2.1 matplotlib简介

在气象、基因、政治经济趋势分析等科研领域，Python常常被数据科学家用于数据密集型研究。基于Python的可视化和分析工具有很多，最流行的之一是matplotlib，这是一个功能性和扩展性很强的代码库。其中，最常用的模组是matplotlib.pyplot，它能让matplotlib以类似MATLAB方式运行的命令式函数集合。使用pyplot函数可以进行图表的创建、指定绘图区域、线条绘制、标注图表等功能。在matplotlib.pyplot中，不同的状态信息通过函数调用来保存，以便记录诸如当前图表和绘图区域等信息。

2.2 matplotlib安装与测试

在Windows操作系统中，首先需要安装微软的Visual Studio。在<http://dev.windows.com/>上，可以搜索、下载并安装Visual Studio Community，这是Visual Studio的社区免费版开发工具。

和许多扩展库一样，matplotlib的安装可以通过pip进行。如果你已经安装了Python 3.x或以上版本，则安装Python时就已经自带了pip功能。打开命令行窗口，输入pip install matplotlib命令，即可自动下载并安装matplotlib。

安装完成后，可以打开一个命令行窗口进入Python终端会话，对之前的安装进行测试。输入import matplotlib命令，尝试导入matplotlib，如果没有出现任何错误信息提示，说明说明matplotlib已经成功安装，可以使用。

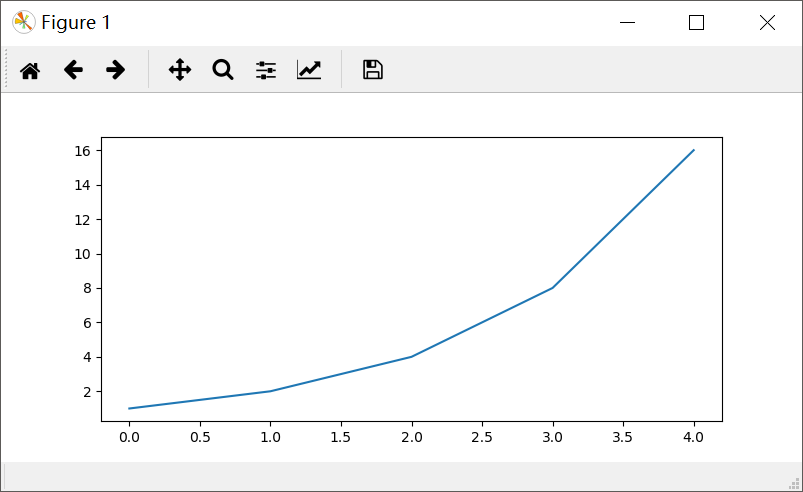
2.3 基本绘图

**2.3.1 绘制基本折线图**

首先我们来试着绘制折线图。新建一个Python文件，并输入以下代码：

|  |
| --- |
| **import** matplotlib.pyplot **as** plt  plt.plot([1, 2, 4, 8, 16]) plt.show() |

在这段代码中，我们首先导入了matplotlib库中的模组pyplot。为方便起见，我们为matplotlib.pyplot设置了在本程序中的别名plt，以简化后续程序的编写。



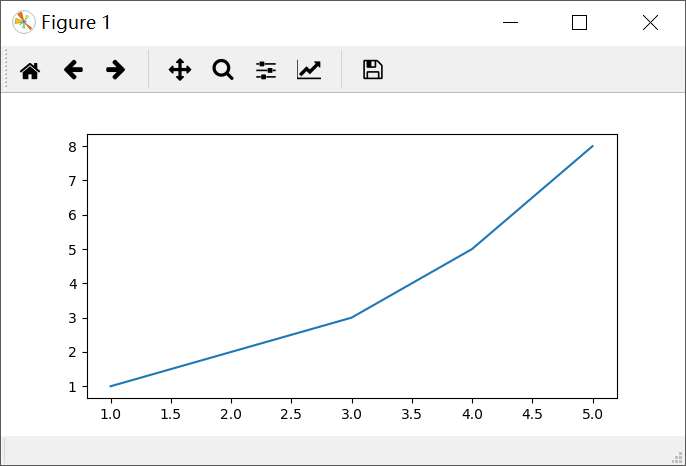
接下来，我们创建了一个包含有2的N次幂的列表，又将其作为参数，传递给了函数plot()，这个函数将根据给定的参数尝试绘制相应的图形。在第二行中，通过函数show()打开matplotlib查看器，并最终显示绘制结果，如上图所示。

在上例中，虽然我们只向plot()函数传递了一个列表或者数组，但仍然绘制出了包含x轴和y轴两个坐标轴信息的图表。这是因为在plot()函数只接收到一个列表作为参数时，将会默认这个列表存储的是y轴的数据，而x轴则默认从0开始，长度与y轴相同。因此，对于上例中给定了单个数组[1, 2, 4, 8, 16]作为参数的情形，x轴的数值为[0, 1, 2, 3, 4]。

plot()函数非常灵活，可以支持任意数量的参数输入。例如，若想同时指定x和y轴的数值，可以输入如下格式的代码：

|  |
| --- |
| plt.plot([1, 2, 3, 4, 5], [1, 2, 3, 5, 8]) |

此时，plot()接受了两个参数，[1, 2, 3, 4, 5]列表参数指定了横坐标的数值，第二个参数为列表[1, 2, 3, 5, 8]，指定了纵坐标的数值。于是可以得到以下图形：



**2.3.2 修改图表样式**

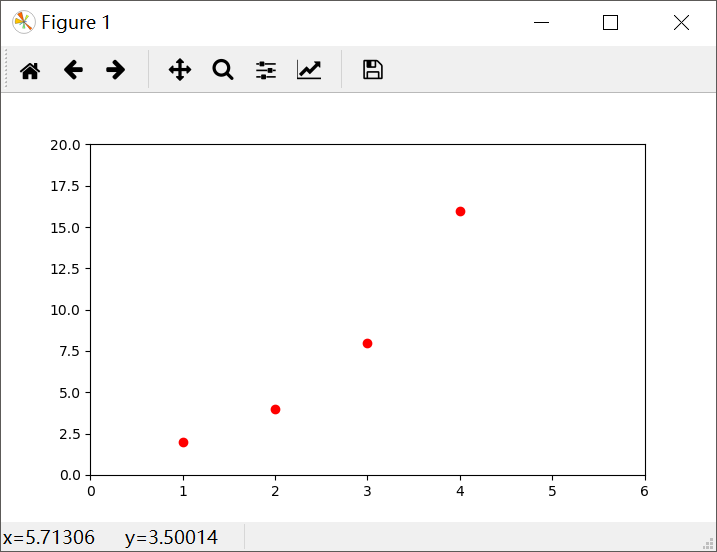
matplotlib允许用户以多种方式修改图表样式，以根据不同的需求和数据，针对性地增强图表可读性。

对于每一对x，y参数而言，都有一个可选的第三参数，用于格式化该对x，y所表示的点的颜色和线条类型。格式化字符串的字母和符号来源于MATLAB，你可以将颜色字符串与线条类型字符串联合使用。默认的格式字符串是“b-”，即上图所示的蓝色实线。例如，若想绘制一幅红色散点图，可以采用以下代码定制：

|  |
| --- |
| plt.plot([1, 2, 3, 4], [2, 4, 8, 16], **'ro'**)  plt.axis([0, 6, 0, 20]) plt.show() |

可以看到，在对plot()函数的调用中，我们除了给出了两个列表作为横纵坐标数值以外，还给出了一个字符串‘ro’作为第三个参数。正是这个参数，让默认的蓝色折线图变为了红色散点图。其中，r指定了点的颜色为红色；o指定了图表格式为散点图。读者可以尝试修改该字符串，例如删去r、删去o，或者将ro变为go、bo，观察图表会怎样发生变化。

在代码的第二行，我们调用了另一个函数axis()。该函数接受一个长度为4的列表，从而指定了坐标轴的显示范围。具体而言，该列表的格式为[xmin, xmax, ymin, ymax],即依次指定了x轴的最小值、最大值和y轴的最小值、最大值。



除了可以处理Python默认的列表格式外，matplotlib还可以处理numpy数组，这也大大拓展了matplotlib的功能性。下例就通过数组输入数据，并用不同格式进行了可视化表示。

|  |
| --- |
| **import** matplotlib.pyplot **as** plt **import** numpy **as** np  *# 每隔0.2距取值* t = np.arange(0, 5, 0.2)  *# 红色虚线，蓝色方块，绿色三角* plt.plot(t, t, **'r--'**, t, t\*\*2, **'bs'**, t, t\*\*3, **'g^'**) plt.show() |

首先，通过import numpy as np导入numpy库。随后调用numpy的arange()函数，创建从0到5，差为0.2的等差数组。arange()是一个用于创建一位数组的非常灵活的函数，其参数可以有多种格式。当参数为3个时，分别表示数组的下限、上限、间隔值。

随后，调用plot()函数，确定要绘制的散点图格式。可以看到，上面这段代码的plot()函数一共有9个参数，其中出现的t即为上面arange()生成的数组。其中每3个变量绘制了一系列散点：“t, t, ‘r--”绘制了一条y=x的红色虚线图像；“t, t\*\*2, ‘bs’”绘制了y=x2的蓝色方点图像；“t, t\*\*3, ‘g^’”绘制了y=x3的绿色三角点图像。

**2.3.3 使用关键字符串绘点**

有些时候，你可能会让

#end