项目 1　学生成绩管理系统

在老师、家长和学生的交谈中，成绩是个永恒的话题，了解孩子在校的成绩是家长最直观指导孩子学习状态的一种方式。同时，对于老师和教务工作者来说，对学生成绩进行数据分析，是掌握教学情况最科学的手段。在大数据时代，手工记录、分析学生成绩的方式早已不适用，需要更加方便快捷的方式统计、分析学生的学习成绩数据，以便进一步促进教学改革。

在项目 1 中使用 Python 语言开发一个学生成绩管理系统，该系统能帮助教师快速录入、查找、修改学生的成绩；除此之外，通过排序、按分数段统计人数功能让教师对课程的考核结果进行宏观评价，促进教学计划调整，提升教学质量。

通过该项目，可以掌握的知识有 Python 运行环境的搭建、Python 语法基础、if 和 for 流程控制语句的使用，无参函数的用法，字典、列表的应用。通过本项目，学生可体验项目开发的基本流程，项目开发的一般规则，对系统基本功能增、删、改、查的实现形成解决思路。

1.1　典型工作环节 1：需求分析

为满足教师对数据录入、分析的需求，学生成绩管理系统应具有以下功能：

1. 录入学生成绩。
2. 修改和删除学生成绩。
3. 查询学生成绩。
4. 对学生成绩进行排序。
5. 统计各分段成绩人数。

1.2　典型工作环节 2：系统设计

1.2.1　系统功能设计

学生成绩管理系统分为 4 大功能模块：信息维护、查询、排序、统计，如图 1.1 所示。

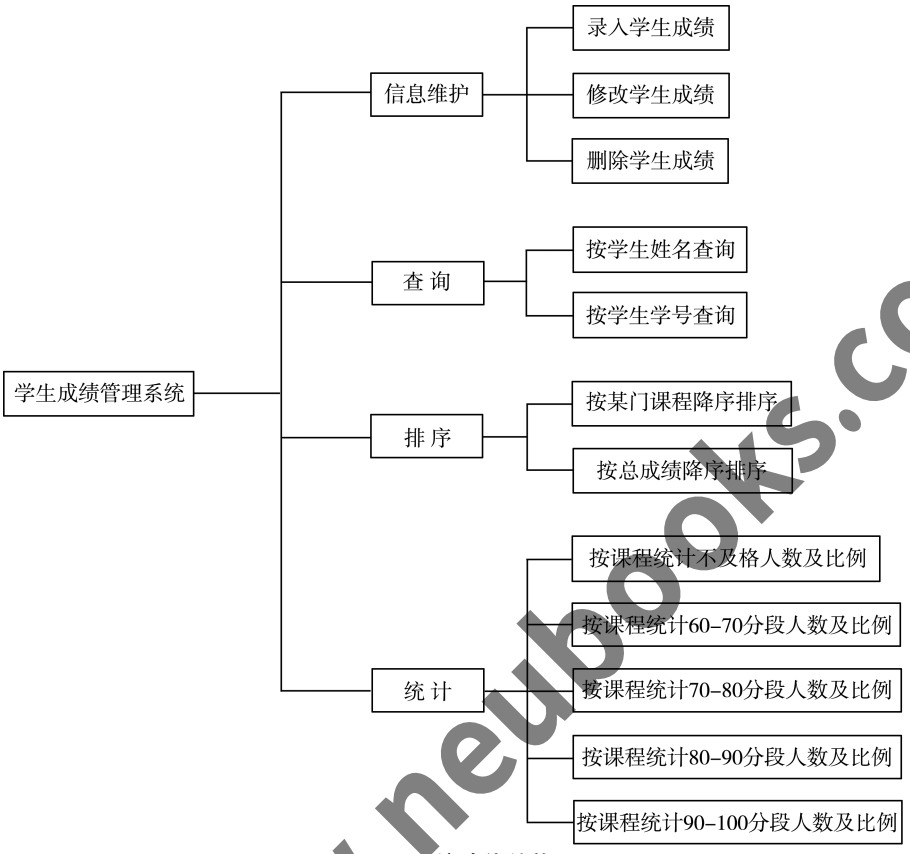

图 1.1 系统功能结构图

1.2.2 系统业务流程

在开发系统之前，应该先了解系统的业务流程，以方便定制开发计划。根据对学生成绩管理系统的需求分析及功能分析，设计如图 1.2 所示的系统业务流程。

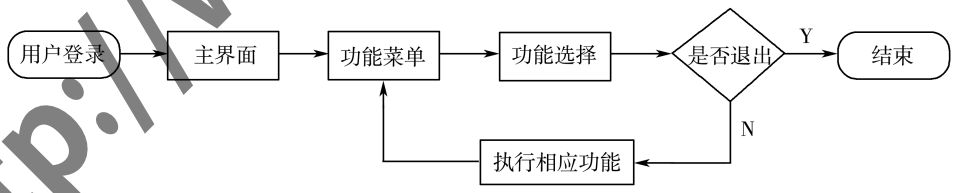

图 1.2 系统业务流程

1.2.3 系统开发环境

本系统开发及运行环境如下：

1. 操作系统：Windows7 或以上。
2. Python 版本：Python3.7。
3. 开发工具：PyCharm。

1.3　典型工作环节 3：软件编码

1.3.1　搭建 Python 开发环境

Python 是一种清晰而强大的面向对象编程语言，可与 Perl，Ruby，Schem 或 Java 相媲美。Python 是一门解释性语言，在运行时需要依赖 Python 解释器。在 Python 官网下载 Python 解释进行安装。Python 支持跨平台，因此在 Windows、Linux 上均可以运行。同时 Python 也有多个发行版，比如 Anconda。支持 Python 开发的工具也有很多，比如 Sublime、Notepad＋＋、Pycharm 等。

Pycharm 是 JetBrains 公司开发的 Python 集成开发环境，Pycharm 不仅包含一般 IDE 的功能，比如调试、单元调试、查看堆栈、即时计算、语法高亮、项目管理、代码格式化、智能提示、版本管理等，还提供了支持友好的 Django、Flask、Google App Engine、Angular、React、Docker 等。为了“平台”开发工作的顺利开展，因此本节主要工作是完成 Python 运行环境的搭建：基于 Windows 安装 Python 解释器，同时安装开发工具：Pycharm。

1.3.1.1　在 Windows 平台上的安装与配置

从官网下载 Python 最新版本如下，3.7 是主要版本号，.3 是分支版本号，amd64.exe 表示 Windows64 位系统。

```
python-3.7.3-amd64.exe
```

步骤 1：双击软件名，打开安装界面，如图 1.3 所示。

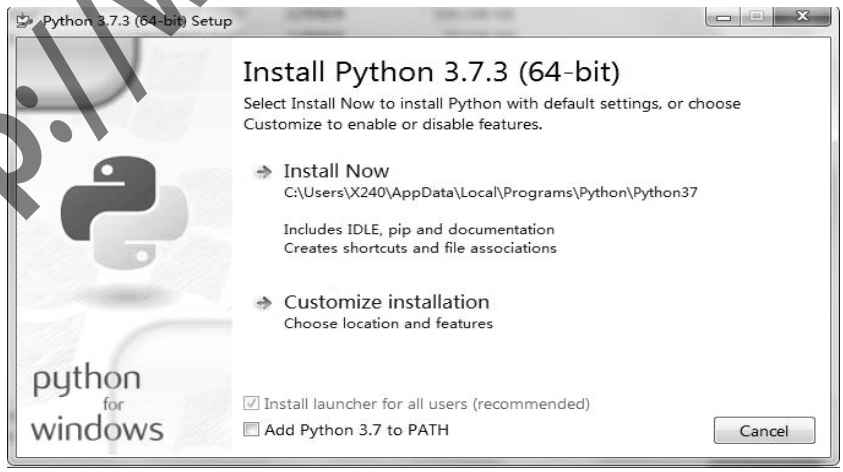

图 1.3　安装 python

步骤 2：选择自定义，保持默认安装，如图 1.4 所示。

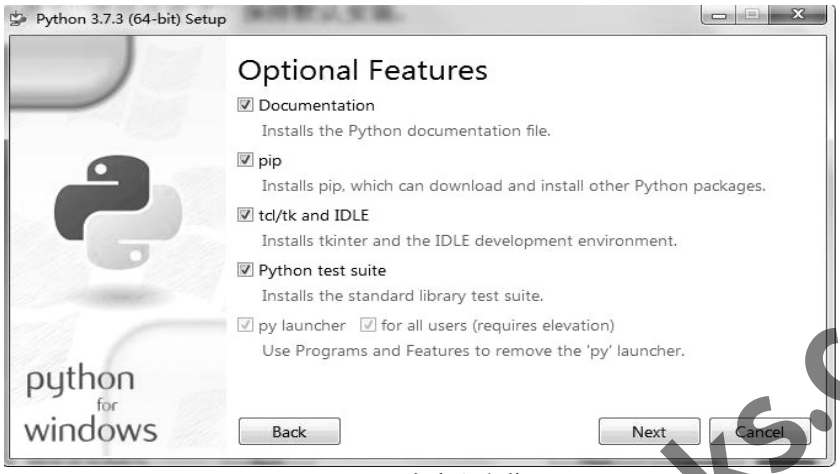

图 1.4　自定义安装

步骤 3：指定 Python 安装位置，如图 1.5 所示。

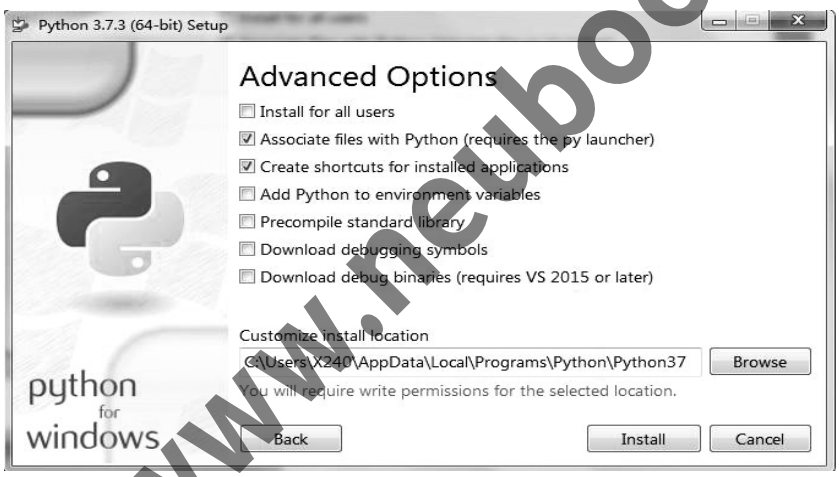

图 1.5　设置安装路径

步骤 4：等待安装完毕，如图 1.6 所示。

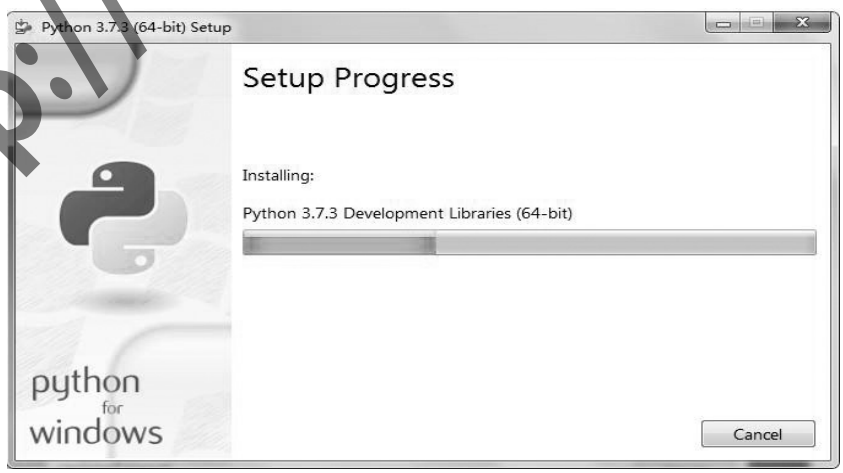

图 1.6　等待安装

步骤 5：配置环境变量，右击我的电脑，选择高级系统设置，如图 1.7 所示。

图 1.7　打开环境变量

步骤 6：选择环境变量，如图 1.8 所示。

步骤 7：在变量值路径中添加 Python 安装路径，如图 1.9 所示。

图 1.8　环境变量

图 1.9　选择 Path

步骤 8：在开始菜单打开 Python3.7Shell 程序，如图 1.10 所示。

步骤 9：在 Shell 窗口后输入内容。

```
print("hello world")
```

图 1.10　打开 Shell

得到如下结果，表示环境正常，如图 1.11 所示。

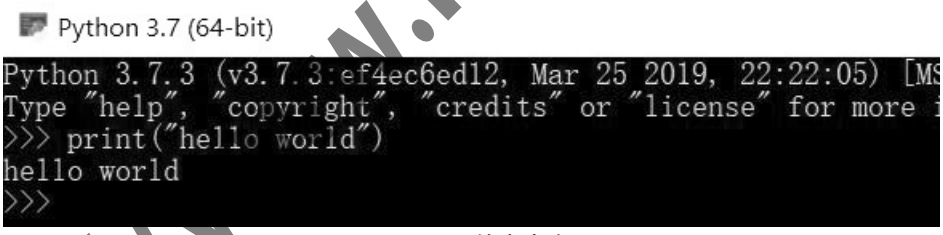

图 1.11　输出内容

1.3.1.2　安装 PyCharm

集成开发环境具备调试、语法高亮、项目管理、智能提示等便于开发和写代码的功能，PyCharm 是 JetBrains 公司开发的 Python 集成环境，功能强大，适合初学者和较大型项目的开发。安装步骤如下：

步骤 1：访问 PyCharm 官网下载，根据系统版本和平台下载合适的安装包，下载完成后双击软件名，打开安装界面，如图 1.12 所示。

步骤 2：选择安装路径，如图 1.13 所示。

步骤 3：选择安装选项，保持默认即可，如图 1.14 所示。

步骤 4：设置开始菜单目录名称，如图 1.15 所示。

步骤 5：等待安装结束，如图 1.16 所示。

步骤 6：安装完毕，如图 1.17 所示。

图 1.12　安装 Pycharm

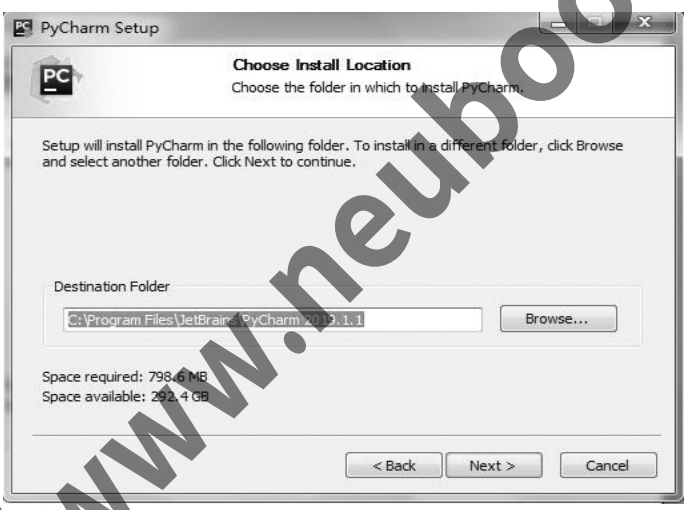

图 1.13　选择路径

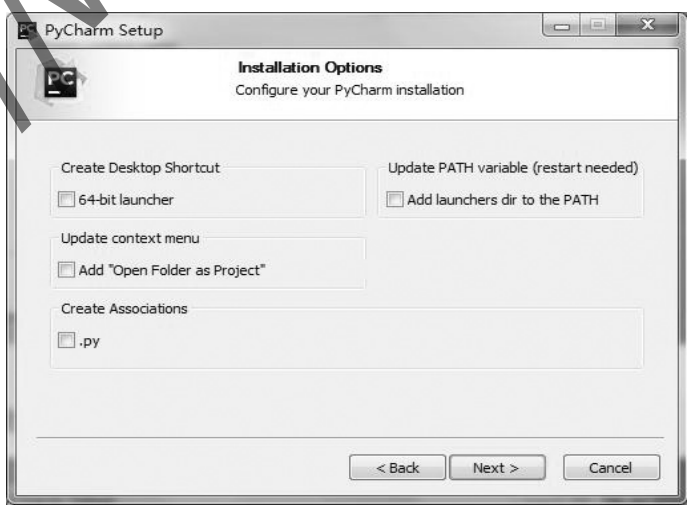

图 1.14　安装选项

图 1.15　快捷键目录

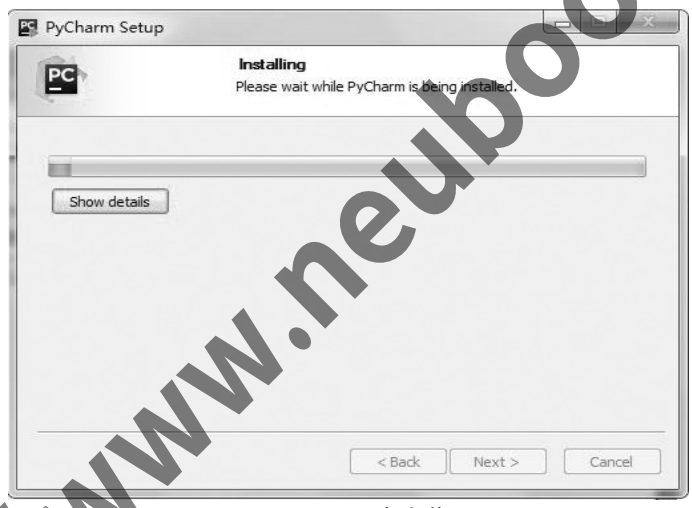

图 1.16　正在安装

图 1.17　安装完毕

1.3.1.3　创建一个 Python 项目

安装完毕后，双击 Pycharm 图标，创建 Python 项目。

步骤 1：首次启动 Pycharm，需要设置主题。保持默认点击右下角的 Next 按钮，如图 1.18所示。

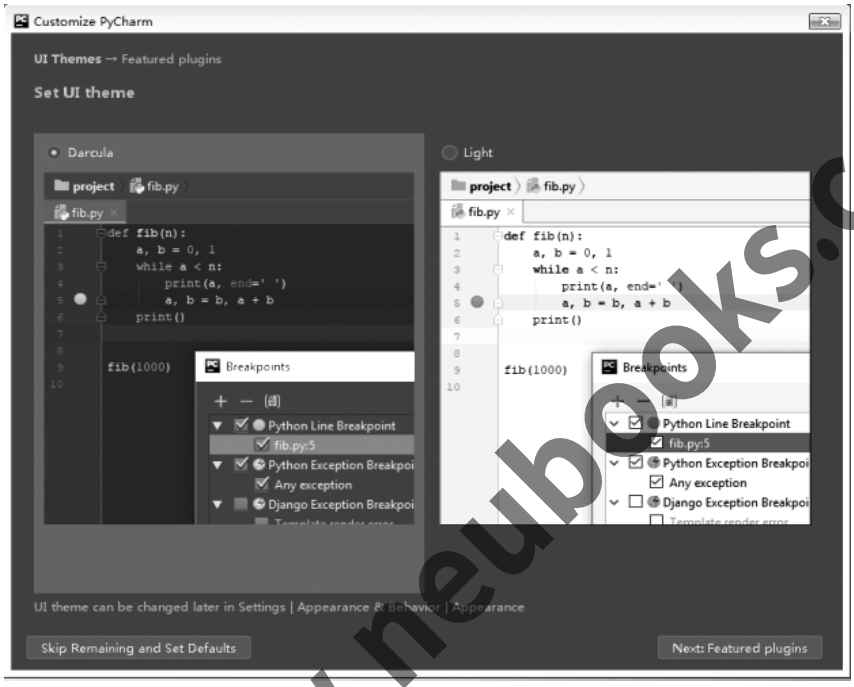

图 1.18　设置主题

步骤 2：选择要安装的其他组件，在左下角选择跳过，如图 1.19 所示。

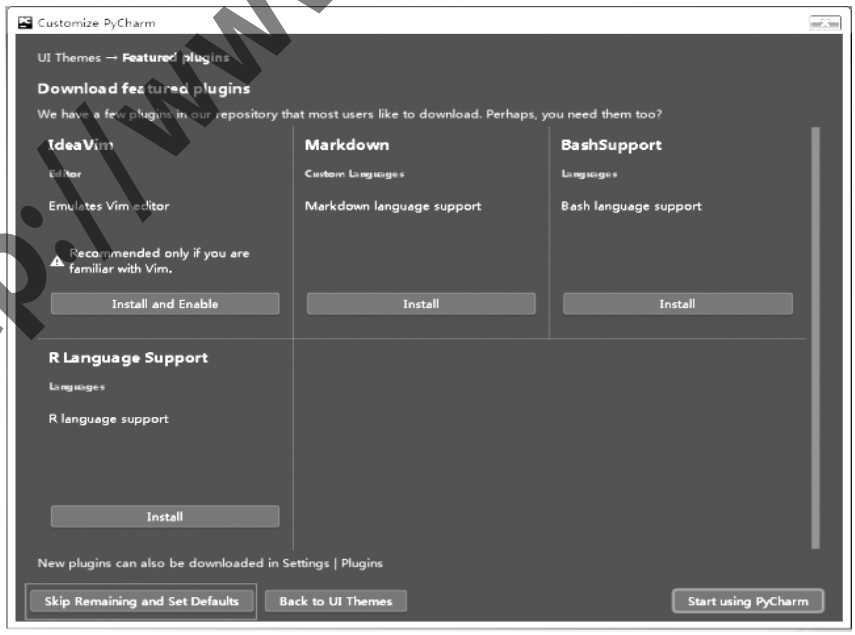

图 1.19　配置安装其他组件

步骤 3：选择评估版本，如图 1.20 所示。

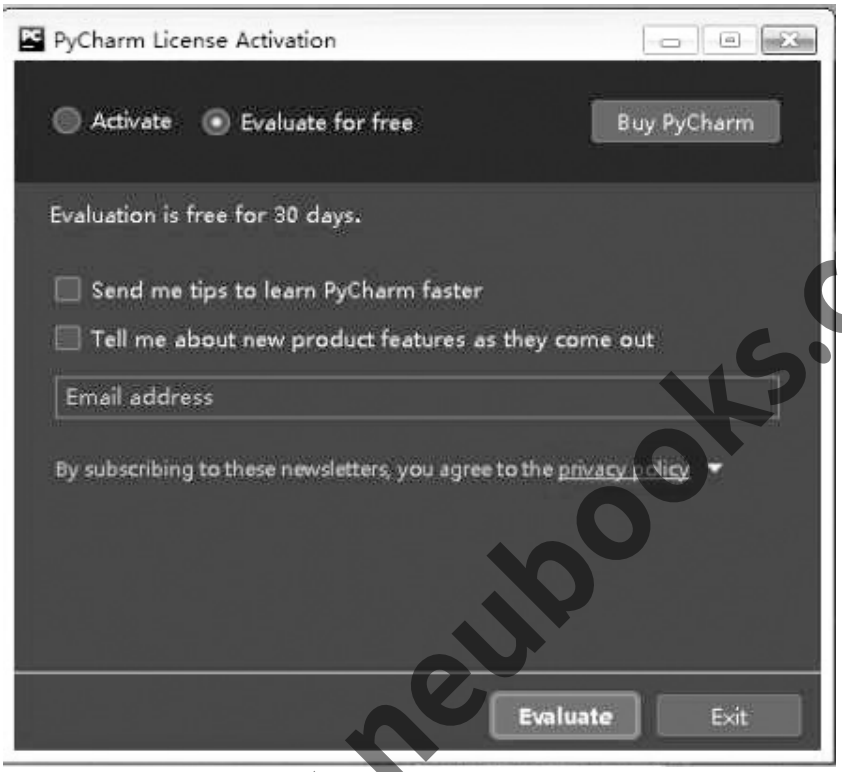

图 1.20　输入账号

步骤 4：正在打开开发环境，如图 1.21 所示。

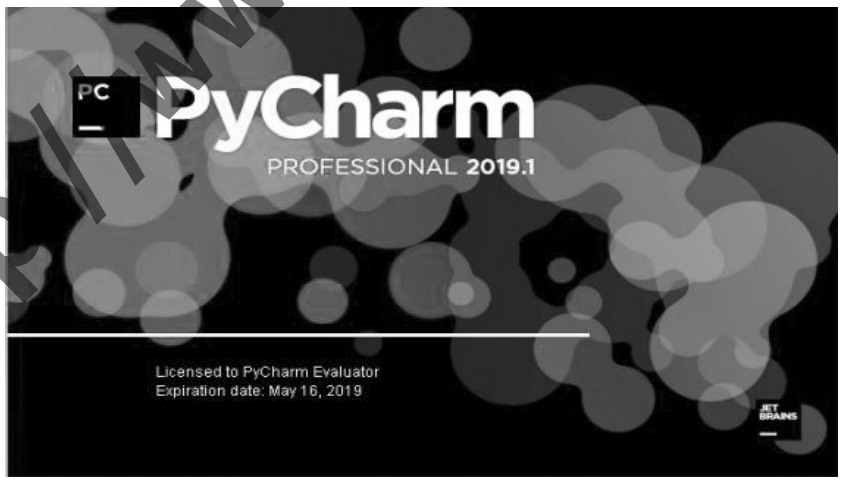

图 1.21　启动画面

步骤 5：左边列表选择 Pure Python，在 Location 后面输入项目保存的路径。在 Existing interpreter 选择 Python 安装路径下的 python.exe 文件，如图 1.22 所示。

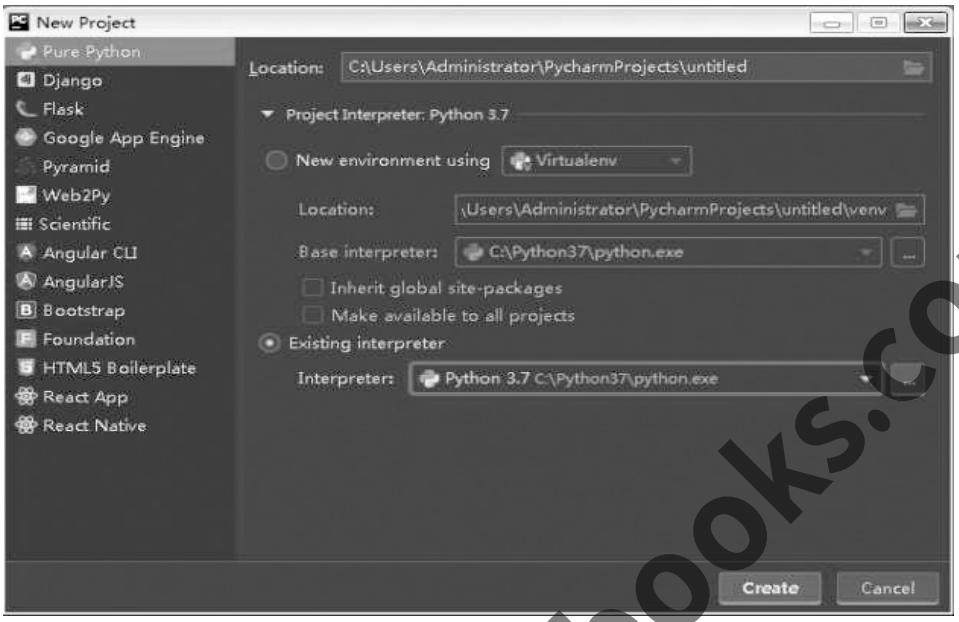

图 1.22　创建项目

步骤 6:设置开始菜单目录名称,如图 1.23 所示。

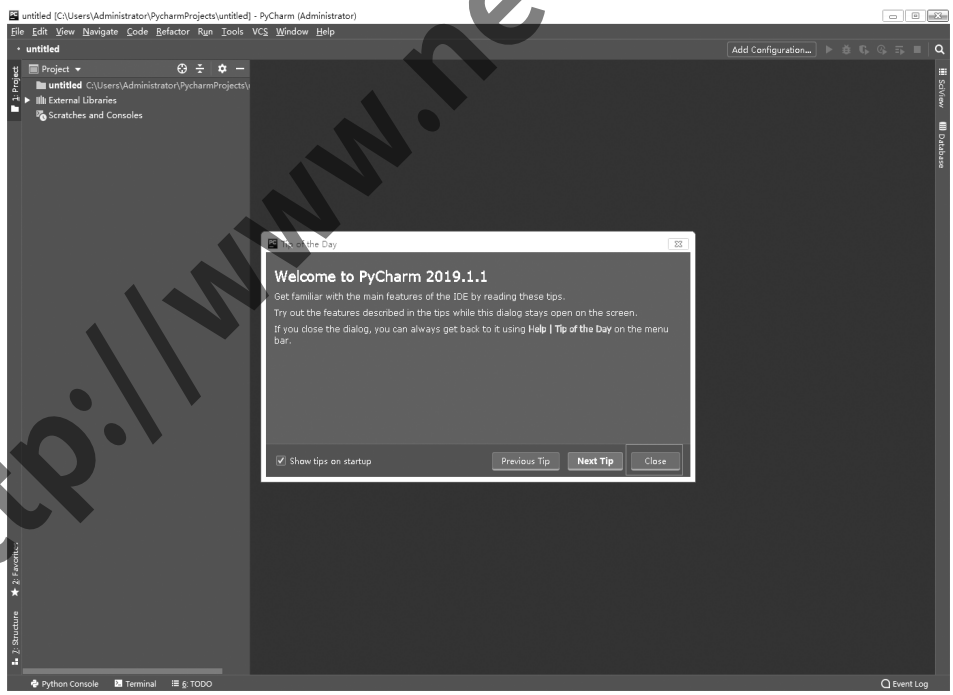

图 1.23　启动项目欢迎界面

步骤 7:选择项目名称单击右键,创建 Python 文件,如图 1.24 所示。

步骤 8:设置主题样式和字体大小,如图 1.25 所示。

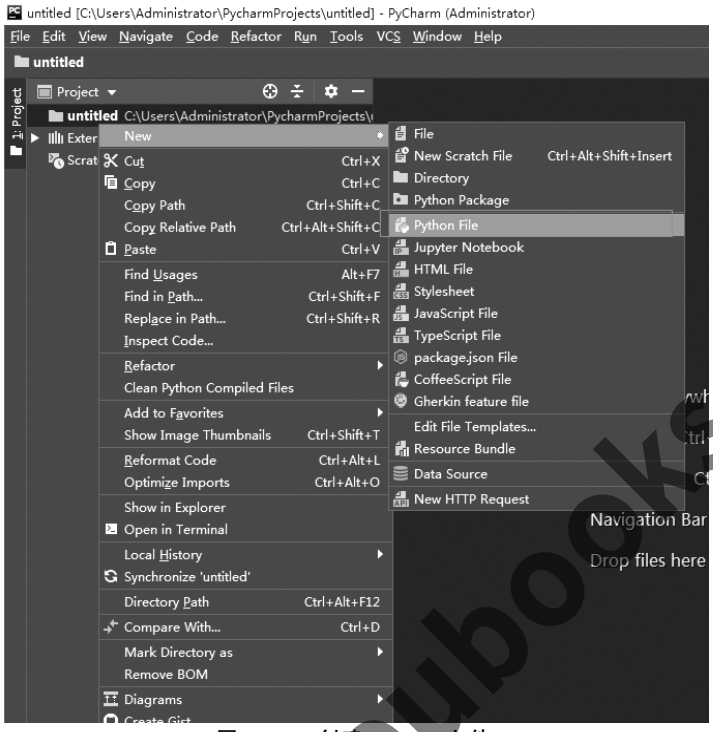

图 1.24　创建 python 文件

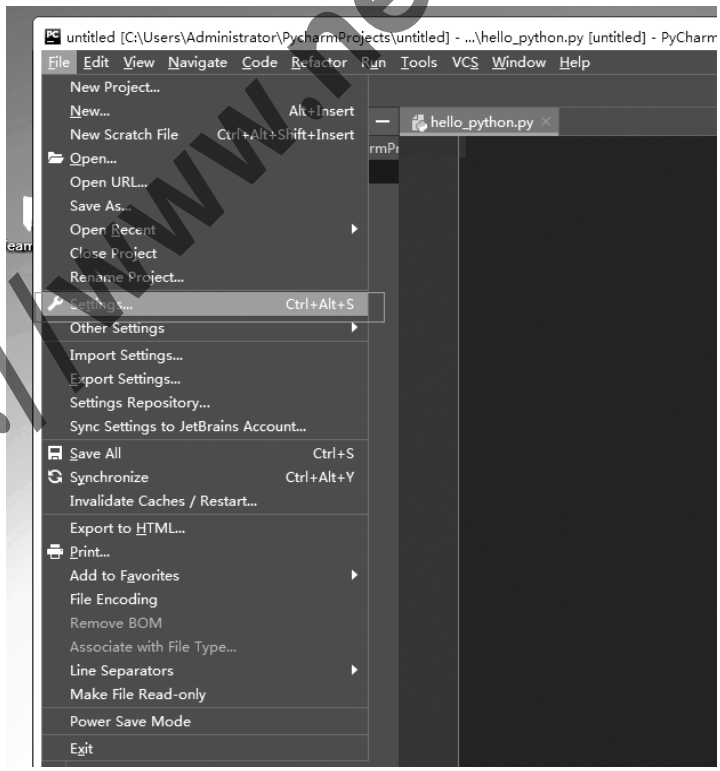

图 1.25　选择主题

步骤 9：单击 Appearance，选择 Theme 下拉框的 Intelij，切换到白色主题，如图 1.26 所示。

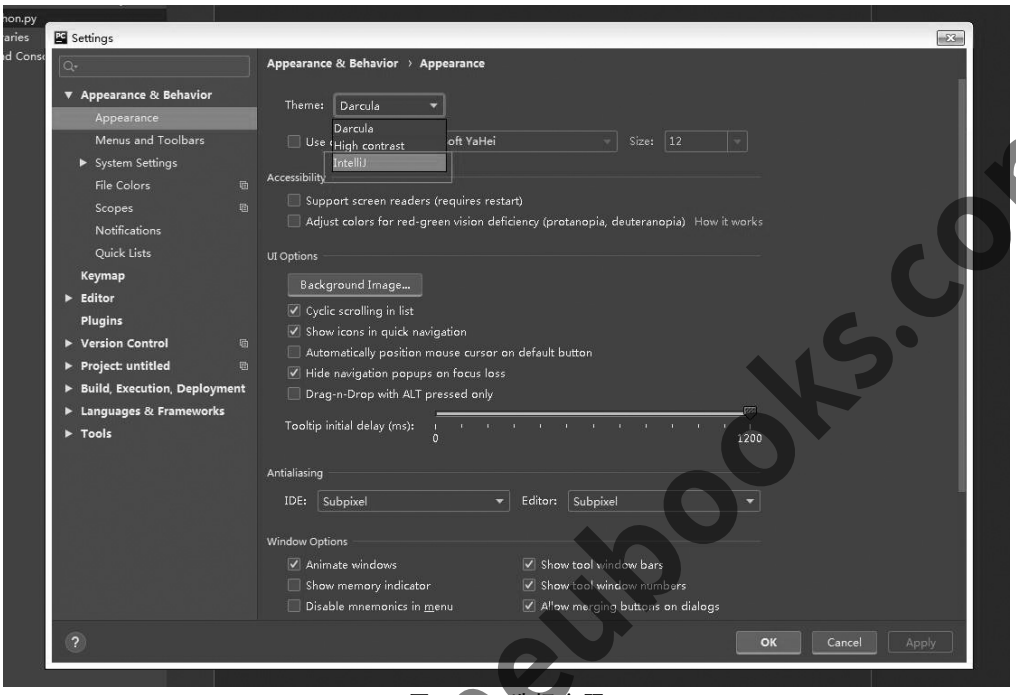

图 1.26　选择主题

步骤 10：选择 Editor 下的 Font，在 Size 框中输入字体大小，如图 1.27 所示。

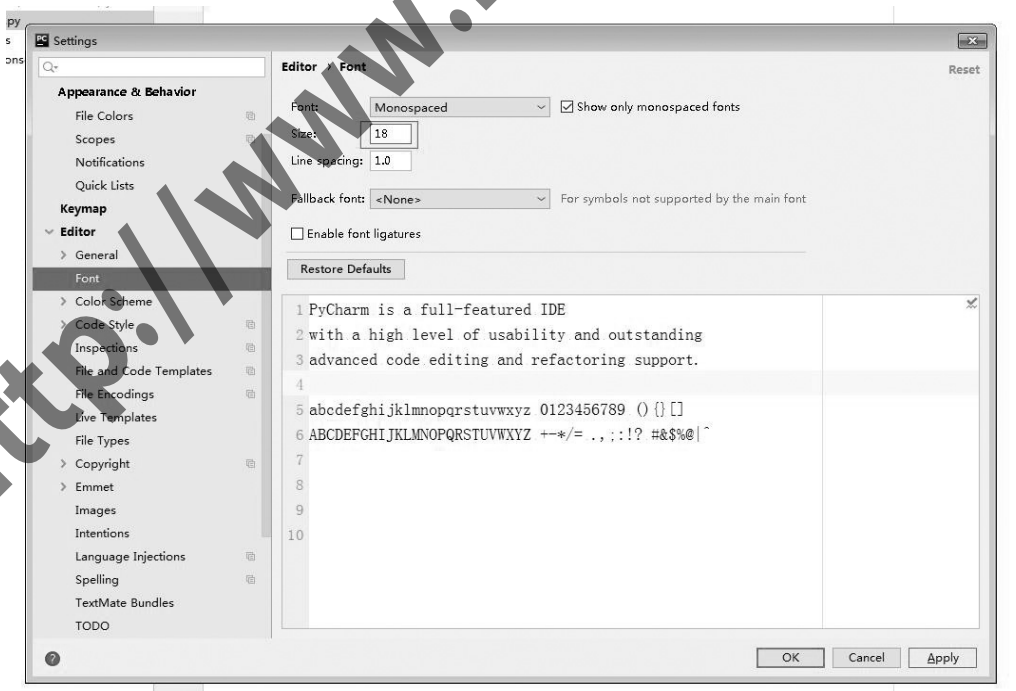

图 1.27　设置字体大小

步骤 11：在文件中输入内容，如图 1.28 所示。

在 Python 中，使用内置的 print()函数将结果输出到标准控制台上，基本语法为：

```
print(输出内容)
```

本例中输入以下内容

```
print("hello world")
```

代码写了之后需要添加注释，反映该段程序的逻辑思路、核心功能、变更情况、输入参数、输出结果等信息。对于单行注释，是在该行文本前加“#”号。

```
#注释内容
```

多行文本注释，使用三个成对的双引号或者单引号表示。

```
'''
注释内容 1
注释内容 2
.......
'''
```

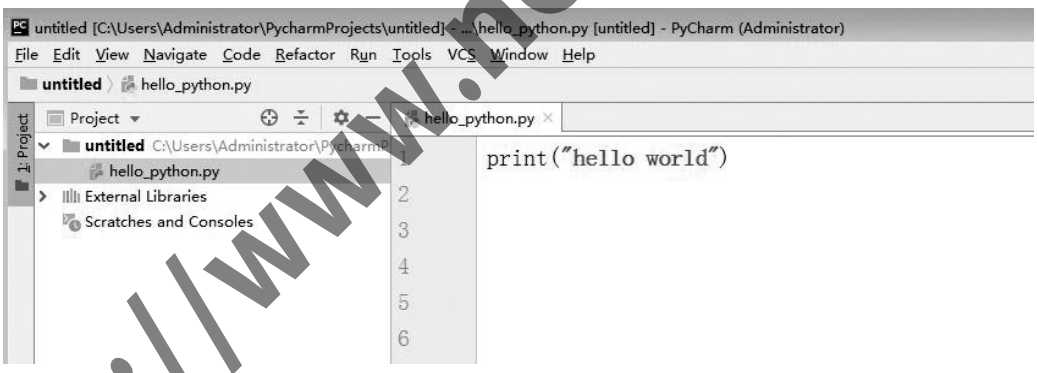

图 1.28　打印内容

这里的“hello word”表示字符串，是 Python 数据类型中的一种，Python 常见的数据类型有四种：整型、浮点型、布尔型、字符串型。在 Python 中整型就是整数，用 int 表示，学生成绩管理系统中学生的成绩就定义为整数型；浮点型就是带小数点的数字，用 float 表示；布尔型是特殊的整数，用 bool 表示，它只有两个值，分别是 True 和 False，如果将布尔型用于整数运算，则 True 代表 1，False 代表 0；字符串是一种表示文本的数据类型，用 str 表示。

步骤 12：在文本编辑区，单击右键，运行脚本文件，如图 1.29 所示。

步骤 13：在控制台观察输出结果，如图 1.30 所示。

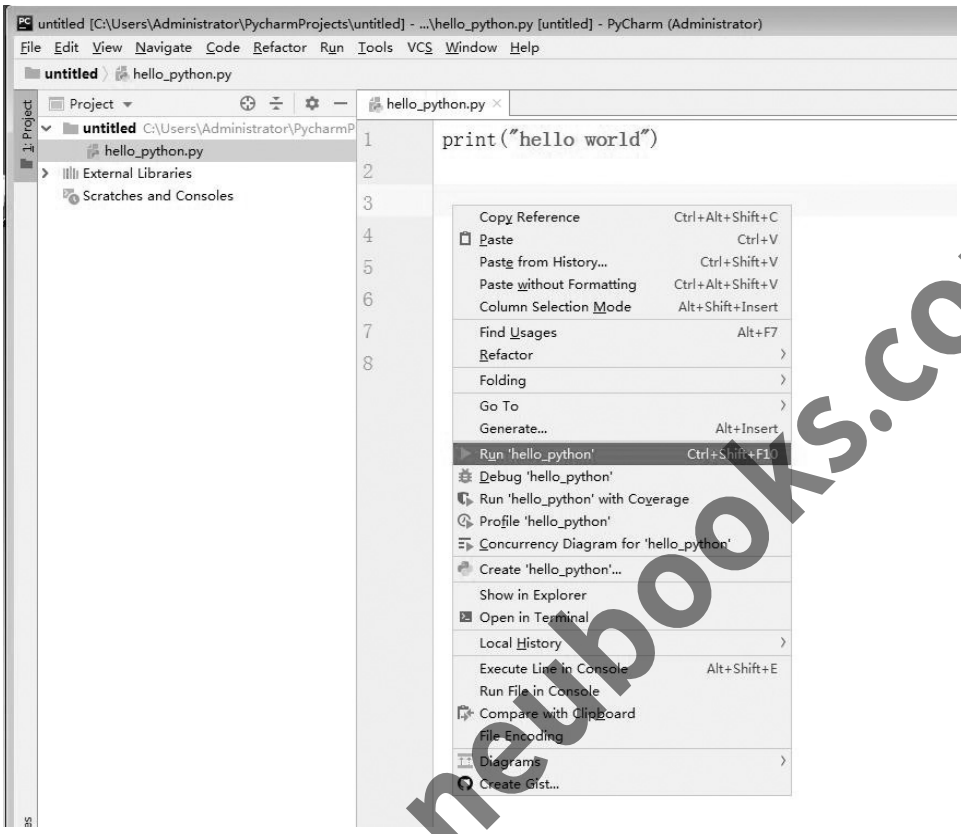

图 1.29　运行脚本

图 1.30　运行结果

1.3.2　主函数设计

函数是 Python 结构的基本组成单元，让我们的功能实现更加清晰灵活，提高应用的模块性和代码的重复利用率。

1.3.2.1 定义变量

在学生成绩管理系统实现过程中需要定义一些变量来存储一些必要的值，变量是计算机内存中的一块区域，用来存储规定范围内的值，在程序运行过程变量的值可以改变。在Python中，直接赋值即可创建各种类型的变量，而不需要先定义后使用。变量的定义有三个要素：一是变量名需要是合法标识符（长度不受限制，只能是字母、数字或者下划线，且第一个字符不能是数字，区分大小写）；二是需要通过“=”为变量赋值；三是变量名尽力做到“见名知意”。变量赋值的语法格式：

```
变量名=value
```

例如在录入成绩环节定义 mark 为循环标记标量，来判定是否进行循环：

```
mark=True
```

1.3.2.2 认识函数的基本概念

1. 函数定义的基本规则

(1)函数的声明以 def 关键词开头，后接函数标识符名称（即函数名）、括号()和冒号：。

(2)函数名必须以下划线或者字母开头，可以包含数字、字母、下划线等组合，不可以包含标点符号。函数名也不能使用保留字。

(3)同一 Python 的脚本文件中定义的函数名称不能一样，否则后面的函数定义会覆盖前面的定义。

(4)声明的函数可以定义参数，任何需要传入的参数和自变量都必须放在函数名后的括号中。参数的个数不固定，可以没有，可以是一个，可以是两个。

(5)声明函数的函数体使用缩进，表示当前函数包裹内容。

(6)声明函数的第一行语句可以选择性地使用文档字符串，用于存放函数说明。

(7)如果函数需要返回值，使用 return [表达式]；如果不需要，return 后不接任何表达式或者不写 return，默认表示返回 None。

2. 函数定义的基本语法

```
def function_name( parameters ):
    "函数_文档字符串"
    function_body
    return [expression]
```

基本语法中包括几个结构部分，相关说明如下：

(1)function_name：函数标识符名称，即函数名；

(2)parameters：参数列表；

(3)function_body：函数体，即函数具体逻辑实现；

(4)expression：返回表达式。

因参数列表数量的不固定，默认情况下，出入参数值和参数名称是按照函数声明中定义

的参数顺序进行匹配的。

1.3.2.3　主函数设计

在系统开发中，通常定义一个名叫 main 的函数作为主函数。在学生成绩管理系统中，通过主函数调用 menu()函数来显示功能菜单，然后根据 if 语句判断用户的输入，选择执行各项功能，实现对学生成绩的录入、查询、显示、排序和统计功能。主界面运行效果如图 1.31 所示。

```
╔--------学生信息管理系统---------╗
|                                  |
|  ============== *功能菜单*==============  |
|                                  |
|    1 录入学生成绩                  |
|    2 查找学生成绩                  |
|    3 删除学生成绩                  |
|    4 修改学生成绩                  |
|    5 成绩排序                      |
|    6 统计成绩                      |
|    7 显示所有学生成绩              |
|    0 退出系统                      |
|  ========================================  |
|  说明：通过数字选择菜单            |
╚----------------------------------╝

请选择：
```

图 1.31　主界面运行效果图

1. 业务流程

在进行主函数设计时，需要先整理出业务流程和实现技术。根据学生成绩管理系统主函数所要实现的功能，设计如图 1.32 所示的业务流程，以便指导我们进行主函数的设计工作。

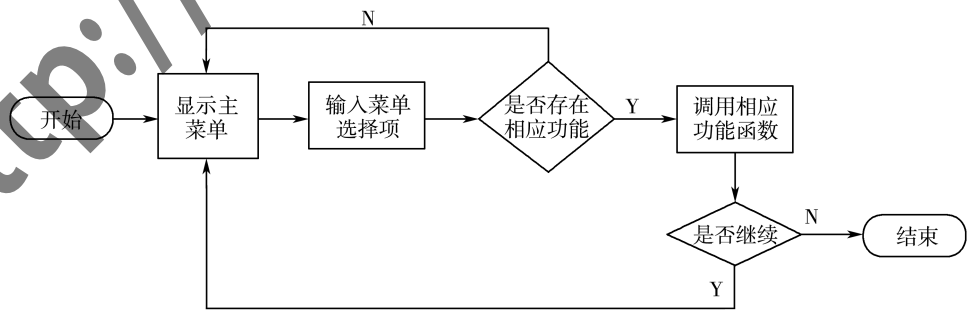

图 1.32　主函数业务流程图

2. 主函数实现

在学生成绩管理系统中，主要实现 4 个功能：学生成绩维护、查询成绩、成绩排序、成绩统计，所以一共需要定义如表 1.1 所示的 8 个函数来实现功能。

表 1.1　　菜单中的数字所对应功能表

数字编号	功能函数	功能描述
0	sys.exit()	通过系统函数退出系统
1	insert()	录入学生成绩
2	search()	查找学生成绩
3	delete()	删除学生成绩
4	modify()	修改学生成绩
5	sort()	成绩排序
6	statistics()	统计成绩
7	show()	显示所有成绩

3. 定义主函数 m

主函数 m()采用循环嵌套结构，Python 中有三种流程控制结构：顺序结构、选择结构、循环结构。

(1)顺序结构

程序从上往下执行按顺序执行每条语句，中间没有判断或跳转。

```
#示例
a=8
b=20
c=a+b
print("c 的值是",c)
```

运行结果：

```
c 的值是 28
```

(2)选择结构

使用 if 控制语句实现，主要有三种不同形式分别是单分支结构(if 语句)、双分支结构(if…else)和多分支结构(if…elif…else)

①用 if 语句实现单分支结构

语法如下：

```
if 表达式:
    语句块
```

其中，if 后面要加冒号。表达式可以是布尔型或变量，也可以是比较表达式或逻辑表达式，如果表达式为真，则执行“语句块”；如果表达式为假，就跳过语句块，执行后面的语句。

```
#示例
grade = 70
if grade >= 60:
    print("合格")
```

运行结果：

```
合格
```

②用 if…else 实现双分支结构

语法如下：

```
if 表达式：
    语句块 1
else：
    语句块 2
```

其中，表达式类型跟分支结构中一样，当表达式为真时，执行语句块 1；当表达式为假时，执行语句块 2。

```
#示例
name ="高等数学"
ifname == "体育":
    result = True
else :
    result = False

print("判断结果为:", result)
```

运行结果：

```
判断结果为:False
```

③用 if…elif…else 实现分支结构

```
语法如下：
if 表达式 1：
    语句块 1
elif 表达式 2：
    语句块 2
elif 表达式 3：
    语句块 3
…
else：
    语句块 n
```

当表达式为真时，执行相应的语句；如果表达式为假，则跳过该语句，进行下一个 elif 的判断。当表达式都为假时，执行 else 中的语句。

```
#示例
name ="高等数学"
if name == "体育":  # 判断 name 变量否为"体育"
```

```
    result = True
elif name == "Python":  # 判断 name 变量否为"Python"
    result = True
elif name == "英语":  # 判断 name 变量否为"英语"
    result = True
else:
    result = False  # 都不满足的情况执行此行语句

print("判断结果为:", result)
```

运行结果：

```
判断结果为: False
```

④if 语句的嵌套

前三种 if 语句可以相互嵌套，嵌套 if 结构可以通过外层语句和内层语句的协调增强程序的灵活性。

语法如下：

```
if 表达式 1:
    if 表达式 2:
        语句块 1
    else:
        语句块 2
else:
    if 表达式 3:
        语句块 3
    else:
        语句块 4
```

对表达式 1 进行判断：当表达式 1 为真时，判断表达式 2，如果表达式 2 为真，则执行语句块 1，否则执行语句块 2；当表达式 1 为假时，判断表达式 3，如果表达式 3 为真，执行语句块 3，否则执行语句块 4。

```
#示例
score=77
if score>=70:
    if score>=90:
        print("perfect")
    else:
        print("wonderful")
else:
    if score>=60:
        print("good")
    else:
        print("try again")
```

运行结果：

```
wonderful
```

(3)循环结构

循环结构主要是重复执行一组语句，主要包含 while 循环和 for 循环。

①while 循环

通过条件来控制是否要反复执行循环体中的语句，语法如下：

```
初始化变量
while 循环条件：
    循环体
```

其中，循环体是一组被重复执行的语句，当条件表达式为真时，则执行循环体中的语句，执行完毕再次对循环条件进行判断，如果为真，继续执行，直到表达式为假时退出循环。

```
#示例
data = ["高等数学", "体育", "英语"]
i = 0
count = len(data)
while True:
    if i < count:
        print("当前科目是:", data[i])
    i += 1
```

遍历集合中的每一个元素并输出，运行结果：

```
当前科目是：高等数学
当前科目是：体育
当前科目是：英语
```

②for 循环

for 循环主要用来遍历数据集合或者迭代器中的元素，语法如下：

```
for 循环变量 in 对象：
    循环体
```

其中，循环变量用来保存读取的值；对象为要遍历或迭代的对象，可以是任何有序的序列对象；循环体为一组被重复执行的语句。执行步骤为：尝试从对象中获取第一个元素，如果能获取到，将获得到的元素赋值给循环变量，执行循环体，然后继续从对象中获取下一个元素重复上一个循环，直到无法获取元素终止循环。

```
#示例
data = ["高等数学", "体育", "英语"]
for item in data:
    print("当前科目是:", item)
```

遍历集合中的每一个元素并输出，运行结果：

```
当前科目是：高等数学
当前科目是：英语
当前科目是：英语
```

③循环嵌套

实际应用中，一个循环可能无法满足要求，这就需要在一个循环体中嵌入另一个循环体，形成循环嵌套。for 循环和 while 循环都可以进行循环嵌套。例如在 while 循环中嵌套 for 循环，格式如下：

```
while 循环条件：
    for 循环变量 in 对象：
        循环体 2
    循环体 1
```

```
#示例：九九乘法表
for i in range(1,10)：
    j = 1
    while j <= i：
        print("%d * %d= %d\t" % (j, i, i * j),end = "")
        j += 1
    print()
```

运行结果：

```
1 * 1=1
1 * 2=22 * 2=4
1 * 3=32 * 3=63 * 3=9
1 * 4=42 * 4=83 * 4=124 * 4=16
1 * 5=52 * 5=103 * 5=154 * 5=205 * 5=25
1 * 6=62 * 6=123 * 6=184 * 6=245 * 6=306 * 6=36
1 * 7=72 * 7=143 * 7=214 * 7=285 * 7=356 * 7=427 * 7=49
1 * 8=82 * 8=163 * 8=244 * 8=325 * 8=406 * 8=487 * 8=568 * 8=64
1 * 9=92 * 9=183 * 9=274 * 9=365 * 9=456 * 9=547 * 9=638 * 9=729 * 9=81
```

其他形式的嵌套不再一一列举

(4)循环跳转语句

在实际开发中，循环语句不一定按照循环条件完成所有内容的遍历，这时就需要用跳转语句 break 或 continue 把控制转移到循环或程序的其他部分。

①break 语句

break 语句用来终止当前循环，然后执行循环后面的语句。

break 只对当前循环有效，如果使用嵌套循环，内层的 break 只会终止内层循环语句的执行，不会终止外层循环。

break 语句只能在 while 循环或 for 循环中使用。

```
#示例
data = ["高等数学", "体育", "英语"]
i = 0
count = len(data)
while True:
    if i < count:
        print("当前科目是:", data[i])
    else:
        print("终止循环")
        break
    i += 1
```

运行结果：

```
当前科目是：高等数学
当前科目是：体育
当前科目是：英语
终止循环
```

②continue 语句

continue 语句用来强制一个循环提前返回，调过此次循环剩余部分，开始下一次循环。只能在 while 循环或 for 循环中使用。

```
#示例
data = ["高等数学", "体育", "英语"]
for i in data:
    if i == "高等数学":
        continue
    print(i)
```

运行结果：

```
体育
英语
```

③try-except 异常处理语句

Python 使用 try…except…来进行异常信息捕获。Exception 是所有异常的基类，因此使用 except Exception 可以捕获到任意类型的异常信息，finally 表示异常捕获后最终要执行的代码。

主函数 m()代码实现如下：

```
def m():
    while (True):
        menu()  # 显示菜单
        option = input("请选择:")  # 选择菜单项
        option_str = re.sub("\D", "", option)  # 提取数字
        if option_str in ['0', '1', '2', '3', '4', '5', '6', '7']:
            option_int = int(option_str)
            if option_int == 0:  # 退出系统
                print('您已退出学生成绩管理系统!')
                sys.exit()
            elif option_int == 1:  # 录入学生成绩信息
                insert()
            elif option_int == 2:  # 查找学生成绩信息
                search()
            elif option_int == 3:  # 删除学生成绩信息
                delete()
            elif option_int == 4:  # 修改学生成绩信息
                modify()
            elif option_int == 5:  # 学生成绩排序
                sort()
            elif option_int == 6:  # 统计学生总数
                statistics()
            elif option_int == 7:  # 显示所有学生信息
                show()
```

注意:在退出系统中调用了 sys.exit(),需要在程序最前面加上 import sys 。

在一个.py 文件中,如果不是在定义函数,也就是说不是在 def 关键字的内嵌结构内,Python 会默认其余部分函数是 main 函数,并自动执行。在正规工程中,一般都会将 main 函数写为:

```
if __name__ == "__main__":
```

可以理解为“if name==”main“:” 这一句与 c 中的 main()函数所表述的是一致的,即作为入口。

为学生成绩管理系统创建程入口,然后调用 m()函数:

```
if __name__ == "__main__":
    m()
```

其中==是比较运算符,Python 中常见的运算符主要有算数运算符(如表 1.2)、比较运算符(如表 1.3)、赋值运算符(如表 1.4)、位运算符(如表 1.5)、逻辑运算符(如表 1.6),具体符号形式和描述如表 1.2 到表 1.6 所示。

表 1.2　　算术运算符

运算符名称	描述
+	两个数据相加
-	两个数据相减
*	两个数据相乘
/	两个数据相除
%	两个数据取模
* *	求次方
//	向下取整,返回商的整数部分

```
#示例:算数运算符
a = 8
b = 20

c = a + b
print("a+b=", c)

c = a - b
print("a-b=", c)

c = a * b
print("a*b=", c)

c = a / b
print("a/b=", c)

c = a % b
print("a%b=", c)

b = 3
c = a // b
print("a//b=", c)

c = a * * b
print("a * * b=", c)
```

运行结果:

```
a+b= 28
a-b= -12
a*b= 160
a/b= 0.4
a%b= 8
a//b= 2
a * * b= 512
```

表 1.3 比较运算符

运算符名称	描述
>	大于
<	小于
>=	大于等于
<=	小于等于
==	等于
!=	不等于

```
#示例:比较运算符
a = 8
b = 20

c = a > b
print("a > b 计算结果:", c)

c = a < b
print("a < b 计算结果:", c)

c = a >= b
print("a >= b 计算结果:", c)

c = a <= b
print("a <= b 计算结果:", c)

c = a == b
print("a==b 计算结果:", c)

c = a != b
print("a! =b 计算结果:", c)
```

运行结果:

```
a > b 计算结果: False
a < b 计算结果: True
a >= b 计算结果: False
a <= b 计算结果: True
a==b 计算结果: False
a! =b 计算结果: True
```

表 1.4　　赋值运算符

运算符名称	描述
=	赋值
+=	加等
-=	减等
*=	乘等
/=	除等
%=	模等
**=	幂运算
//=	除等取整

```
#示例:赋值运算符
a = 8
b = 20

c = a + b
print("经过 c=a + b 计算结果后的值为:", c)

a += b
print("经过 a += b 计算结果后 a 的值为:", a)

a -= b
print("经过 a -= b 计算结果后 a 的值为:", a)

a *= b
print("经过 a *= b 计算结果后 a 的值为:", a)

a /= b
print("经过 a /= b 计算结果后 a 的值为:", a)

a %= b
print("经过 a %= b 计算结果后 a 的值为:", a)

a **= b
print("经过 a **= b 计算结果后 a 的值为:", a)

a //= b
print("经过 a //= b 计算结果后 a 的值为:", a)
```

运行结果：

```
经过 c=a + b 计算结果后 c 的值为：28
经过 a += b 计算结果后 a 的值为：28
经过 a -= b 计算结果后 a 的值为：8
经过 a *= b 计算结果后 a 的值为：160
经过 a /= b 计算结果后 a 的值为：8.0
经过 a %= b 计算结果后 a 的值为：8.0
经过 a **= b 计算结果后 a 的值为：1.152921504606847e+18
经过 a //= b 计算结果后 a 的值为：5.764607523034235e+16
```

表 1.5　　位运算符

运算符名称	描述
<<	左移动运算符，将每个为往左移动，低位补 0
>>	右移动运算符，将每个为往左移动，高位补 0
&	按位与运算，相同位上值为 1 结果就为 1，否则为 0
|	按位或运算，相同位上值其中一个为 1 结果就为 1，否则为 0
^	按位异或运算符，相同位上值不相同位 1
~	按位异或运算符，相同位上值取反，比如是 1 就取 0，是 0 就取 1

```
#示例：位运算符
a = 8
b = 20

c = a << 3
print("a << 3=", c)

c = a >> 3
print("a >> 3=", c)

c = a & b
print("a & b=", c)

c = a | b
print("a | b=", c)

c = a ^ b
print("a ^b=", c)

c = ~a
print("~a=", c)
```

运行结果：

```
a << 3 = 80
a >> 3 = 1
a & b = 0
a | b = 30
a ^ b = 30
~a = -11
```

表 1.6　　逻辑运算符

运算符名称	描述
and	与运算，多个条件同时为 True 则结果为 True
or	或运算，其中一个条件为 True 则结果为 True
not	对布尔值取反，not True 结果为 False，not False 结果为 True

```
#示例：逻辑运算符
a = 8
b = 20

if a > 0 and b > 0:
    print("a 和 b 都大于 0")
else:
    print("a 和 b 其中一个小于等于 0")

b = -20
if a >= 0 or b >= 0:
    print("a 和 b 其中一个大于等于 0")
else:
    print("a 和 b 都小于 0")

c = a - b
print("c 的值为：", c)
if not (c >= 0):
    print("c 小于等于 0")
else:
    print("c 大于 0")
```

运行结果：

```
a 和 b 都大于 0
a 和 b 其中一个大于等于 0
c 的值为： 28
c 大于 0
```

4. 自定义菜单函数 menu

在了解了函数的规则和基本语法之后，我们可以将部分小功能包裹起来定义在自己的函数中。函数分为有参数函数和无参数函数，在学生成绩管理系统中，我们只需要了解无参数函数的定义方式。自定义一个无参函数，在控制台输出学生成绩管理系统的功能菜单。具体代码如下所示：

```
def menu():
    # 输出菜单
    print("""
    ╔————————————学生信息管理系统————————————╗
    |                                          |
    |   ========  *功能菜单*  ========         |
    |                                          |
    |    1 录入学生成绩                        |
    |    2 查找学生成绩                        |
    |    3 删除学生成绩                        |
    |    4 修改学生成绩                        |
    |    5 成绩排序                            |
    |    6 统计成绩                            |
    |    7 显示所有成绩                        |
    |    0 退出系统                            |
    |    ======================                |
    |    说明:通过数字选择菜单                 |
    ╚——————————————————————————————————————╝
    """)
```

函数的自定义提供了函数名、指定参数和函数体代码块。而自定义函数的目的就是将独立的功能抽离出来进行模块封装使用，所以在函数的基本结构完成之后，我们就可以直接在 Python 文件中调用执行，或者通过另一个函数进行调用执行，调用无参数函数只需要通过函数名就可以实现调用。在 m()函数中调用了主菜单函数 menu()(m()函数的第 3 行代码)。

1.3.3 学生成绩维护

学生成绩维护模块主要包含学生成绩录入、学生成绩修改、学生成绩删除，每个学生都对应多个数据，在 Python 中处理多个数据可以采用列表(list)、元组(tuple)、字典 (diet) 和集合(set) 4 种数据结构。

(1)列表的应用

列表是用来存储多个数据的数据结构。Python 使用 list 来创建列表，使用一对中括号表示。列表是一种数据结构，类似于 C/C++语言中的数组。列表是一种数据容器，可以存放任意类型的数据。列表中的数据是有序的，可以使用下标进行访问，下标索引范围是 0 到列表长度减 1。列表是可变对象，意味着可以往列表中添加元素，也可以修改和删除元素。

列表对象有非常多的方法，比如计算列表元素个数、在列表指定位置插入元素、移除某个元素。

由于列表支持使用下标访问，因此还可以按下标范围进行筛选数据，这称为切片。

(2)元组的应用

Python 使用 Tuple 来创建元组，使用一对小括号表示。元组与列表类似，但是元组是不可变对象，意味着元组对象不能添加、删除元素。

(3)字典的应用

Python 使用 Dict 来创建字典，使用一对大括号表示。字典存储的是键值对结构的数据，是可变类型。与列表、元组不一样的地方是，不能使用下标索引进行访问，只能使用键名进行访问。

(4)集合的应用

集合(set)是一种无序的并且里面存放不同元素的序列。集合可以使用大括号 { } 或者 set() 函数创建集合，注意：创建一个空集合必须用 set() 而不是 { }，因为 { } 是用来创建一个空字典。一般集合常用的场景是去重(如：列表去重)和关系测试(如：取交集、取并集、取差集等)。

1.3.3.1　定义录入学生成绩函数 insert

用户在控制台录入学生 ID、姓名、数学成绩、python 成绩、体育成绩等信息，每个学生的成绩保存在字典中，并将字典添加到列表里保存，录入完成会提示是否继续添加，继续添加输入“y”，不再添加输入“n”，录入结束。效果如图 1.33 所示。

```
╔--------学生信息管理系统---------╗
|                                  |
|  ============ *功能菜单* ======  |
|                                  |
|    1 录入学生成绩                |
|    2 查找学生成绩                |
|    3 删除学生成绩                |
|    4 修改学生成绩                |
|    5 成绩排序                    |
|    6 统计成绩                    |
|    7 显示所有成绩                |
|    0 退出系统                    |
|  ================================ |
|  说明：通过数字选择菜单          |
╚----------------------------------╝

请选择：1
请输入ID（如 1001）：1001
请输入名字：张三
请输入数学成绩：90
请输入Python成绩：95
请输入体育成绩：90
是否继续添加？（y/n）：
```

图 1.33　成绩录入界面图

学生成绩录入业务流程如图 1.34：

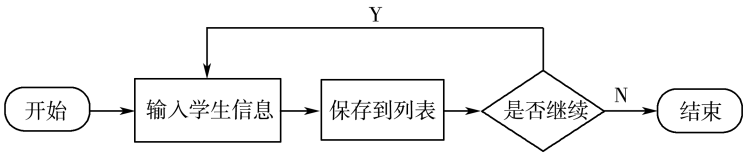

图 1.34 学生成绩录入业务流程图

定义函数 insert()实现学生信息的录入功能，它是一个无参无返回值函数，在该函数中设置嵌套循环，在该循环中使用内置的 input()函数接收用户键盘输入的内容(ID、姓名、数学成绩、python 成绩、体育成绩)，基本语法为：

```
var=input("提示文字")
```

以下为常见的输入语句：

```
#提示输入学号，并保存到变量 studentId 中
    studentId = input("请输入学生的学号：")
#提示输入 Python 成绩，并保存到变量 python 中，对输入的成绩转换为整型 python = int(input
("请输入 Python 成绩："))
```

用户输入的内容如果符合要求则保存到字典中，再将该字典添加到列表中保存，代码实现如下：

```
stuInfo={}
def insert():
    mark = True  # 是否继续添加
    while mark:
        id = input("请输入 ID(如 1001)：")
        if not id:  # ID 为空，跳出循环
            break
        name = input("请输入名字：")
        if not name:  # 名字为空，跳出循环
            break
        try:
            math = int(input("请输入数学成绩："))
            python = int(input("请输入 Python 成绩："))
            PE = int(input("请输入体育成绩："))
        except:
            print("输入无效，不是整型数值....重新录入信息")
            continue
        stdent = {"id":id, "name": name, "math": math, "python": python, "PE": PE}  # 将
输入的学生信息保存到字典
        stuInfo[id]=stdent  # 将学生字典添加到列表中
        inputMark = input("是否继续添加？(y/n)：")
        if inputMark == "y":  # 继续添加
            mark = True
        else:  # 不继续添加
            mark = False
    print("学生信息录入完毕!!!")
```

1.3.3.2　定义删除学生成绩函数 delete

用户通过输入学生的 ID，查找相应学生的信息进行删除，如果找到对相应学生的信息进行删除，如果没找到提示“没有找到 ID 为 xx 的学生信息...”删除完一个后，提示是否继续删除，继续删除输入“y”，不再删除输入“n”，录入结束。

学生成绩删除业务流程如图 1.35：

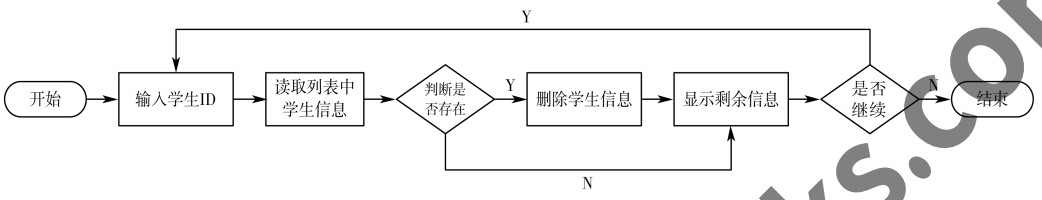

图 1.35　学生成绩删除业务流程图

定义函数 delete()实现学生成绩的删除功能，它是一个无参无返回值函数，该函数中使用嵌套循环，具体代码如下：

```
def delete():
    mark = True  # 标记是否循环
    while mark:
        studentId = input("请输入要删除的学生 ID:")
        if studentId is not "":  # 判断要删除的学生是否存在
            ifdel = False  # 标记是否删除
            if studentId in stuInfo.keys():  # 如果存在学生信息
                stuInfo.pop(studentId)
                print("ID 为 %s 的学生信息已经被删除..." % studentId)
            else:
                print("没有找到 ID 为 %s 的学生信息..." % studentId)

            show()  # 显示全部学生信息
            inputMark = input("是否继续删除? (y/n):")
            if inputMark == "y":
                mark = True  # 继续删除
            else:
                mark = False  # 退出删除学生信息功能
```

1.3.3.3　定义修改学生成绩函数 modify

根据用户输入的学生 ID，查找对应学生的信息，如果找到，提示“找到了这名学生，可以修改他的信息!”，用户可对相应信息进行修改，如果没找到，提示“不存在该学生成绩！修改失败”。最后提示是否继续修改，继续修改“y”，不再修改输入“n”，修改结束。

学生成绩修改业务流程如图 1.36：

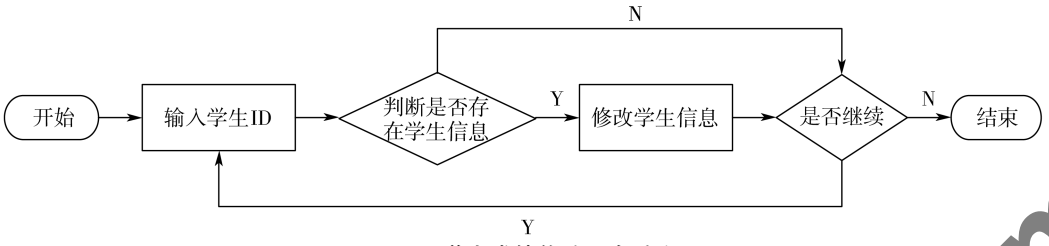

图 1.36　学生成绩修改业务流程图

定义函数 modify()实现学生成绩的修改功能，它是一个无参无返回值函数，在该函数中调用 show()函数显示学生的成绩，根据用户输入的 ID，遍历保存学生信息的列表，将每个元素转换为字典，根据用户输入的信息更新学生的存储信息，具体实现如下：

```
def modify():
    show()  # 显示全部学生信息
    studentid = input("请输入要修改的学生 ID:")
    d={}
    if studentid in stuInfo.keys():  # 是否为要修改的学生
                print("找到了这名学生,可以修改他的信息!")
                while True:  # 输入要修改的信息
                    try:
                        d["name"] = input("请输入姓名:")
                        d["math"] = int(input("请输入数学成绩:"))
                        d["python"] = int(input("请输入 Python 成绩:"))
                        d["PE"] = int(input("请输入体育成绩:"))
                    except:
                        print("您的输入有误,请重新输入。")
                    else:
                        break  # 跳出循环
                stuInfo[studentid]['name']=d["name"]
                stuInfo[studentid]["math"]=d["math"]
                stuInfo[studentid]["python"]=d["python"]
                stuInfo[studentid]['PE']=d["PE"]
                print("修改成功!")
    else:
        print("不存在该学生成绩! 修改失败")
    mark = input("是否继续修改其他学生信息? (y/n):")
    if mark == "y":
        modify()  # 重新执行修改操作
```

其中 show()函数定义如下：

```
    def show():
        student_new = []
        for s in stuInfo.keys():
            student_new.append(stuInfo[s])
        show_student(student_new)
defshow_student(studentList):
    if not studentList:
        print("无数据信息 \n")
        return
    format_title = "{:^6}{:^12}\t{:^8}\t{:^10}\t{:^10}\t{:^10}"
    print(format_title.format("ID", "名字", "数学成绩", "Python 成绩", "体育成绩", "总成绩"))
    format_data = "{:^6}{:^12}\t{:^12}\t{:^12}\t{:^12}\t{:^12}"
    for info in studentList:
        print(format_data.format(info.get("id"), info.get("name"), str(info.get("math")),
str(info.get("python")),
                          str(info.get("PE")),
                          str(info.get("math") + info.get("python") + info.get("PE")).
center(12)))
```

1.3.4　定义查询学生成绩函数 search

查询成绩包含两种方式，按照 ID 和按照姓名查询，用户输入学生的 ID 或者姓名后，如果学生信息存在则返回学生信息，不存在则显示“对不起，没有此学生。”。最后提示是否继续查询，继续查询“y”，不再查询输入“n”，修改结束。

学生成绩查询业务流程如图 1.37：

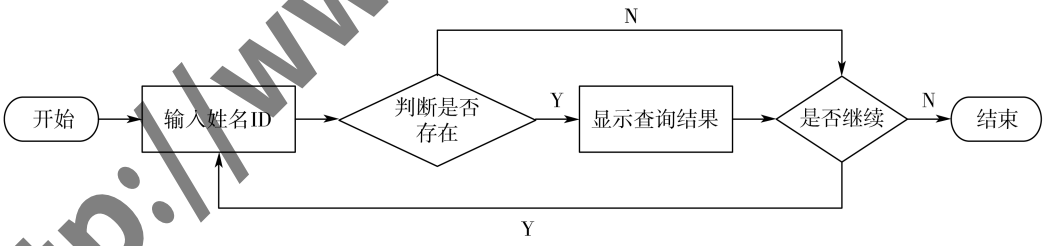

图 1.37　学生成绩查询业务流程图

定义函数 search()实现学生成绩的修改功能，它是一个无参无返回值函数，采用循环嵌套结构，具体实现如下：

```
    def search():
        mark = True
        student_query = []  # 保存查询结果的学生列表
        while mark:
                id = ""
                name = ""
                mode = input("按 ID 查输入 1;按姓名查输入 2:")
```

```
        if mode == "1":
            id = input("请输入学生 ID:")
        elif mode == "2":
            name = input("请输入学生姓名:")
        else:
            print("您的输入有误,请重新输入!")
            search()  # 重新查询
        if id is not "":  # 判断是否按 ID 查
            if id in stuInfo.keys():
                student_query.append(stuInfo[id])  # 将找到的学生信息保存到列表中
        if name is not "":
            for lst in stuInfo.keys():
                if stuInfo[lst]['name']==name:
                    student_query.append(stuInfo[lst])
        if len(student_query)==0:
            print("对不起,没有此学生。")
        else:
            show_student(student_query)
        inputMark = input("是否继续查询?(y/n):")
        if inputMark == "y":
            mark = True
        else:
            mark = False
```

1.3.5 定义学生成绩排序函数 sort

学生成绩排序功能主要包括按照数学成绩、Python 成绩、体育成绩、总成绩排序四种方式进行升序或者降序进行排序,根据用户的选择实现不同的排序方式。

学生成绩排序业务流程如图 1.38:

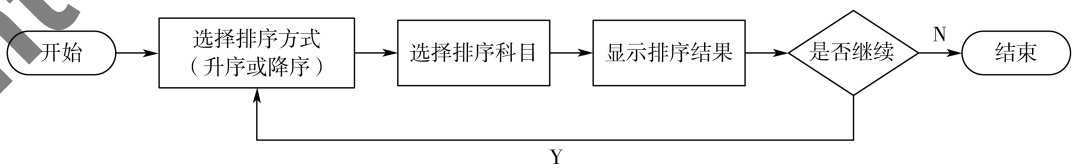

图 1.38 学生成绩排序业务流程

定义排序函数 sort()实现学生成绩的排序功能,它是一个无参、无返回值函数,采用循环嵌套结构,具体实现如下:

```
def sort():
    show()  # 显示全部学生信息
    student_new = []
    for list in stuInfo.keys():
        student_new.append(stuInfo[list])  # 将转换后的字典添加到列表中

    ascORdesc = input("请选择(0 升序;1 降序):")
    if ascORdesc == "0":  # 按升序排序
        ascORdescBool = False            # 标记变量,为 False 表示升序排序
    elif ascORdesc == "1":  # 按降序排序
        ascORdescBool = True             # 标记变量,为 True 表示降序排序
    else:
        print("您的输入有误,请重新输入!")
        sort()
    mode = input("请选择排序方式(1 按数学成绩排序;2 按 Python 成绩排序;3 按体育成绩排序;0 按总成绩排序):")
    if mode == "1":  # 按数学成绩排序
        student_new.sort(key=lambda x: x["math"], reverse=ascORdescBool)
    elif mode == "2":  # 按 Python 成绩排序
        student_new.sort(key=lambda x: x["python"], reverse=ascORdescBool)
    elif mode == "3":  # 按体育成绩排序
        student_new.sort(key=lambda x: x["PE"], reverse=ascORdescBool)
    elif mode == "0":  # 按总成绩排序
        student_new.sort(key=lambda x: x["math"] + x["python"] + x["PE"], reverse=ascORdescBool)
    else:
        print("您的输入有误,请重新输入!")
        sort()
    show_student(student_new)  # 显示排序结果
    mark = input("是否继续排序? (y/n):")
    ifmark == "y":
        sort()  # 重新执行排序操作
```

1.3.6　定义学生成绩统计函数 statistics

学生成绩统计主要实现每个科目 90—100 分、80—90 分、70—80 分、60—70 分、60 分以下五个分数段学生的人数和比例，根据选定的科目显示具体内容。

学生成绩统计业务流程如图 1.39：

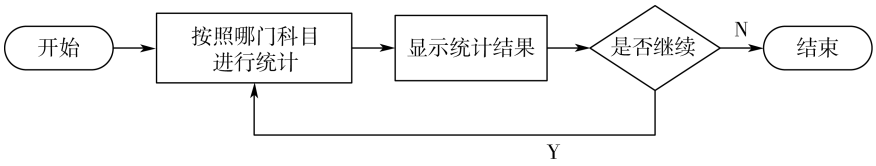

图 1.39　学生成绩统计业务流程图

定义函数 statistics() 实现学生成绩的统计功能，它是一个无参、无返回值函数，采用循环嵌套结构，具体实现如下：

```
def statistics():
    while 1:
        numStu=len(stuInfo)
        _9,_8,_7,_6,_=[0,0,0,0,0] #计算每个分段的人数
        mode=input("要查看什么分数的统计(1.数学 2.Python 3.体育):")
        subject="#定义科目变量 subject
        if mode == '1':
            subject='math'
        elif mode == '2':
            subject='PE'
        elif mode == '3':
            subject='python'
        else:
            print('对不起,没有这个选项,请重新输入!')
            continue
        for lst in stuInfo.keys():
            if stuInfo[lst][subject]>=90:
                _9+=1#统计 90 分及以上的人数
            elif stuInfo[lst][subject]>=80:
                _8+=1#统计 80 分及以上且小于 90 分的人数
            elif stuInfo[lst][subject]>=70:
                _7+=1#统计 70 分及以上且小于 80 分的人数
            elif stuInfo[lst][subject]>=60:
                _6+=1#统计 60 分及以上且小于 70 分
            else:
                _+=1#统计 60 分以下的人数
        print(subject+"的成绩统计如下:")
        print("一共有 %d 名学生!" % numStu)
        print("90—100 分 {} 人,占总人数的 {}%".format(_9,_9/numStu*100))
        print("80—90 分 {} 人,占总人数的 {}%".format(_8,_8/numStu*100))
        print("70—80 分 {} 人,占总人数的 {}%".format(_7,_7/numStu*100))
        print("60—70 分 {} 人,占总人数的 {}%".format(_6,_6/numStu*100))
        print("60 分以下 {} 人,占总人数的 {}%".format(_,_/numStu*100))
        mark = input("是否继续统计?(y/n):")
        if mark == "y":
            statistics()  # 重新执行排序操作
```

1.4　典型工作环节 4：软件测试

软件测试(Software Testing)，描述一种用来促进鉴定软件的正确性、完整性、安全性和质量的过程。换句话说，软件测试是一种实际输出与预期输出之间的审核或者比较过程。软件测试的经典定义是：在规定的条件下对程序进行操作，以发现程序错误，衡量软件质量，并对其能否满足设计要求进行评估的过程。软件测试的方法很多，包括静态测试、动态测试、黑盒测试、白盒测试，本项目体量较小，在这里采取白盒测试检查编码的正确性、合理性。白盒测试又称结构测试、透明盒测试、逻辑驱动测试或基于代码的测试。白盒测试是一种测试用例设计方法，盒子指的是被测试的软件，白盒指的是盒子是可视的，即清楚盒子内部的东西以及里面是如何运作的。“白盒”法全面了解程序内部逻辑结构、对所有逻辑路径进行测试。“白盒”法是穷举路径测试。在使用这一方案时，测试者必须检查程序的内部结构，从检查程序的逻辑着手，得出测试数据。

1. 测试录入功能

进入录入功能，学生的数学成绩输入文字也能显示如图 1.40。

```
╔═══════学生信息管理系统═══════╗
|                                |
|  ============== *功能菜单*==============  |
|                                |
|    1 录入学生成绩              |
|    2 查找学生成绩              |
|    3 删除学生成绩              |
|    4 修改学生成绩              |
|    5 成绩排序                  |
|    6 统计成绩                  |
|    7 显示所有学生成绩          |
|    0 退出系统                  |
|  ======================================  |
|  说明：通过数字选择菜单        |
╚═══════════════════════════════╝

请选择：1
请输入ID（如 1001）：1001
请输入名字：张三
请输入数学成绩：数学
```

图 1.40　成绩显示

分析原因在于如下代码段：

```
try:
    math = input("请输入数学成绩:")
```

未把 math 定义为整型，进行如下修改：

```
try:
    math = int(input("请输入数学成绩:"))
```

2. 测试查找功能

测试查找功能，当一次查找完成后，继续提示输入查找条件，不知道如何跳出查找功能，如图 1.41 所示

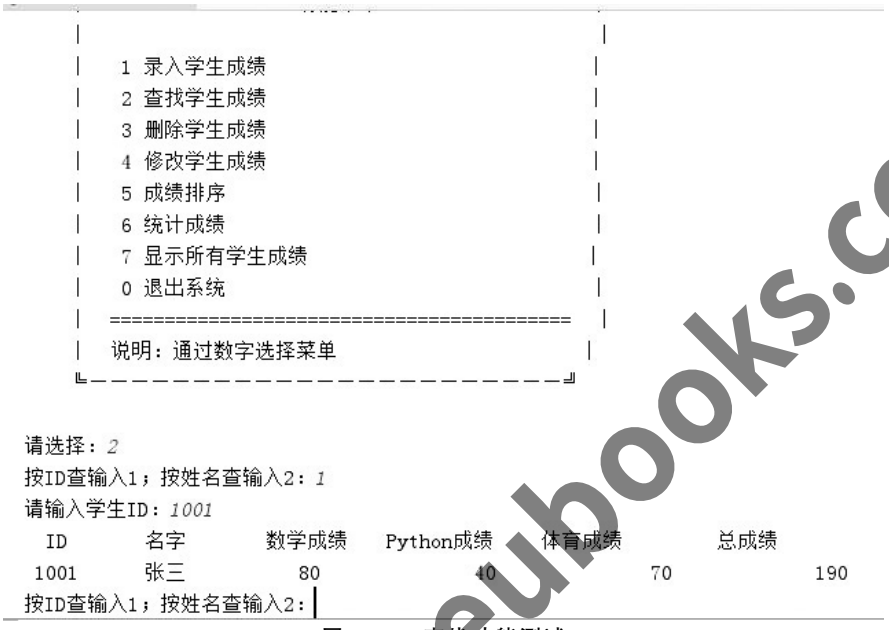

图 1.41 查找功能测试

找到查找功能对应代码段，分析原因问题应在未定义结束或者继续查询的标志，故在代码最后添加一段：

```
inputMark = input("是否继续查询？(y/n):")
if inputMark == "y":
        mark = True
    else:
        mark = False
```

其他功能测试未出现异常。

表 1.7 学生成绩管理系统测试记录表

序号	测试内容	预期结果	实测结果	备注
1	测试录入成绩功能	成绩为 int 型才可输入	输入为字符串型时也可显示	
2	测试查找成绩功能	输入完一次查找信息并返回结果后，应跳出本次查找	未能跳出，继续查找	

1.5 典型工作环节 5：文档编写

在编码、测试、打包功能完成后，需要将可执行程序交付给用户，附上必要的用户手册、

安装手册，这样可以方便用户使用。本项目比较简单，这里只需要编写一个简单的用户手册，具体内容如下：

1. 单击 studentsystem. exe 即可进入如图 1. 42 所示的系统主界面。在该界面中输入数字 0—7 可以选择要使用功能对应的菜单进行不同的操作。

图 1. 42　系统主界面

2. 录入学生信息。输入数字 1，并按下＜Enter＞键，将分别提示输入 ID、名字、数学成绩、Python 成绩和体育成绩，输入正确的信息后，系统会提示是否继续添加；输入 y，可继续添加；输入 n，则将录入学生信息保存其起来，并返回到主界面。如图 1. 43 所示：

图 1. 43　录入学生信息

3. 查找学生信息。在功能菜单上输入功能编号 2，并且按下＜Enter＞键，系将提示用户按 ID 查询还是按姓名查询，如果输入 1，则需输入学生 ID，系统根据 ID 查找该学生信息，如果找到则显示，否则显示“对不起，没有此学生”。最后提示是否继续查找，输入 y，系统将再次提示用户选择查找方式；输入 n，则退出查找学生信息功能。如图 1. 43 所示：

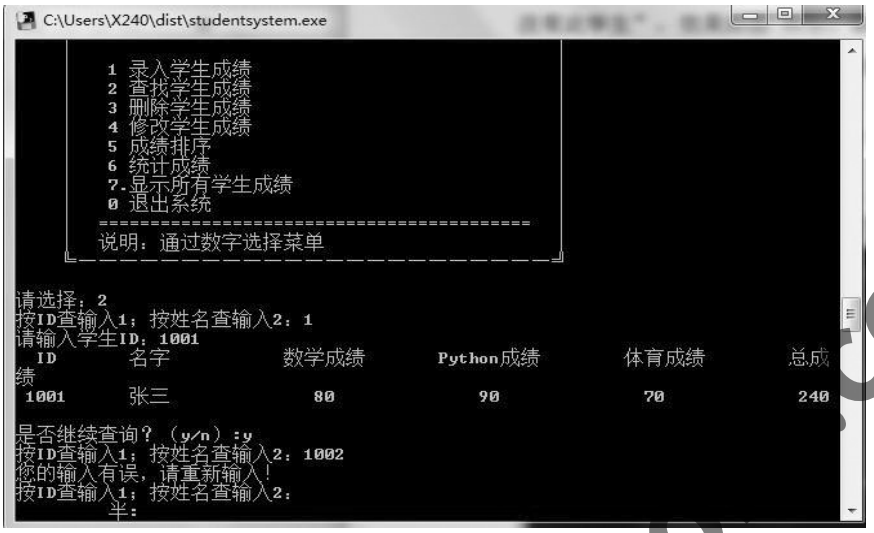

图 1.44　通过学生 ID 查找学生信息

4. 删除学生信息。在功能菜单上输入功能编号 3，并且按下<Enter>键，按照提示输入学生 ID 后，系统会直接删除该学生信；如果 ID 不存在，提示“没有找到 ID 为 * * * * 的学生信息，是否继续删除”，输入 y，系统将会再次提示用户输入要删除的学生编号；输入 n，则退出删除功能。如图 1.45 所示：

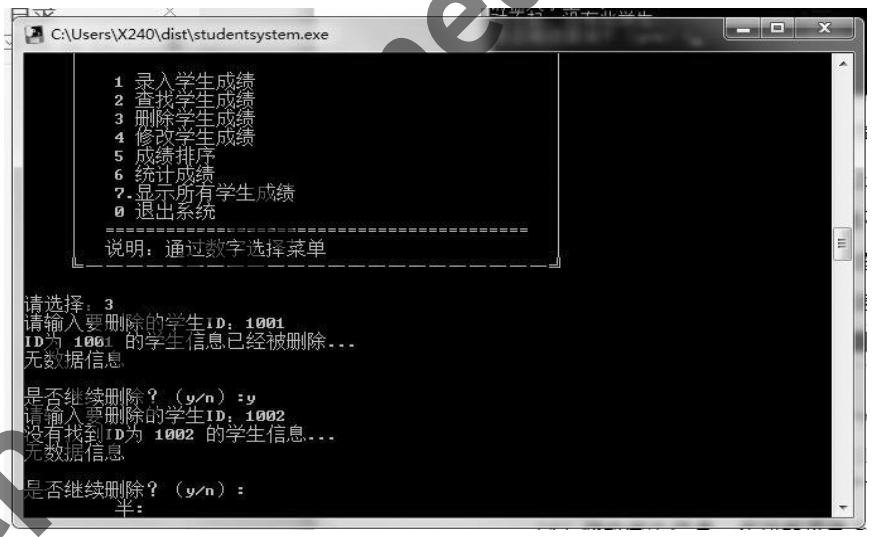

图 1.45　删除学生信息

5. 修改学生信息。在功能菜单上输入功能编号 4，并且按下<Enter>键，系统首先显示全部学生信息列表，再提示输入要修改学生的 ID，输入相应的学生 ID 后，查找相应学生，如果找到，则提示修改相应的信息，否则不修改。最后提示是否继续修改，输入 y，系统将会再次提示用户输入要修改的 ID；输入 n，则退出修改功能。如图 1.46 所示。

6. 成绩排序。在功能菜单上输入功能编号 5，并且按下<Enter>键，系统将先显示不排序的全部学生信息，然后提示选择排序方式，这里输入 1，再选择降序排列“1”，将对学生信息按数学成绩降序排列并显示。最后提示是否继续排序，输入 y，系统将会再次提示用户输入

要排序的方式;输入 n,则退出排序功能。如图 1.47 所示。

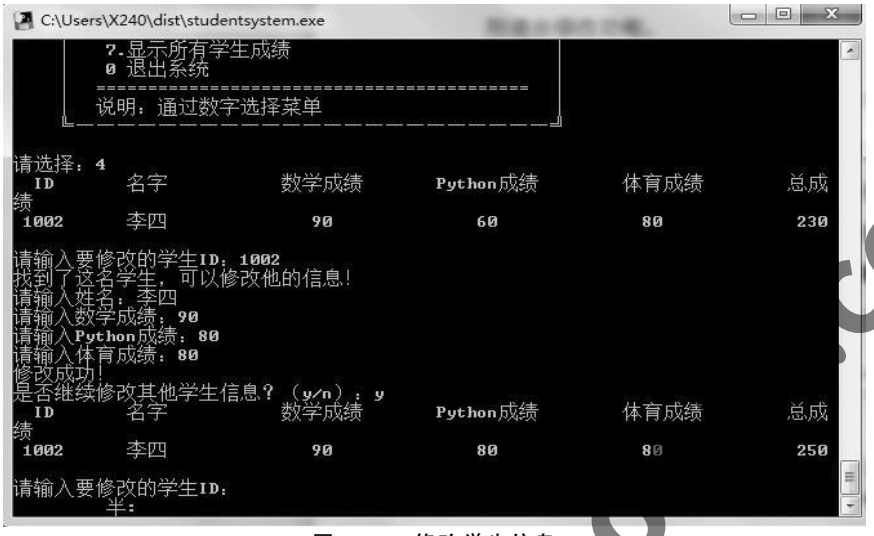

图 1.46　修改学生信息

图 1.47　成绩排序

7.统计学生总人数。在功能菜单上选择 6 ,并按下<Enter>键,系统提示要看什么科目的统计,输入 1 则按照数学成绩统计,此时会显示每个分段的人数及比例。最后提示是否继续统计,输入 y,系统将会再次提示用户输入要统计的科目;输入 n,则退出排序功能。如图 1.48 所示。

8.显示所有学生信息。在功能菜单上选择“7 显示所有学生信息”菜单项,并且按下<Enter>键,系统将获取并显示全部学生信息。

9.退出系统。在功能菜单上输入功能编号 5,并且按下<Enter>键即可退出系统。

图 1.48　学生成绩统计

1.6　典型工作环节 6：程序交付

为了提高可用性，Python 项目编写完成后，可以将其打包成.exe 可执行文件，这样就可以在其他计算机上运行了，这台计算机上面即时不安装 Python 环境也可以运行。实现打包.exe 可执行文件时，需要使用 PyInstaller 模块，该模块是一个第三方模块需要单独安装。PyInstaller 模块支持多种操作系统，如 Windows、Linux、Mac OS X 等，但不支持跨平台，也就是在 Windows 操作系统下打包的.exe 可执行文件，该文件就不能在非 Windows 环境下运行，下面以 Windows 操作系统为例，介绍打包过程。

1. 安装 PyInstaller。在“命令提示符窗口”中输入“pip install pyinstaller”命令进行安装，安装成功界面如图 1.49 所示。

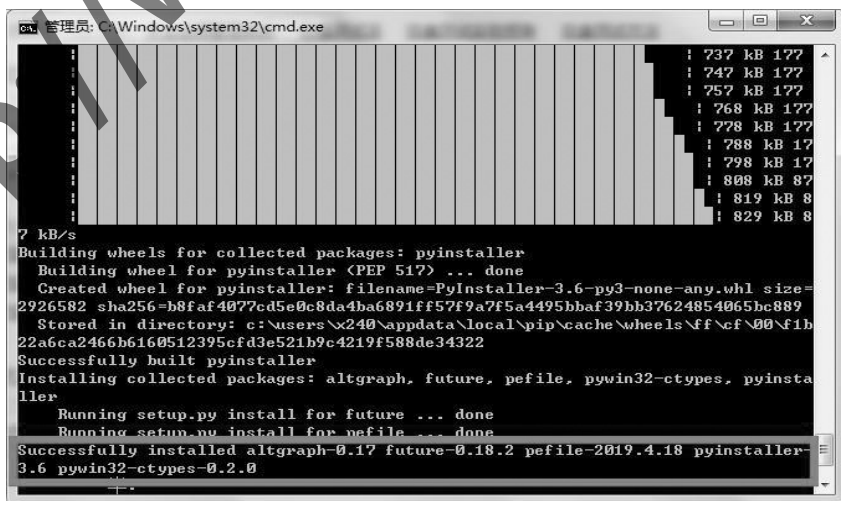

图 1.49　PyInstaller 安装成功界面

2. 打包. py 文件。PyInstaller 模块安装完毕后，在“命令提示符窗口”中输入：pyinstaller-F. py 文件的绝对路径，如图 1.50 所示，打包完成后生成如图 1.51 所示的. exe 可执行文件。

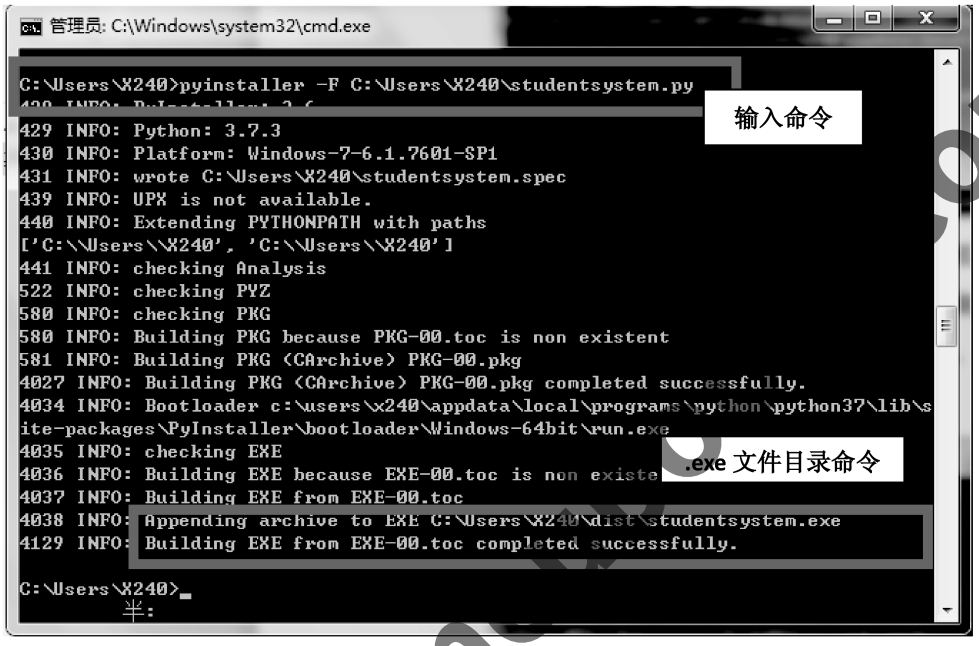

图 1.50　打包. py 文件

图 1.51　打包好的. exe 可执行文件

3. 运行. exe 文件。找到文件保存目录，双击 studentsystem. exe 文件，开始运行本项目，效果如图 1.52 所示。

4. 程序交付。把可执行文件、投票系统用户手册存放在一个文件夹中，就可以分发给用户使用了。

图 1.52 .exe 可执行文件运行效果

1.7 项目总结

1. Python 是一门解释性语言，在运行时需要依赖 Python 解释器。

2. PyCharm 是 JetBrains 公司开发的 Python 集成环境，具备调试、语法高亮、项目管理、智能提示等便于开发和写代码的功能。

3. 掌握 Python 的语法基础，常见数据类型、变量、运算符的应用。

4. Python 中流程控制结构主要包括顺序结构、选择结构、循环结构。顺序结构程序从上往下执行按顺序执行每条语句，中间没有判断或跳转；选择结构使用 if 控制语句实现；循环结构主要是重复执行一组语句，主要包含 while 循环和 for 循环。

5. 无参、无返回值函数是函数应用中最基本的一种，return 后不接任何表达式。

1.8 单项技能训练

一、选择题

1. 关于 Python 语言的注释，以下选项中描述错误的是(　　)。

A. Python 语言的单行注释以 # 开头

B. Python 语言的单行注释以单引号 ' 开头

C. Python 语言的多行注释以 ''' (三个单引号)开头和结尾

D. Python 语言有两种注释方式：单行注释和多行注释

2. 以下选项中不符合 Python 语言变量命名规则的是(　　)。

A. I　　B. 3_1

C. _AI　　D. TempStr

3. 表达式 1001 == 0x3e7 的结果是(　　)。

A. false　　B. False

C. true　　D. True

4. 下面代码的输出结果是 (　　)。

```
x = 12.34 print(type(x))
```

A. <class 'int'>　　B. <class 'float'>

C. <class 'bool'>　　D. <class 'complex'>

5. 以下选项,不是 Python 保留字的选项是(　　)。

A. del　　B. pass

C. not　　D. string

6. 以下关于循环结构的描述,错误的是(　　)。

A. 遍历循环使用 for <循环变量> in <循环结构>语句,其中循环结构不能是文件

B. 使用 range()函数可以指定 for 循环的次数

C. for i in range(5)表示循环 5 次,i 的值是从 0 到 4

D. 用字符串做循环结构的时候,循环的次数是字符串的长度

7. 关于 Python 的分支结构,以下选项中描述错误的是(　　)。

A. 分支结构使用 if 保留字

B. Python 中 if-else 语句用来形成二分支结构

C. Python 中 if-elif-else 语句描述多分支结构

D. 分支结构可以向已经执行过的语句部分跳转

8. 执行以下程序,输入"93python22",输出结果是(　　)。

```
w = input('请输入数字和字母构成的字符串:')
for x in w:
    if '0'<= x <= '9':
        continue
    else:
        w.replace(x,'')
print(w)
```

A. python9322　　B. python

C. 93python22　　D. 9322

9. 执行以下程序,输入 la,输出结果是(　　)。

```
la = 'python'
try:
    s = eval(input('请输入整数:'))
    ls = s * 2
    print(ls)
except:
    print('请输入整数')
```

A. la　　B. 请输入整数

C. pythonpython　　D. python

10. 执行以下程序，输入 qp，输出结果是(　　)。

```
k = 0
while True:
    s = input('请输入 q 退出:')
    if s == 'q':
        k += 1
        continue
    else:
        k += 2
        break
print(k)
```

A. 2　　B. 请输入 q 退出:

C. 3　　D. 1

二、填空题

1. 已知 x=3 和 y=6，执行语句 x，y = y，x 后 x 的值是_____。

2. Python 运算符中用来计算整商的是_____。

3. 表达式 3 | 5 的值为_____。

4. 表达式 4 * 2 的值为_____。

5. 表达式 4 * * 2 的值为_____。

6. 表达式 3<<2 的值为_____。

7. 表达式 65 >> 1 的值为_____。

8. 表达式 int(4 * * 0.5) 的值为_____。

9. 已知 x =2，那么执行语句 x += 7 之后，x 的值为_____。

10. 假设 n 为整数，那么表达式 n&1 == n%2 的值为_____。

11. 在循环语句中，_____语句的作用是提前结束本层循环。

12. 在循环语句中，_____语句的作用是提前进入下一次循环。

13. 对于带有 else 子句的 for 循环和 while 循环，当循环因循环条件不成立而自然结束时_____(会？ 不会?)执行 else 中的代码。

14. 列表、元组、字符串是 Python 的_____(有序？ 无序)序列。

三、编程题

1. 使用 while 循环求 1－100 内所有奇数的平均数。

2. 用 while 循环求除以三余二，除以五余三，除以七余二的数。

3. 如果购买了一张彩票，中奖号码已经公布，是“628735”，请根据输入彩票的号码判断是否中奖，要求用 if…else 语句实现。

1.9　实战演练

实训 1：猜数字游戏

随机生成一个 1—20 之间(包括 1 和 20)的数字作为选定数，玩家每次通过键盘输入一个数字，如果输入的数字和选定数相同，则过关，否则根据提示重新输入，如果用户输入 0，则表示退出游戏。效果如图 1.53 所示。

实训 2：模拟 10086 查询功能

用户通过键盘输入数字，查询相应信息

输入 1，显示当前余额

输入 2，显示当前剩余流量

输入 3，显示当前剩余通话

输入 0，显示自助查询系统

效果如图 1.54 所示。

```
－－－－－－－猜数字游戏－－－－－－－

请输入1－20之间的任意一个数：
12
太大了，请重新输入：
9
太小了，请重新输入：
10
恭喜你，答对了，猜中的数字是： 10

－－－－－－－－游戏结束－－－－－－－－
```

图 1.53　猜数字游戏

```
－－－－－－－10086查询功能－－－－－－－

输入1，查询当前余额
输入2，查询当前剩余流量
输入3，查询当前剩余通话
输入4，查询当前短信剩余量
输入0，退出自助查询系统！
1
当前余额为：100元
2
当前剩余流量为：6G
3
当前剩余通话为：30分钟
4
当前短信剩余量为：55条
0
退出自助查询系统！
```

图 1.54　模拟 10086 查询功能

小提示

与项目 1 的实战演练相配套使用表单详见工作任务单 1。