# 一、实验内容

1）编写代码，实现一个栈（Stack）的类。栈是只能在一端插入和删除数据的序列。它按照先进后出的原则存储数据，先进入的数据被压入栈底，最后的数据在栈顶，需要读数据的时候从栈顶开始弹出数据（最后一个数据被第一个读出来）。

2）编写代码，定义一个形状基类，有2个属性：面积和周长，以及两个无返回值的方法：area()和perimeter()，分别计算形状的面积和周长，从基类派生出三个子类：三角形、矩形、圆，重载基类的两个方法。

# 二、详细实现

1.

class MyStack():

''' My own Stack class'''

def \_\_init\_\_(self):

self.S = []

self.length = 0

def push(self, value):

'''Push value in the stack'''

self.S.append(value)

self.length += 1

def pop(self):

''' Pop the last element in the stack'''

if len(self.S) != 0:

self.length -= 1

return self.S.pop()

def printStack(self):

''' print the all elements in the Stack'''

print("The stack size is {}".format(self.length))

for \_ in self.S:

print(\_, end='')

print()

S = MyStack()

S.push(1)

S.push(2)

S.push(3)

S.printStack()

print(S.pop())

S.printStack()

print(func(src))

2.

class Geometric():

pi = 3.1415926

def \_\_init\_\_(self):

self.area=0

self.perimeter=0

self.Area()

self.Perimeter()

self.printGeometric()

def Area(self):

pass

def Perimeter(self):

pass

#print the value of Geometric

def printGeometric(self):

print("The Area is {}, the Perimeter is {}".format(self.area, self.perimeter))

class Triangle(Geometric):

def \_\_init\_\_(self, a, b, c):

''' need each edge's length(a, b, c)'''

self.a = int(a)

self.b = int(b)

self.c = int(c)

super().\_\_init\_\_()

def Area(self):

a = self.a

b = self.b

c = self.c

p = (a+b+c) / 2

self.area = (p\*(p-a)\*(p-b)\*(p-c))\*\*0.5

def Perimeter(self):

self.perimeter = self.a + self.b + self.c

Angle = Triangle(3, 4, 5)

print(Angle.perimeter)

print(Angle.area)

class Rectangle(Geometric):

''' need the length and width'''

def \_\_init\_\_(self, length, width):

self.length = length

self.width = width

super().\_\_init\_\_()

def Area(self):

self.area = self.length \* self.width

def Perimeter(self):

self.perimeter = 2 \* (self.length + self.width)

Res = Rectangle(2, 2)

print(Res.area)

print(Res.perimeter)

class CircleAround(Geometric):

'''need the "r"'''

def \_\_init\_\_(self, r):

self.r = r

super().\_\_init\_\_()

def Area(self):

self.area = self.pi \* self.r\*\*2

def Perimeter(self):

self.perimeter = self.pi \* 2 \* self.r

r1 = CircleAround(4)

print(r1.area)

# 三、实验结果

调试成功，通过几个简单测试点

# 四、心得体会

熟练使用了制作自己的Class并进行实例化，也学会了写basic类并且使用super()进行多重继承。