LeapYear\_NotOOP

def isLeapYear(newYear):

if newYear % 4 == 0 and newYear % 100 != 0 or newYear % 400 == 0:

return True

else:

return False

for i in range(0,3):

x = input()

if isLeapYear(int(x)):

print("The ", x , " is leap year.")

else:

print("The ", x , " isn't leap year.")

StudentCard\_Composition

class Date:

def \_\_init\_\_(self,month,day,year):

self.\_\_month = month

self.\_\_day = day

self.\_\_year = year

def setMonth(self,month):

self.\_\_month = month

def getMonth(self):

return self.\_\_month

def setDay(self,day):

self.\_\_day = day

def getDay(self):

return self.\_\_day

def setYear(self,year):

self.\_\_year = year

def getYear(self):

return self.\_\_year

def toString(self):

print(self.\_\_month,'\\' ,self.\_\_day,'\\',self.\_\_year,end = '')

class Student:

def \_\_init\_\_(self, name, date, score):

self.\_\_name = name

self.\_\_date = date

self.\_\_score = score

def setName(self, name):

self.\_\_name = name

def getName(self):

return self.\_\_name

def setDate(self, date):

self.\_\_date = date

def getDate(self):

return self.\_\_date

def setScore(self, score):

self.\_\_score = score

def getScore(self):

return self.\_\_score

def toString(self):

print(self.\_\_name,' ', end = '')

self.getDate().toString()

print(' ',self.\_\_score)

from Student import Student

from Date import Date

s1 = Student("John", Date(6, 1, 1999), 90)

s2 = Student("Marry", Date(10, 8, 1997), 80)

s1.setName(input())

s2.setDate( Date( input(), input(), input() ) )

s1.toString()

s2.toString()

Rectangle\_Encapsulation

class Rectangle:

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

self.\_\_length = length

self.\_\_width = width

def setLength(self, length):

self.\_\_length = length

def setWidth(self, width):

self.\_\_width = width

def getLength(self):

return self.\_\_length

def getWidth(self):

return self.\_\_width

def getArea(self):

return (self.\_\_length \* self.\_\_width)

def getPerimeter(self) :

return ((self.\_\_length + self.\_\_width) \* 2)

from Rectangle import Rectangle

a = input()

b = input()

c = input()

d = input()

r1 = Rectangle( int(a), int(c) )

r2 = Rectangle( int(b), int(d) )

print(r1.getArea(),'',r1.getPerimeter())

print(r2.getArea(),'',r2.getPerimeter())

r2.setLength(50)

r2.setWidth(25)

print(r2.getArea(),'',r2.getPerimeter())

GeometricObject\_Inheritance

from Circle import Circle

from Rectangle import Rectangle

x = input()

w = input()

h = input()

c1 = input()

f1 = input()

c2 = input()

f2 = input()

C = Circle( int(x), c1, bool(f1) )

R = Rectangle( int(w), int(h), c2, bool(f2) )

print('Circle:')

print('Radius is ', C.getRadius())

print('Diameter is ', C.getDiameter())

print('Area is ', C.getArea())

print('Perimeter is ', C.getPerimeter())

C.\_\_str\_\_()

print()

print()

print('Rectangle:')

print('Width is ', R.getWidth())

print('Height is ', R.getHeight())

print('Area is ', R.getArea())

print('Perimeter is', R.getPerimeter())

R.\_\_str\_\_()

class GeometricObject:

def \_\_init\_\_(self, color, filled):

self.\_\_color = color

self.\_\_filled = filled

def getColor(self):

return self.\_\_color

def setColor(self, color):

self.\_\_color = color

def isFilled(self):

return self.\_\_filled

def setFilled(self, filled):

self.\_\_filled = filled

def \_\_str\_\_(self):

print('color: ', self.\_\_color, ' and ', 'filled: ', self.\_\_filled)

from GeometricObject import GeometricObject

import math

class Circle(GeometricObject):

def \_\_init\_\_(self, radius, color, filled):

super().\_\_init\_\_(color, filled)

self.\_\_radius = radius

def getRadius(self):

return self.\_\_radius

def setRadius(self, radius):

self.\_\_radius = radius

def getArea(self):

return (self.\_\_radius \* self.\_\_radius \* math.pi)

def getDiameter(self):

return (self.\_\_radius \* 2)

def getPerimeter(self):

return (self.\_\_radius \* 2 \* math.pi)

from GeometricObject import GeometricObject

class Rectangle(GeometricObject):

def \_\_init\_\_(self, width, height, color, filled):

super().\_\_init\_\_(color, filled)

self.\_\_width = width

self.\_\_height = height

def getWidth(self):

return self.\_\_width

def setWidth(self, width):

self.\_\_width = width

def getHeight(self):

return self.\_\_height

def setHeight(self, height):

self.\_\_height = height

def getArea(self):

return (self.\_\_height \* self.\_\_width)

def getPerimeter(self):

return ( (self.\_\_height + self.\_\_width) \* 2 )