化工应用数学 第二章 python 编程基础 讲义

（03.Python 库简介）

Python 中的库：

程序库(library),一个可供使用的各种标准程序、子程序、文件以及它们的目录等信息的有序集合。 汇集在一起的经常应用的程序。针对不同的需求，python 提供了不同的库函数，甚至是针对于统一需求，python 也会有很多不同的库来实现。

Python 安装这些外部库一般使用“$ pip install 模块名”就可以了，比如：“$ pip install numpy”（Python2+版本）或者“$ pip3 install numpy”（Python3+版本）。

而更新相关外部库则在上述命令中加上-U 即可，比如：“$ pip install -U numpy”（Python2+版本）或者“$ pip3 install -U numpy”（Python3+版本）

本课程所涉及到的主要是 numpy、scipy 和 matplotlib，相关链接如下：

NumPy 官网：http://www.numpy.org/

NumPy 源代码：https://github.com/numpy/numpy

SciPy 官网：https://www.scipy.org/

SciPy 源代码：https://github.com/scipy/scipy

Matplotlib 官网：https://matplotlib.org/

Matplotlib 源代码：https://github.com/matplotlib/matplotlib

Git&Github：

Git 是一个版本控制系统（Version Control System，VCS）。版本控制是一种记录一个或若干文件内容变化，以便将来查阅特定版本修订情况的系统。有了版本控制系统，就可以不用担心文件丢失，不小心误修改文件等等“事故”，而且你可以随便回到历史记录的某个时刻。

SVN, CVS 这类早期的集中式版本控制系统，都有一个单一的集中管理的服务器，保存所有文件的修订版本，而协同工作的人们都通过客户端连到这台服务器，取出最新的文件或者提交更新。

而 Git 这类分布式版本控制系统, 才是现代的首选。因为分布式的优势绝对显著。在分布式版本控制系统里，客户端并不只提取最新版本的文件快照，而是把代码仓库完整地镜像下来。这么一来，任何一处协同工作用的服务器发生故障，事后都可以用任何一个镜像出来的本地仓库恢复。因为每一次的提取操作，实际上都是一次对代码仓库的完整备份。

Github 和 Git 是两回事。Git 是版本控制系统，Github 是在线的基于 Git 的代码托管服务。平台上有很多不同功能、不同语言的开源程序。

numpy：

NumPy(Numerical Python) 是 Python 语言的一个扩展程序库，支持大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。

NumPy 是一个运行速度非常快的数学库，主要用于数组计算，包含：一个强大的 N 维数组对象 ndarray、广播功能函数、整合 C/C++/Fortran 代码的工具以及线性代数、傅里叶变换、随机数生成等功能。

使用方式：在 python 中安装好之后，在文件开头使用“import numpy as np”来声明，这样之后在程序中可以使用 np 代表 numpy 来使用 numpy 中的各种函数和方法。

创建数组：np.array()

```
import numpy as np

a=np.array([[1,2,3],[4,5,6]])
print(a)
```

上述程序将生成对应的二行三列的矩阵。

```
>>> a=[[1, 2, 3], [4, 5, 6]]
>>> print(a)
[[1, 2, 3], [4, 5, 6]]
```

注意通过 numpy 产生的是一种特殊形式的对象 ndarray，虽然其使用等与上述一般数组类似，但是 numpy 给这种对象提供了一些特殊的方法，更加适合于矩阵的运算。

同时，numpy 创建时可以同时制定数据类型：

```
import numpy as np

a=np.array([[1,2,3],[4,5,6]],dtype=complex)
print(a)
```

此时，结果显示为：

```
[[1.+0.j 2.+0.j 3.+0.j]
 [4.+0.j 5.+0.j 6.+0.j]]
```

零数组与一数组：np.zeros()和 np.ones()

```
>>> np.zeros((3, 2))
array([[0., 0.],
       [0., 0.],
       [0., 0.]])
```

```
>>> np.ones((3, 2))
array([[1., 1.],
       [1., 1.],
       [1., 1.]])
```

在 numpy 中，对于 ndarray 类型矩阵的加减乘除等数学运算进行了重新定义，见下图：

```
>>> import numpy as np
>>> a=np.array([[1,2,3],[4,5,6]])
>>> b=np.ones((2,3))
>>> c=b*2
>>> c
array([[2., 2., 2.],
       [2., 2., 2.]])
>>> d=a-b
>>> d
array([[0., 1., 2.],
       [3., 4., 5.]])
>>> e=a*c
>>> e
array([[ 2.,  4.,  6.],
       [ 8., 10., 12.]])
>>> f=a/c
>>> f
array([[0.5, 1. , 1.5],
       [2. , 2.5, 3. ]])
>>> a**2
array([[ 1,  4,  9],
       [16, 25, 36]], dtype=int32)
>>> np.sin(a)
array([[ 0.84147098,  0.90929743,  0.14112001],
       [-0.7568025 , -0.95892427, -0.2794155 ]])
>>> a<4
array([[ True,  True,  True],
       [False, False, False]])
```

矩阵乘法：使用 dot 函数或者@运算符（3.5 版本之后）

```
>>> import numpy as np
>>> a=np.array([[1,2],[3,4]])
>>> b=np.array([[5,6],[7,8]])
>>> a*b
array([[ 5, 12],
       [21, 32]])
>>> a.dot(b)
array([[19, 22],
       [43, 50]])
>>> a@b
array([[19, 22],
       [43, 50]])
```

Matplotlib：

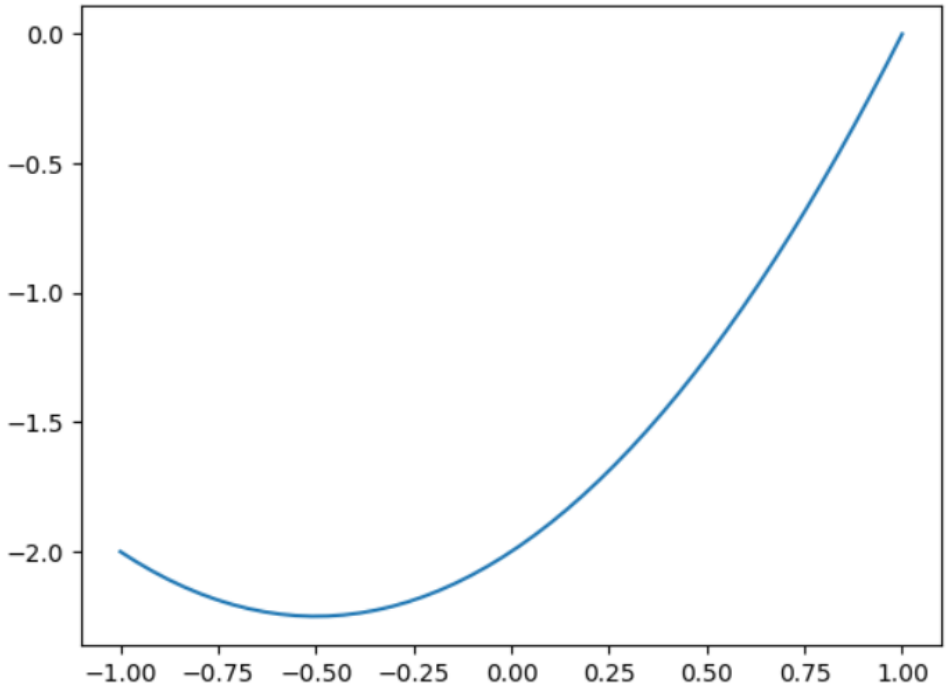
Matplotlib 是一个非常强大的 Python 画图工具，它能够使我们更好的展现所得到的数据。

Matplotlib 能够生成多种不同类型的图像，包括：线图、散点图、等高线图、条形图、柱状图、3D 图形，甚至图形动画等等。

使用 matplotlib 过程中，使用 import 导入模块 matplotlib.pyplot，并简写成 plt，具体程序见下图：

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

x=np.linspace(-1,1,50)
y=x**2+x-2

plt.figure()
plt.plot(x,y)
plt.show()
```

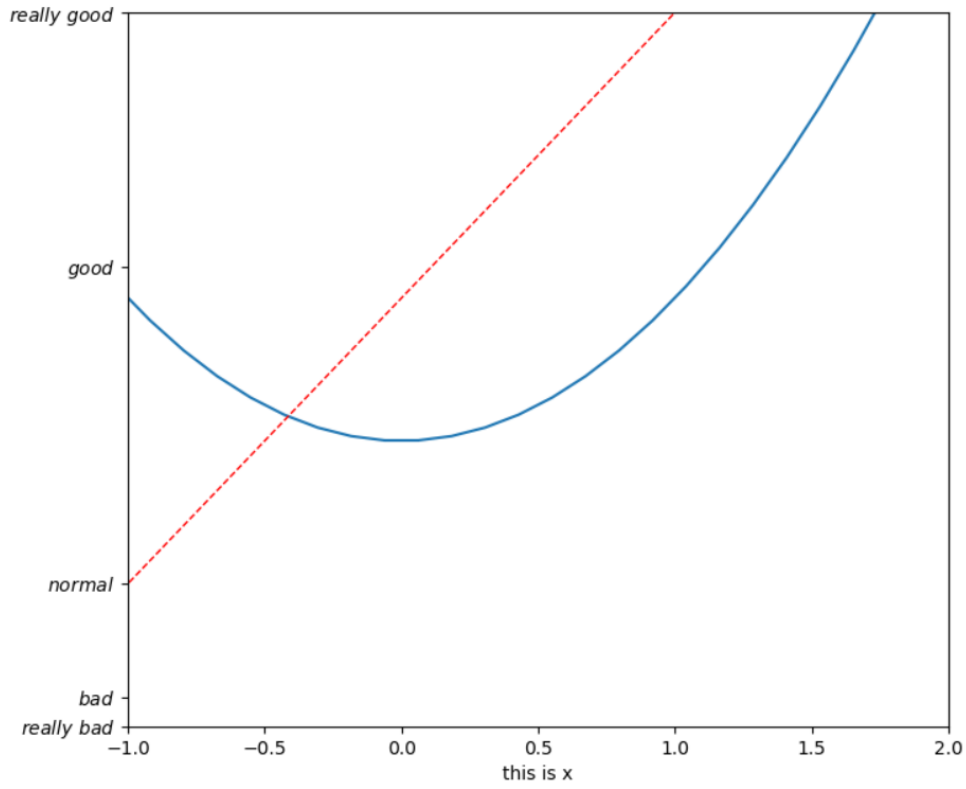
生成图片如下：



当我们需要将多组数据放到同一个图中，并且需要对于坐标轴进行一定的调整时，可以参考下面的实例代码：

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

x=np.linspace(-3,3,50)
y1=x**2
y2=2*x+1

plt.figure(num=3,figsize=(8,7),)
plt.plot(x,y1)
plt.plot(x,y2,color='red',linewidth=1.0,linestyle='--')
plt.xlim((-1,2))
plt.ylim((-2,3))
plt.xlabel('this is x')
plt.yticks([-2, -1.8, -1, 1.22, 3],\
           [r'$really\ bad$', r'$bad$', r'$normal$', r'$good$', r'$really\ good$'])
plt.show()
```

最后的结果如下：

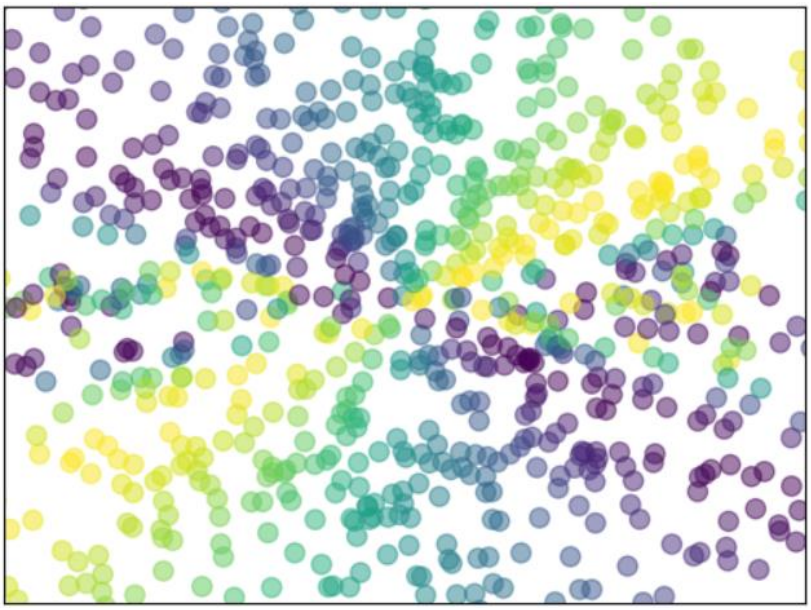


散点图：

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

n = 1024    # data size
X = np.random.normal(0, 1, n)
Y = np.random.normal(0, 1, n)
T = np.sin(X/Y)

plt.scatter(X, Y, s=75, c=T, alpha=0.5)

plt.xlim(-1.5, 1.5)
plt.xticks(())
plt.ylim(-1.5, 1.5)
plt.yticks(())

plt.show()
```

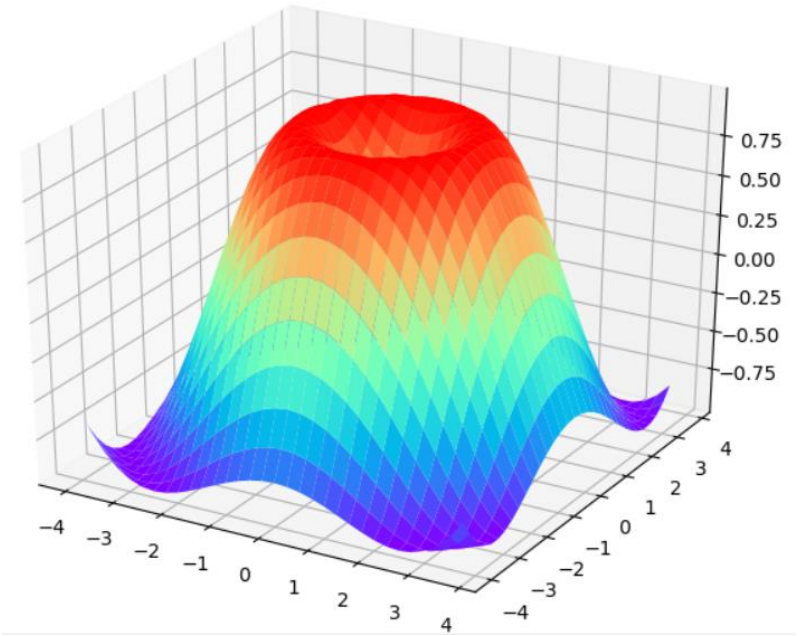
基本语法：.scatter(xData,yData,s=scatterSize,c=color,alpha=alphaData)

最终结果如下：



简单的 3D 图案例：

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure()
ax = Axes3D(fig)

X = np.arange(-4, 4, 0.25)
Y = np.arange(-4, 4, 0.25)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X ** 2 + Y ** 2)
Z = np.sin(R)

ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=plt.get_cmap('rainbow'))
plt.show()
```



作业1.1：

- 针对图中创建的数据x、y1、y2，利用matplotlib在一张图中画出y1和y2的图像，要求：
  - y1为曲线图，曲线为线宽2.0的虚线
  - y2为红色散点图，散点大小100

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

x = np.linspace(-math.pi,math.pi,50)
y1 = np.sin(x)
y2 = np.sin(x)+0.2*np.random.randn(50)
```

- 提交方式：
  - 源代码保存为以“学号-姓名-作业1.1.py”命名的文件
  - 页面https://sanliwuxun.github.io/，左侧点击“化工应用数学”-“作业1”，在跳转页面中提交

15

作业1.2：

- 编写函数checkPrime(n)，要求：
  - 参数n为正整数
  - 如果n为质数，函数返回值为True；否则返回值为False

```
def checkPrime(n):
    pass
```

- 提交方式：
  - 源代码保存为以“学号-姓名-作业1.2.py”命名的文件
  - 页面https://sanliwuxun.github.io/，左侧点击“化工应用数学”-“作业1”，在跳转页面中提交

16