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

def get_modelview_matrix():
    """Retrieves the current modelview matrix from OpenGL."""
    modelview_matrix = (GLfloat * 16)()
    glGetFloatv(GL_MODELVIEW_MATRIX, modelview_matrix)
    return modelview_matrix

def print_matrix(matrix):
    """Prints a 4x4 matrix in a readable format."""
    for i in range(4):
        for j in range(4):
            print(f"{matrix[i * 4 + j]:.4f}", end=" ")
        print()  # Newline after each row

def display():
    """Display callback function."""
    glClear(GL_COLOR_BUFFER_BIT)

    # Print modelview matrix before transformation
    print("Modelview Matrix (Before Transformation):")
    modelview_matrix_before = get_modelview_matrix()
    print_matrix(modelview_matrix_before)

    # Render scene
    glColor3f(0, 1, 0)
    glTranslatef(0.05, -0.1, 0)
    glBegin(GL_TRIANGLES)
    glVertex3f(-0.8, -0.3, -0.1)
    glVertex3f(-0.3, 0.5, 0.0)
    glVertex3f(0.2, 0.3, 0.2)
    glEnd()

    # Print modelview matrix after transformation
    print("Modelview Matrix (After Transformation):")
    modelview_matrix_after = get_modelview_matrix()
    print_matrix(modelview_matrix_after)

    glutSwapBuffers()

def render_scene():
    """Scene render function (unused in this example)."""
    pass  # Empty implementation since display() handles rendering

# Initialize GLUT
glutInit()

# Initialize the window with double buffering and RGB colors
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB)
```

```
    # Set the window size to 500x500 pixels
    glutInitWindowSize(500, 500)

    # Create the window and give it a title
    glutCreateWindow("Drawing 1  a triangle with translation & CTM Printing")

    # Set the initial window position to (50, 50)
    glutInitWindowPosition(50, 50)

    # Define callbacks
    glutDisplayFunc(display)

    # Begin event loop
    glutMainLoop()
```

## Printing Modelview Matrix Before and After Transformation in OpenGL with Python

This code demonstrates how to retrieve and print the Modelview (CTM) matrix in OpenGL before and after applying a translation to a triangle.

**Modelview Matrix Functions:**

- `get_modelview_matrix()`:
    - This function retrieves the current Modelview matrix from OpenGL using `glGetFloatv(GL_MODELVIEW_MATRIX, modelview_matrix)`.
    - It returns a NumPy array-like object containing the 16 floating-point values representing the matrix.
- `print_matrix(matrix)`:
    - This function formats and prints a 4x4 matrix in a readable way.
    - It iterates through each row and column, printing each element with four decimal places.

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

- `glClear(GL_COLOR_BUFFER_BIT)`: Clears the color buffer, erasing the previous frame.
- **Printing Modelview Matrix (Before):**
    - It calls `get_modelview_matrix()` to retrieve the current matrix before any transformations are applied.
    - The retrieved matrix is stored in `modelview_matrix_before`.
    - `print_matrix(modelview_matrix_before)` is called to print the matrix in a formatted way.
- **Drawing Scene:**
    - `glColor3f(0, 1, 0)`: Sets the drawing color to green for the triangle.
    - `glTranslatef(0.05, -0.1, 0)`: Applies a translation transformation to the scene, shifting the triangle 0.05 units to the right, -0.1 units down, and keeping the z-coordinate unchanged.

  - `glBegin(GL_TRIANGLES)`: Starts drawing a triangle.
  - `glVertex3f` lines define the three vertices of the triangle at specific coordinates.
  - `glEnd()`: Ends drawing the triangle.

- **Printing Modelview Matrix (After):**

  - Similar to the before case, it retrieves the current Modelview matrix after the translation is applied and stores it in `modelview_matrix_after`.
  - `print_matrix(modelview_matrix_after)` is called to print the transformed matrix.

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

#### Unused Render Function (render_scene())

- This function is included but left empty (`pass`) as the rendering logic is handled directly within the `display()` function.

#### GLUT Initialization and Configuration

- Standard GLUT initialization steps are performed for window creation, display mode configuration, and callback assignment.

#### Summary

This code showcases how to access and print the Modelview matrix in OpenGL. By printing the matrix before and after a transformation (translation in this case), you can visualize how transformations affect the overall viewing transformation. Note that while `render_scene()` is included, it's not used in this specific example. You could potentially use it for more complex scene rendering logic in the future.