介绍

Matplotlib 可能是 Python 2D-绘图领域使用最广泛的套件。它能让使用者很轻松地将数据图形化，并且提供多样化的输出格式。这里将会探索 matplotlib 的常见用法。

pylab

pylab 是 matplotlib 面向对象绘图库的一个接口。它的语法和 Matlab 十分相近。也就是说，它主要的绘图命令和 Matlab 对应的命令有相似的参数。

图像、子图、坐标轴和记号

基础

1.matplotlib绘图函数接收两个等长list，第一个作为集合x坐标，第二个作为集合y坐标

2.基本函数：

animation.FuncAnimation(fig, update\_point,data)

fig是画布

update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，描述比较费解，看例子就好

data种类很多，包括总帧数（例1）、当前帧数（即不设定data的默认参数，例2）、返回迭代器的函数（例3）、list（作业2）

frames=200 总帧数（非update参数）

interval=20 帧间隔（毫秒）

from matplotlib import pyplot as plt

import matplotlib.animation as animation

import numpy as np

def update\_point(num):

    fig\_points.set\_data(data[:, 0:num])

    return fig\_points,

fig1 = plt.figure()

num\_point = 50

data = np.random.rand(2, num\_point)

fig\_points, = plt.plot([], [], 'ro')

plt.xlim(0, 1)

plt.ylim(0, 1)

plt.xlabel('x')

plt.title('Scatter Point')

# interval

# repeat

# frames

# fargs

# init\_func

anim = animation.FuncAnimation(fig1, update\_point,num\_point)

#anim = animation.FuncAnimation(fig1, update\_point,frames=num\_point, interval=50, blit=False, repeat=False)

plt.show()

利用帧做参数绘制

这种方式每经过interval的时间后会调用函数（传入当前帧号）绘制一幅新图更新原图：

建立子图、空白线

创建动画发生时调用的函数

Init()是我们的动画在在创建动画基础框架(base frame)时调用的函数。这里我们们用一个非常简单的对line什么都不做的函数。这个函数一定要返回line对

象，这个很重要，因为这样就能告诉动画之后要更新的内容，也就是动作的内容是line。--来自（ http://mytrix.me/2013/08/matplotlib-animation-tutorial/ ）

动画函数

在这个动画函数中修改你的图

调用函数生成动态图

绘制正弦波函数：

可以使用多个线对象来同时更新多个子图于同一个动画生成器之中，不过这需要上面1~3步同时支持（就是写出来）多个线对象

import numpy as np

from matplotlib import pyplot as plt

from matplotlib import animation

# 1.First set up the figure, the axis, and the plot element we want to animate

fig = plt.figure()

ax = plt.axes(xlim=(0,2),ylim=(-2,2))

line, = ax.plot([],[],lw=2)

# 2.initialization function: plot the background of each frame

def init():

    line.set\_data([],[])

    return line,

# 3.animation function.  This is called sequentially

# note: i is framenumber

def update(i):

    x = np.linspace(0,2,1000)

    y = np.sin(2 \* np.pi \* (x - 0.01 \* i))  # 调整x相当于向右平移图像

    line.set\_data(x,y)

    return line,

# call the animator.  blit=True means only re-draw the parts that have changed.

# 画布， 使用帧数做参数的绘制函数， init生成器.。

anim = animation.FuncAnimation(fig,update,init\_func=init,frames=200,interval=20,blit=False)

# frames=200   帧数

# interval=20  间隔

# anim.save('anim3.mp4', fps=30, extra\_args=['-vcodec', 'libx264'])      # 保存为mp4

# anim.save('anim3.gif', writer='imagemagick')                           # 保存为gif

plt.show()

迭代器绘制法

绘制衰减波

这里的update函数仅仅绘制图像，数据生成被移植到函数data\_gen中了，而上面的帧方法中update得到帧号后会自己生成数据。

到目前为止，我们都用隐式的方法来绘制图像和坐标轴。快速绘图中，这是很方便的。我们也可以显式地控制图像、子图、坐标轴。Matplotlib 中的「图像」指的是用户界面看到的整个窗口内容。在图像里面有所谓「子图」。子图的位置是由坐标网格确定的，而「坐标轴」却不受此限制，可以放在图像的任意位置。我们已经隐式地使用过图像和子图：当我们调用 plot 函数的时候，matplotlib 调用 gca() 函数以及 gcf() 函数来获取当前的坐标轴和图像；如果无法获取图像，则会调用 figure() 函数来创建一个——严格地说，是用 subplot(1,1,1) 创建一个只有一个子图的图像。

图像

所谓「图像」就是 GUI 里以「Figure #」为标题的那些窗口。图像编号从 1 开始，与 MATLAB 的风格一致，而于 Python 从 0 开始编号的风格不同。以下参数是图像的属性：

|  |  |  |
| --- | --- | --- |
| 参数 | 默认值 | 描述 |
| num | 1 | 图像的数量 |
| figsize | figure.figsize | 图像的长和宽（英寸） |
| dpi | figure.dpi | 分辨率（点/英寸） |
| facecolor | figure.facecolor | 绘图区域的背景颜色 |
| edgecolor | figure.edgecolor | 绘图区域边缘的颜色 |
| frameon | True | 是否绘制图像边缘 |

这些默认值可以在源文件中指明。不过除了图像数量这个参数，其余的参数都很少修改。

你在图形界面中可以按下右上角的 X 来关闭窗口（OS X 系统是左上角）。Matplotlib 也提供了名为 close 的函数来关闭这个窗口。close 函数的具体行为取决于你提供的参数：

不传递参数：关闭当前窗口；

传递窗口编号或窗口实例（instance）作为参数：关闭指定的窗口；

all：关闭所有窗口。

和其他对象一样，你可以使用 setp 或者是 set\_something 这样的方法来设置图像的属性。

子图

你可以用子图来将图样（plot）放在均匀的坐标网格中。用 subplot 函数的时候，你需要指明网格的行列数量，以及你希望将图样放在哪一个网格区域中。此外，[gridspec](http://matplotlib.sourceforge.net/users/gridspec.html" \t "_blank) 的功能更强大，你也可以选择它来实现这个功能。

坐标轴

坐标轴和子图功能类似，不过它可以放在图像的任意位置。因此，如果你希望在一副图中绘制一个小图，就可以用这个功能。

记号

良好的记号是图像的重要组成部分。Matplotlib 里的记号系统里的各个细节都是可以由用户个性化配置的。你可以用 Tick Locators 来指定在那些位置放置记号，用 Tick Formatters 来调整记号的样式。主要和次要的记号可以以不同的方式呈现。默认情况下，每一个次要的记号都是隐藏的，也就是说，默认情况下的次要记号列表是空的——NullLocator。

Tick Locators

下面有为不同需求设计的一些 Locators。

|  |  |
| --- | --- |
| 类型 | 说明 |
| NullLocator | No ticks |
| IndexLocator | Place a tick on every multiple of some base number of points plotted. |
| FixedLocator | Tick locations are fixed. |
| LinearLocator | Determine the tick locations. |
| MultipleLocator | Set a tick on every integer that is multiple of some base. |
| AutoLocator | Select no more than n intervals at nice locations. |
| LogLocator | Determine the tick locations for log axes. |

这些 Locators 都是 matplotlib.ticker.Locator 的子类，你可以据此定义自己的 Locator。以日期为 ticks 特别复杂，因此 Matplotlib 提供了 matplotlib.dates 来实现这一功能。

其他类型的图

接下来的内容是练习。请运用你学到的知识，从提供的代码开始，实现配图所示的效果。具体的答案可以点击配图下载。