Image Processing (Elective I)
Lab 6: JPEG Compression (Guidelines)

1. Read the JPEG for a while:
   I. wiki: https://en.wikipedia.org/wiki/JPEG
   Ii. Documentation:
   https://www.hdm-stuttgart.de/~maucher/Python/MMCodecs/html/jpegUpToQuant.html

   JPEG Algorithm Summarized:

   a. Convert RGB to YCbCr: Separate luminance (Y) from chrominance (Cb, Cr) since human vision is more sensitive to brightness.
   b. Chroma Subsampling (4:2:0): Reduce chrominance resolution by averaging blocks of pixels.
   c. Divide into 8x8 Blocks: Process the image in small blocks for DCT computation.
   d. Apply Discrete Cosine Transform (DCT): Convert spatial data into frequency components.
   e. Quantization: Reduce high-frequency components (lossy step) using a quantization matrix.
   f. Zigzag Scanning: Reorder coefficients for efficient entropy encoding.
   g. Run-Length & Huffman Encoding: Compress data further (not always visualized).

2. Set up environment by installing the following libraries:
   a. numpy, cv2, scikit-image, matplotlib, skimage, etc…
   b. Scipy.fftpack.dct, Scipy.fftpack.idct

3. Load and display the original image. Use the following image:
   img_url =
   'https://upload.wikimedia.org/wikipedia/en/7/7d/Lenna_%28test_image%29.png'

4. Convert RGB to YCbCr: Use opencv to split the image into Luminance (Y), Blue chrominance (Cb), Red Chrominance (Cr). Then visualize each channel.

5. Chroma Subsampling (4:2:0)
   a. Reduce resolution of Cb and Cr by half (averaging 2x2 blocks)
   b. Visualize the subsampled channels

6. Divide the image into 8x8 blocks
   a. Split the Y channel into 8x8 blocks for DCT processing
   b. Display the first block

7. Apply Discrete Cosine Transform (DCT): Compute DCT for an 8x8 block and visualize coefficients.

8. Quantization (Lossy Compression): Divide the DCT coefficients by a quantization matrix (Q) and round values:

```
Q = np.array([[16, 11, 10, 16, 24, 40, 51, 61],
               [12, 12, 14, 19, 26, 58, 60, 55],
              [14, 13, 16, 24, 40, 57, 69, 56],
              [14, 17, 22, 29, 51, 87, 80, 62],
              [18, 22, 37, 56, 68, 109, 103, 77],
              [24, 35, 55, 64, 81, 104, 113, 92],
             [49, 64, 78, 87, 103, 121, 120, 101],
             [72, 92, 95, 98, 112, 100, 103, 99]])
```

9. Dequantization and Reconstruction: Multiply the quantized value by Q and apply inverse DCT (IDCT). Use the following hints:
    a. Dequantized = quantized * Q
    b. Reconstructed = idct(idct(dequantized.T, norm = ‘ortho’).T, norm = ’ortho’) +128

10. Full image compression: Implement a function named “jpeg_compress(img, Q, subsample=True)” that compresses the entire image.

11. PSNR Calculation: Create a function named “psnr(original, compressed)” that calculates and returns the PSNR value.

12. Bonus Tasks:
    a. Experiment with different Q matrices (higher values for more compression)
    b. Disable Chroma Subsampling and compare results.
    Hints: Set subsample = False in jpeg_compress()
    c. Compare with OpenCV’s JPEG Compression.
    Hints: Use cv2.imencode() and cv2.imdecode()