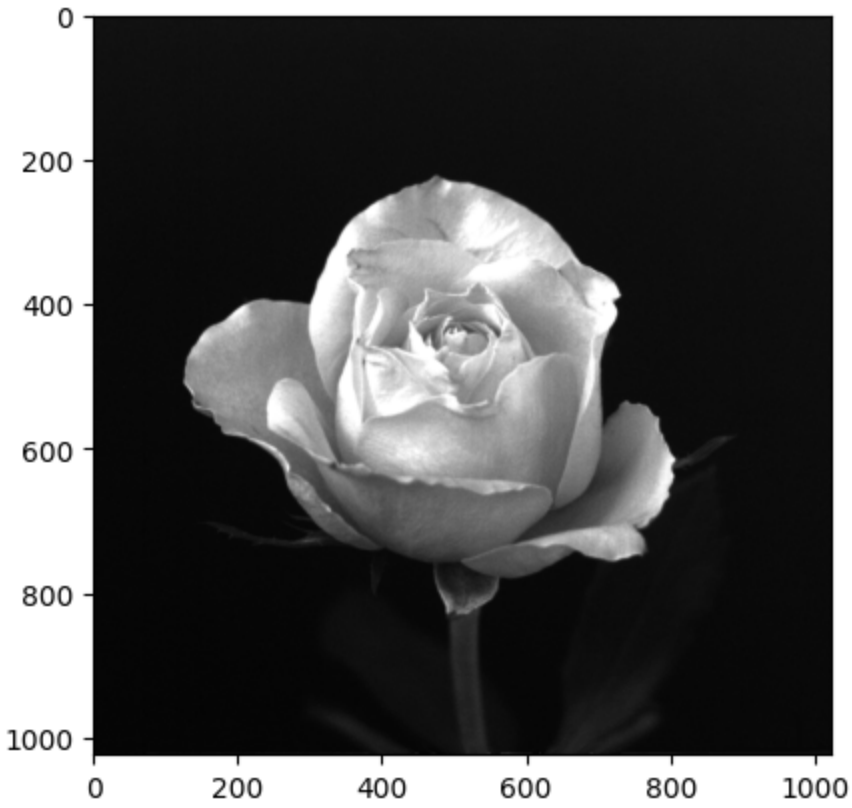
Max Haviv

Assignment 1 Image Resolution

Initial Image

In [186…

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

img = mpimg.imread('rose.jpg')
plt.imshow(img, cmap='gray')
plt.show()
```



Down Sample Function

In [187…

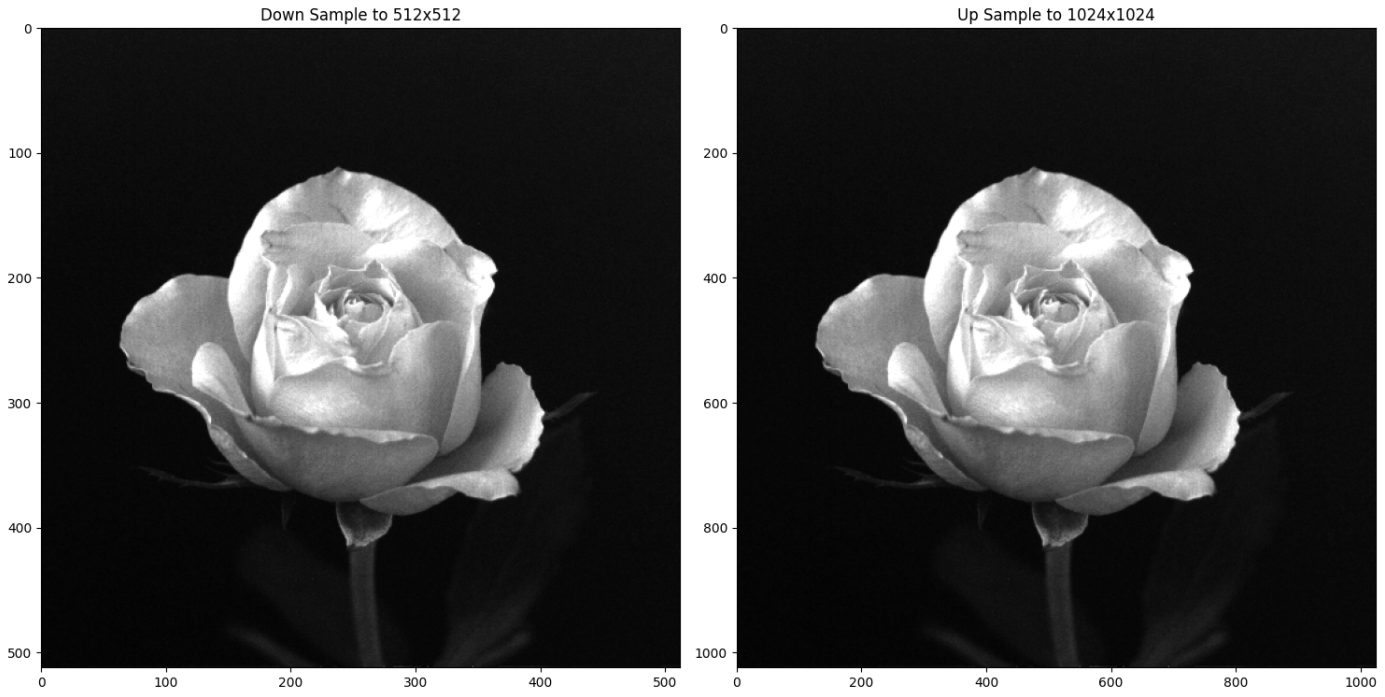
```
def down_sample(input_dimensions, output_dimensions, img):
  factor = int(input_dimensions / output_dimensions)
  down_sampled_img = np.zeros((output_dimensions, output_dimensions))

  # loop through the dimensions of the output to fill in the new image
  for row in range(output_dimensions):
    for col in range(output_dimensions):
      # select a pixel from the original image to set into the downscaled image
      # this pixel is the row and col multiplied by the factor that the image is being d
      selected_pixel= img[factor * row][factor * col]
      down_sampled_img[row,col] = selected_pixel

  return down_sampled_img
```

Up Sample Function

In [188…]
```
def up_sample(input_dimensions, output_dimensions, img):
  factor = int(output_dimensions / input_dimensions)
  up_sample_img = np.zeros((output_dimensions,output_dimensions))

  # loop through the input image
  for row in range(input_dimensions):
    for col in range(input_dimensions):
      selected_pixel = img[row][col]
      # spread the slected pixel through the rows and columns
      up_sample_img[row * factor:(row + 1) * factor,col * factor:(col + 1) * factor] = s
  return up_sample_img
```

512 Down and Up Sample

In [210…]
```
img_512 = down_sample(1024, 512, img)

img_512_upsample = up_sample(512, 1024, img_512)

fig, ax = plt.subplots(1,2, figsize=(15,10))

ax[0].imshow(img_512, cmap='gray', aspect='equal', extent=[0, img_512.shape[1], img_512.
ax[0].set_title(f"Down Sample to 512x512")
ax[0].axis('on')


ax[1].imshow(img_512_upsample, cmap='gray', aspect='equal', extent=[0, img_512_upsample.
ax[1].set_title(f"Up Sample to 1024x1024")
ax[1].axis('on')

plt.autoscale(False)
plt.tight_layout()
plt.show()
```



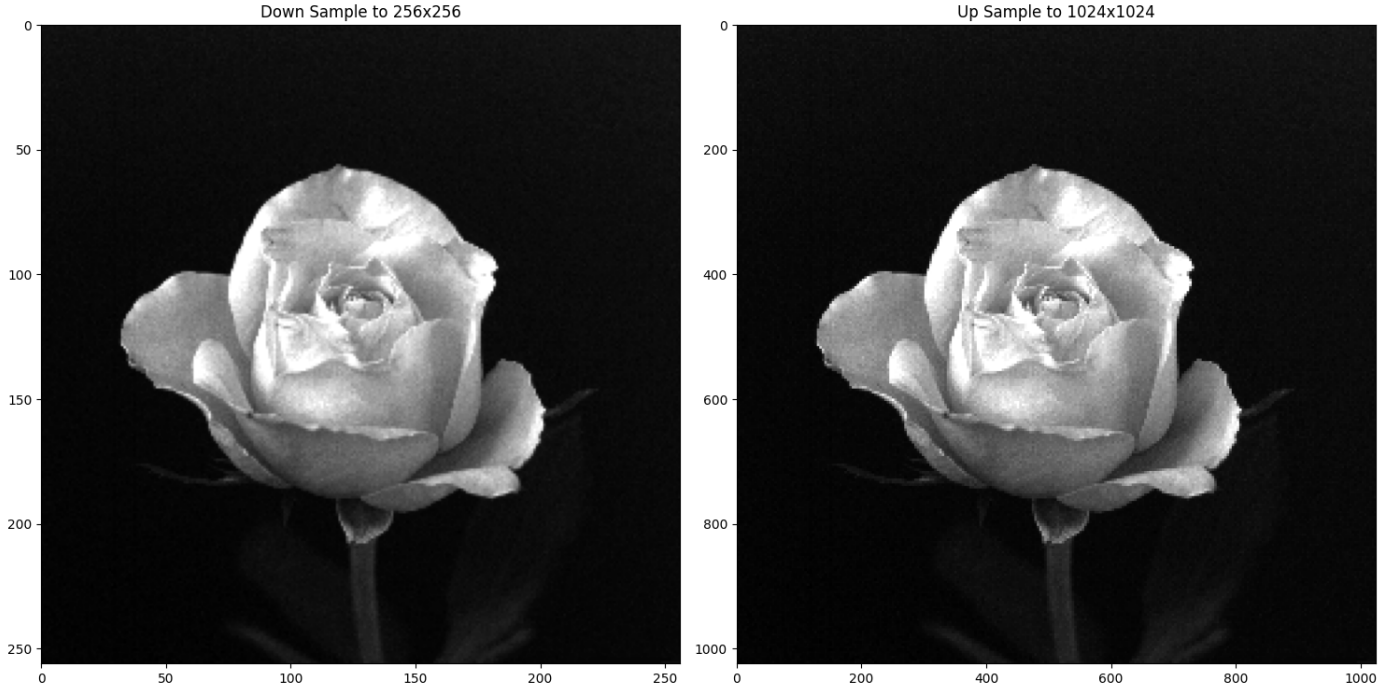
In [253…]
```
img1 = img
img2 = img_512
```

```
img3 = img_512_upsample

# Get image dimensions
img1_height, img1_width = img1.shape[:2]
img2_height, img2_width = img2.shape[:2]
img3_height, img3_width = img3.shape[:2]

spacing = 300

fig, ax = plt.subplots(figsize=(20,10))

ax.imshow(img1, cmap='gray', extent=(0, img1_width, 0, img1_height))
plt.text(img1_width / 2, img1_height + 10, 'Original Image', ha='center', fontsize=12)

ax.imshow(img2, cmap='gray', extent=(img1_width + spacing, img1_width + img2_width + spa
plt.text(img1_width + img2_width / 2 + spacing, max(img1_height, img2_height) + 10, '512

ax.imshow(img3, cmap='gray', extent=(img1_width + img2_width + 2 * spacing, img1_width +
plt.text(img1_width + img2_width + img3_width / 2 + 2 * spacing, max(img1_height, img3_h

# Adjust the axes limits to fit all three images and titles with spacing
ax.set_xlim(0, img1_width + img2_width + img3_width + 2 * spacing)
ax.set_ylim(0, max(img1_height, img2_height, img3_height) + 30)

ax.axis('off')
plt.show()
```

256 Down and Up Sample

In [211…

```
img_256 = down_sample(1024, 256, img)

img_256_upsample = up_sample(256, 1024, img_256)

fig, ax = plt.subplots(1,2, figsize=(15,10))

ax[0].imshow(img_256, cmap='gray', aspect='equal', extent=[0, img_256.shape[1], img_256.
ax[0].set_title(f"Down Sample to 256x256")
ax[0].axis('on')


ax[1].imshow(img_256_upsample, cmap='gray', aspect='equal', extent=[0, img_256_upsample.
ax[1].set_title(f"Up Sample to 1024x1024")
ax[1].axis('on')

plt.autoscale(False)
plt.tight_layout()
plt.show()
```



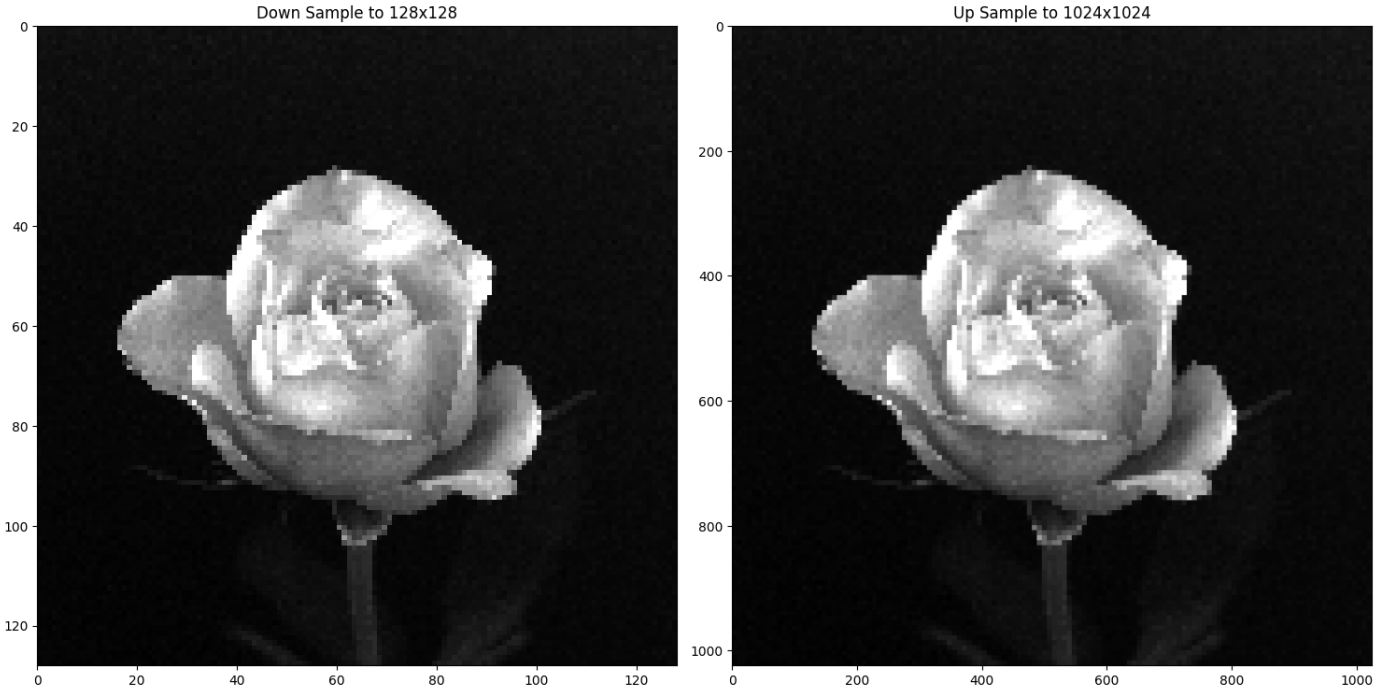
In [254…

```
img1 = img
img2 = img_256
img3 = img_256_upsample

# Get image dimensions
img1_height, img1_width = img1.shape[:2]
img2_height, img2_width = img2.shape[:2]
img3_height, img3_width = img3.shape[:2]

spacing = 300

fig, ax = plt.subplots(figsize=(20,10))

ax.imshow(img1, cmap='gray', extent=(0, img1_width, 0, img1_height))
plt.text(img1_width / 2, img1_height + 10, 'Original Image', ha='center', fontsize=12)

ax.imshow(img2, cmap='gray', extent=(img1_width + spacing, img1_width + img2_width + spa
plt.text(img1_width + img2_width / 2 + spacing, max(img1_height, img2_height) + 10, '256

ax.imshow(img3, cmap='gray', extent=(img1_width + img2_width + 2 * spacing, img1_width +
plt.text(img1_width + img2_width + img3_width / 2 + 2 * spacing, max(img1_height, img3_h

# Adjust the axes limits to fit all three images and titles with spacing
ax.set_xlim(0, img1_width + img2_width + img3_width + 2 * spacing)
ax.set_ylim(0, max(img1_height, img2_height, img3_height) + 30)

ax.axis('off')
plt.show()
```

128 Down and Up Sample

In [212…

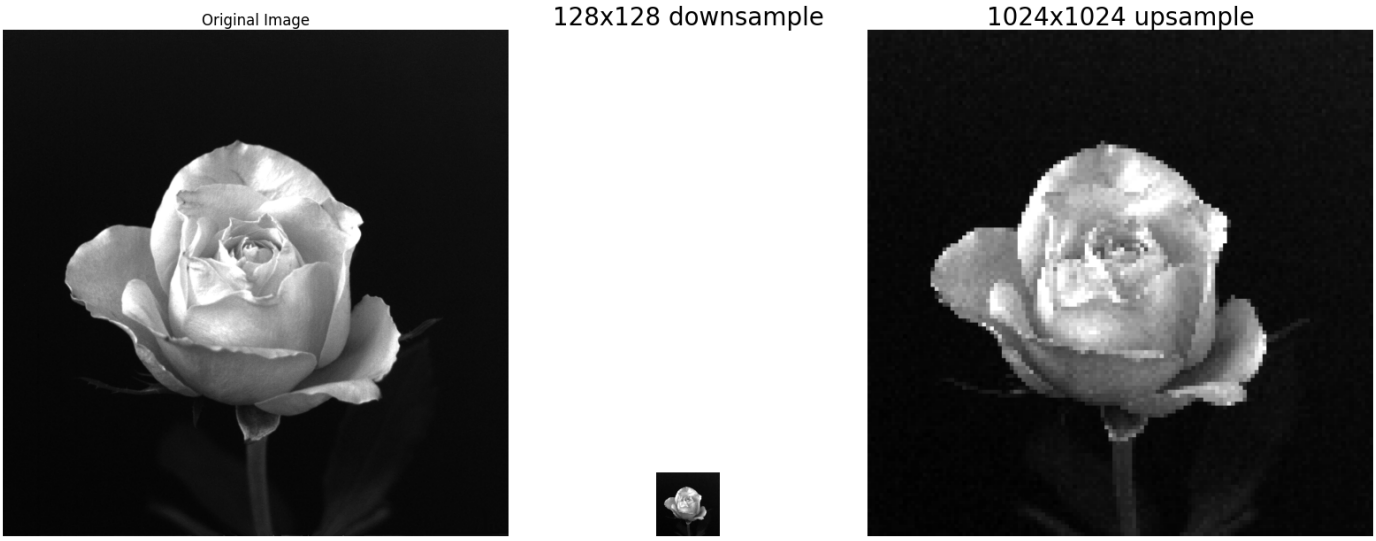
```
img_128 = down_sample(1024, 128, img)

img_128_upsample = up_sample(128, 1024, img_128)

fig, ax = plt.subplots(1,2, figsize=(15,10))

ax[0].imshow(img_128, cmap='gray', aspect='equal', extent=[0, img_128.shape[1], img_128.
ax[0].set_title(f"Down Sample to 128x128")
ax[0].axis('on')


ax[1].imshow(img_128_upsample, cmap='gray', aspect='equal', extent=[0, img_128_upsample.
ax[1].set_title(f"Up Sample to 1024x1024")
ax[1].axis('on')

plt.autoscale(False)
plt.tight_layout()
plt.show()
```



In [255…

```
img1 = img
img2 = img_128
```

```
img3 = img_128_upsample

# Get image dimensions
img1_height, img1_width = img1.shape[:2]
img2_height, img2_width = img2.shape[:2]
img3_height, img3_width = img3.shape[:2]

spacing = 300

fig, ax = plt.subplots(figsize=(20,10))

ax.imshow(img1, cmap='gray', extent=(0, img1_width, 0, img1_height))
plt.text(img1_width / 2, img1_height + 10, 'Original Image', ha='center', fontsize=12)

ax.imshow(img2, cmap='gray', extent=(img1_width + spacing, img1_width + img2_width + spa
plt.text(img1_width + img2_width / 2 + spacing, max(img1_height, img2_height) + 10, '128

ax.imshow(img3, cmap='gray', extent=(img1_width + img2_width + 2 * spacing, img1_width +
plt.text(img1_width + img2_width + img3_width / 2 + 2 * spacing, max(img1_height, img3_h

# Adjust the axes limits to fit all three images and titles with spacing
ax.set_xlim(0, img1_width + img2_width + img3_width + 2 * spacing)
ax.set_ylim(0, max(img1_height, img2_height, img3_height) + 30)

ax.axis('off')
plt.show()
```



Change the gray level of an 8-bit gray level image

define funtion

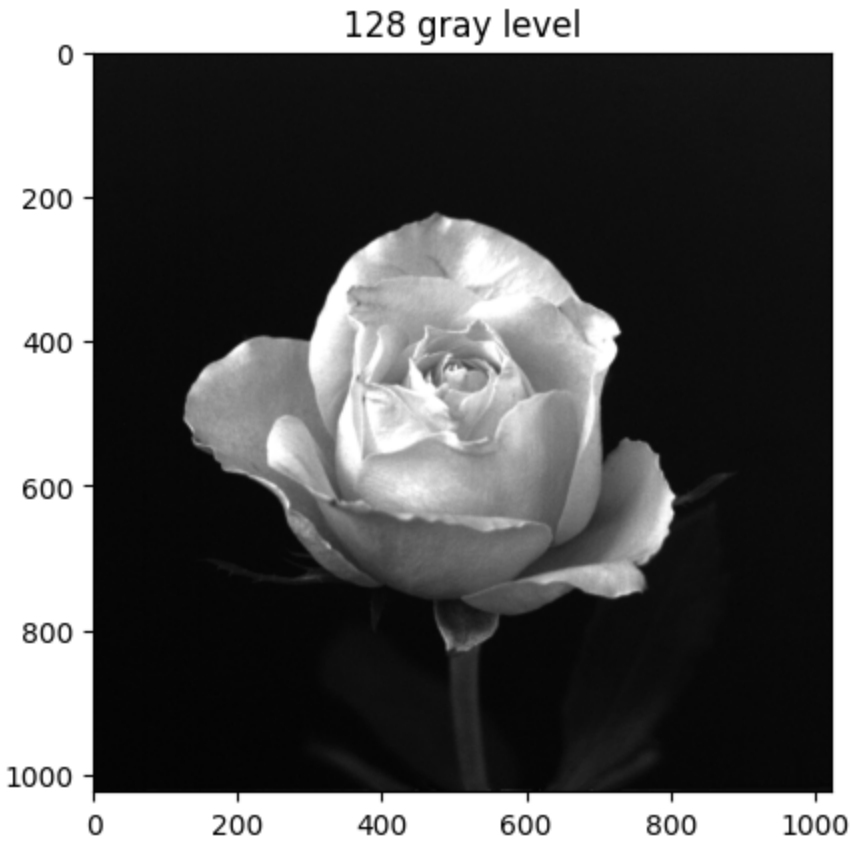
In [256…

```
def change_gray_level(bit):
  new_img = np.zeros((1024,1024))

  for row in range(1024):
    for col in range(1024):
      pixel = img[row,col]
      # create a mask for bits that we want to discard
      mask = 256 - (2 ** (8-bit))
      # bitwise operation clears the lower bits
      bitwise_pixel = pixel & mask
      # shift the bits from their original 8 bits to the bits provided
      final_pixel = bitwise_pixel >> (8 - bit)
```

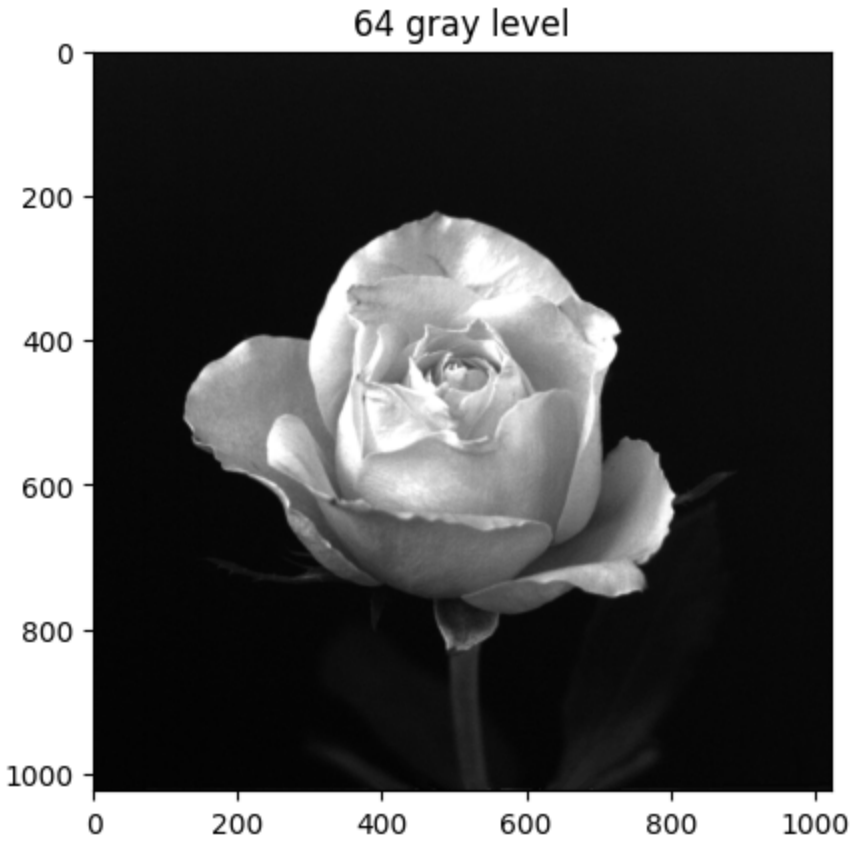
```
      new_img[row, col] = final_pixel

  return new_img
```

In [257…
```
img_128_gray = change_gray_level(7)
img_64_gray = change_gray_level(6)
img_32_gray = change_gray_level(5)
```

In [258…
```
plt.imshow(img_128_gray, cmap='gray')
plt.title("128 gray level")
plt.show()
```

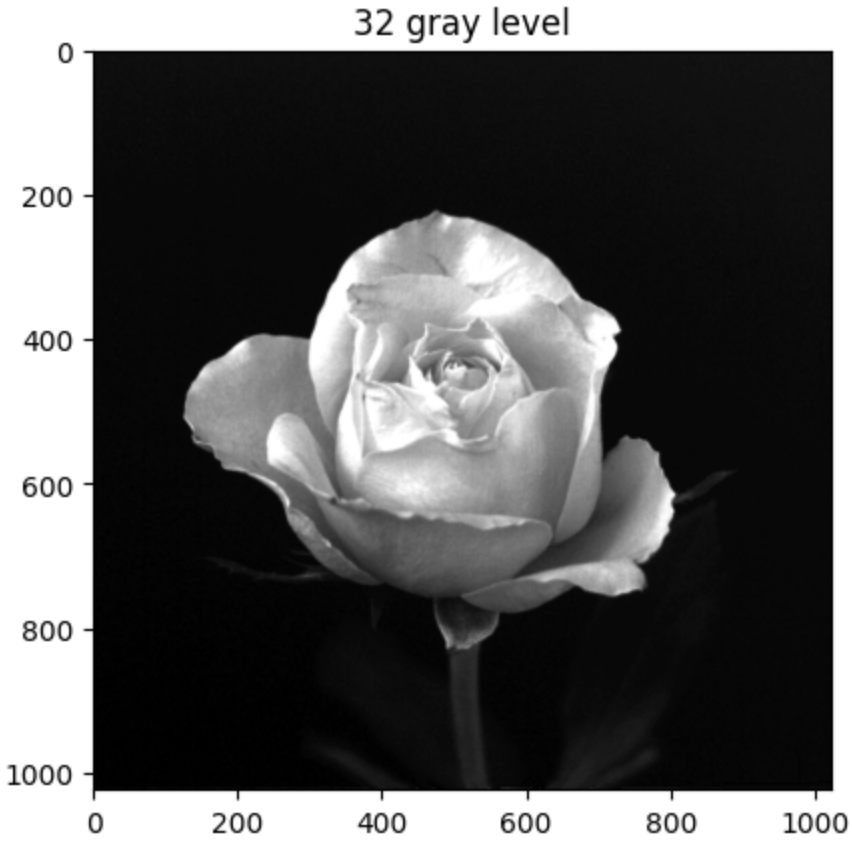


In [259…
```
plt.imshow(img_64_gray, cmap='gray')
plt.title("64 gray level")
plt.show()
```



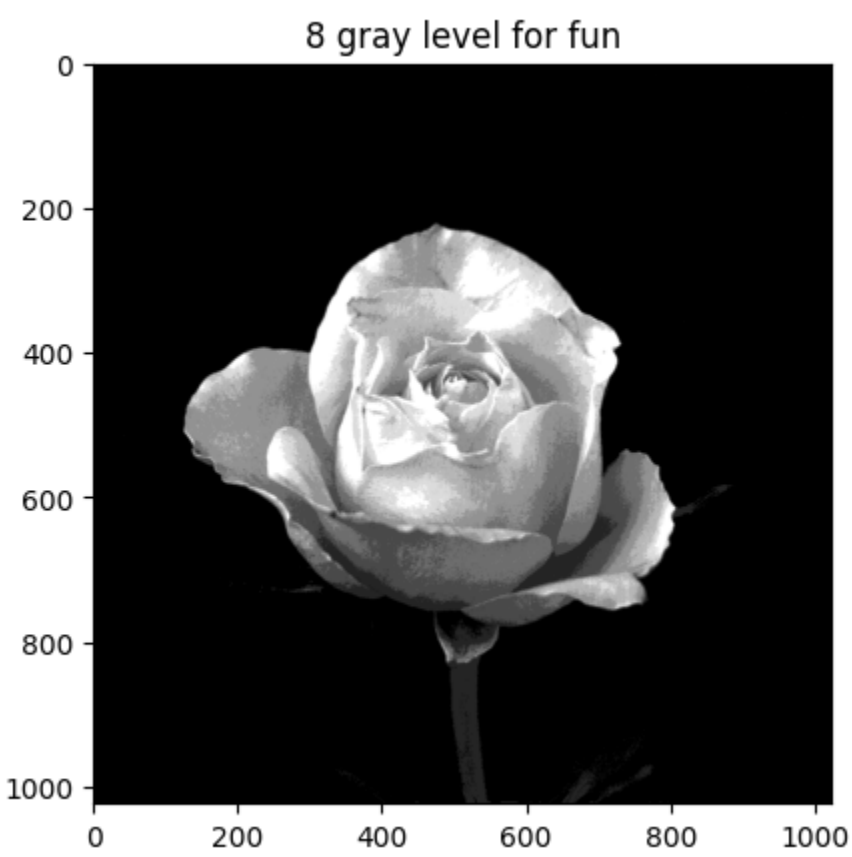
In [260…

```
plt.imshow(img_32_gray, cmap='gray')
plt.title("32 gray level")
plt.show()
```



In [262…

```
plt.imshow(change_gray_level(3), cmap='gray')
plt.title("8 gray level for fun")
plt.show()
```



In [261…

```
plt.imshow(change_gray_level(2), cmap='gray')
plt.title("4 gray level for fun")
plt.show()
```