plot(x, y3, color="blue")[0]
l4 = ax3.plot(x, y4, color="orange")[0] # A second line in the third subplot

# Create the legend
fig.legend([l1, l2, l3, l4],          # The line objects
           labels=line_labels,        # The labels for each line
           loc="center right",        # Position of legend
           borderaxespad=0.1,         # Small spacing around legend box
           title="Legend Title"       # Title for the legend
          )
```

```
# Adjust the scaling factor to fit your legend text completely outside the plot
# (smaller value results in more space being made for the legend)
plt.subplots_adjust(right=0.85)

plt.show()
```

ののはのとおりです。

```
l1 = ax1.plot(x, y1, color="red")[0]
```

`plot()` がびされると、**line2D**オブジェクトのリストがされます。この、ただ1つの**line2D**オブジェクトをつリストがされます。リストは`[0]`インデックスでされ、`l1`されます。

にめることにのがあるすべての**line2D**オブジェクトのリストは、`fig.legend()`のとしてすがあります。
 ◦ `fig.legend()`の2もです。これは、ののラベルとしてするのリストであるとえられています。

`fig.legend()`されるのはにオプションで、のしさをするのにちます。

じのの

`plt.legend()`または`ax.legend()`びすと、のがされ、しいがされます。によると

legendをりしびしてをAxesののハンドルにできるようにしました

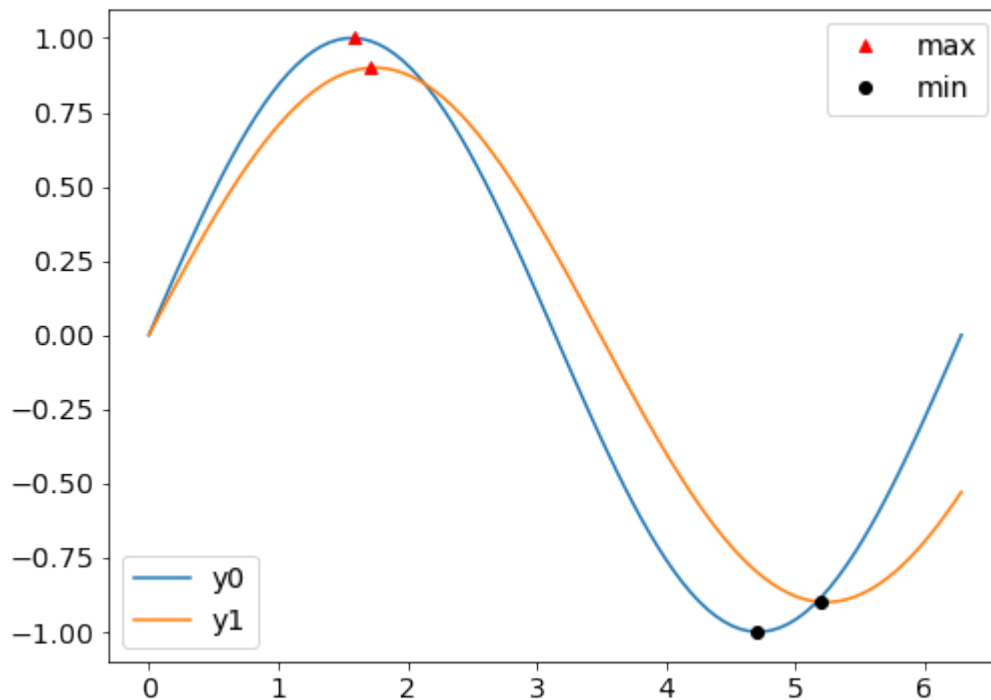
しかしではありません。に2のまたは3、または4のをするのはまだです。このでは、2のをプロットし、それぞれのとにマーカーをプロットします。1つののはのためのものであり、もう1つはマーカーのためのものです。

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

# Generate data for plotting:
x = np.linspace(0,2*np.pi,100)
y0 = np.sin(x)
y1 = .9*np.sin(.9*x)
# Find their maxima and minima and store
maxes = np.empty((2,2))
mins = np.empty((2,2))
for k,y in enumerate([y0,y1]):
    maxloc = y.argmax()
    maxes[k] = x[maxloc], y[maxloc]
    minloc = y.argmin()
    mins[k] = x[minloc], y[minloc]

# Instantiate figure and plot
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(x,y0, label='y0')
ax.plot(x,y1, label='y1')
# Plot maxima and minima, and keep references to the lines
maxline, = ax.plot(maxes[:,0], maxes[:,1], 'r^')
minline, = ax.plot(mins[:,0], mins[:,1], 'ko')
```

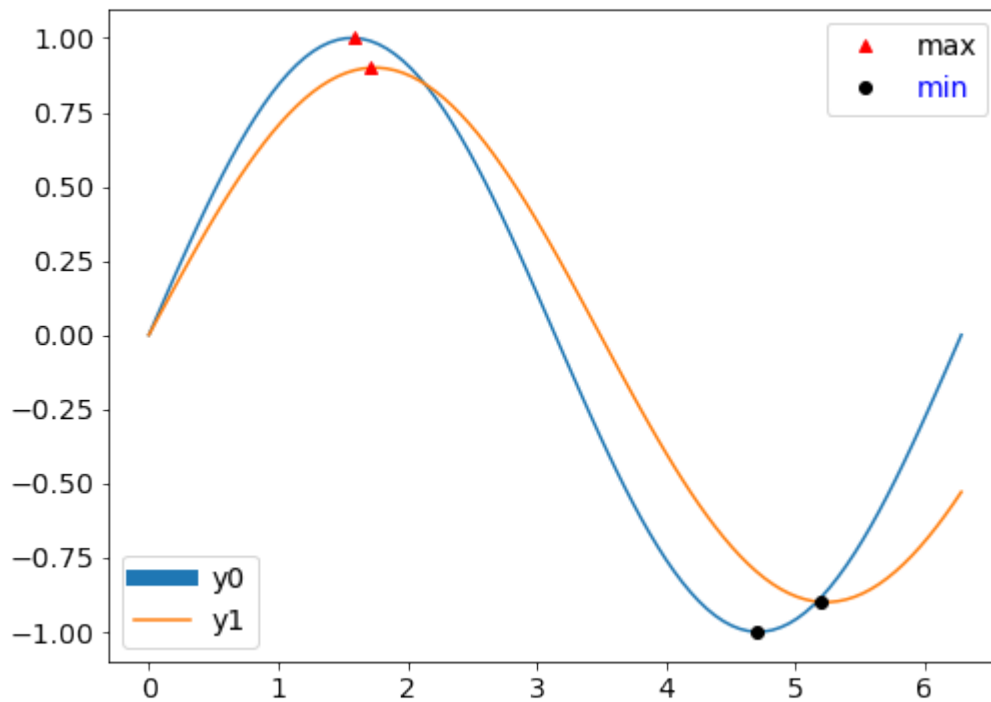
```
# Add first legend: only labeled data is included
leg1 = ax.legend(loc='lower left')
# Add second legend for the maxes and mins.
# leg1 will be removed from figure
leg2 = ax.legend([maxline,minline],['max','min'], loc='upper right')
# Manually add the first legend back
ax.add_artist(leg1)
```



なは、オブジェクトへのがあることをすることです。もしインスタンスのもの `leg1` 2つをすると、からされるが、`leg1` オブジェクトがまだし、バックすることができる `ax.add_artist`。

にらしいことは、あなたはまだのをすることができるということです。たとえば、のコードをのコードのにします。

```
leg1.get_lines()[0].set_lw(8)
leg2.get_texts()[1].set_color('b')
```



に、このでは、プロットされたときにラベルにラベルがえられただけで、`ax.legend()` はそれらのだけを `ax.legend()` することに `ax.legend()` して `leg1`。したがって、マーカーの `leg2` は、インスタンスされたときに、とラベルをとしてとしました。あるいは、プロットされたマーカーをマーカーにけることもできます。しかし、`ax.legend` へののびしには、それぞれのになのみがまれるように、いくつかのながでした。

オンラインでをむ <https://riptutorial.com/ja/matplotlib/topic/2840/>

13: とオブジェクト

Examples

をする

にはすべてのプロットがまれています。 `matplotlib` Figureをするなは、 `pyplot` をうこと `pyplot` 。

```
import matplotlib.pyplot as plt
fig = plt.figure()
```

オプションで、にしたにアクセスするためのをすることができます。がされていないは、にされたFigureのIDがインクリメントされてわりにされます。は0ではなく1からインデックスされます。

```
import matplotlib.pyplot as plt
fig = plt.figure()
fig == plt.figure(1) # True
```

のわりにもでできます。インタラクティブバックエンドをしているは、ウィンドウのタイトルもされます。

```
import matplotlib.pyplot as plt
fig = plt.figure('image')
```

のをするには

```
plt.figure(fig.number) # or
plt.figure(1)
```

の

`matplotlib`にをするには、`pyplot`をうと、オブジェクトのAPIをうがあります。

`pyplot`をう

```
import matplotlib.pyplot as plt

ax = plt.subplot(3, 2, 1) # 3 rows, 2 columns, the first subplot
```

オブジェクトAPIの

```
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_subplot(3, 2, 1)
```

な `plt.subplots()` をして、1つのコマンドでとサブプロットのコレクションをできます。

```
import matplotlib.pyplot as plt

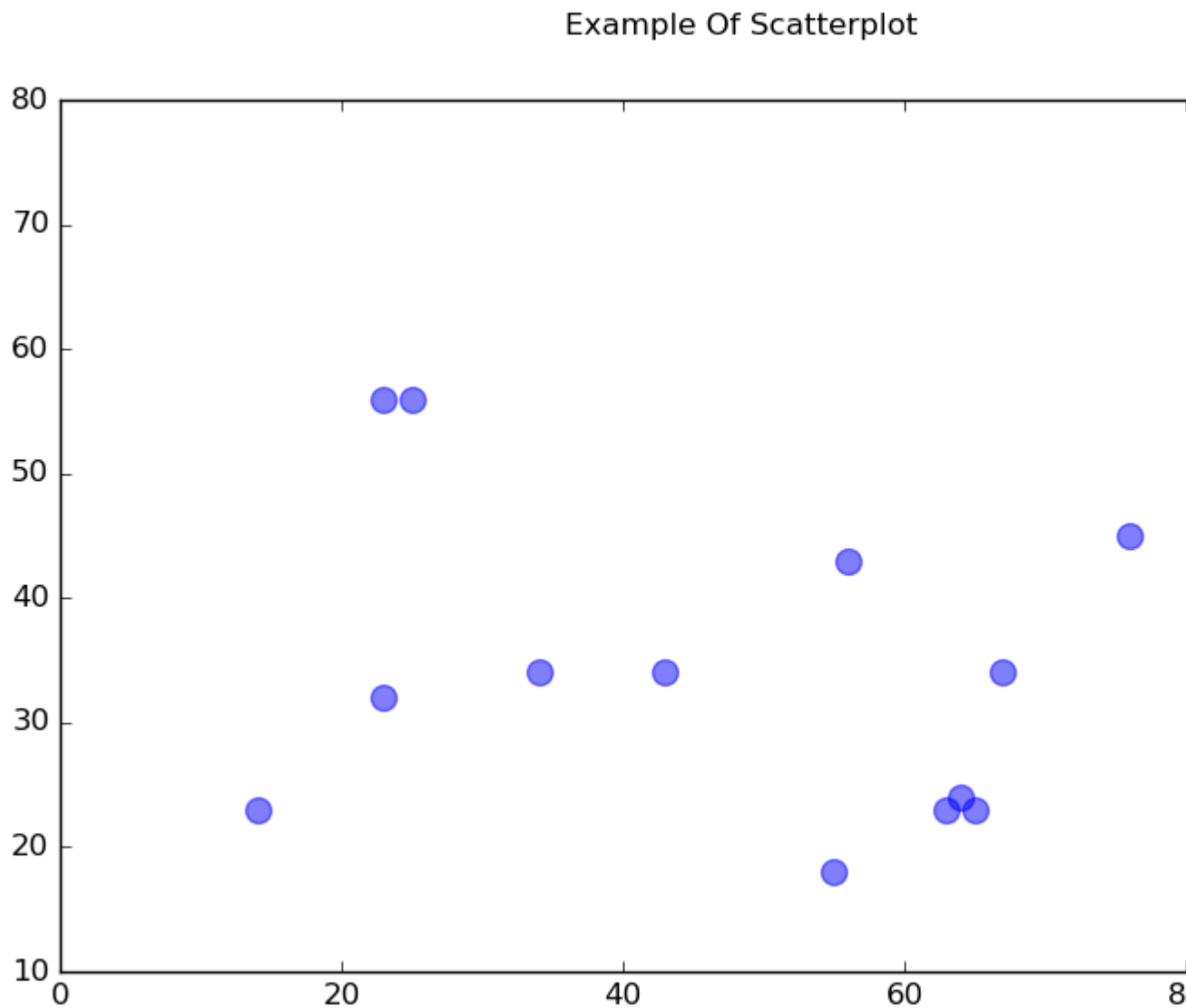
fig, (ax1, ax2) = plt.subplots(ncols=2, nrows=1) # 1 row, 2 columns
```

オンラインでとオブジェクトをむ <https://riptutorial.com/ja/matplotlib/topic/2307/>とオブジェクト

14: プロット

Examples

な



```
import matplotlib.pyplot as plt

# Data
x = [43,76,34,63,56,82,87,55,64,87,95,23,14,65,67,25,23,85]
y = [34,45,34,23,43,76,26,18,24,74,23,56,23,23,34,56,32,23]

fig, ax = plt.subplots(1, figsize=(10, 6))
fig.suptitle('Example Of Scatterplot')

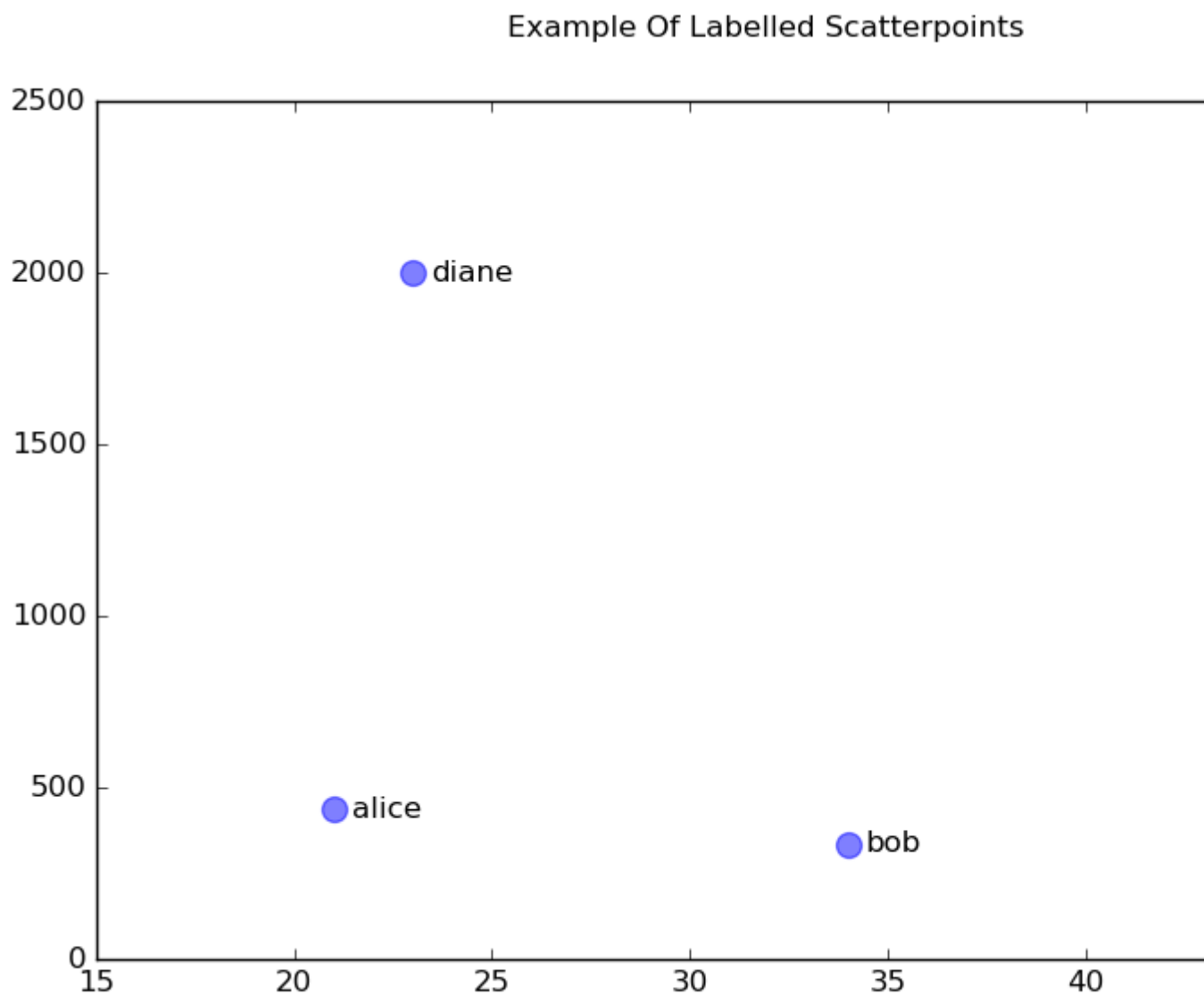
# Create the Scatter Plot
ax.scatter(x, y,
```



```
color="blue",      # Color of the dots
s=100,            # Size of the dots
alpha=0.5,        # Alpha/transparency of the dots (1 is opaque, 0 is transparent)
linewidths=1)     # Size of edge around the dots

# Show the plot
plt.show()
```

ラベルきをむ



```
import matplotlib.pyplot as plt

# Data
x = [21, 34, 44, 23]
y = [435, 334, 656, 1999]
labels = ["alice", "bob", "charlie", "diane"]

# Create the figure and axes objects
fig, ax = plt.subplots(1, figsize=(10, 6))
fig.suptitle('Example Of Labelled Scatterpoints')
```

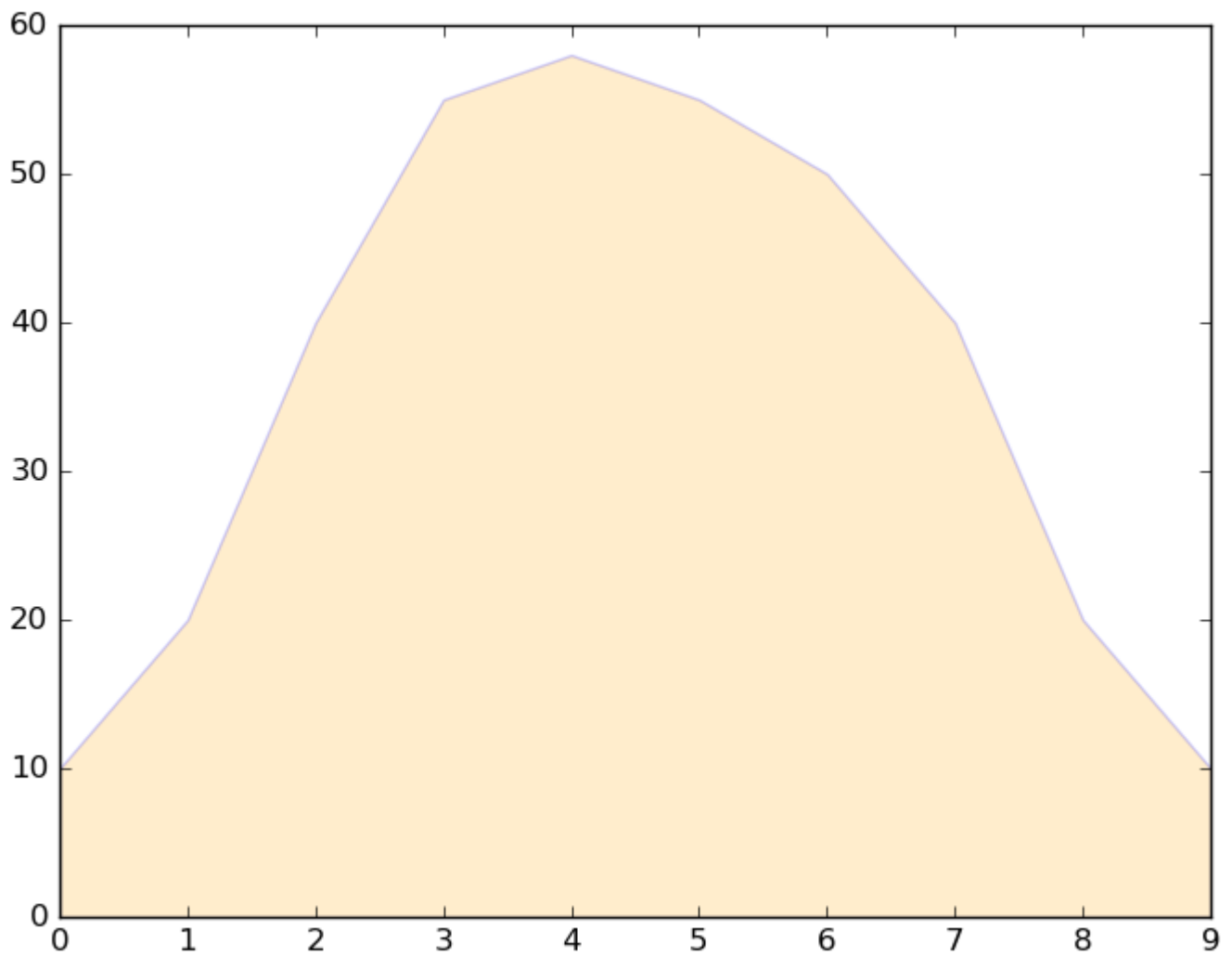
```
# Plot the scatter points
ax.scatter(x, y,
           color="blue", # Color of the dots
           s=100,        # Size of the dots
           alpha=0.5,    # Alpha of the dots
           linewidths=1) # Size of edge around the dots

# Add the participant names as text labels for each point
for x_pos, y_pos, label in zip(x, y, labels):
    ax.annotate(label, # The label for this point
                xy=(x_pos, y_pos), # Position of the corresponding point
                xytext=(7, 0),     # Offset text by 7 points to the right
                textcoords='offset points', # tell it to use offset points
                ha='left',         # Horizontally aligned to the left
                va='center')      # Vertical alignment is centered

# Show the plot
plt.show()
```

きプロット

のののいた



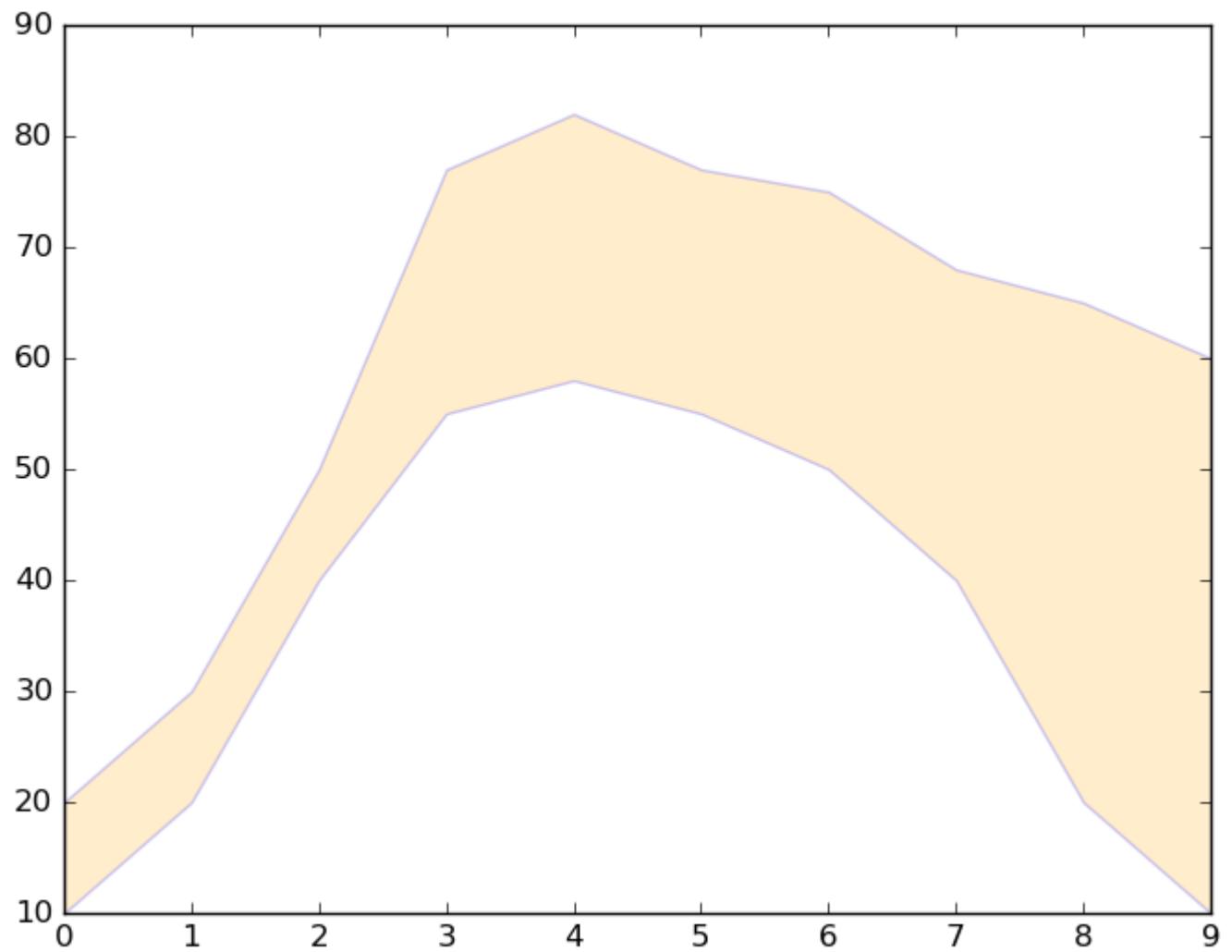
```
import matplotlib.pyplot as plt

# Data
x = [0,1,2,3,4,5,6,7,8,9]
y1 = [10,20,40,55,58,55,50,40,20,10]

# Shade the area between y1 and line y=0
plt.fill_between(x, y1, 0,
                 facecolor="orange", # The fill color
                 color='blue',      # The outline color
                 alpha=0.2)         # Transparency of the fill

# Show the plot
plt.show()
```

のいた2つののの



```
import matplotlib.pyplot as plt

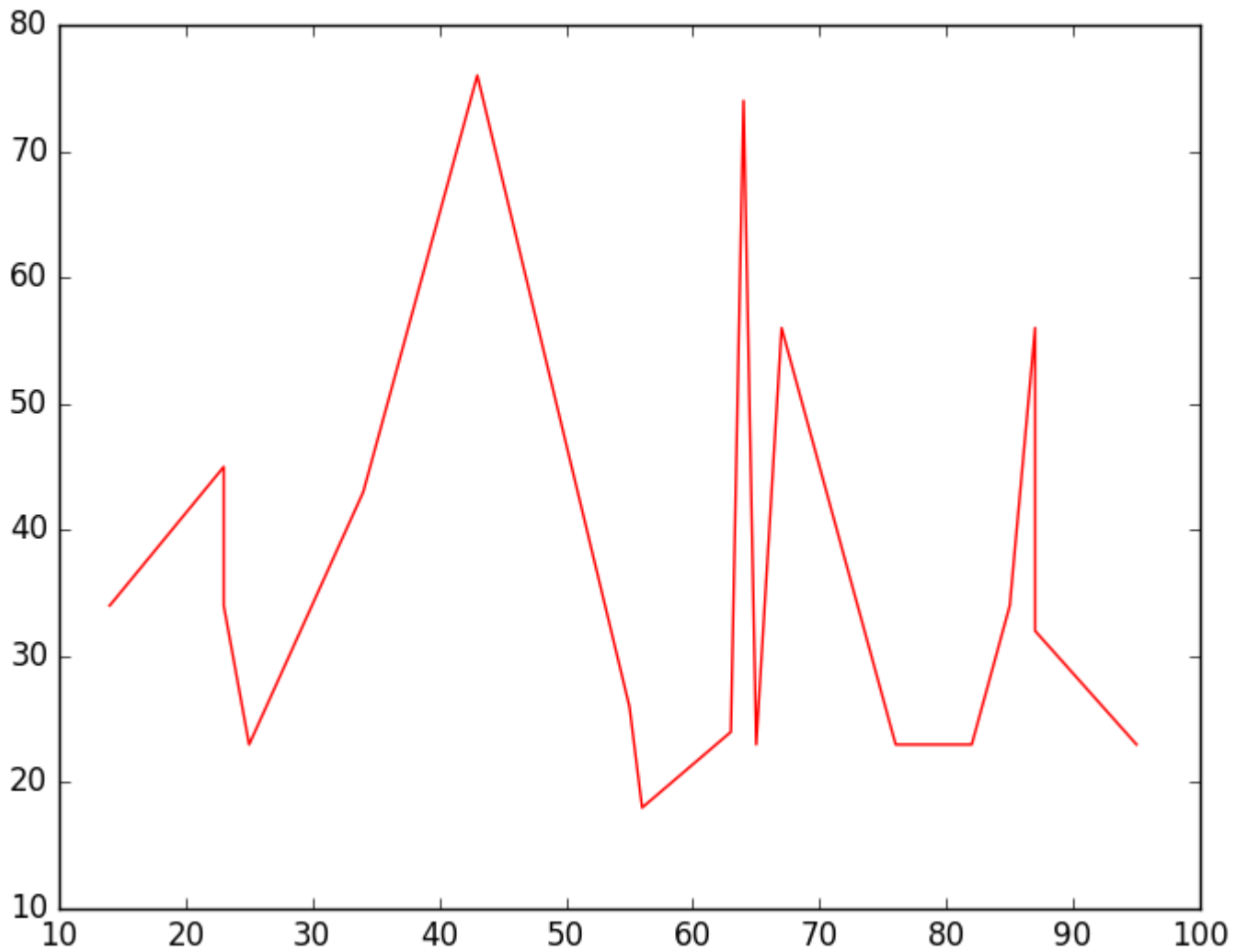
# Data
x = [0,1,2,3,4,5,6,7,8,9]
y1 = [10,20,40,55,58,55,50,40,20,10]
y2 = [20,30,50,77,82,77,75,68,65,60]

# Shade the area between y1 and y2
plt.fill_between(x, y1, y2,
                 facecolor="orange", # The fill color
                 color='blue',      # The outline color
                 alpha=0.2)         # Transparency of the fill

# Show the plot
plt.show()
```

ラインプロット

な



```
import matplotlib.pyplot as plt

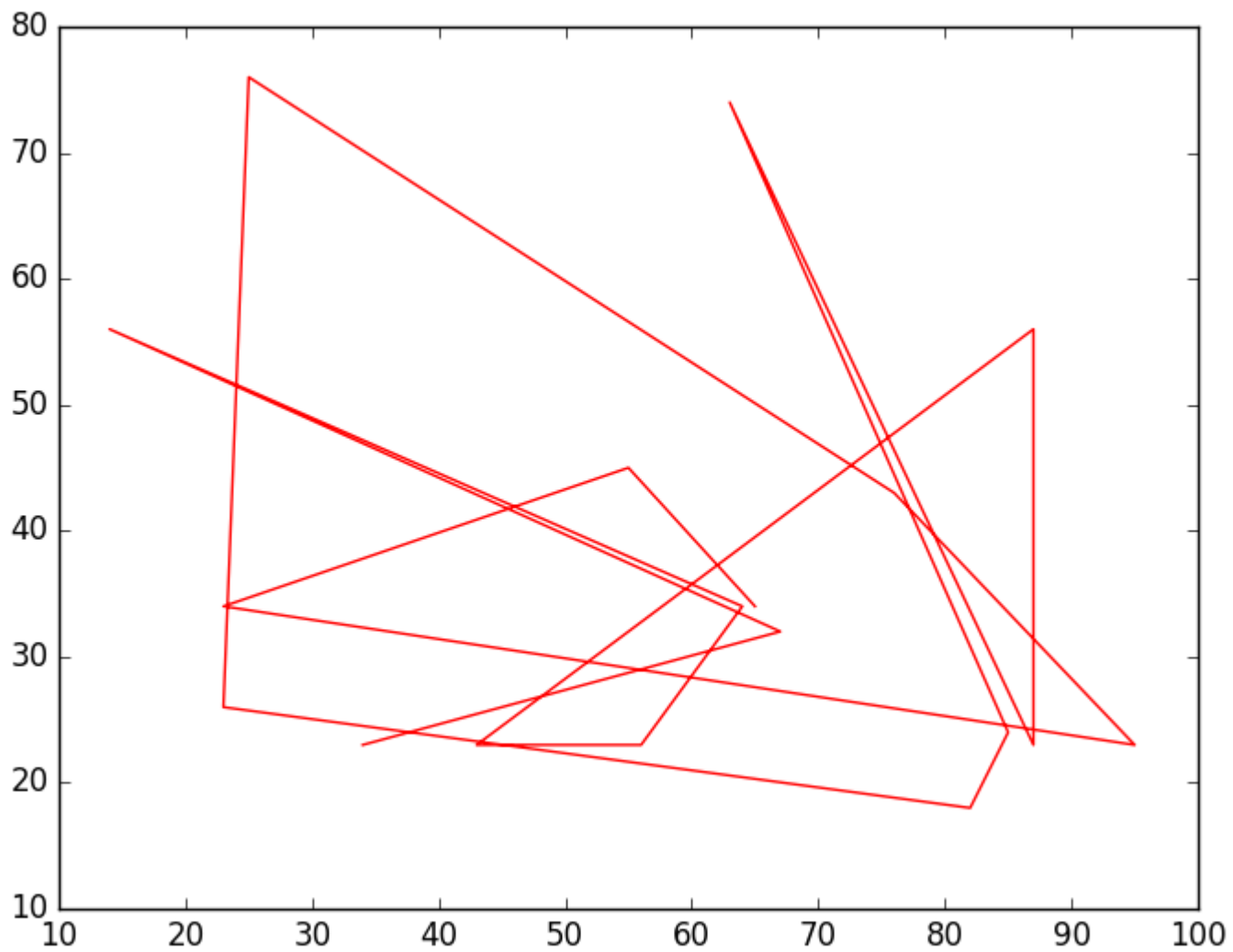
# Data
x = [14,23,23,25,34,43,55,56,63,64,65,67,76,82,85,87,87,95]
y = [34,45,34,23,43,76,26,18,24,74,23,56,23,23,34,56,32,23]

# Create the plot
plt.plot(x, y, 'r-')
# r- is a style code meaning red solid line

# Show the plot
plt.show()
```

に、ことに y のではない x のことも x ソートするはありません。べえられていない x をつラインプロットはのようになります。

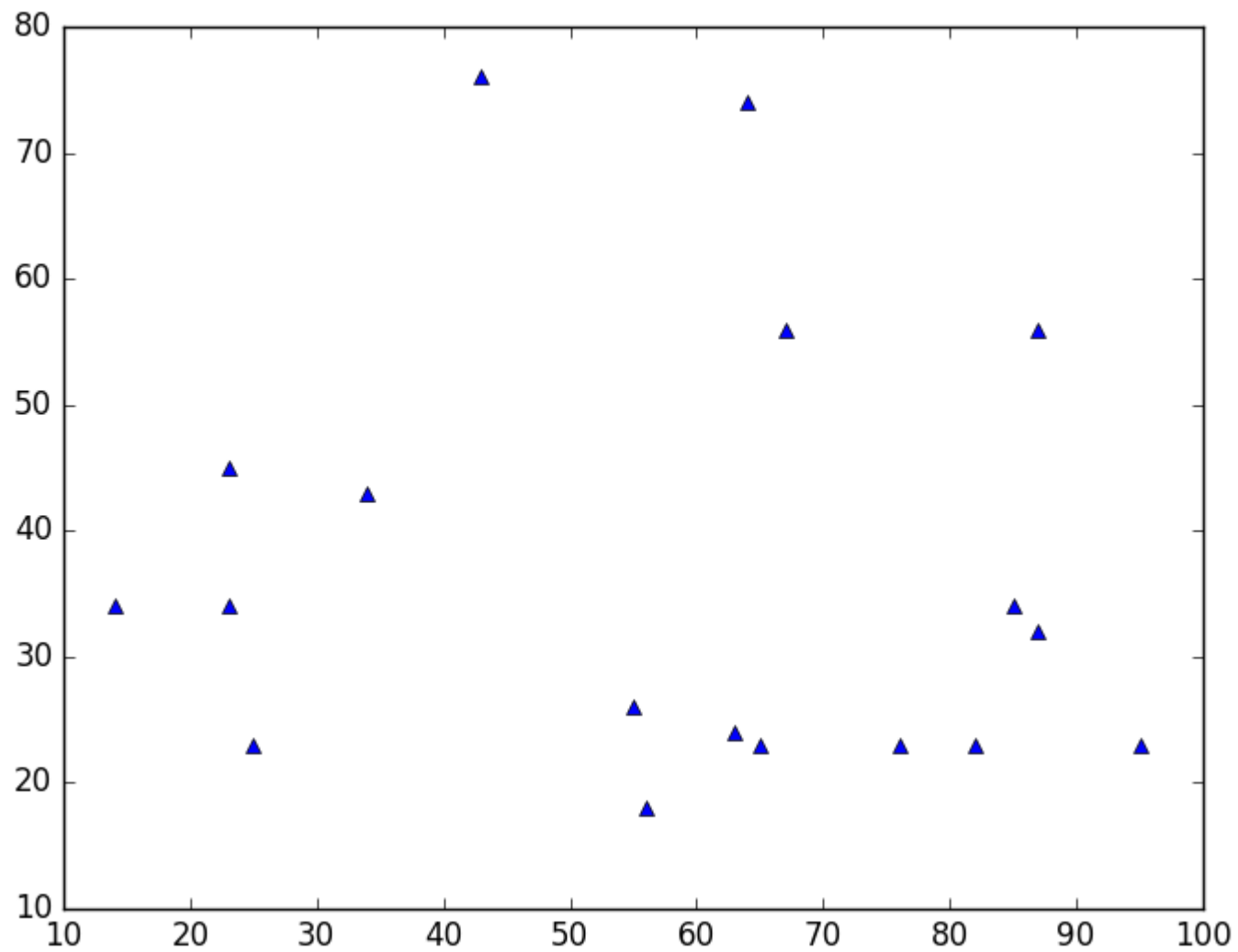
```
# shuffle the elements in x
np.random.shuffle(x)
plt.plot(x, y, 'r-')
plt.show()
```



データプロット

これはにしていますが、わりに`plot()`をします。ここでのコードののいはスタイルです。

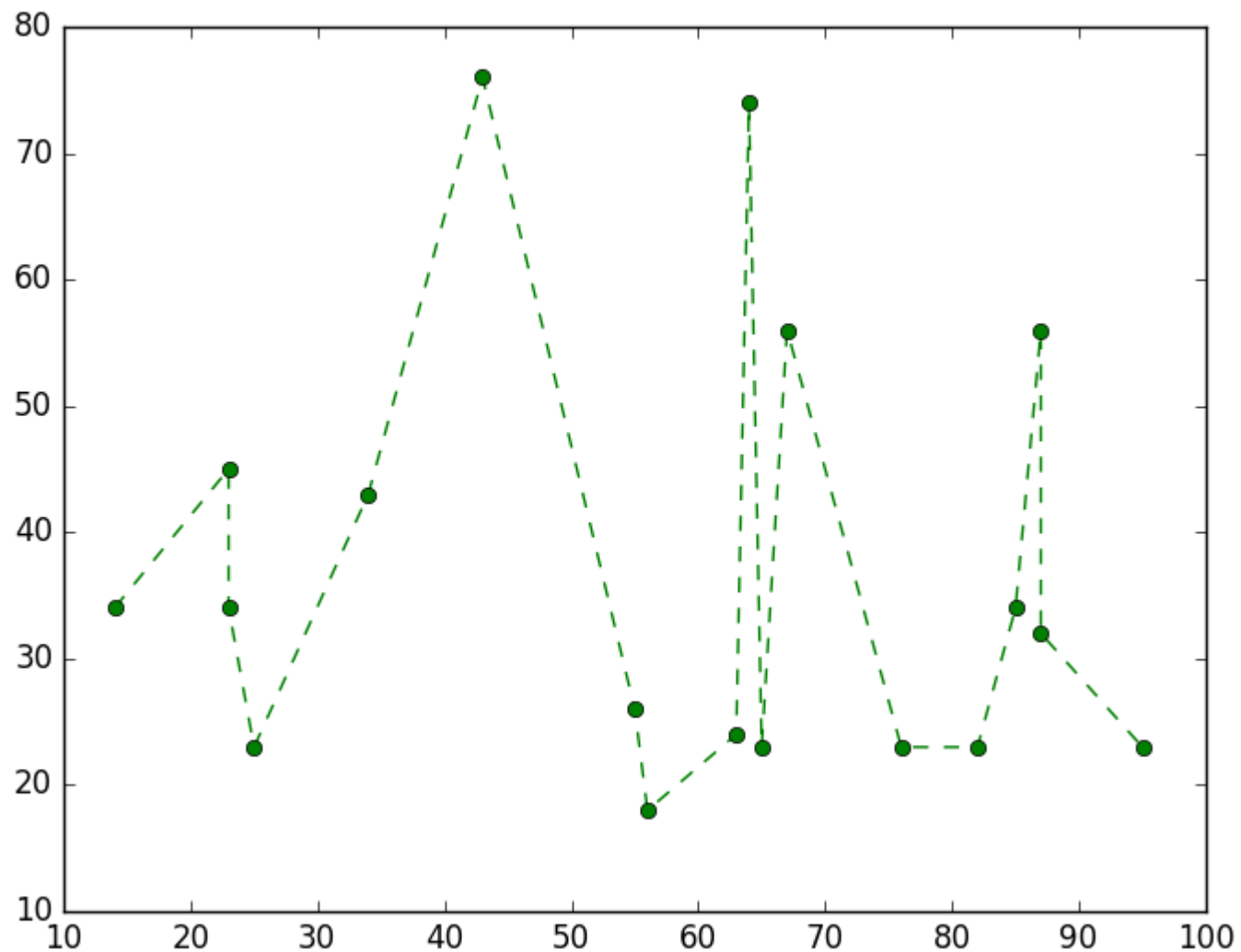
```
plt.plot(x, y, 'b^')  
# Create blue up-facing triangles
```



データとライン

`style`は、マーカーとスタイルののシンボルをることができます

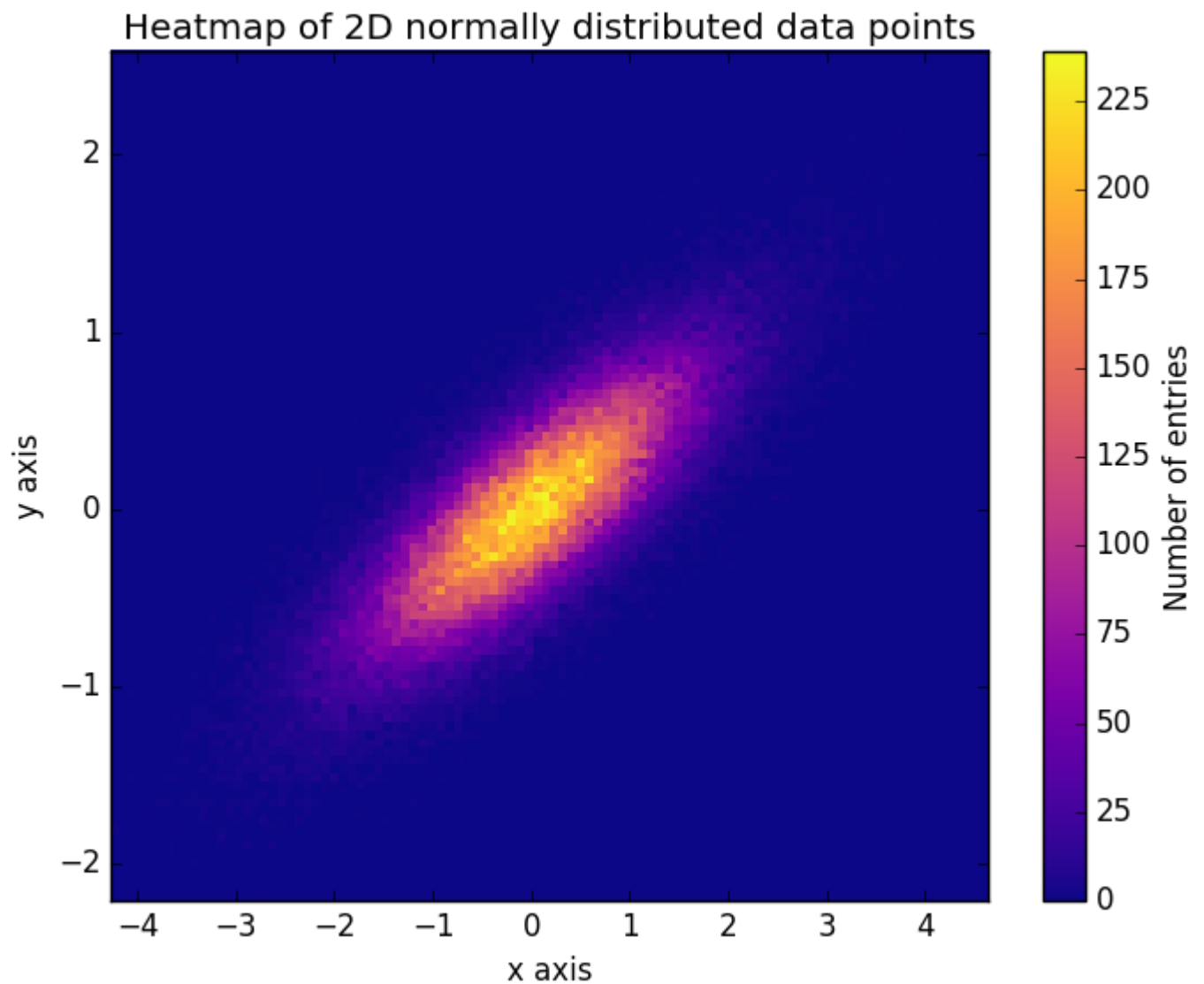
```
plt.plot(x, y, 'go--')  
# green circles and dashed line
```



ヒートマップ

ヒートマップは、2つのスカラーを表現するのに使われます。それらは、2次元ヒストグラム例えば、その値の「分布」を表現します。

ソースコードは、`plt.imshow` を使ってヒートマップを生成する `[0.0, 0.0]` の値を生成します。データは `numpy.random.multivariate_normal` を使って生成されます。 `pyplot.hist2d` の `hist2d` に使われます。



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

# Define numbers of generated data points and bins per axis.
N_numbers = 100000
N_bins = 100

# set random seed
np.random.seed(0)

# Generate 2D normally distributed numbers.
x, y = np.random.multivariate_normal(
    mean=[0.0, 0.0],      # mean
    cov=[[1.0, 0.4],
         [0.4, 0.25]],    # covariance matrix
    size=N_numbers
).T                      # transpose to get columns

# Construct 2D histogram from data using the 'plasma' colormap
plt.hist2d(x, y, bins=N_bins, normed=False, cmap='plasma')
```

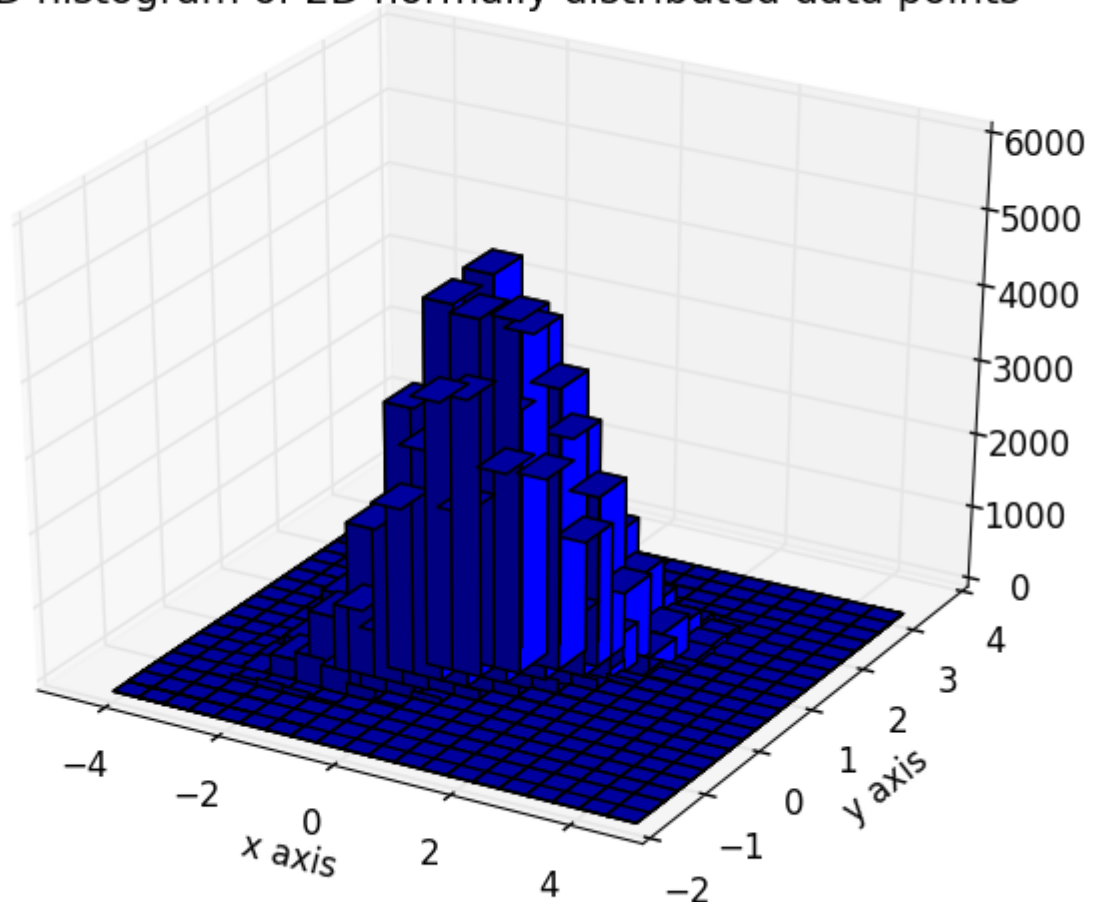
```
# Plot a colorbar with label.
cb = plt.colorbar()
cb.set_label('Number of entries')

# Add title and labels to plot.
plt.title('Heatmap of 2D normally distributed data points')
plt.xlabel('x axis')
plt.ylabel('y axis')

# Show the plot.
plt.show()
```

ここでは3Dヒストグラムとしてされたじデータがありますここではのために20ビンだけをしています。コードは[このmatplotlibデモ](#)についています。

3D histogram of 2D normally distributed data points



```
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import matplotlib
import matplotlib.pyplot as plt

# Define numbers of generated data points and bins per axis.
```

```

N_numbers = 100000
N_bins = 20

# set random seed
np.random.seed(0)

# Generate 2D normally distributed numbers.
x, y = np.random.multivariate_normal(
    mean=[0.0, 0.0],      # mean
    cov=[[1.0, 0.4],
         [0.4, 0.25]],    # covariance matrix
    size=N_numbers
).T                      # transpose to get columns

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
hist, xedges, yedges = np.histogram2d(x, y, bins=N_bins)

# Add title and labels to plot.
plt.title('3D histogram of 2D normally distributed data points')
plt.xlabel('x axis')
plt.ylabel('y axis')

# Construct arrays for the anchor positions of the bars.
# Note: np.meshgrid gives arrays in (ny, nx) so we use 'F' to flatten xpos,
# ypos in column-major order. For numpy >= 1.7, we could instead call meshgrid
# with indexing='ij'.
xpos, ypos = np.meshgrid(xedges[:-1] + 0.25, yedges[:-1] + 0.25)
xpos = xpos.flatten('F')
ypos = ypos.flatten('F')
zpos = np.zeros_like(xpos)

# Construct arrays with the dimensions for the 16 bars.
dx = 0.5 * np.ones_like(zpos)
dy = dx.copy()
dz = hist.flatten()

ax.bar3d(xpos, ypos, zpos, dx, dy, dz, color='b', zsort='average')

# Show the plot.
plt.show()

```

オンラインでプロットをむ <https://riptutorial.com/ja/matplotlib/topic/3266/プロット>

15:

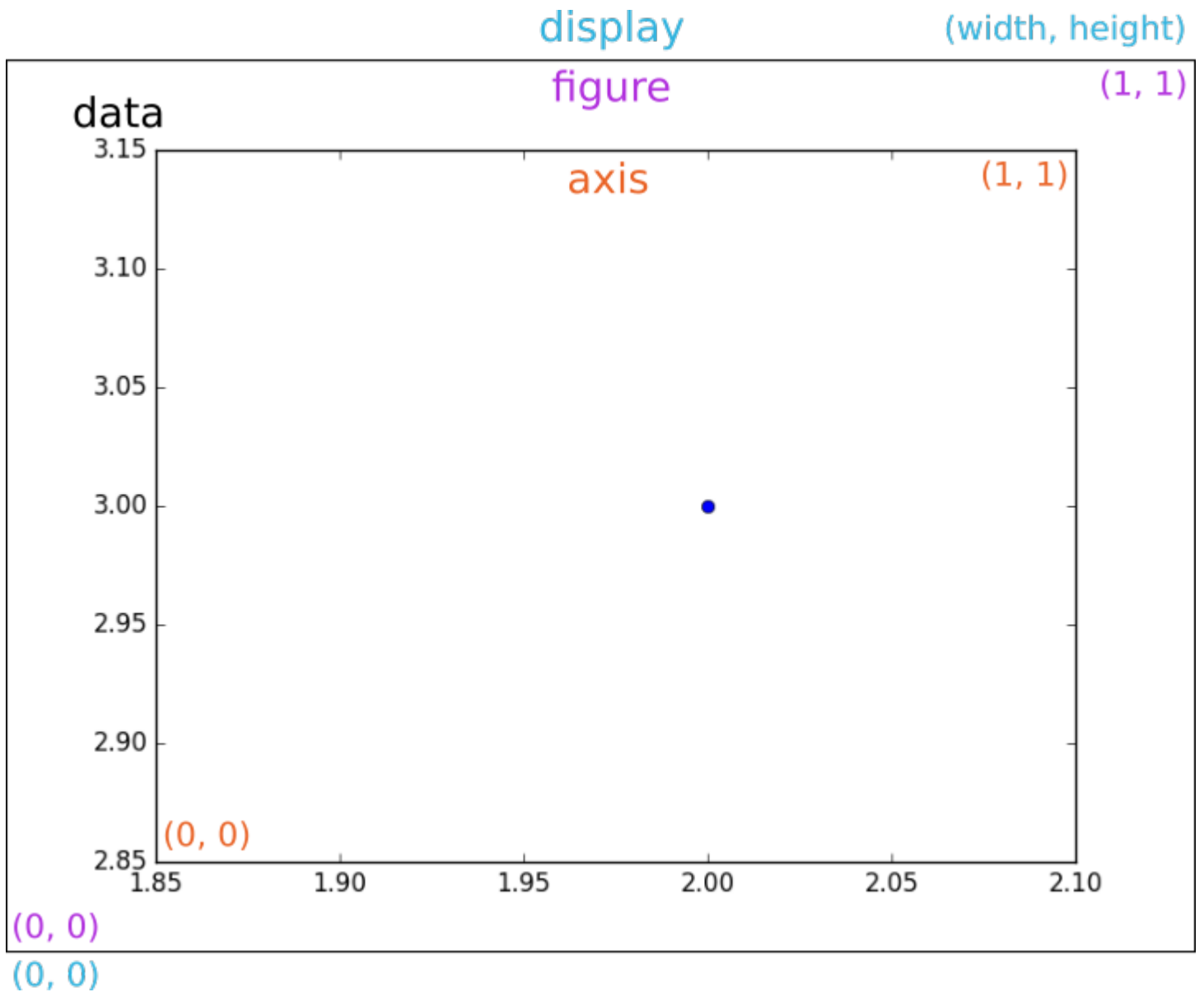
Matplotlibには4つの異なるがあり、異なるオブジェクトテキストなどのをにすることができます。システムは、異なるオブジェクトをし、そのオブジェクトは、そのシステムからのをいわゆるにする。

データは、のデータによってされるシステムです。プロットされたデータになオブジェクトをしようとするときには、はAxes `xlim`と`ylim`プロパティによってえられます。異なるオブジェクトは`ax.transData`です。

AxesはAxes オブジェクトにひけられたシステムです。0、0と1,1は、のとをします。このように、プロットのののように、をにしてするときには、異なるオブジェクトは`ax.transAxes`です。

はにひつけられているをいてにしていFigure。0、0と1,1はのとのコーナーをします。にしてかをしようとするときには、異なるオブジェクトは`fig.transFigure`です。

はピクセルでえられたのシステムです。0、0と、さは、またはディスプレイのおよびのピクセルです。これは、なめにできます。オブジェクトはをこのにするので、システムにはオブジェクトがけられていません。ただし、にじてNoneまたは`matplotlib.transforms.IdentityTransform()`をできます。



はこちらをご覧ください。

Examples

とテキスト

Matplotlibのは、したプロットにをけるときににです。のポイントにラベルをけるときなど、データをにすることがあります。のには、のにテキストをしたいかもしれません。これは、`text()`びしでオブジェクトを`transform`パラメータにすことによって、なをすることでにできます。

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

ax.plot([2.], [3.], 'bo')

plt.text( # position text relative to data
    2., 3., 'important point', # x, y, text,
    ha='center', va='bottom', # text alignment,
```

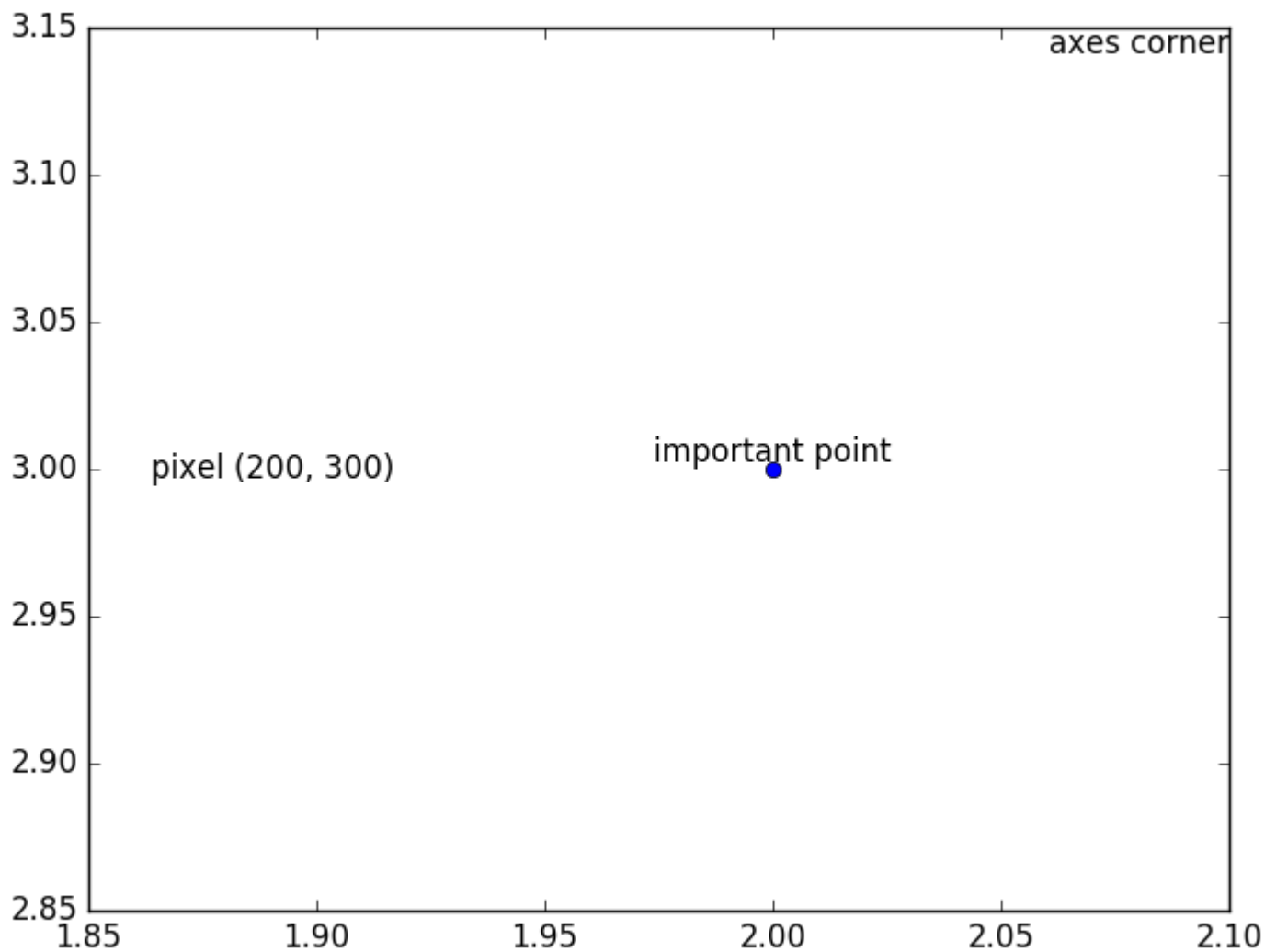
```

    transform=ax.transData      # coordinate system transformation
)
plt.text( # position text relative to Axes
    1.0, 1.0, 'axes corner',
    ha='right', va='top',
    transform=ax.transAxes
)
plt.text( # position text relative to Figure
    0.0, 1.0, 'figure corner',
    ha='left', va='top',
    transform=fig.transFigure
)
plt.text( # position text absolutely at specific pixel on image
    200, 300, 'pixel (200, 300)',
    ha='center', va='center',
    transform=None
)

plt.show()

```

figure corner



オンラインでをむ <https://riptutorial.com/ja/matplotlib/topic/4566/>

16:

Examples

をく

Matplotlibにはのための`image`モジュールがまれています

```
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
```

はファイルからみまれ、`.png`でのみ `imread`

```
img = mpimg.imread('my_image.png')
```

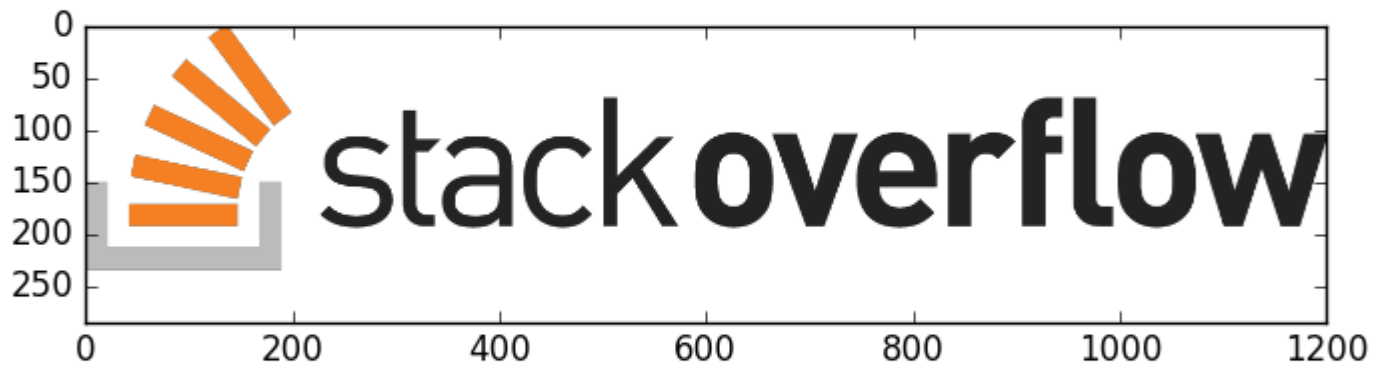
`imshow`によってレンダリングされます。

```
plt.imshow(img)
```

[Stack Overflowのロゴを](#) プロットしましょう

```
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
img = mpimg.imread('so-logo.png')
plt.imshow(img)
plt.show()
```

のプロットは



オンラインでをむ <https://riptutorial.com/ja/matplotlib/topic/4575/>

17: マップ

Examples

なりつぶし

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

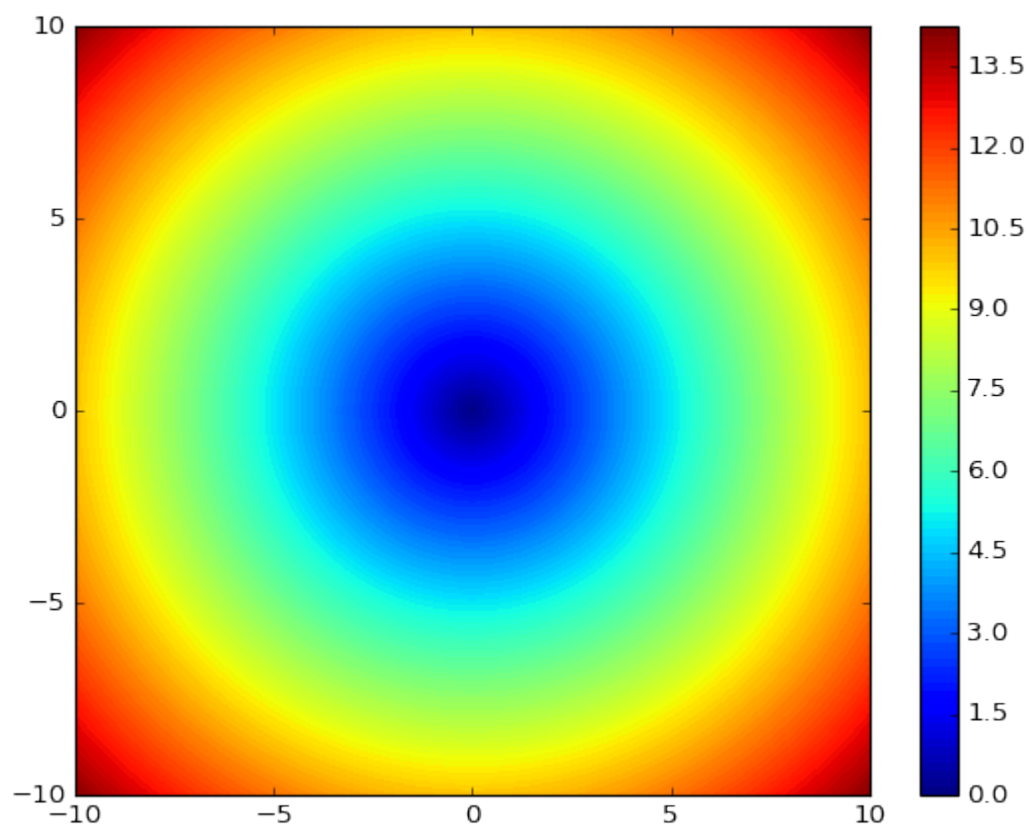
# generate 101 x and y values between -10 and 10
x = np.linspace(-10, 10, 101)
y = np.linspace(-10, 10, 101)

# make X and Y matrices representing x and y values of 2d plane
X, Y = np.meshgrid(x, y)

# compute z value of a point as a function of x and y (z = 12 distance form 0,0)
Z = np.sqrt(X ** 2 + Y ** 2)

# plot filled contour map with 100 levels
cs = plt.contourf(X, Y, Z, 100)

# add default colorbar for the map
plt.colorbar(cs)
```



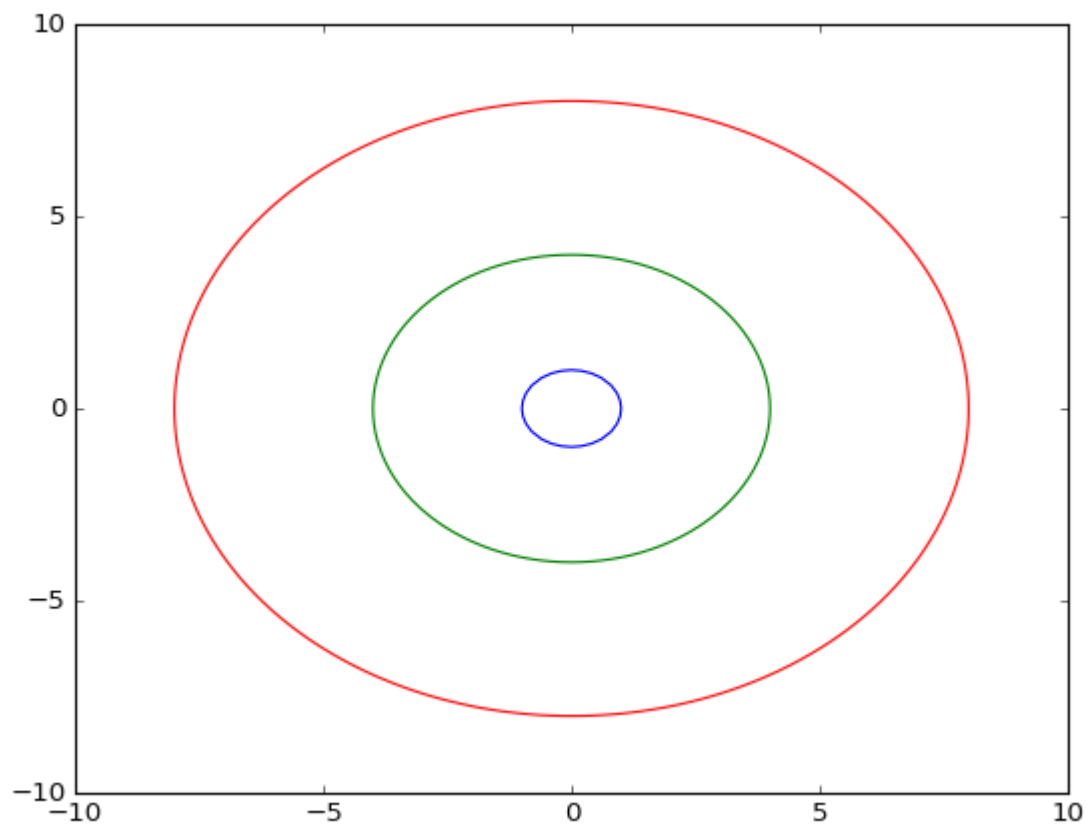
```
import matplotlib.pyplot as plt
import numpy as np

# generate 101 x and y values between -10 and 10
x = np.linspace(-10, 10, 101)
y = np.linspace(-10, 10, 101)

# make X and Y matrices representing x and y values of 2d plane
X, Y = np.meshgrid(x, y)

# compute z value of a point as a function of x and y (z = 12 distance from 0,0)
Z = np.sqrt(X ** 2 + Y ** 2)

# plot contour map with 3 levels
# colors: up to 1 - blue, from 1 to 4 - green, from 4 to 8 - red
plt.contour(X, Y, Z, [1, 4, 8], colors=['b', 'g', 'r'])
```



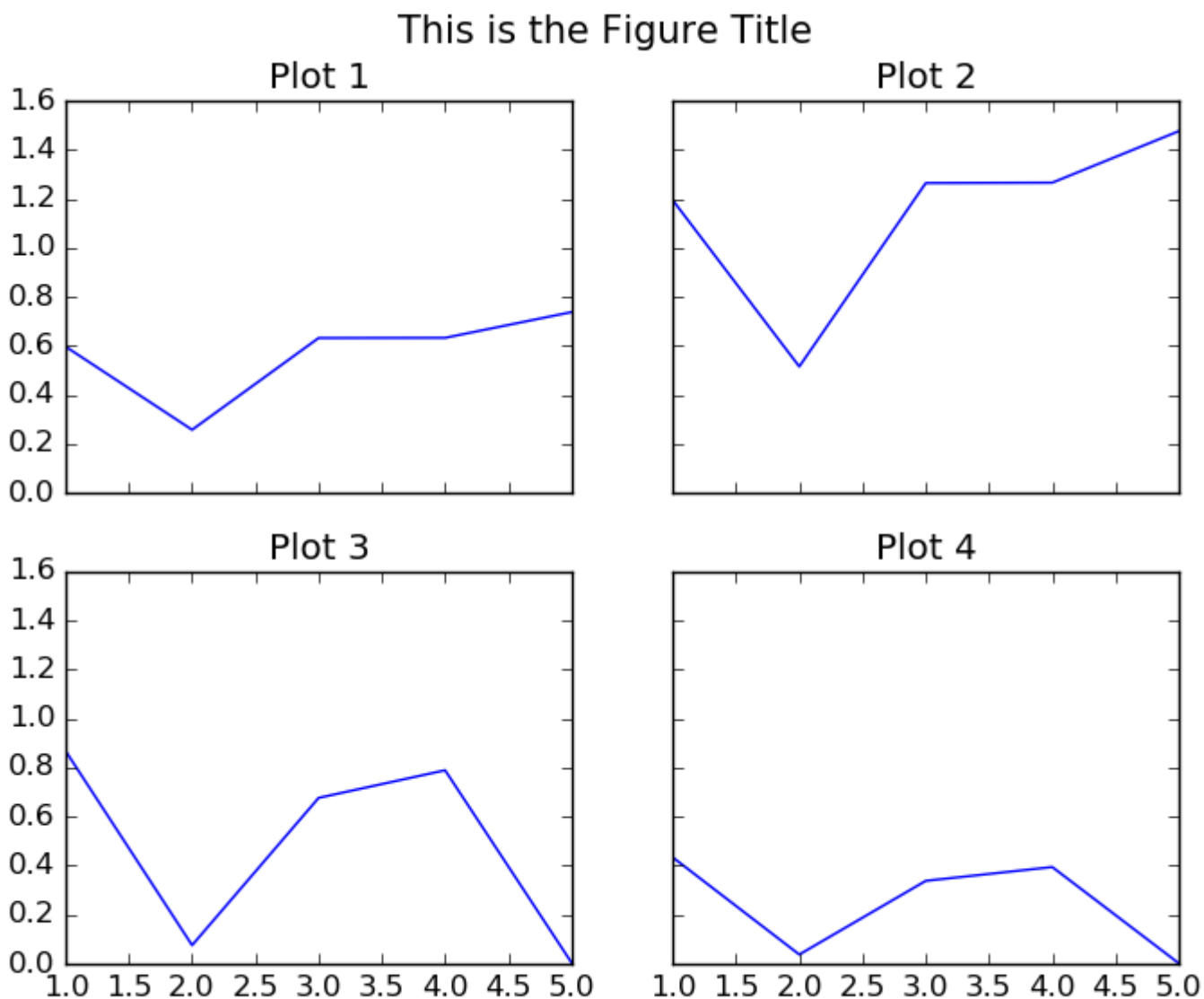
オンラインでマップをむ <https://riptutorial.com/ja/matplotlib/topic/8644/マップ>

18: のプロット

- リストアイテム

Examples

サブプロットをしたサブプロットのグリッド



```
"""
=====
CREATE A 2 BY 2 GRID OF SUB-PLOTS WITHIN THE SAME FIGURE.
=====
"""
import matplotlib.pyplot as plt

# The data
x = [1,2,3,4,5]
```

```

y1 = [0.59705847, 0.25786401, 0.63213726, 0.63287317, 0.73791151]
y2 = [1.19411694, 0.51572803, 1.26427451, 1.26574635, 1.47582302]
y3 = [0.86793828, 0.07563408, 0.67670068, 0.78932712, 0.0043694]
# 5 more random values
y4 = [0.43396914, 0.03781704, 0.33835034, 0.39466356, 0.0021847]

# Initialise the figure and a subplot axes. Each subplot sharing (showing) the
# same range of values for the x and y axis in the plots.
fig, axes = plt.subplots(2, 2, figsize=(8, 6), sharex=True, sharey=True)

# Set the title for the figure
fig.suptitle('This is the Figure Title', fontsize=15)

# Top Left Subplot
axes[0,0].plot(x, y1)
axes[0,0].set_title("Plot 1")

# Top Right Subplot
axes[0,1].plot(x, y2)
axes[0,1].set_title("Plot 2")

# Bottom Left Subplot
axes[1,0].plot(x, y3)
axes[1,0].set_title("Plot 3")

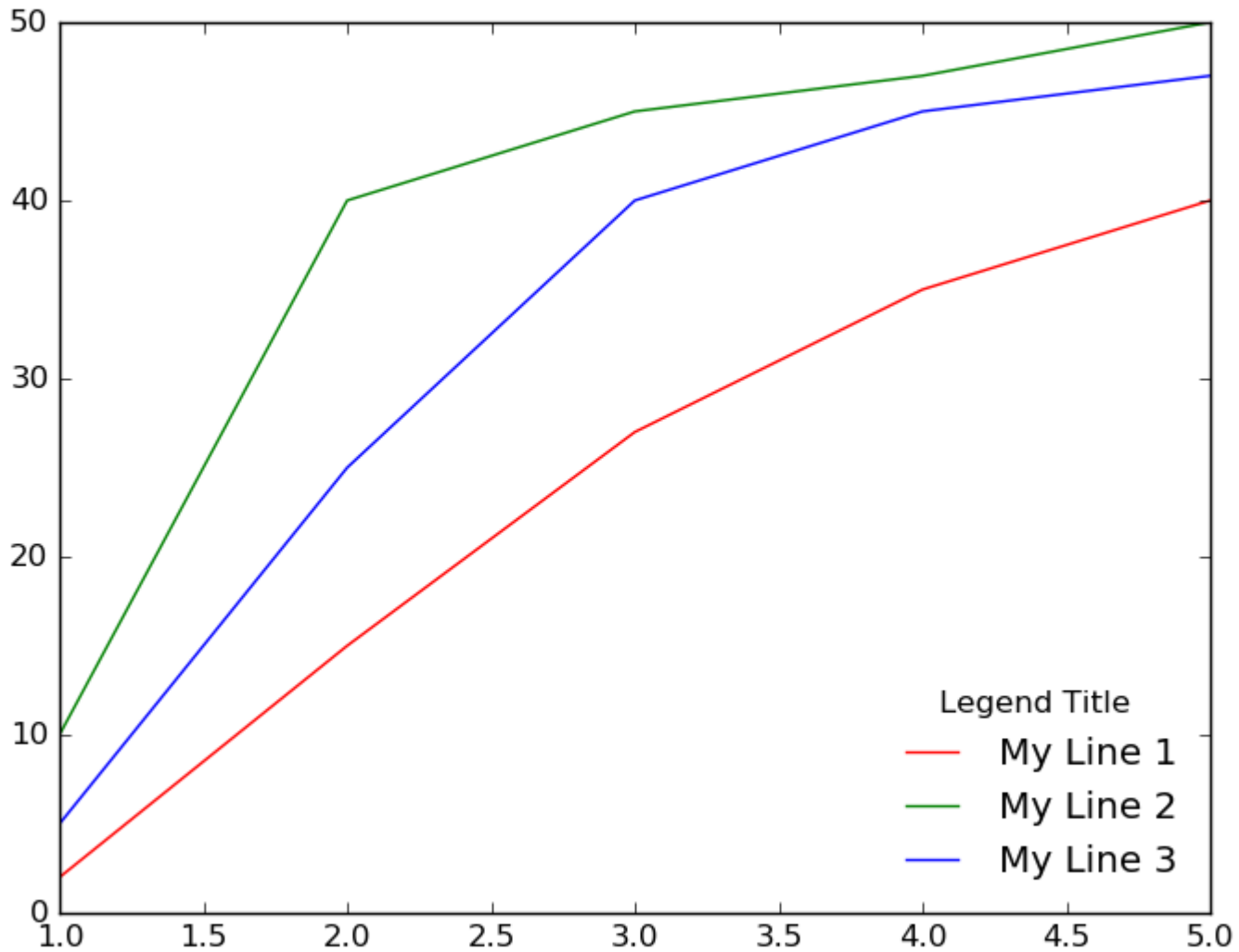
# Bottom Right Subplot
axes[1,1].plot(x, y4)
axes[1,1].set_title("Plot 4")

plt.show()

```

じプロットののI

Multiple Lines in Same Plot



```
"""
=====
                        DRAW MULTIPLE LINES IN THE SAME PLOT
=====
"""
import matplotlib.pyplot as plt

# The data
x = [1, 2, 3, 4, 5]
y1 = [2, 15, 27, 35, 40]
y2 = [10, 40, 45, 47, 50]
y3 = [5, 25, 40, 45, 47]

# Initialise the figure and axes.
fig, ax = plt.subplots(1, figsize=(8, 6))

# Set the title for the figure
fig.suptitle('Multiple Lines in Same Plot', fontsize=15)

# Draw all the lines in the same plot, assigning a label for each one to be
# shown in the legend.
ax.plot(x, y1, color="red", label="My Line 1")
ax.plot(x, y2, color="green", label="My Line 2")
```

```
ax.plot(x, y3, color="blue", label="My Line 3")

# Add a legend, and position it on the lower right (with no box)
plt.legend(loc="lower right", title="Legend Title", frameon=False)

plt.show()
```

gridspecをしたのプロット

gridspecパッケージでは、サブプロットのをよりにできます。それは、プロットのマージンと々のサブプロットのをするがずっとです。さらに、のグリッドをめるをすることにより、じのなるサイズのをすることができます。

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.gridspec import GridSpec

# Make some data
t = np.arange(0, 2, 0.01)
y1 = np.sin(2*np.pi * t)
y2 = np.cos(2*np.pi * t)
y3 = np.exp(t)
y4 = np.exp(-t)

# Initialize the grid with 3 rows and 3 columns
ncols = 3
nrows = 3
grid = GridSpec(nrows, ncols,
                left=0.1, bottom=0.15, right=0.94, top=0.94, wspace=0.3, hspace=0.3)

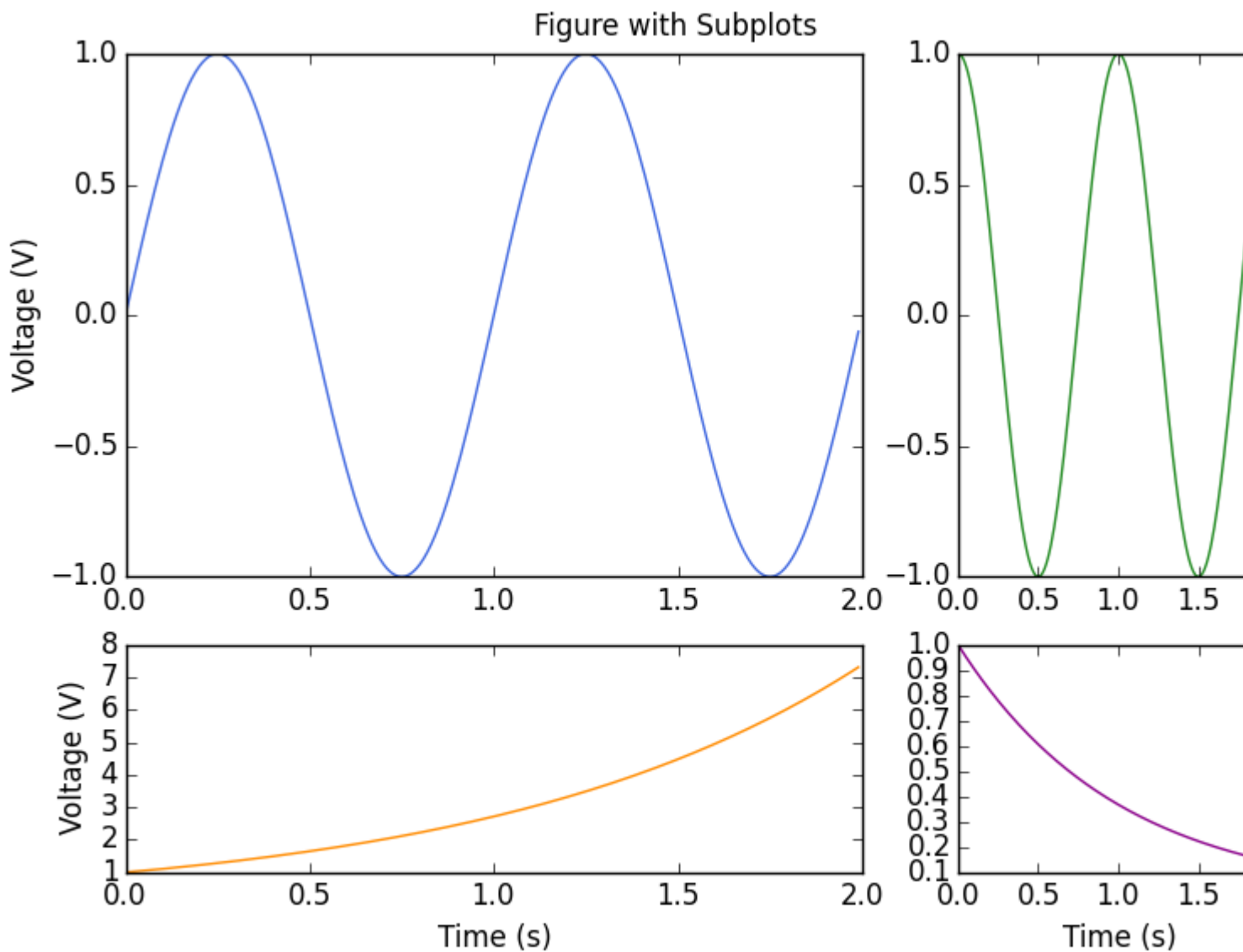
fig = plt.figure(0)
fig.clf()

# Add axes which can span multiple grid boxes
ax1 = fig.add_subplot(grid[0:2, 0:2])
ax2 = fig.add_subplot(grid[0:2, 2])
ax3 = fig.add_subplot(grid[2, 0:2])
ax4 = fig.add_subplot(grid[2, 2])

ax1.plot(t, y1, color='royalblue')
ax2.plot(t, y2, color='forestgreen')
ax3.plot(t, y3, color='darkorange')
ax4.plot(t, y4, color='darkmagenta')

# Add labels and titles
fig.suptitle('Figure with Subplots')
ax1.set_ylabel('Voltage (V)')
ax3.set_ylabel('Voltage (V)')
ax3.set_xlabel('Time (s)')
ax4.set_xlabel('Time (s)')
```

このコードは、にすプロットをします。



xの2つのプロット。

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

# create some data
x = np.arange(-2, 20, 0.5)          # values of x
y1 = map(lambda x: -4.0/3.0*x + 16, x)  # values of y1(x)
y2 = map(lambda x: 0.2*x**2 - 5*x + 32, x) # values of y2(x)

fig = plt.figure()
ax1 = fig.add_subplot(111)

# create line plot of y1(x)
line1, = ax1.plot(x, y1, 'g', label="Function y1")
ax1.set_xlabel('x')
ax1.set_ylabel('y1', color='g')

# create shared axis for y2(x)
ax2 = ax1.twinx()
```

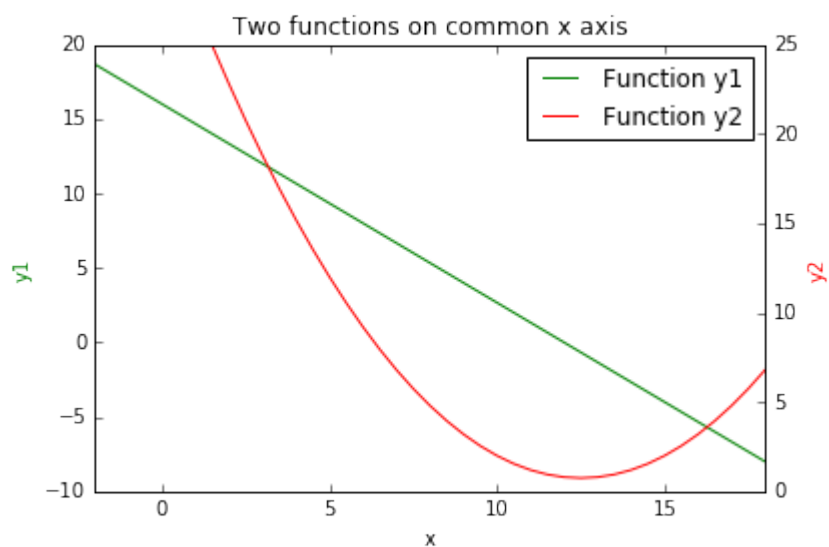
```
# create line plot of y2(x)
line2, = ax2.plot(x, y2, 'r', label="Function y2")
ax2.set_ylabel('y2', color='r')

# set title, plot limits, etc
plt.title('Two functions on common x axis')
plt.xlim(-2, 18)
plt.ylim(0, 25)

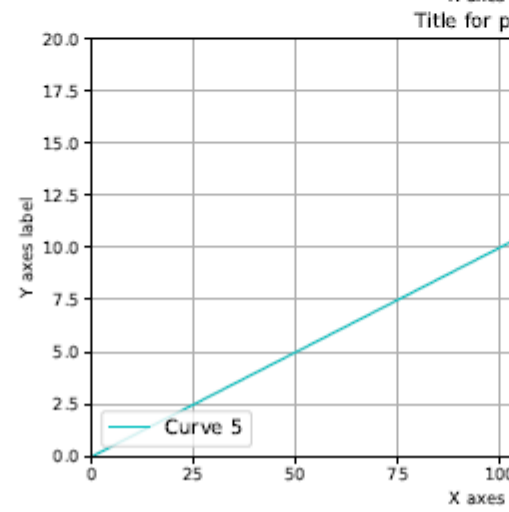
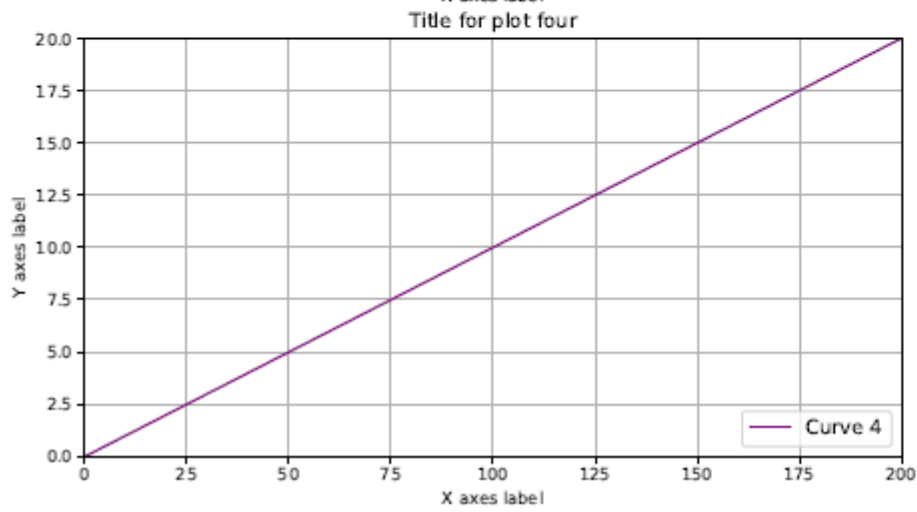
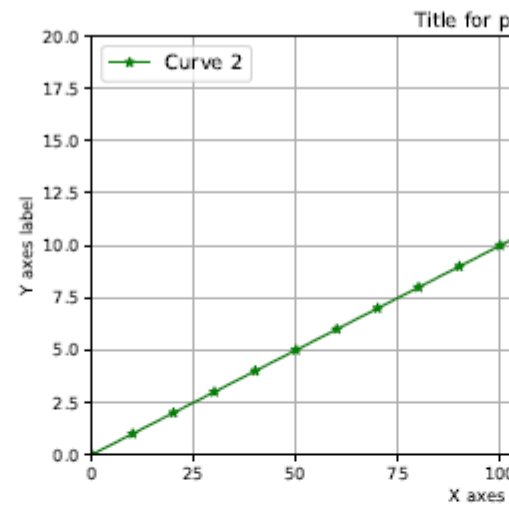
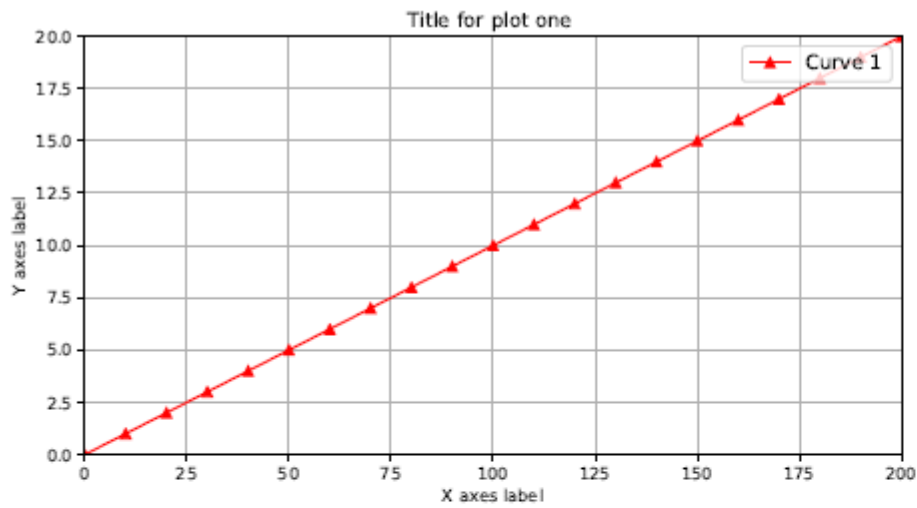
# add a legend, and position it on the upper right
plt.legend((line1, line2), ('Function y1', 'Function y2'))

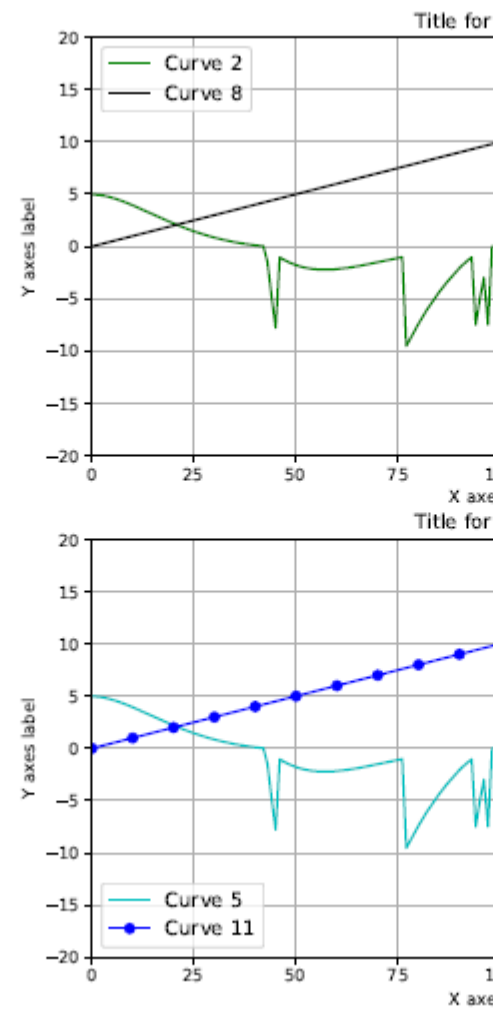
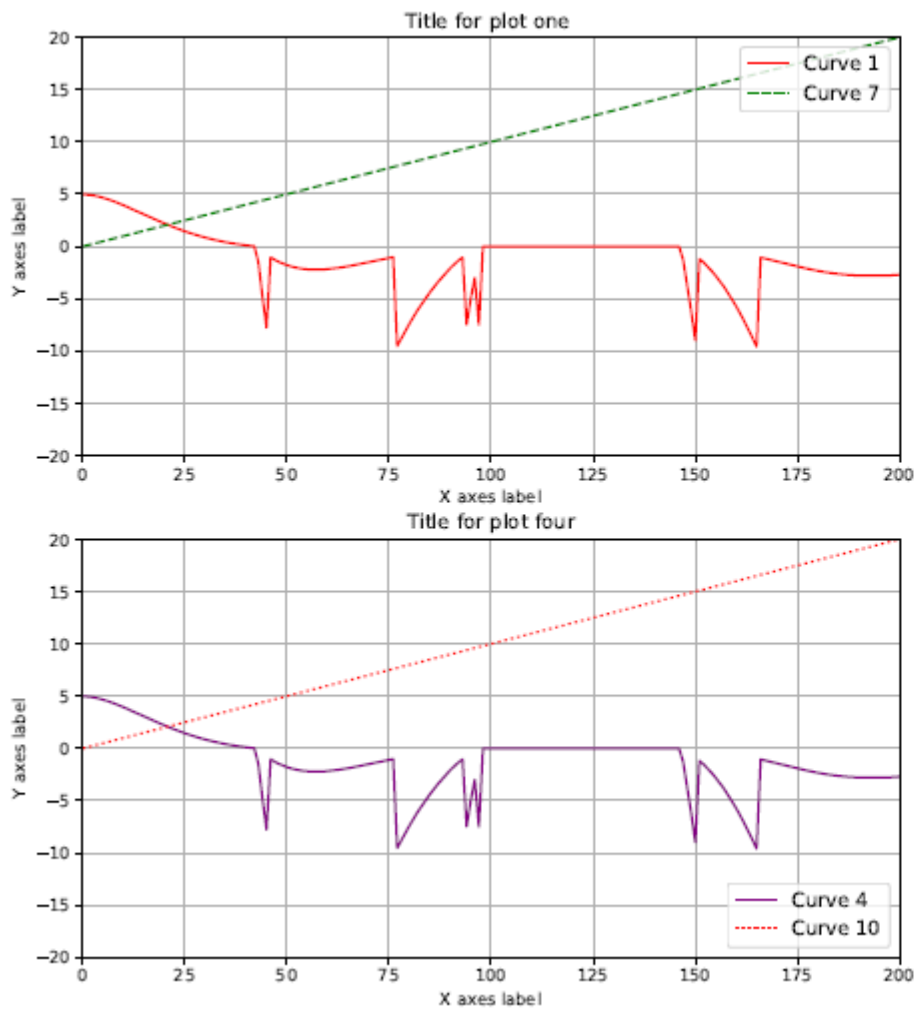
plt.show()
```

このコードは、にすプロットをします。



のプロットとのプロット





CAE.csv

```

1 TIME,Acceleration
2 0,4.992235
3 0.09952711,4.956489
4 0.1999273,4.915645
5 0.2994544,4.850395
6 0.3998545,4.763977
7 0.4993816,4.65888
8 0.5997818,4.537595
9 0.6993089,4.402862
10 0.799709,4.256423
11 0.8992361,4.100522
12 0.9996362,3.937148
13 1.099163,3.768047
14 1.199564,3.579082

```

```

import matplotlib
matplotlib.use("TKAgg")

# module to save pdf files
from matplotlib.backends.backend_pdf import PdfPages

import matplotlib.pyplot as plt # module to plot

import pandas as pd # module to read csv file

```

```

# module to allow user to select csv file
from tkinter.filedialog import askopenfilename

# module to allow user to select save directory
from tkinter.filedialog import askdirectory

#=====
#   User chosen Data for plots
#=====

# User choose csv file then read csv file
filename = askopenfilename() # user selected file
data = pd.read_csv(filename, delimiter=',')

# check to see if data is reading correctly
#print(data)

#=====
#   Plots on two different Figures and sets the size of the figures
#=====

# figure size = (width,height)
f1 = plt.figure(figsize=(30,10))
f2 = plt.figure(figsize=(30,10))

#-----
#   Figure 1 with 6 plots
#-----

# plot one
# Plot column labeled TIME from csv file and color it red
# subplot(2 Rows, 3 Columns, First subplot,)
ax1 = f1.add_subplot(2,3,1)
ax1.plot(data[["TIME"]], label = 'Curve 1', color = "r", marker = '^', markevery = 10)
# added line marker triangle

# plot two
# plot column labeled TIME from csv file and color it green
# subplot(2 Rows, 3 Columns, Second subplot)
ax2 = f1.add_subplot(2,3,2)
ax2.plot(data[["TIME"]], label = 'Curve 2', color = "g", marker = '*', markevery = 10)
# added line marker star

# plot three
# plot column labeled TIME from csv file and color it blue
# subplot(2 Rows, 3 Columns, Third subplot)
ax3 = f1.add_subplot(2,3,3)
ax3.plot(data[["TIME"]], label = 'Curve 3', color = "b", marker = 'D', markevery = 10)
# added line marker diamond

# plot four
# plot column labeled TIME from csv file and color it purple
# subplot(2 Rows, 3 Columns, Fourth subplot)
ax4 = f1.add_subplot(2,3,4)
ax4.plot(data[["TIME"]], label = 'Curve 4', color = "#800080")

```

```

# plot five
# plot column labeled TIME from csv file and color it cyan
# subplot(2 Rows, 3 Columns, Fifth subplot)
ax5 = f1.add_subplot(2,3,5)
ax5.plot(data[["TIME"]], label = 'Curve 5', color = "c")

# plot six
# plot column labeled TIME from csv file and color it black
# subplot(2 Rows, 3 Columns, Sixth subplot)
ax6 = f1.add_subplot(2,3,6)
ax6.plot(data[["TIME"]], label = 'Curve 6', color = "k")

#-----
# Figure 2 with 6 plots
#-----

# plot one
# Curve 1: plot column labeled Acceleration from csv file and color it red
# Curve 2: plot column labeled      TIME      from csv file and color it green
# subplot(2 Rows, 3 Columns, First subplot)
ax10 = f2.add_subplot(2,3,1)
ax10.plot(data[["Acceleration"]], label = 'Curve 1', color = "r")
ax10.plot(data[["TIME"]], label = 'Curve 7', color="g", linestyle = '--')
# dashed line

# plot two
# Curve 1: plot column labeled Acceleration from csv file and color it green
# Curve 2: plot column labeled      TIME      from csv file and color it black
# subplot(2 Rows, 3 Columns, Second subplot)
ax20 = f2.add_subplot(2,3,2)
ax20.plot(data[["Acceleration"]], label = 'Curve 2', color = "g")
ax20.plot(data[["TIME"]], label = 'Curve 8', color = "k", linestyle = '-')
# solid line (default)

# plot three
# Curve 1: plot column labeled Acceleration from csv file and color it blue
# Curve 2: plot column labeled      TIME      from csv file and color it purple
# subplot(2 Rows, 3 Columns, Third subplot)
ax30 = f2.add_subplot(2,3,3)
ax30.plot(data[["Acceleration"]], label = 'Curve 3', color = "b")
ax30.plot(data[["TIME"]], label = 'Curve 9', color = "#800080", linestyle = '-.')
# dash_dot line

# plot four
# Curve 1: plot column labeled Acceleration from csv file and color it purple
# Curve 2: plot column labeled      TIME      from csv file and color it red
# subplot(2 Rows, 3 Columns, Fourth subplot)
ax40 = f2.add_subplot(2,3,4)
ax40.plot(data[["Acceleration"]], label = 'Curve 4', color = "#800080")
ax40.plot(data[["TIME"]], label = 'Curve 10', color = "r", linestyle = ':')
# dotted line

# plot five
# Curve 1: plot column labeled Acceleration from csv file and color it cyan
# Curve 2: plot column labeled      TIME      from csv file and color it blue
# subplot(2 Rows, 3 Columns, Fifth subplot)

```

```

ax50 = f2.add_subplot(2,3,5)
ax50.plot(data[["Acceleration"]], label = 'Curve 5', color = "c")
ax50.plot(data[["TIME"]], label = 'Curve 11', color = "b", marker = 'o', markevery = 10)
# added line marker circle

# plot six
# Curve 1: plot column labeled Acceleration from csv file and color it black
# Curve 2: plot column labeled      TIME      from csv file and color it cyan
# subplot(2 Rows, 3 Columns, Sixth subplot)
ax60 = f2.add_subplot(2,3,6)
ax60.plot(data[["Acceleration"]], label = 'Curve 6', color = "k")
ax60.plot(data[["TIME"]], label = 'Curve 12', color = "c", marker = 's', markevery = 10)
# added line marker square

#=====
#  Figure Plot options
#=====

#-----
#  Figure 1 options
#-----

#switch to figure one for editing
plt.figure(1)

# Plot one options
ax1.legend(loc='upper right', fontsize='large')
ax1.set_title('Title for plot one ')
ax1.set_xlabel('X axes label')
ax1.set_ylabel('Y axes label')
ax1.grid(True)
ax1.set_xlim([0,200])
ax1.set_ylim([0,20])

# Plot two options
ax2.legend(loc='upper left', fontsize='large')
ax2.set_title('Title for plot two ')
ax2.set_xlabel('X axes label')
ax2.set_ylabel('Y axes label')
ax2.grid(True)
ax2.set_xlim([0,200])
ax2.set_ylim([0,20])

# Plot three options
ax3.legend(loc='upper center', fontsize='large')
ax3.set_title('Title for plot three ')
ax3.set_xlabel('X axes label')
ax3.set_ylabel('Y axes label')
ax3.grid(True)
ax3.set_xlim([0,200])
ax3.set_ylim([0,20])

# Plot four options
ax4.legend(loc='lower right', fontsize='large')
ax4.set_title('Title for plot four')
ax4.set_xlabel('X axes label')
ax4.set_ylabel('Y axes label')
ax4.grid(True)
ax4.set_xlim([0,200])

```

```

ax4.set_ylim([0,20])

# Plot five options
ax5.legend(loc='lower left', fontsize='large')
ax5.set_title('Title for plot five ')
ax5.set_xlabel('X axes label')
ax5.set_ylabel('Y axes label')
ax5.grid(True)
ax5.set_xlim([0,200])
ax5.set_ylim([0,20])

# Plot six options
ax6.legend(loc='lower center', fontsize='large')
ax6.set_title('Title for plot six')
ax6.set_xlabel('X axes label')
ax6.set_ylabel('Y axes label')
ax6.grid(True)
ax6.set_xlim([0,200])
ax6.set_ylim([0,20])

#-----
# Figure 2 options
#-----

#switch to figure two for editing
plt.figure(2)

# Plot one options
ax10.legend(loc='upper right', fontsize='large')
ax10.set_title('Title for plot one ')
ax10.set_xlabel('X axes label')
ax10.set_ylabel('Y axes label')
ax10.grid(True)
ax10.set_xlim([0,200])
ax10.set_ylim([-20,20])

# Plot two options
ax20.legend(loc='upper left', fontsize='large')
ax20.set_title('Title for plot two ')
ax20.set_xlabel('X axes label')
ax20.set_ylabel('Y axes label')
ax20.grid(True)
ax20.set_xlim([0,200])
ax20.set_ylim([-20,20])

# Plot three options
ax30.legend(loc='upper center', fontsize='large')
ax30.set_title('Title for plot three ')
ax30.set_xlabel('X axes label')
ax30.set_ylabel('Y axes label')
ax30.grid(True)
ax30.set_xlim([0,200])
ax30.set_ylim([-20,20])

# Plot four options
ax40.legend(loc='lower right', fontsize='large')
ax40.set_title('Title for plot four')
ax40.set_xlabel('X axes label')
ax40.set_ylabel('Y axes label')
ax40.grid(True)
ax40.set_xlim([0,200])

```

```

ax40.set_ylim([-20,20])

# Plot five options
ax50.legend(loc='lower left', fontsize='large')
ax50.set_title('Title for plot five ')
ax50.set_xlabel('X axes label')
ax50.set_ylabel('Y axes label')
ax50.grid(True)
ax50.set_xlim([0,200])
ax50.set_ylim([-20,20])

# Plot six options
ax60.legend(loc='lower center', fontsize='large')
ax60.set_title('Title for plot six')
ax60.set_xlabel('X axes label')
ax60.set_ylabel('Y axes label')
ax60.grid(True)
ax60.set_xlim([0,200])
ax60.set_ylim([-20,20])

#=====
#   User chosen file location Save PDF
#=====

savefilename = askdirectory()# user selected file path
pdf = PdfPages(f'{savefilename}/longplot.pdf')
# using formatted string literals ("f-strings")to place the variable into the string

# save both figures into one pdf file
pdf.savefig(1)
pdf.savefig(2)

pdf.close()

#=====
#   Show plot
#=====

# manually set the subplot spacing when there are multiple plots
#plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace =None, hspace=None )

# Automaticlly adds space between plots
plt.tight_layout()

plt.show()

```

オンラインでのプロットをむ <https://riptutorial.com/ja/matplotlib/topic/3279/のプロット>

クレジット

S. No		Contributors
1	matplotlibをいめる	Amitay Stern , ChaoticTwist , Chr , Chris Mueller , Community , dermen , evtoth , farenorth , Josh , jrjc , pmos , Serenity , tacaswell
2	3プロット	Andras Deak , Serenity , will
3	Figureウィンドウを じる	Brian , David Zwicker
4	LogLog Graphing	ml4294
5	TeX / LaTeXとの	Andras Deak , Bosoneando , Chris Mueller , Næreen , Serenity
6	アニメーションとイ ンタラクティブなブ ロット	FiN , smurfendrek123 , user2314737
7	カラーマップ	Andras Deak , Xevaquor
8	グリッドとり	ronrest
9	ヒストグラム	Yegor Kishilov
10	ボックスプロット	Luis
11		Andras Deak , Franck Dernoncourt , ronrest , saintsfan342000 , Serenity
12	とオブジェクト	David Zwicker , Josh , Serenity , tom
13	プロット	Franck Dernoncourt , Josh , ml4294 , ronrest , Scimonster , Serenity , user2314737
14		jure
15		Bosoneando
16	マップ	Eugene Loy , Serenity
17	のプロット	Chris Mueller , Robert Branam , ronrest , swatchai