Adaptive thresholding is an image processing technique used to convert a grayscale image to a binary image, where the threshold value is calculated for smaller regions. This method differs from simple thresholding, where a global threshold value is used for the entire image. Here are key points about adaptive thresholding:

1. Local Thresholds: Unlike simple thresholding that uses a single global threshold for all pixels, adaptive thresholding determines thresholds for smaller regions, allowing for more precise segmentation in images with varying lighting conditions.

2. Types:
   - Mean Adaptive Thresholding: The threshold value is the mean of the neighborhood area minus a constant.
   - Gaussian Adaptive Thresholding: The threshold value is a weighted sum of the neighborhood values under a Gaussian window, minus a constant.

3. Usage: This method is particularly useful in scenarios where different parts of an image have different lighting conditions, making it difficult to segment the image with a global threshold.

4. OpenCV Implementation: In OpenCV, adaptive thresholding can be implemented using the `cv2.adaptiveThreshold` function.

5. Applications: It's widely used in document scanning and text recognition, especially when dealing with images that have uneven illumination.

In summary, adaptive thresholding provides a more flexible approach to image segmentation, particularly useful in complex lighting environments.

Your Python script demonstrates the use of both basic and adaptive thresholding techniques in OpenCV on an image, presumably a Sudoku puzzle. Let's walk through the key elements of your code:

1. Library Imports:
   - You import `numpy` and `cv2` (OpenCV), essential for image processing tasks.

2. Image Loading and Display:
   - The image ('sudoku.png') is loaded in grayscale mode (`0` as the second argument in `cv2.imread`).
   - The original image is displayed using `cv2.imshow`.

3. Basic Thresholding:
   - A simple binary thresholding is applied using `cv2.threshold`.
   - The threshold value is set to 70, and the maximum value to 255 (pixels above 70 are set to 255).
   - The result (`thresh_basic`) is displayed in a window titled "Basic Binary".

4. Adaptive Thresholding:
  - Adaptive thresholding is implemented using `cv2.adaptiveThreshold`.
  - The maximum value is set to 255.
  - `cv2.ADAPTIVE_THRESH_GAUSSIAN_C` indicates the use of a Gaussian weighted sum of the neighborhood values.
  - The window size for calculating the threshold value is 115, and the constant subtracted from the mean or weighted sum is 1.
  - The adaptive threshold result (`thres_adapt`) is displayed in a window titled "Adaptive Threshold".

5. Window Management:
  - `cv2.waitKey(0)` waits indefinitely for a key press before closing the windows.
  - `cv2.destroyAllWindows()` closes all OpenCV-created windows and releases the resources.

This script is a clear example of comparing basic and adaptive thresholding methods. The adaptive thresholding is particularly effective for images with varying illumination, as it adapts the threshold value for different areas of the image. In your case, this would help in accurately segmenting the Sudoku grid and numbers, especially if there are lighting variations across the image.