複雑系プログラミング特論　matplotlib

概要

matplotlibは論文で利用可能なレベルの質を持ったグラフを作成するためのライブラリです．ここで扱う複雑系研究に限らず，グラフを用いた実験結果の視覚化は，本来複雑な現象を扱う複雑系科学のどの研究領域においてもとても重要です．今回は，このmatplotlibの基本的な使い方について紹介します．

また，ここで紹介した以外にも様々なグラフや詳細な設定が可能です．matplotlibのサイトに詳細な文書があるので，作成したい図に応じて適宜参照する事をすすめます．

参考資料 "Sandro Tosi: Matplotlib for Python Developers, Packt Publishing (2009)"

（注：公開に際し一部変更等した個所があります．）

ところで質問

みなさんグラフ作成に何を使ってますか？

- gnuplot
- matlab
- R
- excel
- mathematica
- ...

matplotlibのよいところ

- pythonを使う
- オープンソース
- 完全なプログラム言語
    - gnuplotと違って
- 豊富な他のモジュールがあるので，必要なものがほぼそろっている
- 豊富なカスタマイズ，拡張可能
    - たくさんのグラフの種類
- 豊富な出力形式
    - EPS, PDF, PNG, PS, SVG
- latexの文法が使える
- クロスプラットフォーム

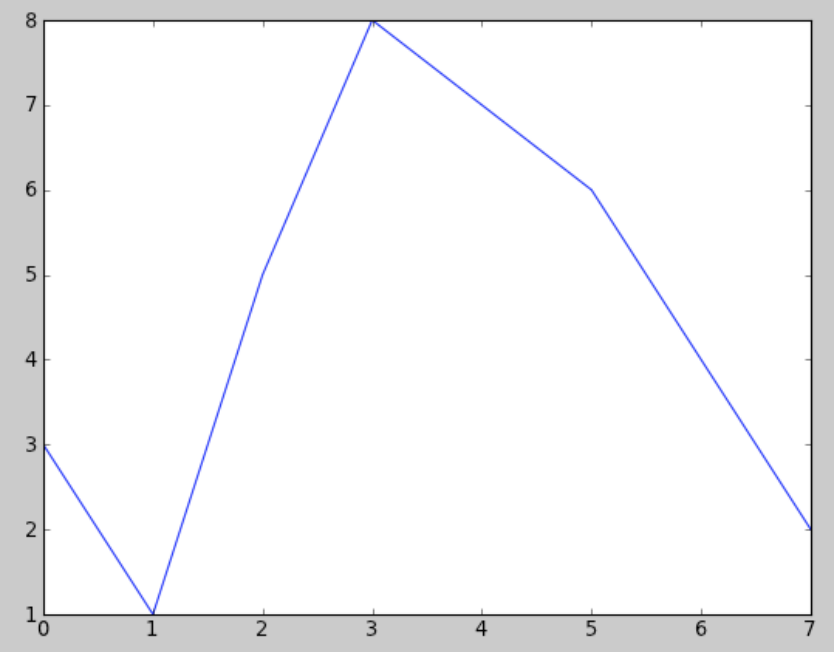
折れ線グラフ

とりあえずグラフを描いてみる.

```
1  import matplotlib.pyplot as plt
2
3  plt.plot([3, 1, 5, 8, 7, 6, 4, 2])
4  plt.show()
```

最初はmatplotlibの中のpyplotという関数群を”plt.関数名”という形で呼び出し可能にするもの．pltの部分は任意に名前を設定可能（省略す る別の書き方もある）だが，この授業では全てpltと書くことにする.

二行目はリスト[3, 1, 5, 8, 7, 6, 4, 2]をデータ点のy座標，対応するx座標を0, 1, 2, 3, 4, 5, 6, 7として折れ線グラフを作成するもの．plot関数は引数が一つの場合はそれをy座標だと思って，x座標を自動的に0から0, 1, 2, ..と割り振ってグラフを作成.
最後はこれまでの命令を元に実際にグラフのウィンドウを出して描画させる関数.



x座標を指定するには，一番目の引数をx座標のリスト，二番目をy座標のリストにする（g2）.

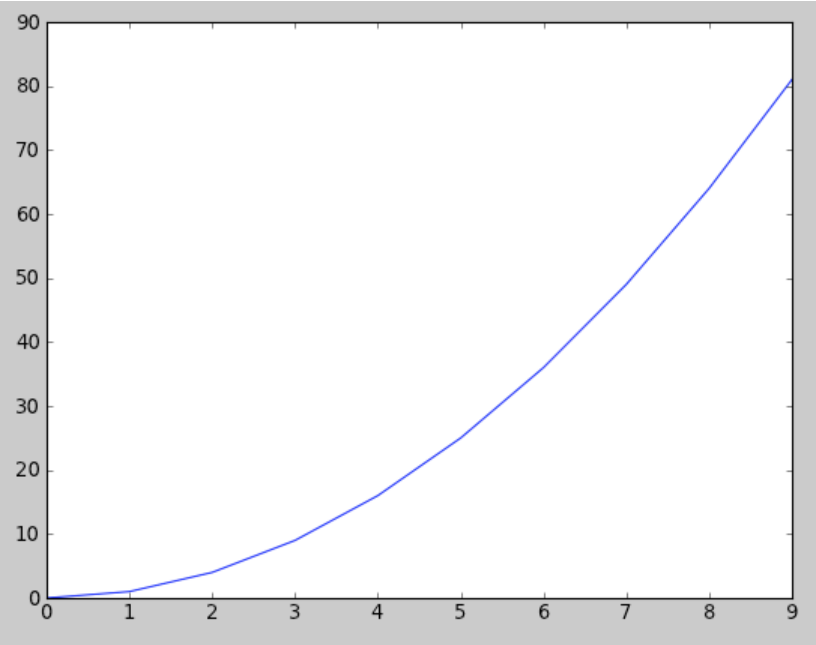
```
1  x=range(10)
2  y= [xt**2 for xt in x]
3  plt.plot(x, y)
4  plt.show()
```

二行目の[xt**2 for xt in x]はちょっと特別なリストに関するfor文の表現です.
リストxについて，その中身を順にxtに代入し，xt**2を計算した結果を順に並べたリストを返します．つまり，以下と同じ.

```
x=range(10)
y=[]
for xt in x:
    y.append(xt**2)
plt.plot(x, y)
plt.show()
```

リストの各値に同じ処理をしたリストをつくるのにとても便利な表現で，これを”リストの内包表現”といいます．

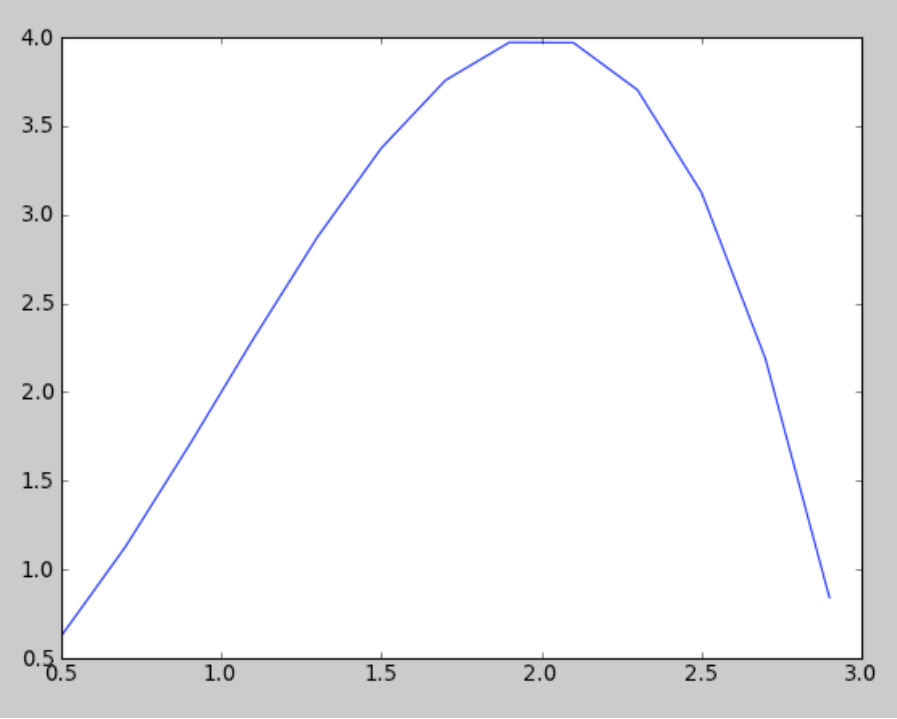
range(10)は0から10-1=9までの整数値からなるリスト，つまり，[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]を返す．より一般的には，range(x, y, z)でxからy-1までの整数値をz刻みで並べたリストを作成（xとzは省略可）．



このようにx座標をリストで表現することがよくあります．でも，pythonに標準でついているrange()関数は整数値のリストしかつくれないので，実数を指 定できる別の類似の関数arange()がnumpyという数値計算のためのライブラリに用意されています．

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

x=np.arange(0.5, 3.0, 0.2)
plt.plot(x, [3.0*xt**2-xt**3 for xt in x])
plt.show()
```

使い方はrange()関数とほぼ同じ．引数に，初期値，最大値（但しそれ自体を含まない），値の増分をそれぞれ指定．



複数の線を同一のグラフにプロットするには，対応するplot文を必要回数だけ実行し，最後にshow関数を実行する（g4）.

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3
4  x=np.arange(0.0, 2.0, 0.1)
5  plt.plot(x, [xt**2 for xt in x])
6  plt.plot(x, [xt**3 for xt in x])
7  plt.show()
```

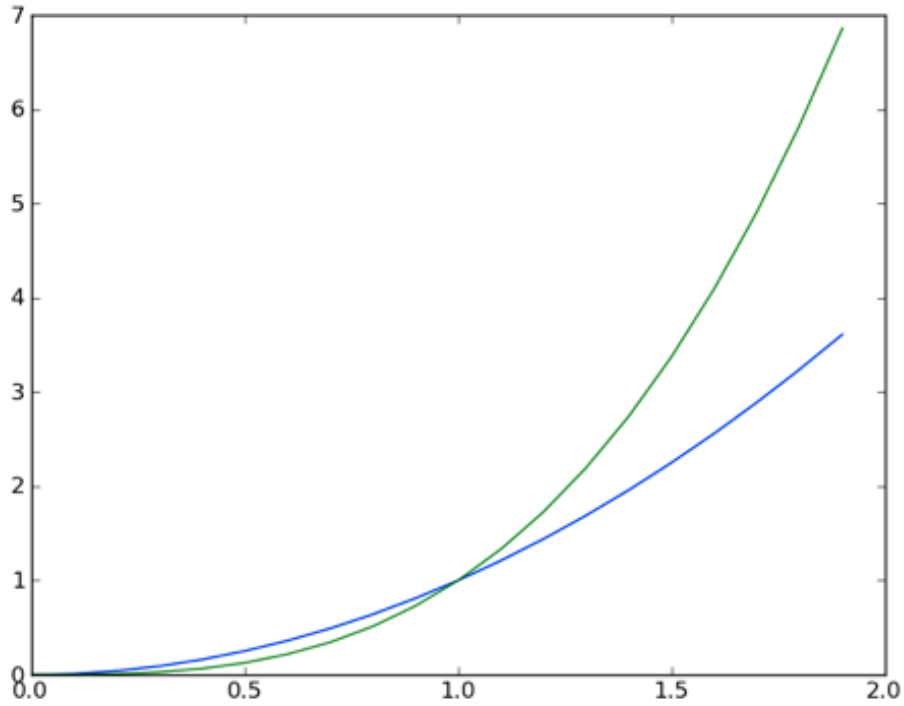
真ん中の2文はこうも書ける.

plt.plot(x, [xt**2 for xt in x], x, [xt**3 for xt in x])

もしくは,

plt.plot(x, x**2, x, x**3)

後者の書き方はarange()で作成したリストに特有の表現で，通常のrange関数でつくったリストではこのような記述はできない．実はnp.arrange で作成したものは単なるリストでなくて，numpyで用意された数値計算のための配列オブジェクト．（詳細はpython文法のページに）



練習１）上の記述法に沿って，y=ax(x-a)の値を，xを横軸にとって0以上10以下の0.1刻みでプロットしてみましょう．なお，aの値は0以上20以下の 2.5刻みで設定し，それぞれの設定で折れ線を1つずつかくこと．

グラフの詳細設定

グリッド線，軸の範囲・名前，タイトル，凡例

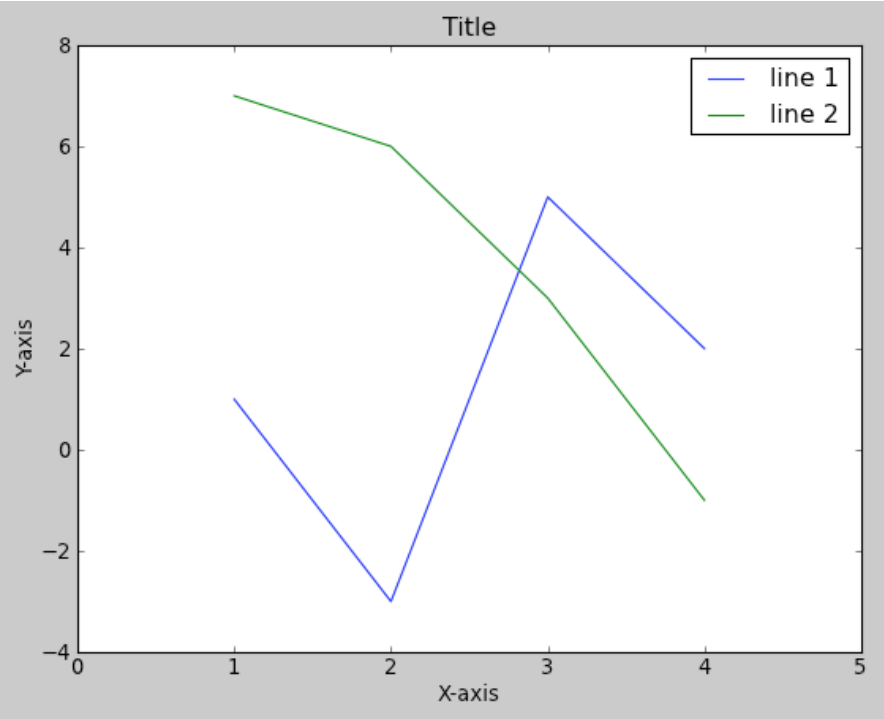
軸の範囲は自動で設定されるが，自分で設定したい場合もある．そのほか，グラフに対して細かな設定を以下のように加えることができる（g5）．

- plt.grid(True)
  グリッド線を足す．Trueはpythonで真偽値を表す．
- plt.axis([xmin, xmax, ymin, ymax])
  x軸の範囲をxminからxmax，y軸の範囲をyminからymaxに設定．
  plt.axis(ymin=値, xmin=値)などでもＯＫ．
- plt.xlabel("文字列")
  plt.ylabel("文字列")
  x軸y軸のラベルをそれぞれを文字列の通りに設定．
- plt.title("文字列")
  タイトルを文字列の通りに設定．
- plt.plot(x, y, label="文字列")
  x，yで指定した折れ線グラフを文字列の通りに名前を付けて描く．付けた名前は凡例で使われる．
- plt.legend()
  凡例を表示する．
- plt.legend(["名前１", "名前２"])

のようにあとからまとめて指定もできる．

- plt.legend(loc="upper left")
  のように位置も設定できる．upper, lower, left, rightがつかえる

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

plt.plot([1, 2, 3, 4], [1, -3, 5, 2], label="line 1")
plt.plot([1, 2, 3, 4], [7, 6, 3, -1], label="line 2")
plt.axis([0, 5, -4, 8])
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Title")
plt.legend(loc="upper right")
plt.show()
```



色，マーカー，線種

各線について，個別に色，マーカー，線種を指定できる．plot()関数は，（x座標，）y座標のデータに続いてもう一つの引数をとることができ，ここで各種 フォーマットを指定する．plt.plot(x, y, "フォーマット")

- 色
  plt.plot(x, y, 'y')
  色をy:yellowに設定．
  b:blue, c: cyan, g: green, k: black, m: magenta, r: red, w: white, y: yellow
  "#FF00FF"などのHTML形式や，"(1, 0, 1)"などのRGB形式にも対応．"'(1, 0, 1, 1)"などのRGBAも．
  グレースケールの場合は，"0.6".
- 線種

plt.plot(x, y, '--')
破線に設定.
"-": 実線, "--": 破線, "-.": 一点破線, ":": 点線.

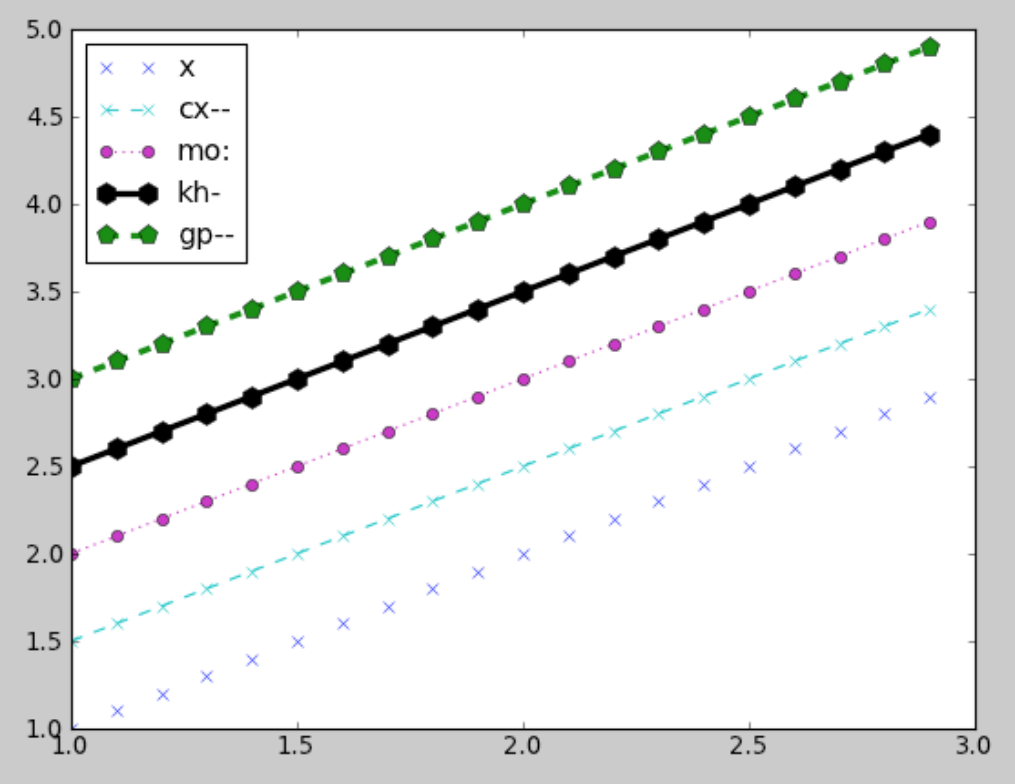
- マーカー
  plt.plot(x, y, '.')
  点に設定.
  ".": 点, ",": ピクセル, "o": 円, "v": 下向き三角, "^"上向き三角, "<": 左向き三角, ">": 右向き三角, "1": 下トライポッド, "2": 上トライポッド,
  "3": 左トライポッド, "4": 右トライポッド, "s": 四角, "p": 五角形, "h": 六角形, "H": 回転六角形, "+": 十字, "x": クロス, "D": 菱形, "d": 薄い菱形,
  "l": 垂直線, "-": 水平線.
- 複数線を同時に設定する場合は,
  plt.plot(x1, y1, format1, x2, y2, format2, ...)

これらを"フォーマット"でまとめて指定する.

さらに詳細なオプションも設定可能. plot()関数の引数の最後尾に以下の引数を付ける. これらのオプションはplot()関数で設定した全ての線に適用され る. （g6）

- color / c
- linestyle
- linewidth
- marker
- markerdgecolor
- markeredgewidth
- markerfacecolor
- markersize

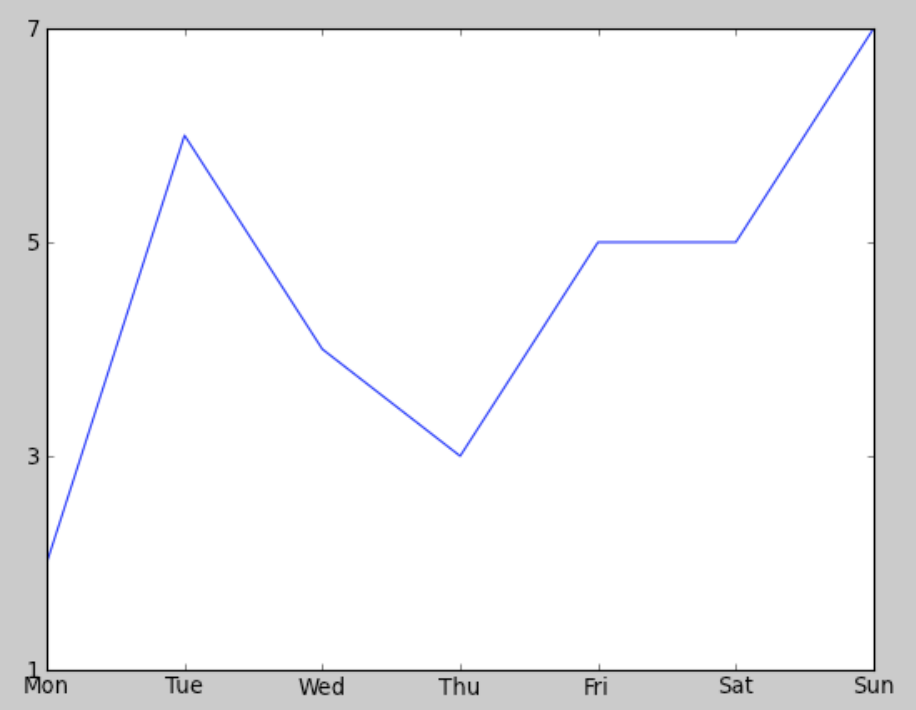
```
import matplotlib.pyplot as plt
import numpy as np

x=np.arange(1, 3, 0.1)
plt.plot(x, x, 'x', x, x+0.5, 'cx--', x, x+1, 'mo:')
plt.plot(x, x+1.5, 'kh-', x, x+2.0, 'gp--', linewidth=3, markersize=10)
plt.legend(['x', 'cx--', 'mo:', 'kh-', 'gp--'], loc="upper left")
plt.show()
```



軸の目盛

軸の目盛はplt.xtics(目盛を付ける値のリスト)，もしくは，plt.xtics(目盛を付ける値のリスト, 対応する名前のリスト)で設定．例えば，（g7）

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

x=range(7)
y=[2, 6, 4, 3, 5, 5, 7]
plt.plot(y)
plt.xticks(x, ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'])
plt.yticks(range(1, 8, 2))
plt.show()
```



練習２）

時刻0で，鉛直上向き（+y方向とする）にボールを初速度v_0[m/s]で投げあげたとき，時刻t[s]でのボールの速度v[m/s]とy座標[m]の位置 は

$$v = v_0 - gt, \qquad y = v_0 t - \frac{1}{2} g t^2$$

(ただし重力加速度g=9.8[m/s^2])

と表す事ができるとします．

このとき，時刻tを横軸に，その時刻でのボールの位置yと速度をy軸にとったグラフを，リストの内包表現を使って描きましょう．ただし，tは0以上3以下の 0.1刻みとし，v_0は10の場合と20の場合の二通りを考え，全ての折れ線を1つのグラフに描くとします．適宜軸の名前やタイトル，凡例と折れ線の種類を設定し，見やすいグラフをつくってみて下さい．

様々なグラフ

種類

matplotlibでは，例えば次の様な様々なグラフが描けます．

- エラーバー付折れ線グラフ
- 棒グラフ
- 円グラフ
- 濃淡図
- ヒストグラム

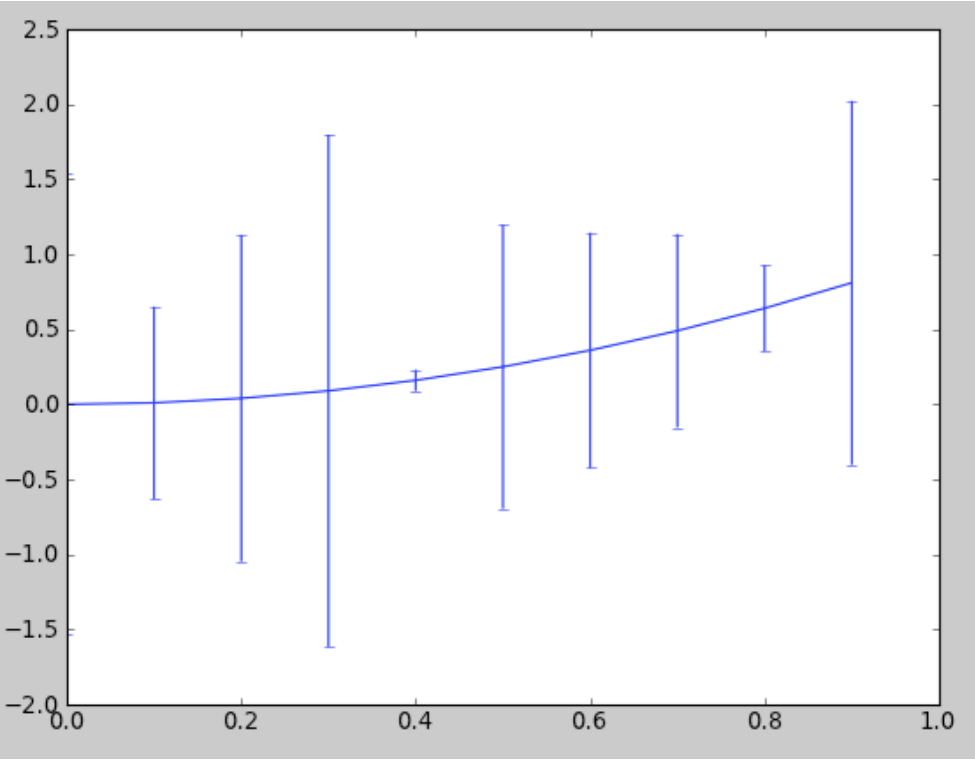
- 散布図

エラーバー付折れ線グラフ

各値の誤差を表すエラーバーつきの折れ線グラフはplt.errorbar関数を使います.

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

x=np.arange(0, 1, 0.1)
y=x**2
e=np.abs(np.random.randn(len(y)))
plt.errorbar(x, y, yerr=e)
plt.show()
```

オプションにはecolor elinewidth, capsizeが指定可能. yerr=[e1, e2]とすると上向き（+y）と下向き（-y）で別の値を誤差に設定可能. x軸に関してもxerrで同様に.



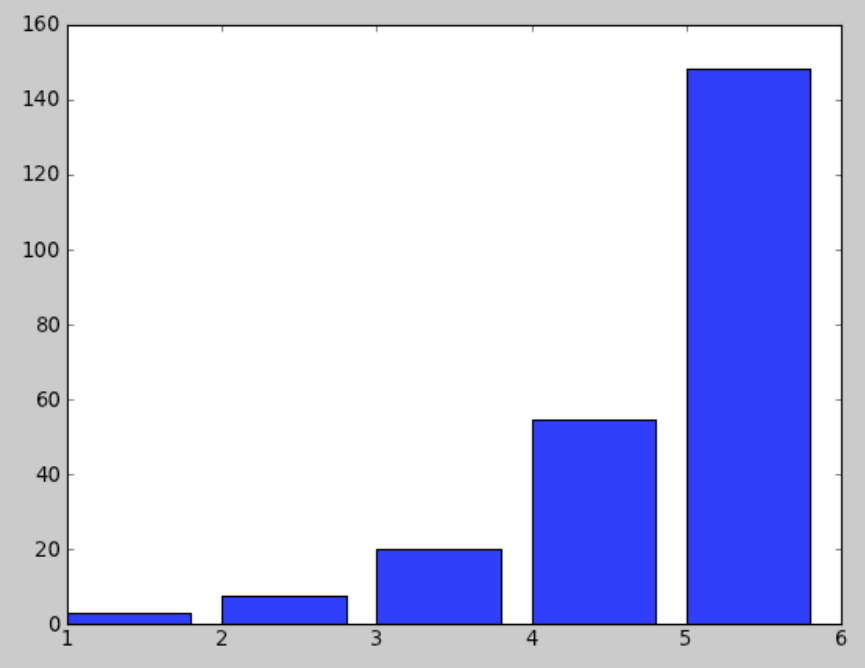
棒グラフ

棒グラフはplt.bar関数で場所と高さを指定します.

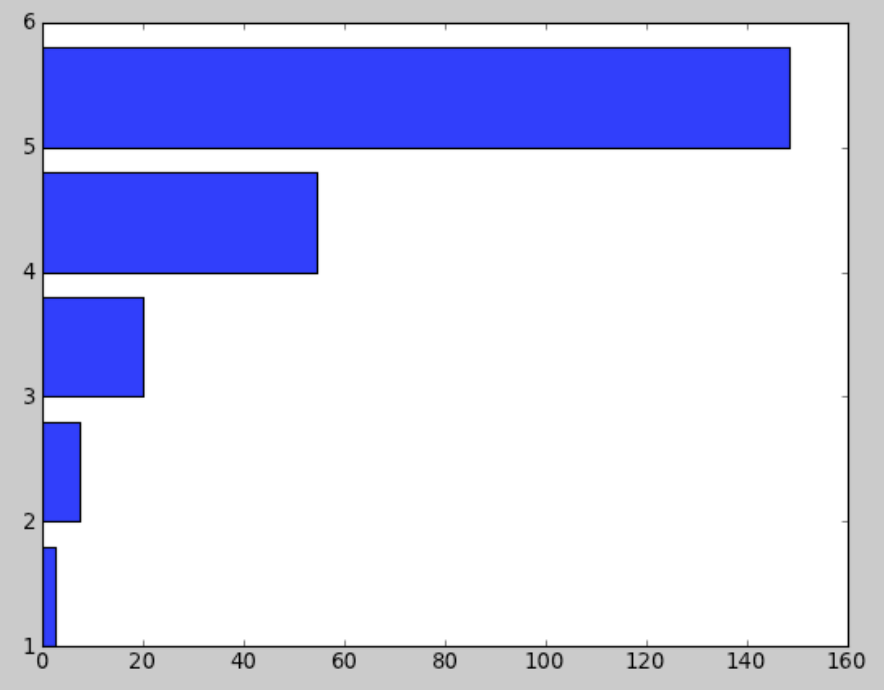
```
import matplotlib.pyplot as plt
import numpy as np

x=np.arange(1, 6)
y=np.exp(x)

plt.barh(x, y)
```

```
8 plt.show()
```

最初の引数のリストは各バーのx座標での左端の位置，二つ目は各バーの高さ．つまり，各バーの左上端の座標を指定する．



np.barをnp.barhにすると横向きのグラフになる



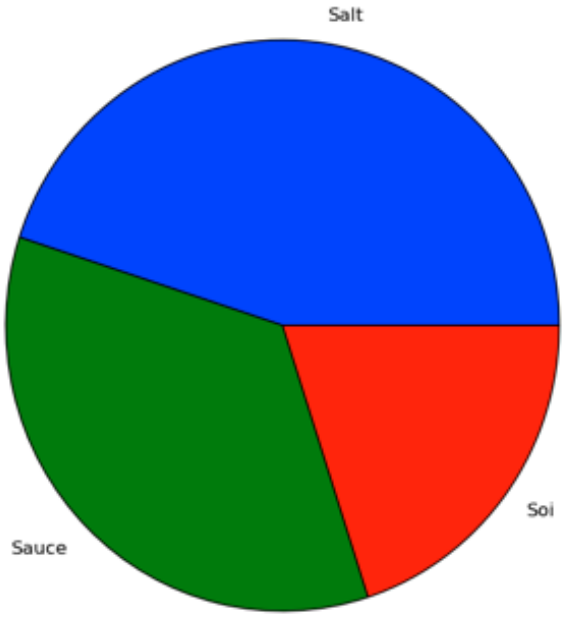
円グラフ

円グラフには各値にラベルを付けることができます．

```
1 x=[45, 35, 20]
2 labels= ['Salt', 'Sauce', 'Soi']
```

```
plt.figure(figsize=(8, 8))
plt.pie(x, labels=labels)
plt.show()
```

3行目はグラフの縦横の比を1対1にするためのもの．こうしないとつぶれた感じになったりする．

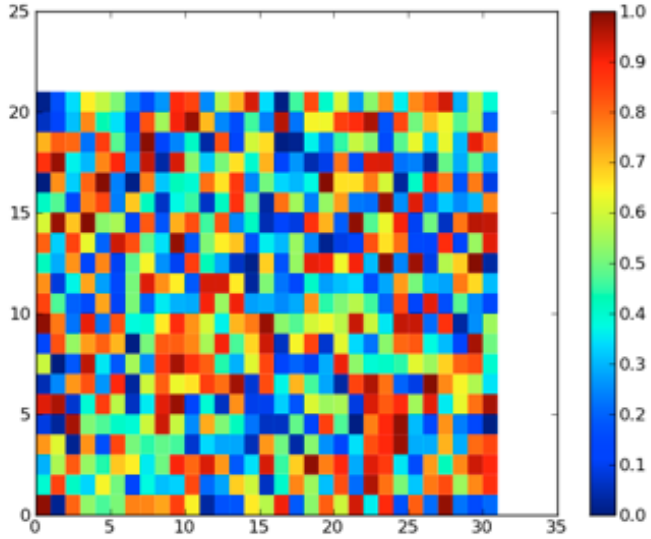


濃淡図

1次元もしくは2次元のnp.array配列に入った値の分布を，1次元，2次元空間上の濃淡（色の違い）で表す．（h8b）

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

matr= np.random.rand(21, 31)
plt.pcolor(matr , vmin=0, vmax=1)
plt.colorbar()
plt.show()
```

np.random.rand(x, y)は[0, 1)の範囲の乱数が入ったx*yの2次元np.array配列を返す関数．
6行目のpcolor関数で表示したい値の入った配列の指定，値の最小（vmin）・最大値（vmax）をそれぞれ指定している．ここにさらに引数とし て，”cmap=plt.cm.gray”や”cmap=plt.cm.binary”等を指定し，色のグラデーションパターンを設定できる．7行目で右 側のカラーバーの表示を設定．

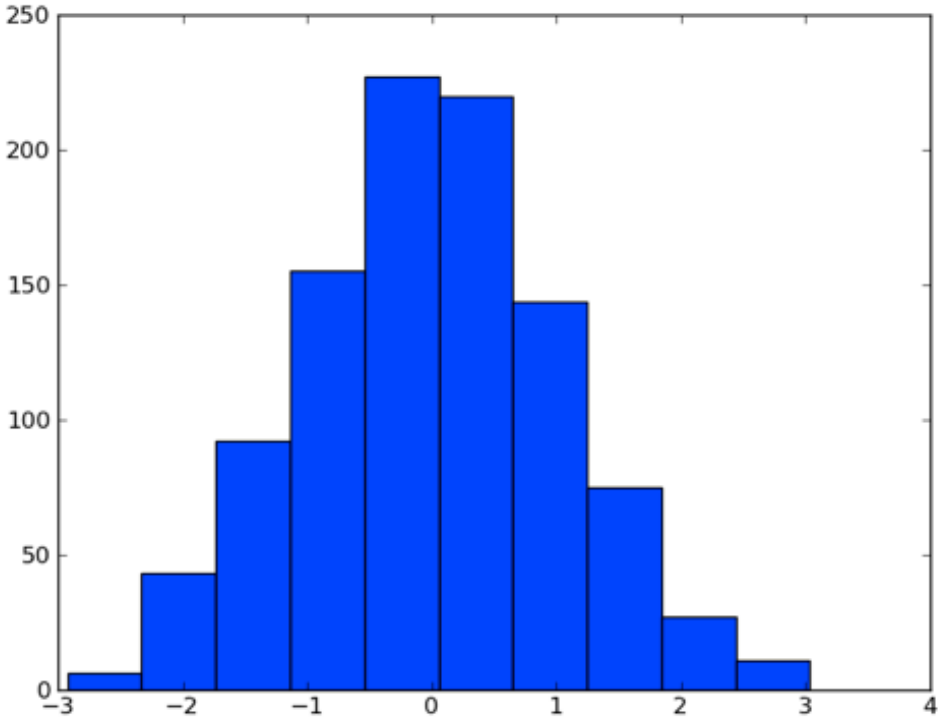


ヒストグラム

np.arrayの配列に入ったデータからヒストグラムが自動でつくれます．例えば，

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

y=np.random.randn(1000)
plt.hist(y, 10)
plt.show()
```

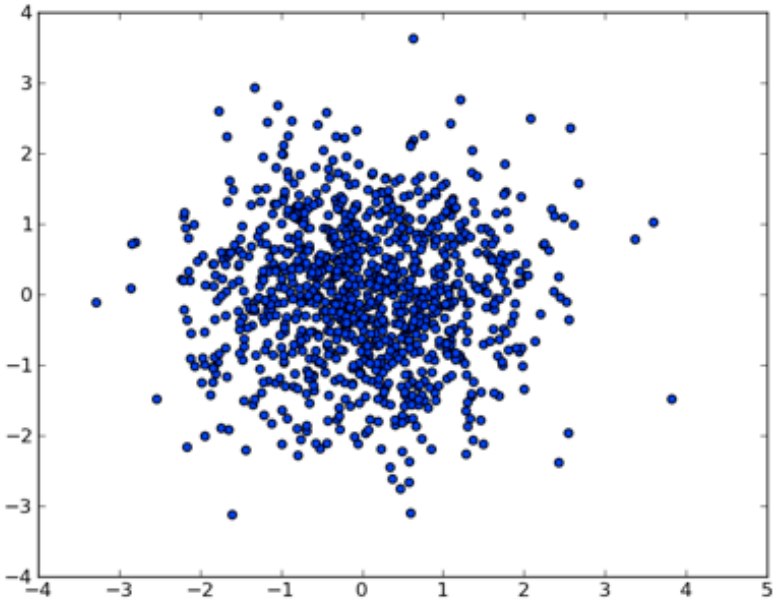
plt.hist(y, n)は配列yの中の要素でヒストグラムをつくる関数．このときデータをn個の階級に分ける（省略可）．
np.random.randn(x)は，標準正規分布（平均0，分散1）からランダムに生成したx個の実数値からなるnp.array配列を返す関数で す．hist関数は，配列の中身から自動的に分布を計算してグラフをつくります．



散布図

データの分布を図示する散布図.

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3  x= np.random.randn(1000)
4  y= np.random.randn(1000)
5  plt.scatter(x, y)
6  plt.show()
```



グラフ中へのテキストの挿入

plt.text(x, y, text)

(x, y)の位置に文字列textを描画.
0が左端，1が右端

texの数式形式でテキストを書く

「$」で囲んでテキストを書くと，texの数式形式で記述できる．ラベルでも，追加のテキストでも．例えば，

plt.text(0.5，0.5, r"$\frac{1}{\alpha}$")

ただし，上のように，クォーテーションの前に「r」をつけて，「\」がpythonに解析（パース）されないようにすること.

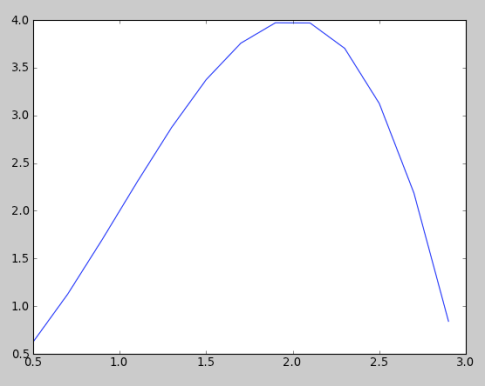
オブジェクト指向を反映したグラフの扱い方

これまでとは違ってオブジェクト指向的にグラフを扱うと，もうすこし込み入ったグラフが描ける場合がある.

グラフ，グラフの構成要素を変数に割り当てる

これまでの方法で書いた以下のソースを考える.

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3
4  x=np.arange(0.5, 3.0, 0.2)
5  plt.plot(x, [3.0*xt**2-xt**3 for xt in x])
6  plt.show()
```



以下の記述は同じ意味.

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


fig= plt.figure()
ax= fig.add_subplot(1, 1, 1)

x=np.arange(0.5, 3.0, 0.2)
ax.plot(x, [3.0*xt**2-xt**3 for xt in x])
plt.show()
```

3行目でグラフ，というよりも，グラフを書くための一枚のキャンバスに相当するオブジェクトを生成し，figに割り当てる.

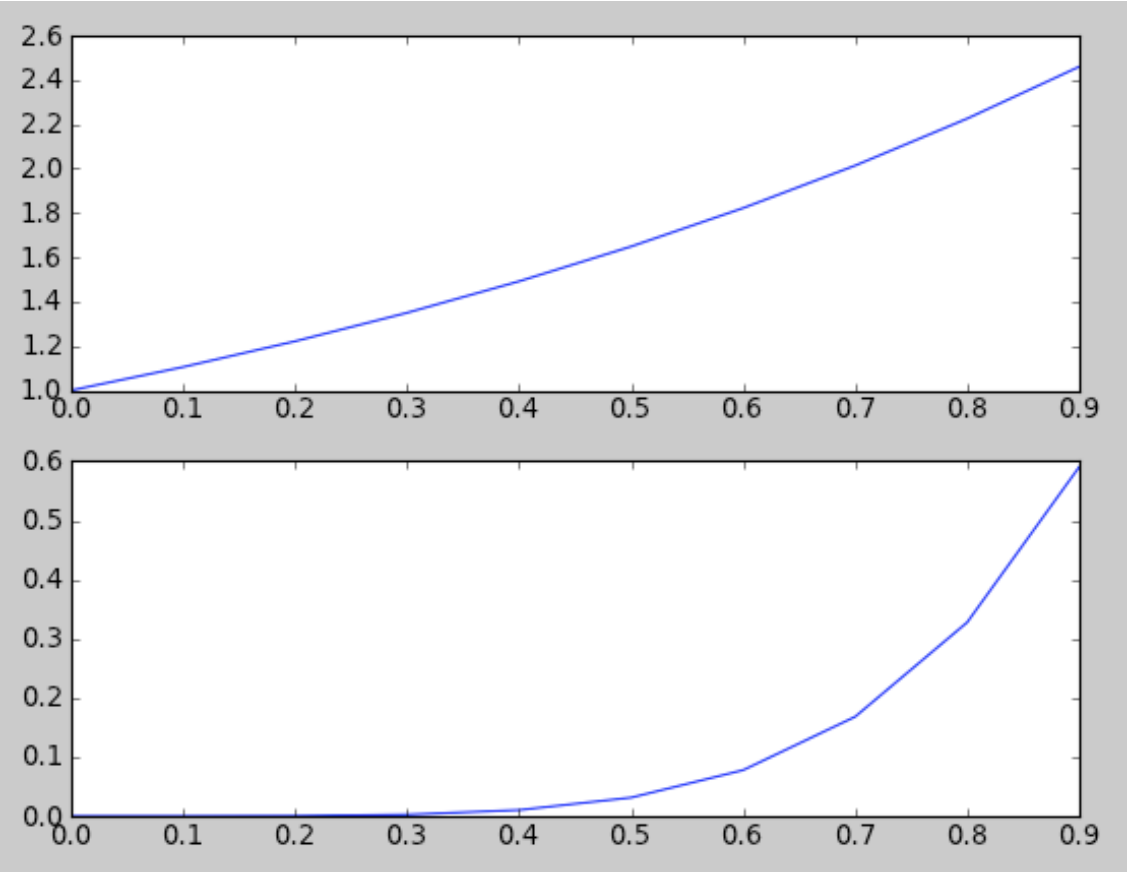
そのfigにはグラフを追加するためのadd_subplot(a, b, c)関数がある．引数a, b, cはキャンバス全体をa*b個に区切ったスペースのc番目のところにグラフを一つつくることを表す．この場合は，全体に一つのグラフをつくる．これをaxに割り当てる.

axはaxisの略で，matplotlibではいわゆるグラフをaxisと呼んでいて，これが慣習のよう．axには，これまで紹介した様々な形のグラフを描く関数に加え，ラベルやタイトルを指定するいくつかの関数があり，指定ができる（ax.set_xlabel(ラベル名), ax.set_title(タイトル名)など）.

一つの図に複数のプロット

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

fig= plt.figure()
x= np.arange(0, 1, 0.1)
ax1= fig.add_subplot(2, 1, 1)
ax1.plot(x, np.exp(x))
ax2=fig.add_subplot(212)
ax2.plot(x, x**5)
plt.show()
```

（前の例と重複するけど）まず最初の文で，グラフを描く領域そのもののオブジェクトをつくり，変数figに割り当てている．つづけて，figに分割グラフを割り当 てる関数add_subplot()を実行し，生成されたサブプロットのオブジェクトに変数ax1を割り当てている.
add_subplot(a, b, c)で，全体をa行b列に分割し，その中のc番目の領域をいま生成するサブプロットに割り当てることになる.
この場合，2x1の領域に分け，1番目の領域を割り当てている.
最後に，ax1に関して，これまでpltについて実行していたのと同様に，plot関数を実行してグラフを生成する.
同様の操作をax2に関しても行い，2つのサブプロットを異なる分割領域に描画している.

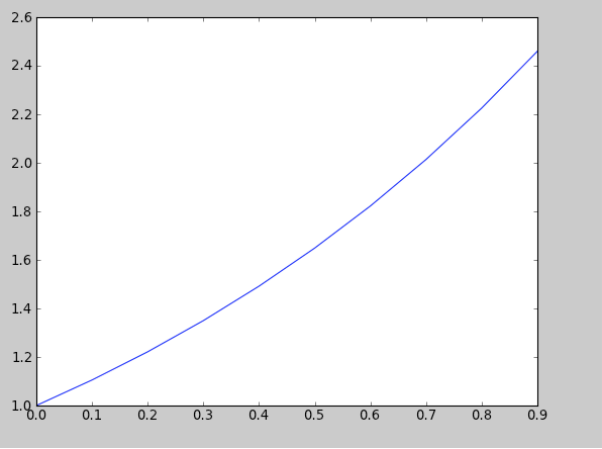


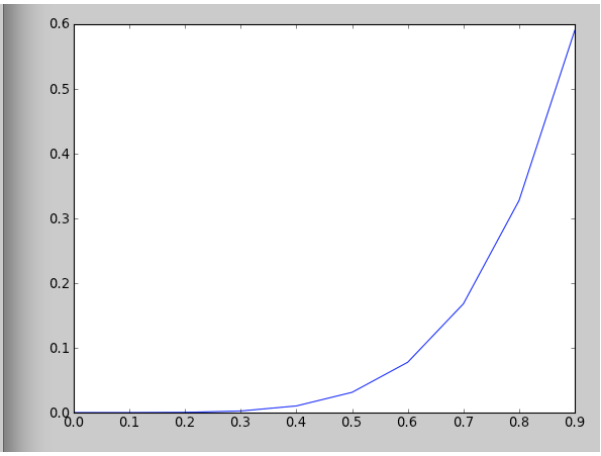
複数の図の（同時）生成

グラフの描画領域を複数生成し，変数にそれぞれ割り当てて，複数の図を同時に描画することもできる.

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3
4  fig= plt.figure()
5  x= np.arange(0, 1, 0.1)
6  ax1= fig.add_subplot(1, 1, 1)
7  ax1.plot(x, np.exp(x))
8  fig2= plt.figure()
9  ax2=fig2.add_subplot(111)
10 ax2.plot(x, x**5)
11 plt.show()
```

ax1はfig1のサブプロットに，ax2はfig2のサブプロットに割り当てられているのに注意.





データの読み込み

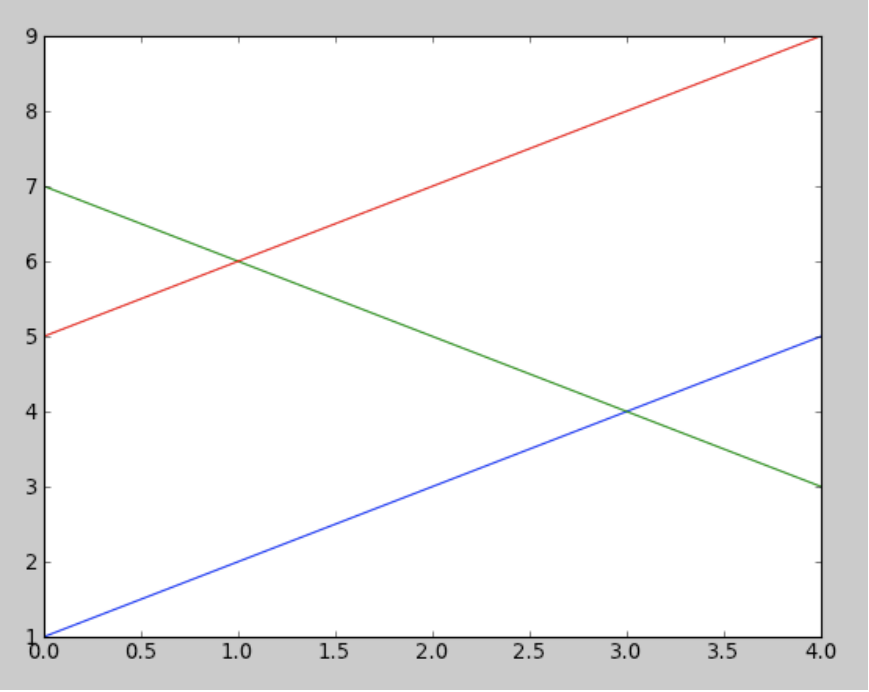
pythonのスクリプト中で生成したデータ以外にも，データが保存された既存のテキストファイルからデータを取り込むことができる.
例えば，ソースコードと同じディレクトリに以下のようなタブ区切りのテキストファイルsampledata.txtがあったとする.
sampledata.txt：

```
1	7	5
2	6	6
3	5	7
4	4	8
5	3	9
```

これを読み込んで折れ線グラフで表示することを考える.

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3
4  import os, sys
5  os.chdir(os.path.dirname(os.path.abspath(sys.argv[0])))
6
7  data= np.loadtxt("sampledata.txt")
8  plt.plot(data)
9  plt.show()
```

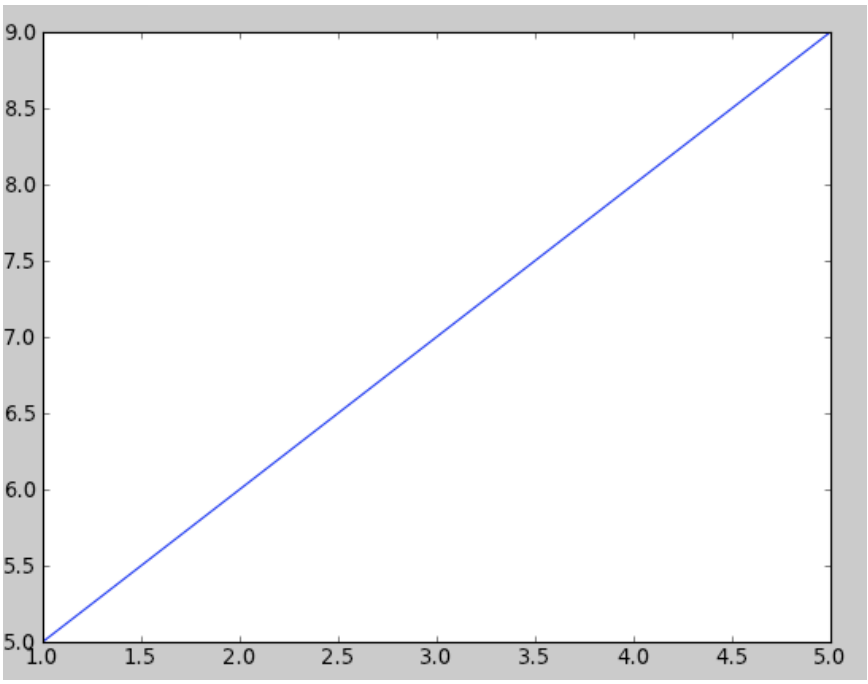
loadtxt関数は指定したファイルにあるいわゆるスプレッドシート形式のタブ区切りのデータを，numpyの配列の二次元データとして取り込む関数．カンマ区 切りの場合は，loadtxt関数の引数に「delimiter=','」と書く.



また，「usecols=(1, 3)」などとして，受け取る列を指定することもできる．「unpack=True」と指定すると，受け取る変数を複数指定し，例えば，

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

a, b= np.loadtxt("sampledata.txt", usecols=(0, 2), unpack=True)

plt.plot(a, b)
plt.show()
```

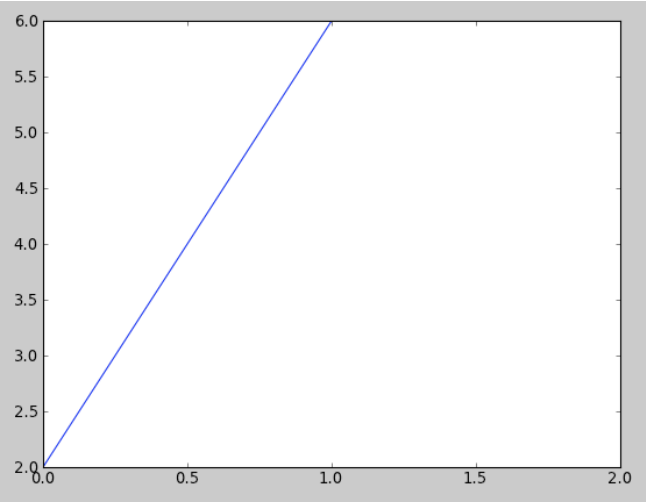
などと書ける．
aには0列目，bには2列目が入る．



注意：
例えば，sampledata.txtの真ん中の列を表示しようとして，

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

data= np.loadtxt("sampledata.txt")
x= data[1]

plt.plot(x)
plt.show()
```

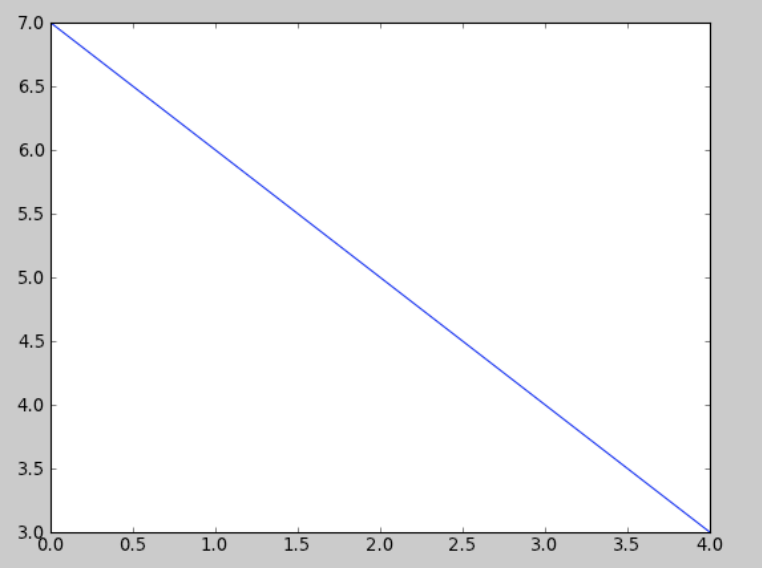
とするのは誤り．data[1]は1行目の横並び3つ分のデータのリスト[2, 6, 6]が入ってしまうから，以下の様になる．



dataにごっそりファイルの内容を移したあと，1列目のデータのみをとりたいときは，

x= [data[i][1] for i in range(len(data))]

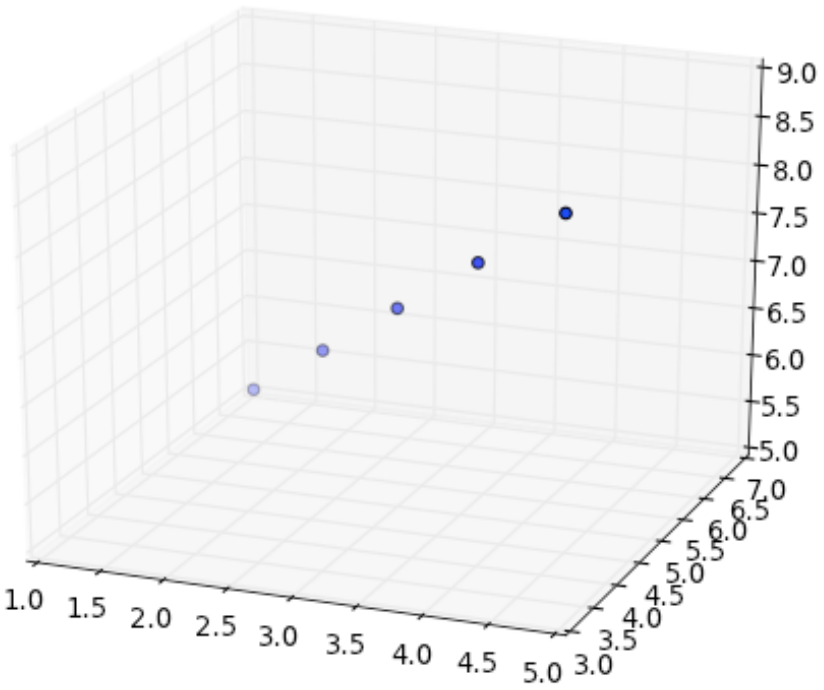
などとする．



3次元グラフ

上のデータをX, Y, Z系列だとおもって，3次元グラフをかくこともできる.

```
import numpy as np
import matplotlib.pyplot as plt
import mpl_toolkits.mplot3d.axes3d

data= np.loadtxt("sampledata.txt")

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

X= [data[i][0] for i in range(len(data))]
Y= [data[i][1] for i in range(len(data))]
Z= [data[i][2] for i in range(len(data))]

ax.scatter(X, Y, Z)

plt.show()
```



練習

N=100個のx座標，y座標がそれぞれ平均0，分散1の正規乱数で決められた点の集合を考える.
横軸をx，縦軸をyとした点集合の散布図と，x座標，y座標それぞれに関する値のヒストグラム，点の数，x座標・y座標それぞれの平均と分散を表示する，以下のよ うなグラフを作成してみましょう.
なお，numpyのarray配列xについて，その値の平均・分散を出す関数がnp.average(x),np.std(x)として定義されていますので，利用してみてください．また，グラフ領域そのものの表示を消した

い場合は，　ax.set_axis_off()という命令が使えます．