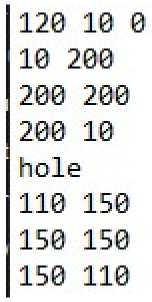
<u>Polygon Hatching</u>

<u>Guide on How to Use:</u>

The program prompts the user for the mode (file or input).

<u>File Mode:</u>

> if the user selects file mode, it displays all files in the directory and prompts the user to enter the name of the file that contains the polygon vertices.
>
> **File Format:**
>
> The first line of the file contains the hatch angle, spacing, and offset values, separated by spaces. The rest of the file contains the vertices of the polygon, with each line representing a single vertex. If there are holes in the polygon, the string 'hole' appears in the file, followed by the vertices of the hole.
>
> **Example:**



```
120 10 0
10 200
200 200
200 10
hole
110 150
150 150
150 110
```

<u>Input Mode:</u>

> If the user selects input mode, the program prompts the user to enter the number of sides for the polygon and the vertices of the polygon. It then prompts the user for the hatch angle, spacing, and offset values, as well as the number of holes in the polygon and the vertices of each hole.

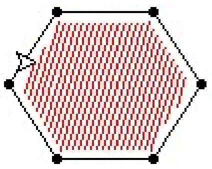
After the vertices and hatch pattern have been input, the program calculates the intersection points of the polygon by iterating over each pair of lines in the polygon and finding their intersection point using the line_intersection() function. It then uses the turtle module to draw hatch lines at the specified angle, spacing, and offset, starting at the intersection points and extending outward from the polygon.

If the user chooses to save the polygon vertices and hatch pattern to a file, the program prompts the user to enter a filename, and then writes the hatch angle, spacing, offset, and polygon vertices to the file in the same format as the input file. If there are holes in the polygon, the string 'hole' is written to the file, followed by the vertices of the hole.

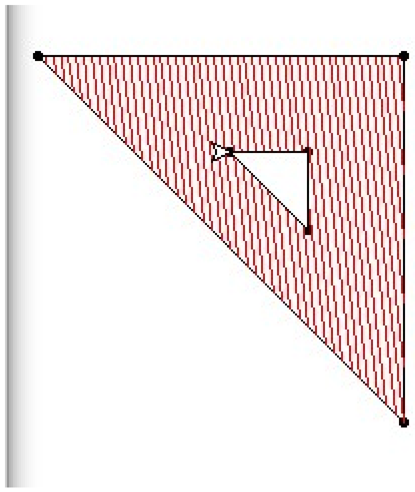
The while loop at the beginning of the program allows the user to create multiple polygons without having to restart the program. The loop runs as long as the run variable is True, which is set at the beginning of the program.

<u>OUTPUTS:</u>

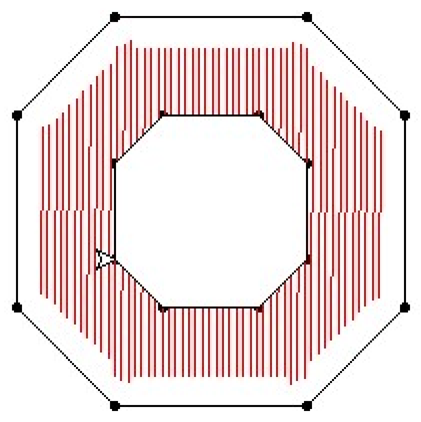
Example of Polygon with offset:



Example of Polygon with Hole:



Example of Polygon with Hole & Offset:



Explanation:

This program uses the turtle module in Python to create a GUI interface for the user to input a polygon or load a file that contains polygon vertices. The program then calculates the intersection points of the polygon and creates a hatch pattern at a specified angle, spacing, and offset. The program also prompts the user to save the polygon vertices and hatch pattern to a file if desired.

The line_intersection() function is used to calculate the intersection point of two lines. It takes two tuples that represent the endpoints of each line as arguments and returns the intersection point as a tuple. If the two lines are parallel or coincident, the function returns None.

The while loop at the beginning of the program allows the user to create multiple polygons without having to restart the program. The loop runs as long as the run variable is True, which is set at the beginning of the program.

The draw_hatchline function in the provided code is used to draw a hatch line across a polygon, which is used to fill the polygon with a pattern of parallel lines.

The function takes in four parameters:

start: The starting point of the hatch line, which is a tuple of x and y coordinates.

end: The ending point of the hatch line, which is also a tuple of x and y coordinates.

angle: The angle of the hatch line in degrees, which is a float value.

spacing: The distance between two adjacent hatch lines, which is a float value.

The function first calculates the length of the hatch line by finding the Euclidean distance between the starting and ending points using the math module. It then calculates the number of hatch lines that will fit in the given length by dividing the length by the spacing and rounding the result up to the nearest integer using the math.ceil() function.

Next, the function calculates the angle of the hatch line in radians by converting the input angle to radians using the math.radians() function. It then calculates the x and y components of the unit vector that points in the direction of the hatch line using the math.cos() and math.sin() functions, respectively.

The function then uses a for loop to draw each hatch line. Inside the loop, it calculates the x and y coordinates of the current hatch line's starting and ending points by adding the appropriate multiples of the unit vector to the starting and ending points. It then moves the turtle to the starting point using the turtle.goto() function and draws a line to the ending point using the turtle.pendown() and turtle.goto() functions.

Finally, the function moves the turtle back to the starting point using the turtle.goto() function and raises the turtle's pen using the turtle.penup() function to prepare for the next hatch line.

In summary, the draw_hatchline function takes in the starting and ending points, angle, and spacing of a hatch line, calculates the necessary parameters to draw the hatch lines, and uses a for loop to draw the desired number of hatch lines with the specified spacing and angle.

Key Strengths & Limitations:

Strengths:

- Accurate polygon and hole drawing.
- Input works with both manual inputs and a data file.
- Correct Spacing between hatchlines.

Limitations:

- Complex polygons with alot of twists and turns will sometimes fail the calculations for hatch lines.
- Calculations can also mess up at some steep angles due to increased complexity.