深 圳 大 学 实 验 报 告

课程名称：Python 程序设计

实验项目名称：实验 5：面向对象编程

学院：计算机与软件学院

专业：软件工程

指导教师：潘浩源

报告人：郑彦薇 学号：2020151022

实验时间：2022/5/6~2022/5/26

实验报告提交时间：2022/5/21

教务部制

一、实验目的

用 python 语言编写解决问题的代码并运行。

运用 python 中面向对象编程的知识点实现对问题的解决。

二、实验方法步骤

1、读题，对每个问题提出解决问题的思路

2、按照得到的思路，利用 python 语言编写解决问题的代码

3、运行代码，调试程序，直至程序可以正确输入输出

三、实验过程及内容

（一）运行以下代码，解释输出的原因

问题(a)：

1. 运行结果：

```
5:30

进程已结束，退出代码为 0
```

2. 输出原因分析：

主函数中，传入类的“5：30”对类的属性 time 进行了初始化。调用类中的 print_time()函数时，虽然该函数中对 time 进行了赋值，但此过程并没有真正更改类属性 time 的值，因此进行输出时，得到结果“5：30”.

问题(b)：

1. 运行结果：

```
10:30

进程已结束，退出代码为 0
```

2. 输出原因分析：

在 Clock 类中，print_time 函数接收传进来的参数，并对该值进行输出。在主函数中，虽然通过对对象的定义初始化属性 time 的值，但在调用类中的 print_time 函数时传入了另一个真正被输出的值。因此输出结果为“10：30”.

问题(c)：

1. 运行结果：

```
10:30
10:30

进程已结束，退出代码为 0
```

2. 输出原因分析：

这段代码定义了一个 clock 对象为 boston_clock，并对属性 time 进行初始化。paris_clock=boston_clock 实际上是将两个对象指向同一个类，在使用 boston_clock 或者 parts_clock 对属性的值进行更改，在进行输出时都会得到同一个结果。因此在执行 paris_clock=“10：30”语句后，会得到上述输出。

（二）编写类 RegularPolygon，表示正 n 边形

1. 解决问题的思路和方法

1) 首先编写构造函数，包含题目要求的四个属性：边数、边长、中心坐标横坐标 x 以及中心坐标横坐标 y；

2) 编写函数计算正 n 边形的周长：因为正 n 边形每一条边长都相同，因此计算周长只需返回边数与边长的乘积；

3) 编写函数计算正 n 边形的面积：

①正 n 边形的面积计算公式：$S=\frac{1}{2}nR^2\sin\varphi$（其中 R 为正 n 边形外接圆的半径，φ 为边所对圆心角）

②正 n 边形边长与外接圆半径关系：$side=2R\sin(\frac{\pi}{n})$

根据上述公式对正 n 边形的面积公式进行推导，有：

$$S=\frac{1}{2}n\sin\varphi\cdot R^2=\frac{1}{2}n\sin(\frac{2\pi}{n})\cdot\frac{1}{4}\frac{side^2}{\sin(\pi/n)^2}=\frac{1}{4}n\cdot side^2/\tan(\pi/n)$$

因此计算面积时对该式子的求解结果进行返回即可；

4) 编写函数计算两个正 n 边形中心的距离：根据点与点之间的距离公式 $dis=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$，在求解时对该公式的计算结果进行返回即可。

代码及具体解释如下：

```
1  import math
2  class RegularPolygon(object):
3      def __init__(self, n=3, side=1, x=0, y=0):
4          self.__n = n
5          self.__side = side
6          self.__x = x
7          self.__y = y
8      def getPerimeter(self):
9          return self.__side * self.__n
10     def getArea(self):
11         return 0.25 * self.__n * (self.__side) ** 2 / math.tan(math.pi / self.__n)
12     def distanceToPolygon(self, Rp):
13         return math.sqrt((self.__x - Rp.__x) ** 2 + (self.__y - Rp.__y) ** 2)
14
15 print("请输入正n边形的信息：")
16 n1 = int(input())
17 side1 = float(input())
18 x1 = float(input())
19 y1 = float(input())
20 Rp1 = RegularPolygon(n1, side1, x1, y1)
21 print(f'C = {Rp1.getPerimeter()}')
22 print(f'S = {round(Rp1.getArea(), 4)}')
23 print("请输入另一个正n边形的信息：")
24 n2 = int(input())
25 side2 = float(input())
26 x2 = float(input())
27 y2 = float(input())
28 Rp2 = RegularPolygon(n2, side2, x2, y2)
29 print(f'D = {round(Rp1.distanceToPolygon(Rp2), 4)}')
```

构造函数的编写，并对私有成员进行初始化

3 个功能函数的编写，根据各自计算公式返回计算结果

主函数的编写，通过输入数据验证函数的正确性

2. 运行结果展示

输入第一个 n 边形的信息为：3，2，0，0；第二个 n 边形的信息为：4，1，1，1；得到结果输出为：

```
请输入正n边形的信息：
3
2
0
0
C = 6.0
S = 1.7321
请输入另一个正n边形的信息：
4
1
1
1
D = 1.4142

进程已结束，退出代码为 0
```

3. **遇到的问题与收获**

在进行面积函数的编写时，对正多边形的边长与外接圆半径的关系、正多边形面积与外接圆半径的关系有了更深的认识。

（三）自定义数据结构栈

1. 解决问题的思路和方法

1) 首先编写构造函数，包含题目要求的三个属性：存放栈数据的列表 content、栈容量 size 以及当前栈数据的个数 current；
2) Isempty 函数的编写：该函数用于判断栈是否为空栈，只需判断当前栈数据的个数是否为 0；
3) Empty 函数的编写：该函数用于清空栈信息，只需将栈当前数据个数设置为 0，将存放栈数据列表的元素清空；
4) setSize 函数的编写：该函数用于重新设置栈容量，只需将新的栈容量的值赋给私有成员 size，并处理以下情况：若新的栈容量小于原始栈容量，将超出栈容量的列表中的元素进行删除；
5) isFull 函数的编写：该函数用于判断栈是否空，只需判断栈的容量与当前栈中的数据个数是否相同；
6) Push 函数的编写：首先判断栈是否已满，若栈已满，则输出栈满信息，不进行新元素的入栈；若栈未满，则根据列表添加元素的方法添加新元素，并将栈当前数据个数的数量加 1；
7) Pop 函数的编写：首先判断栈是否已空，若栈已空，输出栈空信息，不进行任何操作；若栈未空，则根据列表删除元素的方法将当前列表最后一个元素进行删除，并将栈当前数据个数的数量减 1

代码及细节解释如下：

```
1    class Stack(object):
2        def __init__(self, content, size, current):
3            self.__content = content                    构造函数的编写
4            self.__size = size
5            self.__current = current
```

```
    def isempty(self):
        if self.__current == 0:
            return True
        return False
    def empty(self):
        self.__content.clear()
        self.__current = 0
    def setSize(self, temp):
        self.__size = temp
        while self.__size < self.__current:
            self.pop()
    def isFull(self):
        if self.__current >= self.__size:
            return True
        return False
    def push(self, item):
        if self.isFull():
            print("the stack is full!")
            return
        self.__current += 1
        self.__content.append(item)
    def pop(self):
        if self.isempty():
            print("no items!")
            return
        self.__current -= 1
        self.__content.pop()

    def print_stack(self):
        print(f'当前栈数据个数：{self.__current},当前栈容量：{self.__size}')
        print(self.__content)
```

压栈前先判断栈是否已满

出栈前先判断栈是否已空

自行设计测试函数：

```
S = Stack([], 5, 0)#初始化栈信息：空栈，栈的容量为5，栈当前数据个数为0
S.print_stack()
while 1:#压栈与栈满测试
    item = int(input())
    S.push(item)
    S.print_stack()

S = Stack([0, 1, 2, 3], 5, 4)#初始化栈信息：栈的容量为5，栈当前数据个数为4
cnt = 6#为栈空判断设置测试次数，相当于结束条件
while cnt:#测试出栈与栈空
    S.print_stack()
    S.pop()
    cnt -= 1

S = Stack([0, 1, 2, 3], 5, 4)#初始化栈信息：栈的容量为5，栈当前数据个数为4
S.print_stack()
S.empty()#置空栈测试
S.print_stack()

S = Stack([0, 1, 2, 3], 5, 4)#初始化栈信息：栈的容量为5，栈当前数据个数为4
S.print_stack()
print("请输入新的栈容量：")#设置新的栈容量测试
newSize = int(input())
S.setSize(newSize)
S.print_stack()
```

2. 运行结果展示

1) 压栈与栈满测试：

```
当前栈数据个数：0,当前栈容量：5
[]
1
当前栈数据个数：1,当前栈容量：5
[1]
3
当前栈数据个数：2,当前栈容量：5
[1, 3]
4
当前栈数据个数：3,当前栈容量：5
[1, 3, 4]
5
当前栈数据个数：4,当前栈容量：5
[1, 3, 4, 5]
6
当前栈数据个数：5,当前栈容量：5
[1, 3, 4, 5, 6]
7
the stack is full!
当前栈数据个数：5,当前栈容量：5
[1, 3, 4, 5, 6]
```

2) 出栈与栈空测试:

```
当前栈数据个数：4,当前栈容量：5
[0, 1, 2, 3]
当前栈数据个数：3,当前栈容量：5
[0, 1, 2]
当前栈数据个数：2,当前栈容量：5
[0, 1]
当前栈数据个数：1,当前栈容量：5
[0]
当前栈数据个数：0,当前栈容量：5
[]
no items!
当前栈数据个数：0,当前栈容量：5
[]
no items!
```

3) 置空栈测试:

```
当前栈数据个数：4,当前栈容量：5
[0, 1, 2, 3]
当前栈数据个数：0,当前栈容量：5
[]
```

4) 栈容量重置测试：

```
当前栈数据个数：4,当前栈容量：5
[0, 1, 2, 3]
请输入新的栈容量：
6
当前栈数据个数：4,当前栈容量：6
[0, 1, 2, 3]

当前栈数据个数：4,当前栈容量：5
[0, 1, 2, 3]
请输入新的栈容量：
3
当前栈数据个数：3,当前栈容量：3
[0, 1, 2]
```

3. 遇到的问题与收获

在解决该问题时，探索了如何使用列表及如何编写函数实现对栈的操作以及进行一些对栈信息的判断。

（四）继承 1

1. 解决问题的思路和方法

1) Person 类构造函数的编写：将类中的各成员初始化为定义类时传入的参数即可；
2) Person 类各方法（即各 set 函数）的编写：对应方法目的，将私有成员更改为传入的参数；
3) Teacher 类：根据继承规则，按照上述方法补全 setDepartment 函数；对该类各成员的输出，首先调用 person 中的 show 函数对 person 的私有成员进行输出，再对变量 department 进行输出即可。

代码及细节解释如下：

Person 类各方法的实现：

```python
class Person(object):
    def __init__(self, name='', age=20, sex='man'):
        self.__name = name
        self.__age = age
        self.__sex = sex

    def setName(self, name):
        self.__name = name

    def setAge(self, age):
        self.__age = age

    def setSex(self, sex):
        self.__sex = sex

    def show(self):
        print(f'Name:{self.__name}')
        print(f'Age:{self.__age}')
        print(f'Sex:{self.__sex}')
```

Teacher 类各方法的实现：

```python
class Teacher(Person):
    def __init__(self, name='', age=30, sex='man', department='Computer'):
        Person.__init__(self, name, age, sex)
        self.setDepartment(department)

    def setDepartment(self, department):
        self.department = department

    def show(self):
        Person.show(self)
        print(f'Department:{self.department}')
```

主函数：

```python
if __name__ == '__main__':
    zhangsan = Person('Zhang San', 19, 'man')
    zhangsan.show()
    print("--------------")#无实际意义，方便查看输出

    lisi = Teacher('Li Xi', 32, 'man', 'Math')
    lisi.show()
    print("--------------")#无实际意义，方便查看输出
    lisi.setAge(40)
    lisi.setName("Li Si")
    lisi.show()
```

2. 运行结果展示

```
Name:Zhang San
Age:19
Sex:man
--------------
Name:Li Xi
Age:32
Sex:man
Department:Math
--------------
Name:Li Si
Age:40
Sex:man
Department:Math
```

3. 遇到的问题与收获

在实现 teacher 类的输出时：Teacher 类继承了 Person 类，可以直接调用 Person 中的输出函数实现 Teacher 类部分成员的输出；Teacher 存在属于自身的成员变量，该成员按常规类方法进行初始化及输出即可。

（五）继承 2

1. 运行结果展示

```
Li
MingLi
From China
DavidBeckham
From England
```

2. 运行结果分析

1) 首先分析各类之间的关系：通过代码可以看到 Guangdong 和 England 继承了 China 类；
2) 第一行输出的分析：在主函数中，通过 ming=Guangdong()创建了 Guangdong 类，在创建时，‘Ming’和‘Li’作为初始值传入 China 类的构造函数，从而在调用 China 类中的 execute 函数对 family 变量进行输出时，得到第一行结果，即‘Li’；
3) 第二、三行输出的分析：主函数第三行使用 test_person 函数并使用 ming 进行初始化，而在主函数一开始，已经创建了 ming 这个 Guangdong 类，因此在 test_person 函数进行输出时，调用了__str__函数，即对类中的名、姓，以及通过 get_description 对国籍进行输出；从而得到第二、第三行结果；
4) 第四、五行输出的分析：主函数第四行类似第三行的作用，创建了一个 England 类但不进行初始化，在 England 类中，仍然是调用 China 中的__str__函数对名和姓进行输出，再通过 get_description 对国籍进行输出，从而得到第四、第五行结果。

3. 遇到的问题与收获

直接将类作为参数传入函数中，可以调用类中的各成员函数实现函数功能有如该题中的 print 功能。

（六）Numpy 基础

1. 运行结果展示

a. **第一段代码：**

```
The shape: (5,)
The dimensionality: 1
The type: float64
The shape: (2, 5)
The dimensionality: 2
The type: float64
```

b. **第二段代码：**

```
(5, 5)
```

c. **第三段代码：**

```
 [[ 1 36]
  [ 4  9]
  [ 9 25]]
 [[1.         2.44948974]
  [1.41421356 1.73205081]
  [1.73205081 2.23606798]]
 [[1 6]
  [2 3]
  [3 5]]
 [[[9 9 8]
   [7 8 4]
   [1 1 4]]

  [[1 9 4]
   [8 6 4]
   [6 0 1]]

  [[5 1 5]
   [8 9 8]
   [3 0 1]]]
 [[[-11 -11   8]
   [-13 -12   4]
   [  1   1   4]]

  [[-19 -11   4]
   [-12 -14   4]
   [  6   0   1]]

  [[-15 -19   5]
   [-12 -11   8]
   [  3   0   1]]]
```

d. **第四段代码：**

```
 [3.5 1.  2.5]
```

2. **运行结果分析**

a. **第一段代码：**

1) 对此段代码中运用的功能进行解释：

①shape 函数返回一个元组，表示各位维度大小；

②ndim 函数返回一个值，表示数组的维度；ndim 得到的维度为 1，则上述 shape 函数得到的结果元组中只会有一个值（如第一行结果为(5,)而不是(1, 5)）；

③dtype 函数返回数组的数据类型，是一个用于说明数组数据类型的对象。

2) 对 mysqrt 列表的操作：此段代码首先定义并初始化两个列表，将 mysqrt 列表强制类型转换为数组，mysqrt 产生的列表元素是 0~4 开根的结果，通过强制类型转换后，得到一个 1*5 的数组，然后调用 numpy 中上述函数对该数组的信息（即维度、数据

类型）进行输出，得到前三行结果；

3) 对 mycrt 列表的操作：mycrt 产生的列表元素是 0~4 开三次方根的结果，通过强制类型转换同时处理 mysqrt 和 mycrt 列表，得到一个大小为 2*5 的二维数组，同样调用上述 1）中提到的函数对该数组信息进行输出，可以得到后三行结果。

b. 第二段代码：

1) 各数组的定义与初始化：

①zeros 是一个大小为 3 的数组，数组元素均为 0；

②zMat 是一个大小为 4 行 3 列的二维数组，数组元素均为 0；

③ones 是一个大小为 3 的数组，数组元素均为 1；

④oMat 是一个大小为 3 行 2 列的二维数组，数组元素均为 1；

⑤diag 是通过 eye 函数生成一个 4*4 的二维数组，其中对角线元素为 1，其余元素均为 0；

⑥rng 是通过 arrange 函数得到的一个等差数组，这里只有一个参数，默认起点从 0 开始，默认步长为 1；

⑦dm 是通过已经定义的 diag（单位矩阵）以及 rng（等差数组），按照除对角线外其他元素值为 0 的特点，将等差数组中的元素填入单位矩阵中 1 的位置，得到一个大小为 5*5，对角元素为 0~4，其它位置元素为 0 的矩阵。

2) 输出：根据 a 中已知 shape 函数的作用，这里对 dm 数组的维度进行输出，于是得到结果(5, 5).

3) Reshape 语句解释：该语句的作用是重新规定 zMat 数组的维度，根据上面所说可以知道，zMat 数组维度为 4*3，通过该语句，将数组维度重新规定为 6*2，即得到一个 6 行 2 列的、元素均为 0 的二维数组。

c. 第三段代码：

1) 数组的创建与初始化：首先使用 numpy 中的 random 函数创建数组 A、B、C，根据语句可以知道：A 是一个大小为 3*2 的二维数组，数组中的元素为随机生成的整数，且元素数据范围从 0~9；B 是包含三个大小均为 3*3 的二维数组的大矩阵，数组中的元素同样是在范围 0~9 内随机生成的整数；C 是一个大小为 3*1 的二维数组，元素同样是 0~9 的随机整数；

2) 对数组的操作与输出：

①对 A 的第一个操作：首先是对 A 进行平方操作，即输出一个数组，数组元素为 A 中对应元素的平方，注意此过程不会改变 A；

②对 A 的第二个操作：调用 numpy 中的 sqrt 函数，对 A 进行开方操作，即输出一个数组，数组元素为 A 中对应元素的开方，此过程同样不会改变 A；

③A+C：该语句是将数组 A 与数组 C 进行相加，即 A 中每一行元素与 C 中每一行元素进行相加，对于该实例，则是 A 中每一行的两个元素与 C 中对应行的同一个元素进行相加，结果是一个维度为 3*2 的数组；

④B+C：该语句同样是进行两个数组的相加，B 的三个数组分别与 C 进行相加，B 的第一个数组的每一行的每个元素与 C 中对应行的元素进行相加，B 中剩下的数组同理，结果是一个包含 3 个 3*3 的数组的大矩阵；

⑤对 B 的切片操作：该语句是对于 B 中的每一个数组中的元素进行切片操作，执行效果是切取出每一个数组中 0~2 行、0~2 列的元素，对这些元素进行−20 的操作，其他元素则保持不变，结果得到一个与 B 结构相同的大矩阵。

d. **第四段代码：**

1) 首先了解子库与该段代码中使用的功能：linalg 是 numpy 的线性代数子库，solve 函数的作用是求解多元一次方程组；

2) 具体解释：在此段代码中，首先定义了两个数组，一个是维度为 3*3 的 A 数组，一个是长度为 3 的一维数组即 b 数组，solve 函数在此处的作用即是求解出 x 使得数组 A 与 x 相乘得到数组 b，结果如上述运行结果所示。

3. 遇到的问题与收获

在解决该问题时，可以通过对于不做输出的内容进行输出，如每一段代码中创建的数组，尤其是对于随机生成的数组，将这些内容进行输出，与进行一定操作后的输出进行比对，可以更好的观察数组在此过程中的变化，可以更好的理解 numpy 中对于数组操作时一些函数的实际作用。

（七）Numpy 应用

1. 解决问题的思路和方法

1) 2n*2n 矩阵的给出：提供 N 的输入，根据输入确定矩阵大小，并借助 numpy 中的 random 函数对矩阵进行填值；

2) 矩阵的划分：利用切片功能对矩阵进行划分，行列都以 N 为划分点，以 0 为开始，以 2N 为结束，即可实现对四个象限的划分；

3) 平均值的计算：因为结果是一个 2*2 的数组，该数组规模并不大因此在编程中直接通过下标访问每一个位置，借助 numpy 中的 average 函数，对每一个位置赋值为对应象限矩阵的平均值。

代码及细节解释如下：

```
1  import numpy as np
2  print("请输入N:", end=' ')
3  N = int(input())
4  print(f'矩阵的规模为{N*2}行{N*2}列，随机生成此规模矩阵为：')
5  nums = 4*N*N
6  myMat = np.random.randint(0, 10, size=(2*N, 2*N))
7  print(myMat)
8  A1 = myMat[0:N, 0:N]
9  A2 = myMat[0:N, N:2*N]
10 A3 = myMat[N:2*N, 0:N]
11 A4 = myMat[N:2*N, N:2*N]
12 #print(f'A1={A1}\nA2={A2}\nA3={A3}\nA4={A4}')
13 myAver = np.zeros((2, 2))#创建一个全0的2*2数组，用于存放四个象限的平均值
14 myAver[0][0] = np.average(A1)
15 myAver[0][1] = np.average(A2)
16 myAver[1][0] = np.average(A3)
17 myAver[1][1] = np.average(A4)
18 print(f'myAver={myAver}')
```

利用 random 函数对矩阵进行赋值

对矩阵进行划分

对存放平均值的矩阵的每个位置进行赋值

2. 运行结果展示

```
请输入N: 4
矩阵的规模为8行8列，随机生成此规模矩阵为：
[[7 8 2 1 7 8 6 3]
 [1 1 4 9 9 0 3 7]
 [1 5 0 4 7 7 3 9]
 [0 5 2 9 2 5 2 1]
 [3 3 8 8 9 2 1 3]
 [7 3 7 4 6 8 5 9]
 [2 0 1 2 0 2 9 0]
 [4 5 6 7 5 6 8 1]]
myAver=[[3.6875 4.9375]
 [4.375  4.625 ]]
```

3. 遇到的问题与收获

对于矩阵的创建、切片、数值计算等操作更加熟悉。

（八）Numpy 应用 2

1. 解决问题的思路和方法

1) 首先读入需要进行处理的图片，使用 Image 的 open 函数，对指定位置的图片进行读入；
2) 将原始图片转换为灰度图片：使用 convert 函数，将图片进行指定转换即可；再使用 save 函数将图片存放到指定位置，并将其命名为 img_gray.jpg；
3) 将图像进行裁剪：首先将图片转换为数组形式，以获取图片信息；根据裁剪目的，以图片一半的高度进行划分，截取上半部分记为 img_crop，并将该图片存放到指定位置和命名；
4) 将图片垂直翻转：使用 Image 中的 transpose 函数，指定图片的翻转为垂直翻转（即 FLIP_TOP_BOTTOM），并将翻转后的图片进行保存和命名；
5) 其他处理：对图片进行任意角度翻转，输入想要翻转的角度，利用 rotate 函数对图片进行旋转，并将旋转结果进行保存。

代码及细节解释如下：

图片读入以及调用库：

```python
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

#读入图片
image = Image.open('C:/Users/4334/PycharmProjects/pythonProject_test5/view.jpg')
```

将原始图片转换为灰度图片：

```python
#将原始图片转换为灰度图片
def imgToGray():
    img_gray = image.convert('L')
    img_gray.show()
    img_gray.save('C:/Users/4334/PycharmProjects/pythonProject_test5/img_gray.jpg')
```

进行上半部分的裁剪：

```python
#将图像裁剪到剩下图像的上半部分
def imgToCrop():
    image1 = np.array(image)
    H, W = image1.shape[0], image1.shape[1]
    H1 = H//2
    img_crop = image1[0:H1, :, :]
    plt.imshow(img_crop)
    plt.show()
    plt.imsave('C:/Users/4334/PycharmProjects/pythonProject_test5/img_crop.jpg', img_crop)
```

垂直翻转：

```python
#垂直翻转图片
def imgToFlip():
    img_flip_vert = image.transpose(Image.Transpose.FLIP_TOP_BOTTOM)
    img_flip_vert.show()
    img_flip_vert.save('C:/Users/4334/PycharmProjects/pythonProject_test5/img_flip_vert.jpg')
```

指定角度旋转：

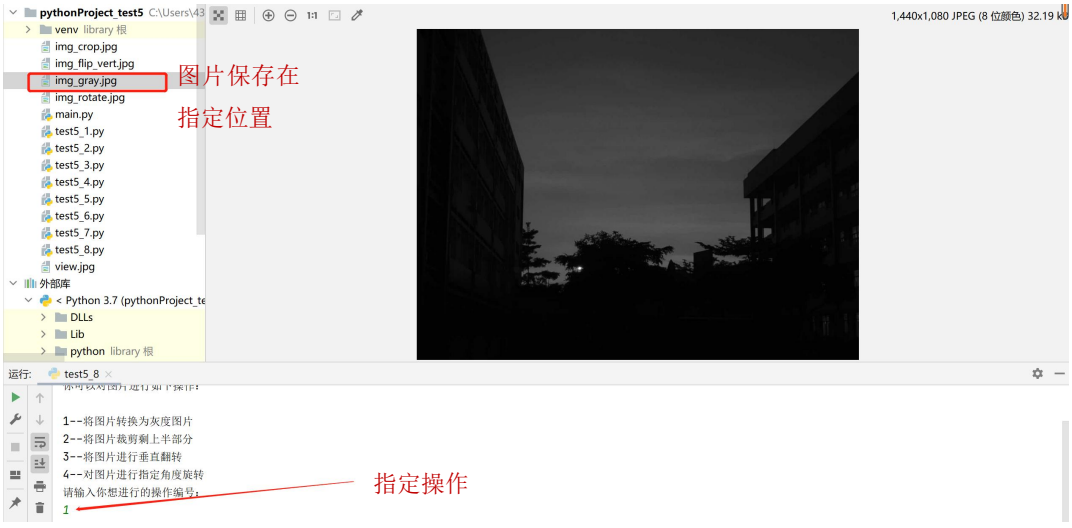
```python
#其他处理：对图片进行任意指定角度的旋转
def imgToRotate():
    print("输入你想翻转的角度：")
    angle = int(input())
    img_rotate = image.rotate(angle)
    img_rotate.show()
    img_rotate.save('C:/Users/4334/PycharmProjects/pythonProject_test5/img_rotate.jpg')
```

主函数：

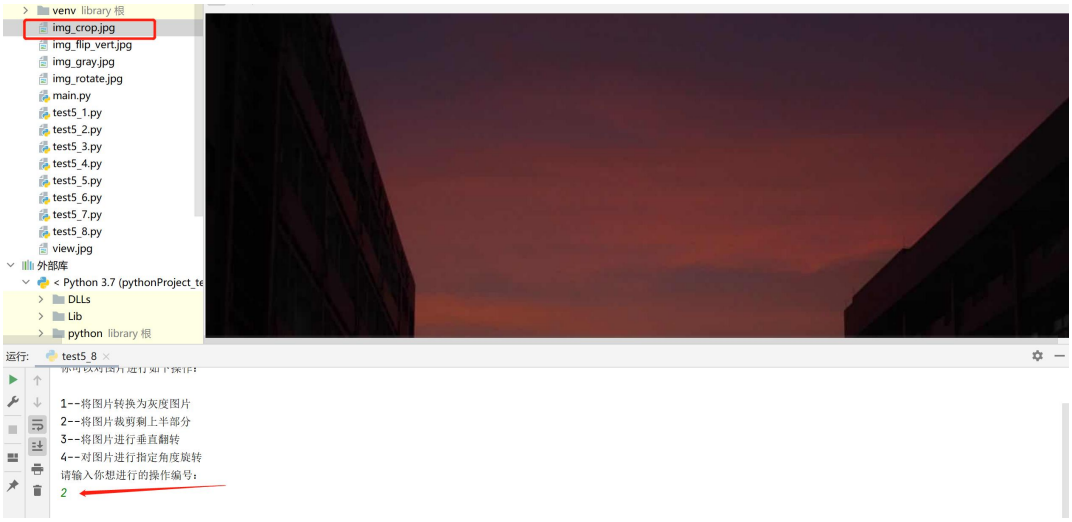
```python
#统一操作
print("现在有一张图片，如图所示\n你可以对图片进行如下操作：\n")
print("1--将图片转换为灰度图片")
print("2--将图片裁剪剩上半部分")
print("3--将图片进行垂直翻转")
print("4--对图片进行指定角度旋转")
print("请输入你想进行的操作编号：")
cnt = int(input())
if cnt == 1:
    imgToGray()
elif cnt == 2:
    imgToCrop()
elif cnt == 3:
    imgToFlip()
elif cnt == 4:
    imgToRotate()
```

2. 运行结果展示：

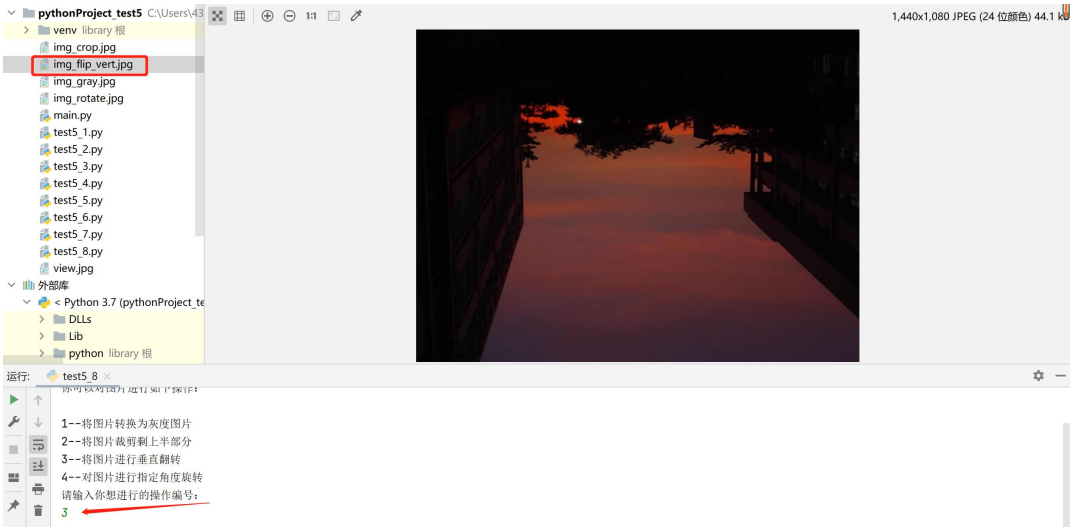
将原始图片转换为灰度图片：



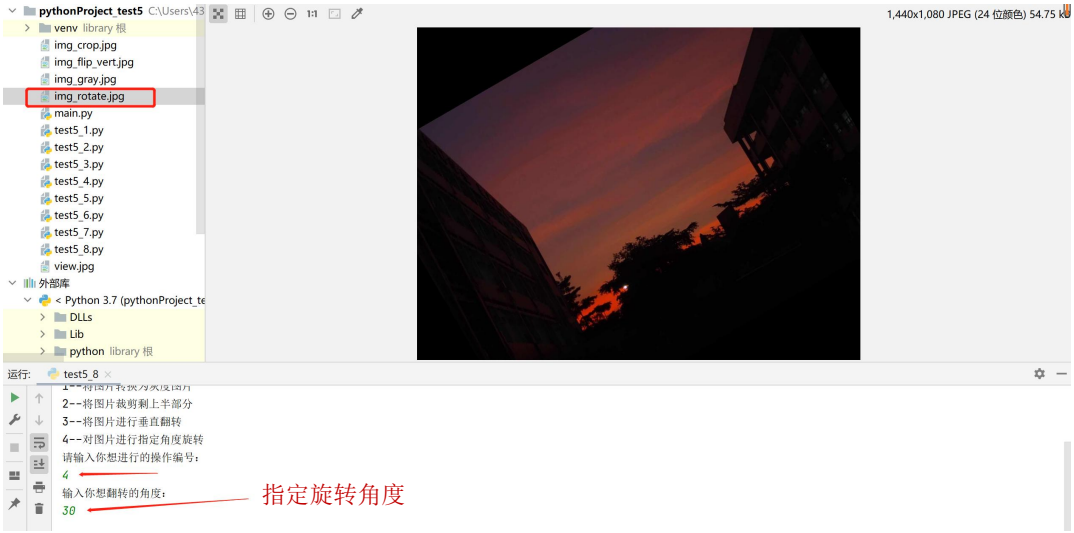
进行上半部分的裁剪（同样进行指定操作以及图片的保存）：



垂直翻转：



指定角度旋转：



3. 遇到的问题与收获

学习了如何读取、修改以及保存图片。

（九）Python 标准库 itertools

1. 解决问题的思路和方法

1) 对于该题的解决，可以使用标准库 itertools 的 combinations 函数实现；

2) 首先对于长度为 8 的列表，其非空子集的长度可以为 1~8，因此首先规定当前子集长度，再使用 combinations 函数得到指定长度下的子集组合，再使用 sum 函数计算当前子集组合中所有数的和，对和进行判断，若存在一组组合数和为 0，则返回 True。

代码及细节解释如下：

```python
import itertools
def sum0(lst):
    for length in range(1, 9):
        subset = itertools.combinations(lst, length)
        for i in subset:
            if sum(i) == 0:
                return True
    return False

lst = [-3, 11, 21, 5, 10, 11, 2, 1]
if sum0(lst):
    print("True")
else:
    print("False")
```

设置循环以对非空子集的各种长度情况进行分析

获取指定长度下不同组合迭代器

计算子集中各数和

2. 运行结果展示

当 lst=[-3, 11, 21, 5, 10, 11, 2, 1]：

```
True
```

当 lst=[2, 3, 4, 5, 6, 7, 8, 9]：

```
False
```

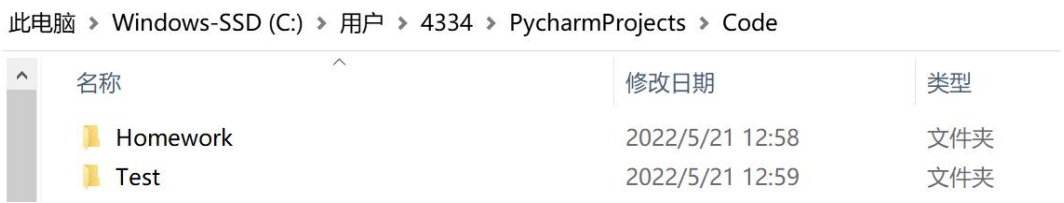
3. 遇到的问题与收获

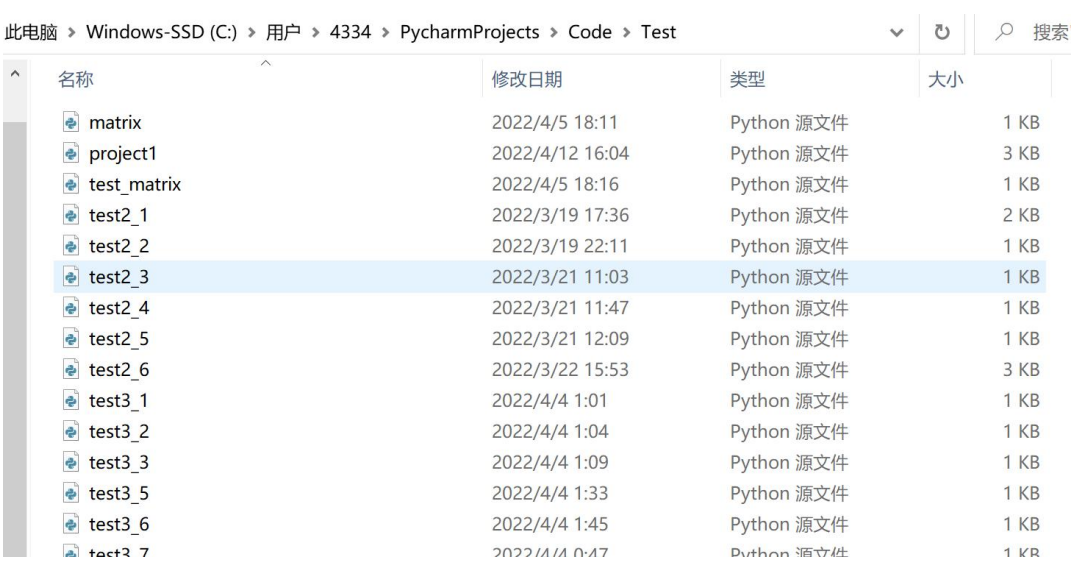
在实现该题的过程中，对标准库 itertools 的功能有了进一步的认识。

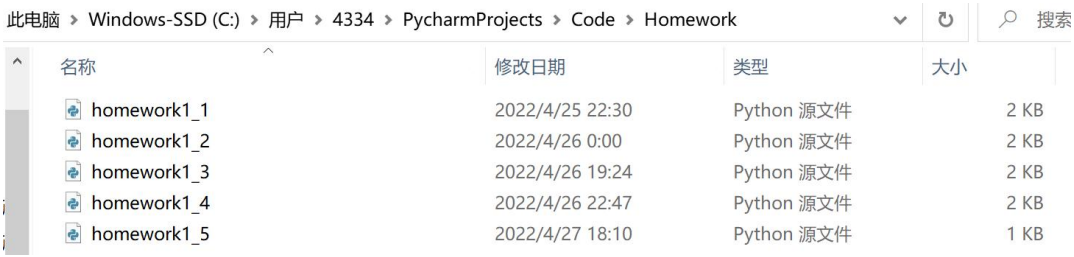
（十）Python 标准库 collections，sys，os：统计目前写过的 python 代码

1. 解决问题的思路和方法

1) 首先根据要求创建相应的文件夹，包括作业 1、实验 1~4 的代码文件；







2) 统计代码文件夹信息的函数编写：首先定义各统计变量并初始化为 0，设置循环，对于位于“代码”文件夹中的文件夹，按照路径格式访问每一个文件；
①统计 python 文件的个数：使用 endswith 功能判断文件名结尾是否是‘.py’，如果是则对应计数器加 1；
②统计 python 文件中的行数：打开当前进行统计的 python 文件，访问每一行，并将行数加一；接着对每一行进行判断：如果是空行，则对应计数器 space_lines 加 1；用 startswith 判断当前行如果以“#”开头，计数器 comments_lines 加 1；
③最后以元组格式返回结果即可。

3) 对用户输入统计：以字符串形式读取路径，在函数中首先对用户输入路径进行判断，若不存在此路径，输出错误信息并返回；该函数只需在（2）的基础上增加对指定 python 文件的统计方法（因为该题所创建文件中只包含 python 代码文件），当读入文件后缀为“.py”时，首先将文件数量加 1，接着访问该文件的每一行，统计 code_line、space_lines、comments_lines 的值，最后以元组形式返回。

代码及细节解释如下：

库的调用：

```
1  import collections
2  import sys
3  import os
```

创建文件信息的统计：

```python
def Collect(myFlie):
    file_num = 0
    code_line = 0
    space_lines = 0
    comments_lines = 0
    for i in os.listdir(myFlie):
        for j in os.listdir(myFlie+'/'+i):
            if j.endswith('.py'):
                file_num += 1
                for line in open(myFlie+'/'+i+'/'+j, encoding='utf-8'):
                    code_line += 1
                    if line == "\n":
                        space_lines += 1
                    if line.startswith('#'):
                        comments_lines += 1
    return(file_num, code_line, space_lines, comments_lines)
```

定义各统计值并初始化为 0

按照路径格式访问代码文件夹中的子文件夹以及各文件信息

用户输入统计：

```python
def Input_Collect(myFlie):
    file_num = 0
    code_line = 0
    space_lines = 0
    comments_lines = 0
    if not os.path.exists(myFlie):
        print("have no such file!")
        return
    if os.path.isdir(myFlie):
        for i in os.listdir(myFlie):
            for j in os.listdir(myFlie + '/' + i):
                if j.endswith('.py'):
                    file_num += 1
                    for line in open(myFlie + '/' + i + '/' + j, encoding='utf-8'):
                        code_line += 1
                        if line == "\n":
                            space_lines += 1
                        if line.startswith('#'):
                            comments_lines += 1
    elif os.path.isfile(myFlie) and myFlie.endswith('.py'):
        file_num += 1
        for line in open(myFlie, encoding='utf-8'):
            code_line += 1
            if line == '\n':
                space_lines += 1
            if line.startswith('#'):
                comments_lines += 1
    return(file_num, code_line, space_lines, comments_lines)
```

路径输入正确性判断

对 python 文件的处理

主函数：

```python
print(Collect('C:/Users/4334/PycharmProjects/Code'))
print("请正确输入你想统计的文件或文件名地址：")
str = input()
print(Input_Collect(str))
```

2. 运行结果展示

对“代码文件的统计”：

所创建的代码文件夹的信息：

```
(39, 1262, 124, 61)
```

用户输入统计：

```
请正确输入你想统计的文件或文件名地址：
C:/Users/4334/PycharmProjects/Coda
have no such file!
None
请正确输入你想统计的文件或文件名地址：
C:/Users/4334/PycharmProjects/Code
(39, 1262, 124, 61)

请正确输入你想统计的文件或文件名地址：
C:/Users/4334/PycharmProjects/Code/Test/project1.py
(1, 124, 14, 8)
```

3. 遇到的问题与收获

对文件夹中的每一个文件进行访问时，通过资料查询学习到只需设置循环并以正确路径访问格式进行访问即可。

四、实验总结

1. 通过本次实验，首先是对于面向对象编程有了更深入的认识，有如类的私有成员的访问方法、类的继承以及继承中对于父类信息的输出、使用类成员函数实现对同一个内容的多种操作等；
2. 其次是更加熟悉如何运用 numpy 中的各种方法解决问题，有如数组的计算、图片处理，了解 numpy 库功能；
3. 最后是对 python 中的标准库的认识与应用，学会了如何借助标准库功能实现对于文件信息的获取和输出，以及如何处理列表。

指导教师批阅意见：

成绩评定：

指导教师签字：

年　　月　　日

备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。