实验课 07

实验7-1 Linux基本操作（下）

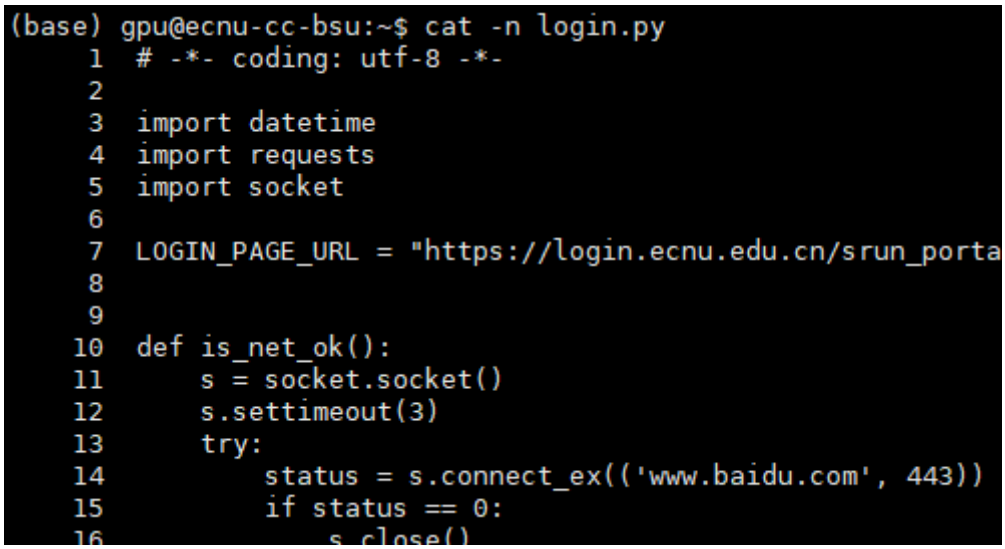
- `cat`命令

使用`cat`命令可以完整查看一个文件内的所有内容，例如

> `cat test.txt` 可以查看test.txt文件的所有内容

有的时候我们查看代码文件，最好像在IDE那样在屏幕左侧显示行号，可以帮助我们快速定位代码位置。可以在`cat`命令后加`-n`参数，例如

> `cat -n login.py` 可以在查看文件内容的左侧显示行号（空白行也会标注行号）

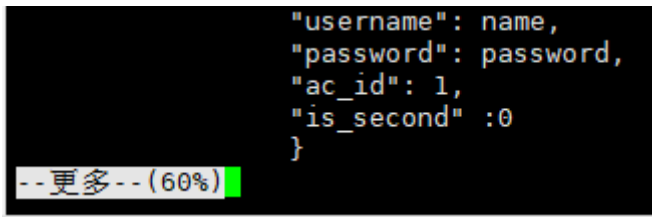


- `more`命令

有的文件很长很长，如果使用`cat`命令一是加载缓慢、二是如果我们寻找的文件内容在文件起始位置，那么我们还没来得及看到内容就被后面的内容覆盖了……这时候我们需要使用`more`命令，让文件内容先显示一屏，当我们按下键盘按键时再显示下一屏。例如

> `more login.py`

此时屏幕会暂时显示第一屏的内容，而且在最下方会出现“--More--(60%)”或者“--更多--(60%)”字样，代表当前文件内容还未显示完，已经显示60%的内容。



现在我们可以按下`Enter`键，此时内容会往下显示一行；

如果按下空格键，会显示下一屏的内容。

如果想退出查看，可以按下`q`键，会立刻退出查看器。

- `head`命令

如果我们只想查看某个文件的前几行，我们可以使用`head`命令快速查看，例如

> `head -n 7 login.py`可以显示文件的前7行内容
>
> `head login.py`如果不加`-n`参数，默认显示文件的前10行内容

实验7-2 NumPy库初步

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。

步骤1 安装NumPy

我们在实验课01中已经配置好了编程环境，由于NumPy库不是Python自带的库，需要我们手动安装。幸好Anaconda提供了自动安装工具可以简单快速地安装第三方库。现在我们先安装NumPy库。

在Anaconda PowerShell Prompt命令行中切换到自己的环境（如有），然后输入`conda install numpy`，根据提示（若有）按下键盘上的`y`键确认安装，conda会自动下载安装包并安装。

步骤2 创建数组

- **创建空数组**

`empty`方法用来创建一个指定形状（shape）、数据类型（dtype）且未初始化的数组。

在使用NumPy的任何方法之前别忘了导入NumPy库。

```
import numpy as np  # 通常习惯以np代指NumPy，后续调用时无需再输入过多前缀
arr1 = np.empty((4,3),dtype=float)  # 创建一个4行3列，元素类型是浮点类型的二维数组，各元素值未知
print(arr1)
```

可以看到，数组元素的值是随机的，创建一个空数组的意义在于便于后续操作无需通过循环创建数组，可直接通过数组索引修改数组值。

- **创建全0（全1）数组**

`zeros`和`ones`方法分别用于创建一个指定形状和数据类型的全0数组或全1数组。

```
arr2 = np.zeros(3)  # 创建一个长度为3的一维全0数组，数据类型默认为浮点型
print(arr2)
```

```
arr3 = np.ones((2,2,2),dtype=int)  # 创建一个2*2*2的三维数组，数据类型为整型的全1数组
print(arr3)
```

- **以现有数据创建数组**

`array`可以以现有的列表、元组创建NumPy数组（ndarray对象）

```
data = [[1,2,3],[4,5,6],[7,8,9]]
arr4 = np.array(data)
print(arr4)
```

- **从数值范围创建数组**

`arange`类似于Python自带的`range`方法，创建一个序列数组。

```
arr5 = np.arange(10)  # 创建 0~9 的序列数组
arr6 = np.arange(3,7)  # 创建 3~6 的序列数组
arr7 = np.arange(1,11,2,dtype = float)  # 创建 1~9 的步长为 2 的，数据类型为浮点型的序列数组
print(arr5)
print(arr6)
print(arr7)
```

- **创建等差数列**

`linspace`函数用于创建一个一维数组，数组是一个等差数列构成的。

```
arr8 = np.linspace(1,10,10) # 创建 1~10 且包含10，等分成10个元素的一维数组
arr9 = np.linspace(10,20,5) # 创建 10~20 且包含20，等分成5个元素的一维数组
print(arr8)
print(arr9)
```

步骤3 数组操作

- **查看数组属性**

一个Numpy数组的基本属性包括维度、形状和元素个数，我们查看一下上一步骤中创建的几个数组的属性信息。

```
print("ndim of arr3 is {}".format(arr3.ndim))  # ndim表示数组的维度
print("shape of arr4 is {}".format(arr4.shape))  # shape表示数组的形状
print("size of arr4 is {}".format(arr4.size))  # size表示数组的元素个数
```

- **修改数组形状**

`reshape`方法可以在不改变数据的前提下修改数组的形状。例如我们首先创建一个长度为24的一维数组，再将其转换为4行6列的二维数组。

```
arr10 = np.arange(24)
print(arr10)
```

```
arr11 = np.reshape(arr10,(4,6))
print(arr11)
```

`transpose`方法用于翻转数组，类似于我们在线性代数中学习的矩阵的转置。

```
arr12 = np.arange(1,10).reshape((3,3))
print(arr12)
```

```
arr13 = np.transpose(arr12)
print(arr13)
```

或者我们可以像表示转置矩阵那样直接使用数组的`T`方法

```
arr14 = arr12.T
print(arr14)
```

步骤4 简单数学操作

在C语言中，如果想对一个数组的所有元素各自增1，我们需要编写循环逐个对数组元素操作。而在NumPy中这一切都非常简单，这都依赖内部的“广播”（Boardcast）机制完成。

```
arr15 = np.arange(5)  # 生成0-4的长度为5的一维数组
print(arr15)
arr15 += 1  # 我们只需简单地进行数学运算就可以对数组全部元素施加操作
print(arr15)
```

```
arr16 = np.arange(9).reshape((3,3))
print(arr16)
arr16 *= 2
print(arr16)
```

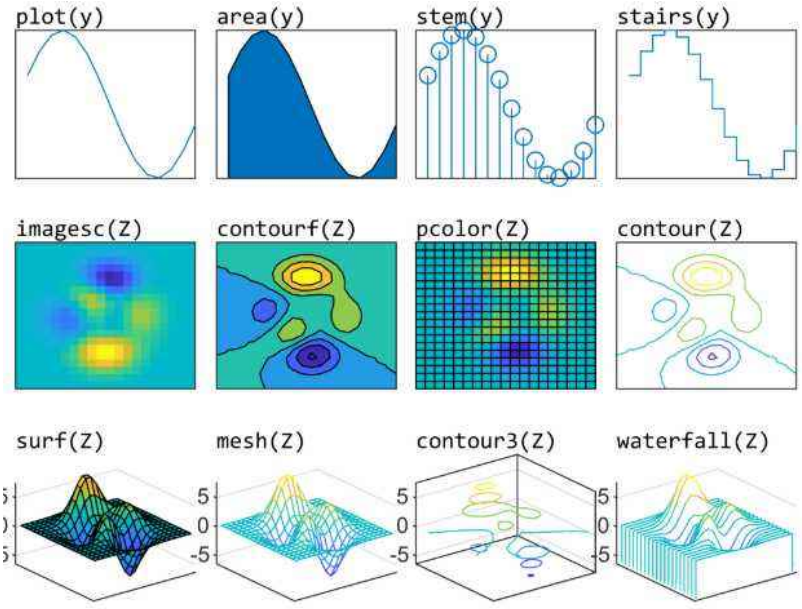
我们还可以从统计学角度处理数据，寻找数组的最值、均值、方差等等，并且可以沿某个轴寻找统计量。

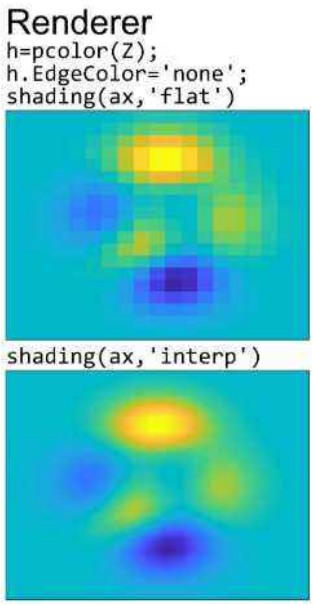
```
arr17 = np.array([[3.2, 1.7, 5.3],
                  [2.1, -5.0 ,3.5],
                  [7.2, 1.1, 0.3]])
print(np.max(arr17))  # 返回数组最大值
print(np.min(arr17,axis=0))  # 返回数组沿第0轴（也就是每一列）的最大值
print(np.mean(arr17,axis=1))  # 返回数组沿第1轴（也就是每一行）的均值
print(np.var(arr17))  # 返回数组所有元素的方差
```

更多NumPy内容可在菜鸟教程中继续学习。

实验7-3 Matplotlib库初步

Matplotlib是Python编程语言及其数值数学扩展包 NumPy的可视化操作界面。可以作出很多精美的实验图像。





步骤1 安装Matplotlib

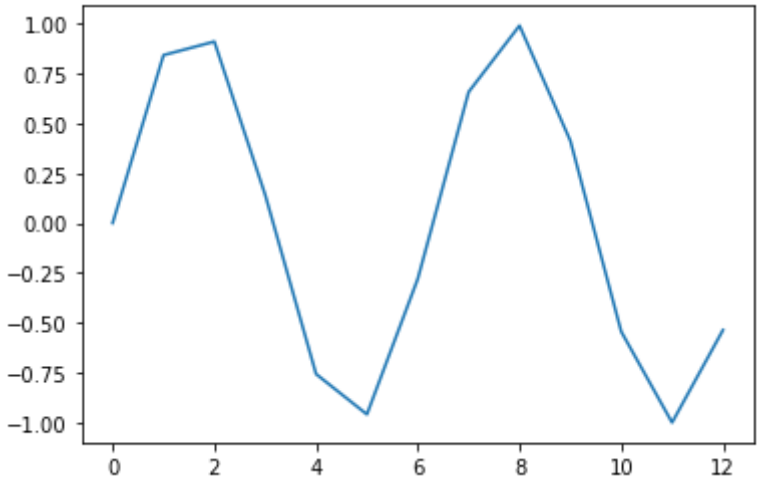
Matplotlib同样是第三方库，需要我们手动安装。

启动Anaconda PowerShell Prompt命令行，切换到自己的环境（如有），在命令行中输入`conda install matplotlib`，conda会自动下载并安装Matplotlib。

步骤2 绘制函数图像

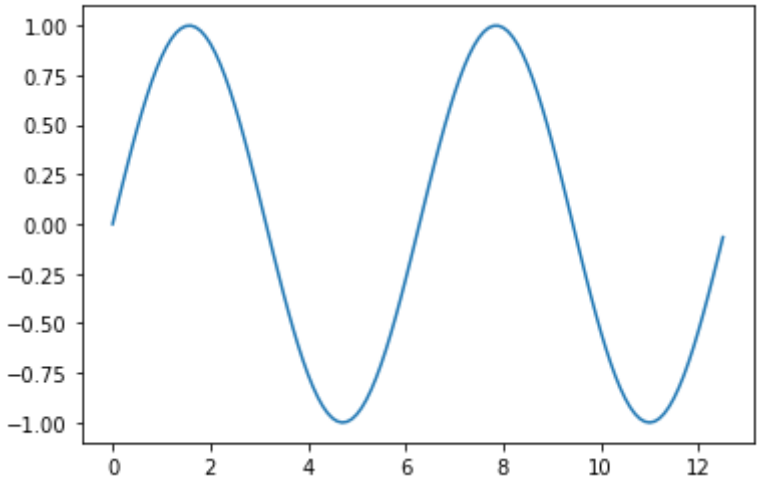
我们从绘制简单的函数图像入手。

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

x = np.arange(0, 4*math.pi)
y = np.sin(x)

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



似乎图像不太“平滑”，这是因为我们的x的间隔是默认值1，Matplotlib将每对(x,y)点用直线连接起来。我们可以把`arange`方法的步长设置的小一点，重新画图。

```
x = np.arange(0,4*math.pi,0.1)
y = np.sin(x)

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

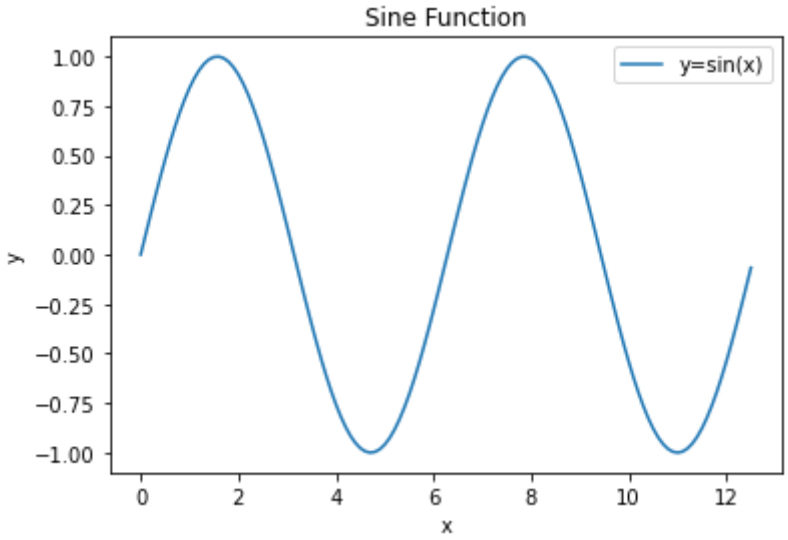


图像目前是平滑多了，不过能在图上添加一些注释信息帮助别人快速读懂就更好了！

我们可以给图像添加：

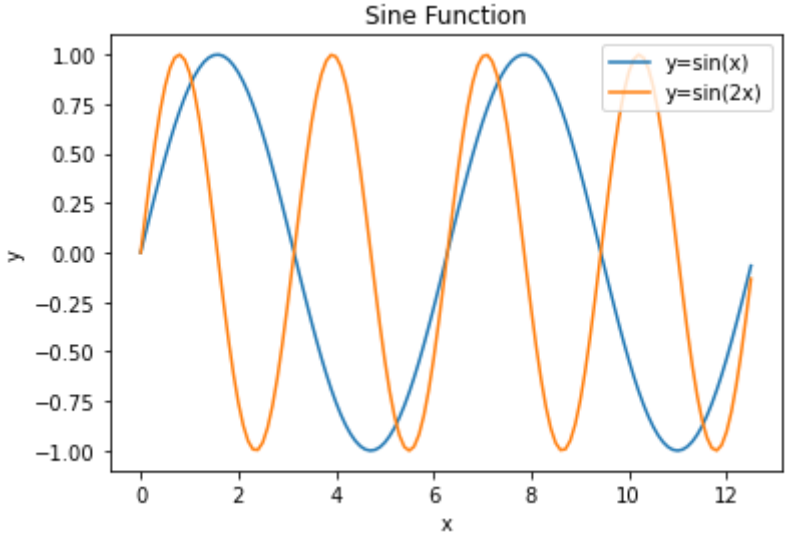
- 标题（`title`方法）
- 坐标轴标签（`xlabel`, `ylabel`方法）
- 图例（`legend`方法）

```
x = np.arange(0,4*math.pi,0.1)
y = np.sin(x)

plt.plot(x,y,label='y=sin(x)')

plt.title('Sine Function')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()

plt.show()
```



有的时候我们需要在同一坐标系中对比多个函数图像，也就是说绘制多条函数。例如我们在同一坐标系中绘制出$y=sin(x)$和$y=sin(2x)$的图像。

```
x = np.arange(0,4*math.pi,0.1)
y1 = np.sin(x)
y2 = np.sin(2*x)

plt.plot(x,y1,label='y=sin(x)')
plt.plot(x,y2,label='y=sin(2x)')
plt.title('Sine Function')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()

plt.show()
```



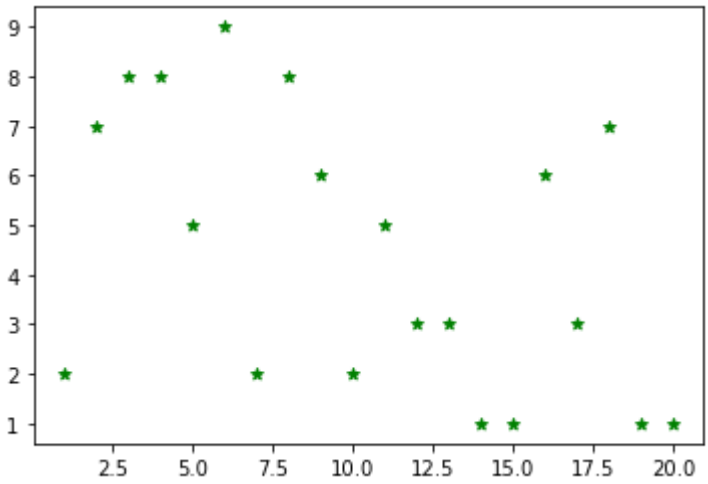
步骤3 绘制散点图

我们在高中数学课上学过，如果一系列数据点的散点图近似于线性形状，则它们之间存在相关性。

当然我们一开始先练习如何绘制散点图，关于线性相关性的判别我们以后再来学习。

假设我们生成20个点，x的取值范围为[1,20]，y的取值范围为[1,10]，且y由随机产生的整数。

```
import random

x = np.arange(1,21)
y = np.random.randint(1,10,size=20)

plt.scatter(x,y,color='green',marker='*')
plt.show()
```



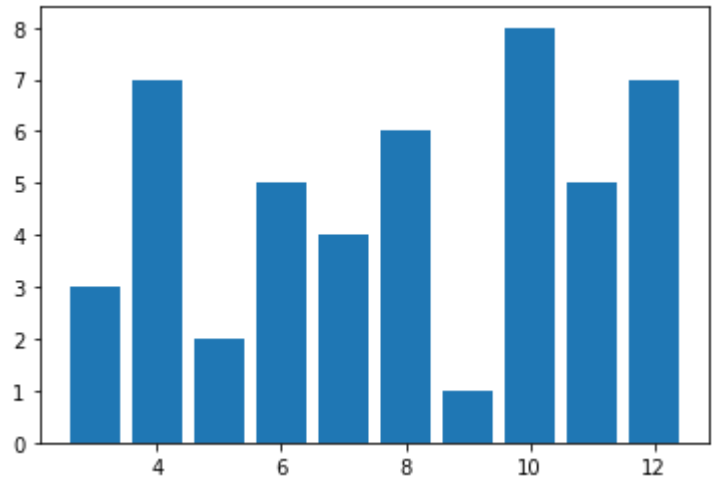
可能你会觉得默认的蓝色太难看了，要么尝试一下绿色？在`plt.scatter()`中添加一个`color`参数并设置为`'green'`，然后重新运行一下看看。

厌倦了传统的圆点？再添加`marker`参数，设置为`'*'`然后重新运行一下看看。

步骤4 绘制条形图

条形图是用条形的长度表示各类别的大小，可以方便地对比各类别数量上的差别。

```
x = np.arange(3,13)
y = np.array([3,7,2,5,4,6,1,8,5,7])

plt.bar(x,y,label='C1')
plt.show()
```

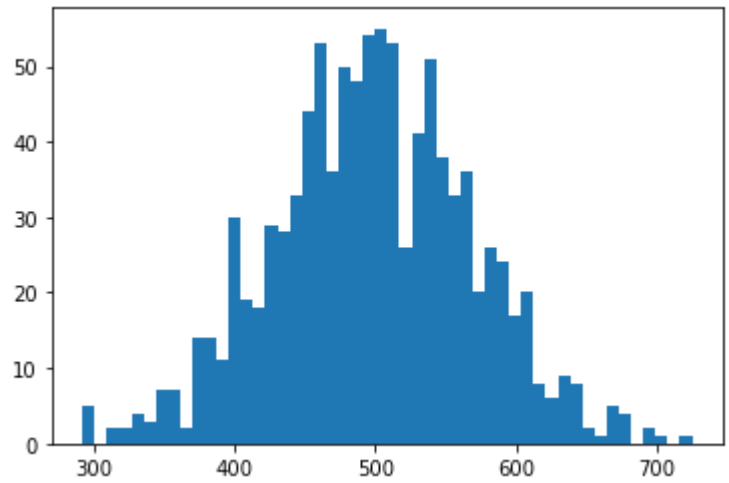


步骤5 绘制直方图

频数分布直方图只需要一列的数据，绘图的x轴将表示这一列数据的种类，y轴表示该类别出现的次数。

假设我们要生成1000个服从均值为500，标准差为70的正态分布的随机数（向上取整），并统计各个数字的频数并展示。

```
import random
import math
import matplotlib.pyplot as plt

data = []
for i in range(1000):
    data.append(math.ceil(random.normalvariate(500,70)))
plt.hist(data, bins=50)  # 以50条柱状图显示
plt.show()
```

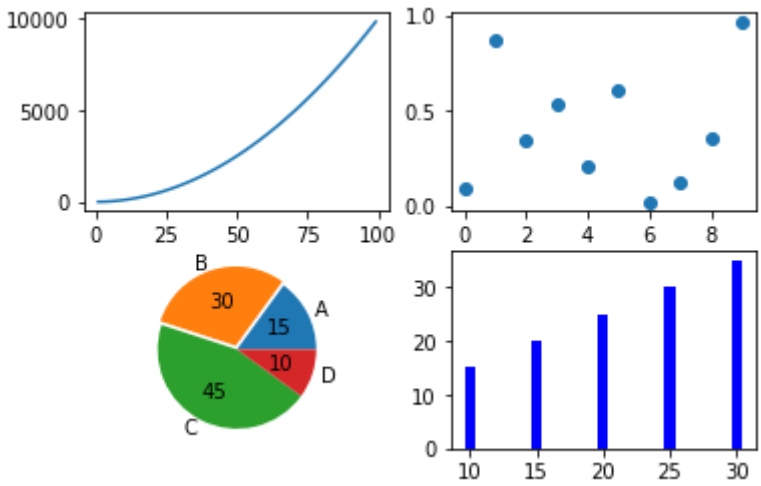


步骤6 创建子图

有的时候我们不想在同一坐标系展示多个函数曲线，或者我们需要在一张图中同时展示函数曲线和散点图，这时候我们需要绘制多个“子图”。

例如我们在一张大图中绘制4幅子图。

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

# 画第1个图：折线图
x=np.arange(1,100)
plt.subplot(221)  # 2行2列，在第1张（序号从左向右从上到下递增）子图中绘制
plt.plot(x,x*x)

# 画第2个图：散点图
plt.subplot(222)
plt.scatter(np.arange(0,10), np.random.rand(10))

# 画第3个图：饼图
plt.subplot(223)
plt.pie(x=[15,30,45,10],labels=list('ABCD'),autopct='%.0f',explode=[0,0.05,0,0])

# 画第4个图：条形图
plt.subplot(224)
plt.bar([20,10,30,25,15],[25,15,35,30,20],color='b')

plt.show()
```



步骤7 保存输出图像

输出的图像要是能保存至磁盘，日后可直接查看就更方便了。

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

x=np.linspace(-2*np.pi,2*np.pi,400)
siny=np.sin(x)
cosy=np.cos(x)
```

```
plt.plot(x,siny,color="red",label="sin(x)")
plt.plot(x,cosy,color="blue",label="cos(x)",linestyle="--")
plt.xlabel("x")
plt.ylabel("y")
plt.title("Sin & Cos")
plt.legend()
plt.savefig('pic.png',dpi=500)
```

这里有更多的精美图像示例，也提供相应源代码供大家下载学习。

实验练习07

1. 在数据学院-功夫编程平台上练习实验7-1的相关命令。
2. `daily_KP_SUN_2020.csv` 取自香港天文台开放数据集，该数据为香港京士柏气象监测站采集的2020年1月1日至9月30日每日日照小时数。请先用Excel或记事本预览该数据集，然后通过Python文件读取，用Matplotlib绘制出每月**总日照小时数**对比的柱状图和每月**平均日照小时数**对比的柱状图（请在画布上绘制两个子图，可左右排列也可上下排列）。

> 注意：源数据文件最后几行含有一些注释内容，你可能需要手动删除他们。

3. 鸢尾花数据集是机器学习和数据分析领域中常用的数据集，通过花萼长度(Sepal Length)，花萼宽度(Sepal Width)，花瓣长度(Petal Length)，花瓣宽度(Petal Width)4个属性预测鸢尾花卉属于(Setosa，Versicolour，Virginica)三个种类中的哪一类。请通过Python读取 `iris.csv` 文件，分别绘制出4个属性之间两两组合组成的6组变量组合的散点图（可在一张画布上显示2*3的子图），寻找是否可通过某两个变量可确定鸢尾花的种类。