```python
"""

4.11.3. Exercise Part 2

Rewrite rectangle and rhombus to
use parallelogram.

"""
import turtle
from q3 import Parallelogram
from sys import exit

class Rectangle_V2():
    """
    Wrapper class that uses Parallelogram for rendering.
    """
    def __init__(self):
        self.renderer = Parallelogram()

    def rectangle(self, width=100, height=100, shift=0):
        self.renderer.parallelogram(l1=width, l2=height, angle=90, shift=shift)

    def undo(self):
        self.renderer.undo()

    def get_input(self):

        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 return to shape selection:
        """

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

            elif len(vars.split(" ")) == 1:
                if vars == "undo":
                    self.undo()
                elif vars == "exit":
                    #return true as an escape condition for the while loop
                    return 1
        except:
            print("There was an error in the entry, try again.")

    def __del__(self):
        """
        Adding a destructor to clean up instance in the main function.
        """
        print("Destructing Rectangle_V2 instance...")

class Rhombus_V2():
    """
    Wrapper class that uses Parallelogram for rendering.
    """
    def __init__(self):
        self.renderer = Parallelogram()

    def rhombus(self, length=100, angle=60, shift=0):
        self.renderer.parallelogram(
                                    l1=length,
                                    l2=length,
                                    angle=angle,
                                    shift=shift
                                    )

    def undo(self):
        self.renderer.undo()

    def get_input(self):
        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 return to shape selection:
        """

        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):
                    self.rhombus(length, angle, shift)

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

                self.rhombus(length, angle)

            elif len(vars.split(" ")) == 1:
                if vars == "undo":
                    self.undo()
                elif vars == "exit":
                    #return true as an escape condition for the while loop
                    return 1

        except:
            print("There was an error in the entry, try again.")

    def __del__(self):
        """
        Adding a destructor to clean up instance in the main function.
        """
        print("Destructing Rhombus_V2 instance...")

if __name__ == '__main__':

    # Continuously prompt the user for inputs until they enter "exit"
    while 1:

        shape_selection = input("Enter 1 for rectangle, 2 for rhombus, 3 for parallelogram, clear,

        #just so you can read the input, I am copying it below:
        """
        Enter 1 for rectangle, 2 for rhombus, 3 for parallelogram, clear,
        or exit:
        """

        if shape_selection == '1':
            shape = Rectangle_V2()
            #not a huge fan of nested while loops, but it works...
            while 1:
                if shape.get_input():
                    break
            #destruct the instance of the class
            del shape

        elif shape_selection == '2':
```

```
        shape = Rhombus_V2()
        #not a huge fan of nested while loops, but it works...
        while 1:
            if shape.get_input():
                break
        #destruct the instance of the class
        del shape

    elif shape_selection == '3':
        shape = Parallelogram()
        #not a huge fan of nested while loops, but it works...
        while 1:
            if shape.get_input(wrapper=True):
                break
        #destruct the instance of the class
        del shape

    elif shape_selection == "clear":
        """
        Instantiate the parallelogram to gain access to the clear function
        clear exists at the shape selection level because 'undo' exists
        at the drawing level
        """
        shape = Parallelogram()
        shape.clearscreen()
        del shape

    elif shape_selection == "exit":
        break
    else:
        print("Invalid input, try again...")

exit()

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