```python
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *

def display():
    glClear(GL_COLOR_BUFFER_BIT)
    render_Scene()
    glutSwapBuffers()

def render_Scene():
    # Draw X-Y axes
    glColor3f(1, 0, 0)  # Red color for axes
    glBegin(GL_LINES)
    # X-axis
    glVertex2f(-1, 0)
    glVertex2f(1, 0)
    # Y-axis
    glVertex2f(0, -1)
    glVertex2f(0, 1)
    glEnd()

    # Coordinates for the triangle
    triangle_coords = [(0.3, 0.2), (0.7, 0.5), (0.5, 0.7)]

    # Draw the triangle in all four quadrants
    glColor3f(1, 0, 0)  # Red color for triangles
    for reflection in [(1, 1), (-1, 1), (-1, -1), (1, -1)]:
        glBegin(GL_LINE_LOOP)
        for x, y in triangle_coords:
            glVertex2f(reflection[0] * x, reflection[1] * y)
        glEnd()

# Initialize GLUT
glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB)
glutInitWindowSize(500, 500)
glutCreateWindow("Triangle Reflections")
glutInitWindowPosition(50, 50)
glClearColor(0, 0, 0, 0)  # Black background
glutDisplayFunc(display)
glutMainLoop()
```

# Drawing Reflected Triangles with OpenGL in Python

This code utilizes OpenGL to render a red triangle reflected across all four quadrants on a black background.

**Display Callback Function (display())**

- `glClear(GL_COLOR_BUFFER_BIT)`: Clears the color buffer, erasing the previous frame.
- `render_Scene()`: Calls the function responsible for drawing the axes and reflected triangles.

- `glutSwapBuffers()`: Swaps the front and back buffers for smooth animation without flickering.

**Scene Render Function (render_Scene())**

- **Drawing Axes (X and Y):**
  - `glColor3f(1, 0, 0)`: Sets the drawing color to red for the axes.
  - `glBegin(GL_LINES)`: Starts drawing lines for the axes.
  - `glVertex2f(-1, 0)`: Specifies the starting vertex for the X-axis at (-1, 0).
  - `glVertex2f(1, 0)`: Specifies the ending vertex for the X-axis at (1, 0).
  - Similar lines are drawn for the Y-axis using `glVertex2f(0, -1)` and `glVertex2f(0, 1)`.
  - `glEnd()`: Ends drawing lines.
- **Triangle Coordinates:**
  - `triangle_coords = [(0.3, 0.2), (0.7, 0.5), (0.5, 0.7)]`: Defines a list containing the x and y coordinates for the three vertices of the triangle.
- **Drawing Reflected Triangles:**
  - `glColor3f(1, 0, 0)`: Sets the drawing color to red for the triangles.
  - A loop iterates through four reflections: `[(1, 1), (-1, 1), (-1, -1), (1, -1)]`.
    - Each reflection represents a quadrant: (1, 1) - upper right, (-1, 1) - upper left, (-1, -1) - lower left, (1, -1) - lower right.
    - `glBegin(GL_LINE_LOOP)`: Starts drawing a closed loop for each reflected triangle.
    - An inner loop iterates through the triangle coordinates:
      - `glVertex2f(reflection[0] * x, reflection[1] * y)`: Calculates and specifies the vertex position for the reflected triangle. The reflection values are applied to the original coordinates (`x` and `y`) to achieve the mirroring effect.
    - `glEnd()`: Ends drawing the current reflected triangle.

**GLUT Initialization and Configuration**

- `glutInit()`: Initializes the GLUT library.
- `glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB)`: Sets the display mode for double buffering and RGB colors.
- `glutInitWindowSize(500, 500)`: Defines the window size to be 500x500 pixels.
- `glutCreateWindow("Triangle Reflections")`: Creates the window with the specified title.
- `glutInitWindowPosition(50, 50)`: Sets the initial window position to (50, 50) on the screen.
- `glClearColor(0, 0, 0, 0)`: Sets the background color to black.
- `glutDisplayFunc(display)`: Assigns the `display()` function to be called for rendering the scene.
- `glutMainLoop()`: Enters the main event loop, continuously listening for events and calling the registered callback function (`display()`) to update the window contents.

**Summary**

This code demonstrates how to leverage OpenGL with Python to draw a colored triangle reflected across all four quadrants, offering a visual representation of geometric transformations.