第一章 Python 语言简介

1.1 Pyhton 语言

Python 是一种解释性脚本语言，是由 Guido van Rossum 设计，并于 1991 年正式发行。Python 语言强调程序的可读性，是最接近自然语言的程序语言之一。2000 年发行了 Python2 版本，得到了广泛应用，积累了非常丰富的程序库。2008 年发行了 Python3，是目前使用的主流版本。本课程选择 Python 作为教学程序语言，原因如下：1）Python 语言简单易学，即使没有专门修过 Python 课程的同学，通过短时间的训练，便能轻松达到本课程所需的程序写作要求； 2）Python 语言有很强的绘图功能，使得同学们能够方便地将计算物理课程中的实例操作结果用可视化的方式表现出来，帮助同学们更直观形象地理解计算结果的物理图像；3）Python 可移植性好，能够在 Linux， Windows，MacOS 等多个平台使用；4）Python 是免费的，且有丰富的程序库供使用。

1.2 一个简单的 Pyhton 程序：随机游走（random walk）。

从坐标原点(0,0)出发的粒子，每一步都以固定的步长在二维平面上随机移动。用蒙特卡洛方法，模拟粒子随机游走过程，画出游走轨迹。

```python
# -*- coding: utf-8 -*-

"""
random walk
@author: wfli
"""
import numpy as np
import matplotlib.pylab as pl
import random

line = input("input the maximal nubmer of walk steps:")
nstep = int(line)
print("the maximal number of walk steps is:", nstep)

if nstep > 10000:
    print('too many steps')
    exit()
elif nstep <10:
    print('too few steps')
    exit()

xlist = []   # create empty list
ylist = []

x = 0.0
y = 0.0
xlist.append(x) # append the initial x value to the xlist.
ylist.append(y) # append the initial y value to the ylist.

random.seed(913) #ramdom number seed.
for i in range(nstep):
    dx = 2*random.random()-1 # random number(-1.0,1.0)
    dy = 2*random.random()-1

    dr = np.sqrt(dx**2+dy**2)
    dx = dx/dr #normalization
    dy = dy/dr

    x = x + dx # updating x position
    y = y + dy
    xlist.append(x)
    ylist.append(y)

pl.plot(xlist[:], ylist[:],'r-',lw=1) #visualization
pl.xlabel('x-axis')
pl.ylabel('y-axis')
pl.show()

```

code-1-1.py

以上 python 可以通过如下两种方式执行：

1） 命令行方式：在当前目录的命令提示符下，直接输入 python 文件名.py。例如，以下是 macOS 操作系统下的运行命令：

```
wflimac:lecture1 wfli$
wflimac:lecture1 wfli$ python code-1-1.py
input the maximal nubmer of walk steps:1000
the maximal number of walk steps is: 1000
```

运行结果如下：

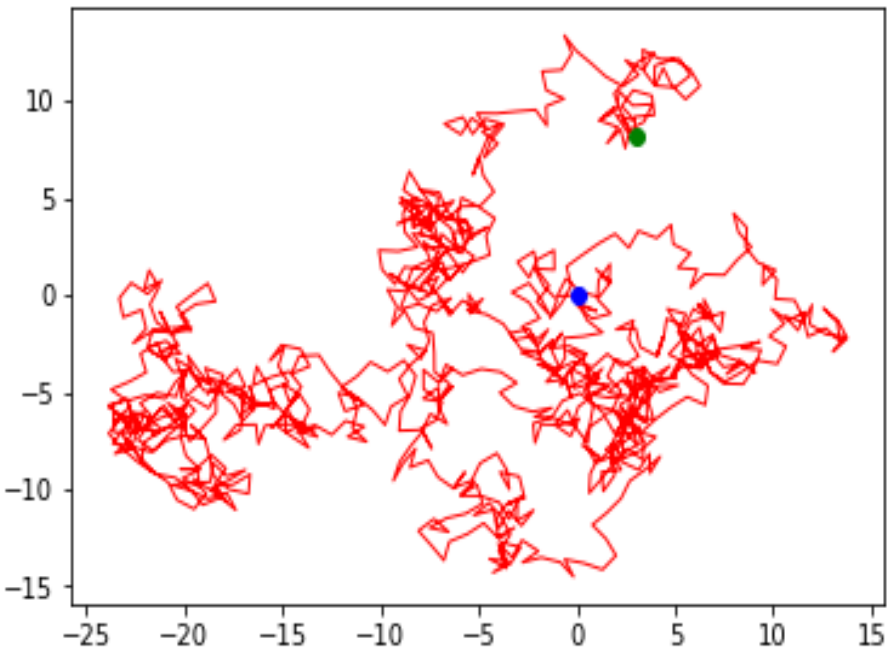


图 1.1 二维随机游走轨迹图。

2） 在 python 的 IDE 工具下运行。例如，可以在 spider 环境下，左侧编辑窗口可以直接创建、编辑程序。右下角的控制台窗口，可以运行程序，命令为 “run code-1-1.py”。运行结果将在控制台窗口显示。也可以直接点击顶部的 “run” 按钮，实现当前程序的运行。

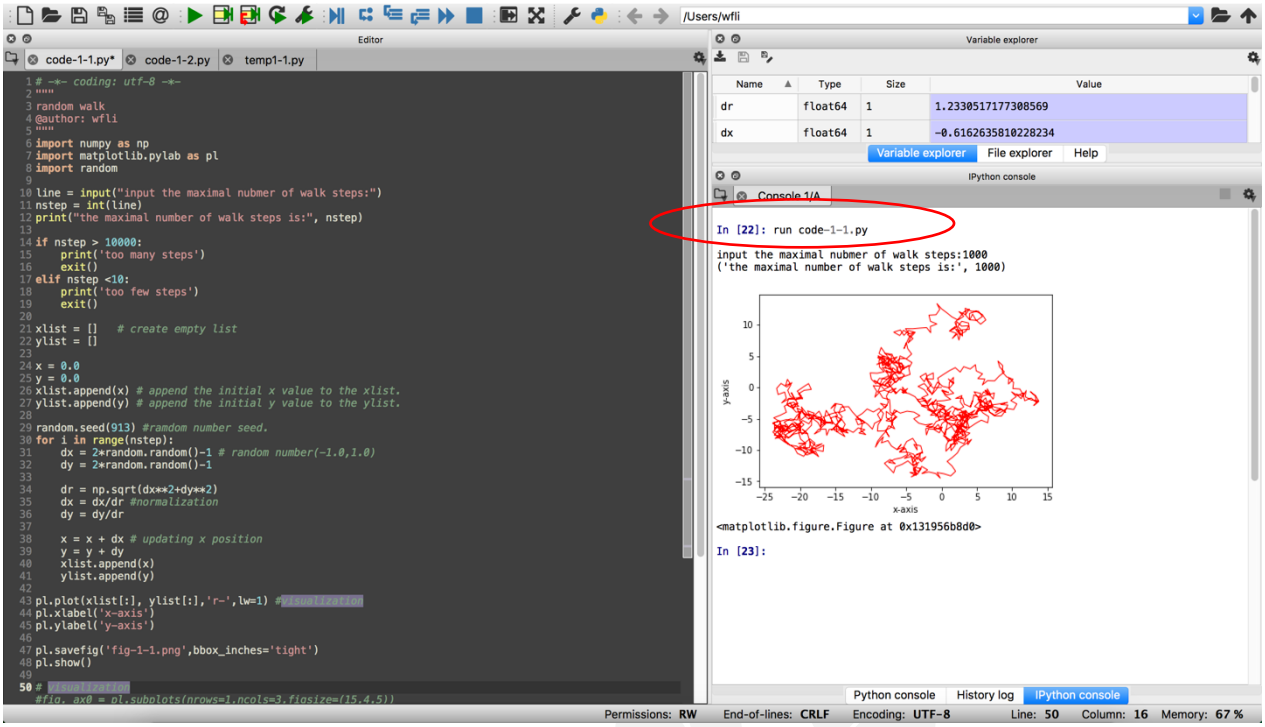


图 1.2 通过 spider 工具实现程序的编辑和运行示意图。

1.3 python 语法简介

下面我们将通过对随机游走程序做详细讲解，介绍相关的 python 基本语法。

1.3.1 注释语句

以上随机游走程序中，前两行为注释行。Python 程序中，如果是多行注释语句，用三个引号来标示。如果是单行注释语句，则在语句前加#号。以上注释语句也可以改为：

```
6
7 # random walk
8 # @author: wfli
9
```

1.3.2 外部函数库

Python 语言的一个重要优势是具有丰富的外部函数库可供使用。以上随机游走程序 *code-1-1* 中，7-9 行调用了 3 个外部函数库，包括 numpy 库，matplotlib 库以及 random 库。其中，numpy 是对 python 数值计算功能的扩展，可以方便的处理数组和矩阵等类型的数据。Matplotlib 是 python 的绘图函数库，利用 Matplotlib 可以非常简单地绘制具有发表水平的高质量图表。random 是随机数生成函数库。

为了在程序中调用这些外部函数，需要在程序的开头先加载相应的函数库。例如，*code-1-1* 中的第 7 行通过 import 加载函数库 numpy，并将 numpy 命名为 np。为了节约内存资源，Python 也可以只加载函数库中的某个模块。*code-1-1* 的第 8 行语句“import matplotlib.pylab as pl”通过 import 加载了 matplotlib 函数库中的 pylab 模块，并将其命名为 pl。

1.3.3 标准输入输出

Python 通过 input()和 print()函数实现标准输入和输出。其中，Input()函数默认的输入为字符串数据，需要通过类型转换函数来实现输入数据类型的转换。*code-1-1* 程序的 11-13 行实现最大随机游走步数的读入，并赋值给字符串变量 line。运行后，程序会等待用户的输入。

```
wflimac:lecture1 wfli$ python code-1-1.py
input the maximal nubmer of walk steps:
```

当键入数字 1000，并回车后，程序读入所输入数据，并通过 int(line)将输入数据转换为整型数字格式。第 13 行的 print()函数将 nstep 数值打印出来。运行结果如下：

```
wflimac:lecture1 wfli$ python code-1-1.py
input the maximal nubmer of walk steps:1000
the maximal number of walk steps is: 1000
```

1.3.4 数据类型，赋值语句，数据类型转换

Python 定义了 5 标准数据类型，包括数字(Numbers)类型，字符串(Strings)类型，列表（List）类型，元组（Tuple）类型，以及字典（Dictionary）类型。数字类型数据包括整型（int），长整型（long），浮点型（float），和复数型（complex）。字符串是由字母、数字以及下划线组成的一串字符，用单（或双）引号标示。例如，*code-1-1* 程序的第 13 行中的“input the maximal number of walk steps:”便是一个字符串。另外，由 input()函数输入的内容（例如 1000），被默认为字符串，变量 line 为字符串变量。等号“=”表述赋值操作。显然，随机游走问题中需要输入的最大游走步数应该为整数，因此该程序通过 int()函数将 line 转换为整型，并赋值给变量 nstep。其中变量 nstep 储存整型数字（这里为 1000）。

列表是 python 语言中使用非常频繁一类复合型数据类型，用[…]标示。例如，[10, 11,12]是一个包含 3 个整数的列表。列表的元素也可以为字符，字符串，浮点数，复数等多种数据类型。列表甚至可以不包含任何元素，称为空列表，记为[ ]。常用的列表操作有赋值，引用，追加元素，求列表长度等。例如，在如下 spider 的 ipython 控制窗口中，a=[10, 11, 12]和 b=[5, 6, 7, 8]实现将两个包含整数数字的列表分别赋值给 a 和 b。c = a+b 实现将 a 和 b 连接在一起，形成新的列表[10, 11, 12, 5, 6, 7, 8]，并赋给变量 c。

```
In [7]: a = [10,11,12]

In [8]: b = [5, 6, 7, 8]

In [9]: c = a + b

In [10]: c
Out[10]: [10, 11, 12, 5, 6, 7, 8]
```

列表可以通过[m:n]格式引用列表中的第 m 到第 n-1 个元素（从第 0 个元素算起），例如，如下程序行中，d=c[3:6]抽取了列表 c 中的第 3，4，5 元素，形成新的列表 [5,6,7]，并赋值给变量 d。

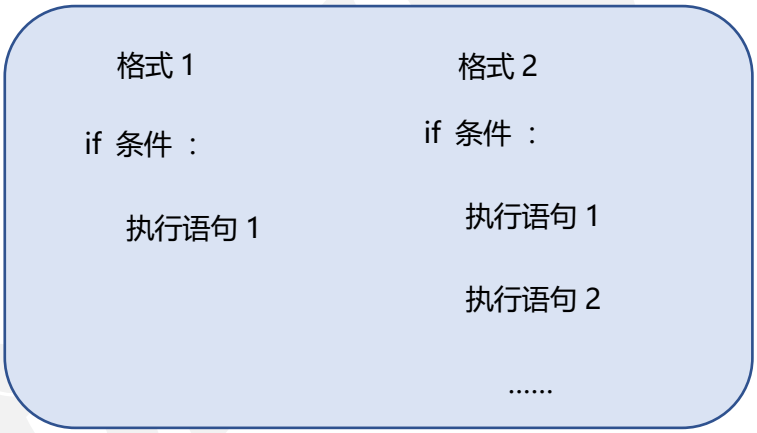
```
In [25]: c = [10,11,12,5,6,7,8]

In [26]: d = c[3:6]

In [27]: d
Out[27]: [5, 6, 7]

In [28]:
```

列表可以通过 append()函数来追加新元素。例如，如下程序行通过 c.append(9)将新元素 9 添加到列表 c。新添加的元素也可以为列表。例如，c.append([10,11])将列表[10,11]添加到列表 c。列表的长度可以通过函数 len()得到。添加新元素后得到的列表 c 的长度为 9。

```
In [28]: c = [10,11,12,5,6,7,8]

In [29]: c.append(9)

In [30]: c.append([10,11])

In [31]: c
Out[31]: [10, 11, 12, 5, 6, 7, 8, 9, [10, 11]]

In [32]: len(c)
Out[32]: 9
```

1.3.5 条件语句

Pyhton 通过 if 语句实现选择执行功能，包括如下两种基本格式：



在随机游走程序 *code-1-1* 的 15-20 行，通过 if 条件语句，将随机游走步数限制在 10-1000 之间。如果所输入的步数大于 1000，则条件成立，执行打印输出语句 print('too many steps'),并退出程序。否则，当所输入的步数小于 10，则执行打印输出语句 print('too few steps')，并退出程序。Python 通过缩进来定义程序区域，具有相同缩进的连续程序行属于同一个程序区域。例如 16-17 行的 print('too many steps')和 exit()语句具有相同的缩进，因此属于同一个程序区域，在满足条件 nstep >10000 时按顺序执行。

1.3.6 列表操作

程序 *code-1-1* 的 22-28 行给出了随机游走的初始化过程。其中列表 xlist 和 ylist 分别记录随机游走过程中的 x 位置和 y 位置，22-23 行将其初始值设为空列表。25-28 行将游走粒子的初始位置(0,0)的两个分量分别赋给浮点型变量 x 和 y，并将其追加到列表 xlist 和 ylist。

1.3.7 随机数产生

Python 的 random 函数库实现具有各种分布特征的随机数序列的产生。在产生随机数之前，通常需要设置随机数种子。Python 可以通过 random 函数库的 seed()函数设定随机数种子。例如，*code-1-1* 程序第 30 行的函数 random.seed(913)设定随机数种子为 913。随机数种子设定后，据可以产生特定的随机数序列。其中，random.random()产生一个处于(0,1)范围内的均匀分布的随机数。当随机数种子相同时，连续调用 random.random()所产生的随机数序列是相同的。因此，为了产生不同的随机数序列，需要设置不同的随机数种子，或通过 random.seed(None)利用计算机时钟来确定随机数种子。常用的随机数生成函数包括见表 1.

表 1:常用的随机数产生函数

函数名	功能
random.random()	产生(0,1)之间的均匀分布随机数
random.uniform(a,b)	产生(a,b)之间的均匀分布随机数
random.randint(a, b)	产生(a,b)之间的均匀分布整数
random.gauss(mu, sigma)	产生平均值为 mu,标准偏差为 sigma 的高斯分布随机数
random.expovariate(lambd)	产生平均值为 lambd 的指数分布随机数

1.3.8 循环语句

Python 可通过如下 for 语句实现循环操作，基本格式为：

```
for 变量 in 序列:
    执行语句 1
```

for 循环语句对给定序列中的所有元素循环，每次循环都要执行缩进部分的所有执行语句。程序 code-1-1 中的 30-42 行，首先由 range(nstep)产生[0,1,…,nstep-1],表示随机游走步的序号，然后通过 for 语句，依次将序列中的元素赋给变量 i，并对每一个 i（随机游走步)执行缩进部分的所有执行语句。其中，32 行的语句 dx=random.random()产生（-1.0，1.0）之间均匀分布的随机数，设置为 x 方向的位移。dy=2*random.random()-1.0 产生（-1.0，1.0）之间均匀分布的随机数，设置为 y 方向的位移。dr=np.sqrt(dr**2+dy**2)给出游走步长。为了使得每步的随机游走步长都为 1.0，通过 36-37 行的语句 dx=dx/dr 和 dy=dy/dr 对位移进行归一化。39-40 行的语句 x=x+dx 和 y=y+dy 实现了随机游走，得到新的位置(x,y)，并通过 41-42 行的 xlist.append(x)和 ylist.append(y)函数将新的坐标位置追加到位置列表中。循环结束后，位置列表 xlist 和 ylist 分别记录了随机游走每一步的 x 坐标和 y 坐标值，即随机游走轨迹，包含 nstep 个点。

1.3.9 matplotlib 作图

Matplotlib 是一个跨平台 python 绘图函数库，能轻松实现具有发表质量的 2D 插图绘制，也是目前应用最为广泛的绘图工具之一。在随机游走程序 code-1-1 的开头部分，通过 import matplotlib.pylab as pl 将 matplotlib 函数库中的 pylab 模块载入，并简写为 pl。在 44-47 行，首先调用 pylab 中的函数 pl.plot()进行绘图，其中 xlist 列表中存储的所有 x 值作为横坐标，ylist 列表中存储的所有 y 值作为纵坐标，‘r-’表示所绘的轨迹线为红色实线，lw=1.0 表示实线的宽度为 1.0。然后 45-46 行通过调用函数 pl.xlable('x-axis')和 pl.ylabel('y-axis')设置横坐标和纵坐标名称。最后，通过 47 行的 pl.show()将图形在窗口显示出来，运行结果如图 1.1 所示。

关于 matplotlib 的使用,建议大家访问官网中的 examples 部分的示例(https://matplotlib.org/3.1.3/gallery/index.html)，参考实例中的图形和程序作图。通常来说，从 examples 中的 gallery 总是能够找到符合要求的图例供参考。

1.3.10 文件读写

在科学计算中，据大多数的数据存储在文件中。因此，将文件中的数据读入，进行处理，并将处理好的数据存入文件是常见的操作。以下示例程序 code-1-3 将随机游走轨迹存入数据文件，code-1-4 将数据从文件读入，并画出轨迹图。

```
1 # -*- coding: utf-8 -*-
2 """
3 random walk
4 @author: wfli
5 """
6
7 import numpy as np
8 import random
9
10 outfile = open('walk-traj.dat', 'w')
11
12 line = input("input the maximal nubmer of walk steps:")
13 nstep = int(line)
14 print("the maximal number of walk steps is:", nstep)
15
16 if nstep > 10000:
17     print('too many steps')
18     exit()
19 elif nstep <10:
20     print('too few steps')
21     exit()
22
23 x = 0.0
24 y = 0.0
25
26 random.seed(913) #ramdom number seed.
27 for i in range(nstep):
28     dx = 2*random.random()-1 # random number(-1.0,1.0)
29     dy = 2*random.random()-1
30
31     dr = np.sqrt(dx**2+dy**2)
32     dx = dx/dr #normalization
33     dy = dy/dr
34
35     x = x + dx # updating x position
36     y = y + dy
37
38     outfile.write('%d\t%.2f\t%.2f\n' %(i, x, y))
39
40 outfile.close()
41
```

code-1-2.py

以上程序 code-1-2 的第 10 行创建一个文件对象 outfile，所建立的新文件的文件名为“walk-traj.dat”，属性为可写。第 38 行将每一随机游走的序号 i 以及位置(x,y)写入文件中。其中%d 表示 i 的格式为整数；%.2f 表示 x,y 的输出格式为包含两位小数的实数。第 40 行将文件关闭。执行 code-1-2 后将在当前目录下创建文件“walk-traj.dat”，其前 10 行内容如下：

```
0       -1.00   -0.06
1       -0.56   0.84
2       0.17    0.16
3       1.14    -0.09
4       1.74    0.71
5       0.97    1.34
6       1.64    0.60
7       1.69    1.59
8       1.11    0.78
9       1.05    -0.21
```

接下来在文件 code-1-3 中，将数据读入，并画出轨迹。其中 4-6 行定义了空列表，为存储读入的数据做准备。第 8-16 行给出一种基本的文件读入格式。其中，第 9 行将只读文件“walk-traj.dat”打开，建立文件对象 infile。10-15 行对文件的每一行进行循环，其中 lines 对应所读入的文件中每一行的内容，默认为字符串。第 11 行通过调用 split()函数，将每一行所读入的字符串按空格划分为不同的字段。12-14 行将三个字段分别放入列表 time，xlist 和 ylist 中。循环结束后，上述三个列表分别存储了有文件读入的美衣游走步的序号，x 坐标和 y 坐标。19-22 行将读入的轨迹作图。程序运行后将产生与图 1-1 完全相同的图形。

```
1 # -*- coding: utf-8 -*-
2 import matplotlib.pylab as pl
3
4 time = []
5 xlist = []
6 ylist = []
7
8 istep=0
9 with open ('walk-traj.dat', 'r') as infile:
10      for lines in infile:
11          words=lines.split()
12          time.append(int(words[0]))
13          xlist.append(float(words[1]))
14          ylist.append(float(words[2]))
15          istep = istep + 1
16 nstep = istep
17
18 print("the maximal number of walk steps is:", nstep)
19 pl.plot(xlist[:], ylist[:],'r-',lw=1) #visualization
20 pl.xlabel('x-axis')
21 pl.ylabel('y-axis')
22 pl.show()
23
24
```

code-1-3.py

1.3.10 继续随机游走

在随机游走问题中，以上给出的单条随机游走轨迹具有随机性，提供的信息非常有限。人们关注更多的是随机游走的统计行为，例如扩散系数、位置分布、始末距离分布等。下面的程序 code-1-4 进一步通过 python 实现多次随机游走模拟，并计算以上统计量。同时，我们将学习 python 的子函数定义，numpy 的直方图方法计算分布函数，scipy 的最小二乘函数库实现数据拟合等。

```python
# -*- coding: utf-8 -*-
"""
random walk
@author: wfli
"""
import numpy as np
import matplotlib.pylab as pl
import random
from scipy.optimize import leastsq

nwalker = 50000
nstep = 100

time = np.arange(nstep)
xr = np.zeros(nwalker)
yr = np.zeros(nwalker)
r = np.zeros(nwalker)
sum_walk = np.zeros(nstep)
y_fit = np.zeros(nstep)
p0 = np.zeros(2)

for iwalk in range(nwalker):
    random.seed(None)
    x = 0.0
    y = 0.0
    r2 = 0.0
    for istep in range(nstep):
        sum_walk[istep] = sum_walk[istep] + r2
        dx = 2*random.random()-1
        dy = 2*random.random()-1
        dr = np.sqrt(dx**2+dy**2)
        x = x + dx/dr
        y = y + dy/dr
        r2 = x**2 + y**2

    xr[iwalk] = x
    yr[iwalk] = y
    r[iwalk] = np.sqrt(x**2+y**2)

for istep in range(nstep):
    sum_walk[istep] = sum_walk[istep]/nwalker

xprob, xedge = np.histogram(xr,bins = 20,normed=True)
yprob, yedge = np.histogram(yr,bins = 20,normed=True)
rprob, redge = np.histogram(r,bins = 20,normed=True)

xx = 0.5*(xedge[0:-1]+xedge[1:])
yy = 0.5*(yedge[0:-1]+yedge[1:])
rr = 0.5*(redge[0:-1]+redge[1:])

#-----------------------------------
def func(x, p):
    a0 = p[0]
    alpha0 = p[1]
    return a0*pow(x,alpha0)
def residuals(p, y, x):
    return y - func(x, p)
#-----------------------------------
p0[0] = 1.0
p0[1] = 1.0
parameters = leastsq(residuals, p0,args=(time,sum_walk))
print(parameters[0][0],parameters[0][1])
for i in np.arange(nstep):
   y_fit[i] = parameters[0][0]*np.power(time[i],parameters[0][1])
#-----------------------------------
fig = pl.figure(figsize=(7,7))
ax1 =fig.add_subplot(2,2,1)
ax2 =fig.add_subplot(2,2,2)
ax3 =fig.add_subplot(2,2,3)
ax4 =fig.add_subplot(2,2,4)

pl.subplots_adjust(left=0.15,bottom=0.1,top=0.9,right=0.95, \
                   hspace=0.25,wspace=0.3)

ax1.plot(time[::2],sum_walk[::2],'ro',fillstyle='none', \
         lw=1.0,label='data')
ax1.plot(time,y_fit, 'b-', linewidth=2.0, label='fitting')

ax2.plot(rr, rprob, 'm-',linewidth=2.0)
ax3.plot(xx, xprob, 'b-', linewidth=2.0)
ax4.plot(yy, yprob, 'k-', linewidth=2.0)

ax1.set_xlabel(r'time',fontsize=15)
ax1.set_ylabel(r'$<r^2>$',fontsize=15)
ax2.set_xlabel(r'r', fontsize=15)
ax2.set_ylabel(r'distribution', fontsize=15)
ax3.set_xlabel(r'x', fontsize=15)
ax3.set_ylabel(r'distribution', fontsize=15)
ax4.set_xlabel(r'y', fontsize=15)
ax4.set_ylabel(r'distribution', fontsize=15)

ax1.set_xlim(0,100)
ax1.set_ylim(0,120)
ax2.set_xlim(0,30)
ax2.set_ylim(0,0.1)
ax3.set_xlim(-30,30)
ax3.set_ylim(0,0.1)
ax4.set_xlim(-30,30)
ax4.set_ylim(0,0.1)
ax1.legend(loc='upper left',fontsize=15)
pl.show()
```

code-1-4.py

程序运行结果如图 3 所示。程序 code-1-4 开头部分的第 9 行加载了 scipy 函数库中的 optimize 模块的 leastsq 函数库。scipy 是专门为科学计算开发的函数库。绝大多数的科学计算问题，例如数值积分、插值、拟合、优化问题、线性 / 非线性方程组求根、求解微分方程、傅立叶变换等，都能在 scipy 函数库中找到合适的函数来实现。本程序中，我们通过调用 scipy 中的 leastsq 函数实现最小二乘拟合，得到随机游走问题对应的扩散系数。

程序 code-1-4 使用了两种函数库加载语句：1）import numpy as np （第 6 行），使用这种加载格式后，在引用该函数库中的函数时，需要采用如下加前缀的格式：np.函数名()。例如，程序中第 14 行的语句 time=np.arange(nstep) 调用了 numpy 函数库中的 arange()函数，产生了包含从 0 到 nstep-1 的 nstep 个数字的 1 维数组，并赋给变量 time。2）from scipy.optimize import leastsq （第 9 行），使用这一加载格式后，可以直接调用函数库中的函数（例如，leastsq()），而不需要在函数前加前缀。需要注意的是，使用格式 2）加载函数库后，程序中的使用的其他函数名称不能和所加载函数库中的函数同名。

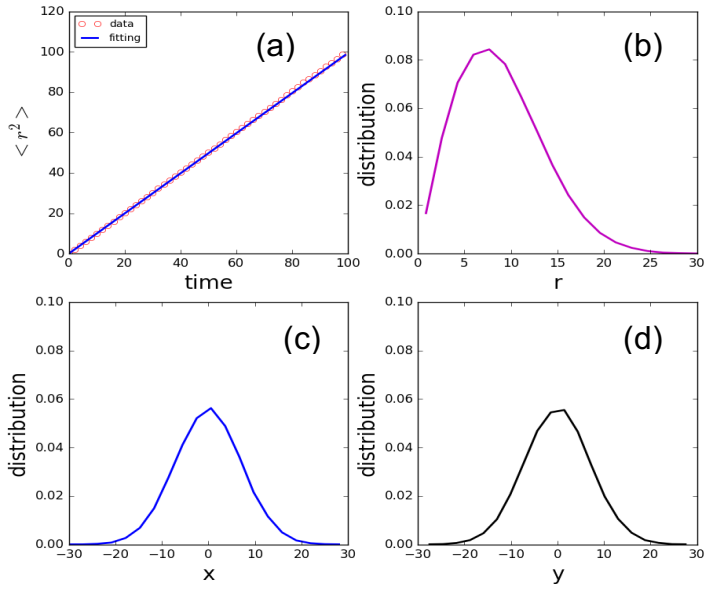


图 1.3 （a）均方位移随游走步数（时间）的变化。蓝实线为最小二乘拟合结果；（b）游走结束后始末位置的距离分布；（c）游走结束后 x 位置的距离分布；（d）游走结束后 y 位置的距离分布。

在 14-20 行，通过调用 numpy 的 zeros(n)函数，定义了包含 n 个元素的数组变量，并将所有元素的数值设置为 0。另外，程序 22-38 使用了嵌套 for 循环。其中外层 for 循环表示对 nwalker 个粒子进行随机游走模拟，内层 for 循环表示对每个粒子进行 nstep 次随机游走模拟。每个粒子完成 nstep 步随机游走后，其终止时的位置(x,y)赋值给数组 xr，yr,并计算其始末位置距离 r. 同时，在随机游走的每一步，计算 nwalker 个粒子的始末位置距离的平方和 sum_walk，为后续计算均方位移和扩散系数做准备。第 40-41 行给出每一游走步的均方位移。

Python 语言可以调用 numpy 函数库中的 histogrm()函数来计算分布函数。例如，程序的 43-45 行调用 histogram()函数，计算出 xr，yr，r 的归一化分布函数。histogram()的返回值包括分布函数(例如 xprob 数组，对应图 1-4 示意图中各窗口的高度)和直方图窗口边界位置数组（xedge，对应图 1-4 示意图中横坐标黑色标线位置）。程序 47-50 行计算出各直方图窗口中心位置，并赋值给数组 xx，yy，rr，对应图 1-4 示意图中横坐标红色标线位置。

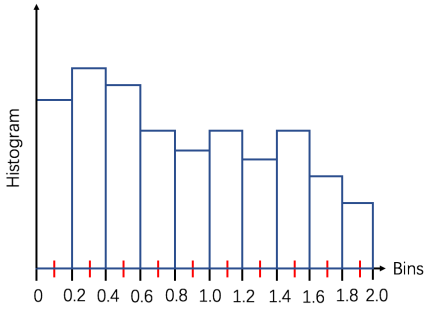


图 1.4 直方图窗口划分示意图。

Python 可以在程序体内自定义子函数。随机游走问题中，我们用函数$\langle r^2\rangle = A\cdot t^{\alpha}$拟合均方位移随时间的变化关系。在 52-64 行，首先定义了子函数 func(x,p)，其中，x 为自变量数组，p 为参数数组，包含 A 和 alpha。子函数的返回值为$\langle r^2\rangle = A\cdot t^{\alpha}$的函数值。子函数 Residuals(p,y,x)为残差函数，返回值为随机模拟得到的数据 y 和由上述解析函数计算得到的函数值之差。Leastsq()是 scipy 库中的子函数，返回参数 A 和 alpha 的最小二乘拟合值。

程序 code-1-4 的 66-101 行涉及更为详细的作图功能。其中 66-70 行创建了包含 2 行 2 列共 4 个子图的图形窗口。83-100 行实现对各个子图的坐标轴名称、横纵坐标范围、字体大小、图例等进行详细设置。

图 1.3(a)的红色圆圈给出的是模拟得到的随机游走的均方位移和游走步数（即时间）的关系。蓝线是由关系式 $\langle r^2\rangle = A\cdot t^{\alpha}$的最小二乘拟合结果。拟合得到的函数关系式为 $\langle r^2\rangle = A\cdot t^{\alpha} \approx 0.99\cdot t^{1.00}$。以上拟合所得到的幂指数约为 1.0,这说明随机游走对应的是正常扩散。基于拟合得到的函数关系式，可以进一步计算得到扩散系数：

$$D = \frac{1}{4}\frac{d < r^2 >}{dt} \approx 0.25$$

图 1.3(b)给出的是模拟得到的多个粒子随机游走结束时的始末位置距离分布。可以看出，距离在 7.0 附近分布最大，随着距离变大，几率分布逐渐变小。如果将每步的随机游走位置看作是高分子链的每个单元的位置，则始末位置距离对应高分子链的脸链端距。以上结果说明，高分子链倾向于塌缩为链端距较小的线团，而较伸展的构象是不稳定的。因此，需要施加拉力才能使高分子链处于伸展的构象状态，即高分子链具有弹型。这种弹性的产生主要是熵效应，因此在高分子领域被称为熵弹性，以区分普通机械弹簧产生的弹性行为。图 1.3(b，c)给出的是随机游走结束时终点位置的在 x 和 y 方向的分布，满足高斯分布。

以上随机游走的例子涵盖了 python 最基本的语法和操作。通过以上程序示例和解读，期望对基本的 python 语法有初步的了解，能够基本满足计算物理课程的程序编写要求。关于 python 的更进一步的语法和高级操作将在课程学期过程中逐渐学习和熟练。