JSS Science & Technological University

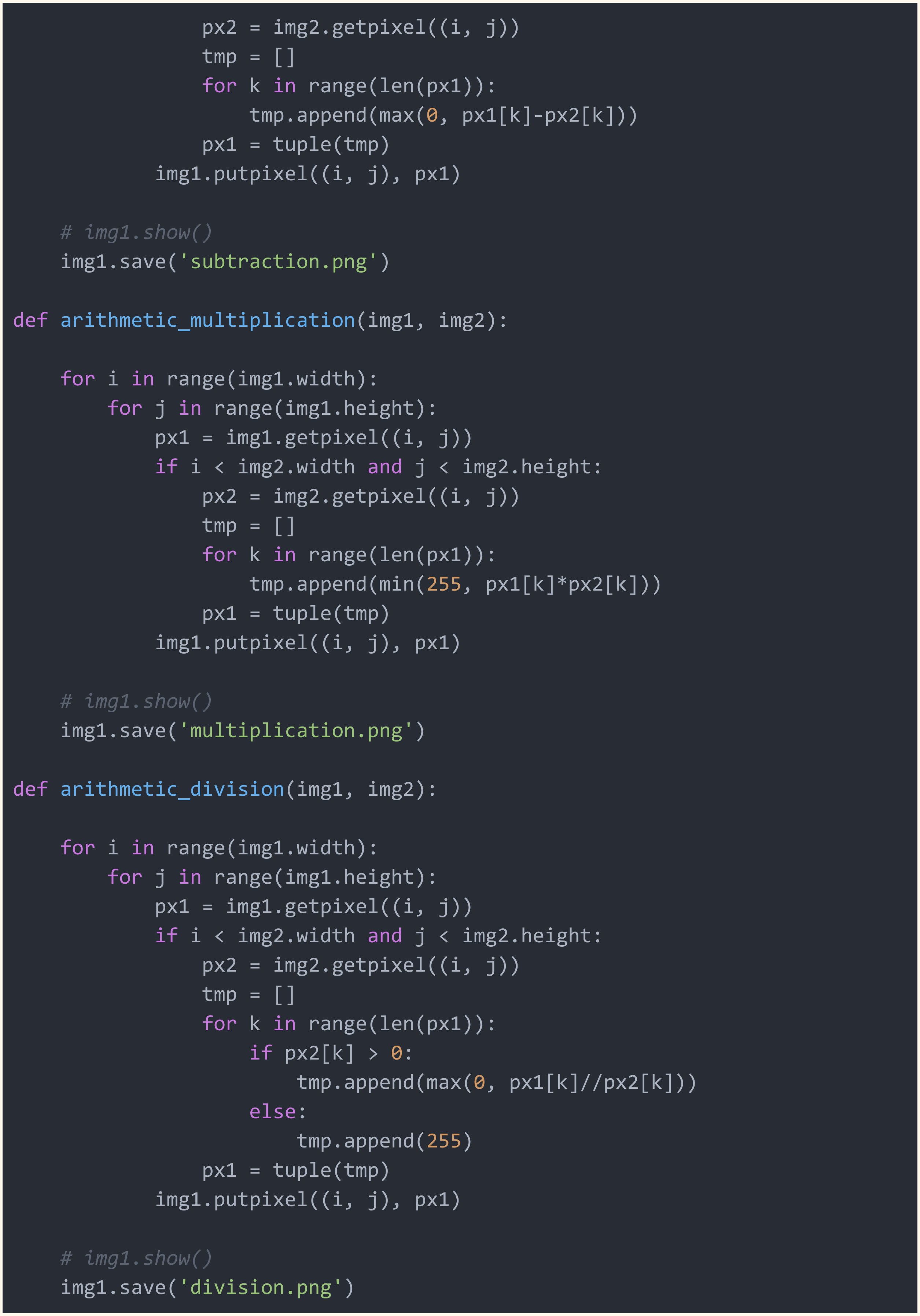
Professor: Shruthi N M

DIGITAL IMAGE PROCESSING LAB

Y PRAHASITH

Program to enhance image using image arithmetic and logical operations

```
from PIL import Image

def arithmetic_addition(img1, img2):

    for i in range(img1.width):
        for j in range(img1.height):
            px1 = img1.getpixel((i, j))
            if i < img2.width and j < img2.height:
                px2 = img2.getpixel((i, j))
                tmp = []
                for k in range(len(px1)):
                    tmp.append(min(255, px1[k]+px2[k]))
                px1 = tuple(tmp)
            img1.putpixel((i, j), px1)

    # img1.show()
    img1.save('addition.png')

def arithmetic_subtraction(img1, img2):

    for i in range(img1.width):
        for j in range(img1.height):
            px1 = img1.getpixel((i, j))
            if i < img2.width and j < img2.height:
```



```
                px2 = img2.getpixel((i, j))
                tmp = []
                for k in range(len(px1)):
                    tmp.append(max(0, px1[k]-px2[k]))
                px1 = tuple(tmp)
            img1.putpixel((i, j), px1)

    # img1.show()
    img1.save('subtraction.png')

def arithmetic_multiplication(img1, img2):

    for i in range(img1.width):
        for j in range(img1.height):
            px1 = img1.getpixel((i, j))
            if i < img2.width and j < img2.height:
                px2 = img2.getpixel((i, j))
                tmp = []
                for k in range(len(px1)):
                    tmp.append(min(255, px1[k]*px2[k]))
                px1 = tuple(tmp)
            img1.putpixel((i, j), px1)

    # img1.show()
    img1.save('multiplication.png')

def arithmetic_division(img1, img2):

    for i in range(img1.width):
        for j in range(img1.height):
            px1 = img1.getpixel((i, j))
            if i < img2.width and j < img2.height:
                px2 = img2.getpixel((i, j))
                tmp = []
                for k in range(len(px1)):
                    if px2[k] > 0:
                        tmp.append(max(0, px1[k]//px2[k]))
                    else:
                        tmp.append(255)
                px1 = tuple(tmp)
            img1.putpixel((i, j), px1)

    # img1.show()
    img1.save('division.png')
```

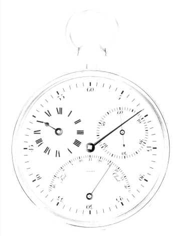
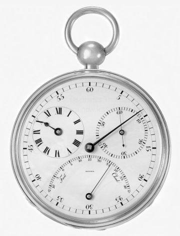
```
def logical_and(img1, img2):

    for i in range(img1.width):
        for j in range(img1.height):
            px1 = img1.getpixel((i, j))
            if i < img2.width and j < img2.height:
                px2 = img2.getpixel((i, j))
                tmp = []
                for k in range(len(px1)):
                    tmp.append(px1[k]&px2[k])
                px1 = tuple(tmp)
            img1.putpixel((i, j), px1)

    # img1.show()
    img1.save('and.png')

def logical_or(img1, img2):

    for i in range(img1.width):
        for j in range(img1.height):
            px1 = img1.getpixel((i, j))
            if i < img2.width and j < img2.height:
                px2 = img2.getpixel((i, j))
                tmp = []
                for k in range(len(px1)):
                    tmp.append(px1[k]|px2[k])
                px1 = tuple(tmp)
            img1.putpixel((i, j), px1)

    # img1.show()
    img1.save('or.png')

def logical_xor(img1, img2):

    for i in range(img1.width):
        for j in range(img1.height):
            px1 = img1.getpixel((i, j))
            if i < img2.width and j < img2.height:
                px2 = img2.getpixel((i, j))
                tmp = []
                for k in range(len(px1)):
                    tmp.append(px1[k]^px2[k])
                px1 = tuple(tmp)
            img1.putpixel((i, j), px1)

    # img1.show()
    img1.save('xor.png')

if __name__ == '__main__':

    print('1. Addition\n2. Subtraction\n3. Multiplication\n4. Division\n5. And\n6. Or\n7. Xor')
    choice = int(input('Pick required operation: '))

    img_name1 = input('Image 1 filename: ')
    img1 = Image.open(img_name1).convert('RGB')

    img_name2 = input('Image 2 filename: ')
    img2 = Image.open(img_name2).convert('RGB')

    if choice == 1:
        arithmetic_addition(img1, img2)

    elif choice == 2:
        arithmetic_subtraction(img1, img2)

    elif choice == 3:
        arithmetic_multiplication(img1, img2)

    elif choice == 4:
        arithmetic_division(img1, img2)

    elif choice == 5:
        logical_and(img1, img2)

    elif choice == 6:
        logical_or(img1, img2)

    elif choice == 7:
        logical_xor(img1, img2)

    else:
        print('Invalid choice!')
```

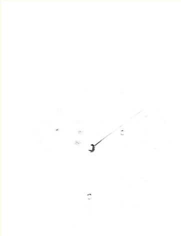
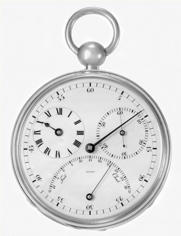
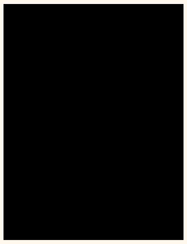
Input Image

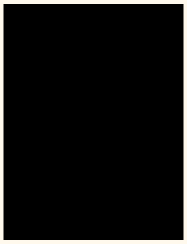
We need to give 2 input images of the same dimensions, but now for the sake of demonstration I am considering two same pictures as the inputs.

Input 1 Input 2

Output Images

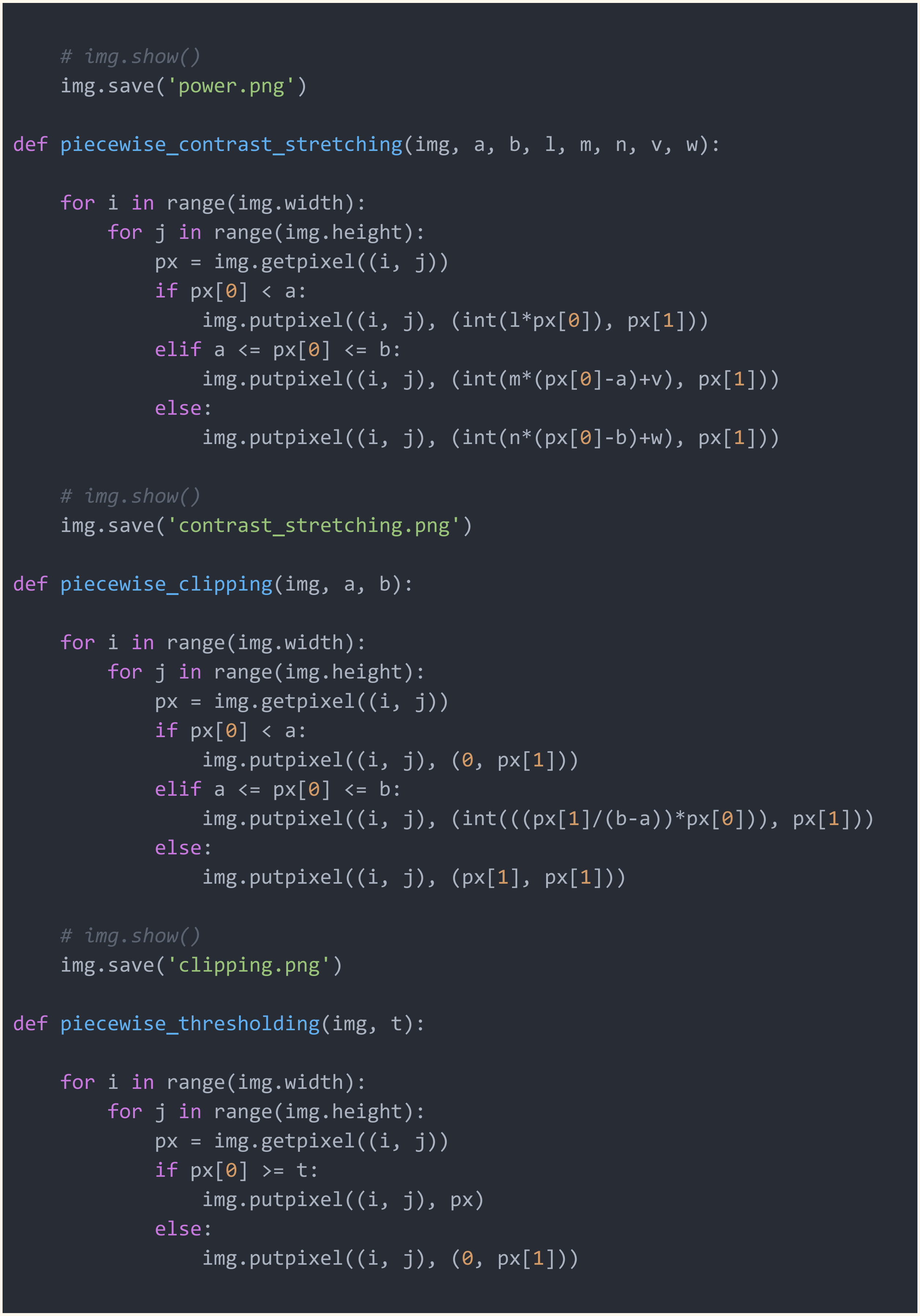
Addition  And  Division

Multiplication  Or  Subtraction 

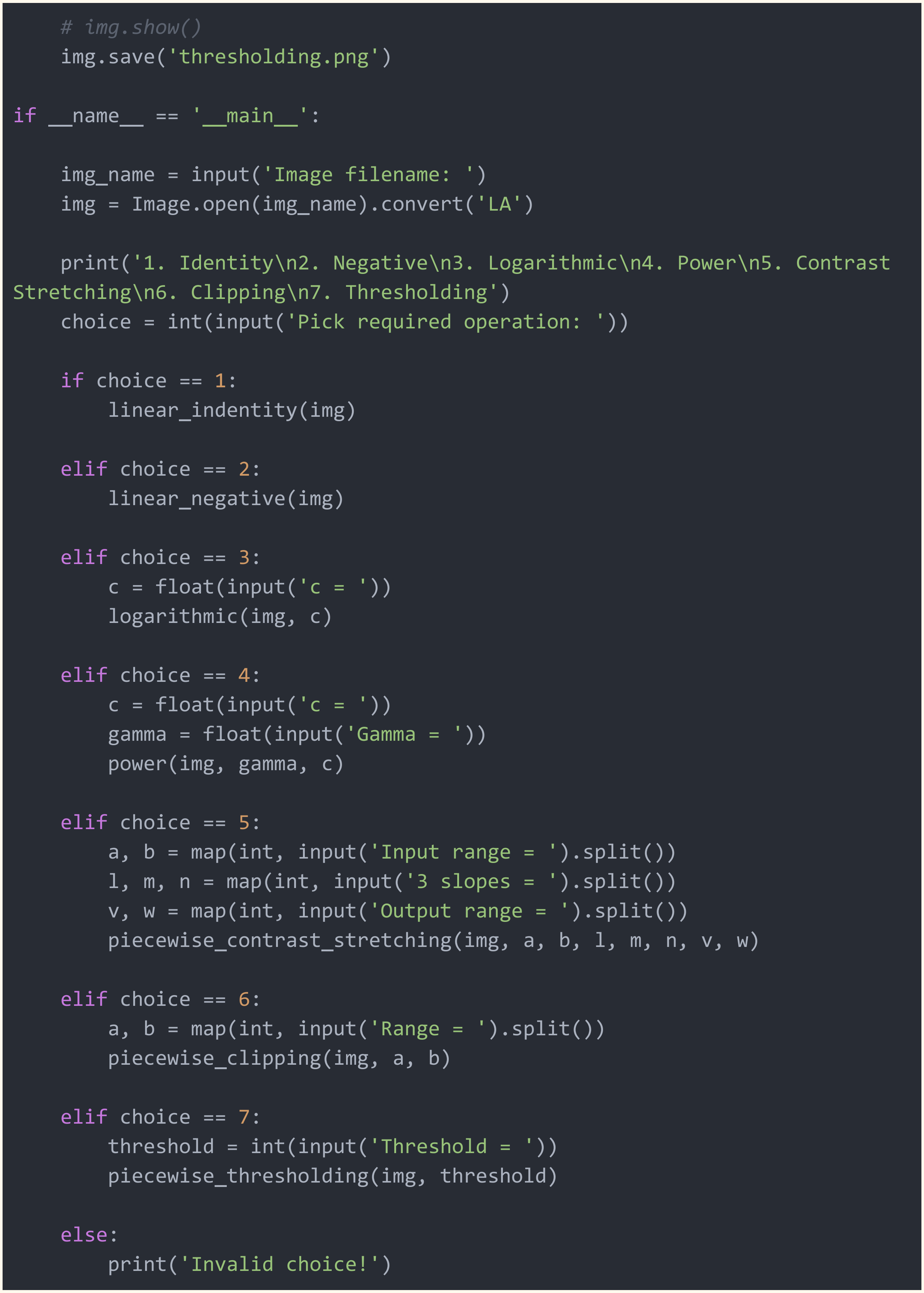
XOR 

Program for an image enhancement using pixel operations

```python
from PIL import Image
import math

def linear_indentity(img):

    for i in range(img.width):
        for j in range(img.height):
            px = img.getpixel((i, j))
            img.putpixel((i, j), px)

    # img.show()
    img.save('identity.png')

def linear_negative(img):

    for i in range(img.width):
        for j in range(img.height):
            px = img.getpixel((i, j))
            img.putpixel((i, j), (px[1]-px[0], px[1]))

    # img.show()
    img.save('negative.png')

def logarithmic(img, c=31.875):

    for i in range(img.width):
        for j in range(img.height):
            px = img.getpixel((i, j))
            img.putpixel((i, j), (int(c * math.log(px[0]+1, 2)), px[1]))

    # img.show()
    img.save('logarithmic.png')

def power(img, gamma, c=31.875):

    for i in range(img.width):
        for j in range(img.height):
            px = img.getpixel((i, j))
            img.putpixel((i, j), (int(c * (px[0] ** (1/gamma))), px[1]))
```



```python
    # img.show()
    img.save('power.png')

def piecewise_contrast_stretching(img, a, b, l, m, n, v, w):

    for i in range(img.width):
        for j in range(img.height):
            px = img.getpixel((i, j))
            if px[0] < a:
                img.putpixel((i, j), (int(l*px[0]), px[1]))
            elif a <= px[0] <= b:
                img.putpixel((i, j), (int(m*(px[0]-a)+v), px[1]))
            else:
                img.putpixel((i, j), (int(n*(px[0]-b)+w), px[1]))

    # img.show()
    img.save('contrast_stretching.png')

def piecewise_clipping(img, a, b):

    for i in range(img.width):
        for j in range(img.height):
            px = img.getpixel((i, j))
            if px[0] < a:
                img.putpixel((i, j), (0, px[1]))
            elif a <= px[0] <= b:
                img.putpixel((i, j), (int(((px[1]/(b-a))*px[0])), px[1]))
            else:
                img.putpixel((i, j), (px[1], px[1]))

    # img.show()
    img.save('clipping.png')

def piecewise_thresholding(img, t):

    for i in range(img.width):
        for j in range(img.height):
            px = img.getpixel((i, j))
            if px[0] >= t:
                img.putpixel((i, j), px)
            else:
                img.putpixel((i, j), (0, px[1]))
```

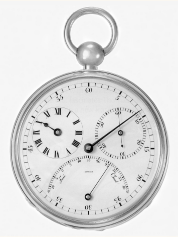


```
    # img.show()
    img.save('thresholding.png')

if __name__ == '__main__':

    img_name = input('Image filename: ')
    img = Image.open(img_name).convert('LA')

    print('1. Identity\n2. Negative\n3. Logarithmic\n4. Power\n5. Contrast Stretching\n6. Clipping\n7. Thresholding')
    choice = int(input('Pick required operation: '))

    if choice == 1:
        linear_indentity(img)

    elif choice == 2:
        linear_negative(img)

    elif choice == 3:
        c = float(input('c = '))
        logarithmic(img, c)

    elif choice == 4:
        c = float(input('c = '))
        gamma = float(input('Gamma = '))
        power(img, gamma, c)

    elif choice == 5:
        a, b = map(int, input('Input range = ').split())
        l, m, n = map(int, input('3 slopes = ').split())
        v, w = map(int, input('Output range = ').split())
        piecewise_contrast_stretching(img, a, b, l, m, n, v, w)

    elif choice == 6:
        a, b = map(int, input('Range = ').split())
        piecewise_clipping(img, a, b)

    elif choice == 7:
        threshold = int(input('Threshold = '))
        piecewise_thresholding(img, threshold)

    else:
        print('Invalid choice!')
```

Input Image

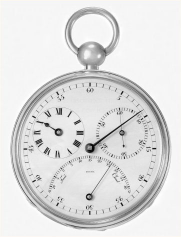
We have considered the same image for all operations . For the processes we have consider, c = 31.875 , Gamma = 0.85 , input range = 0 - 255 , output range = 0 - 255 , 3 slopes as {0,-1,2} and threshold = 8

Input

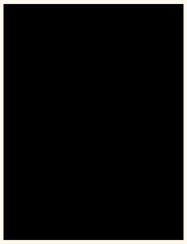


Output Images

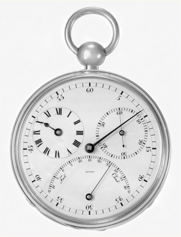
Clipping



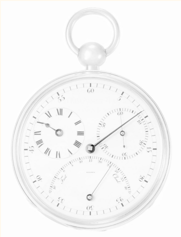
Contrast Stretching



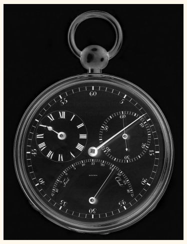
Identity



Logarithmic

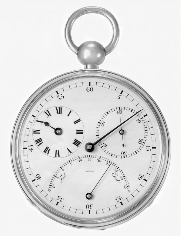


Negative



Power

Thresholding



Program for gray level slicing with and without background

```
from PIL import Image

def gray_level_slicing_with_background(img, a, b):

    for i in range(img.width):
        for j in range(img.height):
            px = img.getpixel((i, j))
            if a <= px[0] <= b:
                img.putpixel((i, j), (px[1], px[1]))

    #img.show()
    img.save('gray_level_slicing_with_background.png')

def gray_level_slicing_without_background(img, a, b):

    for i in range(img.width):
        for j in range(img.height):
            px = img.getpixel((i, j))
            if a <= px[0] <= b:
                img.putpixel((i, j), (px[1], px[1]))
            else:
                img.putpixel((i, j), (0, px[1]))

    #img.show()
    img.save('gray_level_slicing_without_background.png')

if __name__ == '__main__':
    a = int(input('Lower value: '))
    b = int(input('Upper value: '))

    img_name = input('Image filename: ')
    img = Image.open(img_name).convert('LA')

    print('1. Gray level slicing with background\n2. Gray level slicing
without background')

    choice = int(input('Pick required operation: '))

    if choice == 1:
        gray_level_slicing_with_background(img, a, b)
```

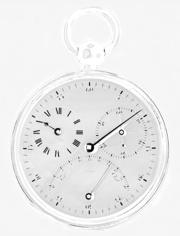
```
elif choice == 2:
    gray_level_slicing_without_background(img, a, b)

else:
    print('Invalid choice!')
```

Input Image

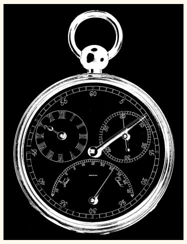
We have considered the same image for all operations . For the processes we have consider, Lower value = 80 and Upper value = 202.

Input

Output Images



Gray Level Slicing with Background



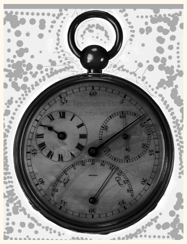
Gray Level Slicing without Background

Program for image enhancement using histogram equalization

```
from PIL import Image

def histogram_equalization(img):

    pixels = img.width * img.height
    grey_levels = img.getpixel((0, 0))[1] + 1

    arr = [0] * grey_levels

    for i in range(img.width):
        for j in range(img.height):
            px = img.getpixel((i, j))
            arr[px[0]] += 1

    for i in range(grey_levels):
        arr[i] = arr[i] / pixels

    for i in range(1, grey_levels):
        arr[i] += arr[i-1]

    for i in range(grey_levels):
        arr[i] *= (grey_levels - 1)

    for i in range(grey_levels):
        arr[i] = round(arr[i])

    # print(arr)

    for i in range(img.width):
        for j in range(img.height):
            px = img.getpixel((i, j))
            img.putpixel((i, j), (arr[px[0]], px[1]))

    # img.show()
    img.save('histogram_equalization.png')

if __name__ == '__main__':

    img_name = input('Image filename: ')
    img = Image.open(img_name).convert('LA')

    histogram_equalization(img)
```

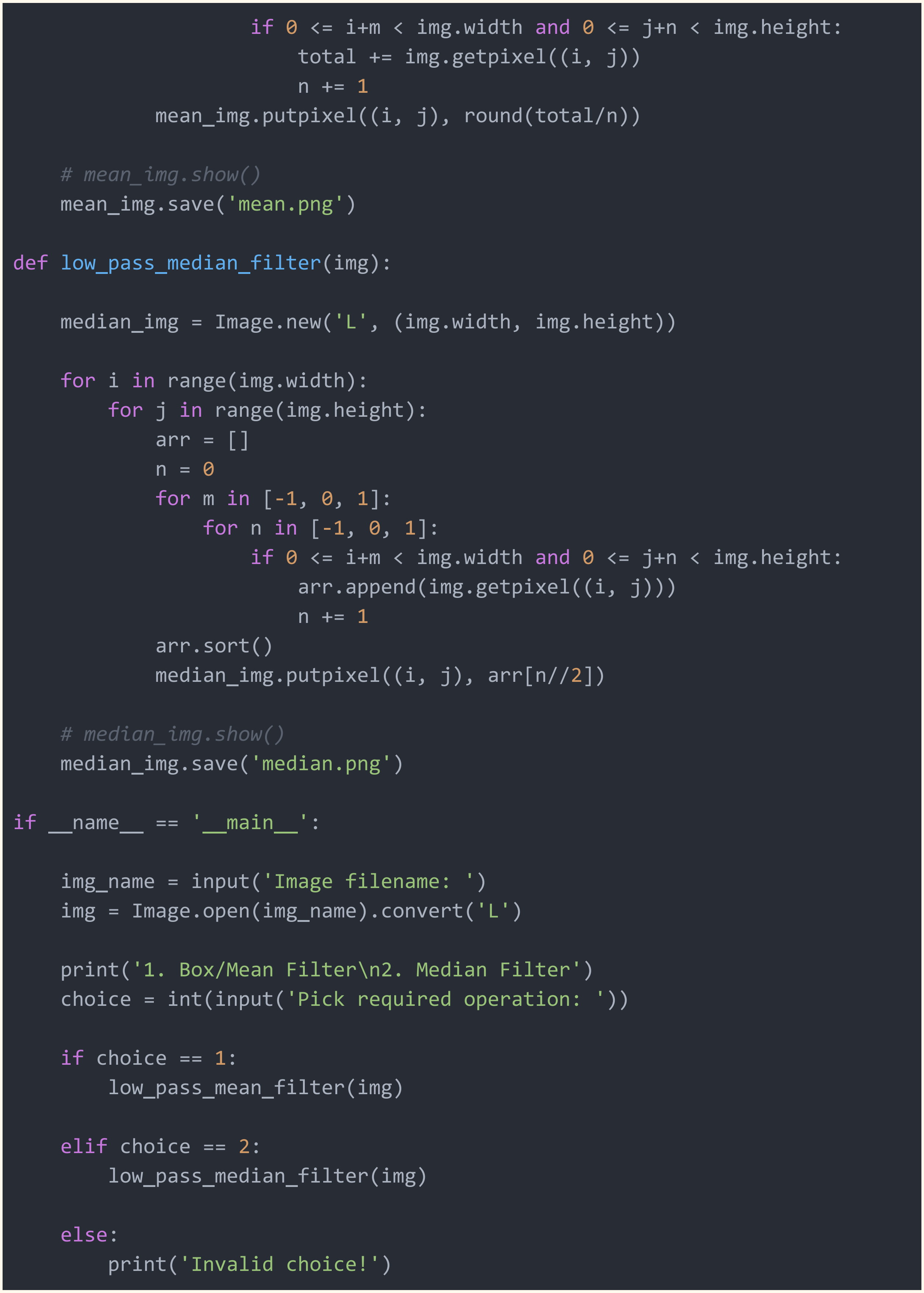
Input Image

Output Images



Program to filter an image using averaging low pass filter in spatial domain and median filter

```
from PIL import Image

def low_pass_mean_filter(img):

    mean_img = Image.new('L', (img.width, img.height))

    for i in range(img.width):
        for j in range(img.height):
            total = 0
            n = 0
            for m in [-1, 0, 1]:
                for n in [-1, 0, 1]:
```

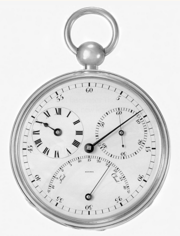


```python
                    if 0 <= i+m < img.width and 0 <= j+n < img.height:
                        total += img.getpixel((i, j))
                        n += 1
            mean_img.putpixel((i, j), round(total/n))

    # mean_img.show()
    mean_img.save('mean.png')

def low_pass_median_filter(img):

    median_img = Image.new('L', (img.width, img.height))

    for i in range(img.width):
        for j in range(img.height):
            arr = []
            n = 0
            for m in [-1, 0, 1]:
                for n in [-1, 0, 1]:
                    if 0 <= i+m < img.width and 0 <= j+n < img.height:
                        arr.append(img.getpixel((i, j)))
                        n += 1
            arr.sort()
            median_img.putpixel((i, j), arr[n//2])

    # median_img.show()
    median_img.save('median.png')

if __name__ == '__main__':

    img_name = input('Image filename: ')
    img = Image.open(img_name).convert('L')

    print('1. Box/Mean Filter\n2. Median Filter')
    choice = int(input('Pick required operation: '))

    if choice == 1:
        low_pass_mean_filter(img)

    elif choice == 2:
        low_pass_median_filter(img)

    else:
        print('Invalid choice!')
```

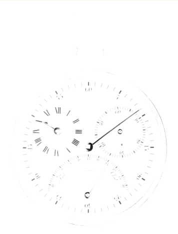
Input Image

Output Images

Median



Mean

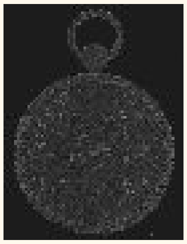


Program to sharpen an image using 2-D laplacian high pass filter in spatial domain.

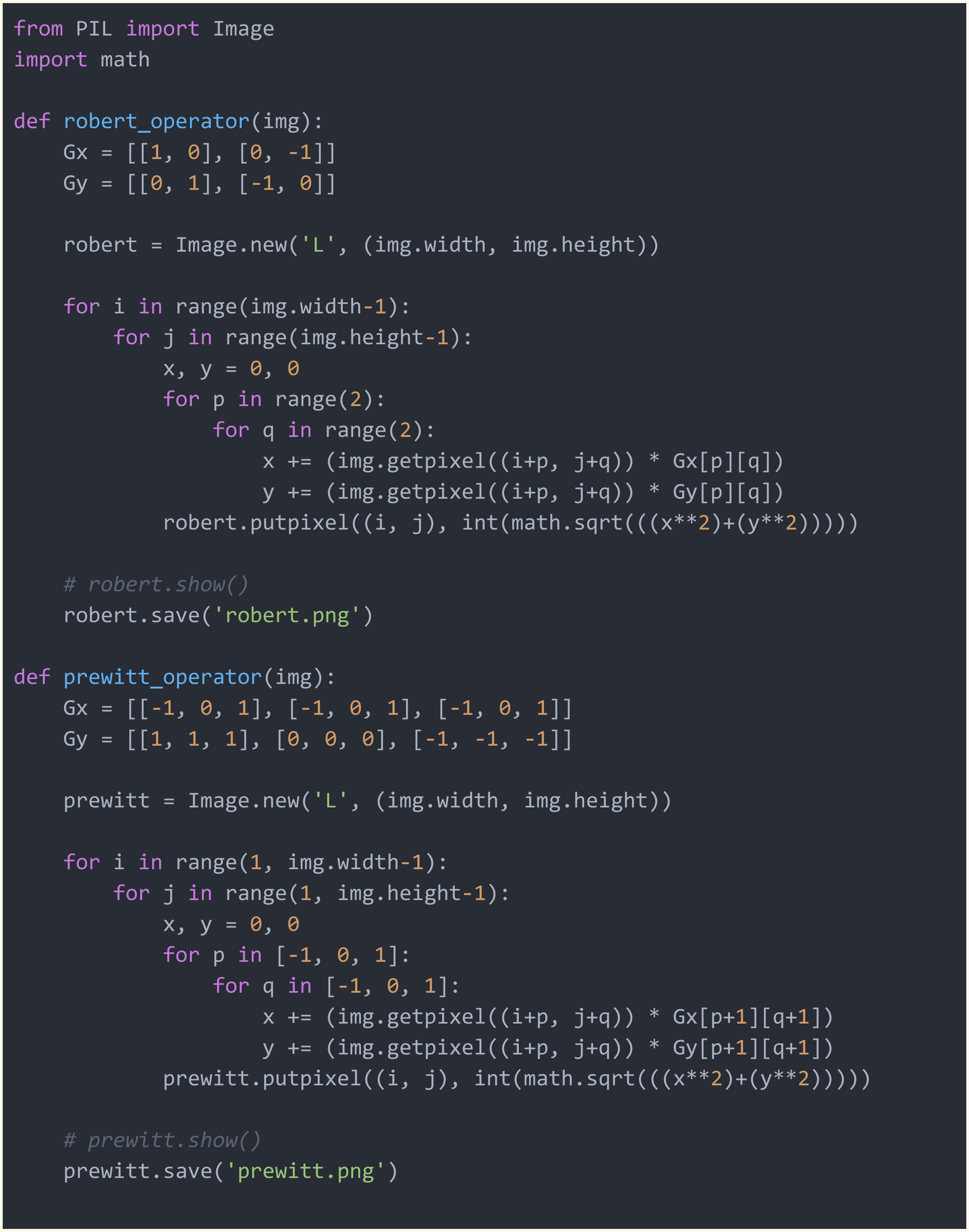
```
from PIL import Image

def high_pass_laplacian_filter(img):

    laplacian_img = Image.new('L', (img.width, img.height))

    for i in range(img.width):
        for j in range(img.height):
            n = 0
            total = 0
            if i-1 >= 0:
                total += img.getpixel((i-1, j))
                n += 1
            if i+1 < img.width:
                total += img.getpixel((i+1, j))
                n += 1
            if j-1 >= 0:
                total += img.getpixel((i, j-1))
                n += 1
            if j+1 < img.height:
                total += img.getpixel((i, j+1))
                n += 1
            px = max(0, total - (n * img.getpixel((i, j))))
            laplacian_img.putpixel((i, j), px)

    # laplacian_img.show()
    laplacian_img.save('laplacian.png')

if __name__ == '__main__':

    img_name = input('Image filename: ')
    img = Image.open(img_name).convert('L')

    high_pass_laplacian_filter(img)
```

Input Image

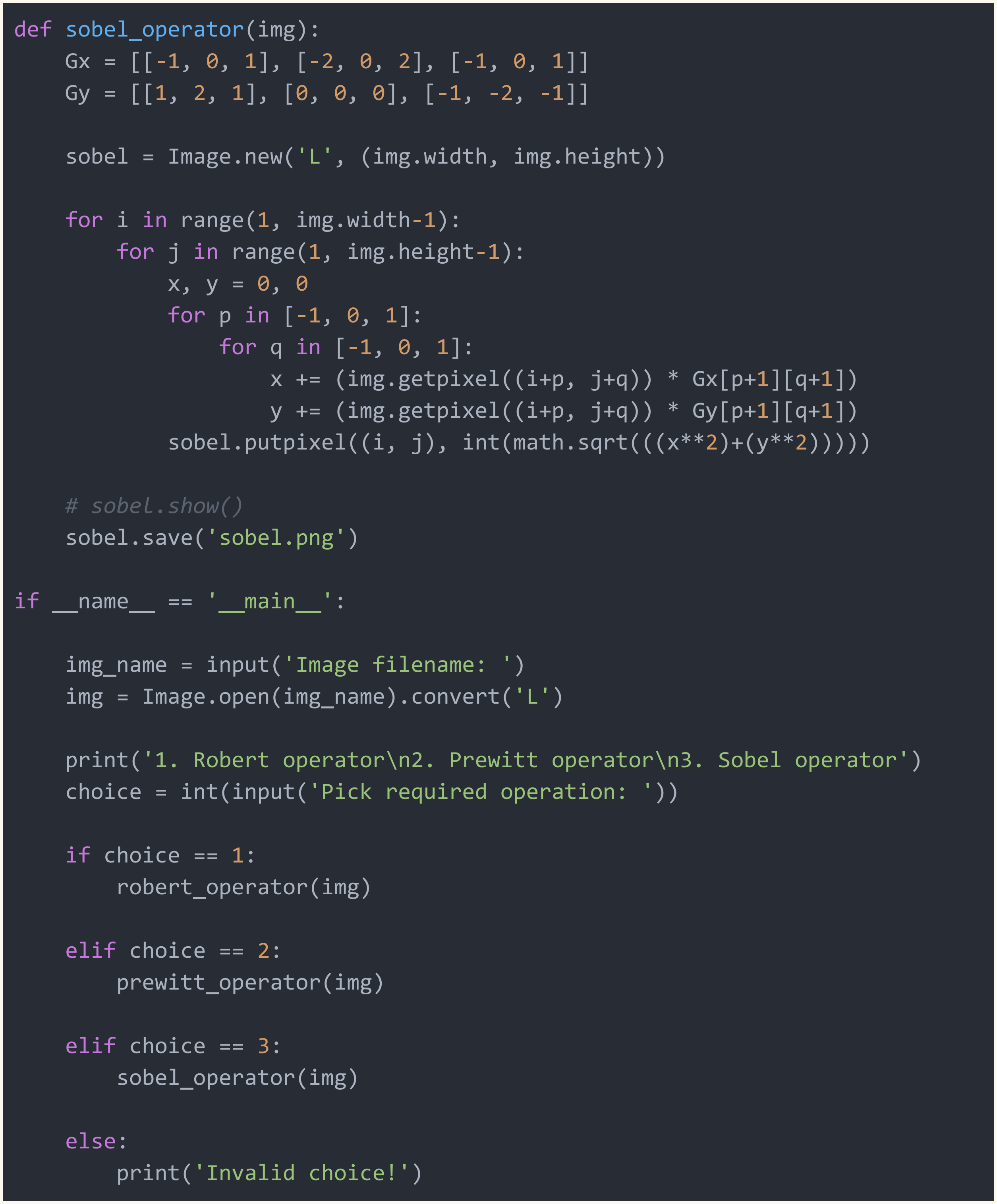
Output Images



Program for detecting edges in an image using Roberts cross gradient operator and sobel operator.

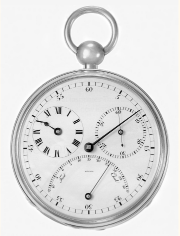


```python
from PIL import Image
import math

def robert_operator(img):
    Gx = [[1, 0], [0, -1]]
    Gy = [[0, 1], [-1, 0]]

    robert = Image.new('L', (img.width, img.height))

    for i in range(img.width-1):
        for j in range(img.height-1):
            x, y = 0, 0
            for p in range(2):
                for q in range(2):
                    x += (img.getpixel((i+p, j+q)) * Gx[p][q])
                    y += (img.getpixel((i+p, j+q)) * Gy[p][q])
            robert.putpixel((i, j), int(math.sqrt(((x**2)+(y**2)))))

    # robert.show()
    robert.save('robert.png')

def prewitt_operator(img):
    Gx = [[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]]
    Gy = [[1, 1, 1], [0, 0, 0], [-1, -1, -1]]

    prewitt = Image.new('L', (img.width, img.height))

    for i in range(1, img.width-1):
        for j in range(1, img.height-1):
            x, y = 0, 0
            for p in [-1, 0, 1]:
                for q in [-1, 0, 1]:
                    x += (img.getpixel((i+p, j+q)) * Gx[p+1][q+1])
                    y += (img.getpixel((i+p, j+q)) * Gy[p+1][q+1])
            prewitt.putpixel((i, j), int(math.sqrt(((x**2)+(y**2)))))

    # prewitt.show()
    prewitt.save('prewitt.png')
```

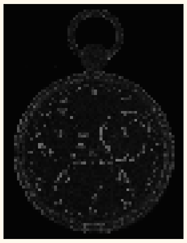
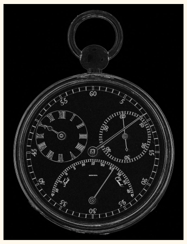


```python
def sobel_operator(img):
    Gx = [[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]]
    Gy = [[1, 2, 1], [0, 0, 0], [-1, -2, -1]]

    sobel = Image.new('L', (img.width, img.height))

    for i in range(1, img.width-1):
        for j in range(1, img.height-1):
            x, y = 0, 0
            for p in [-1, 0, 1]:
                for q in [-1, 0, 1]:
                    x += (img.getpixel((i+p, j+q)) * Gx[p+1][q+1])
                    y += (img.getpixel((i+p, j+q)) * Gy[p+1][q+1])
            sobel.putpixel((i, j), int(math.sqrt(((x**2)+(y**2)))))

    # sobel.show()
    sobel.save('sobel.png')

if __name__ == '__main__':

    img_name = input('Image filename: ')
    img = Image.open(img_name).convert('L')

    print('1. Robert operator\n2. Prewitt operator\n3. Sobel operator')
    choice = int(input('Pick required operation: '))

    if choice == 1:
        robert_operator(img)

    elif choice == 2:
        prewitt_operator(img)

    elif choice == 3:
        sobel_operator(img)

    else:
        print('Invalid choice!')
```

Input Image



Output Images

Robert  Prewitt  Sobel