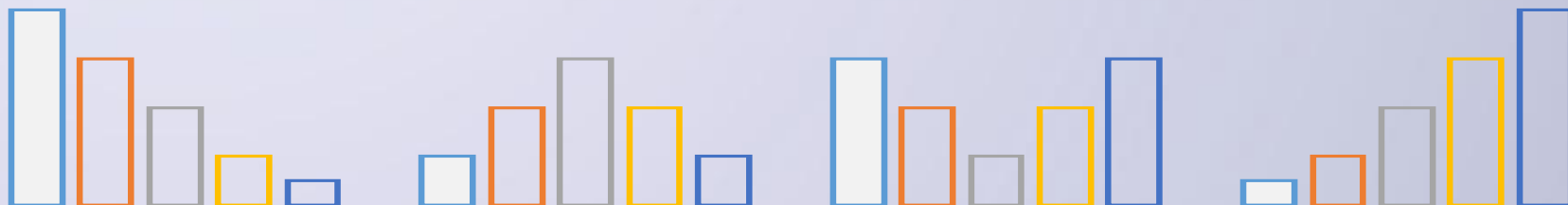
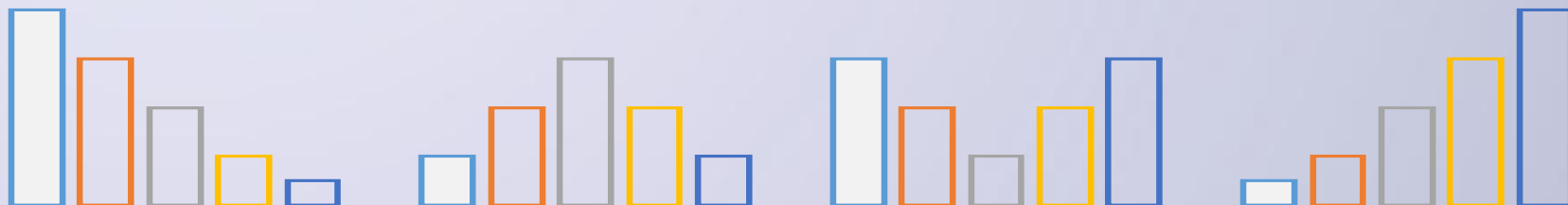


Python语言程序设计

北京理工大学 嵩天



第10 章 科学计算和可视化






计算不再关乎计算机，它与生活处处相关。

Computing is not about computers any more. It is
about living.

——尼古拉斯·尼葛洛庞帝(Nicholas Negroponte)

麻省理工学院媒体实验室的创办人



问题概述



- 科学计算需要采用矩阵运算库numpy和绘制库matplotlib



科学计算

- 科学计算是解决科学和工程中的数学问题利用计算机进行的数值计算，它不仅是科学家在运算自然规律时所使用的计算工具，更是普通人提升专业化程度的必要手段。
- Python 语言为开展人人都能使用的科学计算提供了有力支持。



科学计算

- 开展基本的科学计算需要两个步骤：组织数据和展示数据；
- 组织数据是运算的基础，也是将客观世界数字化的必要手段；
- 展示数据是体现运算结果的重要方式，也是展示结论的有力武器。



- 数学的矩阵 (Matrix) 是一个按照长方阵列排列的复数或实数集合，最早来自于方程组的系数及常数所构成的方阵。
- 矩阵是高等代数学中的常见工具，主要应用于统计数学、物理学、电路学、力学、光学、量子物理、计算机图像和动画等领域。



科学计算

- 科学计算领域最著名的计算平台Matlab 采用矩阵作为最基础的变量类型。矩阵有维度概念，一维矩阵是线性的，类似于列表，二维矩阵是表格状的，这是常用的数据表示形式。
- 科学计算与传统计算一个显著区别在于，科学计算以矩阵而不是单一数值为基础，增加了计算密度，能够表达更为复杂的数据运算逻辑。



拓展：离散和连续

- 矩阵是一个典型的离散变量类型，它将一些数据组织到一起。世界是连续的还是离散的呢？从人类观测角度，世界可以被解释成一个个离散的观测值；从微观角度，世界是原子不停运动的结果，应该是连续的；再微观到量子力学角度，任何连续运动都是最小粒子量子运动的结果，世界应该是离散的。宇宙真有最小粒子吗？
——世界是不确定的，还是确定的？世界是概率的，还是微积分的？
——醒醒，开始看程序！



思考与练习：

- [E10.1]思考在日常工作和生活中科学计算还有什么应用？
- [E10.2]尝试安装numpy 和matplotlib 库。



模块10 numpy 库的使用



- numpy 是用于处理含有同种元素的多维数组运算的第三方库。



numpy 库概述

- Python 标准库中提供了一个array 类型，用于保存数组类型数据，然而这个类型不支持多维数据，处理函数也不够丰富，不适合用于做数值运算。
- 因此，Python 语言的第三方库numpy 得到了迅速发展，至今，numpy 已经成为了科学计算事实上的标准库。



numpy 库概述

- numpy 库处理的最基础数据类型是由同种元素构成的多维数组（ndarray），简称“数组”。
- 数组中所有元素的类型必须相同，数组中元素可以用整数索引，序号从0开始。ndarray 类型的维度(dimensions)叫做轴(axes)，轴的个数叫做秩(rank)。一维数组的秩为1，二维数组的秩为2，二维数组相当于由两个一维数组构成。



numpy 库概述

- 由于numpy 库中函数较多且命名容易与常用命名混淆，建议采用如下方式引用numpy 库：

```
>>>import numpy as np
```

- 其中，as 保留字与import 一起使用能够改变后续代码中库的命名空间，有助于提高代码可读性。简单说，在程序的后续部分中，np 代替numpy。



numpy 库常用的创建数组函数

函数	描述
<code>np.array([x,y,z], dtype=int)</code>	从 Python 列表和元组创造数组
<code>np.arange(x,y,i)</code>	创建一个由 x 到 y，以 i 为步长的数组
<code>np.linspace(x,y,n)</code>	创建一个由 x 到 y，等分成 n 个元素的数组
<code>np.indices((m,n))</code>	创建一个 m 行 n 列的矩阵
<code>np.random.rand(m,n)</code>	创建一个 m 行 n 列的随机数组
<code>np.ones((m,n),dtype)</code>	创建一个 m 行 n 列全 1 的数组，dtype 是数据类型
<code>np.empty((m,n),dtype)</code>	创建一个 m 行 n 列全 0 的数组，dtype 是数据类型



ndarray 类的常用属性

- 创建一个简单的数组后，可以查看ndarray 类型有一些基本属性

属性	描述
<code>ndarray.ndim</code>	数组轴的个数，也被称作秩
<code>ndarray.shape</code>	数组在每个维度上大小的整数元组
<code>ndarray.size</code>	数组元素的总个数
<code>ndarray.dtype</code>	数组元素的数据类型， <code>dtype</code> 类型可以用于创建数组中
<code>ndarray.itemsize</code>	数组中每个元素的字节大小
<code>ndarray.data</code>	包含实际数组元素的缓冲区地址
<code>ndarray.flat</code>	数组元素的迭代器



ndarray 类的常用属性

```
>>>import numpy as np
>>>a = np.ones((4,5))
>>>print(a)
[[ 1.  1.  1.  1.  1.]
 [ 1.  1.  1.  1.  1.]
 [ 1.  1.  1.  1.  1.]
 [ 1.  1.  1.  1.  1.]]
>>>a.ndim
2
>>>a.shape
(4,5)
>>>a.dtype
dtype('float64')
```



ndarray 类的形态操作方法

方法	描述
<code>ndarray.reshape(n,m)</code>	不改变数组 <code>ndarray</code> ，返回一个维度为(n,m)的数组
<code>ndarray.resize(new_shape)</code>	与 <code>reshape()</code> 作用相同，直接修改数组 <code>ndarray</code>
<code>ndarray.swapaxes(ax1, ax2)</code>	将数组 <code>n</code> 个维度中任意两个维度进行调换
<code>ndarray.flatten()</code>	对数组进行降维，返回一个折叠后的一维数组
<code>ndarray.ravel()</code>	作用同 <code>np.flatten()</code> ，但是返回数组的一个视图



ndarray 类的形态操作方法

- 数组在numpy 中被当作对象，可以采用`<a>.()`方式调用一些方法。这里给出了改变数组基础形态的操作方法，例如改变和调换数组维度等。其中，`np.flatten()`函数用于数组降维，相当于平铺数组中数据，该功能在矩阵运算及图像处理中用处很大。



ndarray 类的索引和切片方法

方法	描述
<code>x[i]</code>	索引第 i 个元素
<code>x[-i]</code>	从后向前索引第 i 个元素
<code>x[n:m]</code>	默认步长为 1 ，从前往后索引，不包含 m
<code>x[-m:-n]</code>	默认步长为 1 ，从后往前索引，结束位置为 n
<code>x[n,m,i]</code>	指定 i 步长的由 n 到 m 的索引



ndarray 类的索引和切片方法

- 数组切片得到的是原始数组的视图，所有修改都会直接反映到源数组。如果需要得到的ndarray 切片的一份副本，需要进行复制操作，比如 `arange[5:8].copy()`



ndarray 类的索引和切片方法

```
>>>a = np.random.rand(5,3) #生成 5x3 的数组，用随机数填充
>>>a[2] #获得第 2 行数据
array([ 0.78426574,  0.60171943,  0.98825306])
>>>a[1:3]
array([[ 0.49276756,  0.44735929,  0.10356773],
       [ 0.78426574,  0.60171943,  0.98825306]])
>>>a[-5:-2:2]
array([[ 0.95517757,  0.3634953 ,  0.34138831],
       [ 0.78426574,  0.60171943,  0.98825306]])
```




numpy 库的算术运算函数

函数	描述
<code>np.add(x1, x2 [, y])</code>	$y = x1 + x2$
<code>np.subtract(x1, x2 [, y])</code>	$y = x1 - x2$
<code>np.multiply(x1, x2 [, y])</code>	$y = x1 * x2$
<code>np.divide(x1, x2 [, y])</code>	$y = x1 / x2$
<code>np.floor_divide(x1, x2 [, y])</code>	$y = x1 // x2$, 返回值取整
<code>np.negative(x [, y])</code>	$y = -x$
<code>np.power(x1, x2 [, y])</code>	$y = x1^{**}x2$
<code>np.remainder(x1, x2 [, y])</code>	$y = x1 \% x2$



numpy 库的算术运算函数

- 这些函数中，输出参数`y` 可选，如果没有指定，将创建并返回一个新的数组保存计算结果；如果指定参数，则将结果保存到参数中。例如，两个数组相加可以简单地写为`a+b`，而`np.add(a,b,a)`则表示`a+=b`。



numpy 库的比较运算函数

函数	符号描述
<code>np. equal(x1, x2 [, y])</code>	$y = x1 == x2$
<code>np. not_equal(x1, x2 [, y])</code>	$y = x1 != x2$
<code>np. less(x1, x2, [, y])</code>	$y = x1 < x2$
<code>np. less_equal(x1, x2, [, y])</code>	$y = x1 \leq x2$
<code>np. greater(x1, x2, [, y])</code>	$y = x1 > x2$
<code>np. greater_equal(x1, x2, [, y])</code>	$y = x1 \geq x2$
<code>np.where(condition[x,y])</code>	根据给出的条件判断输出 x 还是 y



numpy 库的比较运算函数

- 其将返回一个布尔数组，它包含两个数组中对应元素值的比较结果，例子如下。
- `where()`函数是三元表达式`x if condition else y` 的矢量版本。

```
>>>np.less([1,2],[2,2])  
array([ True, False], dtype=bool)
```



numpy 库的其他运算函数

函数	描述
<code>np.abs(x)</code>	计算基于元素的整形，浮点或复数的绝对值。
<code>np.sqrt(x)</code>	计算每个元素的平方根
<code>np.square(x)</code>	计算每个元素的平方
<code>np.sign(x)</code>	计算每个元素的符号：1(+), 0, -1(-)
<code>np.ceil(x)</code>	计算大于或等于每个元素的最小值
<code>np.floor(x)</code>	计算小于或等于每个元素的最大值
<code>np rint (x[, out])</code>	圆整,取每个元素为最近的整数,保留数据类型
<code>np.exp(x[, out])</code>	计算每个元素指数值
<code>np.log(x), np.log10(x), np.log2(x)</code>	计算自然对数(e),基于 10,2 的对数, $\log(1 + x)$



numpy 库

- numpy 库还包括三角运算函数、傅里叶变换、随机和概率分布、基本数值统计、位运算、矩阵运算等非常丰富的功能，读者在使用时可以到官方网站查询。



拓展：运算规则

- 实数的算术运算是最为常见的运算规则，类似的，矩阵也有算术运算。一个完备的运算体系包括运算基本单位和运算规则。在numpy 中，运算基本单位是数组，运算规则与实数一样，包括：算术运算、比较运算、统计运算、三角运算、随机运算等。numpy 库的广泛使用与完备的运算体系密切相关。



思考与练习：

- [E10.3]创建一个ndarray 变量有哪些方法？
- [E10.4]如何对ndarray 每个变量求平方根？
- [E10.5]思考ndarray 的降维是什么含义？



实例19 图像的手绘效果



要点

- 这是一个使用numpy 和PIL 库提取图像特征形成手绘效果的实例。



Python与手绘

- 第7章使用PIL 库获取了图像的轮廓，虽然提取了轮廓，但这个轮廓缺少立体感，视觉效果不够丰满



Python与手绘

- 光线照射使立体物出现明暗变化，运用这个原理是空间素描的基本方法，本节介绍采用Python 程序增加深浅层次变化，从而使图像轮廓更富立体感、空间感和色泽感，接近人类手绘效果。



图像的数组表示

- 图像是有规则的二维数据，可以用numpy 库将图像转换成数组对象

```
>>>from PIL import Image  
  
>>>import numpy as np  
  
>>>im = np.array(Image.open('D:\\pycodes\\fcity.jpg'))  
  
>>>print(im.shape, im.dtype)  
  
(881, 1266, 3) uint8
```



图像的数组表示

- 图像转换对应的ndarray 类型是3 维数据，如(881, 1266, 3)，其中，前两维表示图像的长度和宽度，单位是像素，第三维表示每个像素点的RGB 值，每个RGB 值是一个单字节整数。



像素处理

- PIL 库包括图像转换函数，能够改变图像单个像素的表示形式。使用`convert()`函数，这是' L' 模式，表示将像素表示从RGB 的3 字节形式转变为单一数值形式，这个数值范围在0 到255，表示灰度色彩变化



像素处理

- 此时，图像从彩色变为带有灰度的黑白色。转换后，图像的ndarray 类型变为二维数据，每个像素点色彩只由一个整数表示。

```
>>>im = np.array(Image.open('D:\\pycodes\\fcity.jpg').convert('L'))  
  
>>>print(im.shape, im.dtype)  
  
(881, 1266) uint8
```




像素处理

- 通过对图像的数组转换，可以利用numpy 访问图像上任意像素值，例如，获取访问位于坐标(20, 300)像素的颜色值或获取图像中最大和最小的像素值。也可以采用切片方式获取指定行或列的元素值，甚至修改这些值。



像素处理

```
>>>print(im[20,300])  
  
120  
  
>>>print(int(im.min()),int(im.max()))  
  
0 255  
  
>>>print(im[10,:])  
  
[114 113 115 ..., 123 122 123]
```



像素处理

- 将图像读入ndarray 数组对象后，可以通过任意数学操作来获取相应的图像变换。以灰度变换为例，分别对灰度变化后的图像进行反变换、区间变化和像素值平方处理。



像素处理

- 需要注意，有些数学变换会改变图像的数据类型，如变成整数型等，所以在重新生成PIL 图像前要先将数据类型通过`numpy.uint()`变换成整数

像素处理





拓展：灰度值

- 灰度值指黑白图像中点的颜色深度，范围从0 到255，白色为255，黑色为0，因此，黑白图像也被称为灰度图像。黑白图像主要用于构建非可见光图像，例如医学中超声波形成的图像等。RGB 彩色图片可以通过如下公式转换成灰度值：

$$\text{Gray} = R \times 0.3 + G \times 0.59 + B \times 0.11$$

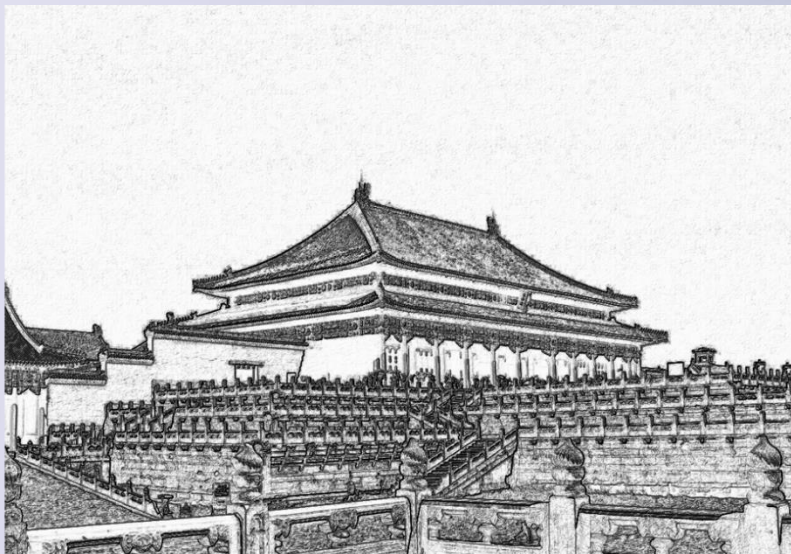
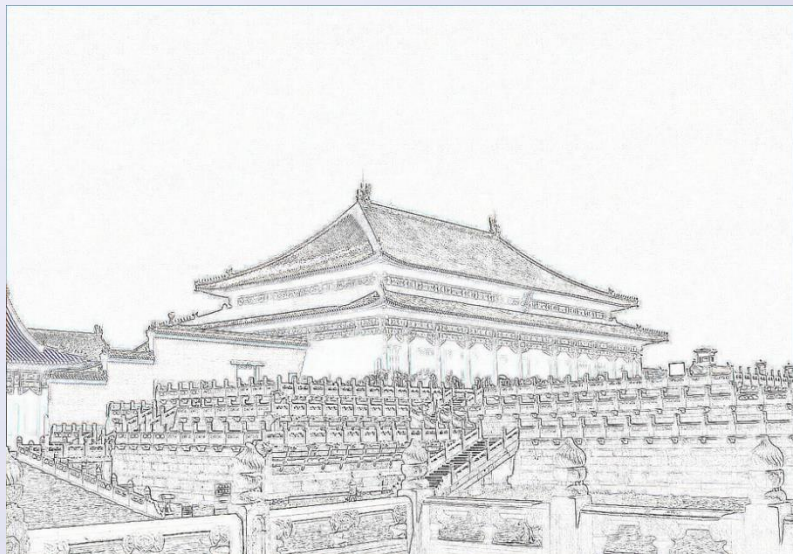
严格说，黑白图像是计算机计算能力或存储能力不充分时形成图像的重要方式，如果单个像素点能获得数据值种类超过256 且计算资源足够，采用彩色图像也可以构建非可见光谱，例如医学应用中新发展的彩色超声波成像等。



图像的手绘效果

- 之前介绍了10 种ImageFilter 类型的滤镜方法。通常获得铅笔画风格图像采用ImageFilter.CONTOUR 滤镜，它能够将图像的轮廓信息提取出来，原图像在视觉上更加的立体，获得的轮廓图像缺乏立体感。

图像轮廓和手绘效果的对比





图像的手绘效果

- 为了实现手绘风格，即黑白轮廓描绘，首先需要读取原图像的明暗变化，即灰度值。从直观视觉感受上定义，图像灰度值显著变化的地方就是梯度，它描述了图像灰度变化的强度。



图像的手绘效果

- 通常可以使用梯度计算来提取图像轮廓，numpy 中提供了直接获取灰度图像梯度的函数`gradient()`，传入图像数组表示即可返回代表x 和y 各自方向上梯度变化的二维元组



图像的手绘效果

```
1  #e19.1HandDrawPic.py
2  from PIL import Image
3  import numpy as np
4  vec_el = np.pi/2.2  # 光源的俯视角角度, 弧度值
5  vec_az = np.pi/4.   # 光源的方位角度, 弧度值
6  depth = 10.          # (0-100)
7  im = Image.open('fcity.jpg').convert('L')
8  a = np.asarray(im).astype('float')
9  grad = np.gradient(a)  #取图像灰度的梯度值
10 grad_x, grad_y = grad  #分别取横纵图像梯度值
11 grad_x = grad_x*depth/100.
12 grad_y = grad_y*depth/100.
13 dx = np.cos(vec_el)*np.cos(vec_az)  #光源对 x 轴的影响
14 dy = np.cos(vec_el)*np.sin(vec_az)  #光源对 y 轴的影响
```



图像的手绘效果

```
15  dz = np.sin(vec_e1)    #光源对 z 轴的影响
16  A = np.sqrt(grad_x**2 + grad_y**2 + 1.)
17  uni_x = grad_x/A
18  uni_y = grad_y/A
19  uni_z = 1./A
20  a2 = 255*(dx*uni_x + dy*uni_y + dz*uni_z) #光源归一化
21  a2 = a2.clip(0,255)
22  im2 = Image.fromarray(a2.astype('uint8')) #重构图像
23  im2.save('fcityHandDraw.jpg')
```



图像的手绘效果

- 手绘图像的基本思想是利用像素之间的梯度值（而不是像素本身）重构每个像素值。为了体现光照效果，设计一个光源，建立光源对各点梯度值的影响函数，进而运算出新的像素值，从而体现边界点灰度变化，形成手绘效果。



图像的手绘效果

- 具体来说，为了更好体现立体感，增加一个 z 方向梯度值，并给 x 和 y 方向梯度值赋权值 $depth$ 。这种坐标空间变化相当于给物体加上一个虚拟光源，根据灰度值大小模拟各部分相对于人视角的远近程度，使画面显得有“深度”。



图像的手绘效果

- 在利用梯度重构图像时，对应不同梯度取0-255 之间不同的灰度值，depth 的作用就在于调节这个对应关系。depth 较小时，背景区域接近白色，画面显示轮廓描绘；depth 较大时，整体画面灰度值较深，近似于浮雕效果



图像的手绘效果

- 将光源定义为三个参数：方位角`vec_az`、俯视角`vec_el` 和深度权值`depth`。两个角度的设定和单位向量构成了基础的柱坐标系，体现物体相对于虚拟光源的位置，如实例代码19.1 的第4 到6 行。



图像的手绘效果

- 通过`np.gradient()`函数计算图像梯度值作为新色彩计算的基础。为了更直观的进行计算，可以把角度对应的柱坐标转化为xyz 立体坐标系。`dx`、`dy`、`dz` 是像素点在施加模拟光源后在x、y、z 方向上明暗度变化的加权向量，如代码13 到15 行



图像的手绘效果

- A 是梯度幅值，也是梯度大小。各个方向上总梯度除以幅值得到每个像素单元的梯度值。利用每个单元的梯度值和方向加权向量合成灰度值，clip 函数用预防溢出，并归一化到0-255 区间。最后从数组中恢复图像并保存。



思考与练习：

- [E10.6]numpy 的ndarray 类型表示的彩色图像是几维？
- [E10.7]如何将彩色图片转换成灰度图片？之后如何处理每一个像素？
- [E10.8]哪个函数能将numpy 的ndarray 类型变成图像？



模块11 matplotlib库的使用



- matplotlib 是提供数据绘图功能的第三方库，其 pyplot 子库主要用于实现各种数据展示图形的绘制。



matplotlib.pyplot 库概述

- matplotlib.pyplot 是matplotlib 的子库，引用方式如下：

```
>>>import matplotlib.pyplot as plt
```

- 上述语句与import matplotlib.pyplot 一致，as 保留字与import 一起使用能够改变后续代码中库的命名空间，有助于提高代码可读性。简单说，在后续程序中，plt 将代替matplotlib.pyplot。



matplotlib.pyplot 库概述

- 为了正确显示中文字体，请用以下代码更改默认设置,其中'SimHei'表示黑体字。

```
>>>import matplotlib  
  
>>>matplotlib.rcParams['font.family']='SimHei'  
  
>>>matplotlib.rcParams['font.sans-serif'] = ['SimHei']
```



matplotlib.pyplot 库概述

- matplotlib 库由一系列有组织有隶属关系的对象构成，这对于基础绘图操作来说显得过于复杂。因此，matplotlib 提供了一套快捷命令式的绘图接口函数，即pyplot 子模块。pyplot 将绘图所需要的对象构建过程封装在函数中，对用户提供了更加友好的接口。pyplot 模块提供一批预定义的绘图函数，大多数函数可以从函数名辨别它的功能。



拓展：字体

- 字体是计算机显示字符的方式，均由人工设计，并采用字体库方式部署在计算机中。西文和中文字体都有很多种类，下表给出最常用的10 种中文字体及其英文表示，这些字体的英文表示在程序设计中十分常用，但需要注意，部分字体无法在matplotlib 库中使用。



拓展：字体

字体名称	字体英文表示
宋体	SimSun
黑体	SimHei
楷体	KaiTi
微软雅黑	Microsoft YaHei
隶书	LiSu
仿宋	FangSong
幼圆	YouYuan
华文宋体	STSong
华文黑体	STHeiti
苹果丽中黑	Apple LiGothic Medium



matplotlib.pyplot 库解析

- plt 子库提供了一批操作和绘图函数，每个函数代表对图像进行的一个操作，比如创建绘图区域、添加标注或者修改坐标轴等。
- 这些函数采用plt.()形式调用，其中是具体函数名称。



plt 库的绘图区域函数

函数	描述
<code>plt.figure(figsize=None, facecolor=None)</code>	创建一个全局绘图区域
<code>plt.axes(rect, axisbg='w')</code>	创建一个坐标系风格的子绘图区域
<code>plt.subplot(nrows, ncols, plot_number)</code>	在全局绘图区域中创建一个子绘图区域
<code>plt.subplots_adjust()</code>	调整子图区域的布局



plt 库的绘图区域函数

- 使用figure()函数创建一个全局绘图区域，并且使它成为当前的绘图对象，figsize参数可以指定绘图区域的宽度和高度，单位为英寸。鉴于figure()函数参数较多，这里采用指定参数名称的方式输入参数。

```
>>> plt.figure(figsize=(8,4))
```



plt 库的绘图区域函数

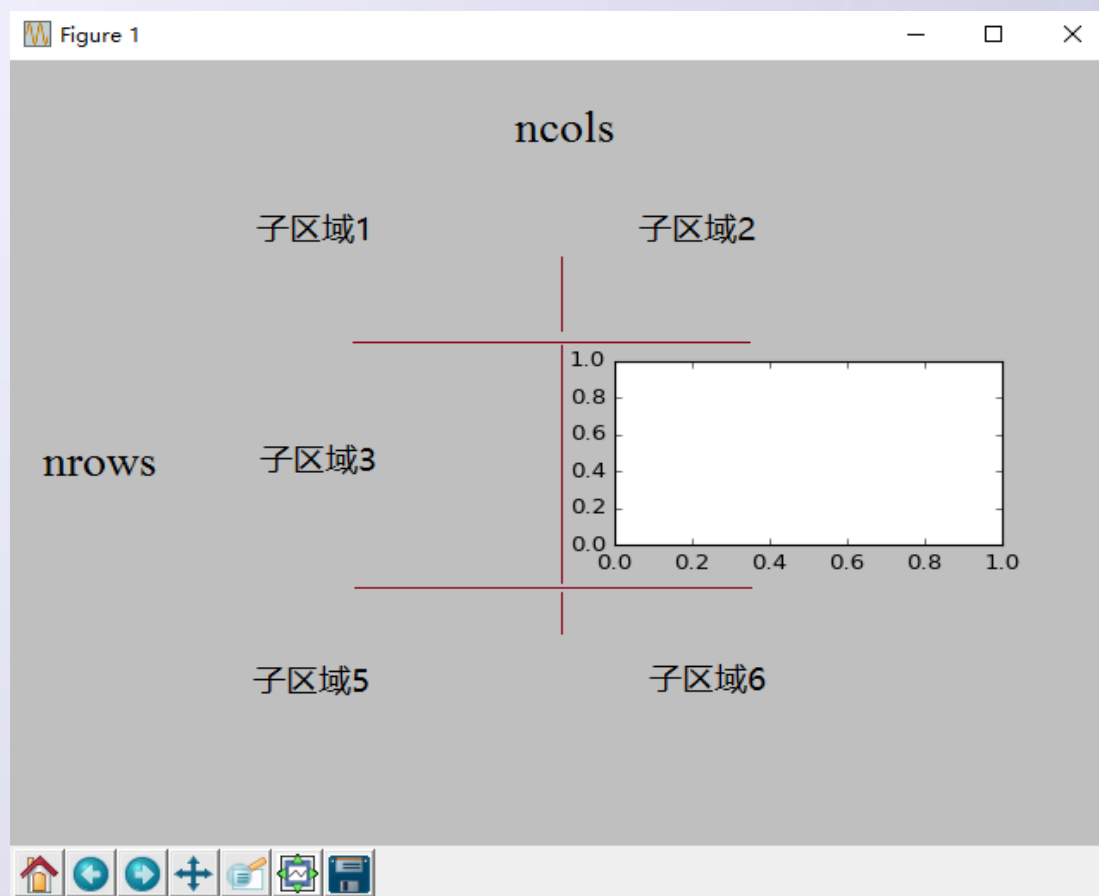
- `subplot()`都用于在全局绘图区域内创建子绘图区域，其参数表示将全局绘图区域分成`nrows` 行和`ncols` 列，并根据先行后列的计数方式在`plot_number` 位置生成一个坐标系，实例代码如下，三个参数关系如图10.3 所示。其中，全局绘图区域被分割成3x2 的网格，其中，在第4 个位置绘制了一个坐标系。

```
>>>plt.subplot(324)

>>>plt.show()
```



plt 库的绘图区域函数





plt 库的绘图区域函数

- `axes()` 默认创建一个 `subplot(111)` 坐标系，参数 `rect = [left, bottom, width, height]` 中四个变量的范围都为 `[0,1]`，表示坐标系与全局绘图区域的关系；
`axisbg` 指背景色，默认为 `white`。

```
>>> plt.axes([0.1, 0.1, 0.7, 0.3], axisbg='y')  
  
>>> plt.show()
```




plt 库的读取和显示函数

- plt 子库提供了一组读取和显示相关函数，用于在绘图区域中增加显示内容及读入数据，如下表所示，这些函数需要与其他函数搭配使用。



plt 库的读取和显示函数

函数	描述
<code>plt.legend()</code>	在绘图区域中方式绘图标签（也称图注）
<code>plt.show()</code>	显示创建的绘图对象
<code>plt.matshow()</code>	在窗口显示数组矩阵
<code>plt.imshow()</code>	在 axes 上显示图像
<code>plt.imsave()</code>	保存数组为图像文件
<code>plt.imread()</code>	从图像文件中读取数组



plt 库的基础图表函数

操作	描述
<code>plt.plot(x, y, label, color, width)</code>	根据 x, y 数组绘制直/曲线
<code>plt.boxplot(data, notch, position)</code>	绘制一个箱型图 (Box-plot)
<code>plt.bar(left, height, width, bottom)</code>	绘制一个条形图
<code>plt.barh(bottom, width, height, left)</code>	绘制一个横向条形图



plt 库的基础图表函数

<code>plt.polar(theta, r)</code>	绘制极坐标图
<code>plt.pie(data,explode)</code>	绘制饼图
<code>plt.psd(x, NFFT=256, pad_to, Fs)</code>	绘制功率谱密度图
<code>plt.specgram(x, NFFT=256, pad_to, F)</code>	绘制谱图
<code>plt.cohere (x, y, NFFT=256, Fs)</code>	绘制 X-Y 的相关性函数
<code>plt.scatter()</code>	绘制散点图 (x, y 是长度相同的序列)
<code>plt.step(x, y, where)</code>	绘制步阶图
<code>plt.hist(x, bins, normed)</code>	绘制直方图



plt 库的基础图表函数

<code>plt.contour(X, Y, Z, N)</code>	绘制等值线
<code>plt.vlines()</code>	绘制垂直线
<code>plt.stem(x, y, linefmt, markerfmt, basefmt)</code>	绘制曲线每个点到水平轴线的垂线
<code>plt.plot_date()</code>	绘制数据日期
<code>plt.plotfile()</code>	绘制数据后写入文件



plt 库的基础图表函数

- `plot()`函数是用于绘制直线的最基础函数，调用方式很灵活，`x`和`y` 可以是numpy计算出的数组，并用关键字参数指定各种属性。其中，`label` 表示设置标签并在图例(legend)中显示，`color` 表示曲线的颜色，`linewidth` 表示曲线的宽度。在字符串前后添加"\$"符号，`matplotlib` 会使用其内置的`latex` 引擎绘制的数学公式。

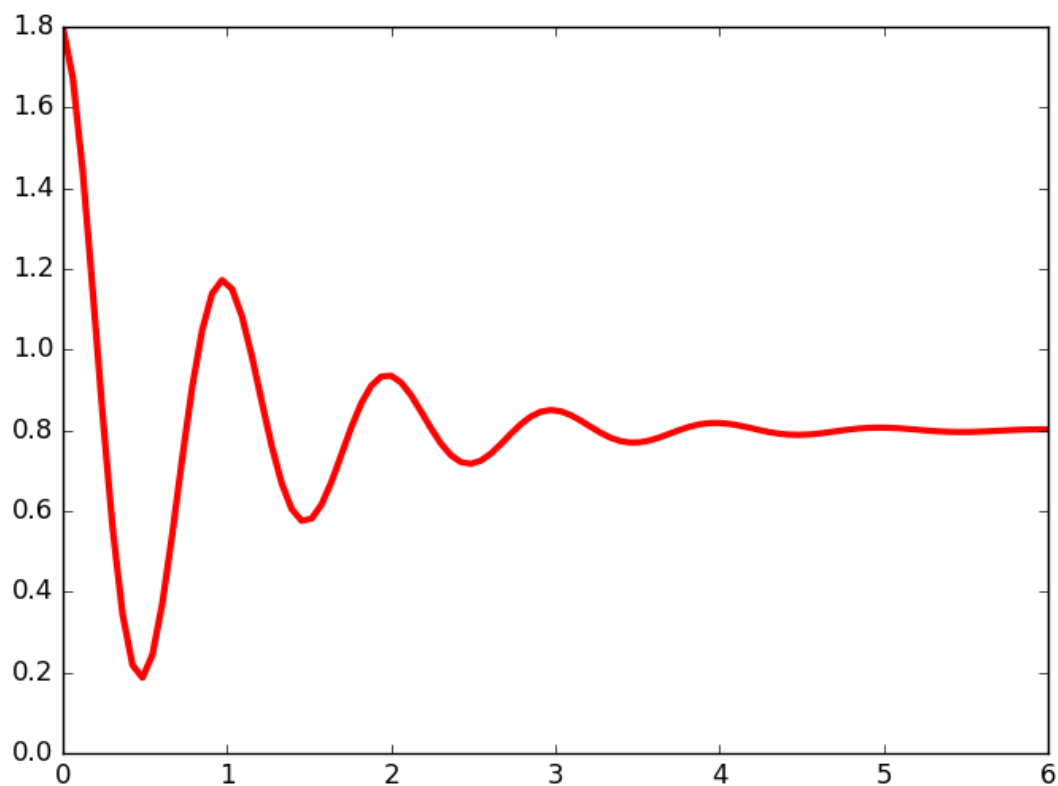


微实例10.1 绘制基本的三角函数

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 x = np.linspace(0, 6, 100)
4 y = np.cos(2 * np.pi * x) * np.exp(-x)+0.8
5 plt.plot(x, y, 'k', color='r', linewidth=3, linestyle="-")
6 plt.show()
```



微实例**10.1** 绘制基本的三角函数





plt 库的坐标轴

- plt 库有两个坐标体系；图像坐标和数据坐标。图像坐标将图像所在区域左下角视为原点，将x 方向和y 方向长度设定为1。整体绘图区域有一个图像坐标，每个axes()和subplot()函数产生的子图也有属于自己的图像坐标。axes()函数参数rect 指当前产生的子区域相对于整个绘图区域的图像坐标。数据坐标以当前绘图区域的坐标轴为参考，显示每个数据点的相对位置，这与坐标系里面标记数据点一直。



plt 库的坐标轴设置函数

函数	描述
<code>plt.axis('v','off','equal','scaled','tight','image')</code>	获取设置轴属性的快捷方法
<code>plt.xlim(xmin, xmax)</code>	设置当前 x 轴取值范围
<code>plt.ylim(ymin, ymax)</code>	设置当前 y 轴取值范围
<code>plt.xscale()</code>	设置 x 轴缩放
<code>plt.yscale()</code>	设置 y 轴缩放
<code>plt.autoscale()</code>	自动缩放轴视图的数据
<code>plt.thetagrids(angles, labels, fmt, frac)</code>	设置极坐标网格 <code>theta</code> 的位置
<code>plt.grid(on/off)</code>	打开或者关闭坐标网格



plt 库的坐标轴设置函数

```
>>>plt.plot([1,2,4], [1,2,3])
```

```
>>>plt.axis() #获得当前坐标轴范围
```

```
(1.0, 4.0, 1.0, 3.0)
```

```
>>>plt.axis([0,5,0,8]) #4个变量分别是[xmin,xmax,ymin,ymax]
```



plt 库的标签设置函数

函数	描述
<code>plt.figlegend(handles, label, loc)</code>	为全局绘图区域放置图注
<code>plt.legend()</code>	为当前坐标图放置图注
<code>plt.xlabel(s)</code>	设置当前 x 轴的标签
<code>plt.ylabel(s)</code>	设置当前 y 轴的标签
<code>plt.xticks(array, 'a', 'b', 'c')</code>	设置当前 x 轴刻度位置的标签和值
<code>plt.yticks(array, 'a', 'b', 'c')</code>	设置当前 y 轴刻度位置的标签和值
<code>plt.clabel(cs,v)</code>	为等值线图设置标签



plt 库的标签设置函数

<code>plt.get_figlabels()</code>	返回当前绘图区域的标签列表
<code>plt.figtext(x, y, s, fontdic)</code>	为全局绘图区域添加文字
<code>plt.title()</code>	设置标题
<code>plt.suptitle()</code>	为当前绘图区域添加中心标题
<code>plt.text(x, y, s, fontdic, withdash)</code>	为坐标图轴添加注释
<code>plt.annotate(note, xy, xytext, xycoords, textcoords, arrowprops)</code>	用箭头在指定数据点创建一个注释或一段文本



微实例10.2 带标签的坐标系

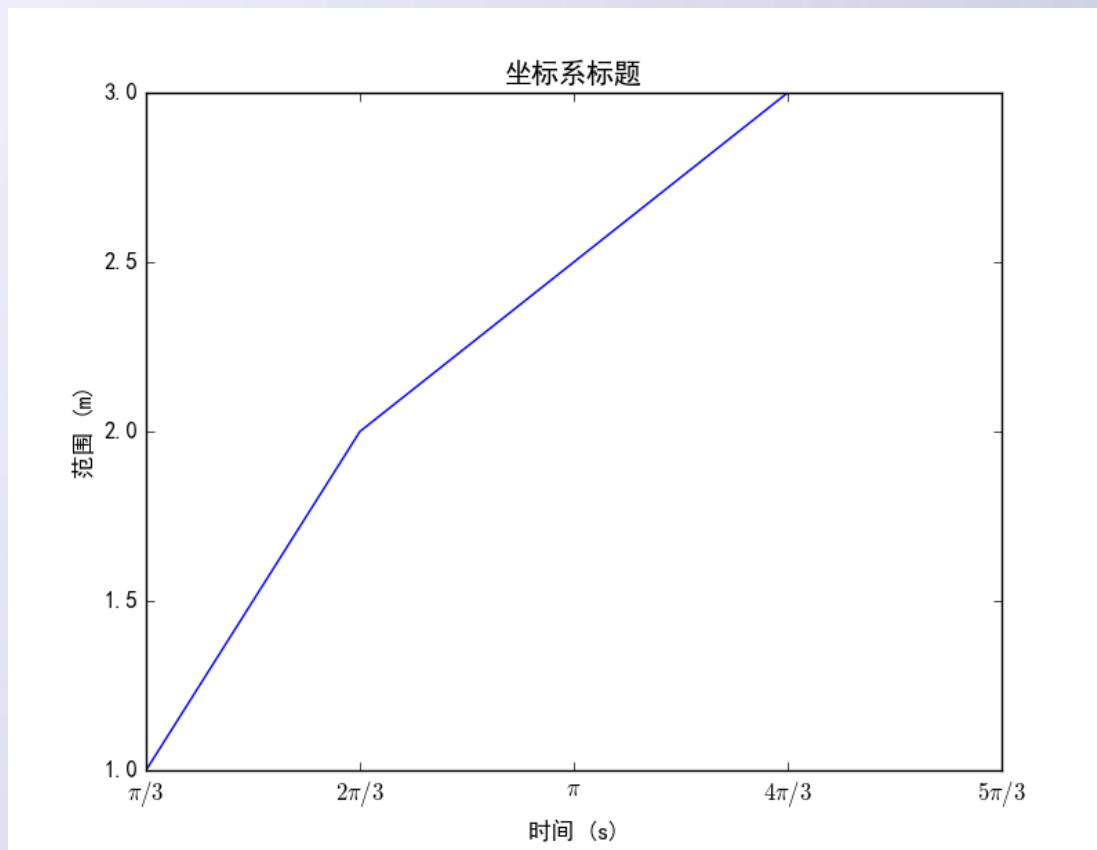
微实例 10.2

m10.2PlotCoordinate.py

```
1 import matplotlib.pyplot as plt
2 import matplotlib
3 matplotlib.rcParams['font.family']='SimHei'
4 matplotlib.rcParams['font.sans-serif'] = ['SimHei']
5 plt.plot([1,2,4], [1,2,3])
6 plt.title("坐标系标题")
7 plt.xlabel('时间 (s)')
8 plt.ylabel('范围 (m)')
9 plt.xticks([1,2,3,4,5],[r'$\pi/3$', r'$2\pi/3$', r'$\pi$', \
                        r'$4\pi/3$', r'$5\pi/3$'])
10 plt.show()
```



微实例10.2 带标签的坐标系





plt 库的区域填充函数

- plt 库提供了3 个区域填充函数，对绘图区域填充颜色

函数	描述
<code>fill(x,y,c,color)</code>	填充多边形
<code>fill_between(x,y1,y2,where,color)</code>	填充两条曲线围成的多边形
<code>fill_betweenx(y,x1,x2,where,hold)</code>	填充两条水平线之间的区域

微实例10.3 带局部阴影的坐标系

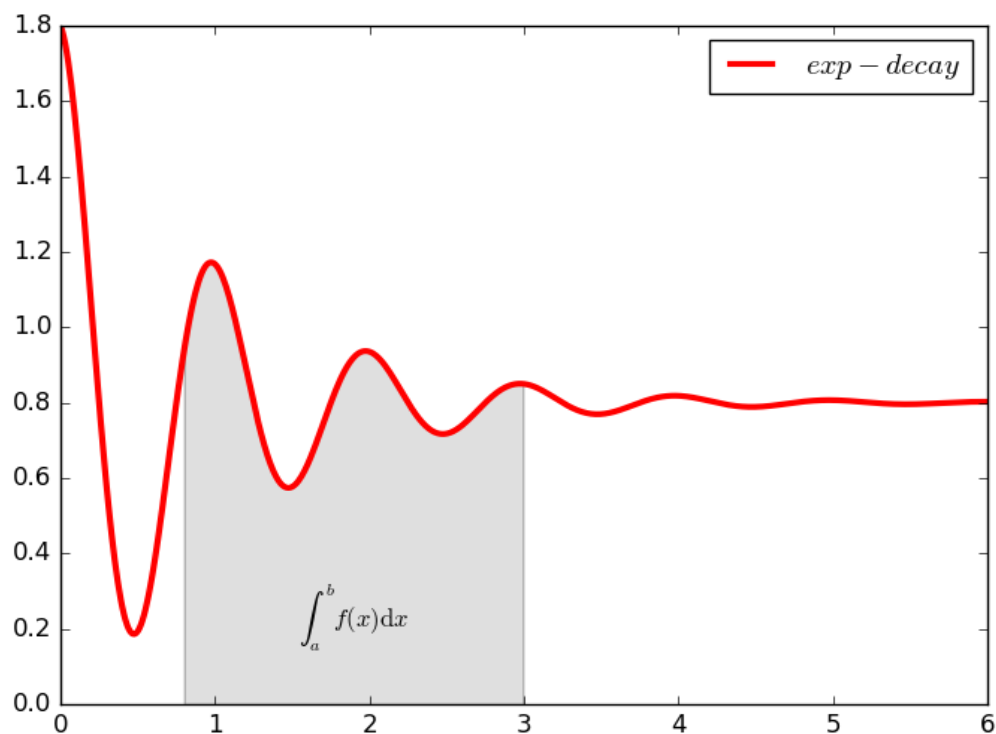
微实例 10.3

m10.3PlotDarkCoordinate.py

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 x = np.linspace(0, 10, 1000)
4 y = np.cos(2*np.pi*x) * np.exp(-x)+0.8
5 plt.plot(x,y, 'k', color='r', label="$exp-decay$", linewidth=3)
6 plt.axis([0,6,0,1.8])
7 ix = (x>0.8) & (x<3)
8 plt.fill_between(x, y ,0, where = ix,
                   facecolor='grey', alpha=0.25)
9 plt.text(0.5*(0.8+3), 0.2, r"$\int_a^b f(x)\mathrm{d}x$",
          horizontalalignment='center')
10 plt.legend()
11 plt.show()
```



微实例10.3 带局部阴影的坐标系





思考与练习：

- [E10.9]如何在一个绘制区域中回执上下两个坐标系？
- [E10.10]思考一个专业的科学坐标系都有哪些组成部分？
- [E10.11]绘制函数曲线的最精简代码是什么？



实例20 科学坐标图绘制



- 使用matplotlib.pyplot 子库绘制科学坐标图并适当标注。



科学坐标图绘制

- 采用坐标系绘制和展示数据趋势是经常使用的功能，掌握绘制专业的科学坐标系将会为生活和工作提供更高效的支持。
- 科学坐标图绘制有4 个要素：坐标轴、数据曲线、标题和图注。这个实例以阻尼衰减曲线绘制来具体阐述科学坐标系的绘制方法。

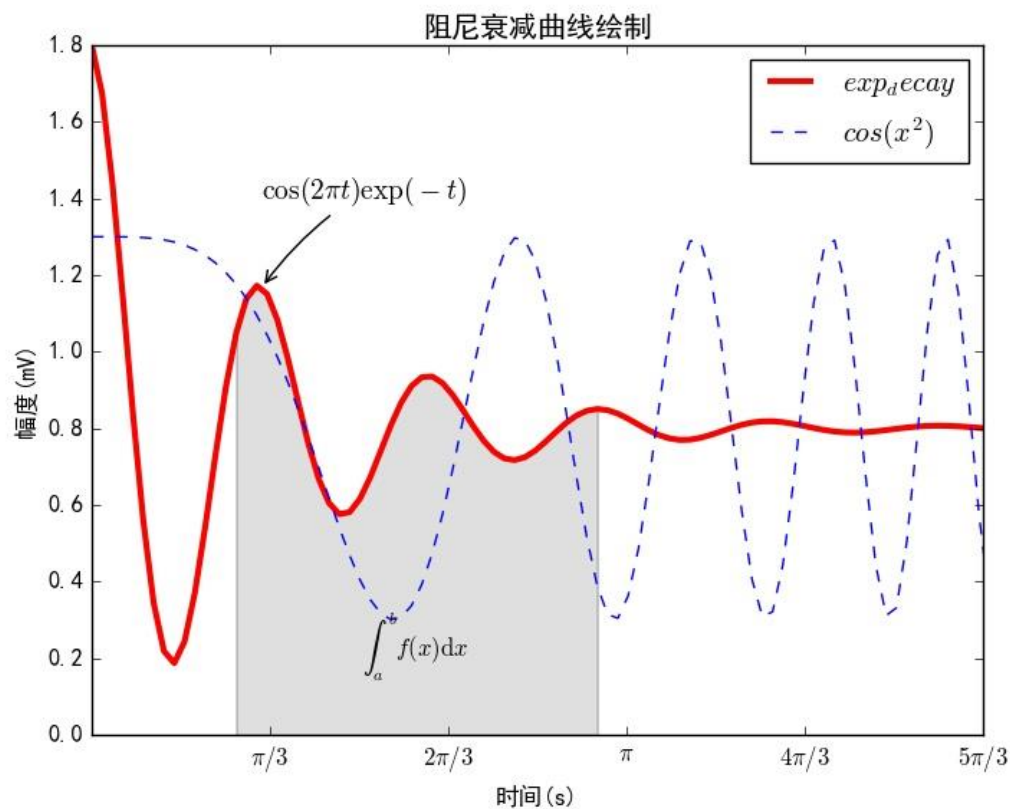


科学坐标图绘制

- 本实例同时展示了在同一个区域用不同颜色和线条绘制两种曲线的方法，两条曲线分别为 (x,y) 和 (x,z) 。

```
x = np.linspace(0.0, 6.0, 100)
y = np.cos(2 * np.pi * x) * np.exp(-x)+0.8
z = 0.5 * np.cos(x ** 2)+0.8
```

阻尼衰减曲线坐标图绘制





阻尼衰减曲线坐标图绘制

```
1  #e20.1PlotDamping.py
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import matplotlib
5  matplotlib.rcParams['font.family']='SimHei'
6  matplotlib.rcParams['font.sans-serif'] = ['SimHei']
```



阻尼衰减曲线坐标图绘制

```
7 def Draw(pcolor, nt_point, nt_text, nt_size):
8     plt.plot(x, y, 'k', label="$exp\_decay$", color=pcolor, \
               linewidth=3, linestyle="--")
9     plt.plot(x, z, "b--", label="$cos(x^2)$", linewidth=1)
10    plt.xlabel('时间(s)')
11    plt.ylabel('幅度(mV)')
12    plt.title("阻尼衰减曲线绘制")
13    plt.annotate('$\cos(2 \pi t) \exp(-t)$', xy=nt_point, \
                  xytext=nt_text, fontsize=nt_size, arrowprops= \
                  dict(arrowstyle='->', connectionstyle="arc3,rad=.1"))
```



阻尼衰减曲线坐标图绘制

```
14 def Shadow(a, b):
15     ix = (x>a) & (x<b)
16     plt.fill_between(x,y,0,where=ix,facecolor='grey',alpha=0.25)
17     plt.text(0.5 * (a + b), 0.2, r"$\int_a^b f(x) \mathrm{d}x$", \
              horizontalalignment='center')
18 def XY_Axis(x_start, x_end, y_start, y_end):
19     plt.xlim(x_start, x_end)
20     plt.ylim(y_start, y_end)
21     plt.xticks([np.pi/3, 2 * np.pi/3, 1 * np.pi, 4 * np.pi/3, \
                  5*np.pi/3], ['$\pi/3$', '$2\pi/3$', '$\pi$', '$4\pi/3$', '$5\pi/3$'])
```



阻尼衰减曲线坐标图绘制

```
22 x = np.linspace(0.0, 6.0, 100)
23 y = np.cos(2 * np.pi * x) * np.exp(-x)+0.8
24 z = 0.5 * np.cos(x ** 2)+0.8
25 note_point,note_text,note_size = (1, np.cos(2 * np.pi) *
26 np.exp(-1)+0.8), (1, 1.4), 14
27 fig = plt.figure(figsize=(8, 6), facecolor="white")
28 plt.subplot(111)
29 Draw("red", note_point, note_text, note_size)
30 XY_Axis(0, 5, 0, 1.8)
31 Shadow(0.8, 3)
32 plt.legend()
33 plt.savefig('sample.JPG')
34 plt.show()
```



阻尼衰减曲线坐标图绘制

- 为了便于阅读，程序设计了Draw()、Shadow()和XY_Axis()函数，分别用于绘制曲线、设置阴影和修改坐标轴。第13 行 annotate()函数配合箭头在曲线绘图界面添加动态注释，箭头的线条和尖端都有多种样式和参数，可以通过arrowprops 和 connectionstyle 选择，具体请参考官方文档。plt.savefig()函数能够将产生的坐标图保存为文件。



拓展：科学计算可视化

- 可视化技术与科学计算相结合形成了可视化技术的一个重要分支——科学计算可视化 (Visualization in Scientific Computing)。科学计算可视化将科学数据如测量获得的数值、图像或是计算产生的数字信息等以直观的、图形图像方式展示。通过直观展示，宇宙空间有了颜色、物理现象更为直观，感性理解和理性求证相辅相成共同促进科学计算的深入发展。



思考与练习：

- [E10.12]解释实例代码20.1 第13 行代码的含义。
- [E10.13]解释实例代码20.1 第32 行绘制图注的内容在哪里设定？
- [E10.14]解释实例代码20.1 绘制曲线的颜色和线形在哪里设定？



实例21 多级雷达图绘制



要点：

- 使用matplotlib.pyplot 绘制圆形和多边型雷达图，
通过绘制多属性数据展示对象间差异。



多级雷达图绘制

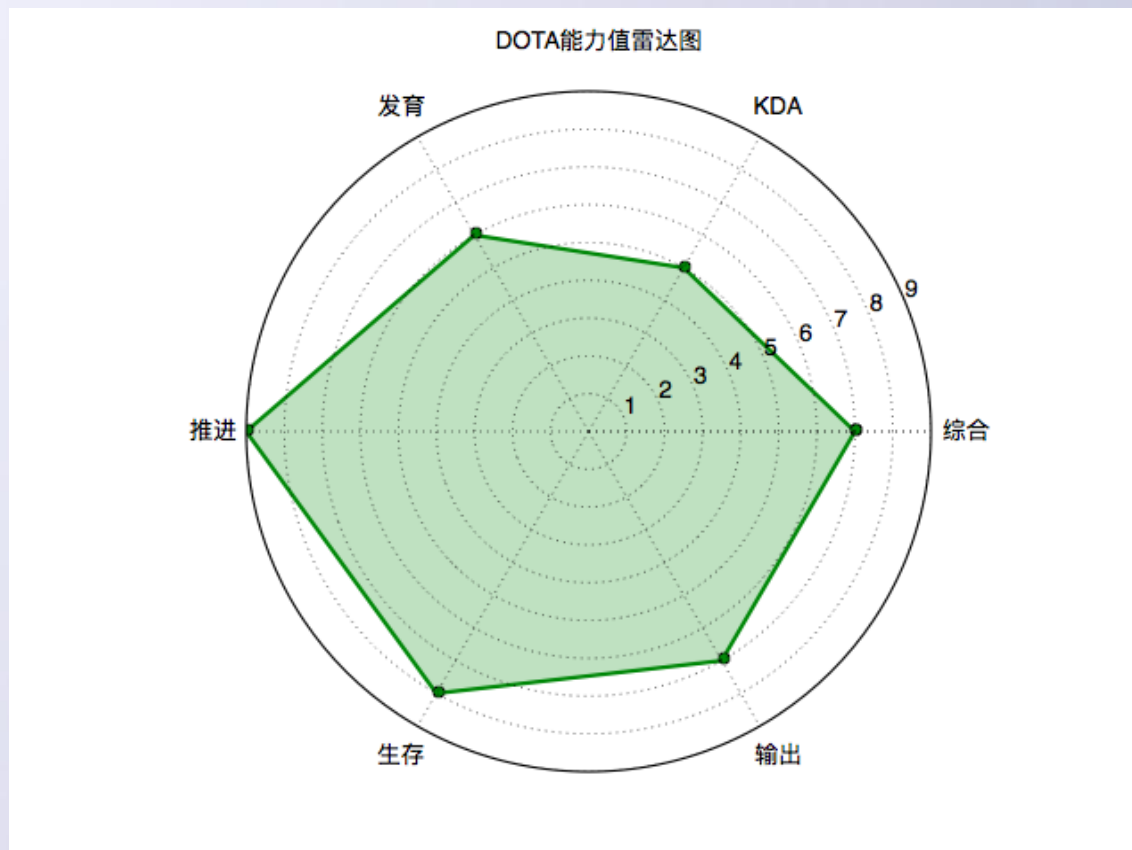
- 雷达图是通过多个离散属性比较对象的最直观工具，掌握绘制雷达图将会为生活和工作带来乐趣。

- 游戏角色中经常出现表示人物能力值的雷达图。

DOTAMAX 测试版曾经推出过显示玩家能力值分布的雷达图，只要点击自己或是好友头像，就可以看到能力值在综合、KDA、发育、推进、生存、输出等方面的能力分布



DOTA人物能力值雷达图





DOTA人物能力值雷达图

- 本实例使用Python 来绘制这样的多级雷达图，即在一组同心圆上填充不规则六边形，其每个顶点到圆心的距离代表人物某个属性的数据。

实例代码 21.1

e21.1DrawDotaRadar.py

```
1  #e21.1DrawDotaRadar.py
2  import numpy as np
3  import matplotlib.pyplot as plt
```



DOTA人物能力值雷达图

```
4 import matplotlib
5 matplotlib.rcParams['font.family']='SimHei'
6 matplotlib.rcParams['font.sans-serif'] = ['SimHei']
7 labels = np.array(['综合', 'KDA', '发育', '推进', '生存', '输出'])
8 nAttr = 6
9 data = np.array([7, 5, 6, 9, 8, 7]) #数据值
10 angles = np.linspace(0, 2*np.pi, nAttr, endpoint=False)
11 data = np.concatenate((data, [data[0]]))
12 angles = np.concatenate((angles, [angles[0]]))
13 fig = plt.figure(facecolor="white")
```



DOTA人物能力值雷达图

```
14 plt.subplot(111, polar=True)
15 plt.plot(angles,data,'bo-',color='g',linewidth=2)
16 plt.fill(angles,data,facecolor='g',alpha=0.25)
17 plt.thetagrids(angles*180/np.pi, labels)
18 plt.figtext(0.52, 0.95, 'DOTA 能力值雷达图', ha='center')
19 plt.grid(True)
20 plt.savefig('dota_radar.JPG')
21 plt.show()
```



DOTA人物能力值雷达图

- 实例代码21.1 中第4 到6 行用于支持中文。
- 由于DOTA 实例包含6 个属性，设置属性标签labels，并预设一组玩家数据data。



DOTA人物能力值雷达图

- `np.linspace()`函数设定起点为0、末值为 2π 、返回一个两端点间数值平均分布的长为`nAttr` 的数组 `angles` , 它表示从一个属性点到下一个属性点笔画需要旋转的角度, 它取决于属性`nAttr` 的大小, 也是雷达图的多边形边数。



DOTA人物能力值雷达图

- `np.concatenate()`函数用于将数据和角度的数组首尾闭合起来，便于调用`plot()`函数绘制。



多级雷达图绘制

- 建立基本绘图对象后，使用subplot()函数建立极坐标系的子分区。Polar 参数指定了绘制类型为极坐标，这是subplot()除默认正方形坐标系外唯一支持的内置坐标图。



多级雷达图绘制

- 建立极坐标后，使用`plot()`函数依照`data` 提供的数据画出不规则六边形，然后使用`fill()`函数填充半透明颜色。`thetagrids()`函数为极坐标设置标签，这里把标签安放在六角形的顶点上，需要将角度数据和文字一起作为参数传给`thetagrids` 函数。



多级雷达图绘制

- 除了DOTA 游戏，雷达图应用广泛。美国约翰霍普金斯大学霍兰德教授认为兴趣是人们活动的巨大动力，凡是具有职业兴趣的职业，都可以提高人们的积极性，促使人们积极地、愉快地从事该职业。因此，他研究了人格类型、兴趣与职业间的关系，提出了“霍兰德职业兴趣理论”，认为人格可分为现实型、研究型、艺术型、社会型、企业型和常规型等六种类型。



拓展：兴趣是最好的老师

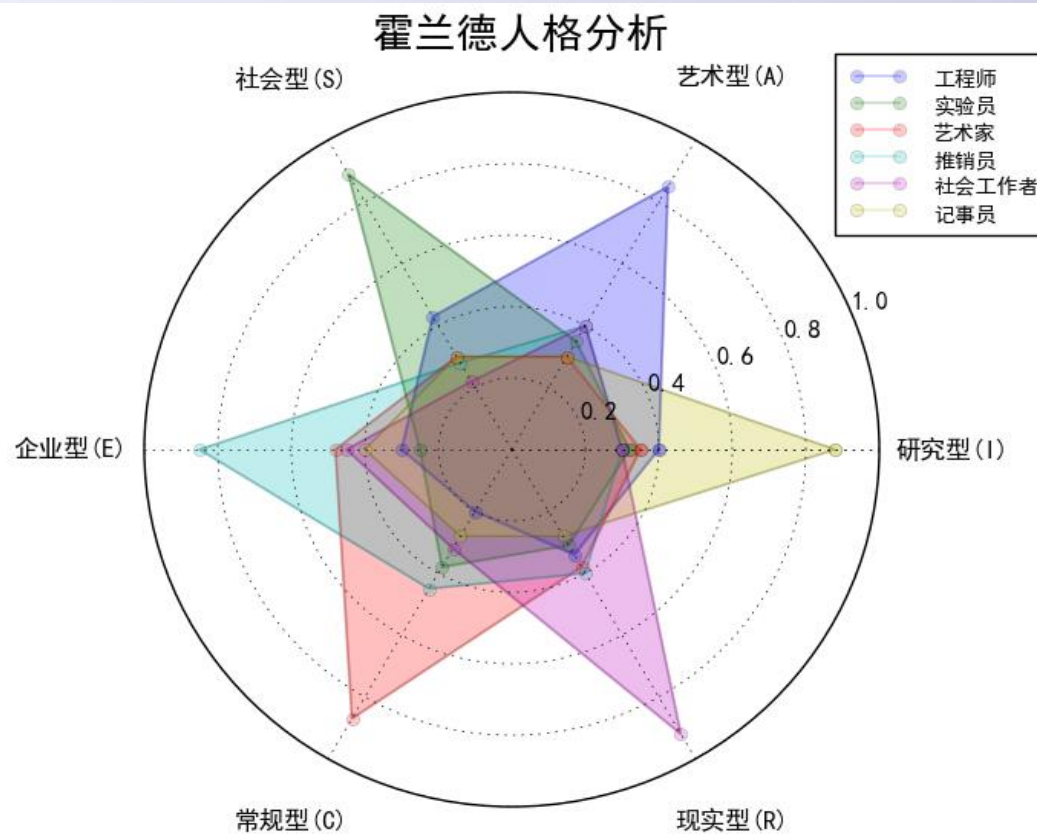
- 爱因斯坦说过，“兴趣是最好的老师”。兴趣来源于好奇，好奇心则是人类与生俱来的。如果读者对本书设计的实例感到好奇，这说明，您已经找到了最好的老师。全书共25 个实例，绝大部分为作者结合实际教学效果原创制作，它们将为读者带来学习 Python 语言的最佳体验。



多级雷达图绘制

- 展示霍兰德人格分析最有效的工具是雷达图。以工程师、实验员、艺术家、推销员、社会工作者、记事员等6 个职业数据为例，给出了霍兰德人格分析绘制的雷达图，

霍兰德人格分析雷达图





霍兰德人格分析雷达图

```
8  nAttr = 6
9  data = np.array([[0.40, 0.32, 0.35, 0.30, 0.30, 0.88],
10                  [0.85, 0.35, 0.30, 0.40, 0.40, 0.30],
11                  [0.43, 0.89, 0.30, 0.28, 0.22, 0.30],
12                  [0.30, 0.25, 0.48, 0.85, 0.45, 0.40],
13                  [0.20, 0.38, 0.87, 0.45, 0.32, 0.28],
14                  [0.34, 0.31, 0.38, 0.40, 0.92, 0.28]]) #数据值
15  data_labels=('工程师','实验员','艺术家','推销员','社会工作者','记事员')
16  angles = np.linspace(0, 2*np.pi, nAttr, endpoint=False)
17  data = np.concatenate((data, [data[0]]))
18  angles = np.concatenate((angles, [angles[0]]))
```




霍兰德人格分析雷达图

```
19 fig = plt.figure(facecolor="white")
20 plt.subplot(111, polar=True)
21 #plt.plot(angles,data,'bo-',color
22 ='gray',linewidth=1,alpha=0.2)
23 plt.plot(angles,data,'o-', linewidth=1.5, alpha=0.2)
24 plt.fill(angles,data, alpha=0.25)
25 plt.thetagrids(angles*180/np.pi, radar_labels,frac = 1.2)
26 plt.figtext(0.52, 0.95, '霍兰德人格分析', ha='center', size=20)
27 legend=plt.legend(data_labels,loc=(0.94,0.80),labelspacing=0.1)
28 plt.setp(legend.get_texts(), fontsize='small')
29 plt.grid(True)
30 plt.savefig('holland_radar.JPG')
31 plt.show()
```



思考与练习：

- [E10.15]解释实例代码21.1 第14 行代码的含义。
- [E10.16]解释实例代码21.1 第18 行代码的含义。
- [E10.17]解释实例代码21.2 第27 行代码的含义。



本章小结



本章小结

- 本章以科学计算和可视化为中心，介绍了两个强大的工具库numpy 和matplotlib.pyplot，通过生成手绘风格图片、绘制科学坐标系和绘制雷达图等实例展示了科学计算的强大功能。



程序练习题

- [P10.1] 在信号处理理论中，方波可近似表示为多个正弦波的叠加。事实上，任意一个方波信号都可以使用傅里叶变换为多个正弦波表示。利用numpy 和matplotlib在坐标系中绘制方波的无穷级数表示。请尝试调节正弦波的个数、幅度以及周期，尽可能使方波边缘平滑。方波无穷级数表达式

$$f = \sum_{k=1}^{\infty} \frac{4\sin(2k-1)t}{(2k-1)\pi}。$$



程序练习题

- [P10.2] 笛卡尔心形线也称为心脏线，它是有一个尖点的外摆线。当一个圆沿着另一个半径相同的圆滚动时，圆上一点的轨迹就是心脏线。请调研笛卡尔心形线，并使用numpy 和matplotlib 绘制一幅笛卡尔心形线。



程序练习题

- [P10.3] 修改实例19，使手绘效果更符合您的审美特点。
- [P10.4] 参考实例20，绘制读者感兴趣的一个数学或物理规律。
- [P10.5] 参考实例21，为中国乒乓球选手绘制雷达图，至少建立4个以上属性值。