项目5

水墨钟表(Python模块)

项目描述

本项目主要学习 Python 模块的相关知识，完成本项目后，可以使用 Python 程序设计语言编写一个如图 5-1 所示的“水墨钟表”，要求完成钟表的绘制和钟表时间的同步显示。



图 5-1　水墨钟表效果展示

学习目标

- 了解模块的概念和导入方式。
- 掌握标准模块的使用。
- 掌握自定义模块的使用。
- 了解第三方模块的下载安装。

国家安全是安邦定国的重要基石，维护国家安全是全国各族人民根本利益所在，做好新时代国家安全工作，必须把保密工作放在首位。Python 中的模块就是通过将相关的函数、类和变量封装在一个独立的文件中，可以增强代码的保密性、提高代码的安全性。模块是 Python 程序设计中非常重要的概念，它提供了代码组织、复用和维护的方式，使得代码更加清晰、易于管理和扩展。通过合理使用模块，可以提高代码质量、开发效率和团队协作能力。在 Python 中，常见的模块有内置模块、标准库模块、第三方库模块和自定义模块等。其中，内置模块是 Python 解释器内置的模块，可以直接在代码中使用，而无须导入。标准库模块是 Python 自带的模块，不需要额外安装，直接导入即可使用。第三方库模块是由第三方开发者编写的，需要通过 pip 语句或其他包管理工具进行安装，这些模块提供了各种额外的功能和工具。自定义模块是由开发人员自己编写的模块，用于组织和封装自己的代码，可用于个性化功能开发。

5.1 模块概述

5.1.1 模块的概念

在 Python 中,模块是一种组织代码的方式,用于将相关的函数、类和变量封装在一起,以便重用和维护。每个 Python 文件(以".py"为扩展名)都可以看作一个模块,用于封装定义的函数、类、变量等。在当前".py"文件中导入其他".py"文件,可以使用被导入文件中定义的内容,如类、变量、函数等。模块可以被其他程序或模块导入,并在其他地方重复使用,从而提高代码的重用性。

Python 中的模块大致可分为三类,分别是内置模块、第三方模块和自定义模块,相关介绍如下。

(1) 内置模块是 Python 内置标准库中的一部分,包含了各种功能丰富的工具和库,如 math、random、os 等。这些模块可以直接导入并在程序中使用,无须额外安装。

(2) 第三方模块是社区或其他非官方制作发布的 Python 模块,提供了各种额外的功能和工具,如 numpy、pandas、matplotlib、requests 等。在使用之前需要开发人员自行安装,可以使用 pip 命令或其他包管理工具进行安装。

(3) 自定义模块是开发人员自行编写的 Python 代码并保存在".py"文件中,用于组织和封装功能性代码。一个完整大型的 Python 程序通常被组织为模块和包的集合。自定义模块通常包含函数、类、变量等,可以提高代码的可重用性和结构化水平。

5.1.2 模块的导入

Python 模块的导入方式分为使用 import 导入和使用 from...import...导入两种。

1. 使用 import 导入

使用 import 导入模块的语法格式如下:

```
import 模块 1,模块 2,...
```

import 可以一次导入一个或多个模块,模块之间用逗号分隔。

```
import time
import time,random
```

模块导入之后便可以通过"."(成员运算符)调用该模块中的函数或类,例如使用 random 模块中的 randint()函数,可以写为 random.randint()。

如果在开发过程中需要导入一些名称较长的模块,可以使用 as 为这些模块起别名,后续可直接通过模块的别名使用模块中的内容。例如给导入的 random 模块起别名为 rd,可以写为 import random as rd。

2. 使用 from...import...方式导入

使用 from...import...方式导入模块之后,无须添加前缀,可以像使用当前程序中的函数一样使用模块中的内容。语法格式如下:

```
from 模块名 import 函数/类/变量
```

from...import...也支持一次导入多个函数、类或变量，多个函数、类或变量之间使用逗号隔开。例如使用“from math import sqrt, pi”将 Python 中 math 模块的 sqrt()函数和 pi()函数导入以后，可以直接在代码中使用，而不需要在调用时使用 math.这个前缀。

利用通配符“*”可导入模块中的全部内容，格式如下：

```
from 模块名 import *
```

from...import...也支持为模块或模块中的函数起别名，例如将 time 模块中的 sleep()函数起别名为 sl，示例如下：

```
from time import sleep as sl
```

以上介绍的两种模块的导入方式在使用上大同小异，可根据不同的场景选择合适的导入方式。虽然“from 模块名 import *”方式可以简化代码，但在大多数情况下并不推荐使用，因为它会导入整个模块的内容，可能造成命名冲突。因此，相对而言使用 import 语句导入模块更为安全。

5.1.3 模块的导入特性

1. __all__属性


例 5.1

在 Python 模块的开头通常会定义一个__all__属性，该属性实际上是一个列表，列表中包含的元素决定了在使用“from 模块名 import * ”语句导入模块内容时，通配符(*)所包含的内容。如果__all__属性中只包含模块的部分内容，那么“from 模块名 import * ”语句只会将__all__属性中包含的部分内容导入程序。

__all__属性通常是一个包含字符串元素的列表，只有其中列出的部分内容才会被导入，其他内容不会被导入。这样可以灵活地控制模块中对外可见的内容，提高模块的封装性和可维护性。

以一个简单的四则运算模块为例说明__all__属性的用法。

【例 5.1】 创建一个名为 calculator 的模块，并在主函数中调用该模块。

```
# calculator.py
def add(x, y):
    return x + y
def subtract(x, y):
    return x - y
def multiply(x, y):
    return x * y
def divide(x, y):
    return x / y

__all__ = ["add", "subtract"]
```

在 main.py 程序中使用 calculator 模块，代码如下。

```
# main.py
from calculator import *
print(add(5, 3))          # 可以正常调用,输出 8
print(subtract(10, 4))    # 可以正常调用,输出 6
```

由于 calculator 模块定义了__all__ = ["add", "subtract"],因此在 main.py 中使用“from calculator import *”语句时,add()和 subtract()两个函数会被导入。

继续尝试使用 calculator 模块中 multiply()和 divide()函数,代码如下。

```
# main.py
from calculator import *
print(multiply(2, 6))   # multiply 不在__all__列表中,输出 name 'multiply' is not defined
print(divide(8, 2))     # divide 不在__all__列表中,输出 name 'divide' is not defined
```

因为 multiply()和 divide()函数不在__all__列表中,运行结果报错。

2. __name__属性

在较大型的项目开发中,通常由多名开发人员共同开发一个项目,每名开发人员负责不同的模块。为了确保各模块在整合后能够正常运行,开发人员通常需要编写额外的测试代码,对自己负责的模块进行测试。然而,对整个项目来说,这些测试代码是无用的。为了避免在项目运行时执行这些测试代码,Python 中引入了__name__属性。

__name__属性通常与 if 条件语句一起使用。如果当前模块是启动模块,则__name__的值为“main”;如果该模块被其他程序导入,则__name__的值为文件名。以 calculator 模块为例来演示__name__属性的用法。


例 5.2

【例 5.2】 在 calculator 模块中增加__name__属性。

```
# calculator.py
def add(x, y):
    return x + y
def subtract(x, y):
    return x - y
def multiply(x, y):
    return x * y
def divide(x, y):
    return x / y
#增加__name__属性代码
if __name__ == "__main__":
    print("Running test cases for calculator module...")
    print("5 + 3 =", add(5, 3))
    print("10 - 4 =", subtract(10, 4))
    print("2 * 6 =", multiply(2, 6))
    print("8 / 2 =", divide(8, 2))
else:
    print(__name__)
```

calculator.py 代码运行结果如图 5-2 所示。

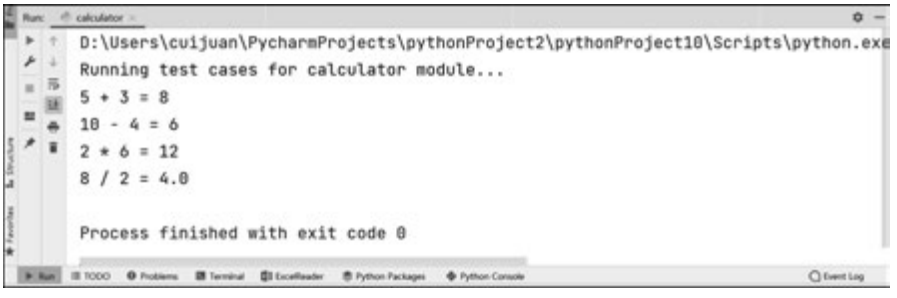


图 5-2 calculator.py 代码运行结果

```
# main.py
from calculator import *
print(add(5, 3))          # 可以正常调用
print(subtract(10, 4))    # 可以正常调用
```

主程序 main.py 代码不变,再次运行主程序 main.py,运行结果如图 5-3 所示。



图 5-3　main.py 运行结果

由以上结果可知,单独运行模块 calculator 时,测试代码会被执行。但是当 calculator 模块被导入 main.py 中执行时,则不会执行测试代码,因为__name__不等于"__main__"。这样的做法通常可以确保模块在被导入时只执行必要的初始化工作,并且可以在需要时进行测试。

5.2　标准模块

标准模块是 Python 内置的模块,它们包含在 Python 解释器中,无须额外安装即可直接使用。这些标准模块提供了丰富的功能,涵盖了各种领域,例如文件操作、网络通信、日期时间处理、数据处理等。接下来介绍几个常用的标准模块。

5.2.1　os 模块

os 模块是 Python 标准库中的一个重要模块,用于与操作系统进行交互。它提供了许多函数来处理文件、目录、路径等操作系统相关的功能。以下是 os 模块中一些常用的函数及其说明,如表 5-1 所示。

表 5-1　os 模块中常用的函数及其说明

函　数	说　明
getcwd( )	获取当前工作路径,即当前 Python 脚本所在的路径
chdir(path)	该函数用于改变当前工作目录为指定的路径 path
remove(path)	该函数用于删除指定路径 path 的文件
_exit(status)	该函数是一个底层的退出函数,用于直接退出 Python 解释器,其中的 status 参数表示退出状态码
path.abspath(path)	该函数接收一个路径 path 参数,并返回该路径的绝对路径。如果输入的路径是相对路径,该函数会返回其对应的绝对路径
path.split(path)	该函数将路径 path 分割为目录部分和文件名部分,并以元组的形式返回。如果输入的路径只是一个文件名,则目录部分为空字符串

1. 使用 os.path.abspath() 函数获取绝对路径

【例 5.3】 定义一个相对路径 file.txt,然后使用 os.path.abspath()函数获取该文件的绝对路径。

```
import os
# 定义一个相对路径
relative_path = "file.txt"
# 使用 os.path.abspath() 函数获取绝对路径
absolute_path = os.path.abspath(relative_path)
# 输出结果
print("Relative Path:", relative_path)
print("Absolute Path:", absolute_path)
```

运行结果如图 5-4 所示。

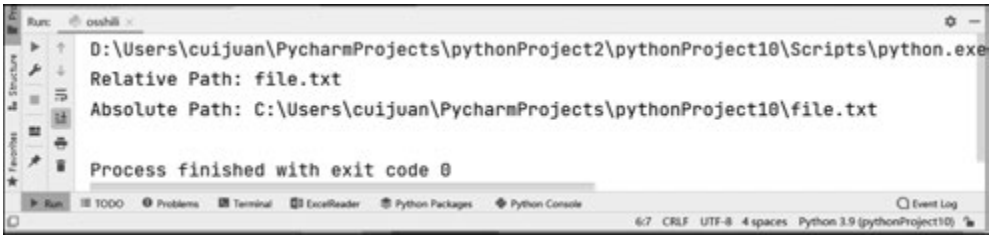


图 5-4 获取绝对路径运行结果

例 5.3

例 5.4

2. 使用 os.path.split(path)函数分割路径

【例 5.4】 定义一个文件路径 /path/to/directory/file.txt,然后使用 os.path.split()函数将该路径分割为目录部分 /path/to/directory 和文件名部分 file.txt。最后,输出分割后的结果。

```
import os
# 定义一个文件路径
file_path = "/path/to/directory/file.txt"
# 使用 os.path.split( ) 函数分割路径
directory, file_name = os.path.split(file_path)
# 输出结果
print("Original Path:", file_path)
print("Directory:", directory)
print("File Name:", file_name)
```

运行结果如图 5-5 所示。



图 5-5 分割路径运行结果

3. 使用 os.remove(path) 函数移除文件

【例 5.5】 创建一个测试文件 test_file.txt，然后使用 os.remove(path) 函数移除该文件。

```
import os
# 创建一个测试文件
test_file_path = "test_file.txt"
with open(test_file_path, "w") as file:
    file.write("This is a test file.")
# 移除文件
os.remove(test_file_path)
print("Removed file:", test_file_path)
```

运行结果如图 5-6 所示。



图 5-6　移除文件运行结果

例 5.5

5.2.2　random 模块

例 5.6

random 模块是 Python 中用于生成随机数的标准库模块，提供了多种函数来生成伪随机数。以下是 random 模块中常用的函数及其说明，如表 5-2 所示。

表 5-2　random 模块中常用的函数及其说明

函　　数	说　　明
random()	生成一个 0～1 的随机浮点数
randint(a, b)	生成一个介于 a 和 b 之间的随机整数，包括 a 和 b
shuffle(seq)	随机打乱序列 seq 中元素的顺序
choice(seq)	从序列 seq 中随机选择一个元素并返回
sample(population, k)	从总体 population 中随机选择 k 个不重复的元素，并返回一个列表

除了上述函数外，random 模块还提供了其他一些生成随机数的函数，如生成高斯分布的随机数、生成随机字符串等。

【例 5.6】 使用 random 模块生成随机数。

```
import random
# 生成一个随机浮点数
rand_float = random.random()
print("Random float:", rand_float)
# 生成一个随机整数
rand_int = random.randint(1, 10)
print("Random integer:", rand_int)
# 从列表中随机选择一个元素
```

```
my_list = ['apple', 'banana', 'cherry', 'date']
rand_choice = random.choice(my_list)
print("Random choice:", rand_choice)
# 随机打乱列表元素的顺序
random.shuffle(my_list)
print("Shuffled list:", my_list)
# 从列表中随机选择不重复的元素
rand_sample = random.sample(my_list, 2)
print("Random sample:", rand_sample)
```

运行结果如图 5-7 所示。

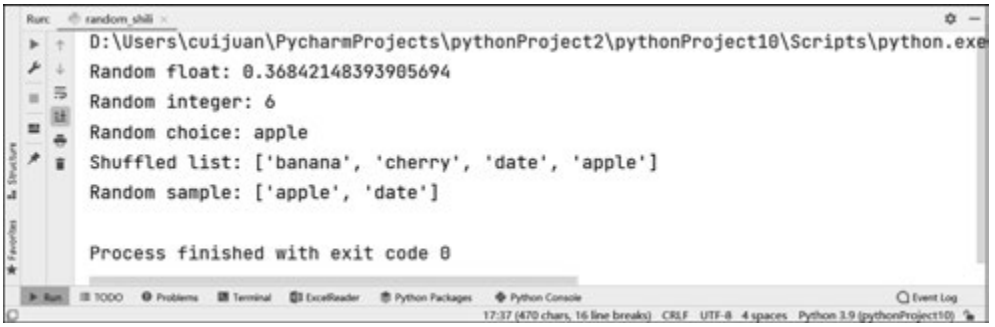


图 5-7　random 模块运行结果

例 5.7

5.2.3　datetime 模块

datetime 模块是 Python 中用于处理日期和时间的标准库模块,提供了多个类和函数来操作日期、时间以及它们的组合。表 5-3 所示为 datetime 模块中常用的函数及其说明。

表 5-3　datetime 模块中常用的函数及其说明

函　　数	说　　明
datetime.datetime	该类表示日期和时间的组合,并提供了多种方法来操作日期时间数据
datetime.date	该类表示日期,包括年、月、日
datetime.time	该类表示时间,包括小时、分钟、秒、毫秒
datetime.timedelta	该类表示时间间隔,可以用来进行日期时间的加减运算
datetime.now()	该函数返回当前日期时间
datetime.strptime(date_string,format)	该函数将指定格式的字符串转换为日期时间对象
datetime.strftime(format)	该函数将日期时间对象格式化为指定格式的字符串

【例 5.7】 使用 datetime 模块操作日期和时间。

```
import datetime
# 获取当前日期时间
current_datetime = datetime.datetime.now()
print("Current datetime:", current_datetime)
# 创建一个特定日期时间
specific_datetime = datetime.datetime(2024, 12, 25, 10, 30)
print("Specific datetime:", specific_datetime)
```

```
# 计算日期时间差
time_diff = specific_datetime - current_datetime
print("Time difference:", time_diff)
# 将字符串转换为日期时间对象
date_str = "2024-06-30"
date_obj = datetime.datetime.strptime(date_str, "%Y-%m-%d")
print("Date object:", date_obj)
# 将日期时间对象格式化为字符串
formatted_date = date_obj.strftime("%A, %d %B %Y")
print("Formatted date:", formatted_date)
```

运行结果如图 5-8 所示。



图 5-8　datetime 模块运行结果

5.2.4　turtle 模块

turtle 模块是 Python 标准库中的一个绘图库，专门用于绘制简单的图形和动画。turtle 模块提供了一个类似于乌龟绘图的程序设计环境，让用户可以通过控制一个虚拟的乌龟(turtle)来绘制图形，主要包括以下功能。

(1) 绘制基本图形：turtle 模块可以绘制线、圆、多边形等基本图形，通过调用相应的方法和函数实现。

(2) 控制乌龟移动：用户可以通过控制乌龟的移动、转向来绘制复杂的图形，例如绘制螺旋线、迷宫等。

(3) 交互式绘图：turtle 模块提供了一个交互式的绘图窗口，用户可以实时看到图形的绘制过程，并且可以根据需要进行调整。

(4) 动画效果：用户可以利用 turtle 模块创建简单的动画效果，例如移动的小球、旋转的图形等。

turtle 模块可以帮助用户理解程序设计概念、培养逻辑思维，并且可以创造出简单有趣的图形和动画效果。表 5-4 所示为 turtle 模块中常用的函数及其说明。

表 5-4　turtle 模块中常用的函数及其说明

函　数	说　明
turtle.Turtle()	创建一个乌龟对象，用于绘制图形
forward()	将乌龟向前移动指定距离
backward()	将乌龟向后移动指定距离

续表

函　数	说　明
left()	将乌龟向左旋转指定角度
right()	将乌龟向右旋转指定角度
penup()	抬起画笔,乌龟移动时不绘制图形
pendown()	落下画笔,乌龟移动时绘制图形
pencolor()	设置画笔颜色
pensize()	设置 turtle 画笔的宽度
fillcolor()	设置填充颜色
bgcolor()	设置绘制窗口的背景颜色
speed()	设置乌龟移动的速度
shape()	用于指定 turtle 对象在屏幕上显示的形状,可以是预定义的形状或自定义的形状。当调用 shape("turtle")函数时,turtle 对象将会显示为乌龟形状
goto(x, y)	移动 turtle 到指定的坐标位置(x, y)

【例 5.8】 使用 turtle 模块和 random 模块实现绘制彩色的小乌龟赛跑游戏。


例 5.8

```
import turtle
import random
# 设置画布背景颜色
turtle.bgcolor("lightblue")
# 创建乌龟对象
turtles = []
colors = ["red", "blue", "green", "orange", "purple"]
# 创建五只彩色乌龟
for i in range(5):
    turtles.append(turtle.Turtle())
    turtles[i].shape("turtle")
    turtles[i].color(colors[i])
    turtles[i].penup()
    turtles[i].goto(-200, 80 - i * 40)
# 乌龟赛跑
winner_announced = False
while not winner_announced:
    for turtle in turtles:
        turtle.forward(random.randint(1, 5))
        if turtle.xcor() >= 200:
            winner_color = turtle.color()[0]
            print(f"The {winner_color} turtle wins!")
            winner_announced = True
            break
```

小乌龟赛跑游戏图形界面和运行结果如图 5-9 和图 5-10 所示。

可以在图形界面中看到乌龟在画布上进行赛跑,并且当其中一只乌龟到达终点时,会输出赢得比赛的乌龟的颜色。这个程序展示了如何使用 turtle 模块创建简单的动画效果和逻辑控制。



图 5-9　小乌龟赛跑游戏图形界面



图 5-10　小乌龟赛跑游戏运行结果

例 5.9

5.3　自定义模块

在 Python 中，每个 Python 文件都可以作为一个模块，文件的名字就是模块的名字。现有一个文件 myModule.py，它定义了 myMin()和 myMax()两个函数，如下所示。

【例 5.9】 编程实现 myMin()和 myMax()两个函数，分别求最小值和最大值。

```
def myMin(a,b):
    c = a
    if a > b:
        c = b
    return c
def myMax(a,b):
    c = a
    if a < b:
        c = b
    return c
```

把 myModule.py 保存到 E:\Users\Administrator\PycharmProjects\下，此时，如果在相同的路径下新建 main.py 文件，那么就可以在 main.py 中引入 myModule 模块，使用其中的 myMin()和 myMax()函数。具体 main.py 示例如下。

```
import myModule
min_val = myModule.myMin(5,8)
```

```
max_val = myModule.myMax(5,8)
print("大数是: ",max_val)
print("小数是: ",min_val)
```

运行结果如图 5-11 所示。

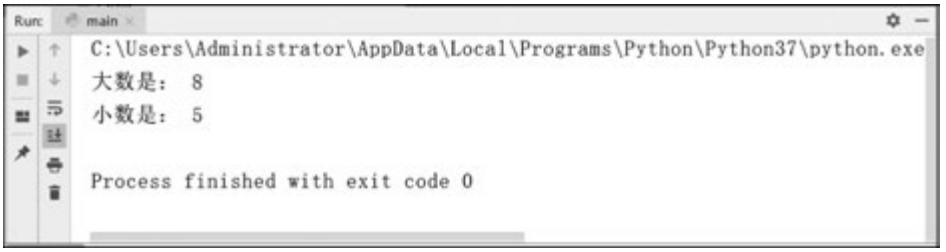


图 5-11 自定义模块运行结果

由此可见,程序是在 main.py 中通过 import myModule 语句引入了 myModule 模块,因此在 main.py 程序中可以使用 myModule.py 中定义的 myMin()和 myMax()函数。

5.4 第三方模块

第三方模块是指不在 Python 标准库中内置的模块,而是由其他开发者或组织开发并发布。这些模块通常提供了额外的功能和工具,可以帮助开发者更快速、高效地完成任务。

5.4.1 下载安装第三方模块

在 Python 中,可以使用 pip 工具来安装第三方模块。例如,如果想要安装一个名为 requests 的第三方模块,可以在命令行中运行"pip install requests"命令下载,并且在 Python 项目中使用它。在 PyCharm 中安装第三方模块的方法和步骤如图 5-12～图 5-14 所示。

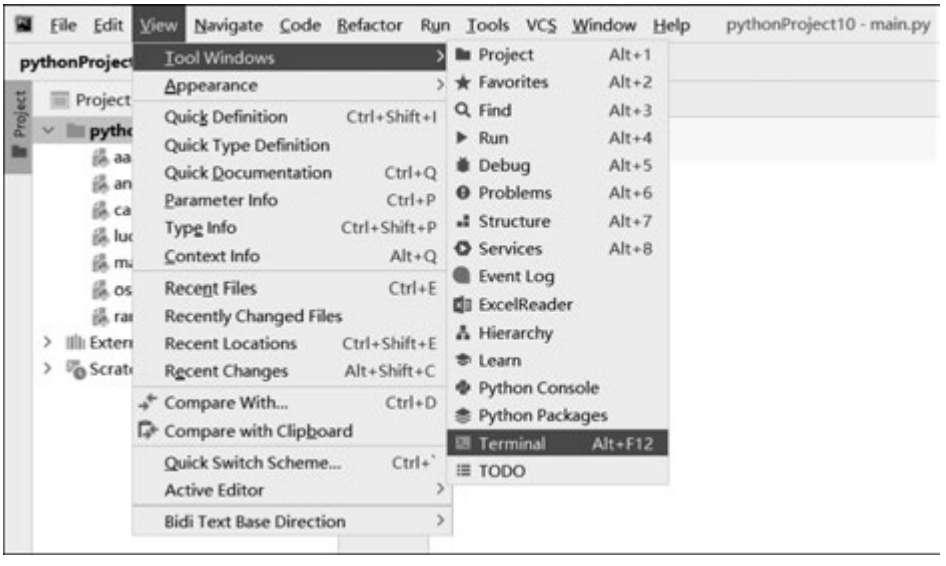


图 5-12 打开菜单栏中的 Terminal 窗口



图 5-13　在 Terminal 窗口中输入安装命令

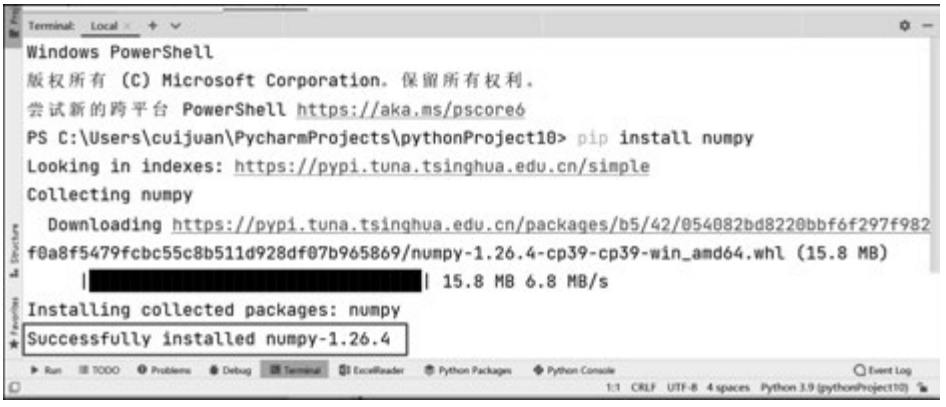


图 5-14　安装成功

例 5.10

5.4.2　常见的第三方模块

在 Python 中,第三方模块的数量是庞大的。表 5-5 列举了常见的第三方模块及其说明。

表 5-5　常见的第三方模块及其说明

模　块	说　明
requests	用于发送 HTTP 请求和处理响应的模块
matplotlib	用于绘制图表和图形的数据可视化库
numpy	用于科学计算和数据处理的数值计算库
pandas	用于数据分析和处理的数据结构和工具集
Django	用于构建 Web 应用程序的 Web 框架
scikit-learn	用于机器学习和数据挖掘的库
Beautiful Soup	用于解析 HTML 和 XML 文档的库,方便从网页中提取信息和进行数据抓取
pygame	用于制作 2D 游戏的 Python 库,提供了丰富的功能和工具,如图形绘制、音频处理、事件处理等。它是一个轻量级的库,适合初学者和中级开发者使用

这些第三方模块为 Python 开发者提供了丰富的工具和功能,涵盖了多个领域,包括网络编程、数据处理、数据可视化、Web 开发、机器学习等。选择合适的第三方模块可以帮助开发者更高效地开发应用程序和解决问题。接下来举例说明第三方模块的使用方法。

【例 5.10】　numpy 模块的使用案例。

```
import numpy as np
# 创建一个一维数组
```

```
arr1 = np.array([1, 2, 3, 4, 5])
print("一维数组:")
print(arr1)
# 创建一个二维数组
arr2 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print("\n二维数组:")
print(arr2)
#访问数组元素
print("\n访问数组元素:")
print("第一个数组的第三个元素:", arr1[2])
print("第二个数组的第一行第二列元素:", arr2[0, 1])
# 数学运算
print("\n数组加法:")
arr3 = np.array([5, 6, 7, 8, 9])
result = arr1 + arr3
print(result)
# 基本统计操作
print("\n数组统计:")
print("第一个数组的平均值:", np.mean(arr1))
print("第二个数组的最大值:", np.max(arr2))
```

运行结果如图 5-15 所示。

例 5.11

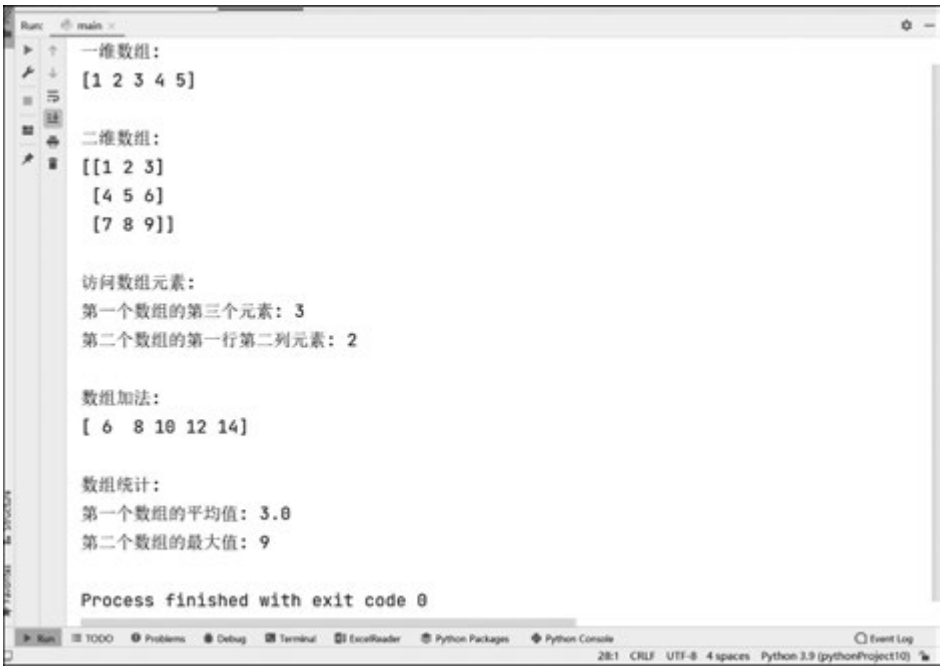


图 5-15　numpy 模块案例的运行结果

【例 5.11】 matplotlib 模块实现折线图案例。

```
import matplotlib.pyplot as plt
# 创建示例数据
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]
# 创建折线图
```

```
plt.plot(x, y, marker = 'o', color = 'b', linestyle = '-', linewidth = 2)
# 添加标题和标签
plt.title('Simple Line Plot')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
# 显示图表
plt.show()
```

运行结果如图 5-16 所示。

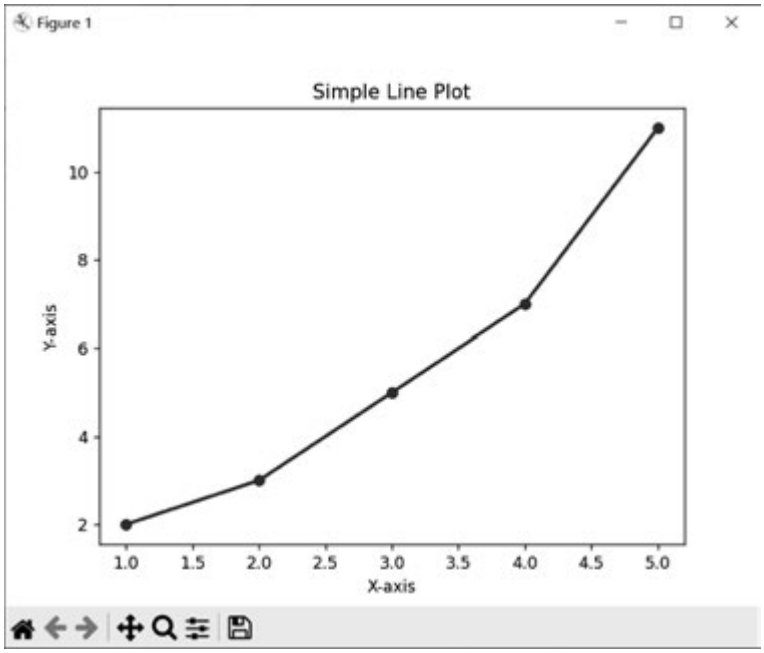


图 5-16 matplotlib 模块案例的运行结果


例 5.12

【例 5.12】 pygame 模块演示彩色小球弹跳游戏。

```
import pygame, sys, random
pygame.init()
# 游戏设置
W, H = 800, 600
screen = pygame.display.set_mode((W, H))
pygame.display.set_caption("彩色小球弹跳")
font = pygame.font.SysFont(["SimHei", "WenQuanYi Micro Hei", "Heiti TC"], 24)
# 小球属性(固定大小,随机颜色)
BALL_SIZE = 20 # 所有小球大小一致
balls = []
# 初始化 5 个小球
for _ in range(5):
    x, y = random.randint(BALL_SIZE, W - BALL_SIZE), random.randint(BALL_SIZE, H//2)
    color = (random.randint(50, 255), random.randint(50, 255), random.randint(50, 255))
    balls.append([x, y, random.uniform(-5, 5), random.uniform(1, 3), 0.2, 0.8, color])
clock, running = pygame.time.Clock(), True
while running:
    screen.fill((240, 240, 250))       # 浅灰背景
```

```
        # 事件处理
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False
            # 单击添加新球(同样大小,不同颜色)
            elif event.type == pygame.MOUSEBUTTONDOWN:
                x, y = pygame.mouse.get_pos()
                color = (random.randint(50, 255), random.randint(50, 255), random.randint(50, 255))
                balls.append([x, y, random.uniform(-5, 5), random.uniform(1, 3), 0.2, 0.8, color])
        # 更新并绘制所有小球
        for b in balls:
            x, y, vx, vy, g, e, color = b
            # 应用重力和更新位置
            vy += g
            x += vx
            y += vy
            # 边界碰撞检测
            if x <= BALL_SIZE or x >= W - BALL_SIZE:
                vx = -vx * e
                x = BALL_SIZE if x <= BALL_SIZE else W - BALL_SIZE
            if y <= BALL_SIZE or y >= H - BALL_SIZE:
                vy = -vy * e
                y = BALL_SIZE if y <= BALL_SIZE else H - BALL_SIZE
                if abs(vy) < 1 and y >= H - BALL_SIZE - 1:
                    vy, g = 0, 0
            # 更新小球位置
            b[:4] = [x, y, vx, vy]
            pygame.draw.circle(screen, color, (int(x), int(y)), BALL_SIZE)
        # 显示信息
        screen.blit(font.render(f"小球数量:{len(balls)} 单击添加", True, (50, 50, 50)), (10, 10))
        pygame.display.flip()
        clock.tick(60)
    pygame.quit()
    sys.exit()
```

运行结果如图 5-17 所示。

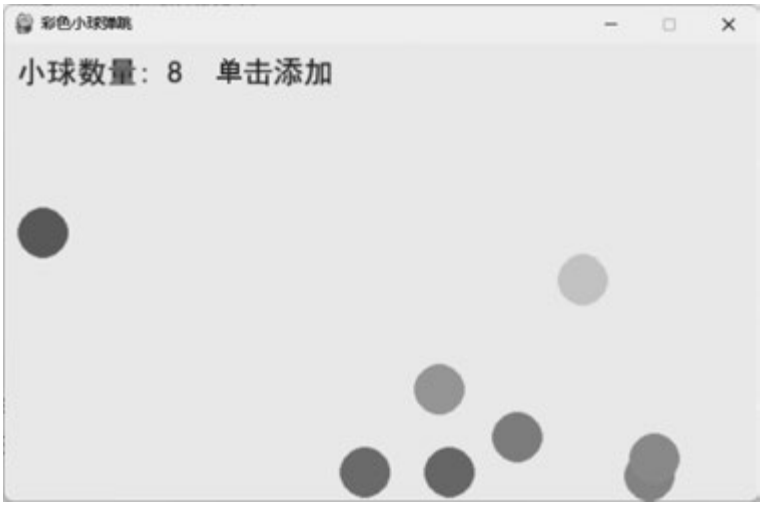


图 5-17 pygame 模块案例的运行结果

项目实现

1. 项目背景

时钟在日常生活中扮演着重要的角色，可以帮助人们更好地管理时间，有序地安排生活和工作，提高生活质量和工作效率，是一个不可或缺的日常工具。使用 Python 程序实现水墨钟表，主要是利用 turtle 图形库创建模拟时钟，使用 datetime 模块来获取当前的日期和时间，具体要实现的功能描述如下。

任务一：设置钟表的水墨背景。

任务二：设置钟表表盘。表盘的样式由 60 个刻度围成圆形，显示时间、日期和星期跟随本地时间实时变化。

任务三：设置钟表表盘中的指针。时针、分针和秒针一端被固定在表盘中心，另一端按照顺时针的方向围着表盘中心位置旋转，秒针每旋转一周，分针移动一个刻度，分针每旋转一周，时针移动 5 个刻度。

2. 项目解析

任务一考查学生导入素材，调用模块设置背景的技能，较为简单。

任务二考查学生使用库函数和模块在坐标图上进行绘制图案的技能，需要学生综合运用 turtle 模块进行画笔的控制，实现过程中要注意坐标位置。

任务三绘制时钟表盘中的指针，并且要求指针要按照目前的日期显示并随着时间的变动变化。此任务是本项目的难点，考查学生综合运用 turtle 模块、datetime 模块进行时钟指针的显示设置，各个指针之间的位置关系，以及时间刷新设置等。

水墨钟表项目实现

3. 项目实现过程

本项目使用 turtle 和 datetime 模块绘制如图 5-18 所示的水墨钟表，使钟表的时间、星期和日期随本地时间实时变化。



图 5-18 水墨钟表项目实现效果

钟表的模拟程序可以分为外观绘制、时间与日期处理两个大任务。其中，外观绘制任务可以细分为绘制表盘刻度，绘制指针，绘制日期显示文本和水墨图案背景设置 4 个子任务；时间与日期处理任务可以细分为处理日期和处理周日期 2 个子任务。

开发程序时，需要先导入程序开发所用到的库或模块，这里需要导入 turtle 和 datetime 模块，具体代码如下。

```
import turtle
from turtle import *
from datetime import *
```

步骤 1：水墨图案背景设置。

水墨图案背景设置使用 turtle 模块中的背景设置方法，示例如下。

```
def draw_clock_background():
    # 设置背景图片
```

```
    screen = turtle.Screen()
    screen.bgpic("bg.png")
```

步骤 2：设置画笔。

skip()函数的功能是将画笔移动指定的距离，它包含了抬起画笔、移动画笔和落下画笔3个固定的步骤。绘制刻度时，无论是绘制线段还是圆点，都会涉及画笔的移动，skip()函数可以避免多个位置出现重复的代码。skip()函数的定义示例如下。

```
def skip(step):
    #设置画笔的起落和绘图
    penup()
    forward(step)
    pendown()
```

步骤 3：绘制钟表外框刻度。

setup_clock()函数的功能是绘制钟表的刻度。在绘制钟表刻度时，以钟表的中心为起始点，从中心位置开始移动画笔到指定的位置落笔，绘制完线段或圆点之后再次回到中心点，改变画笔的角度，移动画笔到指定的位置落笔绘制下一个刻度，循环实现钟表刻度的绘制。setup_clock()函数的定义示例如下。

```
def setup_clock(radius):
    #建立表的外框
    reset()
    pensize(7)
    for i in range(60):
        skip(radius)
        if i % 5 == 0:
            forward(10)
            skip(-radius-10)
        else:
            dot(5)
            skip(-radius)
        right(6)
```

以上函数中使用 for 语句控制循环执行 60 次，绘制 60 个刻度，通过 if-else 语句区分画短线与圆点两种情况：若能整除 5 则画短线，否则画圆点。

步骤 4：绘制指针。

make_hand()函数的功能是注册 turtle 形状。钟表中的时针、分针、秒针 3 个指针长度不同，但是它们的形状相似。为了防止冗余，使用 make_hand()函数实现直线的绘制功能。示例如下。

```
def make_hand(name, length):
    #注册 turtle 形状,建立表针 turtle
    reset()
    skip(-length*0.1)
    begin_poly()
```

```
    forward(length * 1.1)
    end_poly()
    handForm = get_poly()
    register_shape(name, handForm)
```

Init()函数的功能是初始化程序,包括创建和初始化钟表指针、日期和星期的 turtle 对象。Init()函数的代码示例如下。

```
def Init():
    global secHand, minHand, hurHand, printer
    mode("logo")                    #重置 turtle 指向北
    #创建三个表针 turtle 并初始化
    #第二个参数为长度
    make_hand("secHand", 125)     #秒针
    make_hand("minHand", 130)     #分针
    make_hand("hurHand", 90)      #时针
    secHand = Turtle()
    secHand.shape("secHand")
    minHand = Turtle()
    minHand.shape("minHand")
    hurHand = Turtle()
    hurHand.shape("hurHand")
    for hand in secHand, minHand, hurHand:
        hand.shapesize(1, 1, 3)    #调整三个指针的粗细
        hand.speed(0)
    #创建输出文字 turtle
    printer = Turtle()
    printer.hideturtle()
    printer.penup()
```

步骤 5:绘制星期显示文本。

Week()函数的功能是处理星期,以“星期 *”的格式显示。

```
def Week(t):
    week = ["星期一", "星期二", "星期三",
            "星期四", "星期五", "星期六", "星期日"]
    return week[t.weekday()]
```

步骤 6:绘制日期显示文本。

Date()函数的功能是处理日期,以“年月日”的格式显示。

```
def Date(t):
    y = t.year
    m = t.month
    d = t.day
    return "%s %d %d" % (y, m, d)
```

步骤 7:实时显示时间、日期和星期。

Tick()函数的功能是实时获取本地当前的时间、日期和星期,并且进一步获取当前指针

的指向角度。函数的实现代码如下。

```
def Tick():
    #绘制表针的动态显示
    #当前时间
    t = datetime.today()
    #获取秒、分钟和小时,正确显示指针当前时间
    second = t.second + t.microsecond * 0.000001
    minute = t.minute + second/60.0
    hour = t.hour + minute/60.0
    #将秒针(secHand)、分针(minHand)、时针(hurHand)的方向设置为当前时间各指针应指向的角度
    secHand.setheading(6 * second)
    minHand.setheading(6 * minute)
    hurHand.setheading(30 * hour)
    #接入 tracer 函数以控制刷新速度
    tracer(False)
    printer.forward(65)   #画笔(海龟)向当前方向移动指定的距离
    #在当前位置写入星期文本
    printer.write(Week(t), align = "center",
            font = ("Courier", 14, "bold"))
    printer.back(130)     #画笔(海龟)向当前方向后退指定的距离
    #在当前位置写入日期文本
    printer.write(Date(t), align = "center",
             font = ("Courier", 14, "bold"))
    printer.home()        #将画笔移动到坐标原点(0, 0)的位置,同时将画笔的方向设置为默认方向
    tracer(True)          #开启定时器,每 100ms 调用函数 Tick()
    ontimer(Tick, 100)    #100ms 后继续调用 tick
```

调用 home()函数将位置和方向恢复到初始状态,位置的初始坐标为(0,0),初始方向有两种情况:若为 standard 模式,则初始方向为 right,表示朝向东;若为 logo 模式,则初始方向为 up,表示朝向北。

步骤 8:主函数设置。

main()是程序的主函数,负责组织程序的主要逻辑。水墨钟表的逻辑为:初始化—绘制水墨背景—绘制钟表框—绘制动态指针—启动事件循环。根据该逻辑定义函数,具体代码实现如下所示。

```
def main():
    draw_clock_background()
    screen = turtle.Screen()
    screen.tracer(False)
    Init()
    setup_clock(160)
    screen.tracer(True)
    Tick()
    mainloop()
```

步骤 9:功能演示。

调用主函数,执行程序,代码如下。

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

程序运行结果如图 5-19 所示。



图 5-19　水墨钟表运行结果

项目小结

本项目主要学习了 Python 中的标准模块、自定义模块和第三方模块，并且使用 datetime 模块和 turtle 模块实现了实时显示时间的水墨钟表。通过本案例，使学生掌握模块的导入和使用，体会模块化程序设计的逻辑，实现了调用不同的模块功能函数实现整体项目功能，巩固了模块的基本知识。

拓展练习

1. 编写一个程序，导入 math 模块并计算圆的面积。用户输入圆的半径，程序输出圆的面积。提示：圆的面积公式为 πr^2，其中 r 为半径，π 可以使用 math.pi 获取。

2. 编写一个程序，导入 random 模块并生成一个随机数猜谜游戏。程序随机生成一个 1～100 范围内的整数，用户通过输入猜测的数字来猜测这个随机数，直到猜中为止。程序会提示用户猜的数字是太大还是太小。

课后习题

一、选择题

1. 模块是(　　)。

A. 一个 Python 文件　　B. 一个函数

C. 一个变量　　D. 一个数据类型

2. 在 Python 中，如何导入一个模块？(　　)

A. import module_name　　B. include module_name

C. use module_name　　D. add module_name

3. Python 内置的用于处理日期和时间的模块是(　　)。

A. time　　B. datetime　　C. calendar　　D. date

4. 使用 math 模块时,如何计算一个数的平方根?(　　)

A. math.square_root(num)　　B. math.sqrt(num)

C. math.power(num, 0.5)　　D. math.root(num)

5. 如何生成一个 1～100 范围内的随机整数?(　　)

A. random.randint(1, 100)　　B. random.random(1, 100)

C. random.int(1, 100)　　D. random.range(1, 100)

6. 如何删除一个文件或目录?(　　)

A. remove(path)　　B. delete(path)

C. erase(path)　　D. cleanup(path)

7. Python 中的 os 模块主要用于(　　)。

A. 处理日期和时间　　B. 生成随机数

C. 执行操作系统相关的任务　　D. 处理数学运算

8. 如何获取当前工作目录的路径?(　　)

A. os.get_path()　　B. os.current_dir()

C. os.getcwd()　　D. os.get_current_path()

9. 如何在 Python 中导入自定义模块 my_module?(　　)

A. import my_module　　B. include my_module

C. from my_module import *　　D. use my_module

10. 如何导入 random 模块以生成一个随机浮点数?(　　)

A. random.random_float()　　B. random.randint()

C. random.uniform()　　D. random.random()

二、判断题

1. Python 的 math 模块主要用于处理日期和时间信息。(　　)

2. 在 Python 中,可以使用 import 语句导入其他 Python 文件中定义的函数。(　　)

3. os.path.exists()可以用来检查文件或目录是否存在。(　　)

4. Python 的 os 模块主要用于处理数学运算。(　　)

5. datetime.now()可以返回当前的日期和时间。(　　)

6. 使用 random.choice()可以从列表中随机选择一个元素。(　　)

7. 使用 import 语句时,可以给导入的模块设置别名。(　　)

8. os.getcwd() 函数可以返回当前工作目录的绝对路径。(　　)

9. 使用 random.randrange(1, 10, 2) 可以生成一个 1～10 范围内的随机奇数。(　　)

10. 使用 os.remove() 可以用于创建一个新的文件。(　　)

三、填空题

1. os.getcwd()函数可以返回当前工作目录的_____路径。

2. 使用 random.randint(1, 10) 可以生成一个 1～10 范围内的随机_____。

3. datetime.now()可以返回_____的日期和时间。

4. random.choice()可以从列表中随机选择一个_____。

5. 使用 from module_name import * 可以导入模块中的所有_____和变量。

四、简答题

1. 什么是 Python 的模块?
2. 请简要说明 Python 中的 os 模块的作用。
3. 请简要说明 Python 中的 random 模块的作用。
4. 如何在 Python 中处理日期和时间?
5. 请简要说明 Python 中的 math 模块的作用。