STEGANOGRAPHY

INTRODUCTION TO INNOVATIVE PROJECT

Vellore Institute Of Technology

B.TECH COMPUTER SCIENCE

-FACULTY NAME:PROF. KALAINATHAN S

TEAM MEMBERS:-

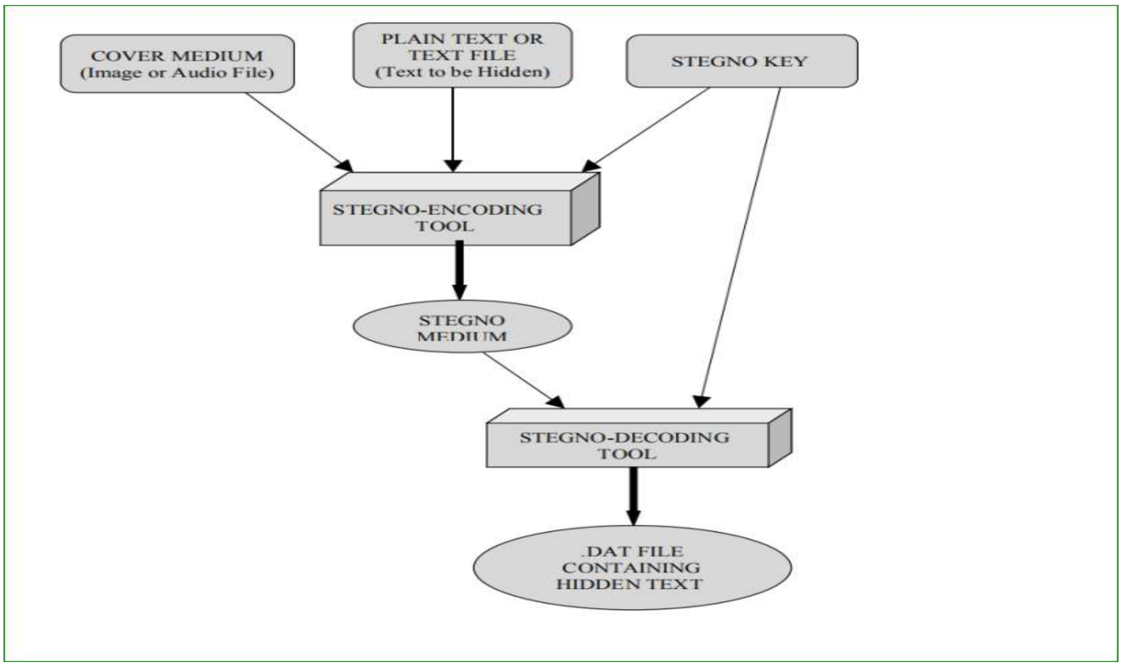
SAURAV RANJAN(19BCE0455)
SAKSHAM MINOCHA(19BCE0466)
SATYANSH TEWARI(19BCE0520)
AYUSH KUMAR(19BCE0805)
SAYAN JANA (19BCE0912)
DARAHAS VANDANA (19BCE0923)
RITWIK GOEL(19BCE0940)
MOHAMMAD MOIN (19BCE2276)
DAVYA VUYYURU(19BCE2436)
SANTOSH RAM NITISH YERAMILLI(19BCE2496)

ABSTRACT:

Steganography is a popular technique which refers to hiding of data within some form of other ordinary, plain file, be it in text, image, audio or video file. Our project is based on implementing the various aspects of steganography to help those people who do not have a technical background or sound knowledge about this subject. The applications of steganography are wide and useful. Our aim is that anyone who wishes to use steganalysis and steganography but are not able to do so, can now reach a platform where their technical skills do not form a barrier in their work. Our trustworthy and efficient web application will not just provide the options to the user regarding the choice of cover file they want to use, but also use encryption to protect the files from third-parties, hence ensuring both convenience and privacy.

In this project we will first design some algorithms which will help us in implementing the image and audio steganography. These algorithms will be discussed further in methodology section. After that we will design the webpages of our website by using HTML and CSS. We will add dynamic aspects to our web pages by using Java Script, SAAS and React JS. After this we will make our backend of the website by using Django (a Python Framework). Django will allow us to maintain our database and also will help the user to upload their text, images and audio files. Django will also perform the CSRF Tokens which will prevent the Cross Site Request Forgery (CSRF) and will keep the data our users secure.

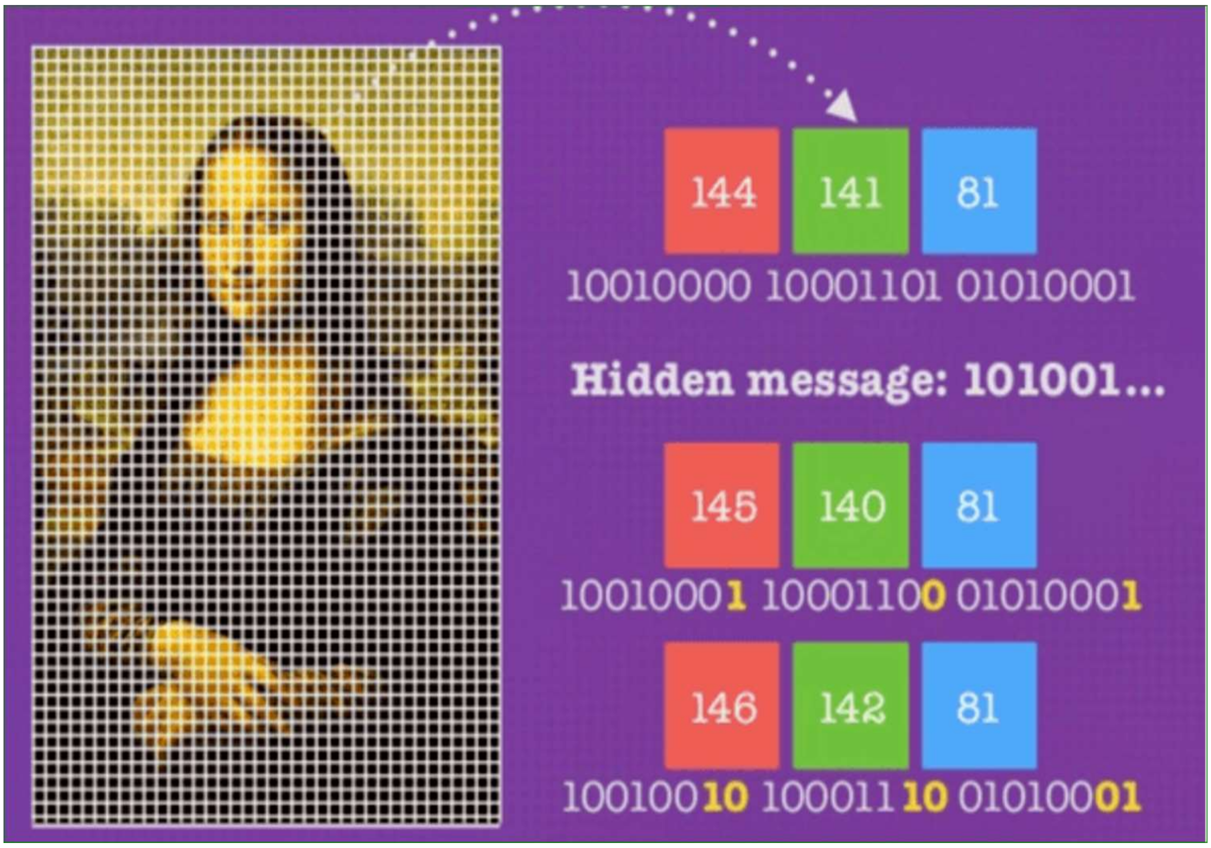
METHODOLOGY:


COVER MEDIUM (Image or Audio File)
PLAIN TEXT OR TEXT FILE (Text to be Hidden)
STEGNO KEY
STEGNO-ENCODING TOOL
STEGNO MEDIUM
STEGNO-DECODING TOOL
.DAT FILE CONTAINING HIDDEN TEXT

<u>STEGANOGRAPHY VS CRYPTOGRAPHY</u>

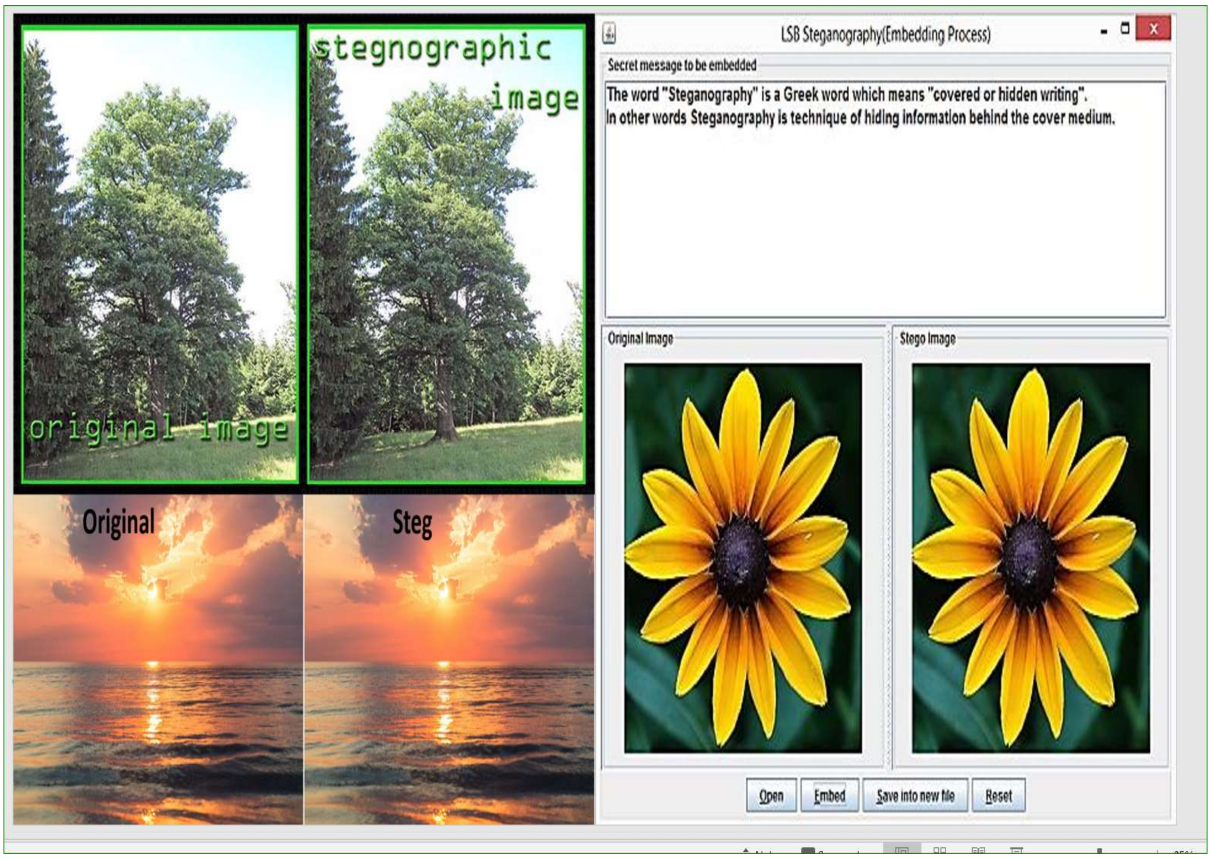
- **Steganography can be viewed as akin to cryptography. Both have been used throughout recorded history as means to protect information. At times these two technologies seem to converge while the objectives of the two differ.**
- **Cryptographic techniques "scramble" messages so if intercepted, the messages cannot be understood. Steganography, an essence, "camouflages" a message to hide its existence and make it seem "invisible" thus concealing the fact that a message is being sent altogether. An encrypted message may draw suspicion while an invisible message will not.**
- **Steganography cannot be detected. Therefore, it is used when encryption is not permitted. Or, more commonly, steganography is used to supplement encryption. An encrypted file may still hide information using steganography, so even if the encrypted file is deciphered, the hidden message is not seen.**

EXAMPLE:



Steganalysis:

- **Analyzing images for possible hidden data**
- **Many algorithms perform a statistical analysis to detect anomalies in the cover object (E.g. repetitive patterns could indicate use of a steganography tool or hidden message)**
- **Investigating pixel "neighborhoods" to find inconsistencies with ways files are compressed**



The word "Steganography" is a Greek word which means "covered or hidden writing".
In other words Steganography is technique of hiding information behind the cover medium.

CRITICAL REVIEW:

The word steganography is of Greek origin and means "covered, or hidden writing". It is the science of hiding information. Whereas the goal of cryptography is to make data unreadable by a third party, the goal of steganography is to hide the data from a third party. In steganography information can be hidden in carriers such as images, audio files, text files, and video and data transmissions. When message is hidden in the carrier a stego carrier is formed for example a stego image. It will be perceived to be as close as possible to the original carrier or cover image by the human senses. Steganography and cryptography are closely related. Cryptography scrambles messages so they cannot be understood. Steganography, on the other hand, will hide the message so that there is no knowledge of the existence of the message in the first place. Steganography includes the concealment of information within computer files. In digital steganography, electronic communications may include steganographic coding inside of a transport layer, such as a document file, image file, program or protocol.

Image steganography can be broadly classified into spatial domain, transform domain, spread spectrum. In spatial domain, secret message is embedded in pixel value directly whereas transform domain methods achieve embedding by first transforming the image from spatial to frequency domain using some methods/algorithms

The current research methods in spatial domain are s discrete cosine transform (DCT), discrete wavelet transform (DWT), Hadamard transform, Dual tree DWT, double density dual tree DWT (DD DT DWT), ridge let transform, curvelet transform.

In this Project we will implement the Least Significant Bit (LSB) for storing the text in images and audio files.

LSB METHOD:

- Most common form of digital steganography
- In a RGB image, Information is hidden in the LSB[s] of the RGB values of each pixel
- In a 24-bit bitmap, each pixel represented by 3 bytes.

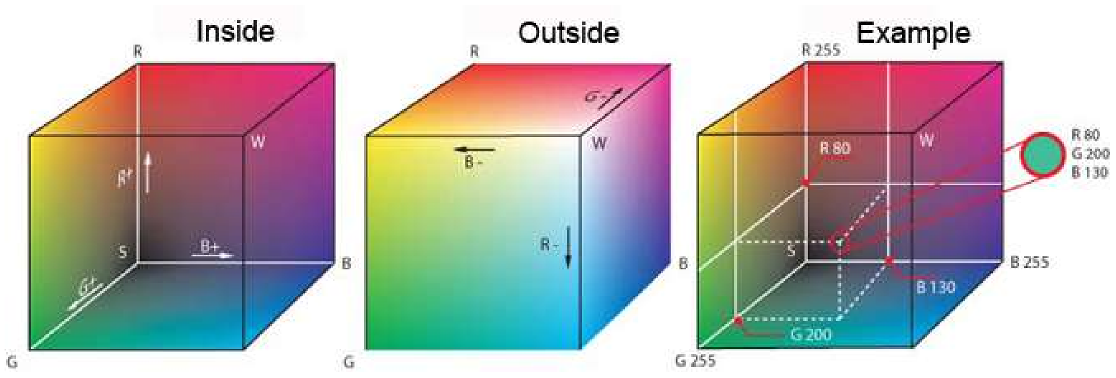
  8 bits representing red value = 2^8 = 256 shades of RED

  8 bits representing green value = 2^8 = 256 shades of GREEN

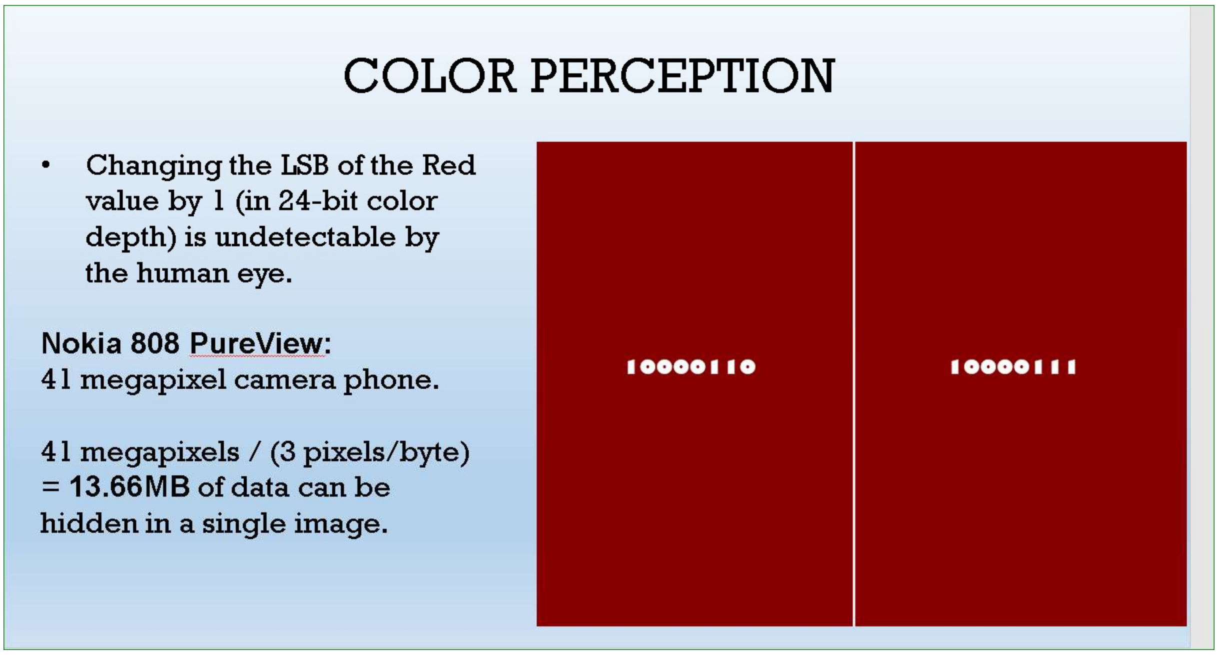
  8 bits representing blue value = 2^8 = 256 shades of BLUE

  16,777,216 possible colors

- Effectively we have 3-4 bits of data to hide information in for every pixel.



- 32bpp format contains an alpha channel
- 8 required for ASCII character



COLOR PERCEPTION

- Changing the LSB of the Red value by 1 (in 24-bit color depth) is undetectable by the human eye.

Nokia 808 PureView:
41 megapixel camera phone.

41 megapixels / (3 pixels/byte) = 13.66MB of data can be hidden in a single image.

10000110
10000111

PROCESS AND THE MATHETMATICS:

I. The data to be concealed is compressed and hidden within another file.

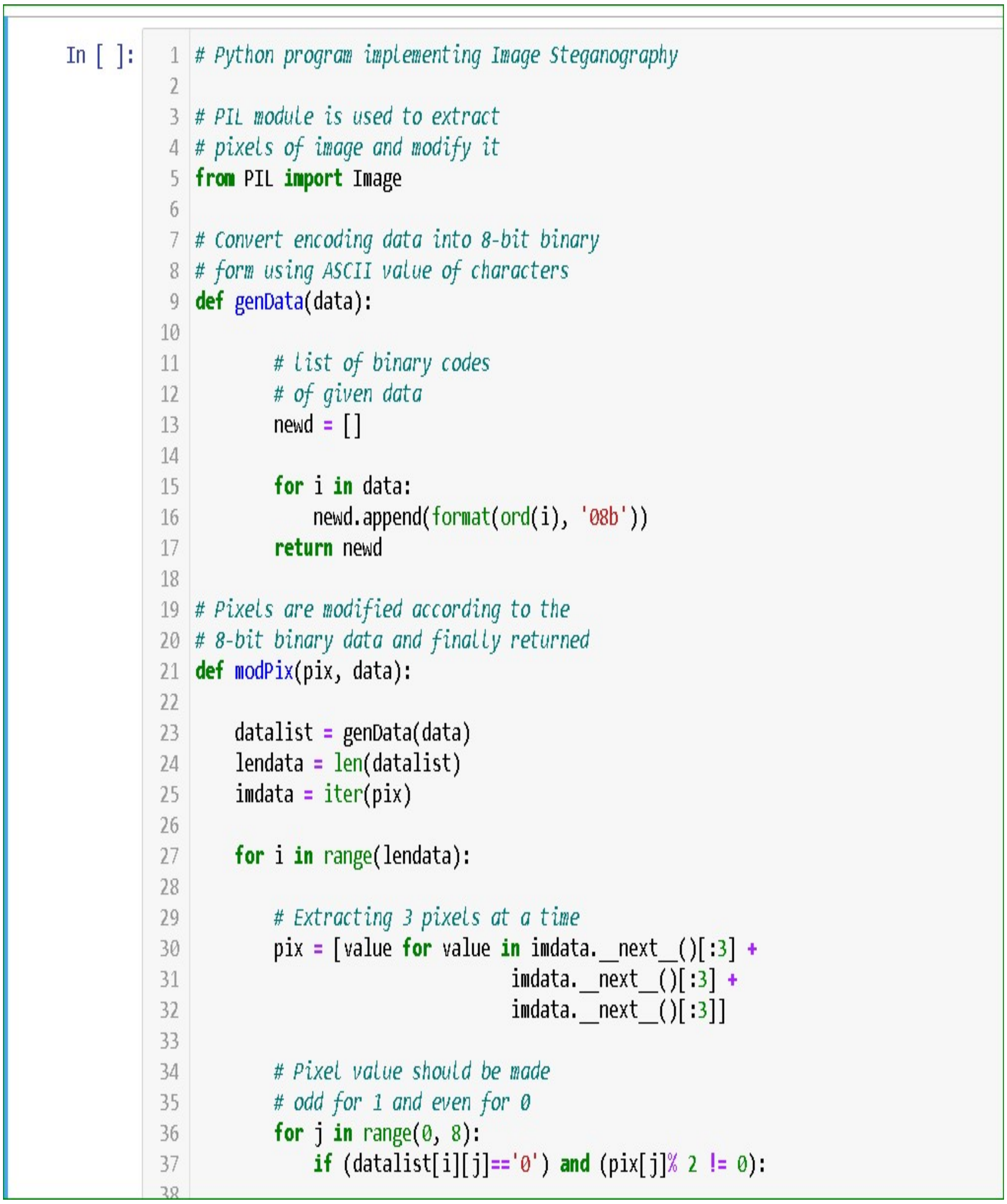
II. The first step is to find a file which will be used to hide the message (also called a carrier or a container).

III. The next step is to embed the message one wants to hide within the carrier using a steganographic technique.

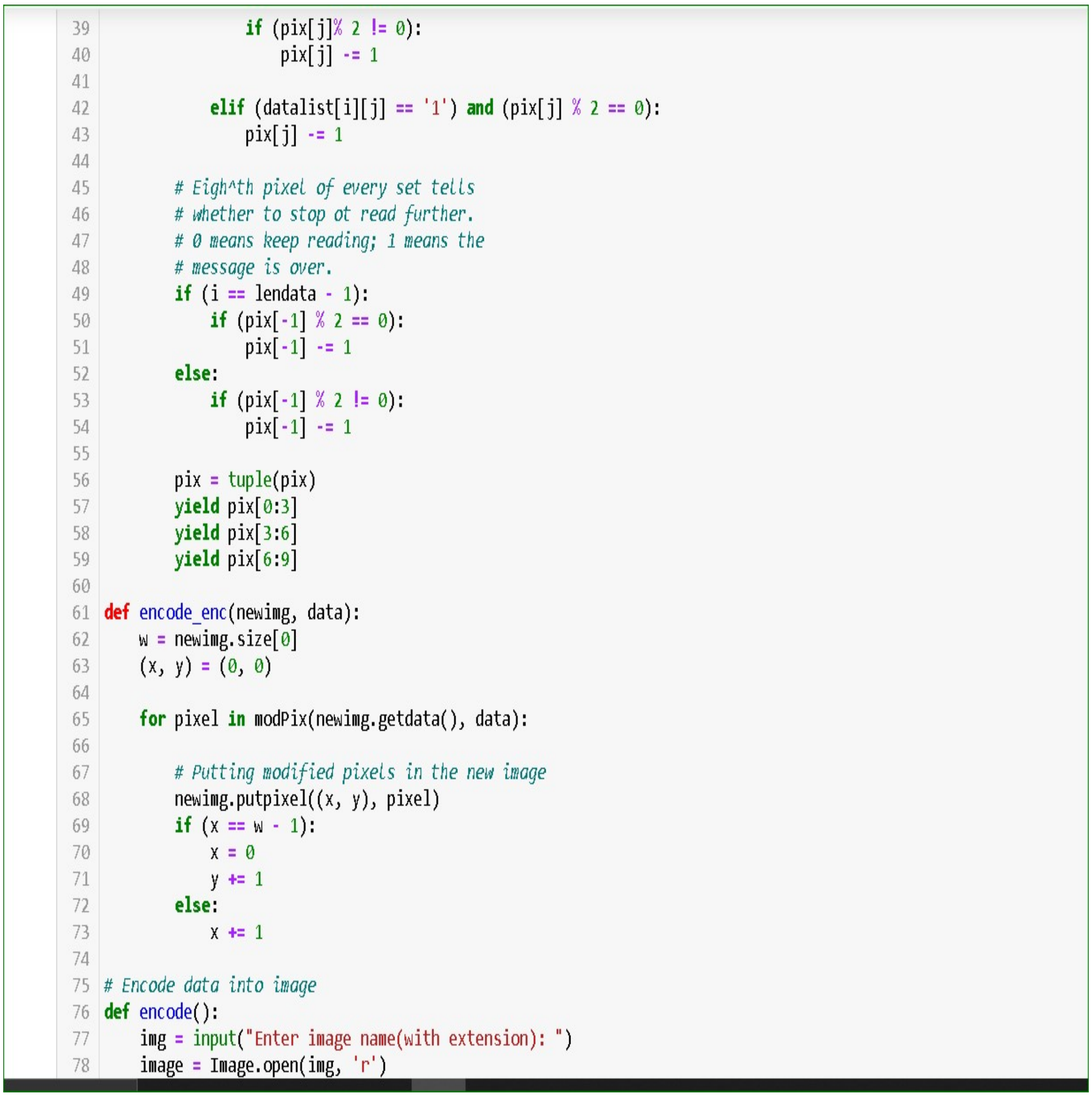
IV. Two different techniques commonly used for embedding are:

- Replace the least significant bit of each byte in the [carrier] with a single bit for the hidden message.
- Select certain bytes in which to embed the message using a random number generator; resampling the bytes to pixel mapping to preserve color scheme, in the case of an image...; hiding information in the coefficients of the discrete cosine, fractal or wavelet transform of an image; and applying mimic functions that adapt bit pattern to a given statistical distribution.

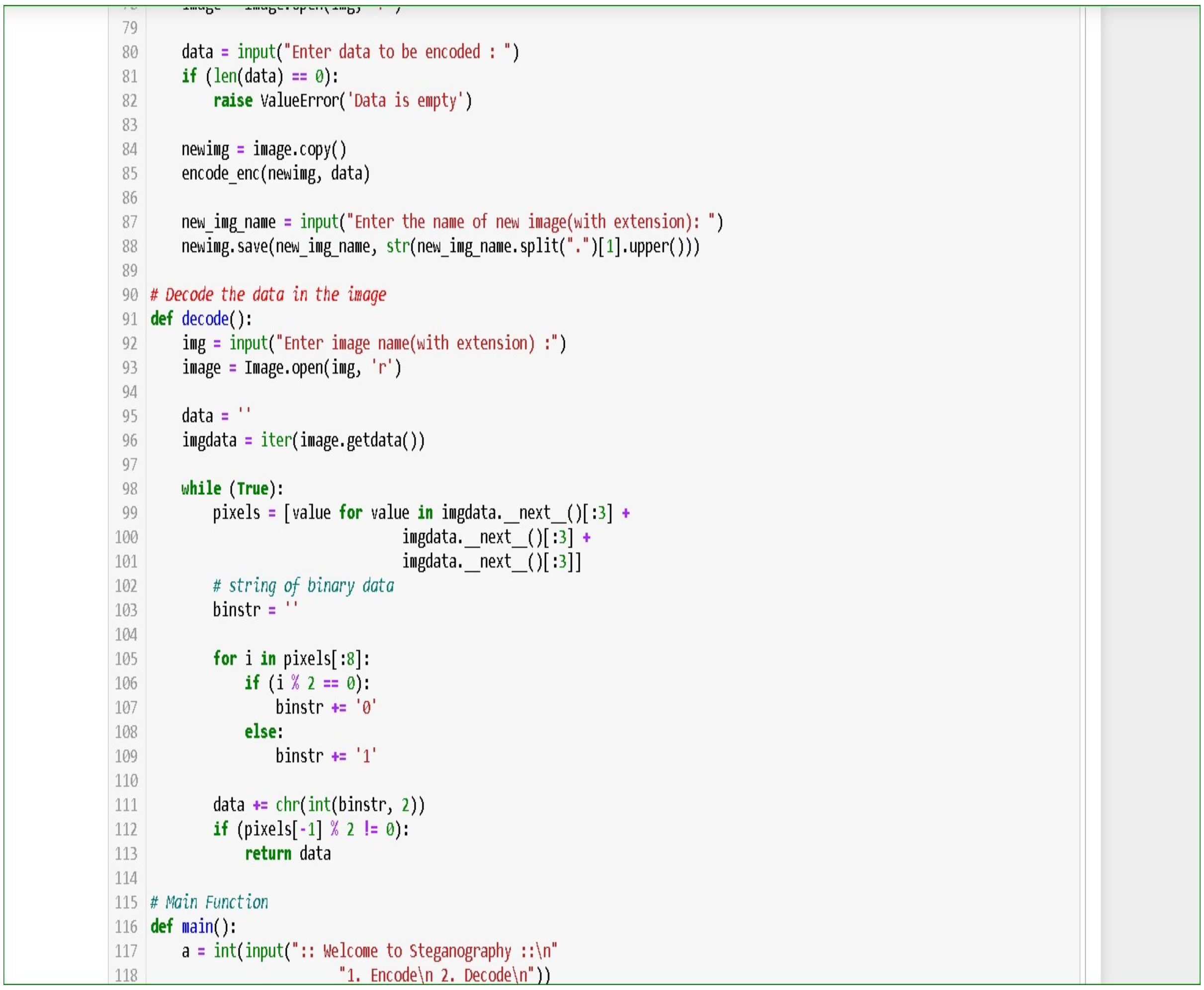
IMPLEMENTATION:



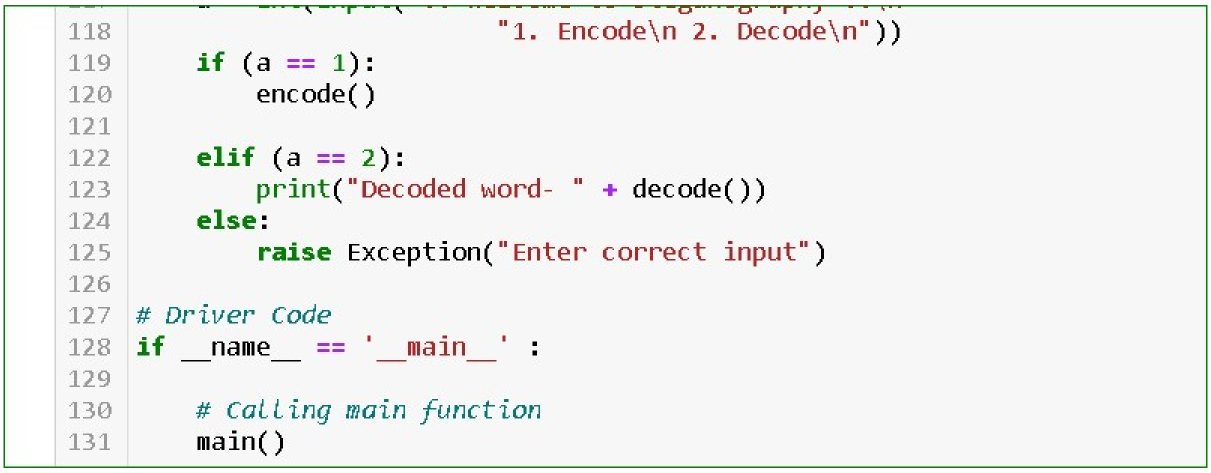
```
In [ ]:
# Python program implementing Image Steganography

# PIL module is used to extract
# pixels of image and modify it
from PIL import Image

# Convert encoding data into 8-bit binary
# form using ASCII value of characters
def genData(data):

        # list of binary codes
        # of given data
        newd = []

        for i in data:
            newd.append(format(ord(i), '08b'))
        return newd

# Pixels are modified according to the
# 8-bit binary data and finally returned
def modPix(pix, data):

    datalist = genData(data)
    lendata = len(datalist)
    imdata = iter(pix)

    for i in range(lendata):

        # Extracting 3 pixels at a time
        pix = [value for value in imdata.__next__()[:3] +
                                  imdata.__next__()[:3] +
                                  imdata.__next__()[:3]]

        # Pixel value should be made
        # odd for 1 and even for 0
        for j in range(0, 8):
            if (datalist[i][j]=='0') and (pix[j]% 2 != 0):
```



```python
                if (pix[j]% 2 != 0):
                    pix[j] -= 1

            elif (datalist[i][j] == '1') and (pix[j] % 2 == 0):
                pix[j] -= 1

        # Eigh^th pixel of every set tells
        # whether to stop ot read further.
        # 0 means keep reading; 1 means the
        # message is over.
        if (i == lendata - 1):
            if (pix[-1] % 2 == 0):
                pix[-1] -= 1
        else:
            if (pix[-1] % 2 != 0):
                pix[-1] -= 1

        pix = tuple(pix)
        yield pix[0:3]
        yield pix[3:6]
        yield pix[6:9]

def encode_enc(newimg, data):
    w = newimg.size[0]
    (x, y) = (0, 0)

    for pixel in modPix(newimg.getdata(), data):

        # Putting modified pixels in the new image
        newimg.putpixel((x, y), pixel)
        if (x == w - 1):
            x = 0
            y += 1
        else:
            x += 1

# Encode data into image
def encode():
    img = input("Enter image name(with extension): ")
    image = Image.open(img, 'r')
```



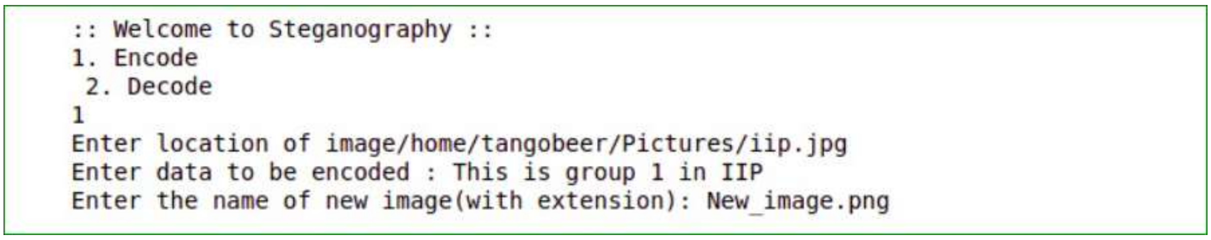
```python

    data = input("Enter data to be encoded : ")
    if (len(data) == 0):
        raise ValueError('Data is empty')

    newimg = image.copy()
    encode_enc(newimg, data)

    new_img_name = input("Enter the name of new image(with extension): ")
    newimg.save(new_img_name, str(new_img_name.split(".")[1].upper()))

# Decode the data in the image
def decode():
    img = input("Enter image name(with extension) :")
    image = Image.open(img, 'r')

    data = ''
    imgdata = iter(image.getdata())

    while (True):
        pixels = [value for value in imgdata.__next__()[:3] +
                                  imgdata.__next__()[:3] +
                                  imgdata.__next__()[:3]]
        # string of binary data
        binstr = ''

        for i in pixels[:8]:
            if (i % 2 == 0):
                binstr += '0'
            else:
                binstr += '1'

        data += chr(int(binstr, 2))
        if (pixels[-1] % 2 != 0):
            return data

# Main Function
def main():
    a = int(input(":: Welcome to Steganography ::\n"
                        "1. Encode\n 2. Decode\n"))
```



```python
                        "1. Encode\n 2. Decode\n"))
    if (a == 1):
        encode()

    elif (a == 2):
        print("Decoded word-  " + decode())
    else:
        raise Exception("Enter correct input")

# Driver Code
if __name__ == '__main__' :

    # Calling main function
    main()
```

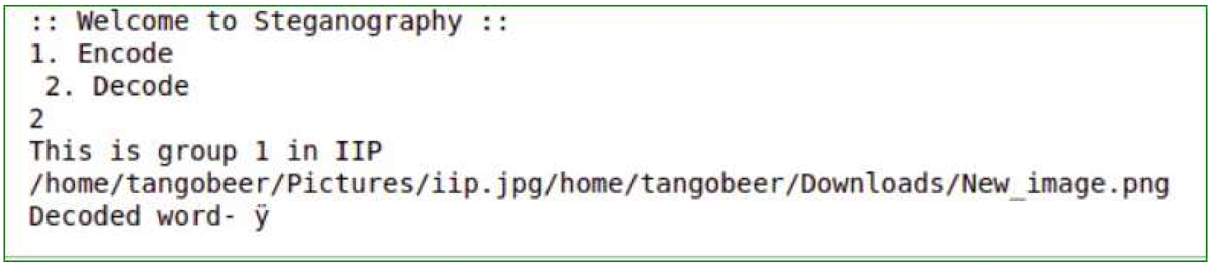
INPUT AND OUTPUT SCREENSHOTS:

ENCODING:



```
:: Welcome to Steganography ::
1. Encode
 2. Decode
1
Enter location of image/home/tangobeer/Pictures/iip.jpg
Enter data to be encoded : This is group 1 in IIP
Enter the name of new image(with extension): New_image.png
```

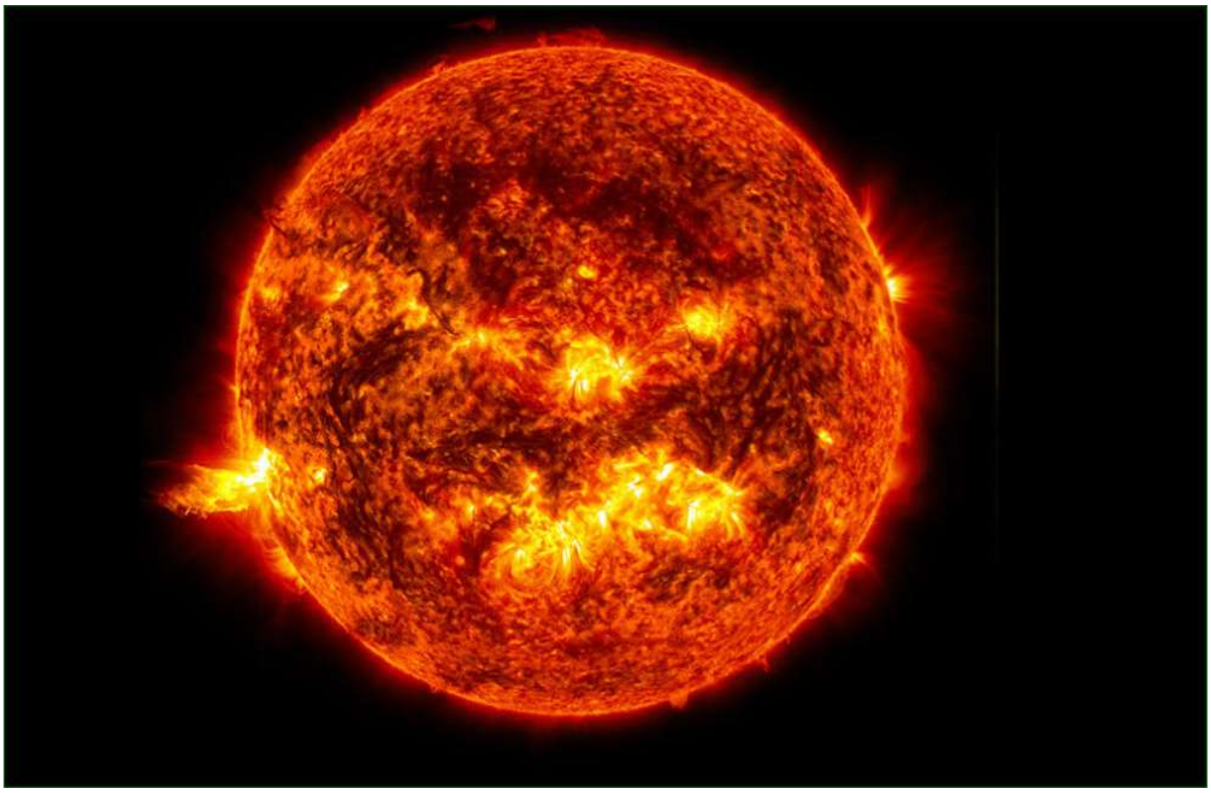
DECODING:



```
:: Welcome to Steganography ::
1. Encode
 2. Decode
2
This is group 1 in IIP
/home/tangobeer/Pictures/iip.jpg/home/tangobeer/Downloads/New_image.png
Decoded word- ÿ
```

IMAGE AFTER ENCODING AND DECODING:

BEFORE ENCODING:



AFTER ENCODING: