

实验报告√ 实践报告□

课程名称： Python编程设计R22

实验、实践名称： Python编程设计

实验、实践地点： 行逸楼A103

专业班级： 软件2212班

学号： 2022005212

学生姓名： 徐鹏宇

指导教师： 廖丽娟

2023年 5月

目 录

[实验一 Python安装与开发环境搭建 1](#_Toc2547)

[实验二 Python序列结构 2](#_Toc2548)

[实验三 Python函数 3](#_Toc2549)

[实验四 Python面向对象程序设计 4](#_Toc2550)

**实验一 Python安装与开发与开发环境搭建**

**一、实验目的和要求**

**1、目的**

本实验的目的是使学生熟悉Python程序运行方式，在熟练掌握Python解释器安装及开发环境搭建的基础上，能编写简单Python程序，培养学生对应用软件的认知、结构、设计及实施能力。

**2、 要求**

（1）熟练掌握Python解释器安装与基本用法，执行Python命令和脚本文件的方法。

（2）掌握Python集成开发环境Pycharm的使用方法

（3）熟练掌握使用pip 命令安装Python 扩展库。

**二、实验内容及步骤**

1、安装Python解释器及集成开发环境Pycharm

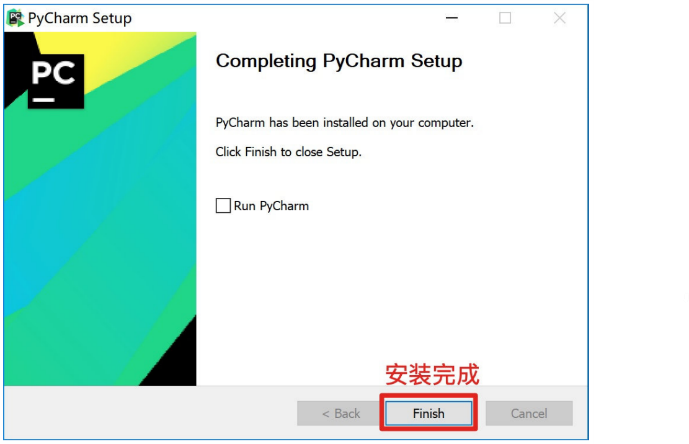
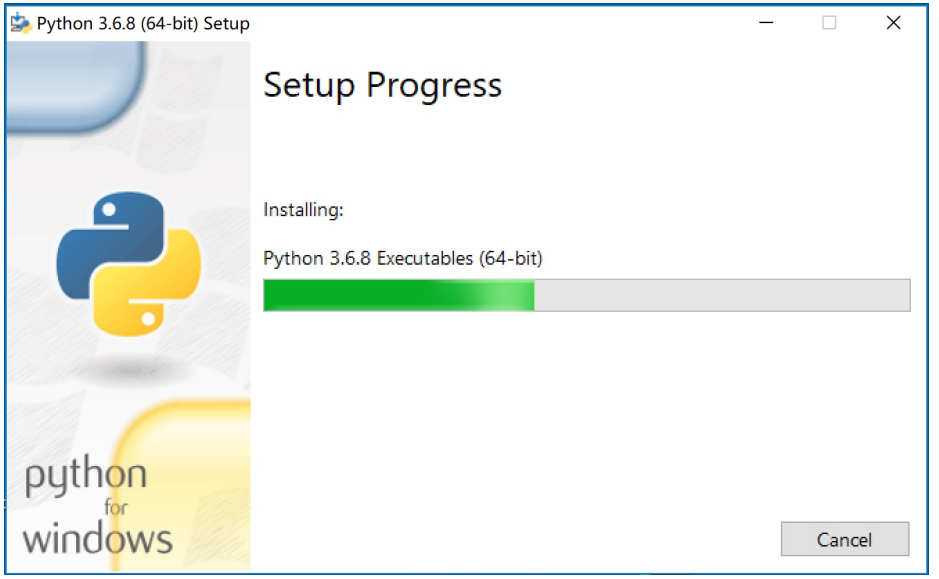


图1-1 图1-2

2、使用pip命令安装Python扩展库

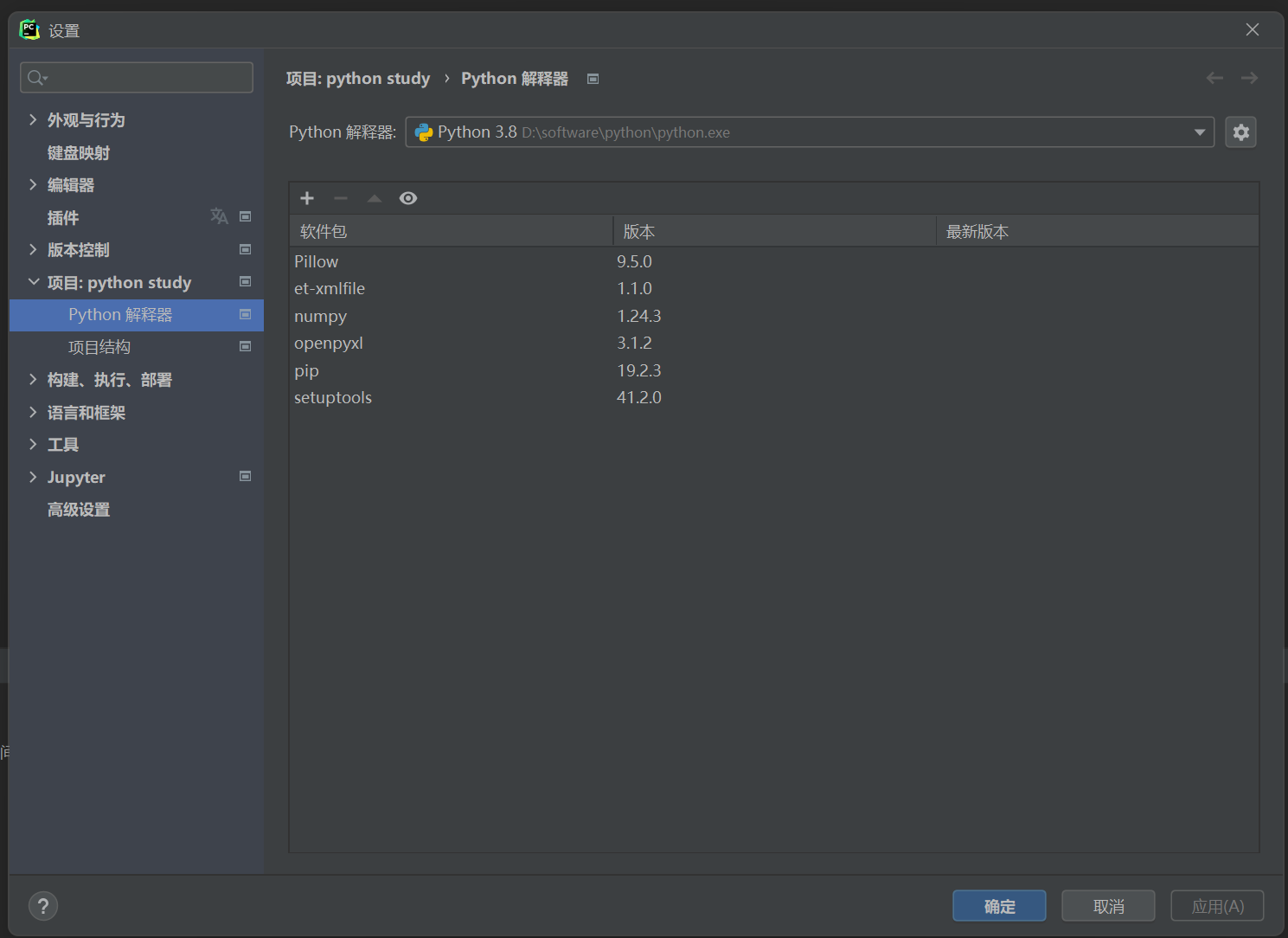


图1-3

**三、小结**

（1）熟练掌握Python解释器安装与基本用法，执行Python命令和脚本文件的方法。

（2）掌握Python集成开发环境Pycharm的使用方法

（3）熟练掌握使用pip 命令安装Python 扩展库。

**实验二 Python序列结构**

**一、实验目的和要求**

**1、目的**

通过Python序列结构的学习，熟练掌握列表、元组、字典和集合的使用方法以及它们之间的区别和适用的不同场景，以提高同学们编程技巧和水平。

**2、要求**

1． 请编写一个程序，能接收用户输入的一个复数的实部和虚部，输出其复数 表示形式，并求其模。

2． 按公式 s=12+22+32+...+n2,求累计加和，s 不超过 1000 的最大项数 n

3． 编写程序，输入一个大于 2 的自然数，然后输出小于该数字的所有素数组 成的列表。

4． 编写程序，输入两个集合 setA 和 setB，分别输出它们的交集、并集和差 集 setA-setB。

5． 阅读下面的代码，解释其功能。

>>> x = [range(3\*i, 3\*i+5) for i in range(2)]

>>> x = list(map(list, x))

>>> x = list(map(list, zip(\*x)))

6． 编写程序，生成一个包含 20 个随机整数的列表，然后对其中偶数下标的元 素进行降序排列，奇数下标的元素不变。（提示：使用切片。）

7． 编写程序，首先生成包含 500 个随机字符的字符串，然后统计每个字符的 出现次数，按每个字符出现次数递减的顺序显示结果。

8． 编写程序，首先生成一个包含 1000 个随机字符的字符串（字符中含有英文 大、小写字母、数字和标点符号），返回一个元组，该元组第一个元素为标 点符号个数、第二个元素为数字字符个数、第三个元素为大写字母个数、第 四个元素为小写字母个数。

**三、实验代码清单**

**1、实验内容1的代码清单**

import math  
x = complex(input())  
print(x)  
print(math.sqrt(math.pow(x.imag, 2) + math.pow(x.real, 2)))

**2、实验内容2的代码清单**

s = 0  
n = 1  
while s < 1000:  
 num = n \*\* 2  
 s += num  
 n += 1  
print(n - 2)

**3、实验内容3的代码清单**

num = int(input('请输入一个大于2的自然数：'))  
num\_1 = []  
for i in range(2, num):  
 for n in range(2, i):  
 if i % n == 0:  
 break  
 else:  
 num\_1.append(i)  
print(num\_1)

**4、实验内容4的代码清单**

setA = {}  
setB = {}  
setA = eval(input('请输入一个集合：'))  
setB = eval(input('再输入一个集合：'))  
setA = set(setA)  
setB = set(setB)  
print('交集：', setA & setB)  
print('并集：', setA | setB)  
print('setA-setB：', setA - setB)

**5、实验内容5的代码清单**

x = [range(3\*i, 3\*i+5) for i in range(2)]

x = list(map(list, x))

x = list(map(list, zip(\*x)))

**6、实验内容6的代码清单**

import random  
x = [random.choice(range(100)) for i in range(20)]  
print(x)  
a = sorted(x[0::2])  
b = x[1::2]  
c = []  
for i in a[:]:  
 c.append(i)  
 a.remove(i)  
 for j in b:  
 c.append(j)  
 b.remove(j)  
 break  
print(c)

**7、实验内容7的代码清单**

import random  
s = ''.join(random.choices('abcdefghijklmnopqrstuvwxyz', k=500))  
char\_count = {}  
for char in s:  
 if char in char\_count:  
 char\_count[char] += 1  
 else:  
 char\_count[char] = 1  
sorted\_char\_count = sorted(char\_count.items(), key=lambda x: x[1], reverse=True)  
for char, count in sorted\_char\_count:  
 print(f'{char}: {count}', end=' ')  
print()

**8、实验内容8的代码清单**

import random  
import string  
import re  
  
def count\_characters(string):  
 punctuation\_count = len(re.findall(r"[^\w\s]", string))  
 digit\_count = len(re.findall(r"\d", string))  
 uppercase\_count = len(re.findall(r"[A-Z]", string))  
 lowercase\_count = len(re.findall(r"[a-z]", string))  
  
 return (punctuation\_count, digit\_count, uppercase\_count, lowercase\_count)  
  
def generate\_random\_string(length):  
 characters = string.ascii\_letters + string.digits + string.punctation  
 random\_string = ''.join(random.choice(characters) for \_ in range(length))  
 return random\_string  
  
random\_string = generate\_random\_string(1000)  
counts = count\_characters(random\_string)  
  
print("Random String:", random\_string)  
print("Punctuation Count:", counts[0])  
print("Digit Count:", counts[1])  
print("Uppercase Count:", counts[2])  
print("Lowercase Count:", counts[3])

**四、实验运行结果**

**1、实验内容1的运行结果**

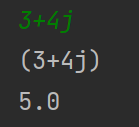
****

图2-1

**2、实验内容2的运行结果**

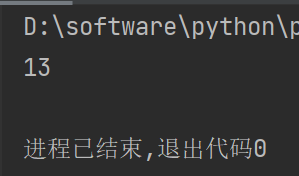
****

图2-2

**3、实验内容3的运行结果**

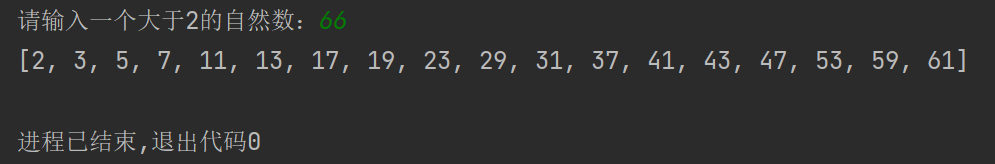
****

图2-3

**4、实验内容4的运行结果**

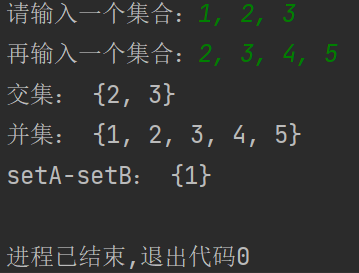
****

图2-4

**5、实验内容5的运行结果**

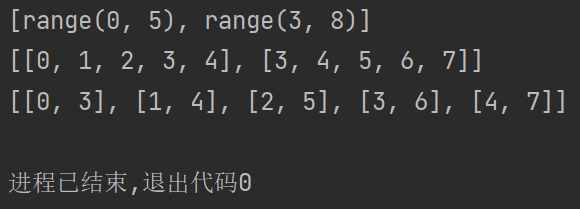


图2-5

**6、实验内容6的运行结果**

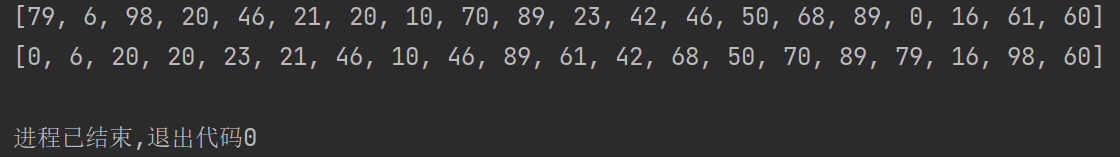
****

图2-6

**7、实验内容7的运行结果**

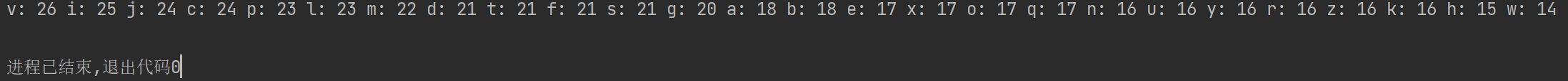
****

图2-7

**8、实验内容8的运行结果**

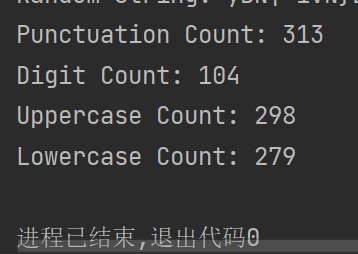
****

图2-8

**五、小结**

通过Python序列结构的学习，已经熟练掌握列表、元组、字典和集合的使用方法以及它们之间的区别和适用的不同场景

**实验三 Python函数**

**一、实验目的和要求**

**1、目的**

通过函数的设计，熟练掌握函数的使用，进一步了解函数的各类参数、及参数传递过程，实现减少程序的代码量，同时提高程序执行时的效率和可维护性。

**2、要求**

1. 编写函数，求出一个数除 1 和自身以外的因子。从键盘输入一个数，调用 该函数输出除 1 和它自身以外的所有因子。

2. 编写两个函数，分别求两个整数的最大公约数和最小公倍数，在主函数调 用这两个函数，并输出结果。两个整数由键盘输入。

3. 编写程序，利用元组作为函数的返回值，求系列类型的最大值、最小值和 元素个数，并编写测试代码。

4. 定义一个函数，可以求任意个数字的和。

5. 定义一个函数，可以将指定列表中的所有的偶数，保存到一个新的列表中 返回。

6. 蒙蒂霍尔悖论游戏 假设你正参加一个有奖游戏节目，并且有 3 道门可选：其中一个后面是汽 车，另外两个 后面是山羊。你选择一个门，比如说 1 号门，主持人当然知道 每个门后面是什么并且打开了 另一个门，比如说 3 号门，后面是一只山羊。 这时，主持人会问你"你想改选 2 号门吗？"， 然后根据你的选择确定最终要 打开的门，并确定你获得山羊（输）或者汽车（赢）。 编写程序，模拟上面的游戏。

7. 创建一个函数 power 来为任意数字做幂运算，如 n \*\* i。 提示：递归实现

**三、实验代码清单**

**1、实验内容1的代码清单**

def s(num):  
 f = []  
 for i in range(2, num):  
 if num % i == 0:  
 f.append(i)  
 return f  
num = int(input("请输入一个正整数: "))  
factors = s(num)  
print("除1和自身以外的因子为:", factors)

**2、实验内容2的代码清单**

def compute\_gcd(a, b):  
 while b != 0:  
 a, b = b, a % b  
 return a  
  
def compute\_lcm(a, b):  
 gcd = compute\_gcd(a, b)  
 lcm = (a \* b) // gcd  
 return lcm  
  
num1 = int(input("请输入第一个整数: "))  
num2 = int(input("请输入第二个整数: "))  
  
gcd = compute\_gcd(num1, num2)  
lcm = compute\_lcm(num1, num2)  
  
print("最大公约数:", gcd)  
print("最小公倍数:", lcm)

**3、实验内容3的代码清单**

def analyze\_series(series):  
 if len(series) == 0:  
 return None  
   
 max\_value = max(series)  
 min\_value = min(series)  
 count = len(series)  
   
 return max\_value, min\_value, count  
  
S1=[9,7,8,3,2,1,55,6]  
result = analyze\_series(S1)  
print("系列类型:", S1)  
print("最大值:", result[0])  
print("最小值:", result[1])  
print("元素个数:", result[2])

**4、实验内容4的代码清单**

def sum\_numbers(\*args):  
 total = 0  
 for num in args:  
 total += num  
 return total  
  
result1 = sum\_numbers(1, 2, 3, 4, 5)  
print(result1)

**5、实验内容5的代码清单**

def f(alist):  
 nlist = []  
 for i in alist:  
 if i % 2 == 0:  
 nlist.append(i)  
 return nlist  
  
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
result = f(numbers)  
print(result)

**6、实验内容6的代码清单**

import random  
  
def monty\_hall\_game():  
 doors = [0, 0, 0] # 三道门，0表示山羊  
 car\_door = random.randint(0, 2) # 随机选择一道门放置汽车  
 doors[car\_door] = 1  
  
 first\_choice = random.randint(0, 2) # 随机选择一道门作为初始选择  
  
 # 主持人打开另一道门（非汽车，非初始选择）  
 opened\_door = 0  
 while opened\_door == car\_door or opened\_door == first\_choice:  
 opened\_door += 1  
  
 # 根据玩家是否改变选择，确定最终选择的门  
 final\_choice = first\_choice  
 change\_choice = random.choice([True, False]) # 随机决定是否改变选择  
 if change\_choice:  
 final\_choice = 3 - first\_choice - opened\_door # 切换选择到剩下的门  
  
 # 判断最终结果  
 if doors[final\_choice] == 1:  
 return "Win" # 最终选择是汽车，获胜  
 else:  
 return "Lose" # 最终选择是山羊，失败  
  
# 进行10000次模拟  
wins = 0  
losses = 0  
  
for \_ in range(10000):  
 result = monty\_hall\_game()  
 if result == "Win":  
 wins += 1  
 else:  
 losses += 1  
  
# 打印结果  
print("获胜次数：", wins)  
print("失败次数：", losses)

**7、实验内容7的代码清单**

def power(n, i):  
 if i == 0:  
 return 1  
 elif i < 0:  
 return 1 / power(n, -i)  
 else:  
 return n \* power(n, i - 1)  
  
print(power(2, 3))  
print(power(5, 0))  
print(power(3, -2))

**四、实验运行结果**

**1、实验内容1的运行结果**

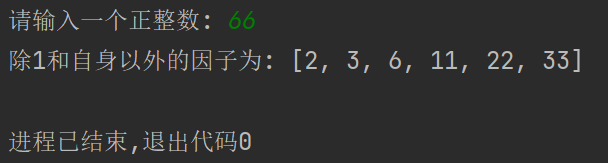
****

图3-1

**2、实验内容2的运行结果**

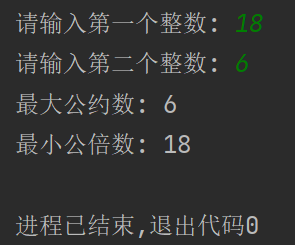


图3-2

**3、实验内容3的运行结果**

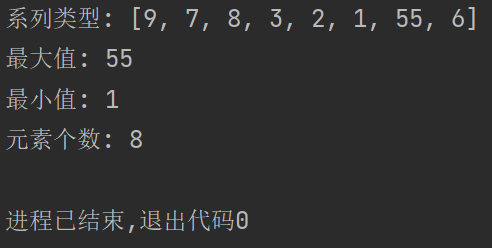


图3-3

**4、实验内容4的运行结果**

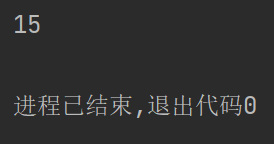


图3-4

**5、实验内容5的运行结果**

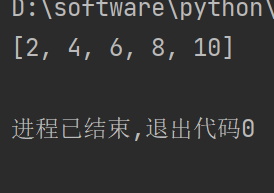


图3-5

**6、实验内容6的运行结果**

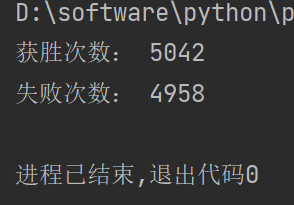


图3-6

**7、实验内容7的运行结果**

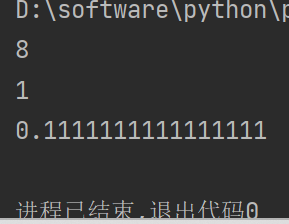
****

图3-7

**五、小结**

已熟练掌握函数的使用，进一步了解函数的各类参数、及参数传递过程。

**实验四 Python面向对象程序设计**

**一、实验目的和要求**

**1、目的**

面向对象程序设计的方法是从日常处理问题思路中形成的一种新的设计方法。 通过了解对象、类、封装、继承、方法、构造函数和析构函数等面向对象的程序设 计的基本概念，掌握 Python 类的定义、类的方法、类的继承等，为解决复杂现实 问题，培养编程能力及工程应用技能，打下坚实的基础。

**2、要求**

1． 了解面向对象的程序设计思想

2． 掌握 Python 对象、类、封装、继承、方法、构造函数和析构函数等.

3． 掌握 Python 的类的继承与组合

**二、实验内容**

1. 设计一个 Circle 类来表示园，这个类包含圆的半径以及求周长和面积的函 数。再使用这个类创建半径为 1~100 的圆，并计算出相应的周长及面积。

2. 编写程序并测试，有一个汽车类 Vehicle，类中的私有数据成员为车轮个数 wheels 和车重 weight；小车类 Car 是汽车类 vehicle 派生类，其中包含载客人数 passenger\_load，默认 4 人；卡车类 Truck 是汽车类 vehicle 派生类，其中包含载 客人数 passenger\_load 和载重量 payload。

提示：编写类时要使用 get 和 set 方法，Vehicle 类定义 display()显示车轮 和重量，Car 和 Truck 中要对 display()进行同名覆盖。

3. 编写一个学生和教师数据输入和输出的程序。学生数据包括编号、姓名、 班号、和成绩；教师的数据含有编号、姓名、职称和部门。要求设计一个 person 类，作为学生数据操作类 student 和教师数据操作类 teacher 的基类

**三、实验代码清单**

**1、实验内容1的代码清单**

import math  
class Circle:  
 def \_\_init\_\_(self, r):  
 self.r = r  
  
 def C(self):  
 c = math.pi \* 2 \* self.r  
 return c  
  
 def S(self):  
 s = math.pi \* self.r \* self.r  
 return s  
  
for radius in range(1, 101):  
 circle = Circle(radius)  
 circumference = circle.C()  
 area = circle.S()  
 print(f"半径为 {radius} 的圆，周长为 {circumference:.2f}，面积为 {area:.2f}")

**2、实验内容2的代码清单**

class Vehicle:

def \_\_init\_\_(self, wheels, weight):

self.\_\_wheels = wheels

self.\_\_weight = weight

def get\_wheels(self):

return self.\_\_wheels

def get\_weight(self):

return self.\_\_weight

class Car(Vehicle):

def \_\_init\_\_(self, wheels, weight, passenger\_load=4):

super().\_\_init\_\_(wheels, weight)

self.\_\_passenger\_load = passenger\_load

def get\_passenger\_load(self):

return self.\_\_passenger\_load

class Truck(Vehicle):

def \_\_init\_\_(self, wheels, weight, passenger\_load, payload):

super().\_\_init\_\_(wheels, weight)

self.\_\_passenger\_load = passenger\_load

self.\_\_payload = payload

def get\_passenger\_load(self):

return self.\_\_passenger\_load

def get\_payload(self):

return self.\_\_payload

car = Car(4, 1500)

print("Car - Wheels:", car.get\_wheels())

print("Car - Weight:", car.get\_weight())

print("Car - Passenger Load:", car.get\_passenger\_load())

truck = Truck(6, 5000, 2, 10000)

print("Truck - Wheels:", truck.get\_wheels())

print("Truck - Weight:", truck.get\_weight())

print("Truck - Passenger Load:", truck.get\_passenger\_load())

print("Truck - Payload:", truck.get\_payload())

**3、实验内容3的代码清单**

class Person:

def \_\_init\_\_(self, id, name):

self.id = id

self.name = name

class Student(Person):

def \_\_init\_\_(self, id, name, class\_num, score):

super().\_\_init\_\_(id, name)

self.class\_num = class\_num

self.score = score

def display(self):

print(f"Student ID: {self.id}")

print(f"Name: {self.name}")

print(f"Class Number: {self.class\_num}")

print(f"Score: {self.score}")

class Teacher(Person):

def \_\_init\_\_(self, id, name, title, department):

super().\_\_init\_\_(id, name)

self.title = title

self.department = department

def display(self):

print(f"Teacher ID: {self.id}")

print(f"Name: {self.name}")

print(f"Title: {self.title}")

print(f"Department: {self.department}")

student = Student(1, "Alice", "Class A", 90)

student.display()

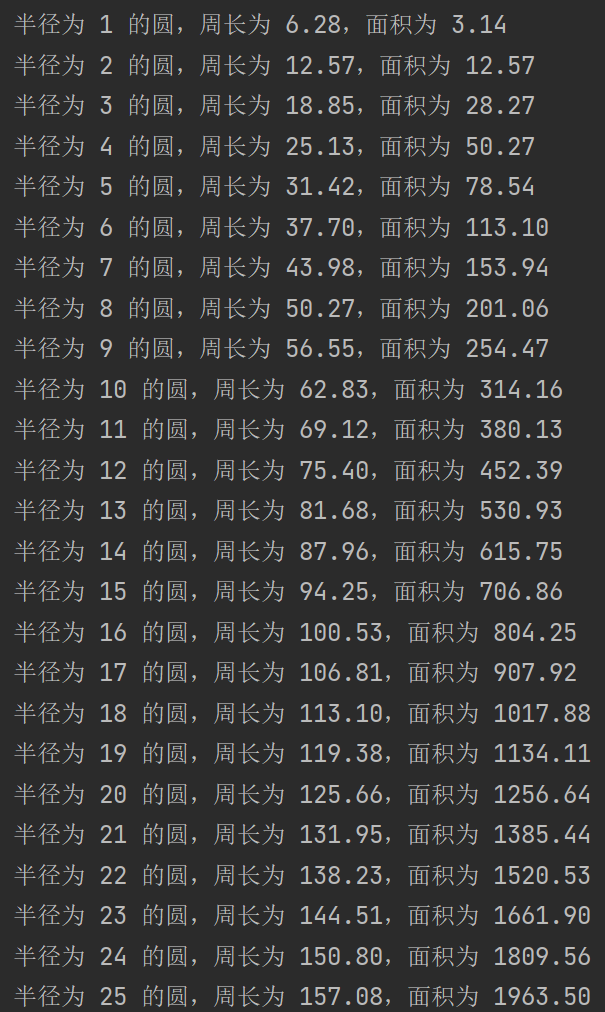
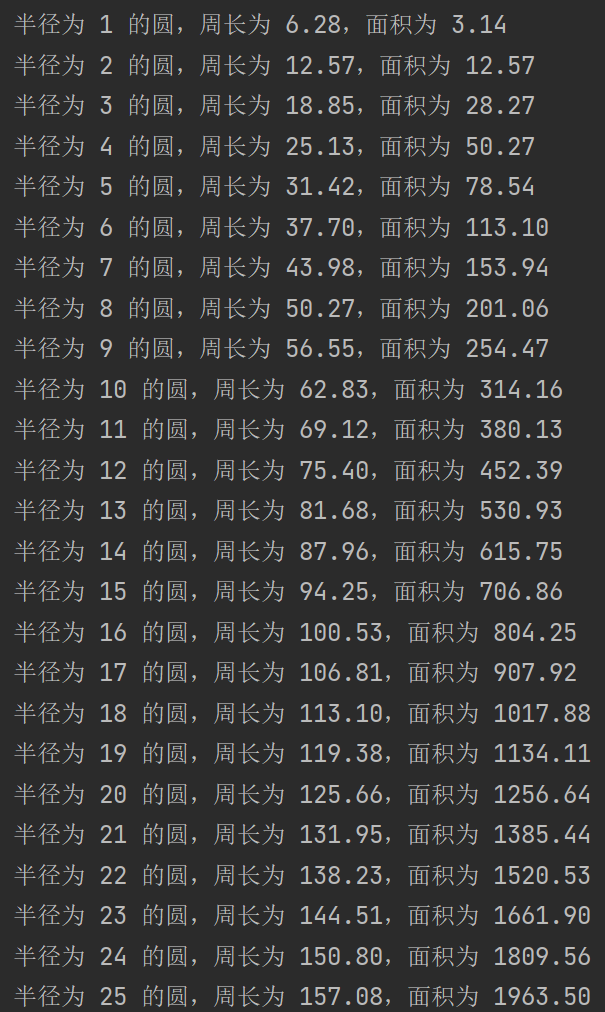
print()

teacher = Teacher(101, "John", "Professor", "Mathematics")

teacher.display()

**四、实验运行结果**

**1、实验内容1的运行结果**

****

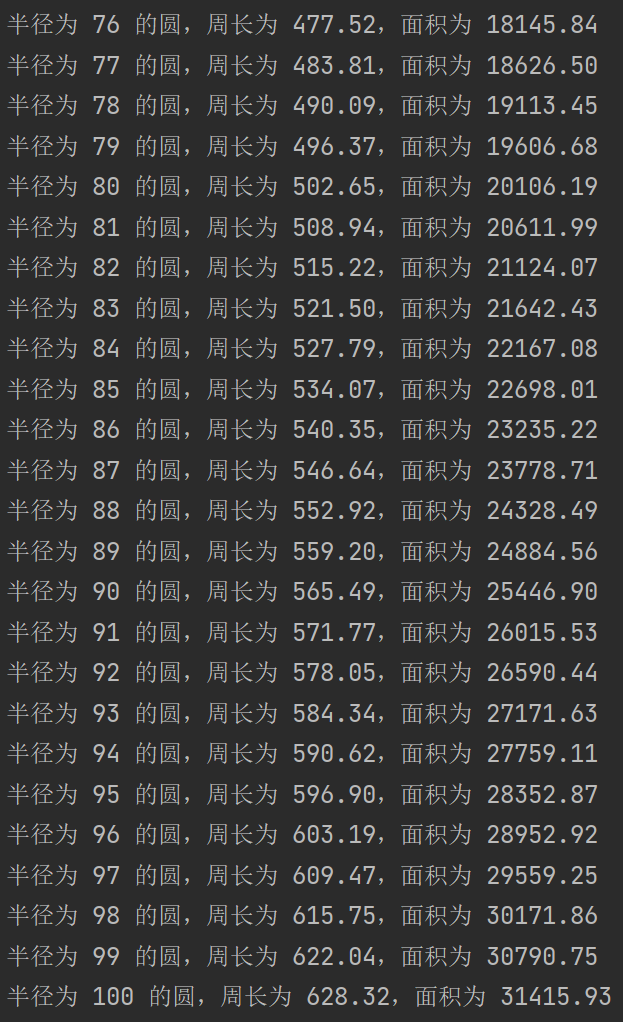
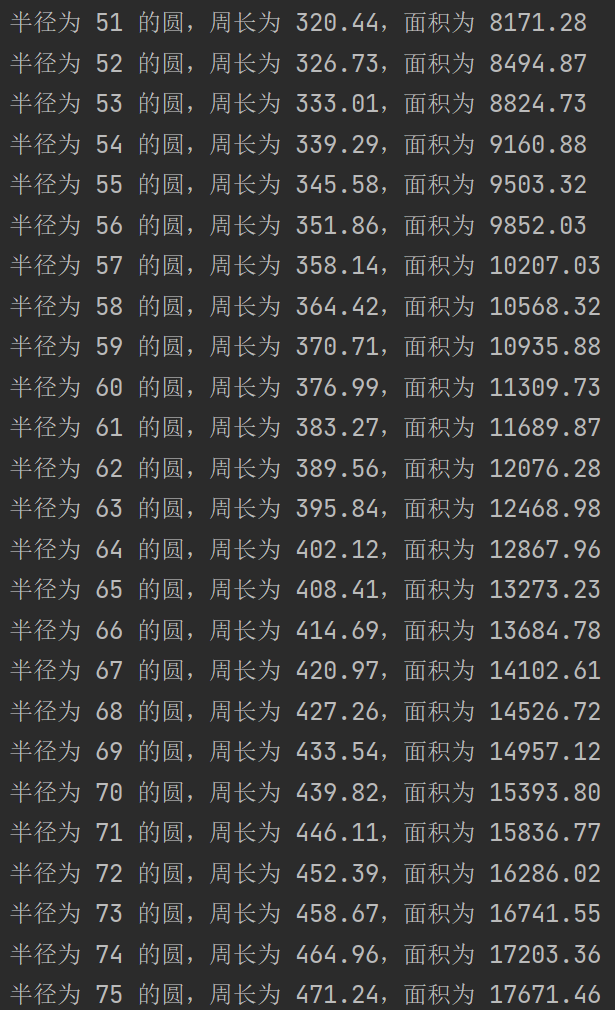


图4-1

**2、实验内容2的运行结果**

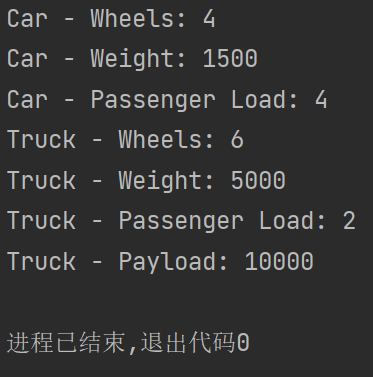
****

图4-2

**3、实验内容3的运行结果**

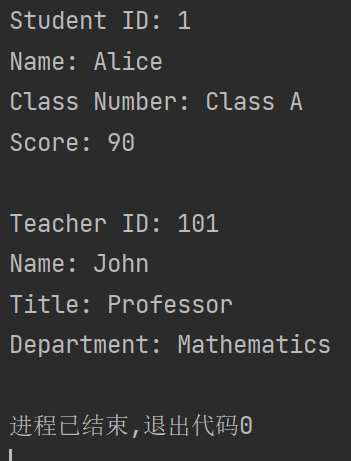


图4-3

**五、小结**

（1）已熟练掌握 Python 对象、类、封装、继承、方法、构造函数和析构函数等.

（2）已熟练掌握 Python 的类的继承与组合