#Albert George

=============================== RESTART: Shell ===============================

>>> from graphics import \*

>>> from math import \*

>>> #This program displays information about a triangle drawn by the user.

>>> def main():

p1,p2,p3=DrawTriangle()

A,B,C=GetLengths(p1,p2,p3)

area=GetArea(A,B,C)

Perimeter=GetPerimeter(A,B,C)

print(“Area=“, area,”Perimeter=“, Perimeter)

>>> def DrawTriangle():

win=GraphWin()

#Get three inputs from mouse

p1=win.getMouse()

p1.draw(win)

p2=win.getMouse()

p2.draw(win)

p3=win.getMouse()

p3.draw(win)

triangle=Polygon(p1,p2,p3)

triangle.setOutline("black")

triangle.draw(win)

return p1,p2,p3

>>> def GetLengths(p1,p2,p3):

X1=p1.getX()

Y1=p1.getY()

X2=p2.getX()

Y2=p2.getY()

X3=p3.getX()

Y3=p3.getY()

Dist1=sqrt((X1-X2)\*\*2+(Y1-Y2)\*\*2)

Dist2=sqrt((X2-X3)\*\*2+(Y2-Y3)\*\*2)

Dist3=sqrt((X3-X1)\*\*2+(Y3-Y1)\*\*2)

return Dist1,Dist2,Dist3

>>> def GetArea(A,B,C):

S=(A+B+C)/2

area=sqrt(S\*(S-A)\*(S-B)\*(S-C))

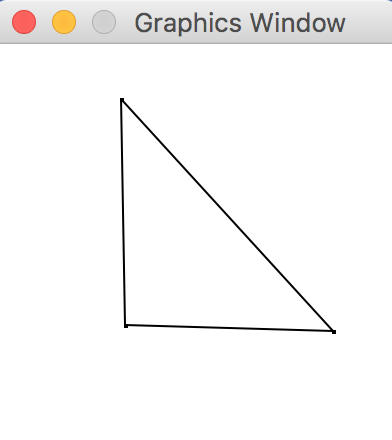
return area

>>> def GetPerimeter(A,B,C):

perimeter=A+B+C

return perimeter

>>> main()

Area= 5872.999999999999 Perimeter= 374.197840965908

=============================== RESTART: Shell ===============================

>>>#Page 55 number 2

>>> class Fraction:

def \_\_init\_\_(self,top,bottom):

self.num=top

self.den=bottom

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

return str(self.num) + "/" + str(self.den)

def \_\_add\_\_(self,otherfraction):

newnum = self.num\*otherfraction.den + self.den\*otherfraction.num

newden = self.den \* otherfraction.den

cd = self.gcd(newnum,newden)

return Fraction(newnum//cd,newden//cd)

def gcd(self,m,n):

while m%n!=0:

oldm=m

oldn=n

m=oldn

n=oldm%oldn

return n

def \_\_eq\_\_(self,otherFraction):

return (self.num==otherFraction.num and self.den==otherFraction.den)

>>> f1=Fraction(1,4)

>>> f2=Fraction(1,2)

>>> f3=f1+f2

>>> print(f3)

3/4

=============================== RESTART: Shell ===============================

>>>#Page 55 number 3

>>> class Fraction:

def \_\_init\_\_(self,top,bottom):

self.num=top

self.den=bottom

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

return str(self.num) + "/" + str(self.den)

def \_\_sub\_\_(self,otherfraction):

newnum = self.num\*otherfraction.den - self.den\*otherfraction.num

newden = self.den \* otherfraction.den

cd = self.gcd(newnum,newden)

return Fraction(newnum//cd,newden//cd)

def \_\_mul\_\_(a,b):

return Fraction(a.num \* b.num, a.den \* b.den)

def \_\_truediv\_\_(c,d):

return Fraction(c.num\*d.den,d.num\*c.den)

def gcd(self,m,n):

while m%n!=0:

oldm=m

oldn=n

m=oldn

n=oldm%oldn

return n

def \_\_eq\_\_(self,otherFraction):

return (self.num==otherFraction.num and self.den==otherFraction.den)

>>> f1=Fraction(1,4)

>>> f2=Fraction(1,2)

>>> f3=f1/f2

>>> print(f3)

2/4

>>> f3=f1-f2

>>> print(f3)

-1/4

>>> f3=f1\*f2

>>> print(f3)

1/8

>>>