Python 语言程序设计基础(第 2 版)

Python Programming Language: An Ecosystem Practice

(2nd Edition)

嵩 天　　礼 欣　　黄天羽 著

高等教育出版社 北京

本书特色

本书提出了以理解和运用计算生态为目标的 Python 语言教学思想和学习路径。

与传统编程语言学习不同，本书不仅强调学习 Python 语言基本语法，同时强调掌握运用 Python 函数库的能力。

本书绝大部分实例将令人激动，原来编程语言会这么有趣！

内容提要

本书提出了以理解和运用计算生态为目标的Python语言教学思想，不仅系统讲解了Python语言语法，同时介绍了从数据理解到图像处理的14个Python函数库，向初学Python语言读者展示了全新的编程语言学习路径。全书一共设计了25个非常具有现代感的实例，从绘制蟒蛇、理解天天向上的力量到机器学习、网络爬虫，从文本进度条、统计名著人物重要性到图像手绘效果、雷达图绘制，绝大多数实例为作者原创，将随着内容深入不断激发读者学习Python语言的热情，因为“编程是件很有趣的事儿”。

本书内容丰富、叙述清晰、循序渐进，采用新形态构建，提供大量扩展阅读资料、学习资料和学习视频。本书作者作为中国大学MOOC平台“Python语言程序设计”课程的主讲教师，建议广大读者借助在线开放课程深入学习本书内容。本书适合初学Python语言读者使用，也适合各类大专院校开展教学，同时也可以作为对Python感兴趣读者的自学指导书。

第 1 版前言

本书是在国内外广泛关注且推进“计算思维”教学理念的大背景下编写的，在该书成稿之时，如何将“计算思维”理念转化为大学计算机基础课程的教学内容仍处于探讨中。无论“计算思维”的内涵和外延如何，具有“计算思维”的学习者应该能够深刻理解问题的计算特性并善于利用计算机解决问题。本书以此为出发点，期望实现两个目标：使读者掌握一门终身受用的程序设计语言（Python 语言）；使读者体验利用程序设计语言解决实际问题的过程和思路。

选择 Python 语言作为“终身受用的程序设计语言”来教学并非因为作者在 10 年前就接触并使用它，而是因为 Python 语言是一种简洁且强大的语言。相比其他高级语言，它的语法简介质朴，可以用优美来形容。最关键的，它是一种开源的脚本语言，这个特点促使世界上出现了最大的围绕 Python 程序设计的开放社区。至今，该社区已经提供了超过 3 万个不同功能的开源函数库，为基于 Python 语言的快速开发提供了强大支持。

超凡脱俗、简洁优美、功能强大、平台无关、经济实惠等词语都可以用来形容 Python 语言，但这些还不够，Python 语言是仅次于 C 语言的第二语言。而更通常的情况是，如果程序不是以执行性能为首要设计目标，Python 语言是首选。

Python 语言是一门非常简单易学的语言。作者曾经设计过“1 小时学 Python”的教学实验，实践证明，大多数没有任何程序设计基础的大一学生都可以在 1 小时内理解 Python 设计方法并具备十几行代码的编写能力。这曾让作者既喜且忧，喜在学生们似乎找到了一种简单易学、编写快速并能解决问题的合适语言，忧在至今 Python 语言还没有进入大学计算机基础课程的教学计划。本书是一个尝试，希望更多同行关注 Python 语言，在大学计算机基础课程教学中讲授 Python 语言，让学生们终身受益。

本书具有较强的现代气息，书中所涉及的问题不仅包括房屋贷款计算，还包括 PM2.5 雾霾预警、圆周率计算、GPS 定位和科赫曲线（分形几何）等等，读者在解决一个个问题的同时一定不会觉得乏味，因为这些问题就在身边。为了配合读者学习

或高校教师开展 Python 语言教学，本书通过配套网站提供电子资源，网址为 www.python123.org，内容包括：教学用 PPT，更多习题和答案，更多程序设计问题和实例，在线程序测试平台，读者在线问题解答等。

北京理工大学计算机学院李凤霞教授是本书的主审，她的直觉、睿智和丰富经验启发了作者对教学 Python 程序设计语言的思路。还要感谢在本书撰写和出版过程中给予帮助的人，包括研究生史湘君、骆世瑛、徐金楠、万云凯、李玮、陈潇、张运大、刘翼和易琳等，以及北京理工大学孙新老师，他们对本书部分段落和例子都有实际贡献。本书得到了北京市教育委员会“北京高等学校青年英才计划项目”的资助，在此一并感谢。限于水平，错误之处在所难免，敬请读者和同行批评指正。作者的电子邮件地址是 songtian@bit.edu.cn。

第 2 版前言

（尚在酝酿中）

序言

本书无序言。

目录

第 9 章 人机接口和图形编程

相比其他技术领域，美对于计算来说更为重要，因为软件超乎寻常的复杂，而美是对复杂性的一种终极防御。

Beauty is more important in computing than anywhere else in technology because software is so complicated. Beauty is the ultimate defence against complexity.

——大卫·盖勒特(David Gelernter)

美国艺术家、作家、耶鲁大学计算机科学系教授

学习目标

1. 了解计算机图形学的概念；
2. 掌握图形编程的基本方法；
3. 掌握交互式图像编程的方法；
4. 了解鼠标、键盘、文本输入框的工作原理；
5. 掌握事件的原理和信号触发机制；
6. 运用交互式编程进行界面开发；
7. 掌握艺术图像的绘制方法。

从 DOS 命令行到 Windows 窗口，图形用户界面引领了计算机交互行业一次又一次革命。如今，超市的购物篮演变成了网络上的购物车，各种聊天工具的对话框逐渐替代了面对面的眼神交流，任何信息只要点按发送就嗖的弹射出去，历史记录无论何时都忠实地保存着用户的一言一行。这些已经见怪不怪的功能蕴含了计算机交互功能的规律。

一起来了解这些规律，咱们也编个聊天界面与 QQ 来个正面对决！

9.1 问题概述

要点：人机接口是人与计算机的交互方式，GUI 是主要手段之一。

计算机技术的快速发展使人类经历了对计算能力惊喜、熟悉和习惯的过程，从单一追求计算能力逐步向追求用户体验过渡。图形图像应用已经成为程序不可分割的重要部分。

计算机图形学原理支撑了现代计算机很多图形图像应用。大部分熟悉的应用程序都有图形用户界面，这是侠义上的软件界面。很多应用程序都由众多的图形对象来支持，例如，办公软件 powerpoint 等、照片管理软件、视频播放软件都离不开图形的支持。没有图形，计算机的世界将不再丰富多彩！

人与计算机之间的交互方式称为人机接口。现代商用计算机程序大多提供良好的人机接口设计，以提高用户体验。最常用的图形用户界面（Graphical User Interface，简称 GUI）是人机接口的重要方式之一，它提供了可视化元素，如窗体、图标、按钮以及菜单等。

——Python 能编写带按钮和输入框的图形界面程序吗？

——可以，但要选择合适的第三方库。

拓展：人机接口（Human-Computer Interface）

人机接口是指人与计算机之间建立联系、交换信息的输入/输出设备的接口，这些设备包括键盘、显示器、打印机、鼠标等。

计算机系统的人机接口是用户与操作系统之间的桥梁，通过人机接口，用户只需进行简单操作，就能实现复杂的应用计算与处理。人机接口分为直接接口和间接程序接口两类。直接接口是用户通过交互命令、图形界面或网络界面直接对计算机或网络进行操作。QQ 软件的用户界面也是一种直接接口。间接接口供用户以程序方式进行操作。程序员使用操作系统或第三方提供的应用程序设计接口 API 来调用系统提供的例行程序，实现既定的操作。Windows 的程序接口以 Windows API 函数

为主。

Python 标准库内置了一个 GUI 库——tkinter,这个库基于 Tck/Tk 开发,然而,这个库十分陈旧，提供的开发控件也很有限，编写出来的 GUI 风格与现代程序 GUI 风格相差甚远，从用户体验角度说，tkinter 库并不成熟。

Python 语言的价值在于它可以使用大量的第三方库，GUI 开发也不例外。有 3 较好的 GUI 第三方库，分别是：PyQt5、wxpython 和 pygtk。这些库都提供了较多控件和较好的 GUI 风格,能够为用户带来良好的使用体验。本书采用 PyQt5 库介绍 GUI 开发，这个库是 Python 语言当前最好的 GUI 第三方库，它可以在 Windows、Linux 和 Mac OS X 等操作系统上跨平台使用，所提供的“信号槽”机制为编写事件驱动应用提供了简洁机制。

在一个图形用户接口环境中，用户通过点击按钮、选择菜单栏选项、在屏幕文本框中输入文字等方式与应用程序交互，这个过程就是事件驱动。用户移动鼠标，点击按钮或者键盘输入时会产生一个事件。例如，点击按钮会产生一个按钮事件，该事件将会被传送给用户编写的按钮点击处理程序，实现用户定义的功能。事件驱动是图形用户接口中最重要的概念。

本章除了介绍 GUI 开发外，还将围绕 turtle 库介绍图像编程，并利用 turtle 库函数开展艺术作品设计。图像编程不是 GUI 开发，因为所绘制的图形不响应外部事件。然而，绘图是表达思想的重要方式之一，通过编写程序表达独特设计思想是一件很酷的事儿。希望读者能够从本书实例中体会：

编程 = 有趣 + 好玩 + 酷炫 + 实用 + 价值 + 终身受用。

思考与练习：

[E9.1] GUI 编程与图像编程有何区别和联系？

[E9.2] 请解释什么是事件驱动？

9.2 模块 8：PyQt5 库的使用

要点：PyQt5 是 Qt5 应用框架的 Python 第三方库，它有超过 620 个类和近 6000 个函数和方法，它是最为成熟的商业级 GUI 第三方库。

9.2.1 PyQt5 库概述

PyQt5 是 Qt5 应用框架的 Python 第三方库，它有超过 620 个类和近 6000 个函数和方法。它是 Python 中最为成熟的商业级 GUI 第三方库。可以采用 pip 指令安装 PyQt5 函数库，方法如下。

```
:\>pip install PyQt5
```

使用 PyQt5 库创建一个图形界面的基本步骤如下：

步骤 1：导入 PyQt5 库。

步骤 2：创建一个图形用户接口程序的主窗体。

步骤 3：添加组件或更多的图形用户接口程序。

步骤 4：调用的 exec_方法退出。

PyQt5 库的导入方式如下，注意，请采用大小写混合方式。为了在命令行中方便的接收 sys.argv 参数，一般在引用 PyQt 库的时候也会调用 sys 库（非强制）。

```
>>>from PyQt5 import QtWidgets
>>>import sys
```

PyQt5 的类共划分为 11 类模块，如表 9.1 所示。

表 9.1: PyQt5 库的基础模块（11 类）

模块	描述
QtCore	包含核心非 GUI 功能，这个模块用于时间、文件和目录、变量数量类型、数据流、URLs、线程和进程

QtGui	包含窗口系统集成、事件处理、2D 绘画、基本成像、字体和文本
QtWidgets	包含了一系列 UI 元素,用于创建典型的桌面风格用户接口
QtMultimedia	包含处理多媒体内容和访问摄像机、无线电等功能的 APIs
QtBluetooth	包含设备扫描和设备连接与互动.
QtNetwork	包含网络编程方法，例如 TCP/IP 和 UDP 客户端、服务端等
Enginio	包含用于访问 Qt 云服务管理应用运行时客户端的库
QtWebKit	包含基于 WebKit2 库实现网络浏览器的类
QtSql	包含用于数据库操作的类
QtXml	包含用于 XML 文件操作的类，提供 SAX 和 DOM 接口实现
QtSvg	包含用于显示 SVG 文件内容的类

9.2.2 PyQt5 库解析：创建窗体

QtWidgets 是 PyQt5 界面设计中最常用的模块,它包含了一系列 UI 元素,用于创建典型的桌面风格用户接口。QtWidgets 的常用组件如表 9.2 所示。

表 9.2：QtWidgets 类的常用组件

组件	描述
QWidget	界面组件，所有界面对象类的基类
QDesktopWidget	提供了访问屏幕信息的功能
QToolBox	工具栏容器
QTabWidget	多标签容器
QPushButton	普通按钮
QToolButton	工具按钮：通常在工具栏使用
QRadioButton	单选框
QCheckBox	复选框
QLabel	标签
QTextBrowser	文本区域

QGraphicsView	图像显示
QLineEdit	单行文本框
QTextEdit	多行文本框（富文本）
QPlainTextEdit	多行文本框（纯文本）
QSpinBox	整数范围调节器
QTimeEdit	时间输入框
QDateEdit	日期输入框
QDateTimeEdit	时间日期输入框

表 9.2 中的组件提供了统一的操作方法，如表 9.3 所示。

表 9.3: QtWidgets 组件的通用操作方法

方法	描述
resize()	调整大小
move()	基于屏幕坐标移动或摆放，底边为 x 轴，左边为 y 轴，值域>=0
setWindowTitle()	设置窗口标题
setGeometry()	调整顶点坐标位置和窗口大小
setMaximumSize()	设置控件大小上限
setMinimumSize()	设置控件大小下限
setContentsMargins()	设置内容边界
setSpacing()	设置组件间距
show()	显示窗口

微实例 9.1：创建一个程序主窗体。

利用 PyQt5 库创建一个程序主窗体，代码如下，生成窗体效果如图 9.1 所示。

微实例 9.1　　m9.1MainWin.py

```
import sys
from PyQt5.QtWidgets import *
app = QApplication(sys.argv)
mywidget = QWidget()
mywidget.setGeometry(200, 200, 600, 300)
mywidget.setWindowTitle('Hello PyQt')
mywidget.show()
sys.exit(app.exec_())
```

[图 9.1: 微实例 9.1 的运行效果]

所有窗口程序都要先建立一个 QApplication 对象，它是图形用户接口程序的主窗体，可以把它想象成计算机交互编程的开始仪式，这里将其命名为 app。所有界面程序最后都要使用 exec_()方法退出主窗体。setGeometry(x, y, width, height)用来设定生成窗口的位置和大小，其中(x,y)表示窗体左顶点距离显示屏幕左侧和顶部的距离，width 和 height 是窗口的宽度和高度。setWindowTitle()方法设置窗口程序的标题内容。show()方法将小部件在屏幕上显示，在 PyQt 中，一个 widget 小部件首先被创建于内存中，然后通过 show 方法显示于屏幕。

当调用 exec_()时，应用进入了主循环。主循环会接受事件并把它们发送给相应

对象。sys.exit()方法确保了退出程序时对内存清理的可靠性，建议保留。

程序启动后，可以调整窗口位置。使用 QDesktopWidget()模块 screenGeometry()方法获取屏幕大小。屏幕的长/宽减去窗口的长/宽的一半，即为窗口位于中心时左上角坐标，尝试如下代码。

```
screen = QDesktopWidget().screenGeometry()
size = chatwidget.geometry()
chatwidget.move((screen.width() - size.width()) / 2, \
                (screen.height() - size.height()) / 2)
```

9.2.3 PyQt5 库解析：“信号-槽”机制

在图形用户接口环境中，用户通过点击按钮、选择菜单栏里的选项、在屏幕文本框中输入文字等与应用程序进行交互，这就是事件驱动。图形程序在屏幕上绘制了一系列界面元素，然后等待用户操作。

点击一下鼠标左键产生一个事件（event）。操作系统会将事件发送给相应应用程序处理，应用程序负责响应事件。事件驱动模型有三个参与者：

- 事件源（event source）
- 事件对象（event object）
- 事件目标（event target）

事件源是那些状态改变的对象,它产生事件;事件对象封装了事件源的状态改变;事件目标是需要被通知的目标程序。事件源把处理事件对象委托给了事件目标。

程序的主循环主要负责不断刷新程序界面以获取新的事件，当用户点击鼠标或敲击键盘后，主循环会接受这个操作产生的信号并将它们发送给指定对象。PyQt5 采用“信号-槽”（signal-slot）机制将事件和对应的处理程序进行绑定。

简单说，信号就是事件，槽是事件的处理程序。例如：鼠标点击操作是信号，处理点击鼠标的程序是槽。一个信号可以连接多个槽，也可以不连接槽。信号可以连接其他信号。PyQt 窗体有很多内置信号，也可以自定义信号。按钮最常见的内置

信号即为 cliked，即按下按钮。连接信号和槽的基本语句形式如下：

sender.singal.connect(receiver.slot)

sender 是发送信号的对象。receiver 是接受信号的对象,slot 是信号绑定的方法（函数）。

拓展：事件驱动

事件驱动是一种操作方式,即系统处于响应状态,对每个发生的事件进行处理。事件驱动广泛存在于各种 GUI 设计中，此外，它也是社会工作模式的一种。例如，银行柜员就处于事件驱动机制中，他们需要服务每一个到达客户的请求，如果没有客户到达，他们将处于空闲状态。自我驱动与事件驱动相反。例如，大学的科研人员一般处于自我驱动模式，他们会不断探索一个个感兴趣的研究问题，推动科技进步。

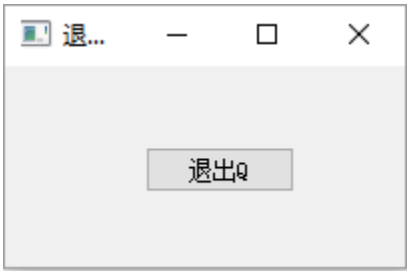
微实例 9.2：按钮的事件触发机制。

利用按钮建立一个简单的事件触发机制：点击按钮退出程序，代码如下，生成窗体效果如图 9.2 所示。

微实例 9.2　　m9.2ClickExit.py

```
import sys
from PyQt5.QtWidgets import QWidget,QPushButton,QApplication
from PyQt5.QtCore import QCoreApplication
app = QApplication(sys.argv)
exp = QWidget()
qtn = QPushButton('退出Q', exp)
qtn.resize(qtn.sizeHint())
qtn.clicked.connect(QCoreApplication.quit)
qtn.move(70,40)
exp.setGeometry(300,300, 200, 100)
```

```
11 exp.setWindowTitle('退出窗口')
12 exp.show()
13 sys.exit(app.exec_())
```



[图 9.2: 微实例 9.2 的运行效果]

微实例 9.2 导入 QtCore 和 QCoreApplication 模块,代码第 8 行通过 connect()函数将 QPushButton 的 clicked 信号与程序的退出连接起来，退出表示为 QCoreApplication.quit。代码第 7 行通过 resize()函数修改按钮大小，使其适应内部文字。特别介绍 resize()函数的 sizeHint()方法，它可以让 PyQt 根据文字来自适应按钮位置和大小。

9.2.4 PyQt5 库解析：窗体布局

PyQt5 有三种基本布局方式：QBoxLayout、QGridLayout 和 QFormLayout。

QBoxLayout 也称为框布局，又分为横向布局（QHBoxLayout）和纵向布局（QVBoxLayout)。框布局需要将控件排成一列或者一行，即 QBoxLayout 占用的空间分成一行或者一列框，然后把需要布局的控件依次填进去。对两个按钮进行框布局的代码如下。

```
1 mainLayout=QHBoxLayout()
2 mainLayout.addWidget(button1)
3 mainLayout.addWidget(button2)
4 self.setLayout(mainLayout)
```

Layout 对象类似一个封装器，Layout 里面还可以包含 Layout，都可以通过 addWidget 方法来确立封装关系。

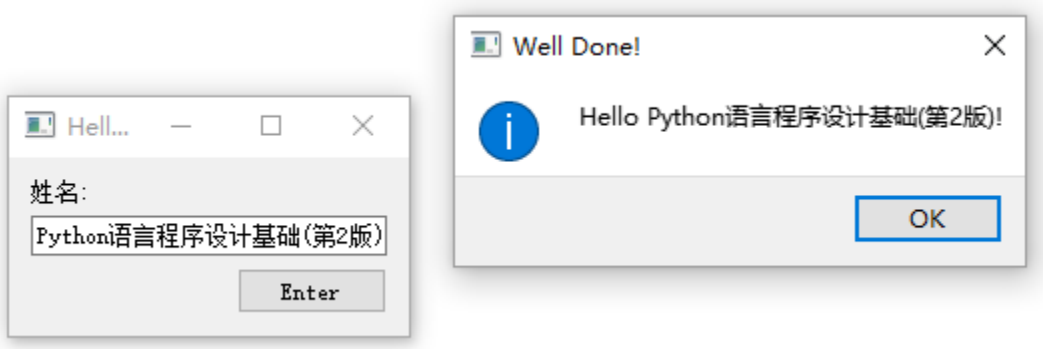
微实例 9.3：文本框回声程序。

向输入文本框输入字符串，点击按钮后在弹出的新窗口中显示键入的字符串。程序代码如下，生成窗体效果如图 9.3 所示。

微实例 9.3　　m9.3EchoName.py

```
import sys,os
from time import *
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
def show_name(name):
    name = nameLine.text()
    if name == "":
        QMessageBox.information(nameLine,"输入为空","请输入姓名")
        return
    else:
        QMessageBox.information(nameLine,"Well Done!", \
                                "Hello {}!".format(name))
app = QApplication(sys.argv)
Enter = QWidget()
nameLabel = QLabel("姓名:")
nameLine = QLineEdit(Enter)
EnterButton = QPushButton("Enter", Enter)
subLayout = QHBoxLayout()
subLayout.addStretch(1)
subLayout.addWidget(EnterButton)
bodyLayout = QVBoxLayout()
```

```
22  bodyLayout.addWidget(nameLabel)
23  bodyLayout.addWidget(nameLine)
24  bodyLayout.addLayout(subLayout)
25  EnterButton.clicked.connect(show_name)
26  Enter.setLayout(bodyLayout)
27  Enter.setWindowTitle("Hello Qt")
28  Enter.show()
29  sys.exit(app.exec_())
```



[图 9.3: 微实例 9.3 的运行效果]

微实例 9.3 首先定义主窗体窗口、单行文字编辑框 QLineEdit 和命名为‘Enter’的按钮，使用信号-槽机制将 EnterButton 按下的信号与 show_time()函数绑定。

show_time()函数定义如下：使用 text()函数取 LineEdit 的内容 name，若没有键入字符，则使用 QMessageBox.information 函数返回"请输入姓名"的弹窗，否则弹窗显示'Hello'加上键入的字符串。

利用 QHBoxLayout()定义一个简单的横向子布局 subLayout，将 Enter 按钮放在右下角。addStretch 函数是一个占位符，相当于一个可伸缩的 box，辅助布局。addWidget()是按顺序添加组建，先添加 addStretch()再添加 Button，此时 Button 被挤到横向布局的最右边。

用同样的方法进行纵向布局，按顺序加入标签、输入框，使用 addLayout 将包含其他控件的子布局 subLayout 也添加进去。如果一个布局不是最顶层布局，则一定需要将其放置到父窗口或者主布局里面。最后使用 setLayout()显示布局。

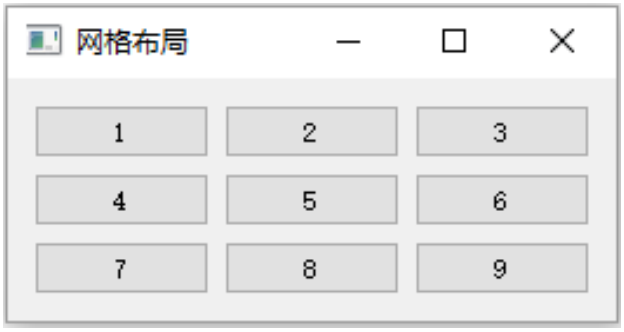
PyQt5 的另一个常用布局是网格布局 QGridLayout。网格布局的用法与 QBoxLayout 类似，addWidget()方法还可以接受两个额外参数表示行和列。

微实例 9.4：数字按钮的网格布局。

生成 1-9 共 9 个按钮，采用 3x3 方式布局。程序代码如下，生成窗体效果如图 9.4 所示。

微实例 9.4　　m9.4GridNumbers.py

```
from PyQt5.QtWidgets import QApplication, QWidget
from PyQt5.QtWidgets import QPushButton, QGridLayout
import sys
app = QApplication(sys.argv)
screen = QWidget()
bodyLayout = QGridLayout()
for i in range(1,10):
    button = QPushButton(str(i))
    bodyLayout.addWidget(button,(i-1)//3,(i-1)%3)
screen.setLayout(bodyLayout)
screen.setWindowTitle("网格布局")
screen.show()
sys.exit(app.exec_())
```



[图 9.4: 微实例 9.4 的运行效果]

微实例 9.4 演示了 QGridLayout 的使用。其中(i-1)//3 将数字变成了所在行数，(i-1)%3 将数字变成了所在列出，对应网格中行列的概念。

第三种布局是 QFormLayout，称为表单布局。这种方式常用于提交某个配置信息的表单。表单是提示用户进行交互的一种模式，其主要有两个列组成，第一个列用于显示信息，给予用户提示，称为 label 域；第二列需要用户选择输入，称为 field 域。表单布局完全可以使用表格布局实现，对应于多行两列的列表，但表单布局能提供一种相对完善的布局策略。感兴趣的读者请查阅 PyQt5 官方资料扩展学习。

思考与练习：

[E9.3] 解释 PyQt5 的信号槽机制的含义？

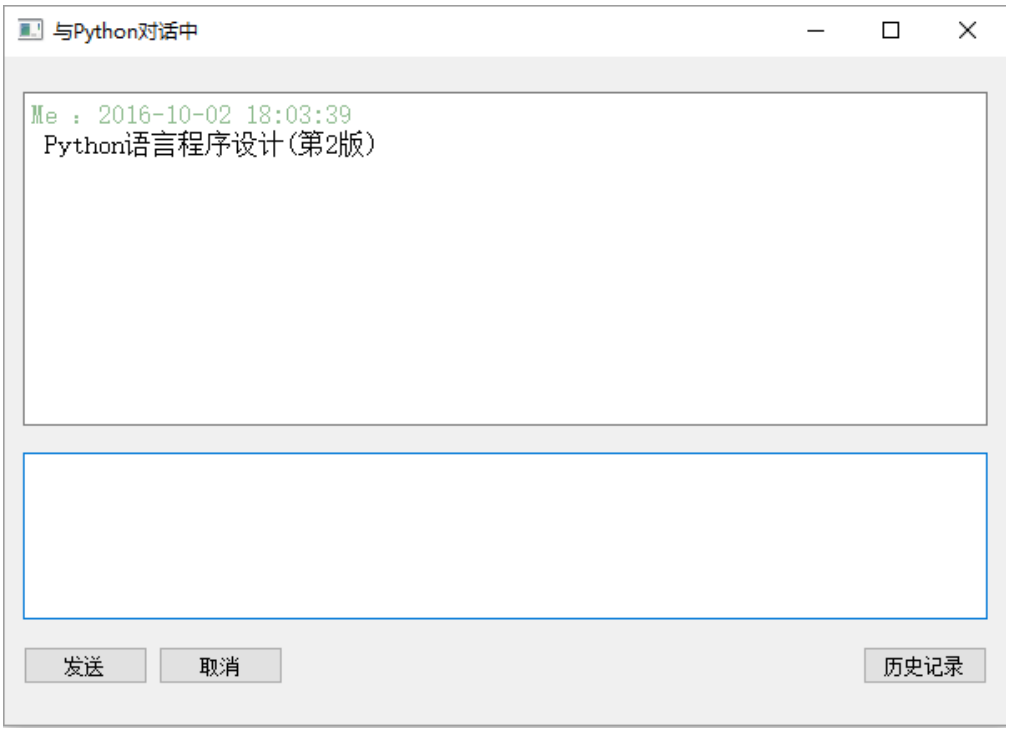
[E9.4] 微实例 9.3 第 25 行代码是什么含义？

[E9.5] 微实例 9.4 第 9 行代码是什么含义？

9.3 实例 17：聊天工具 GUI 开发

要点：这是一个采用 PyQt5 第三方库开发聊天工具 GUI 的实例。

聊天软件是日常网络交流必不可少的软件，如腾讯 QQ、淘宝旺旺等。聊天窗体一般分为上下两部分，上面为聊天信息窗体，展示双方的聊天记录；下面是当前输入的信息框和控制按钮。本节将实现一个聊天工具的 GUI 窗口，如图 9.5 所示。



[图 9.5：聊天工具 GUI 实例效果]

拓展：即时通信

即时通信指能够即时发送和接收互联网消息的应用。最早的即时通信软件是 ICQ，由三个以色列青年在 1996 年开发，ICQ 名字来源于英文“I Seek You”。ICQ 推出后获得了极大成功，最多时有 1 亿多用户，主要分布在美洲和欧洲。QQ 模仿了 ICQ 及同时期其他即时通信软件，在 1999 年 2 月第一次推出，直到今日，QQ 和微信几乎垄断了中国在线即时通讯类的软件市场。

简单将软件窗体的划分为三块区域：聊天记录区、输入区、按钮区，由一个横向布局和一个纵向布局合成。使用 PyQt 库中的 QBoxLayout()控件来实现区域布局。

上：聊天记录区：聊天历史信息显示

中：输入区：当前信息编辑区域

下：按钮区：放置发送、取消，历史记录等多个按钮

首先，导入相关库并定义主窗口和控件。

```
import sys,os
from time import *
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
chatapp = QApplication(sys.argv)
chatwidget = QWidget()
```

窗体中具体控件包括聊天消息显示控件、当前信息输入的 Text 控件、历史信息存储和读取的 Text 控件、消息发送按钮控件、消息取消按钮控件。

信息输入是典型的 Text 控件对象，这里选择多行文本框控件 QTextEdit，命名为 chatText。将信息显示区域创建为多行文本框的对象，命名为 outputarea。使用组件的通用方法 setMaximumSize 设定两个文本框的大小,在纵向上规定大致高度比为 2：1，PyQt 可以自适应产生等长不等高两个文本框。为了更加清晰地区分输入区和显示区文本，使用 setFont(QFont('SimSun', 12))方法将显示的所有字符字体设为 12 磅大小的宋体字，这部分代码如下。

```
chatText = QTextEdit(chatwidget)
chatText.setMaximumSize(QSize(600,100))
outputarea = QTextEdit(chatwidget)
outputarea.setReadOnly(True)
outputarea.setFont(QFont('SimSun', 12))
outputarea.setMaximumSize(QSize(600, 200))
```

创建 3 个按钮 btnSend、btnCancel 和 btnHistory，它们都以点击为信号，分别连接到不同的处理函数，代码如下。

```
btnSend = QPushButton('发送', chatwidget)
btnCancel = QPushButton('取消', chatwidget)
btnHistory = QPushButton('历史记录', chatwidget)
btnSend.clicked.connect(showDialog)
```

```
btnCancel.clicked.connect(cancelMsg)
btnHistory.clicked.connect(getMsg)
```

其中，btnCancel 按钮与 cancelMsg()函数绑定，用于清除当前输入框的文字，函数定义如下。

```
def cancelMsg():
    chatText.clear()
```

btnSend 按钮与 showDialog()绑定，将输入框文字显示到 outputarea，并为每次显示内容添加名字“Me”和时间日期结果。label 包含提示字符和从 time 库获取的时间日期，这里采用 CSS 的网页显示格式将其显示为绿色。利用 toPlainText()从输入框提取字符信息，和 label 一起添加到显示区域，为了不抹消之前传入的消息，使用 append 而不是 setText 来传入字符。在每次传入消息后，使用 clear()函数清空当前输入框，产生刷新的视觉效果，并将光标重新移动到 chatText 区域准备好接受下一次输入，相关代码如下。

```
def showDialog():
    label = "<span style=' color:#8FBC8F;'>Me : {} </span>".\
          format(strftime("%Y-%m-%d %H:%M:%S", localtime()))
    message =  chatText.toPlainText()
    outputarea.append(label)
    outputarea.append(' '+message)
    chatText.clear()
    chatText.setFocus()
    saveMsg(message)
```

上段程序最后的 saveMsg(message)函数将每次输入的信息按顺序存入程序同目录的文本文件中，通过点击“历史消息”按钮方便读入所有历史文本。历史消息的相关操作包含了文本的存取函数，saveMsg()和 getMsg()。saveMsg()用于打开

'save.txt'并写入最新的文本记录。调用 getMsg()函数时判断当前消息文本是否存在，如果不存在历史消息文本，则在输出区域显示'No Record'；反之则读取并按行写入 outputarea。相关代码如下。

```
def saveMsg(txt):
    file = open('save.txt', 'a')
    file.write(txt + '\n')
    file.close()
def getMsg():
    if os.path.exists('save.txt'):
        message = '\nGet The Message:'
        outputarea.append(message)
        file = open('save.txt', 'r')
        txt = file.read() + '\n'
        outputarea.append(txt)
        file.close()
    else:
        outputarea.append('No Record\n')
```

设置完所有控件后就开始界面布局，分别创建一个横向一个纵向两个方向的 Box，分别命名为 hbox 和 vbox，将 hbox 作为 vbox 的子布局传入显示。hbox 主要负责三个按钮的布局,将‘发送’和‘取消’按钮排在左边,再加入一个弹簧分隔符，将‘历史消息’按钮排在最右边，即可模拟一个熟悉的对话框风格。vbox 负责纵向三大主区域的布局，依次添加显示区、输入区和按钮子布局即可。相关代码如下。

```
hbox = QHBoxLayout()
hbox.addWidget(btnSend)
hbox.addWidget(btnCancel)
hbox.addStretch(1)
hbox.addWidget(btnHistory)
vbox = QVBoxLayout()
vbox.addWidget(outputarea)
```

```
vbox.addWidget(chatText)
vbox.addLayout(hbox)
chatwidget.setLayout(vbox)
```

最后，按照微实例 17.1 的窗口显示方法以及补充介绍的窗口居中方法操作，完成整个聊天界面的设计。实例代码 17.1 如下。

实例代码 17.1　　e17.1TxtMsgGUI.py

```
import sys,os
from time import *
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
def showDialog():
    label = "<span style=' color:#8FBC8F;'>Me : {} </span>"\
         .format(strftime("%Y-%m-%d %H:%M:%S", localtime()))
    message =  chatText.toPlainText()
    outputarea.append(label)
    outputarea.append(' '+message)
    chatText.clear()
    chatText.setFocus()
    saveMsg(message)
def cancelMsg():
    chatText.clear()
def saveMsg(txt):
    file = open('save.txt', 'a')
    file.write(txt + '\n')
    file.close()
def getMsg():
    if os.path.exists('save.txt'):
        message = '\nGet The Message:'
```

```
        outputarea.append(message)
        file = open('save.txt', 'r')
        txt = file.read() + '\n'
        outputarea.append(txt)
        file.close()
    else:
        outputarea.append('No Record\n')
chatapp = QApplication(sys.argv)
chatwidget = QWidget()
chatText = QTextEdit(chatwidget)
chatText.setMaximumSize(QSize(600,100))
outputarea = QTextEdit(chatwidget)
outputarea.setReadOnly(True)
outputarea.setFont(QFont('SimSun', 12))
outputarea.setMaximumSize(QSize(600, 200))
btnSend = QPushButton('发送', chatwidget)
btnCancel = QPushButton('取消', chatwidget)
btnHistory = QPushButton('历史记录', chatwidget)
btnSend.clicked.connect(showDialog)
btnCancel.clicked.connect(cancelMsg)
btnHistory.clicked.connect(getMsg)
hbox = QHBoxLayout()
hbox.addWidget(btnSend)
hbox.addWidget(btnCancel)
hbox.addStretch(1)
hbox.addWidget(btnHistory)
vbox = QVBoxLayout()
vbox.addWidget(outputarea)
vbox.addWidget(chatText)
vbox.addLayout(hbox)
chatwidget.setLayout(vbox)
chatwidget.setGeometry(0, 0, 600, 400)
```

```
55  chatwidget.setWindowTitle('与 Python 对话中')
56  screen = QDesktopWidget().screenGeometry()
57  size = chatwidget.geometry()
58  chatwidget.move((screen.width() - size.width()) / 2, \
                    (screen.height() - size.height()) / 2)
59  chatwidget.show()
60  sys.exit(chatapp.exec_())
```

思考与练习：

[E9.6] 实例代码 17.1 如何布局三个按钮控件？

[E9.7] 实例代码 17.1 如何处理发送按钮点击事件？

[E9.8] 实例代码 17.1 如何提取历史记录？

9.4 模块 9：turtle 库的使用

要点：turtle 是一个简单的图形绘制标准库。

9.4.1 turtle 库概述

turtle 是 Python 的内置图形化模块。Python 3 系列版本安装目录的 Lib 文件夹下可以找到 turtle.py 文件。因为 turtle 绘制图形的命令简单且直观，从本书第 2 章开始已经逐步介绍了 turtle 库的常用函数。本节将更系统的介绍 turtle 库使用。更多信息请查阅 turtle 库官方文档。

https://docs.python.org/3/library/turtle.html

引入 turtle 库可以采用如下两种方式。

```
>>>import turtle  #或者
>>>from turtle import *
```

对于 turtle 库的初步介绍请参考第 2.4 节，与该库相关的基本概念不再赘述。

拓展：用户体验

用户体验是用户使用产品过程中建立的一种纯主观感受。ISO9241-210 标准将用户体验定义为“人们对于针对使用或期望使用的产品、系统或者服务的认知印象和回应”。简单说，用户体验指好不好用、方不方便。用户体验的共性能够经由良好设计实验来认识。软件开发过程中，要考虑使用者的用户体验，因为用户而不是技术决定产品的价值。

9.4.2 turtle 库解析

turtle 库使用画笔 pen 绘制图形，表 9.4 给出了控制画笔绘制状态的方法。

表 9.4：turtle 库控制画笔绘制状态的方法

方法	描述
pendown()	放下画笔
penup()	提起画笔，与 pendown()配对使用
pensize(width)	设置画笔线条的粗细为指定大小

turtle 以小海龟爬行角度来绘制曲线，小海龟即画笔，表 9.5 给出了控制 turtle 运动的方法。

表 9.5：turtle 库控制画笔运动的方法

方法	描述
forward()	沿着当前方向前进指定距离
backward()	沿着当前相反方向后退指定距离
right(angle)	向右旋转 angle 角度
left(angle)	向左旋转 angle 角度
goto(x,y)	移动到绝对坐标（x,y）处

setx( )	将当前 x 轴移动到指定位置
sety( )	将当前 y 轴移动到指定位置
setheading(angle)	设置当前朝向为 angle 角度
home()	设置当前画笔位置为原点，朝向东。
circle(step)	绘制一个指定半径，角度、以及绘制步骤 step 的圆
dot(r,color)	绘制一个指定半径 r 和颜色 color 的圆点
undo()	撤销画笔最后一步动作
speed()	设置画笔的绘制速度，参数为 0-10 之间

微实例 9.5：三边形到多边形绘制。

编写一个程序，绘制三角形、四边形、五边形、六边形和圆形。程序代码如下，运行效果如图 9.6 所示。

微实例 9.5　　m9.5DrawPolygon.py

```
import turtle
turtle.pensize(3) # 三角形
turtle.penup()
turtle.goto(-200,-50)
turtle.pendown()
turtle.circle(40,steps=3)
turtle.penup()# 四边形
turtle.goto(-100,-50)
turtle.pendown()
turtle.circle(40,steps=4)
turtle.penup()# 五边形
turtle.goto(0,-50)
turtle.pendown()
turtle.circle(40,steps=5)
turtle.penup()  # 六边形
```

```
turtle.goto(100,-50)
turtle.pendown()
turtle.circle(40,steps=6)
turtle.penup()  # 圆形
turtle.goto(200,-50)
turtle.pendown()
turtle.circle(40)
turtle.down()
```

[图 9.6: 三边形到多边形的绘制效果]

表 9.6 列出了与画笔颜色和字体相关的方法。

表 9.6: turtle 库控制画笔颜色和字体的方法

方法	描述
color()	设置画笔的颜色
begin_fill()	填充图形前，调用该方法
end_fill()	填充图形结束
filling()	返回填充的状态，True 为填充，False 为未填充
clear()	清空当前窗口，但不改变当前画笔的位置
reset()	清空当前窗口，并重置位置等状态为默认值
screensize()	设置画布的长和宽
hideturtle()	隐藏画笔的 turtle 形状
showturtle()	显示画笔的 turtle 形状
isvisible()	如果 turtle 可见，则返回 True
write()	输出 font 字体的字符串

微实例 9.6：三边形到多边形的彩色绘制。

编写一个程序，绘制三角形、四边形、五边形、六边形和圆形。通过调用 turtle 的 begin_fill()和 end_fill()方法将绘制的图形填充上颜色。通过调用 write()函数输出一行文字，字体为 Times，字号为 18，粗体。程序代码如下，运行效果如图 9.7 所示。

微实例 9.6　　m9.6DrawFillPolygon.py

```
import turtle
turtle.pensize(3)  #三角形
turtle.penup()
turtle.goto(-200,-50)
turtle.pendown()
turtle.begin_fill()
turtle.color('red')
turtle.circle(40,steps=3)
turtle.end_fill()
turtle.penup()    #四边形
turtle.goto(-100,-50)
turtle.pendown()
turtle.begin_fill()
turtle.color('blue')
turtle.circle(40,steps=4)
turtle.end_fill()
turtle.penup()   #五边形
turtle.goto(0,-50)
turtle.pendown()
turtle.begin_fill()
turtle.color('green')
turtle.circle(40,steps=5)
```

```
turtle.end_fill()
turtle.penup()  #六边形
turtle.goto(100,-50)
turtle.pendown()
turtle.begin_fill()
turtle.color('yellow')
turtle.circle(40,steps=6)
turtle.end_fill()
turtle.penup()  #圆形
turtle.goto(200,-50)
turtle.pendown()
turtle.begin_fill()
turtle.color('purple')
turtle.circle(40)
turtle.down()
turtle.end_fill()
turtle.color('green') #文字
turtle.penup()
turtle.goto(-100,50)
turtle.pendown()
turtle.write(("Cool Colorful Shapes"), font=("Times",18,"bold"))
turtle.hideturtle()
turtle.done()
```

[图 9.7: 三边形到多边形的彩色绘制效果]

思考与练习：

[E9.9] 如何获得 turtle 当前画笔的坐标？

[E9.10] 如何将画笔移动到某个位置且不在绘图区域里留下痕迹？

[E9.11] 如何在画布的特定位置上输出文字？

[E9.12] 微实例 9.6 第 44 行代码是什么功能？

9.5 实例 18：“雪景”图形艺术

要点：这是一个利用 turtle 库绘制“雪景”图形艺术的实例。

turtle 图形艺术，又称 turtle 艺术（turtle Art），指利用 turtle 库画笔创造性绘制的绚丽多彩的艺术图形。图 9.8 给出一些 turtle 图形艺术编程实例。

[图 9.8：turtle 图形艺术编程实例]

上述 turtle 图形艺术效果中隐含着很多随机元素，如随机颜色、尺寸、位置和数量等。因此，在图形艺术绘制中需要引入随机函数库 random。常用 randint()函数，生成指定范围内的随机数，例如 randint(-350,350)生成-350 到 350 的随机整数。

拓展：自动生成艺术

自动生成艺术是艺术家应用计算机程序，或一系列自然语言规则，或一个机器，或其它发明物，产生出的具有一定自控性的过程，该过程的直接或间接结果是一个完整的艺术品。自动生成艺术诞生于上世纪 60 年代，至今仍然是人们感兴趣的话题。

本实例选择图 9.8 中“雪景”作为编程目标。

“雪景图形艺术背景为黑色，分为上下两个区域，上方是漫天彩色雪花，下方是由远及近的灰色横线渐变。该图运用了随机元素，如雪花位置、颜色、大小、花瓣数目、地面灰色线条长度、线条位置等，需要使用 turtle 库和 random 库。

首先构建图的背景，设定窗体大小为 800x600 像素，窗体颜色为 black。然后，定义上方雪花绘制函数 drawSnow()和下方雪地绘制函数 drawGround()。

为保证艺术效果，drawSnow()函数首先隐藏 turtle 画笔、设置画笔大小、绘制速度，然后使用 for 循环绘制 100 朵雪花。为了使雪花颜色随机，使用 random()函数生成画笔颜色 rgb 值。turtle 库可以接受两种 RGB 取值范围，一种是 0-1 浮点数，一种是 0-255 整数，这里取值为 0-1 之间的浮点数。在指定区域随机选择坐标绘制雪花，利用 randint(-350,350)生成-350 到 350 范围随机数。上方区域定义为 x 坐标范围（-350,350）、y 坐标范围（1,270）内的区域。雪花大小 snowsize、雪花花瓣数 dens 都分别设定为一定数值范围随机数。最后通过 for 循环绘制出多彩雪花。

drawGround()函数使用 for 循环绘制地面 400 个小横线，画笔大小 pensize、位置坐标 x、y、线段长度均通过 randint()函数作为随机数产生。由于小横线的灰度呈现由远及近的效果，因此 r、g、b 的值应根据 y 坐标进行变化；灰色的 r、g、b 值颜色相等，为了控制在 0-1 间，它们的计算方式均为-y/280。

“雪景”绘制在尺寸为 800*600 的 Python 窗体中，背景颜色设为黑色。图形艺术程序往往动画时间较长，可以在主程序调用 Tracer(False)函数关闭动画效果，从而一次性绘制所有效果。

“雪景”图形艺术全部代码为实例代码 18.1。运行程序，即可以绘制出美丽的雪景图，如图 9.9 所示。

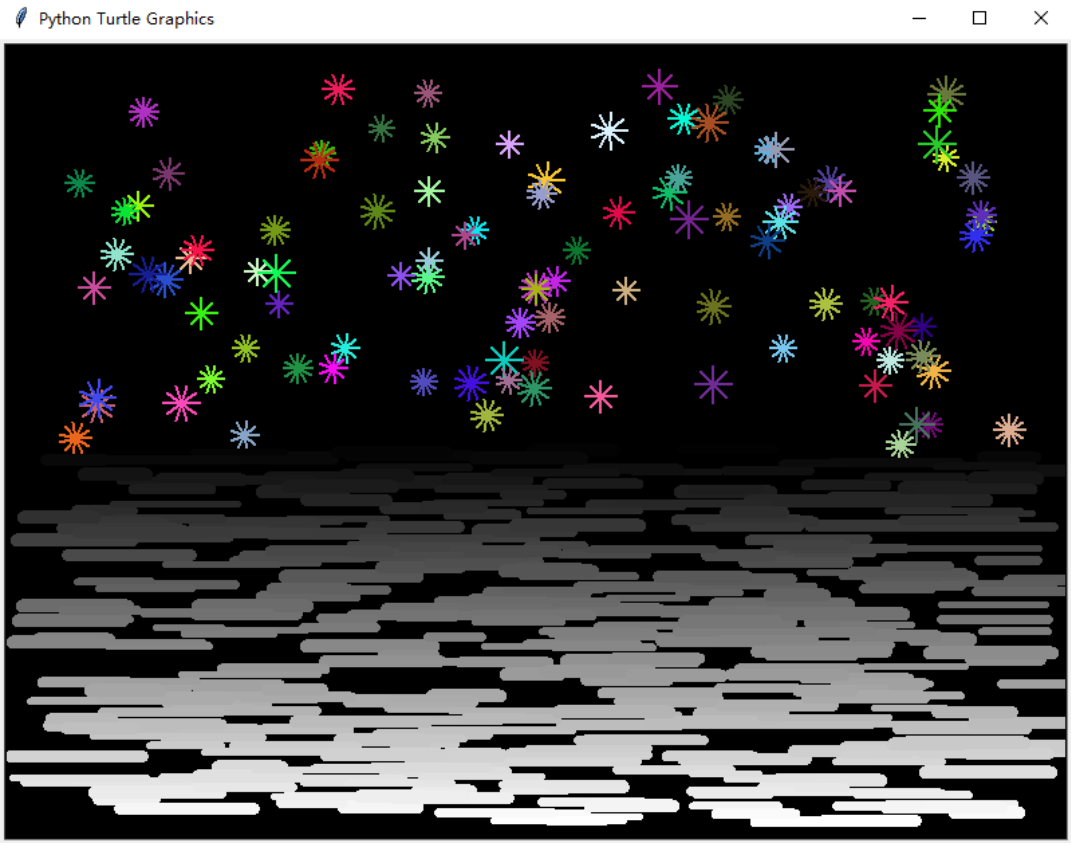
——图形还不够美？

——期待您的艺术作品！

实例代码 18.1　　e18.1SnowView.py

```
#e18.1SnowView.py
from turtle import *
from random import *
import math
def drawSnow():
    hideturtle()
    pensize(2)
    for i in range(100):
        r, g, b = random(), random(), random()
        pencolor(r,g,b)
        penup()
        setx(randint(-350,350))
        sety(randint(1,270))
        pendown()
        dens = randint(8,12)
        snowsize = randint(10,14)
        for j in range(dens):
            forward(snowsize)
            backward(snowsize)
            right(360/dens)
def drawGround():
    hideturtle()
    for i in range(400):
        pensize(randint(5,10))
        x = randint(-400,350)
        y = randint(-280,-1)
        r, g, b = -y/280, -y/280, -y/280
```

```
28          pencolor((r,g,b))
29          penup()
30          goto(x,y)
31          pendown()
32          forward(randint(40,100))
33  setup(800,600,200,200)
34  tracer(False)
35  bgcolor("black")
36  drawSnow()
37  drawGround()
38  done()
```



[图 9.9: “雪景”图形艺术运行效果]

思考与练习：

[E9.13] 实例代码 18.1 第 10 行接收的 RGB 值分别在什么范围？

[E9.14] 实例代码 18.1 第 34 行代码的作用是什么？

[E9.15] 实例代码 18.1 第 38 行代码的作用是什么？

本章小结

本章介绍了简单的图形编程方法，重点介绍了人机交互界面编程思想，及使用 PyQt4 库实现图形用户接口的方法。此外，本章详细介绍了 turtle 库及绘制基本图形对象的方法。本章通过聊天工具 GUI 绘制和图形艺术实例展示了人机接口和图形编程的价值。

程序练习题

[P9.1] 参考实例 17，实现一个自动问答程序的 GUI，对用户输入的每个问题机器都有回应。

[P9.2] 采用 PyQt5 实现一个简单的计算器程序。

[P9.3] 采用 PyQt5 实现一个简单的记事本。

[P9.4] 使用 turtle 库绘制一朵玫瑰花。

[P9.5] 使用 turtle 库绘制一个颜色图谱。