

Python程序设计

陈远祥

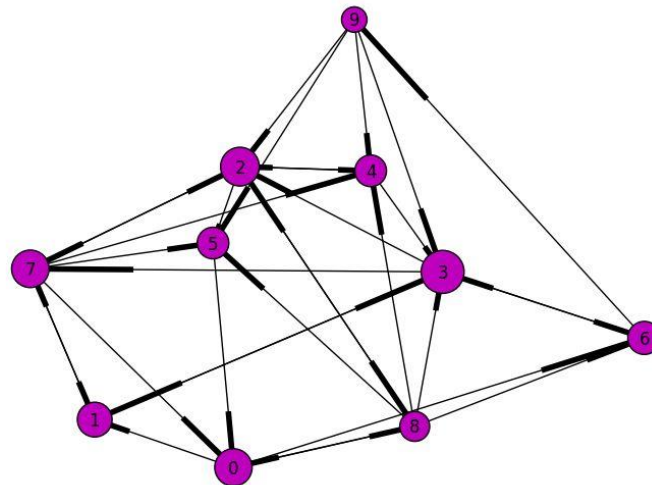
chenyxmail@gmail.com

北京邮电大学 电子工程学院





Matplotlib



Matplotlib

- `matplotlib`是python最著名的绘图库，它提供了一整套和`matlab`相似的命令API，十分适合交互式地进行制图。而且也可以方便地将它作为绘图控件，嵌入GUI应用程序中
- `matplotlib`文档相当完备，并且Gallery页面中有上百幅缩略图，打开之后都有源程序。因此如果你需要绘制某种类型的图，只需要在这个页面中浏览/复制/粘贴一下，基本上都能搞定

Matplotlib

■ 展示页面的地址:

<http://matplotlib.sourceforge.net/gallery.html>

Matplotlib

■ 快速绘图

- ✓ matplotlib的pyplot子库提供了和matlab类似的绘图API，方便用户快速绘制2D图表

Matplotlib

■ 快速绘图

- ✓ matplotlib中的快速绘图的函数库可以通过如下语句载入：`import matplotlib.pyplot as plt`
- ✓ 接下来调用figure创建一个绘图对象，并且使它成为当前的绘图对象：`plt.figure(figsize=(8,4))`
- ✓ 通过figsize参数可以指定绘图对象的宽度和高度，单位为英寸；dpi参数指定绘图对象的分辨率，即每英寸多少个像素，缺省值为80。因此本例中所创建的图表窗口的宽度为 $8*80=640$ 像素

Matplotlib

■ 快速绘图

- ✓ 也可以不创建绘图对象直接调用接下来的`plot`函数直接绘图，`matplotlib`会自动创建一个绘图对象
- ✓ 如果需要同时绘制多幅图表的话，可以是给`figure`传递一个整数参数指定图标的序号，如果所指定序号的绘图对象已经存在的话，将不创建新的对象，而只是让它成为当前绘图对象

Matplotlib

■ 快速绘图

- ✓ 下面的两行程序通过调用`plot`函数在当前的绘图对象中进行绘图：

```
plt.plot(x,y,label="$sin(x)$",color="red",linewidth=2)  
plt.plot(x,z,"b--",label="$cos(x^2)$")
```

- ✓ `plot`函数的调用方式很灵活，第一句将`x, y`数组传递给`plot`之后，用关键字参数指定各种属性：

`label`: 给所绘制的曲线一个名字，此名字在图示(`legend`)中显示。只要在字符串前后添加 ‘`$`’ 符号，`matplotlib`就会使用其内嵌的`latex`引擎绘制的数学公式；`color`: 指定曲线的颜色；`linewidth`: 指定曲线的宽度；第三个参数 ‘`b--`’ 指定曲线的颜色和线型

Matplotlib

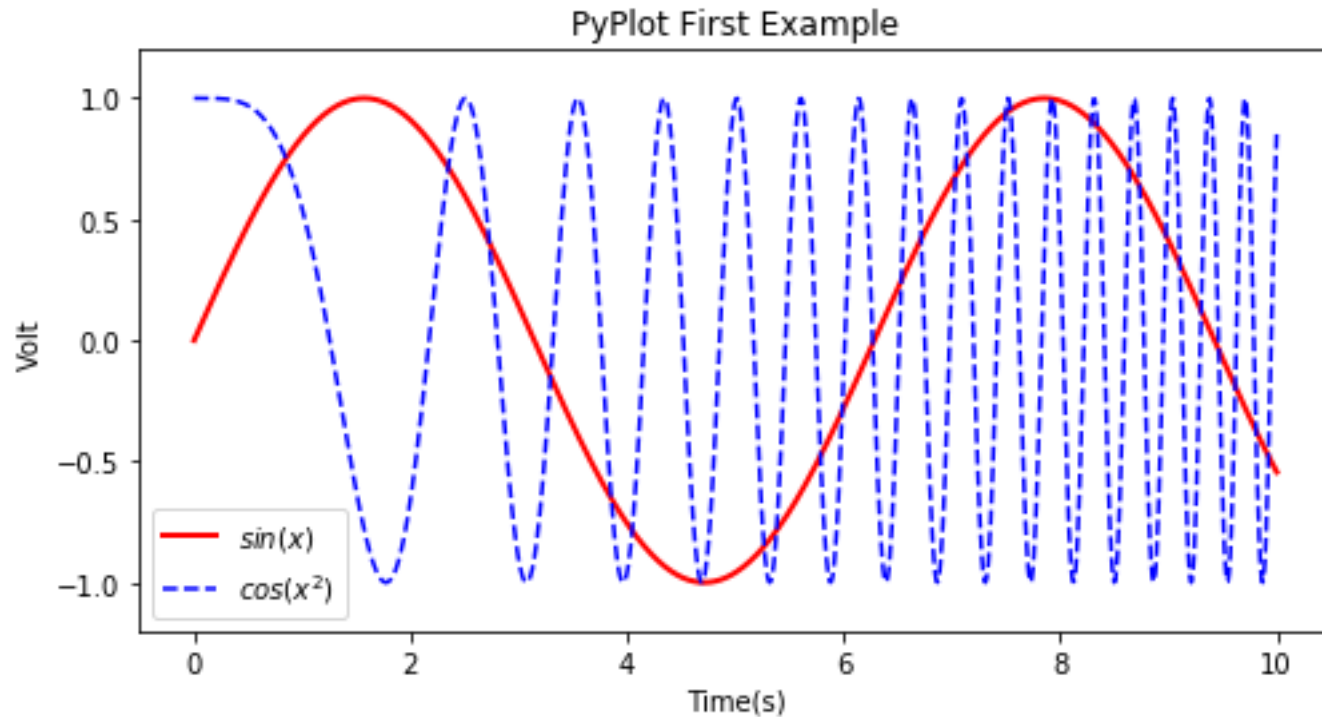
■ 快速绘图

- ✓ 通过一系列函数设置绘图对象的各个属性:

```
plt.xlabel("Time(s)")  
plt.ylabel("Volt")  
plt.title("PyPlot First Example")  
plt.ylim(-1.2,1.2)  
plt.legend()  
plt.show()
```

- ✓ `xlabel/ylabel`: 设置X轴/Y轴的文字
- ✓ `title`: 设置图表的标题
- ✓ `ylim`: 设置Y轴的范围
- ✓ `legend`: 显示图示
- ✓ `plt.show()` 显示出创建的所有绘图对象

Matplotlib



`matplotlib_simple_plot`

Matplotlib

■ 快速绘图

- ✓ 可以调用`plt.savefig()`将当前的Figure对象保存成图像文件，图像格式由图像文件的扩展名决定。下面的程序将当前的图表保存为“test.png”，并且通过`dpi`参数指定图像的分辨率为120，因此输出图像的宽度为“ $8 \times 120 = 960$ ”个像素
- ✓ 可以直接用`savefig()`将图表保存成图像文件. 使用这种方法可以很容易编写出批量输出图表的程序

```
run matplotlib_simple_plot.py  
plt.savefig("test.png",dpi=120)
```

Matplotlib

■ 绘制多轴图

- ✓ 一个绘图对象 (figure) 可以包含多个轴 (axis)，在 Matplotlib 中用轴表示一个绘图区域，可以将其理解为子图。上面的第一个例子中，绘图对象只包括一个轴，因此只显示了一个轴 (子图 (Axes))。可以使用 subplot 函数快速绘制有多个轴的图表。subplot 函数的调用形式如下：

```
subplot(numRows, numCols, plotNum)
```

Matplotlib

■ 绘制多轴图

- ✓ `subplot`将整个绘图区域等分为`numRows`行和`numCols`列个子区域，然后按照从左到右，从上到下的顺序对每个子区域进行编号，左上的子区域的编号为1。如果`numRows`，`numCols`和`plotNum`这三个数都小于10的话，可以把它们缩写为一个整数，例如`subplot(323)`和`subplot(3, 2, 3)`是相同的。`subplot`在`plotNum`指定的区域中创建一个轴对象。如果新创建的轴和之前创建的轴重叠的话，之前的轴将被删除

Matplotlib

■ 绘制多轴图

- ✓ 下面的程序创建3行2列共6个轴，通过`axisbg`参数给每个轴设置不同的背景颜色

```
for idx, color in enumerate("rgbk"):
    plt.subplot(320+idx+1, axisbg=color)
plt.show()
```

- ✓ 如果希望某个轴占据整个行或者列的话，可以如下调用`subplot`:

```
plt.subplot(221) # 第一行的左图
plt.subplot(222) # 第一行的右图
plt.subplot(212) # 第二整行
plt.show()
```

- ✓ #多轴图

```
@author: cnenlox
"""
import numpy as np
import matplotlib.pyplot as plt
#for idx, color in enumerate("rgbyck"):
#    plt.subplot(320+idx+1, axisbg=color)
#plt.show()

plt.subplot(221) # 第一行的左图
plt.subplot(222) # 第一行的右图
plt.subplot(212) # 第二整行
plt.show()
```

Matplotlib

■ 循环绘图

- ✓ `subplot()` 返回它所创建的Axes对象，可以将它用变量保存起来，然后用 `sca()` 交替让它们成为当前Axes对象，并调用 `plot()` 在其中绘图。如果需要同时绘制多幅图表，可以给 `figure()` 传递一个整数参数指定Figure对象的序号，如果序号所指定的figure对象已经存在，将不创建新的对象，而只是让它成为当前的Figure对象
- ✓ #循环绘图


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

plt.figure(1) # 创建图表1
plt.figure(2) # 创建图表2
ax1 = plt.subplot(211) # 在图表2中创建子图1
ax2 = plt.subplot(212) # 在图表2中创建子图2

x = np.linspace(0, 3, 100)
for i in range(5):
    plt.figure(1) # 选择图表1
    plt.plot(x, np.exp(i*x/3))
    plt.sca(ax1) # 选择图表2的子图1
    plt.plot(x, np.sin(i*x))
    plt.sca(ax2) # 选择图表2的子图2
    plt.plot(x, np.cos(i*x))
plt.show()
```

Matplotlib

■ 添加文字说明和注释

- ✓ `plt.text()` : 在图中的任意位置添加文字, 并支持 LaTeX 语法
- ✓ `plt.annotate()` : 注释图中的特征
- ✓ #注释和说明

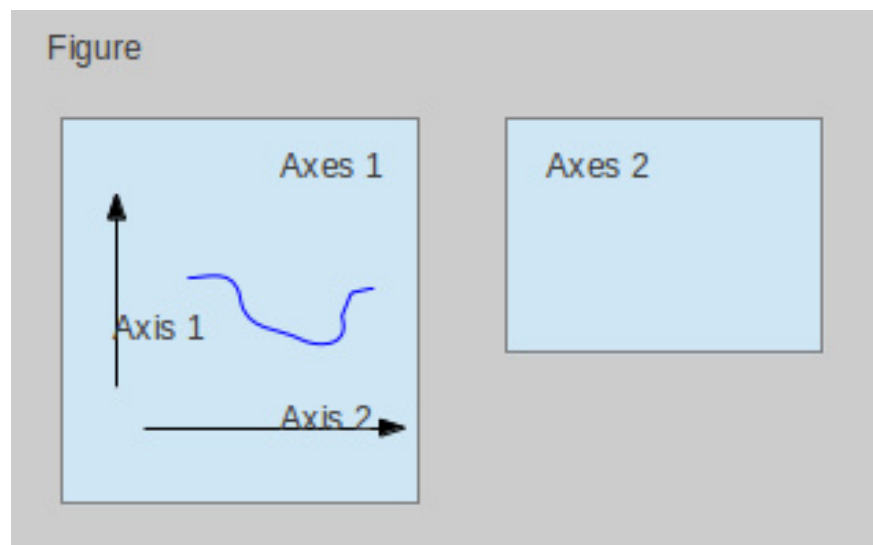
```
import numpy as np
import matplotlib.pyplot as plt
ax = plt.subplot(111)
t = np.arange(0.0, 5.0, 0.01)
s = np.cos(2*np.pi*t)
line, = plt.plot(t, s, lw=2)
plt.text(1, 1, '(1,1)')

plt.annotate('local max', xy=(2, 1), xytext=(3, 1.5),
            arrowprops=dict(facecolor='black', shrink=0.05),
            )
plt.ylim(-2,2)
plt.show()
```

Matplotlib

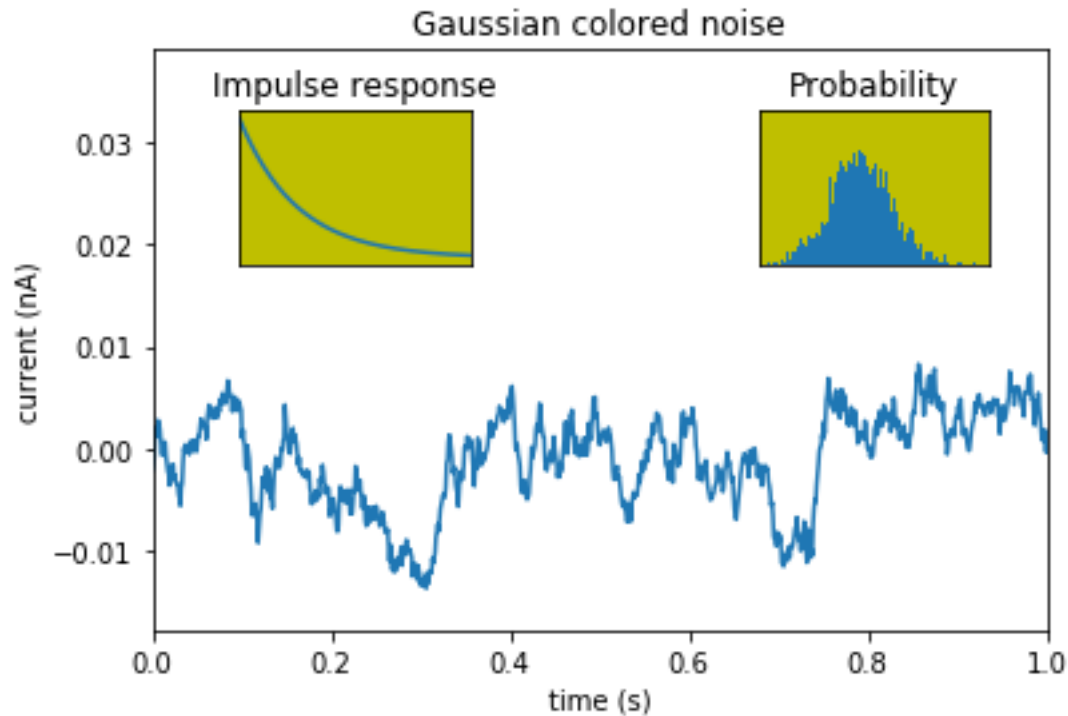
■ 图中插入图：plt.axes()

- ✓ 在matplotlib中，整个图像为一个Figure对象。在Figure对象中可以包含一个，或者多个Axes对象。每个Axes对象都是一个拥有自己坐标系统的绘图区域。其逻辑关系如下



Matplotlib

■ `plt.axes()` : #图中图



Matplotlib

■ 对数坐标图：

- ✓ 绘制对数坐标图的函数有三个：`semi logx()`、`semi logy()`和`log log()`，它们分别绘制X轴为对数坐标、Y轴为对数坐标以及两个轴都为对数坐标时的图表

Matplotlib

■ 对数坐标图：

- ✓ 使用4种不同的坐标系绘制低通滤波器的频率响应曲线。其中，左上图为`plot()`绘制的算术坐标系，右上图为`semi logx()`绘制的X轴对数坐标系，左下图为`semi logy()`绘制的Y轴对数坐标系，右下图为`log log()`绘制的双对数坐标系。使用双对数坐标系表示的频率响应曲线通常被称为波特图
- ✓ #对数坐标

Matplotlib

■ 极坐标图：

- ✓ 极坐标系是和笛卡尔 (X-Y) 坐标系完全不同的坐标系，极坐标系中的点由一个夹角和一段相对中心点的距离来表示
- ✓ #极坐标

Matplotlib

■ 柱状图：

- ✓ 柱状图用其每根柱子的长度表示值的大小，它们通常用来比较两组或多组值
- ✓ #柱状图

Matplotlib

■ 直方图:

✓ #直方图

Matplotlib

■ 散点图:

✓ #散点图

Matplotlib

■ 等高线图:

✓ #等高线图

Matplotlib

■ 饼图:

✓ #饼图

Matplotlib

■ 3D图表：

- ✓ Matplotlib中也能支持一些基础的3D图表，比如曲面图，散点图和柱状图。这些3D图表需要使用mpl_toolkits模块
- ✓ #3d曲面图
- ✓ #3d散点图

Matplotlib

- 除了绘制三维曲面之外，Axes3D对象还提供了许多其他的三维绘图方法。可以通过下面的链接地址找到各种三维绘图的演示程序
- ✓ <https://matplotlib.org/examples/mplot3d/index.html>

Matplotlib

■ 图像：

- ✓ `imread()` 和 `imshow()` 提供了简单的图像载入和显示功能
- ✓ `imread()` 可以从图像文件读入数据，得到一个表示图像的NumPy数组。它的第一个参数是文件名或文件对象，`format`参数指定图像类型，如果省略，就由文件的扩展名决定图像类型
- ✓ 对于灰度图像，它返回一个形状为 (M, N) 的数组；对于彩色图像，返回形状为 (M, N, C) 的数组。其中， M 为图像的高度， N 为图像的宽度， C 为3或4，表示图像的通道数

Matplotlib

■ 图像:





Matplotlib

■ 图片故事：

- ✓ 1973年6月，美国南加州大学的一名教授想找一幅图像来做图像压缩的测试，他已厌倦了手头繁杂的照片，想找张能让人眼前一亮的照片。恰好这时，一人拿着《花花公子》走了进来，Lena的照片确实够让教授眼前一亮了。教授便将《花花公子》的这期插图用扫描了下来截取其中的一部分作为了他研究使用的样例图像。这位教授就是IPL（图像处理研究所）的 William K. Pratt博士
- ✓ 从此，这幅512*512的经典图像就诞生了。之后从事影像数据的压缩、运算、传输、解压缩等处理时，都经常采用这张图像来当测试样本。采用这张图像的原因，除了因为它很赏心悦目外，就“测试标准”来说，它的鉴别度也相当的高。这张图像的确具备“测试标准”所应有的充分条件，平整的区块、清晰细致的纹路、渐渐变化的光影、颜色的深浅层次等，使它在验证影像处理演绎法则时，相当有成效

Matplotlib

■ 图像：

- ✓ 从“lena.jpg” 中读入图像数据，得到的数组img是一个形状为(393, 512, 3)的单字节无符号整数数组。这是因为通常使用的图像都是采用单字节分别保存每个像素的红、绿、蓝三个通道的分量：
- ✓ `img = plt.imread("lena_black.jpg")`
- ✓ `img1 = plt.imread("lena_cor.jpg")`
- ✓ `imshow()` 可以用来显示`imread()` 返回的数组
- ✓ #图像

Matplotlib

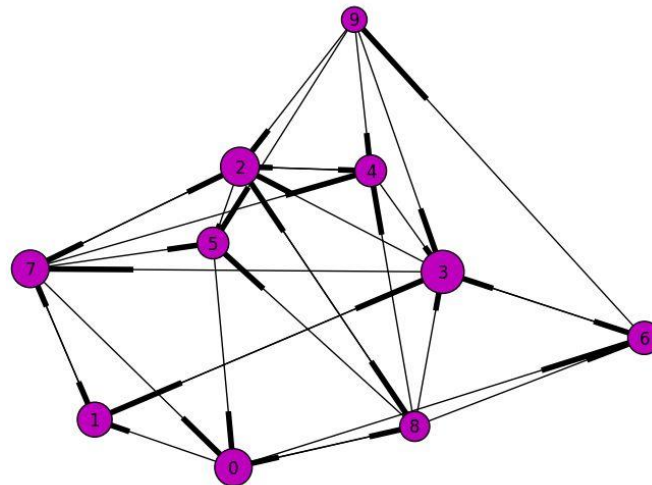
- 还可以使用 `imshow()` 显示任意的二维数据，
例如下面的程序使用图像直观地显示了二元函数

$$f(x, y) = xe^{x^2 - y^2}$$

✓ #二元函数



mayavi



mayavi

- Mayavi项目是一个基于开源C++图形库VTK的3D图形工具包。和matplotlib一样，mayavi也能集成到开发环境实现交互式使用。通过鼠标和键盘操作，图形可以被平移，旋转，缩放
- Mayavi在梵语中的意思是魔术师，它是一种数据可视化工具，绑定了具有强大可视化工具包（VTK）的Python来进行图形化显示

mayavi

■ Mayavi. lab

类 别	说明
绘图函数	barchar、contour3d、contour_surf、flow、imshow、mesh、plot3d、points3d、quiver3d、surf、triangular_mesh
图形控制函数	clf、close、draw、figure、gcf、savefig、screenshot、sync_camera
图形修饰函数	colorbar、scalarbar、xlabel、ylabel、zlabel
相机控制函数	move、pitch、roll、view、yaw
其他函数	animate、axes、get_engine、show、set_engine.....
Mlab管线控制	Open、set_vtk_src、adddataset、scalar_cut_plane

mayavi

■ 常用的三维绘图函数和简单例子

- ✓ `barchart` (柱状图): 传递参数可以是一维 (1-D), 二维 (2-D) 或3维 (3-D) 的给定向量长度的数组
- ✓ `#mayavi_barchart`

mayavi

- 常用的三维绘图函数和简单例子
 - ✓ `surf` (表面图) : 输入三维数组
 - ✓ `#mayavi_surf`
 - ✓ `#mayavi_surf`, 双峰值三维高斯模型

mayavi

■ 常用的三维绘图函数和简单例子

- ✓ mesh (表面图) : 输入三维数组
- ✓ #mayavi_mesh
- ✓ #mayavi_mesh2

mayavi

- 常用的三维绘图函数和简单例子
 - ✓ `contour3d`（轮廓图）：输入三维数组
 - ✓ `#mayavi_contour`

mayavi

- 常用的三维绘图函数和简单例子
 - ✓ Plot3d (数据点之间绘制线段)
 - ✓ #mayavi_plot3d

mayavi

- 常用的三维绘图函数和简单例子
 - ✓ `points3d`
 - ✓ `#mayavi_point3d`

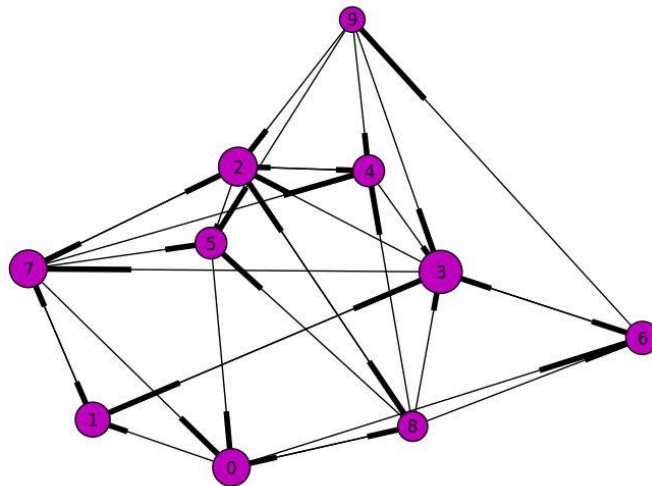
mayavi

■ 常用的三维绘图函数和简单例子

- ✓ `quiver3d`: 绘制字形（如箭头），指示所提供位置处矢量的方向
- ✓ `flow()`: 绘制三维三维阵列描述的矢量场中的粒子轨迹，给出网格上的u, v, w分量
- ✓ `#mayavi_quiver3d`



动画



动画

- `animation.FuncAnimation(fig, animate, init_func, frames, interval, blit)`
 - ✓ `fig`是画布
 - ✓ `animate`是绘画函数需自己定义，需要一个参数，会自动接收`data`，需要返回`plt.plot`对象
 - ✓ `data`种类很多，包括总帧数、当前帧数、返回迭代器的函数、`list`
 - ✓ `frames`总帧数（非`update`参数）
 - ✓ `interval`帧间隔（毫秒）

动画

- #sine
- #sineline
- #sinedot
- #rollball
- #mayavi_animaiton

谢谢