from PIL import Image

def encode\_image(img, msg):

# Ensure message is short enough

length = len(msg)

if length > 255:

print("Message is too long!")

return False

# Ensure image is large enough

width, height = img.size

if width \* height < length:

print("Image is too small!")

return False

# Add length of message to front of message

msg = chr(length) + msg

# Convert message to binary

b\_message = ''.join([format(ord(i), "08b") for i in msg])

# Get the pixels from the image

pixels = list(img.getdata())

# Change LSB of each pixel according to the binary message

new\_pixels = []

index = 0

for pixel in pixels:

if index < len(b\_message):

new\_pixel = (pixel[0] & ~1 | int(b\_message[index]), pixel[1], pixel[2])

index += 1

else:

new\_pixel = pixel

new\_pixels.append(new\_pixel)

# Create a new image with the new pixels

new\_img = Image.new(img.mode, img.size)

new\_img.putdata(new\_pixels)

return new\_img

def decode\_image(img):

pixels = list(img.getdata())

bits = ""

for pixel in pixels:

bits += str(pixel[0] & 1)

# Convert bits to characters

chars = [chr(int(bits[i:i+8], 2)) for i in range(0, len(bits), 8)]

# Extract the message length

length = ord(chars[0])

msg = ''.join(chars[1:length + 1])

return msg

# Usage example

img = Image.open("input\_image.png")

msg = "Hello, this is a hidden message!"

encoded\_img = encode\_image(img, msg)

if encoded\_img:

encoded\_img.save("encoded\_image.png")

decoded\_msg = decode\_image(encoded\_img)

print(f"Decoded message: {decoded\_msg}")