annular area.py

```python
import numpy as np
import matplotlib.pyplot as plt

def monte_carlo_annulus_area(N, plot=False):
    # Generate N random (x, y) points in [-2, 2] x [-2, 2]
    x = np.random.uniform(-2, 2, N)
    y = np.random.uniform(-2, 2, N)

    # Compute distance squared from origin
    r_squared = x**2 + y**2

    # Find number of points in annular region (1 ≤ r² ≤ 4)
    mask = (r_squared >= 1) & (r_squared <= 4)
    N_frac = np.sum(mask)

    # Area of square is 4 x 4 = 16
    estimated_area = (N_frac / N) * 16

    # Optional: Plotting
    if plot:
        plt.scatter(x[mask], y[mask], s=1, color='green', label='Inside Annulus')
        plt.scatter(x[~mask], y[~mask], s=1, color='red', alpha=0.5, label='Outside')
        plt.gca().set_aspect('equal')
        plt.title(f'N = {N} | Estimated Area ≈ {estimated_area:.4f}')
        plt.legend()
        plt.grid(True)
        plt.axvline(x=0, color='black', ls='--')
        plt.axhline(y=0, color='black', ls='--')
        plt.show()

    return estimated_area

# Part (a): Monte Carlo Estimations
for N in [100, 1000, 10000,100000]:
    area = monte_carlo_annulus_area(N, plot=True)
    print(f"Monte Carlo Estimated Area for N = {N}: {area:.4f}")

# Part (b): Exact Area using Numerical Integration (known formula)
exact_area = np.pi * (4 - 1)
print(f"\nExact Area using Integration: {exact_area:.4f}")

'''
OUTPUT:
Monte Carlo Estimated Area for N = 100: 9.7600
Monte Carlo Estimated Area for N = 1000: 9.4880
Monte Carlo Estimated Area for N = 10000: 9.3152
Monte Carlo Estimated Area for N = 100000: 9.4678

Exact Area using Integration: 9.4248
'''

```

```python
# Plotting the same bit in the same figure for better visualization
import numpy as np
import matplotlib.pyplot as plt

def generate_and_estimate(N):
    # Generate N random (x, y) points in [-2, 2] x [-2, 2]
    x = np.random.uniform(-2, 2, N)
    y = np.random.uniform(-2, 2, N)

    # Compute r² = x² + y²
    r_squared = x**2 + y**2

    # Mask: points inside the annulus (1 ≤ r² ≤ 4)
    mask = (r_squared >= 1) & (r_squared <= 4)
    N_frac = np.sum(mask)

    # Estimate area = (N_frac / N) * Area of square (16)
    estimated_area = (N_frac / N) * 16

    return x, y, mask, estimated_area

# Values of N
N_values = [100, 1000, 10000]

# Create subplots
fig, axes = plt.subplots(1, 3, figsize=(18, 6))
fig.suptitle('Monte Carlo Estimation of Annular Area', fontsize=16)

for i, N in enumerate(N_values):
    x, y, mask, est_area = generate_and_estimate(N)
    ax = axes[i]
    ax.scatter(x[~mask], y[~mask], s=1, color='red', alpha=0.5, label='Outside')
    ax.scatter(x[mask], y[mask], s=1, color='green', label='Inside Annulus')
    ax.set_title(f'N = {N}\nEstimated Area ≈ {est_area:.4f}')
    ax.set_aspect('equal')
    ax.set_xlim([-2, 2])
    ax.set_ylim([-2, 2])
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.grid(True)
    ax.legend()

plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()

# Part (b): Exact value using analytical method
exact_area = np.pi * (4 - 1)
print(f"Exact Area using Integration: {exact_area:.4f}")

```