

## Abstract

This semester's work is related to implementing the steganographic technique in the code. This project work demonstrates the image steganography which means hiding a message in the image. I've worked on this technique to hide textual data in the image.

Tools used - **Platform** : Google collab, **Programming language** : Python

## Introduction

Steganography is a cryptographic art (method/science) that is related to hiding the information inside the information or hiding the message inside the hidden message. Steganography and cryptography enhance the confidentiality of the message. If we think about hiding a message using this technique, we do not need to encrypt the message. Reduces the encryption/decryption burden.

Some popular steganography techniques are:

- a> Text steganography – Hiding into plain and formatted text.
- b> Image steganography – Hiding into images
- c> Audio steganography – Hiding into audio
- d> Video steganography – Hiding into video
- e> Network steganography – Hiding into network protocols
- f> Steganography on executables – Hiding into executables

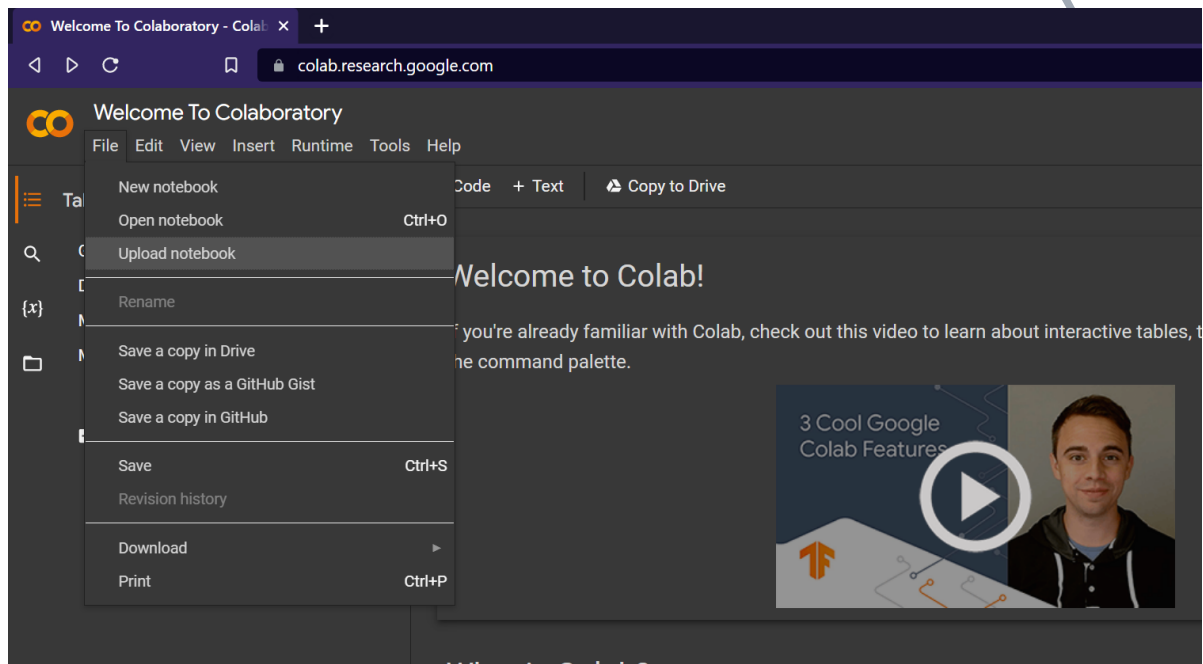
## System manual

### Step 1:

Upload .ipynb(Interactive Python Notebook) in Google Drive and open.

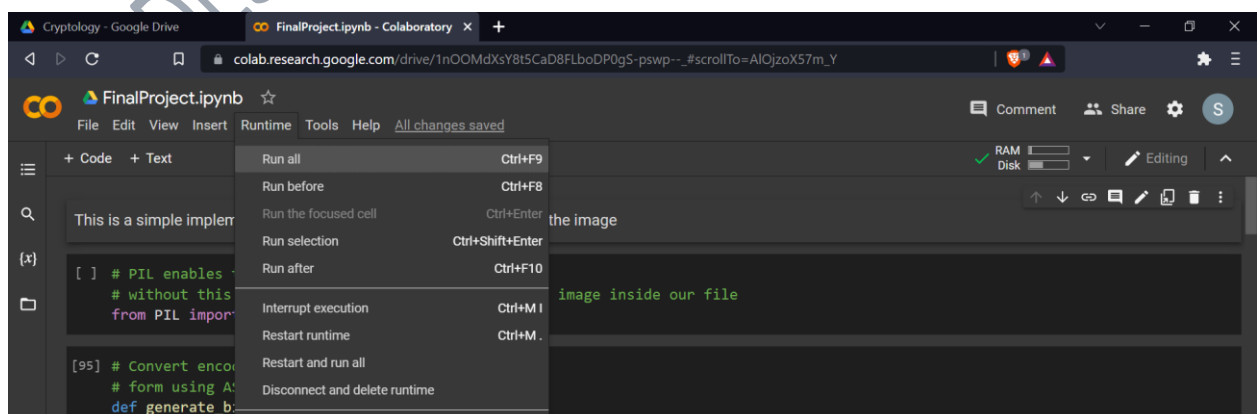
Or

Go through <https://colab.research.google.com/> and upload the Notebook file.



### Step 2:

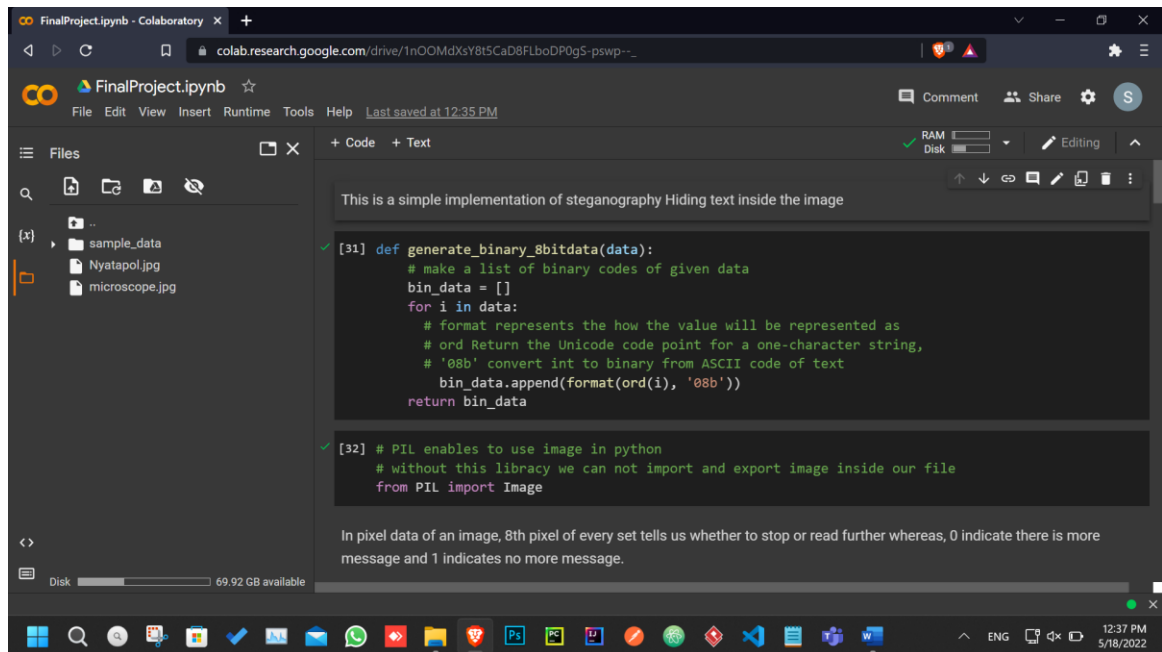
Once you successfully upload the file, now you can open that file.



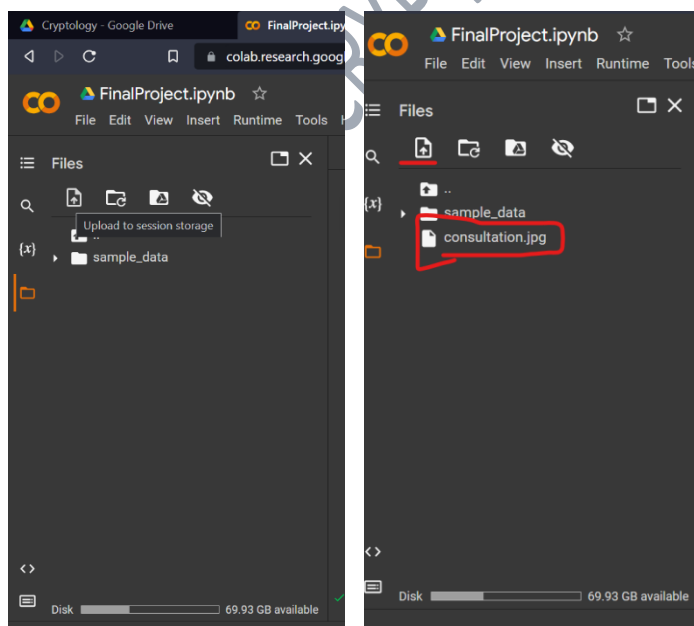
Once you click Run all from the Runtime, all cells of the file will run.

### Step 3:

Now time for uploading the image inside the working directory.



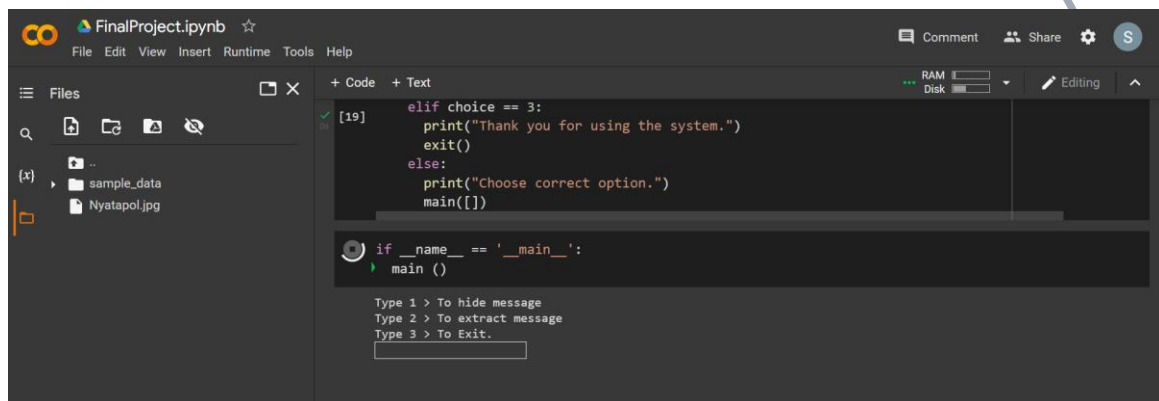
Click the file button in the system. The system will be shown in the above picture. Now you can either upload the file to the session or drag and drop it below the *sample\_data* repository.



The output files are also will store in this area. This is session storage. After the session ends the files are no longer available. Fortunately, we can download the available files in this area to our local machine.

#### Step 4 (Hide message):

Let's try to hide a message. Once we run the file (Runtime -> run all). The system will ask for a choice, we will type 1 to hide our message.



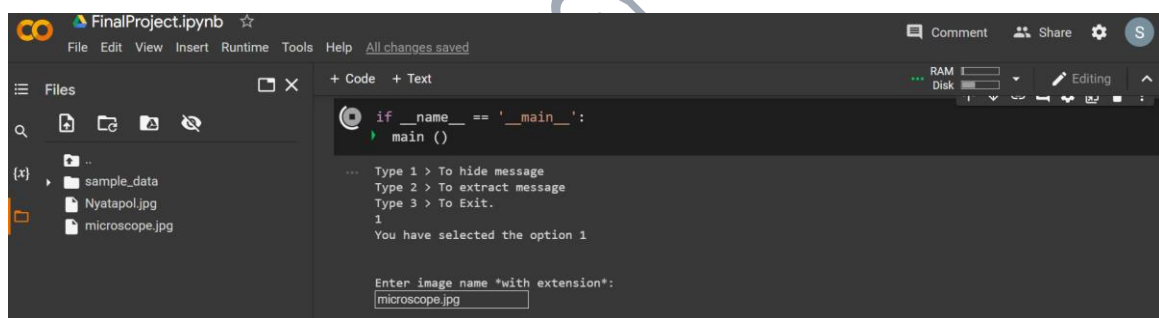
The screenshot shows a Jupyter Notebook interface with a file explorer on the left containing 'sample\_data' and 'Nyatapoli.jpg'. The code cell contains the following Python code:

```
[19] elif choice == 3:
    print("Thank you for using the system.")
    exit()
else:
    print("Choose correct option.")
    main([])

if __name__ == '__main__':
    main ()
```

The output area shows the prompt: "Type 1 > To hide message", "Type 2 > To extract message", "Type 3 > To Exit." followed by an input field containing the number 1.

It's time to provide the image name with an extension. Then just press enter.

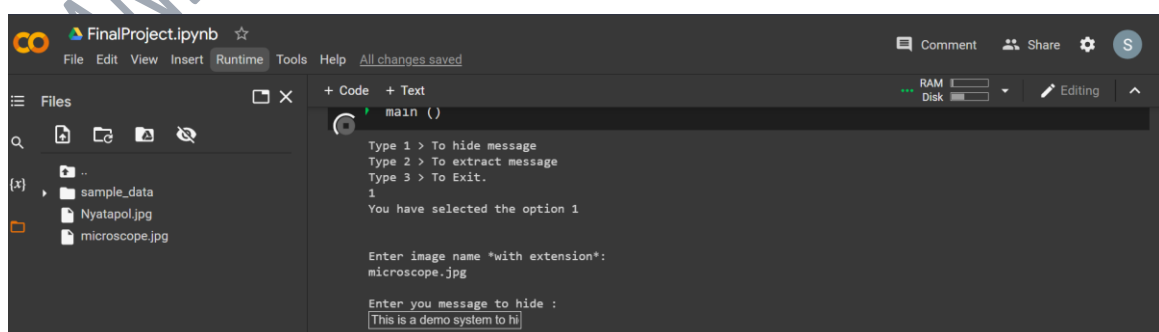


The screenshot shows the same Jupyter Notebook interface. The code cell now only contains the main function call:

```
if __name__ == '__main__':
    main ()
```

The output area shows the same menu as before, followed by the input '1' and the message "You have selected the option 1". Below this, a new prompt appears: "Enter image name \*with extension\*:" followed by an input field containing "microscope.jpg".

If the image name is correct (system finds image), it will ask you to provide the message (text). Just type the text and press enter.

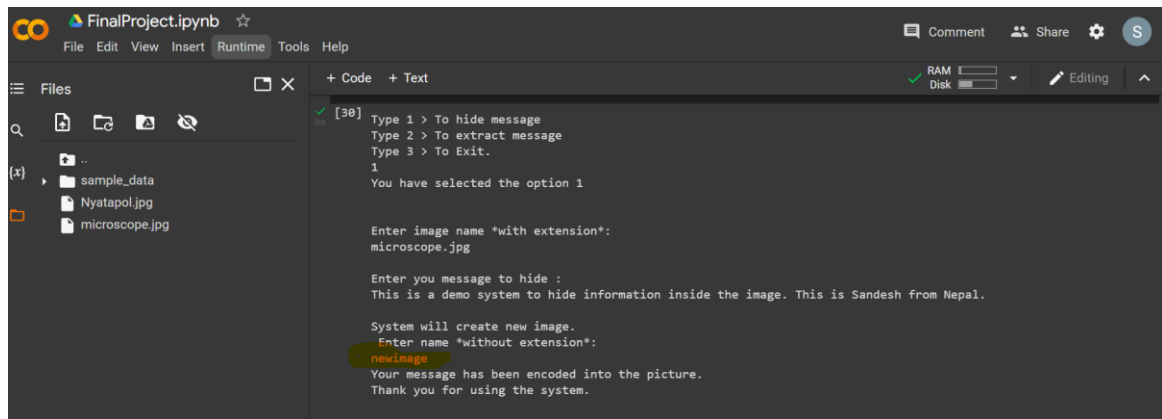


The screenshot shows the Jupyter Notebook interface with the 'Runtime' tab selected. The code cell contains the main function call:

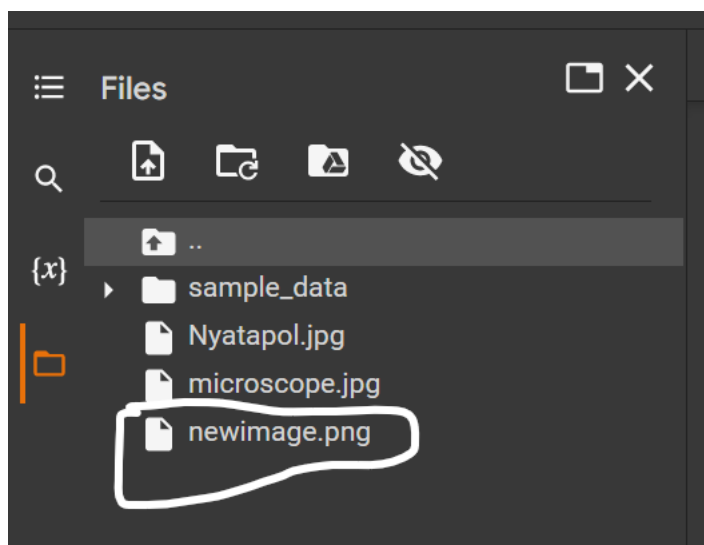
```
main ()
```

The output area shows the same menu, followed by the input '1' and the message "You have selected the option 1". Below this, the prompt "Enter image name \*with extension\*:" is followed by the input "microscope.jpg". A new prompt then appears: "Enter you message to hide :" followed by an input field containing "This is a demo system to hi".

Once it is done, you have to provide a new image name but this time **only name**, the output will be in png format. After pressing enter you can see the window like below;



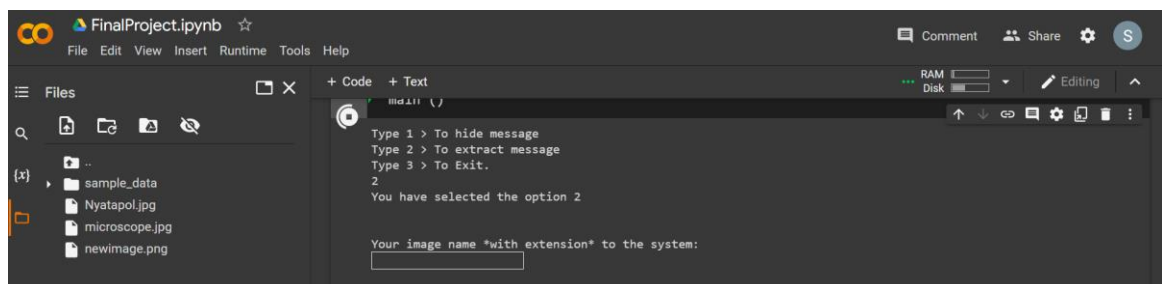
Just refresh the files to see the new image.



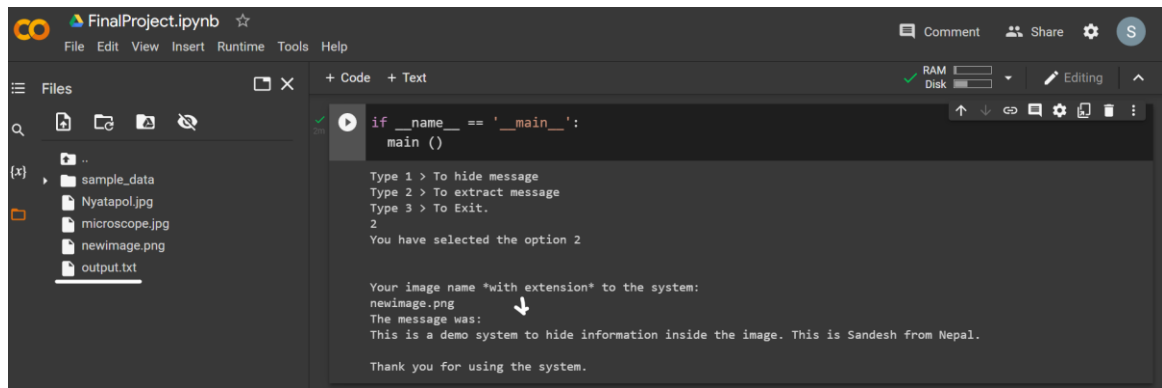
### Step 5 (Extract Message):

Once we are done with hiding the text inside the image, in the files we can find the new image. To extract the message we have to run the file again ( Runtime -> Run all). This time we need to select option 2. Just type 2 and enter for extraction.

The system will ask you to provide the image name **with its extension**. You can see the image in the files area. Type image name and press enter.



Once message extraction is done, you can refresh the file area the new text file will appear.



The screenshot shows a Jupyter Notebook titled 'FinalProject.ipynb'. The left sidebar displays a file explorer with a folder 'sample\_data' and files 'Nyatapol.jpg', 'microscope.jpg', 'newimage.png', and 'output.txt'. The main area contains a code cell with the following Python code:

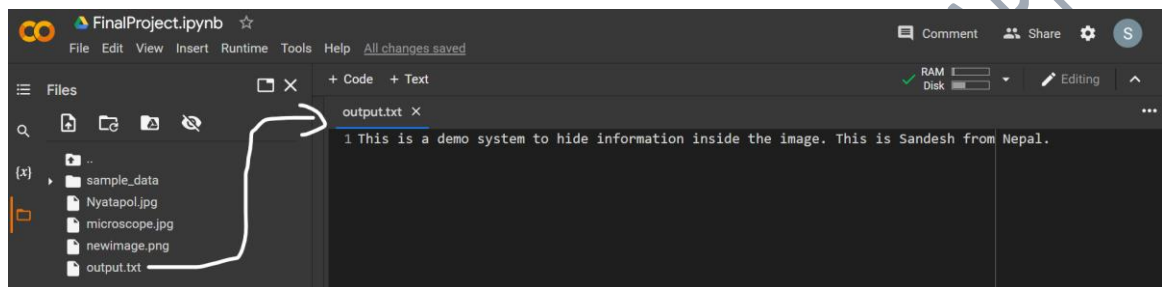
```
if __name__ == '__main__':  
    main ()
```

The output of the code cell shows a simulated user interface with the following text:

```
Type 1 > To hide message  
Type 2 > To extract message  
Type 3 > To Exit.  
2  
You have selected the option 2  
  
Your image name *with extension* to the system:  
newimage.png  
The message was:  
This is a demo system to hide information inside the image. This is Sandesh from Nepal.  
  
Thank you for using the system.
```

An arrow points to the text 'newimage.png' in the output.

Now you can click on the text file to see its contents on it.



The screenshot shows the same Jupyter Notebook interface, but the 'output.txt' file is now open in a text editor. The text editor displays the following content:

```
1 This is a demo system to hide information inside the image. This is Sandesh from Nepal.
```

A white arrow points from the 'output.txt' file in the file explorer to the text editor.

## Functionalities

In the system, there are mainly two functions one for hiding the message and another for extracting that message.

a) `hide_message()`

This function is responsible for hiding the message. It asks the user to provide an image name with an extension (mainly compatible with jpg and png) once the image successfully loads to the system it again asks the user to provide the message use wishes to hide. It provides the reference image and message data to the *encode\_enc* function which handles the rest of the operations.

b) `extract_message()`

Once we are done with hiding the message in the image, to extract that message this function comes to the work. This function asks the user to provide an image containing a message (the image created by the system, hiding some message on it) and reads the message from hidden pixels in the image, and returns textual data.

Other related functions:

a) `generate_binary_8bitdata(data)`

This function gives us encoding data into the 8-bit binary form using the ASCII value of characters.

b) `def write_in_pixel(pixel, msg)`

This function takes pixel data and message data. Pixels are modified according to the 8-bit binary data and finally returned to form a new image. This is the core function where everything related to modifying the image pixels is done. We extract 3 pixels at once because every 3 pixels contain binary data, and then we work on those extracted pixels to set value 1 for odd and value 0 for even. In 8th pixel of every set tells us whether to stop or read further whereas 0 indicates there is more message and 1 indicates no more message.



RGB Codes of Pixels

R-144 G-56 B-17	R-144 G-212 B-228	R-18 G-112 B-9	R-90 G-173 B-228
R-107 G-112 B-129	R-202 G-217 B-206	R-250 G-135 B-241	R-250 G-135 B-10
R-0 G-100 B-255	R-255 G-0 B-0	R-0 G-255 B-255	R-246 G-168 B-158
R-0 G-0 B-0	R-0 G-255 B-0	R-238 G-238 B-238	R-170 G-255 B-238

Original Image with rgb code of each pixel  
Source: [www.programiz.com](http://www.programiz.com)

Coded Image

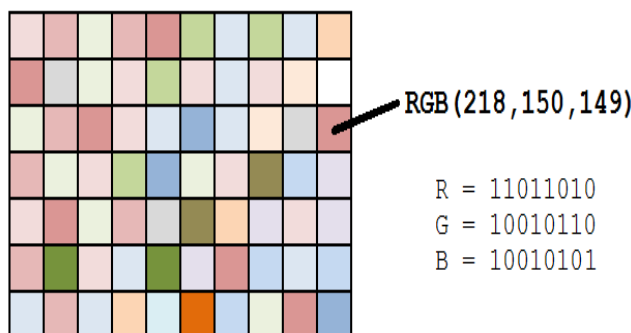


RGB Codes of Pixels

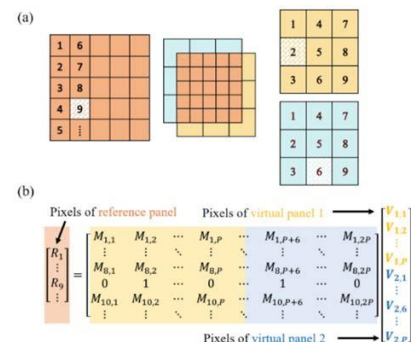
R-145 G-57 B-16	R-145 G-212 B-228	R-18 G-113 B-8	R-90 G-172 B-228
R-106 G-112 B-129	R-202 G-217 B-207	R-251 G-134 B-241	R-251 G-135 B-11
R-0 G-100 B-254	R-254 G-1 B-0	R-1 G-254 B-254	R-247 G-169 B-158
R-1 G-1 B-1	R-1 G-254 B-1	R-238 G-239 B-239	R-170 G-254 B-239

Coded image with rgb code of each pixel  
Source: [www.programiz.com](http://www.programiz.com)

- ❖ Digital image constructed by 2-Dimensions array of the pixels.
- ❖ Each pixel is identified by its coordinates (co-x, co-y) and its value and characterized by matrix size, pixel depth, and resolution.
- ❖ The size of the matrix is determined by the number of columns (cn) and the number of rows (rn) of the image matrix ( $cn \times rn$ ).
- ❖ Matrix size is may vary because it is selected by the operator.
- ❖ Higher the matrix dimension higher the resolution of the image.



Pixel color binary logic  
Source: [dataquest.io](http://dataquest.io)



Pixel illustration and (b) mapping matrix construction for image  
Source: [researchgate.net](http://researchgate.net)

c) `encode_enc(newimg, msg)`

This function allows us to provide our image to the system along with the message that we are going to hide on. But this is not the exact place where we take the input. We call this function from another function. We find the size of the pixel and create a pixel tuple (python object separated by comma but can't change like a list) and put those values to the pixel for further modification. Now it allows finding pixels efficiently.



d) `remove(string)`

It is just a simple function to remove the leading and trailing spaces. We do not have more specific work on this function.

e) `text_to_file(msg)`

Once we extract the message from the image, this function creates a new .txt file in the working directory. Once we successfully extract the message, we need to refresh the directory to see the output file.

f) `main()`

This is the main function where all functions are handled appropriately. It gives the system functionality to the user. User can select what s/he wants to do on the system. It asks the user for their choice and calls the related functions.

SANDESH CRYPTOLOGY STEGANOGRAPHY

## References

1. Simplelearn, 2022. [online] Available at: <<https://www.simplilearn.com/what-is-steganography-article>> [Accessed 13 May 2022].
2. Programiz Blog. 2022. *Steganography: How to Hide Information inside Pictures*. [online] Available at: <<https://www.programiz.com/blog/steganography-hiding-information-inside-pictures/>> [Accessed 14 May 2022].
3. Datagenetics.com. 2022. *Steganography*. [online] Available at: <<https://datagenetics.com/blog/march12012/index.html>> [Accessed 14 May 2022].
4. Tao Zhan, 2022. [online] Available at: <[https://www.researchgate.net/figure/a-Pixel-illustration-and-b-mapping-matrix-construction-for-image-generation\\_fig5\\_333097779](https://www.researchgate.net/figure/a-Pixel-illustration-and-b-mapping-matrix-construction-for-image-generation_fig5_333097779)> [Accessed 16 May 2022].

SANDESH CRYPTOLOGY STEGANOGRAPHY