How you structured the code?

MobiusStrip Class

Encapsulates all data and behaviour related to the Möbius strip.

1. __init__ Method

- Initializes strip parameters: radius R, width w, and resolution n.
- Generates parameter arrays u, v.
- Precomputes:
    - 3D mesh coordinates (x, y, z)
    - Surface area
    - Edge length

2. Private Helper Methods

Each computation is delegated to a dedicated method for clarity and reusability.

- calculate_coordinates()
  Generates the 3D mesh using parametric equations.
- calculate_surface_area()
  Computes surface area using dblquad over u and v ranges.
- calculate_edge_points()
  Extracts boundary points at one edge (v = -w/2).
- calculate_edge_length()
  Computes total edge length by summing distances between successive edge points.

3. Visualization

- plot() method uses matplotlib for a clear 3D surface plot with labeled axes and custom styling.

- **if __name__ == '__main__' Block**
    - Instantiates the MobiusStrip with sample parameters.
    - Prints all key properties.
    - Visualizes the strip.

How you Approximated Surface Area?

The surface area of the Möbius strip is approximated using **numerical integration** with scipy.integrate.dblquad over the parametric domain.

Given parametric equations:

$$x(u,v) = (R + v\cos(u/2))\cos(u)$$
$$y(u,v) = (R + v\cos(u/2))\sin(u)$$
$$z(u,v) = v\sin(u/2)$$
$$\text{where } u \in [0, 2\pi], v \in [-w/2, w/2]$$

Surface Area Formula (Double Integral):

$$A = \iint \left\| \frac{\partial \vec{r}}{\partial u} \times \frac{\partial \vec{r}}{\partial v} \right\| dv\, du$$

Any Challenges Faced?

- Parameter Interpretation
  Understanding the parameters correctly to handle geometry and math effectively.
- Surface Area Computation
  Ensuring the correctness of gradient direction during surface area calculation.
- 3D Plotting Precision
  Plotting the 3D model with appropriate colours, orientation, and resolution to produce an appealing and accurate visual output.