Python 程序设计 实验 5：面向对象编程

注意事项：

（1） 实验报告提交**截止日期：2022.05.26， 23:59pm**，迟交扣 20%，缺交 0 分。

（2） 实验报告内容包括：解决问题的思路与方法（如代码的解释）、遇到的问题以及收获（简单描述即可）、代码运行结果的展示。

（3） 实验报告提交方法：**blackboard**。

（4） 提交要求：实验报告+源代码，打包上传，命名：学号_姓名_实验报告_5。

（5） **禁止抄袭，一经发现 0 分处理（包括抄袭者和提供代码或实验报告者）!**

1. 运行以下代码，解释输出的原因。

(a)

```
class Clock(object):
    def __init__(self, time):
        self.time = time
    def print_time(self):
        time = "6:30"
        print(self.time)

clock = Clock("5:30")
clock.print_time()
```

(b)

```
class Clock(object):
    def __init__(self, time):
        self.time = time

    def print_time(self, time):
        print (time)

clock = Clock("5:30")
clock.print_time("10:30")
```

(c)

```
class Clock(object):
    def __init__(self, time):
        self.time = time
```

```
    def print_time(self):
        print(self.time)

boston_clock = Clock("5:30")
paris_clock = boston_clock
paris_clock.time = "10:30"
boston_clock.print_time()
paris_clock.print_time()
```

2. 编写类 RegularPolygon，表示正 n 边形。类包括：
   - 私有成员 n，要求为整型，代表正 n 边形边的数量，注意 $n \geq 3$；
   - 私有成员 side，代表正 n 边形每条边的长度；
   - 私有成员 x，代表正 n 边形中心的坐标在 x 轴上的数值；
   - 私有成员 y，代表正 n 边形中心的坐标在 y 轴上的数值；
   - 构造函数，输入参数为 n（默认为 3），side（默认为 1），x（默认为 0），y（默认为 0）；
   - 方法 getPerimeter，返回正 n 边形的周长；
   - 方法 getArea，返回正 n 边形的面积；
   - 方法 distanceToPolygon，输入参数为另外一个正 n 边形，返回两个正 n 边形中心的距离。

   自行设计测试函数验证 RegularPolygon 类代码的正确性。

3. 自定义数据结构栈。栈是一种后进先出（Last-In-First-Out）的数据结构。编写类 Stack，实现入栈、出栈、判断栈是否为空，是否满栈、以及改变栈容量等操作。Stack 类包括：
   - 私有成员 content，为一个列表，代表栈里的数据；
   - 私有成员 size，要求为整型，代表栈的容量；
   - 私有成员 current，要求为整型，代表栈当前数据的个数；
   - 方法 isempty，判断栈是否为空，返回 True/False；
   - 方法 empty，置空栈；
   - 方法 setSize，输入参数为新的栈的容量。注意新的栈容量可能小于原有的栈容量，统一将后进的元素删除；
   - 方法 isFull，判断栈是否为空，返回 True/False；
   - 方法 push，入栈，输入参数为新的元素；
   - 方法 pop，出栈；

- 方法 show，打印当前栈的数据。

自行设计测试函数验证 Stack 类代码的正确性。

4. 继承 1：补充代码 lab5_4.py,使得代码输出如下。注：不允许在类中添加新的方法。

```
if __name__ =='__main__':
    zhangsan = Person('Zhang San', 19, 'man')
    zhangsan.show()
    #Name: Zhang San
    #Age: 19
    #Sex: man

    lisi = Teacher('Li Xi',32, 'man', 'Math')
    lisi.show()
    #Name: Li Xi
    #Age: 32
    #Sex: man
    #Department: Math

    lisi.setAge(40)
    lisi.setName("Li Si")
    lisi.show()
    #Name: Li Si
    #Age: 40
    #Sex: man
    #Department: Math
```

5. 继承 2:研究以下代码，思考代码的输出，并解释。

```
class China:
    def __init__(self, given, family):
        self.given = given
        self.family = family
    def __str__(self):
        return self.given + ' '   + self.family + '\n' + self.get_description()
```

```
        def get_description(self):
            return 'From China'
        def execute(self):
            print(self.family)

    class Guangdong(China):
        def __init__(self):
            China.__init__(self, 'Ming', 'Li')

    class England(China):
        def __init__(self):
            China.__init__(self, 'David', 'Beckham')
        def get_description(self):
            return 'From England'

    def test_person(person):
        print(person)

    ming = Guangdong()
    ming.execute()
    test_person(ming)
    test_person(England())
```

6. **Numpy 基础:** 运行以下代码，理解每句代码的意思或输出结果（print 语句）：

(a)

```
    mysqrt = [math.sqrt(x) for x in range(0,5)]
    mycrt = [x**(1/3) for x in range(0,5)]

    npData = np.array(mysqrt)
    print("The shape:", npData.shape)
    print("The dimensionality:", npData.ndim)
    print("The type:", npData.dtype)

    twoDarray = np.array([mysqrt, mycrt])
    print("The shape:", twoDarray.shape)
    print("The dimensionality:", twoDarray.ndim)
    print("The type:", twoDarray.dtype)
```

(b)

```
zeros = np.zeros(3)
zMat = np.zeros((4,3))
ones = np.ones(3)
oMat = np.ones((3,2))
diag = np.eye(4)
rng = np.arange(5)
dm = np.diag(rng)
print(dm.shape)
zMat_re = zMat.reshape(6,2)
```

(c)

```
A = np.random.randint(0,10, size = (3,2))
B = np.random.randint(0,10, size = (3,3,3))
C = np.random.randint(0,10, size = (3,1))
print(A**2)
print(np.sqrt(A))
print(A + C)
print(B + C)
B[:, 0:2 , 0:2 ] -= 20
print(B)
```

(d)

```
from numpy import linalg
A = np.array([[2, 1, -2],[1,-1,-1], [1, 1 ,3]])
b = np.array([3, 0, 12])
x = linalg.solve(A,b)
print(x)
```

7. **Numpy 应用 1**：给定一个矩阵 2n*2n，将该矩阵分成四个象限（参见示例），然后返回一个新的 2*2 矩阵，包含每个象限的平均值。

例子：

原矩阵：

```
1, 2 , 5 , 7
4, 1 , 8 , 0
2, 0 , 5 , 1
0, 2 , 1 , 1
```

```
 1, 2 | 5 , 7
 4, 1 | 8 , 0
----------------
 2, 0 | 5 , 1
 0, 2 | 1 , 1
```

新矩阵：

```
(1+2+4+1)/ 4   | (5+7+8+0)/4                    2.0 , 5.0
----------------------------          =>       1.0 , 2.0
(2+0+0+2)/4    | (5+1+1+1)/4
```

8. **Numpy 应用 2**：仿照课件中利用 numpy 处理图片的方法，选择一张自己喜欢的图片进行处理。
   （1） 将原始图片转换为灰度图片。在灰度图片中。对于特定的 X，Y 坐标的像素值，RGB 3 个通道值是相同的，由 p=0.299R+0.587G+0.114B 得到，其中，R、G、B 是每个对应通道的值。将输出图片保存到 img_gray.png。
   （2） 将图像裁剪到剩下图像的上半部分。将输出图片保存到 img_crop.png。
   （3） 垂直翻转原始图片。也就是说，将其沿水平线翻转，将翻转的图片保存到 img_flip_vert.png
   （4） 其他你喜欢的处理。

9. **Python 标准库 itertools**：编写函数 sum0(lst)，接受一个包含 8 个整数的列表 lst。如果列表中的某些非空子集的总和返回 0，则返回 True。例如，lst=[-3, 11, 21, 5, 10, 11, 2, 1]返回 True，因为非空子集[-3, 2, 1]中数字加起来总和为 0；又如 lst=[2, 3, 4, 5, 6, 7, 8, 9]时，函数返回 False。

10. **Python 标准库 collections, sys, os: 统计目前写过的 python 代码**。

(1) 把实验、作业的代码文件整理好，分开放在“作业”文件夹和“实验”文件夹，它们又放到同一个文件夹“代码”中（你也可以增加其他文件或文件夹）。

(2) 编写函数，统计“代码”文件夹中 python 文件的个数（file_num）,写过的代码行数（code_line），代码中空行的行数（space_lines），注释的行数（comments_lines，只统计以#开头的），返回元组记录上述信息。

(3) 在（2）基础上，允许用户输入指定的文件或文件夹，统计输入文件或文件夹的信息。例如，假设 python 文件名为 code_stat.py，运行方法如下：

```
>>> python code_stat.py  代码/作业                  #统计“作业”文件夹
>>> python code_stat.py  代码/实验/实验 2.py     #只统计实验 2.py
```