Pract 5 : Write a Python script to perform image filtering in frequency domain.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
def display_image(img, title="Image"):
    """Display an image using matplotlib."""
    plt.figure(figsize=(6, 6))
    plt.imshow(img, cmap='gray')
    plt.title(title)
    plt.axis('off')
    plt.show()

def apply_frequency_filter(image, filter_type, cutoff):
    """Apply a frequency domain filter to an image.
Args:
        image (numpy.ndarray): Grayscale input image.
        filter_type (str): Type of filter ('low-pass' or 'high-pass').
        cutoff (int): Cutoff frequency for the filter.
Returns:
        numpy.ndarray: Filtered image.
    ""# Perform Fourier Transform
    dft = np.fft.fft2(image)
    dft_shift = np.fft.fftshift(dft)

    # Create a mask for the filter
    rows, cols = image.shape
    crow, ccol = rows // 2, cols // 2
    mask = np.zeros((rows, cols), np.uint8)

 if filter_type == 'low-pass':
    # Create a circular low-pass filter
        cv2.circle(mask, (ccol, crow), cutoff, 1, -1)
    elif filter_type == 'high-pass':
        # Create a circular high-pass filter
        mask[:, :] = 1
        cv2.circle(mask, (ccol, crow), cutoff, 0, -1)
```

```
    else:
        raise ValueError("Invalid filter type. Use 'low-pass' or 'high-pass'.")

    # Apply the mask in the frequency domain
    filtered_dft_shift = dft_shift * mask

    # Perform inverse Fourier Transform
    dft_shift_inverse = np.fft.ifftshift(filtered_dft_shift)
    filtered_image = np.fft.ifft2(dft_shift_inverse)
    filtered_image = np.abs(filtered_image)
    return filtered_image

def main():
    # Load a grayscale image
    image_path = input("Enter the path to the image: ")
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

    if image is None:
        print("Error: Could not load the image.")
        return
display_image(image, "Original Image")

    # Apply low-pass filter
    cutoff = int(input("Enter the cutoff frequency for low-pass filter: "))
    low_pass_filtered = apply_frequency_filter(image, 'low-pass', cutoff)
    display_image(low_pass_filtered, "Low-Pass Filtered Image")

    # Apply high-pass filter
    cutoff = int(input("Enter the cutoff frequency for high-pass filter: "))
    high_pass_filtered = apply_frequency_filter(image, 'high-pass', cutoff)
    display_image(high_pass_filtered, "High-Pass Filtered Image")

if __name__ == "__main__":
    main()
```