```
"""

4.12.1. Exercise

Write a function called rhombus that draws a rhombus with a given side length
and a given interior angle. For example, here's a rhombus with side length 50
and an interior angle of 60 degrees.

"""

import turtle
import sys

class Rhombus():
    def __init__(self):

        #create the canvas and turtle
        self.t = turtle.Turtle()

        #This dictionary contains the action step and it's inverse function
        self.actions = {
            "left" : self.t.right,
            "right" : self.t.left,
            "forward" : self.t.back,
            "backward" : self.t.forward
        }

        #track all of the steps taken in a dictionary.
        self.steps = []

    def rhombus(self, length=100, angle=60, shift=0):
        """
        Takes in a width and height and draws a rectangle based on those dims.
        """
        #take the steps required to make a rectangle
        self.t.forward(length)
        self.steps.append(("forward", length))

        self.t.left(angle)
        self.steps.append(("left", angle))

        self.t.forward(length)
        self.steps.append(("forward", length))

        self.t.left(180-angle)
        self.steps.append(("left", 180-angle))

        self.t.forward(length)
        self.steps.append(("forward", length))

        self.t.left(angle)
        self.steps.append(("left", angle))

        self.t.forward(length)
        self.steps.append(("forward", length))

        if shift:
```

```
            self.t.left(shift)
            self.steps.append(("left", shift))

    def is_drawn(self):
        """
        Track whether a rectangle has been drawn, this will
        evaluate as true if self.steps > 0
        """
        return len(self.steps)

    def undo(self):
        """
        If the rhombus is drawn, iterate through the drawing steps and
        complete the inverse.
        """
        if self.is_drawn():
            self.steps.reverse()
            self.t.pencolor("white")
            for step in self.steps:
                #we first need to invert the direction we are traveling in
                action = self.actions[step[0]]
                action(int(step[1]))
            self.steps = []
            self.t.pencolor("black")
        else:
            print("Nothing to undo.")


if __name__ == '__main__':

    #Instantiate the rhombus
    rhomb = Rhombus()

    # Continuously prompt the user for rhombus inputs until the enter "exit"
    while 1:
        vars = input("Enter the side length and interior angle separated by a space, 'undo' to clea

        #just so you can read the input, I am copying it below:
        """
        Enter the side length and interior angle separated by a space, 'undo'
        to clear the existing rhombus, or 'exit' to close the window:
        """

        try:
            if len(vars.split(" ")) > 3:

                """
                allowing for special 'iterations' and 'shift' keywords that
                will repeat the draw sequence 'iterations' times with 'shift'
                angular offset each time
                """

                length, angle, iterations, shift = vars.split(" ")

                """
                In my code I assigned all of these in one line, but they
                get cut off in the PDF
```

```
            """
            length = int(length)
            angle = int(angle)
            iterations = int(iterations)
            shift = int(shift)

            for i in range(iterations):
                rhomb.rhombus(length, angle, shift)

        elif len(vars.split(" ")) == 2:
            length, angle = vars.split(" ")
            length, angle = int(length), int(angle)

            rhomb.rhombus(length, angle)

        elif len(vars.split(" ")) == 1:
            if vars == "undo":
                rhomb.undo()
            elif vars == "exit":
                break
    except:
        print("There was an error in the entry, try again.")

sys.exit()

#this keeps the canvas open
turtle.done()
```