Sure! Let's solve this **step by step** using NumPy:

We have an **8×8 binary image** (values 0 or 1), and we want the **mean of each 4×4 segment**.

Step 1: Create 8×8 Binary Image

```
import numpy as np

# Create 8x8 binary image (random 0s and 1s)
image = np.random.randint(0, 2, (8, 8))
print("8x8 Binary Image:\n", image)
```

Explanation:

- `np.random.randint(0,2,(8,8))` → creates an 8×8 array with values 0 or 1.

Sample Output:

```
[[1 0 1 0 1 1 0 1]
 [0 1 1 0 1 0 1 0]
 [1 1 0 1 0 1 0 1]
 [0 1 0 1 1 0 1 0]
 [1 0 1 0 1 1 0 1]
 [0 1 0 1 0 1 1 0]
 [1 0 1 0 1 0 1 1]
 [0 1 0 1 1 0 1 0]]
```

Step 2: Divide into 4×4 Segments

We can divide the image into **4 quadrants of 4×4**:

```
# Top-left segment
seg1 = image[0:4, 0:4]

# Top-right segment
seg2 = image[0:4, 4:8]

# Bottom-left segment
seg3 = image[4:8, 0:4]

# Bottom-right segment
seg4 = image[4:8, 4:8]
```

Step 3: Compute Mean of Each Segment

```
mean1 = seg1.mean()
mean2 = seg2.mean()
mean3 = seg3.mean()
mean4 = seg4.mean()

print("Mean of Top-left 4x4:", mean1)
print("Mean of Top-right 4x4:", mean2)
print("Mean of Bottom-left 4x4:", mean3)
print("Mean of Bottom-right 4x4:", mean4)
```

Explanation:

- `.mean()` calculates the **average of all elements** in the 4×4 segment.
- Since values are 0 or 1, the mean represents the **fraction of pixels that are 1**.

Sample Output (based on random image above):

```
Mean of Top-left 4x4: 0.5625
Mean of Top-right 4x4: 0.5625
Mean of Bottom-left 4x4: 0.5
Mean of Bottom-right 4x4: 0.5
```

Step 4: Full Code

```
import numpy as np

# Create 8x8 binary image
image = np.random.randint(0, 2, (8, 8))
print("8x8 Binary Image:\n", image)

# Divide into 4x4 segments
seg1 = image[0:4, 0:4]
seg2 = image[0:4, 4:8]
seg3 = image[4:8, 0:4]
seg4 = image[4:8, 4:8]

# Compute mean of each segment
print("Mean of Top-left 4x4:", seg1.mean())
print("Mean of Top-right 4x4:", seg2.mean())
print("Mean of Bottom-left 4x4:", seg3.mean())
print("Mean of Bottom-right 4x4:", seg4.mean())
```

If you want, I can also show a **vectorized approach** that automatically computes the **mean of all 4×4 blocks without manually slicing**, which is cleaner for bigger images.

Do you want me to show that?