Python Program for: “The First 25 Iterations”

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

# Set a lower resolution for faster output
width, height = 400, 400  # Lower resolution for faster iteration and grid summary

# Define the range of values for the complex plane
re_min, re_max = -2.0, 2.0
im_min, im_max = -2.0, 2.0

# Number of iterations for the fractal
max_iter = 128  # Fewer iterations for faster processing

# Define the grid of complex values
re, im = np.linspace(re_min, re_max, width), np.linspace(im_min, im_max, height)
X, Y = np.meshgrid(re, im)
C = X + 1j * Y  # The grid of complex numbers

# Initialize the figure
fig, ax = plt.subplots(figsize=(8, 8))

# Store frames for the grid summary
frames_list = []

# Function to compute the fractal with varying parameters
def compute_fractal(C, max_iter, param):
    Z = np.zeros(C.shape, dtype=np.complex128)
    fractal = np.zeros(C.shape, dtype=int)
    for n in range(max_iter):
        # Calculate theta using the angle of Z and apply the parameter as a scale
        Theta = np.angle(Z) + param * C  # Vary the scaling factor with the parameter

        # Euler's Identity applied to theta with a power term
        Z = np.exp(1j * np.pi * Theta) + Z**2 + C  # Combine Z^2 to create a fractal-like feedback

        # Prevent overflow by clipping Z's absolute values to a reasonable range
        Z = np.where(np.abs(Z) > 1e10, 1e10, Z)  # Clip large values to prevent overflow

        # Update the fractal based on escape time
        mask = np.abs(Z) < 100  # Adjust escape threshold
        fractal[mask] = n
```

```
    return fractal

# Generate and store the 25 unique frames
for i in range(25):
    param = 0.5 + 0.05 * i  # Vary the parameter for each frame
    fractal = compute_fractal(C, max_iter, param)
    frames_list.append(fractal)  # Store each unique frame

# Now create a 5x5 grid summary of the 25 unique frames
fig_grid, axes = plt.subplots(5, 5, figsize=(12, 12))  # 5 rows, 5 columns for exactly 25 frames

# Plot the 25 frames in the grid
for i, ax in enumerate(axes.flat):
    if i < len(frames_list):
        ax.imshow(frames_list[i], cmap='twilight_shifted', extent=(re_min, re_max, im_min, im_max))
        ax.set_title(f"{i + 1}")
        ax.axis('off')  # Hide axis for cleaner presentation

# Display the grid summary
plt.tight_layout()
plt.savefig("euler_identity_fractal_summary.png")
plt.show()
```