```python
"""

4.11.1. Exercise

Write a function called rectangleangle that draws a rectangleangle with given
side lengths. For example, here's a rectangleangle that's 80 units wide and 40
units tall.

"""

import turtle
import sys

class Rectangle():
    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 rectangle(self, width=100, height=100, 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(width)
        self.steps.append(("forward", width))

        self.t.left(90)
        self.steps.append(("left", 90))

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

        self.t.left(90)
        self.steps.append(("left", 90))

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

        self.t.left(90)
        self.steps.append(("left", 90))

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

        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 rectangle 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 rectangle
    rect = Rectangle()

    # Continuously prompt the user for rectangle inputs until the enter "exit"
    while 1:
        vars = input("Enter the width and height separated by a space, undo to clear the existing r

        #just so you can read the input, I am copying it below:
        """
        Enter the width and height separated by a space, undo to clear the
        existing rectangle, or exit to close the window:
        """
        try:
            if len(vars.split(" ")) > 2:

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

                width, height, iterations, shift = vars.split(" ")

                """
                In my code I assigned all of these in one line, but they
                get cut off in the PDF
                """
                width = int(width)
                height = int(height)
                iterations = int(iterations)
                shift = int(shift)

                for i in range(iterations):
                    rect.rectangle(width, height, shift)

            elif len(vars.split(" ")) == 2:
                width, height = vars.split(" ")
                width, height = int(width), int(height)
                rect.rectangle(width, height)

            elif len(vars.split(" ")) == 1:
                if vars == "undo":
                    rect.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()
```