第 13 章　图 形 绘 制

Python 提供了丰富的图形绘制功能。本章主要讲述基于 tkinter 模块的绘图、基于 turtle 模块的绘图和基于 Matplotlib 模块的绘图。

13.1　Python 绘图模块概述

Python 标准库中包括下列图形绘制相关模块。

（1）tkinter：画布绘图。

（2）turtle：海龟绘图。

常用的开源绘图模块库如下。

（1）Matplotlib（官网 http://matplotlib.sourceforge.net/）。Matplotlib 是一个由 John Hunter 等开发的、用以绘制二维图形的 Python 模块。它利用了 Python 下的数值计算模块 Numeric 及 Numarray，克隆了许多 MATLAB 中的函数，用以帮助用户轻松地获得高质量的二维图形。Matplotlib 可以绘制多种形式的图形，包括普通的线图、直方图、饼图、散点图以及误差线图等；还可以比较方便地定制图形的各种属性，例如图形的类型、颜色、粗细、字体的大小等。Matplotlib 能够很好地支持一部分 TeX 排版命令，可以比较美观地显示图形中的数学公式。

（2）Chaco。Chaco 是一个 2D 的绘图库，官网地址：http://code.enthought.com/chaco/。

（3）Python Google Chart。Python Google Chart 是 Google Chart API 的一个完整封装，官网地址：http://pygooglechart.slowchop.com/。

（4）PyCha。PyCha 是 Cairo 类库的一个简单封装和优化，官网地址：https://bitbucket.org/lgs/pycha/wiki/Home。

（5）pyOFC2。pyOFC2 是 Open Falsh Library 的 Python 类库，官网地址：http://btbytes.github.com/pyofc2/。

（6）Pychart。Pychart 用于创建高品质封装的 PostScript、PDF、PNG 或 SVG 图表 Python 库。官网地址：http://home.gna.org/pychart/。

（7）PLPlot。PLPlot 是用于创建科学图表的跨平台软件包，以 C 类库为核心，支持各种语言（C、C++、FORTRAN、Java、Python 等）。官网地址：http://plplot.sourceforge.net/。

（8）Reportlab。Reportlab 用于绘制 PDF 报表。官网地址：http://www.reportlab.com/software/opensource/。

（9）VPython。VPython 是 Visual Python 的简写，它是一个 Python 3D 绘图模块。官网地址：http://www.vpython.org/index.html。

13.2 基于 tkinter 的图形绘制

13.2.1 基于 tkinter 画布绘图概述

Canvas（画布）是一个长方形的区域，用于图形绘制或复杂的图形界面布局，可以在画布上绘制图形、文字，放置各种组件和框架。

Canvas 组件对象包括下列方法（绘制函数），用于绘制各种图形对象。

create_arc()：绘制圆弧。

create_bitmap()：绘制位图。

create_image()：绘制位图图像。

create_line()：绘制线。

create_oval()：绘制椭圆。

create_polygon()：绘制多边形。

create_rectangle()：绘制矩形。

create_text()：绘制文本。

create_window()：绘制子窗口。

13.2.2 创建画布对象

画布的左上角为坐标原点(0,0)，右下角为画布的大小(x,y)。创建画布对象的语法格式如下：

```
c = tkinter.Canvas(parent, option=value, …)
```

其中，parent 为父组件，默认无；option 为选项，通过如下命令可列举其选项：

```
>>> from tkinter import *
>>> c = Canvas(); c.keys()
['background', 'bd', 'bg', 'borderwidth', 'closeenough', 'confine',
'cursor', 'height', 'highlightbackground', 'highlightcolor',
'highlightthickness', 'insertbackground', 'insertborderwidth',
'insertofftime', 'insertontime', 'insertwidth', 'offset', 'relief',
'scrollregion', 'selectbackground', 'selectborderwidth',
'selectforeground', 'state', 'takefocus', 'width', 'xscrollcommand',
'xscrollincrement', 'yscrollcommand', 'yscrollincrement']
```

【例 13.1】 创建一个大小为 200×100、背景为黄色的画布（canvas.py）。程序运行结果如图 13-1 所示。

```
from tkinter import *
root = Tk()
c = Canvas(root,bg = 'yellow', width=200, height=100);
```

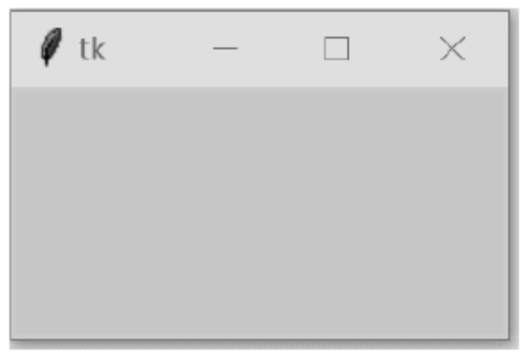

图 13-1 创建画布

```
#创建200×100、背景为黄色的画布
c.pack()
```

13.2.3 绘制矩形

在 Canvas 对象 c 上绘制线条（直线或折线）的方法如下：

```
id = c.create_rectangle(x0, y0, x1, y1, option, …)
```

其中，(x0, y0)是左上角的坐标；(x1, y1)是右下角的坐标。

【例 13.2】 矩形绘制示例（create_rectangle.py）。程序运行结果如图 13-2 所示。

图 13-2 绘制矩形

```
from tkinter import *
root = Tk()
c = Canvas(root,bg = 'white', width=130, height=70); c.pack()
                                                    #创建并显示Canvas
c.create_rectangle(10,10,60,60,fill='red')  #红色填充矩形
c.create_rectangle(70,10,120,60,fill='red',outline='blue',width=5)
                                                    #蓝色边框红色填充矩形，边框宽度5
```

13.2.4 绘制椭圆

在 Canvas 对象 c 上绘制椭圆的对象方法如下：

```
id = c.create_oval(x0, y0, x1, y1, option, …)
```

其中，(x0, y0)是左上角的坐标；(x1, y1)是右下角的坐标。

【例 13.3】 椭圆绘制示例（create_oval.py）。程序运行效果如图 13-3 所示。

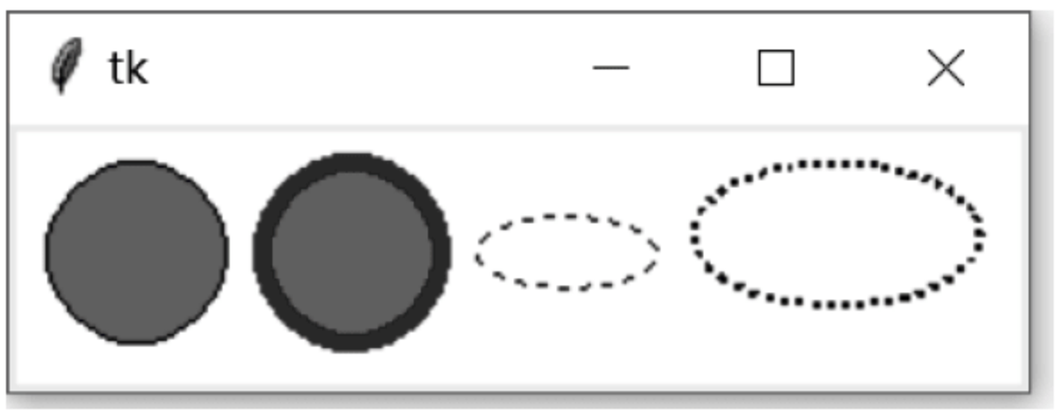

图 13-3 绘制椭圆

```
from tkinter import *
root = Tk()
c = Canvas(root,bg = 'white', width=280, height=70); c.pack()
                                                    #创建并显示Canvas
c.create_oval(10,10,60,60,fill='red')       #红色填充椭圆
c.create_oval(70,10,120,60,fill='red',outline='blue',width=5)
                                                    #红色填充蓝色边框宽度5的椭圆
c.create_oval(130,25,180,45,dash=(4,))          #虚线椭圆
c.create_oval(190,10,270,50,dash=(4,),width=2)  #宽度为2的虚线椭圆
```

13.2.5 绘制圆弧

在 Canvas 对象 c 上绘制圆弧的对象方法如下：

```
id = c.create_arc(x0, y0, x1, y1, option, …)
```

其中，(x0, y0)是左上角的坐标；(x1, y1)是右下角的坐标；option 为选项，其中，选项 start（开始角度，默认为 0）和 extent（圆弧角度，从 start 开始逆时针，默认为 90）决定圆弧的角度范围，选项 style 用于设置圆弧的样式，取值为：tk.PIESLICE（默认值）、tk.CHORD 和 tk.ARC。对应的形状样式如图 13-4 所示。

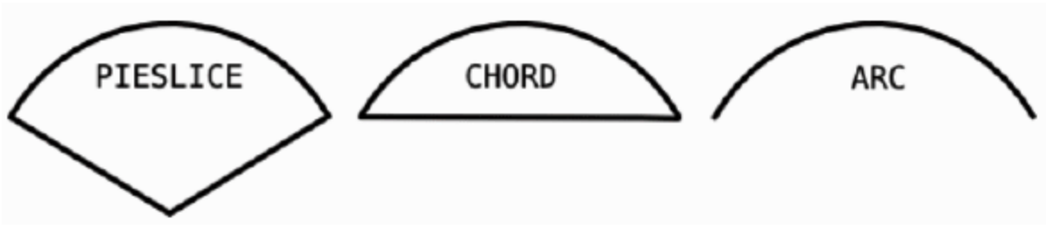

图 13-4　圆弧形状样式

【例 13.4】 圆弧绘制示例（create_arc.py）。程序运行效果如图 13-5 所示。

图 13-5　绘制圆弧

```
from tkinter import *
root = Tk()
c = Canvas(root,bg = 'white', width=250, height=70); c.pack()
                                                    #创建并显示Canvas
c.create_arc(10,10,60,60, style=PIESLICE)           #绘制PIESLICE样式圆弧
c.create_arc(70,10,120,60, style=CHORD)             #绘制CHORD样式圆弧
c.create_arc(130,10,180,60, style=ARC)              #绘制ARC样式圆弧
for i in range(0, 360, 60):                         #绘制菊花瓣蓝色边框红色填充的图形
    c.create_arc(190,10,240,60,fill='red',outline='blue',start=i,extent=30)
```

13.2.6 绘制线条

在 Canvas 对象 c 上绘制线条（直线或折线）的对象方法如下：

```
id = c.create_line(x0, y0, x1, y1, …, xn, yn, option, …)
```

其中，(x0, y0)、(x1, y1)、…、(xn, yn)是线条上各个点的坐标。

【例 13.5】 线条绘制示例（create_line.py）。程序运行效果如图 13-6 所示。

图 13-6　绘制线条

```
from tkinter import *
root = Tk()
c = Canvas(root,bg = 'white', width=250, height=70); c.pack()
                                                    #创建并显示Canvas
c.create_line(10,10,60,60,arrow=BOTH,arrowshape=(3,5,4))    #双向箭头
c.create_line(70,10,95,10,120,60)                           #折线
c.create_line(130,10,180,10,130,60,180,60,width=10,arrow=BOTH)
                                                    #Z字形双向箭头
c.create_line(190,10,240,10,190,60,240,60,width=10,joinstyle=MITER)
                                                    #Z字形
```

13.2.7　绘制多边形

在 Canvas 对象 c 上绘制多边形的对象方法如下：

```
id = c.create_polygon(x0, y0, x1, y1,…, option,…)
```

其中，(x0, y0)、(x1, y1)、…、(xn, yn)是多边形各个顶点的坐标。

【例 13.6】　多边形绘制示例（create_polygon.py）。程序运行效果如图 13-7 所示。

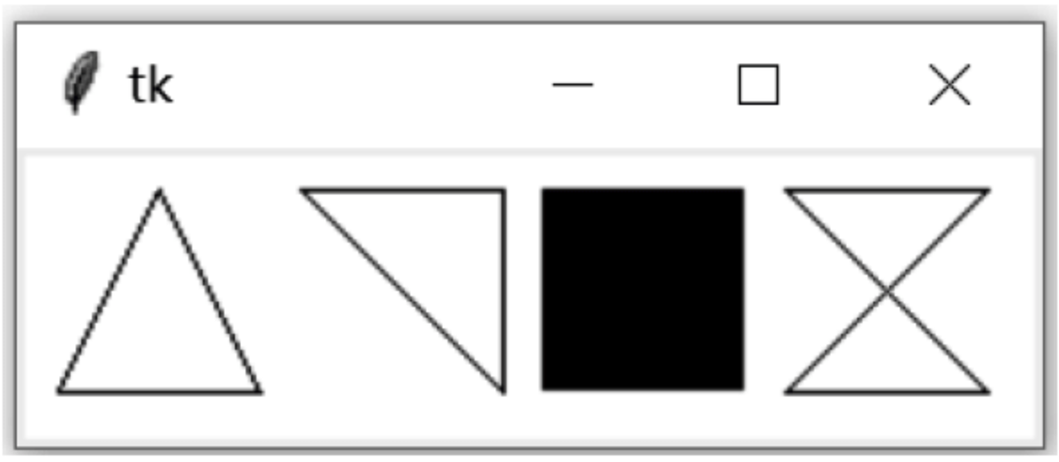

图 13-7　绘制多边形

```
from tkinter import *
root = Tk()
c = Canvas(root,bg = 'white', width=250, height=70); c.pack()
                                                    #创建并显示Canvas
c.create_polygon(35,10,10,60,60,60,fill='white',outline='black')
                                                    #黑边等腰三角形
c.create_polygon(70,10,120,10,120,60,fill='white',outline='black')
                                                    #黑边直角三角形
```

```
c.create_polygon(130,10,180,10,180,60,130,60)                    #黑色填充的正方形
c.create_polygon(190,10,240,10,190,60,240,60,fill='white',outline='black')
                                                                 #对顶三角形
```

13.2.8 绘制字符串

在 Canvas 对象 c 上绘制字符串的对象方法如下：

```
id = c.create_text(x, y, option, …)
```

其中，(x, y)是字符串放置的中心坐标；option 为选项，包括：activefill、activestipple、anchor、disabledfill、disabledstipple、fill、font、justify、offset、state、stipple、tags、text、width。其中，text 用于指定要绘制的字符串；font 用于指定字体；justify 用于指定对齐方式。

13.2.9 应用举例：函数图形

【例 13.7】 字符串和图形绘制（sin.py）：绘制给定函数 y=sin(x)的图形。程序运行效果如图 13-8 所示。

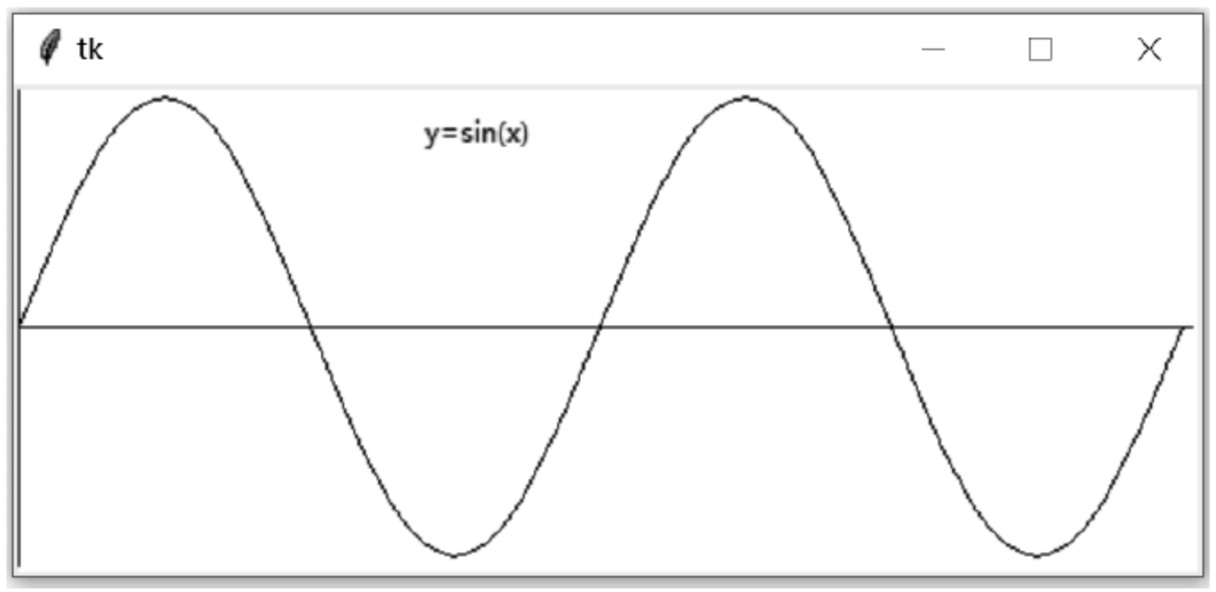

图 13-8 绘制函数图形

```
from tkinter import *
import math
WIDTH = 510; HEIGHT = 210                          #画布宽度、高度
ORIGIN_X = 2; ORIGIN_Y = HEIGHT / 2                #原点（X=2、Y=窗体左边中心）
SCALE_X = 40; SCALE_Y = 100                        #X轴、Y轴的缩放倍数
END_ARC = 360 * 2                                  #函数图形画多长（两个周期）
ox = 0; oy = 0; x = 0; y = 0                       #坐标初始值
arc = 0                                            #弧度
root = Tk()
c = Canvas(root,bg = 'white', width=WIDTH, height=HEIGHT);c.pack()
                                                   #创建并显示Canvas
c.create_text(200, 20, text='y=sin(x)')            #绘制文字
c.create_line(0, ORIGIN_Y, WIDTH, ORIGIN_Y)        #绘制Y纵轴
c.create_line(ORIGIN_X, 0, ORIGIN_X, HEIGHT)       #绘制X横轴
for i in range(0, END_ARC+1, 10):                  #绘制函数图形
```

```
        arc = math.pi * i * 2 / 360
        x = ORIGIN_X + arc * SCALE_X
        y = ORIGIN_Y - math.sin(arc) * SCALE_Y
        c.create_line(ox, oy, x, y)
        ox = x; oy = y
```

13.3 基于 turtle 模块的海龟绘图

13.3.1 海龟绘图概述

所谓的海龟绘图，即假定一只海龟（海龟带着一支钢笔）在一个屏幕上来回移动，当它沿直线移动时会绘制直线。海龟可以沿直线移动指定的距离，也可以旋转一个指定的角度。

通过编写代码，可以控制海龟移动和绘图，从而绘制出图形。使用海龟作图，不仅能够使用简单的代码创建出令人印象深刻的视觉效果，而且还可以跟随海龟，动态查看程序代码如何影响到海龟的移动和绘制，从而帮助我们理解代码的逻辑。

13.3.2 turtle 模块概述

Python 标准库中的 turtle 模块基于 tkinter 的画布，实现了海龟绘图的功能。使用 turtle 模块绘图，一般遵循如下步骤。

（1）导入 turtle 模块。

```
from turtle import *                    #将turtle中的所有方法导入
```

（2）创建海龟对象（turtle 模块同时实现了函数模式，故也可以不创建海龟对象，直接调用函数，直接绘图）。

```
p = Turtle()                            #创建海龟对象
```

（3）设置海龟的绘图属性（画笔的属性，颜色、画线的宽度）。

```
pensize(width)/width(width)             #绘制图形时的宽度
color(colorstring)                      #绘制图形时的画笔颜色和填充颜色
pencolor(colorstring)                   #绘制图形的画笔颜色
fillcolor(colorstring)                  #绘制图形的填充颜色
```

（4）控制和操作海龟绘图。

```
pendown()/pd()/down()                   #移动时绘制图形，默认时为绘制
penup()/pu()/up()                       #移动时不绘制图形
forward(distance)/fd(distance)          #向前移动distance指定的距离
backward(distance)/bk(distance)/back(distance)  #向后移动distance指定的距离
right(angle)/rt(angle)                  #向右旋转angle指定的角度
left(angle)/lt(angle)                   #向左旋转angle指定的角度
goto(x,y)/setpos(x,y)/setposition(x,y)  #将画笔移动到坐标为(x,y)的位置
```

```
dot(size=None, *color)                          #绘制指定大小的圆点
circle(radius, extent=None, steps=None)#绘制指定大小的圆
write(arg, move=False, align='left', font=('Arial', 8, 'normal'))
                                                #绘制文本
stamp()                                         #复制当前图形
speed(speed)                                    #画笔绘制的速度（[0,10]之间的整数）
showturtle()/st()                               #显示海龟
hideturtle()/ht()                               #隐藏海龟
clear()                                         #清除海龟绘制的图形
reset()                                         #清除海龟绘制的图形并重置海龟属性
```

13.3.3 绘制正方形

【例 13.8】 使用海龟绘图绘制一个正方形（square.py）。最终结果如图 13-9 所示。

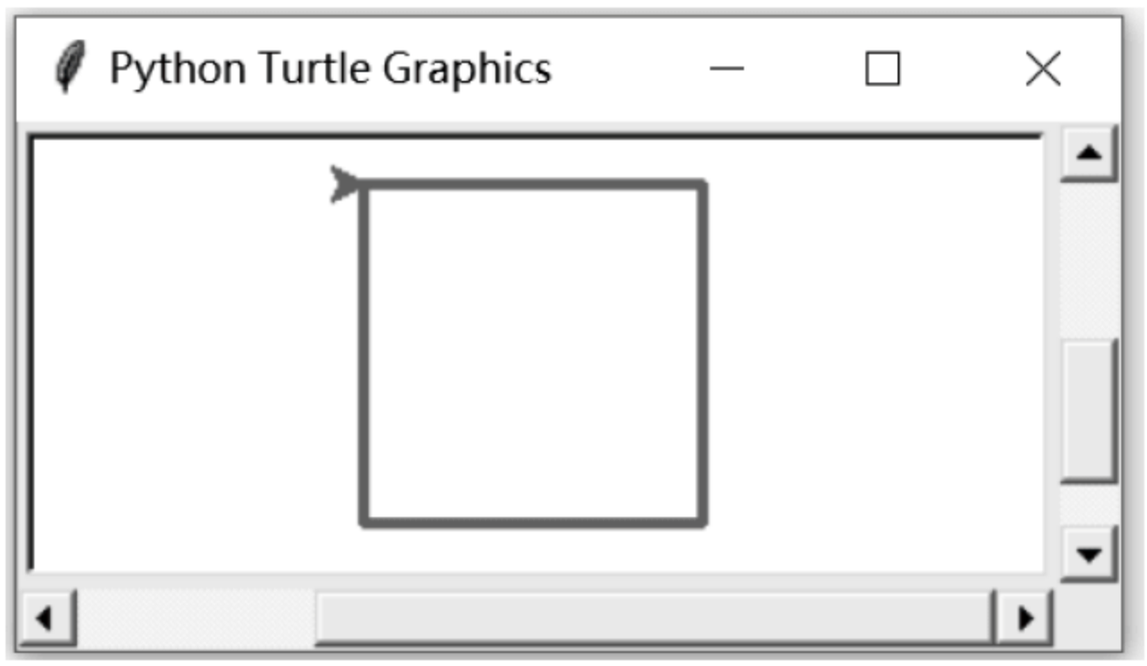

图 13-9 使用海龟绘图绘制一个正方形

```
import turtle
p = turtle.Turtle()                             #创建海龟对象
p.color("red")                                  #设置绘制时画笔的颜色
p.pensize(3)                                    #定义绘制时画笔的线条宽度
turtle.speed(1)                                 #定义绘图的速度（"slowest"或者1均表示最慢）
p.goto(0,0)                                     #移动海龟到坐标原点(0,0)
for i in range(4):                              #绘出正方形的4条边
   p.forward(100)                               #向前移动100
   p.right(90)                                  #向右旋转90°
```

说明：海龟绘图时，其原点(0,0)位于画布区域中央位置。

13.3.4 绘制多边形

【例 13.9】 使用海龟绘图绘制三角形、正方形、正五边形、……、正十边形等多边形（polygon.py）。最终结果如图 13-10 所示。

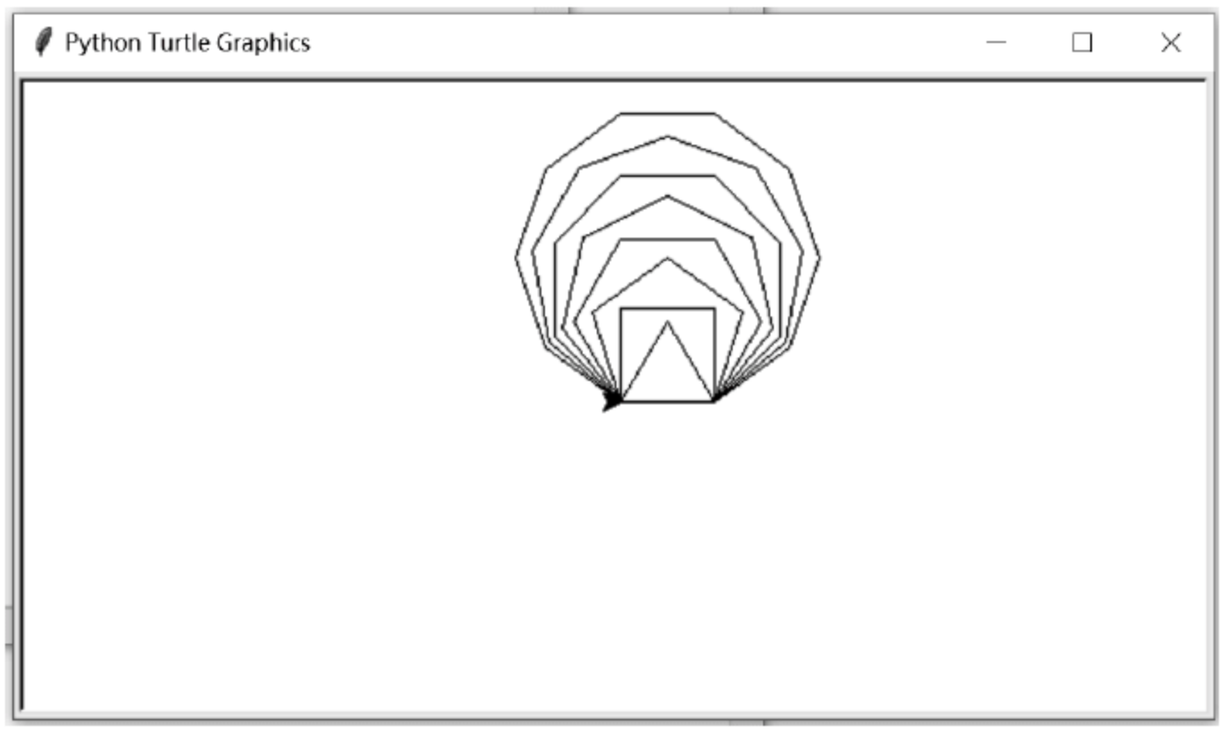

图 13-10　使用海龟绘图绘制各种多边形

```
import turtle
def draw_polygon(sides, side_len):  #绘制指定边长度的多边行
    for i in range(sides):
        turtle.forward(side_len)      #绘制边长
        turtle.left(360.0/sides)      #旋转角度
def main():
    for i in range(3,11):             #绘制三角形、正方形、正五边形、……、正十边形
        step = 50                     #边长（海龟步长）为50
        draw_polygon(i, step)         #绘制多边形
if __name__ == '__main__': main()
```

说明：本示例直接调用 turtle 模块的海龟绘图函数，没有创建海龟对象。

13.3.5　绘制圆形螺旋

【例 13.10】 使用海龟绘图分别绘制红、蓝、绿、黄 4 种颜色的圆形螺旋（spiral.py）。最终结果如图 13-11 所示。

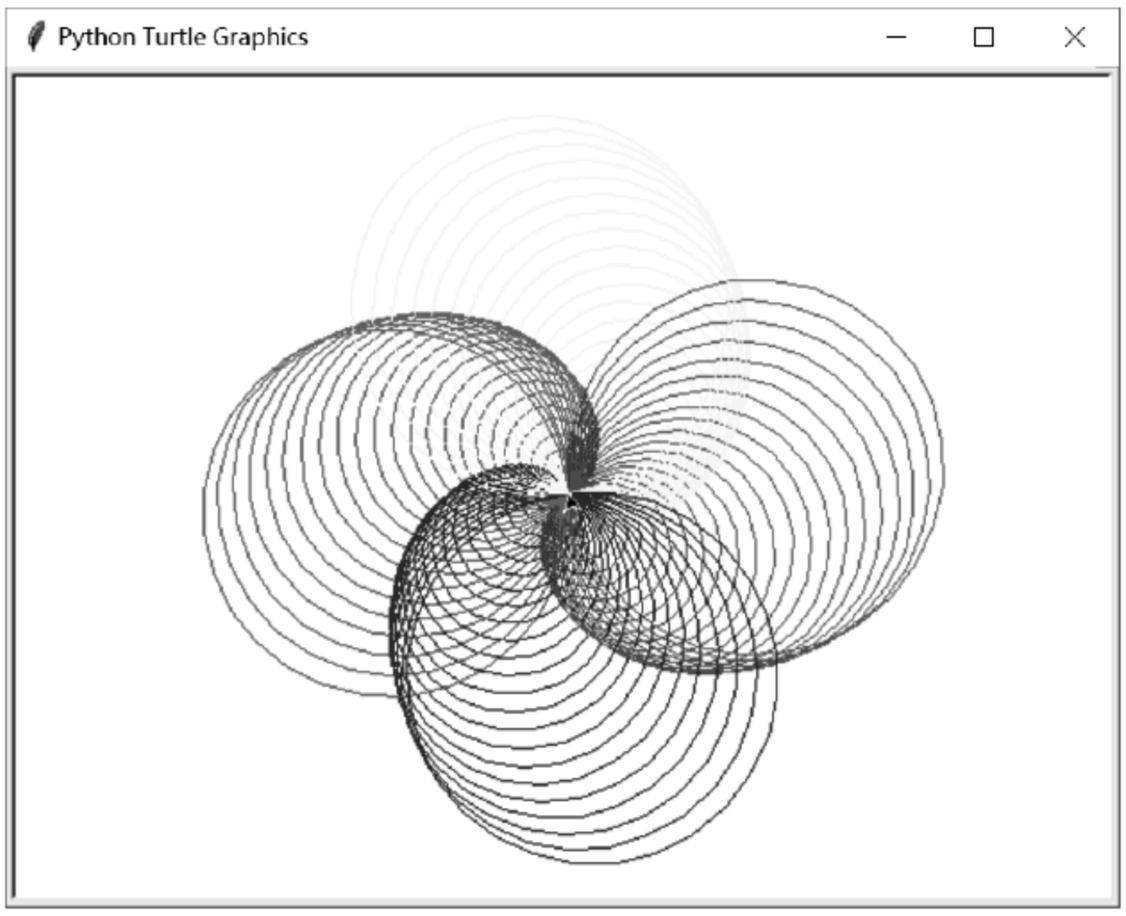

图 13-11　使用海龟绘图绘制 4 种颜色的圆形螺旋

```
import turtle
p = turtle.Turtle()                          #创建海龟对象
p.speed(0)                                   #定义绘图的速度（"fastest"或者0均表示最快）
colors = ["red", "blue", "green", "yellow"] #红、蓝、绿、黄4种颜色
for i in range(100):                         #i=0~99
    p.pencolor(colors[i%4])                  #设置画笔颜色（红或蓝或绿或黄）
    p.circle(i)                              #画圆
    p.left(91)                               ##向左旋转91°
```

13.3.6 递归图形

分形（Fractal）概念由法国数学家曼德布罗在 1975 年提出，用于形容局部与整体相似的形状。分形图可以使用简单的递归绘图方案实现，从而产生复杂的图像。分形图可以模拟自然界的树、蕨类、云等。

【例 13.11】 科赫曲线的绘制（Koch.py）。

n 阶科赫曲线的递归绘制算法步骤如下。

（1）基本情况（当 n=0 时）：绘制一条直线。

（2）递归步骤（当 n≥1 时）：绘制一条阶数为 n-1 的科赫曲线；向左旋转 60°，绘制第二条阶数为 n-1 的科赫曲线；向右旋转 120°，绘制第三条阶数为 n-1 的科赫曲线；向左旋转 60°，绘制第四条阶数为 n-1 的科赫曲线。

n 阶科赫曲线的绘制线条长度是 n-1 阶科赫曲线长度的 1/3。

3 阶科赫曲线的绘制结果如图 13-12 所示。

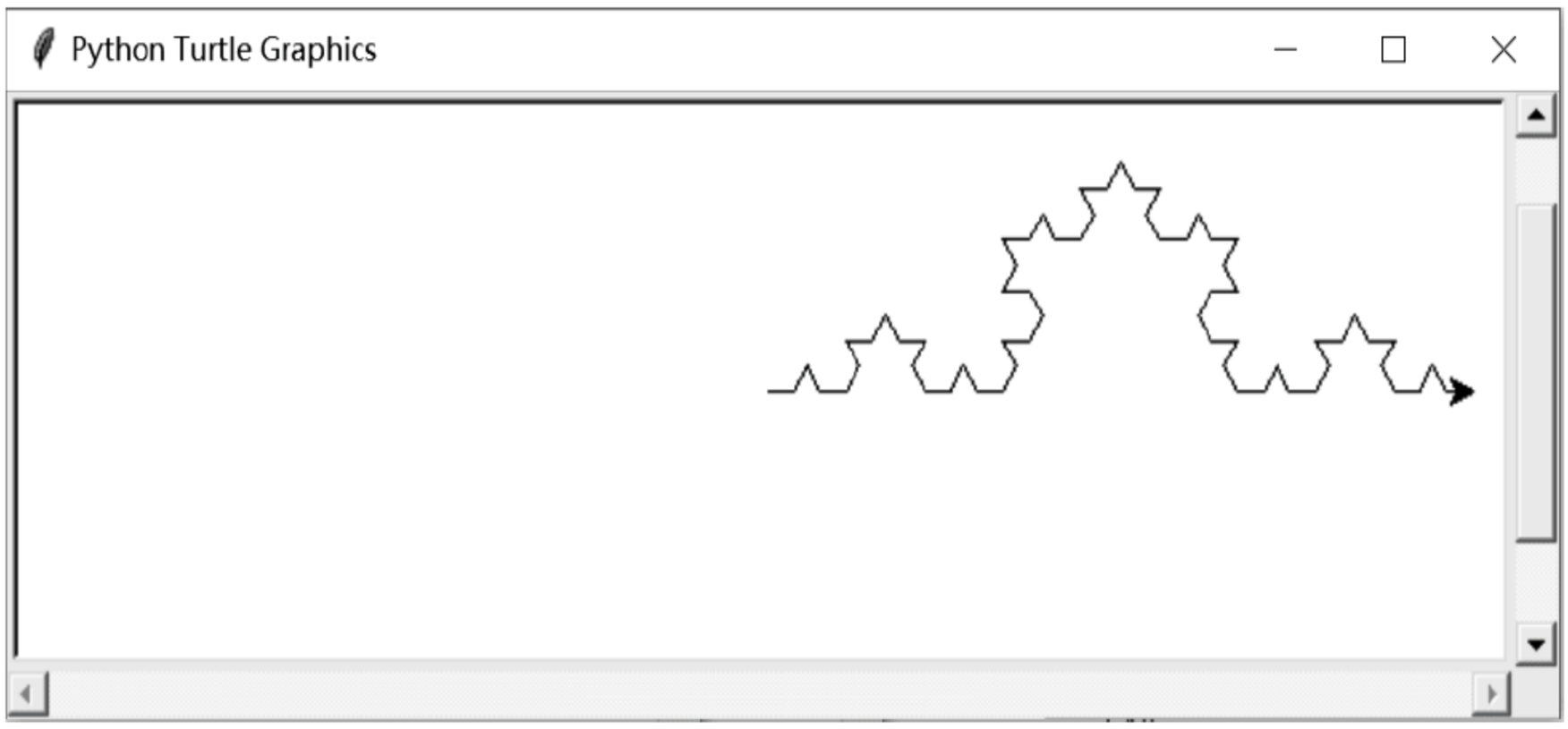

图 13-12　使用海龟绘图绘制 n 阶科赫曲线

```
import sys
import turtle
def koch(t, order, size):
        if order == 0:                       #当n等于0时，绘制一条直线
            t.forward(size)
```

```
            else:                                  #否则，递归绘制n阶科赫曲线
                koch(t, order-1, size/3)  #递归绘制一条阶数为n-1的科赫曲线，长度1/3
                t.left(60.0)                       #向左旋转60°
                koch(t, order-1, size/3)  #递归绘制一条阶数为n-1的科赫曲线，长度1/3
                t.right(120.0)  #t.left(-120.0) #向右旋转120°（或者向左旋转-120°）
                koch(t, order-1, size/3)  #递归绘制一条阶数为n-1的科赫曲线，长度1/3
                t.left(60.0)                       #向左旋转60°
                koch(t, order-1, size/3)  #递归绘制一条阶数为n-1的科赫曲线，长度1/3
    def main():
            n = int(sys.argv[1])
            step = 300
            p = turtle.Turtle()
            koch(p, n, step)
    if __name__ == '__main__': main()
```

13.3.7 海龟绘图的应用实例

【例 13.12】 使用 Python Turtle Demo 学习海龟绘图的应用实例。

（1）运行 Python 内置集成开发环境 IDLE。

（2）运行 Python Turtle Demo。执行 IDLE 菜单命令 Help | Turtle Demo，打开 Turtle Demo 源代码编辑器，如图 13-13 所示。

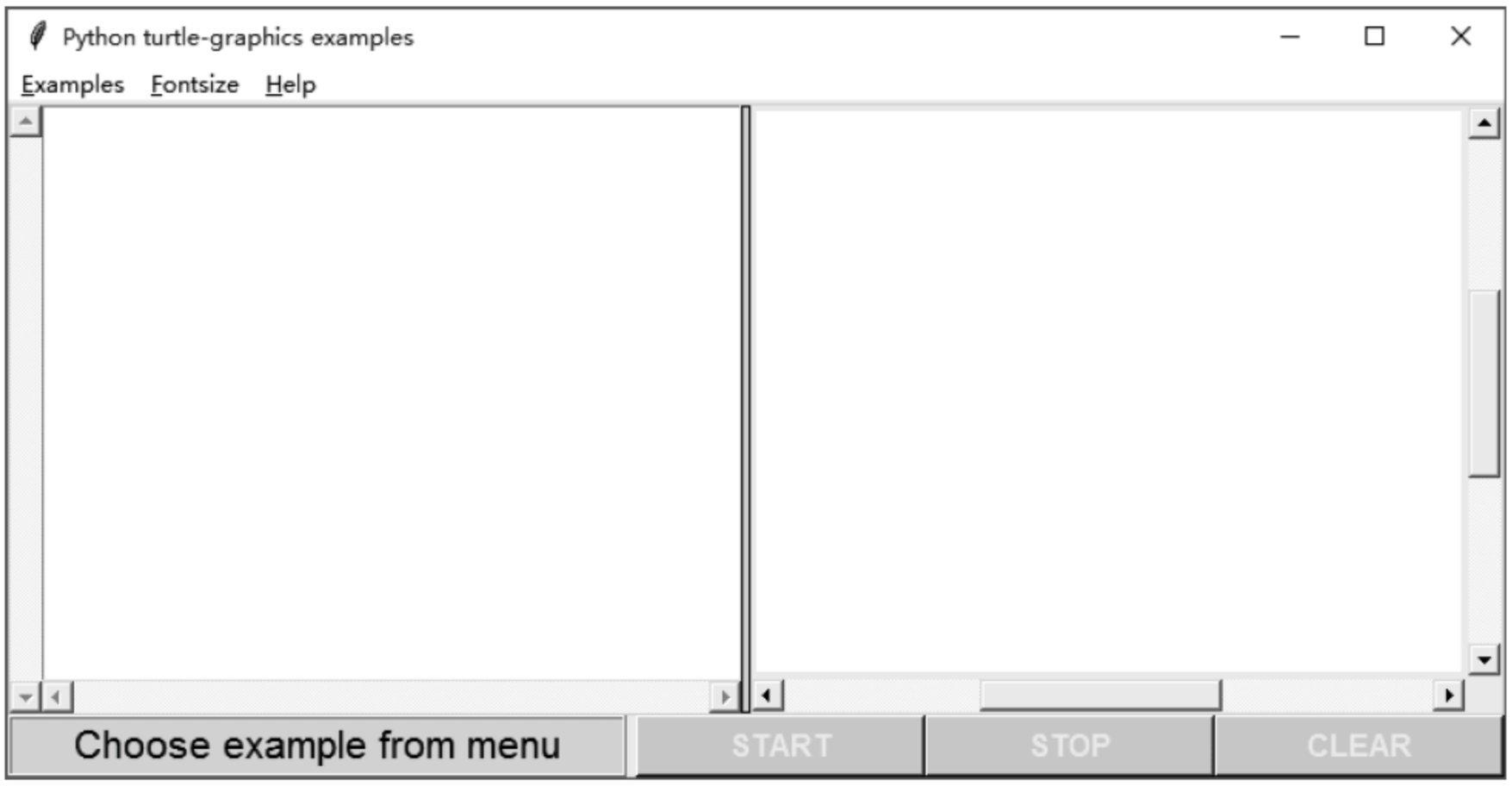

图 13-13　Turtle Demo 源代码编辑器

（3）查看海龟绘图示例 clock。执行 Turtle Demo 菜单命令 Examples | clock，打开 clock 示例，左边窗格显示源代码。单击窗口下部的 START 按钮，查看程序运行结果，如图 13-14 所示。

（4）查看其他海龟绘图示例。参照步骤（3），查看其他示例。

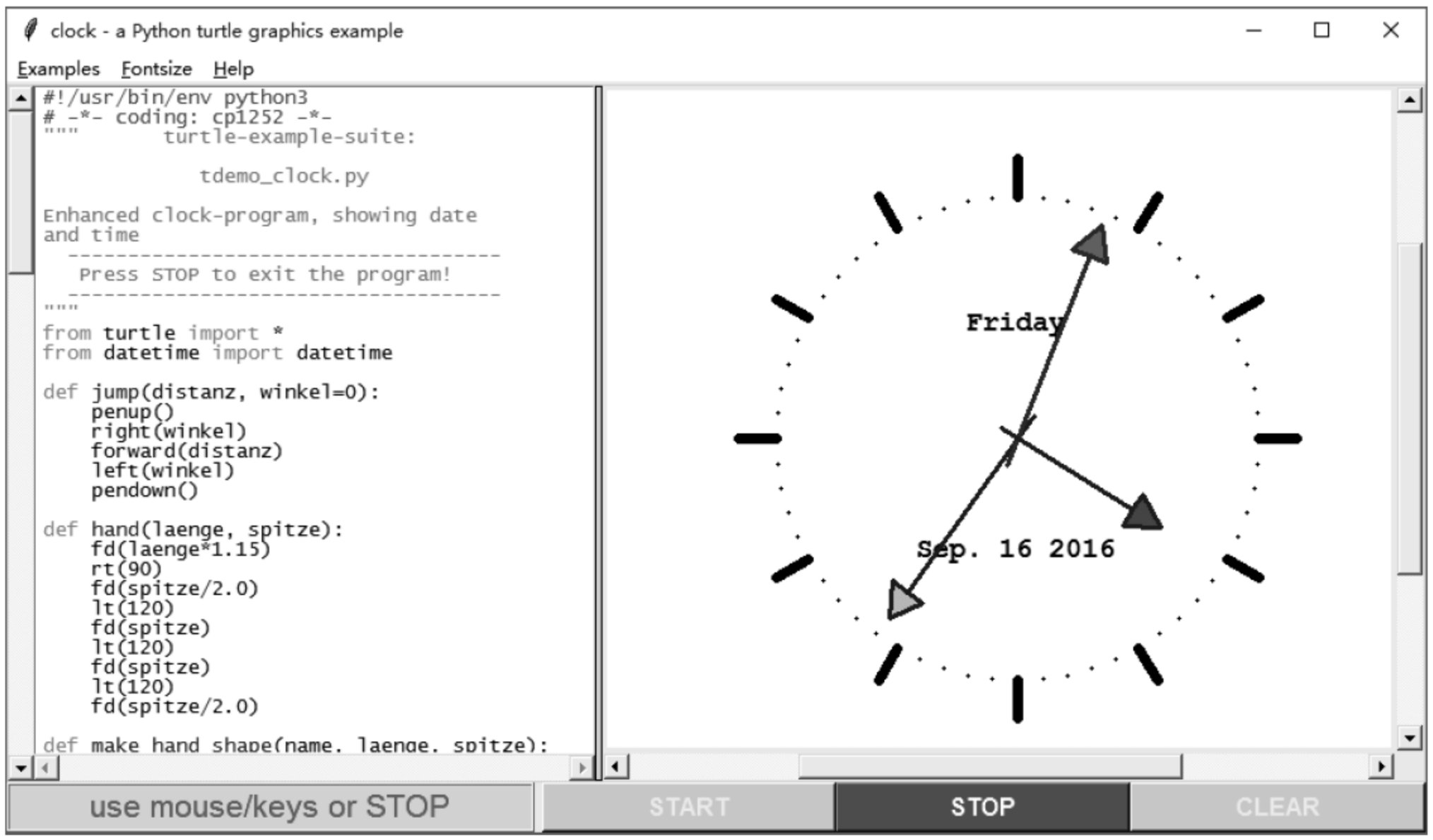

图 13-14　查看海龟绘图示例 clock

13.4　基于 Matplotlib 模块的绘图

13.4.1　Matplotlib 模块概述

Matplotlib 是 Python 最著名的绘图库之一，提供了一整套和 MATLAB 相似的命令 API，既适合交互式地进行制图，也可以作为绘图控件方便地嵌入 GUI 应用程序中。

Matplotlib 的 pyplot 子库提供了和 MATLAB 类似的绘图 API，方便用户快速绘制 2D 图表，包括直方图、饼图、散点图等。

Matplotlib 配合 NumPy 模块使用，可以实现科学计算结果的可视化显示。

Matplotlib 的文档相当完备，其官网的 Gallery 页面（http://matplotlib.org/gallery.html）中包括各种类型图形的示例图，如图 13-15 所示。选择需要绘制某种类型的图形，可以查看相应的源代码，复制修改源代码后满足实际需求。其官网的 examples 页面（http://matplotlib.org/examples.html）中包含大量的示例程序。

13.4.2　安装 Matplotlib 模块

Matplotlib 的官网地址是 http://matplotlib.org/。可以直接从官网下载安装 Matplotlib 模块。

建议使用 Python 安装和管理包 pip，安装 Matplotlib 模块。详细步骤请参照本教程第 1 章的相关内容。

图 13-15　Matplotlib 官网的 Gallery 页面

13.4.3　使用 Matplotlib 模块绘图概述

使用 Matplotlib 模块绘图，主要使用了 Matplotlib.pyplot 工具包。

Matplotlib 是一套面向对象的绘图库，其绘制的图表中的每个绘图元素（例如线条、文字、刻度等）都是对象。

为了实现快速绘图，Matplotlib 的子模块 pyplot 提供了与 MATLAB 类似的绘图 API，封装了复杂的绘图对象结构。用户只需要调用 pyplot 模块所提供的函数，就可以实现快速绘图，并设置图表的各种细节。

Matplotlib.pyplot 是命令行式函数的集合，每一个函数都对图像做了修改，例如创建图形、在图像上创建画图区域、在画图区域上画线、在线上标注等。

【例 13.13】 使用 plot()函数画图（linecurve.py）：绘制 x 轴坐标值为 0、1、2、3、4，所对应的 y 轴坐标值为 1、2、5、6、8 的折线图。

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 5, 6, 8])
plt.ylabel('some numbers')    #为y轴加注释
plt.show()
```

程序运行结果如图 13-16 所示。

13.4.4　绘制函数曲线

【例 13.14】 使用 Matplotlib 模块绘制 y=sin(x)的函数曲线（sine.py）。

```
import matplotlib.pyplot as plt
import math
x = [2*math.pi*i/100 for i in range(100)]
y = [math.sin(i) for i in x]
```

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

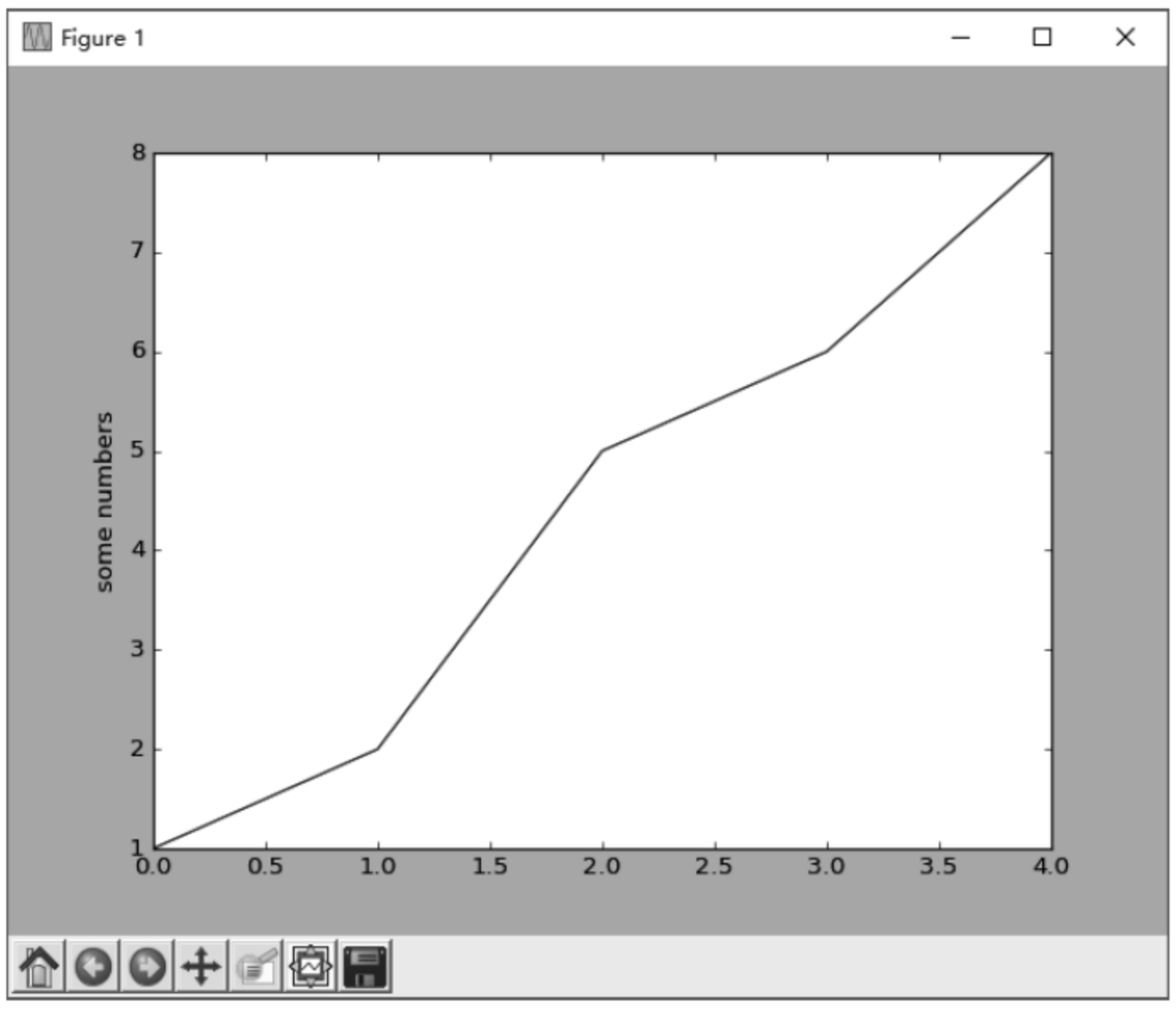

图 13-16　使用 plot()函数画折线图

程序运行结果如图 13-17 所示。

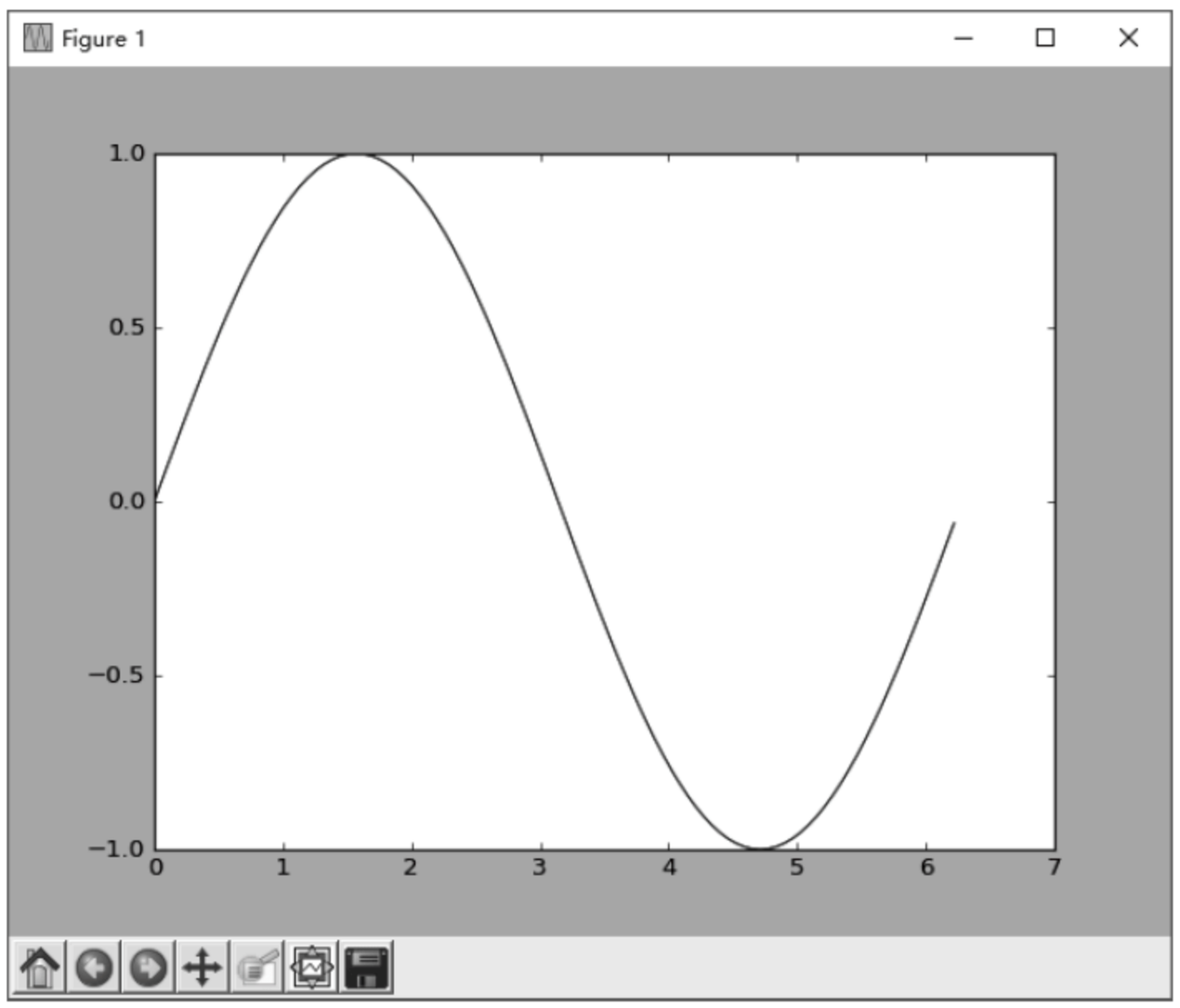

图 13-17　使用 Matplotlib 模块绘制 y=sin(x)的函数曲线

13.4.5　绘制多个图形

pyplot 和 MATLAB 一样，支持绘制多个子图。figure()命令用于指定图形，subplot()命令用于指定一个坐标系。例如，figure(1)（默认）指定图形 1，subplot(211)指定子图 1（上

下两个子图的第一幅图像），subplot(212)指定子图 2（上下两个子图的第二幅图像）。指定了子图后，所有绘图命令都是针对当前图像和当前子图。

【例 13.15】 绘制多个子图示例（multifig.py）。利用 NumPy 模块和 Matplotlib.pyplot 工具包绘制 $y=e^{-t}\times\cos(2\pi x)$以及 $y=\cos(2\pi x)$的函数曲线。

```
import numpy as np
import matplotlib.pyplot as plt
def f(t):
  return np.exp(-t) * np.cos(2*np.pi*t)
t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)
plt.figure(1)
plt.subplot(211)
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')
plt.subplot(212)
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
plt.show()
```

程序运行结果如图 13-18 所示。

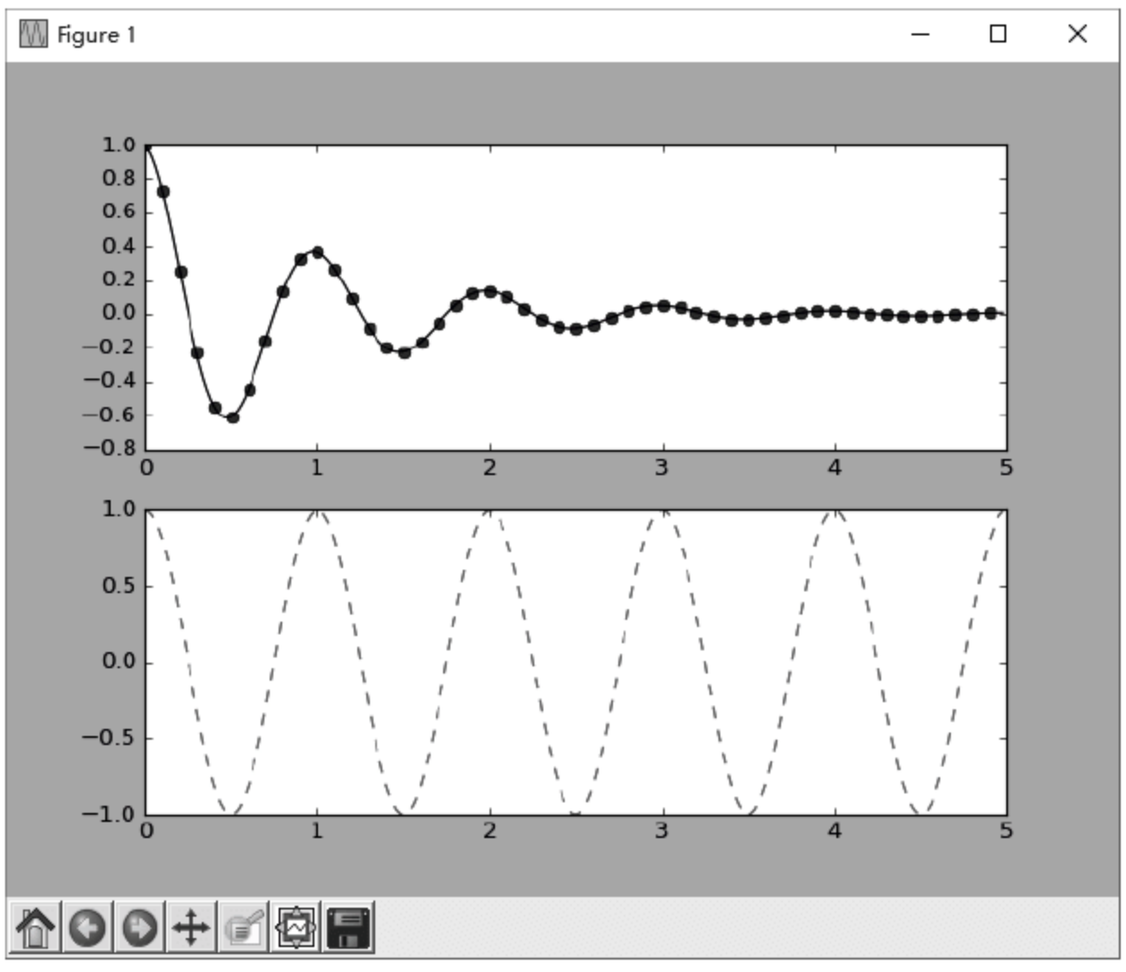

图 13-18　使用 Matplotlib 模块绘制多个子图

说明：程序中使用了 NumPy 模块，用于生成用于绘制的图形数据。

13.4.6　绘制直方图

直方图是由一系列高度不等的纵向条纹或线段表示数据分布的统计报告图。使用 Matplotlib.pyplot 的 hist()函数，可以方便快捷地绘制直方图。

【例 13.16】 使用 Matplotlib.pyplot 的 hist()函数绘制直方图示例（histfig.py）：随机生成满足 mu 为 100、sigma 为 20 的正态分布的 10 万个智商数据，并绘制其直方图。

```
import numpy as np
import matplotlib.pyplot as plt
#随机生成满足mu为100、sigma为20的正态分布的10万个智商数据
mu, sigma = 100, 20
x = mu + sigma * np.random.randn(100000)
#绘制直方图
plt.hist(x, 50, normed=1, facecolor='g', alpha=0.75)
#绘制坐标等信息
plt.xlabel('IQ');plt.ylabel('Probability')
plt.title('Histogram of IQ')
#设置坐标和网格
plt.axis([40, 180, 0, 0.03])
plt.grid(True)
plt.show()
```

程序运行结果如图 13-19 所示。

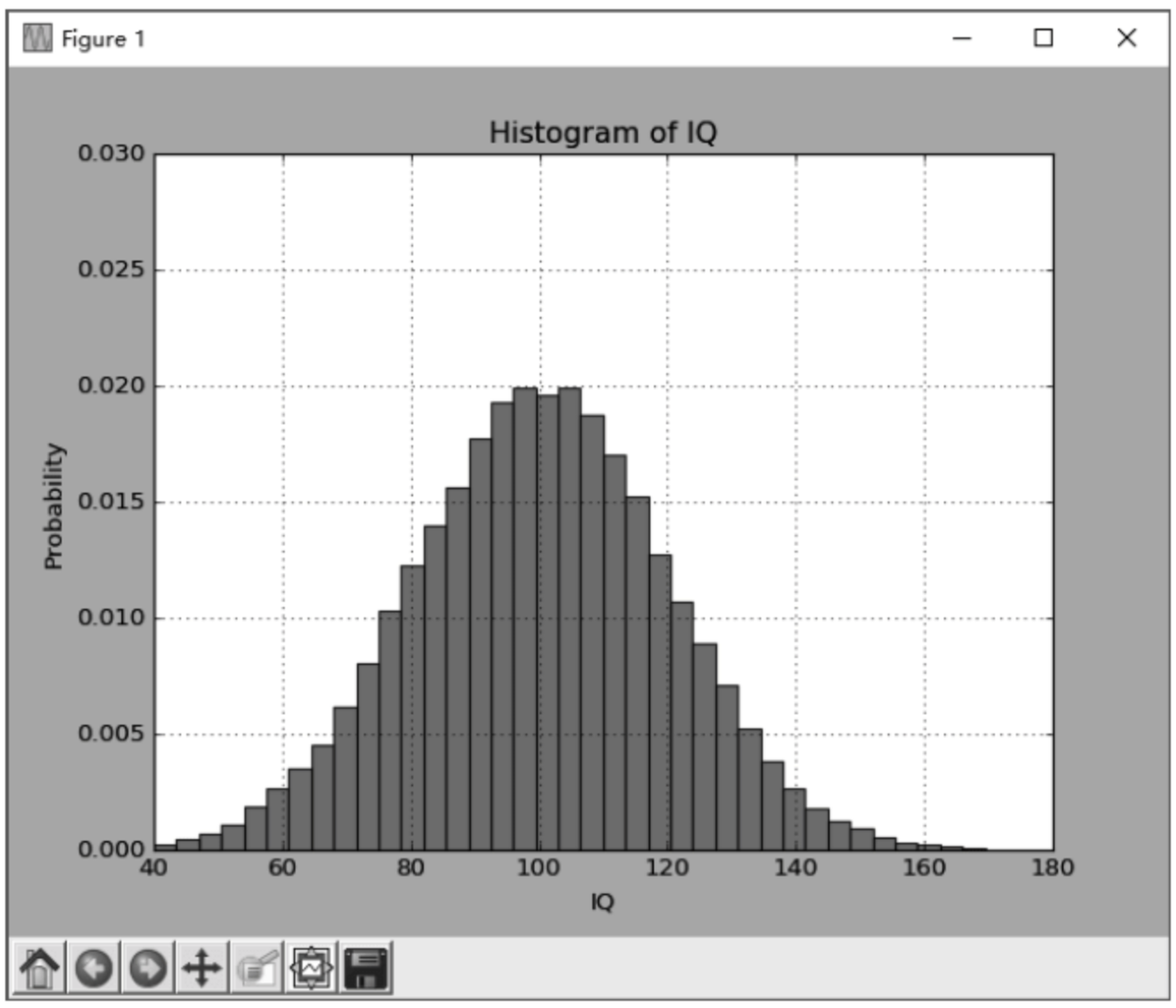

图 13-19　使用 Matplotlib.pyplot 的 hist()函数绘制直方图

复　习　题

1．Python 标准库中包括图形绘制相关模块，其中，模块_____用于画布绘图，模块_____用于海龟绘图。

2．_____是一个长方形的区域，用于图形绘制或复杂的图形界面布局。可以在其上绘制图形、文字，放置各种组件和框架。

3．画布的_____为坐标原点(0,0)，_____为画布的大小(x,y)。

4．Matplotlib 是 Python 最著名的绘图库之一，其_____子库（子模块）提供了和 MATLAB 类似的绘图 API，方便用户快速绘制 2D 图表，包括直方图、饼图、散点图等。