```
from cv2 import cv2
import numpy as np
import os

def to_bin(data):
    """Convert `data` to binary format as string"""
    if isinstance(data, str):
        return ''.join([ format(ord(i), "08b") for i in data ])
    elif isinstance(data, bytes) or isinstance(data, np.ndarray
):
        return [ format(i, "08b") for i in data ]
    elif isinstance(data, int) or isinstance(data, np.uint8):
        return format(data, "08b")
    else:
        raise TypeError("Type not supported.")


def encode(image_name, secret_data):
    # read the image
    image = cv2.imread(image_name)
    # maximum bytes to encode
    n_bytes = image.shape[0] * image.shape[1] * 3 // 8
    print("[*] Maximum bytes to encode:", n_bytes)
    if len(secret_data) > n_bytes:
        raise ValueError("[!] Insufficient bytes, need bigger i
mage or less data.")
    print("[*] Encoding data...")
    # add stopping criteria
    secret_data += "====="
    data_index = 0
    # convert data to binary
    binary_secret_data = to_bin(secret_data)
    # size of data to hide
    data_len = len(binary_secret_data)

    for row in image:
```

```
        for pixel in row:
            # convert RGB values to binary format
            r, g, b = to_bin(pixel)
            # modify the least significant bit only if there is
 still data to store
            if data_index < data_len:
                # least significant red pixel bit
                pixel[0] = int(r[:-1] + binary_secret_data[data
_index], 2)
                data_index += 1
            if data_index < data_len:
                # least significant green pixel bit
                pixel[1] = int(g[:-1] + binary_secret_data[data
_index], 2)
                data_index += 1
            if data_index < data_len:
                # least significant blue pixel bit
                pixel[2] = int(b[:-1] + binary_secret_data[data
_index], 2)
                data_index += 1
            # if data is encoded, just break out of the loop
            if data_index >= data_len:
                break
    return image


def decode(image_name):
    print("[+] Decoding...")
    # read the image
    image = cv2.imread(image_name)
    binary_data = ""
    for row in image:
        for pixel in row:
            r, g, b = to_bin(pixel)
            binary_data += r[-1]
            binary_data += g[-1]
```

```
            binary_data += b[-1]

    # split by 8-bits
    all_bytes = [ binary_data[i: i+8] for i in range(0, len(bin
ary_data), 8) ]
    # convert from bits to characters
    decoded_data = ""
    for byte in all_bytes:
        decoded_data += chr(int(byte, 2))
        if decoded_data[-5:] == "=====":
            break
    return decoded_data[:-5]


if __name__ == "__main__":
    import argparse
    parser = argparse.ArgumentParser(description="Steganography
 encoder/decoder, this Python scripts encode data within images
.")
    parser.add_argument("-t", "--text", help="The text data to
encode into the image, this only should be specified for encodi
ng")
    parser.add_argument("-e", "--encode", help="Encode the foll
owing image")
    parser.add_argument("-d", "--decode", help="Decode the foll
owing image")

    args = parser.parse_args()
    secret_data = args.text
    if args.encode:
        # if the encode argument is specified
        input_image = args.encode
        print("input_image:", input_image)
        # split the absolute path and the file
        path, file = os.path.split(input_image)
        # split the filename and the image extension
```

```
        filename, ext = file.split(".")
        output_image = os.path.join(path, f"{filename}_encoded.
{ext}")
        # encode the data into the image
        encoded_image = encode(image_name=input_image, secret_d
ata=secret_data)
        # save the output image (encoded image)
        cv2.imwrite(output_image, encoded_image)
        print("[+] Saved encoded image.")
    if args.decode:
        input_image = args.decode
        # decode the secret data from the image
        decoded_data = decode(input_image)
        print("[+] Decoded data:", decoded_data)
```