A white background with blue text

Description automatically generated

# Abstract:

This coursework presents the development of a steganography application using Python, focusing on the least significant bit (LSB) technique for text embedding within images. The project offers both Command-Line Interface (CLI) and Graphical User Interface (GUI) modes for user interaction, making it accessible to a wide range of users. Leveraging the Python Imaging Library (PIL) for image manipulation and the stegano library for LSB-based steganography operations, the application enables users to hide and reveal confidential text seamlessly. The main objective is to explore steganography principles, implement the LSB technique effectively, and create an interactive tool that showcases Python programming proficiency. Through this coursework, students gain insights into data concealment methods and strengthen their understanding of information security concepts.

# Objective:

The primary objective of this coursework is to develop a steganography application using Python, emphasizing the least significant bit (LSB) technique for embedding text within images. It aims to provide both CLI and GUI modes, facilitating user-friendly interaction. By utilizing the Python Imaging Library (PIL) for image processing and integrating the stegano library for LSB steganography, the project showcases proficiency in external library usage. The coursework also focuses on enhancing Python programming skills, fostering a deeper comprehension of data concealment principles, and promoting awareness of information security applications. Ultimately, the application serves as an educational tool for students to explore steganography, its practical implementation, and its role in ensuring data privacy and secure communication.

Contents

[Abstract: 2](#_Toc142611475)

[Objective: 2](#_Toc142611476)

[Procedure 5](#_Toc142611477)

[Importing Libraries: 5](#_Toc142611478)

[GUI Functions: 7](#_Toc142611479)

[Main Function: 9](#_Toc142611480)

[Script Execution: 10](#_Toc142611481)

[The working: Execution 11](#_Toc142611482)

[Conclusion 14](#_Toc142611483)

# Table of Figures:

[Figure 1 5](#_Toc142611501)

[Figure 2 6](#_Toc142611502)

[Figure 3 6](#_Toc142611503)

[Figure 4 7](#_Toc142611504)

[Figure 5 7](#_Toc142611505)

[Figure 6 8](#_Toc142611506)

[Figure 7 9](#_Toc142611507)

[Figure 8 9](#_Toc142611508)

[Figure 9 10](#_Toc142611509)

[Figure 10 11](#_Toc142611510)

[Figure 11 11](#_Toc142611511)

[Figure 12 12](#_Toc142611512)

[Figure 13 12](#_Toc142611513)

[Figure 14 12](#_Toc142611514)

[Figure 15 13](#_Toc142611515)

[Figure 16 13](#_Toc142611516)

[Figure 17 13](#_Toc142611517)

# Procedure

## Importing Libraries:

The script starts by importing the necessary libraries:

A black screen with white text

Description automatically generated

Figure

- `tkinter`: This library provides the GUI functionality.

- `filedialog` and `messagebox`: These modules are part of tkinter and used to handle file dialogs and message boxes, respectively.

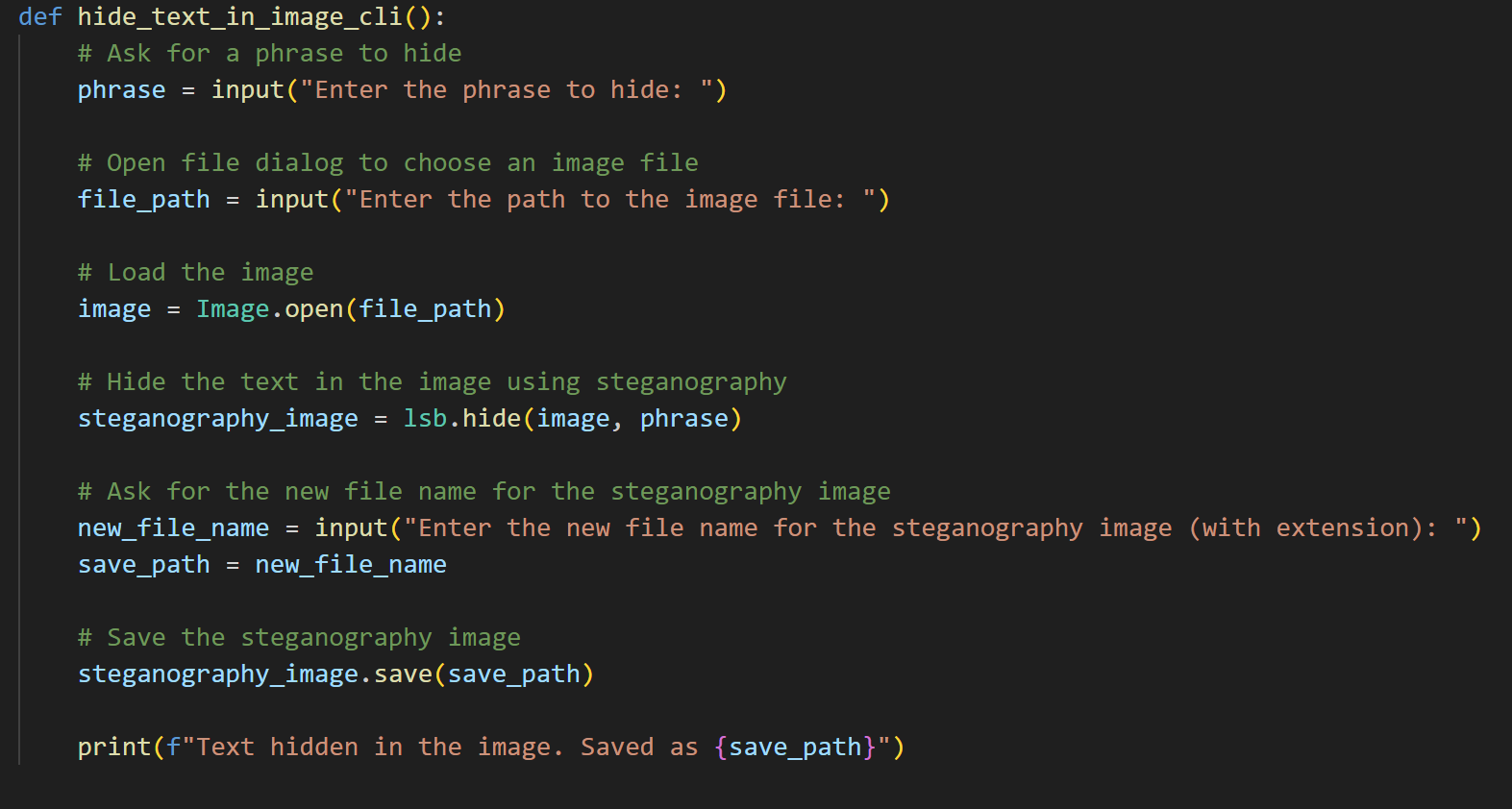
- `PIL.Image`: This module from the Python Imaging Library (PIL) is used to work with images.

- `stegano.lsb`: This is the module from the Stegano library that allows us to perform steganography using the least significant bit (LSB) technique.

**Command-Line Interface (CLI) Functions:**

Next, the script defines four functions that implement the steganography functionality in the command-line interface (CLI):

These functions perform the following tasks:



Figure

- `hide\_text\_in\_image\_cli()`: This function allows users to hide text in an image. It prompts the user to enter the phrase to hide and the path to the image file. It then loads the image, hides the text using steganography, and saves the new steganography image.

A computer screen with text

Description automatically generated

Figure

- `decode\_image\_cli()`: This function allows users to decode hidden text from a steganography image. It prompts the user to enter the path to the steganography image file, loads the steganography image, and reveals the hidden text if present.

## GUI Functions:

The script defines two functions that implement the steganography functionality in the graphical user interface (GUI):

These functions use tkinter to create GUI windows for the user to interact with. They perform the following tasks:



Figure

A screen shot of a computer program

Description automatically generated

Figure

- `hide\_text\_in\_image\_gui()`: This function creates a GUI window with an input field for the phrase and a button to choose an image file. After selecting the image file, it hides the text in the image using steganography and saves the new steganography image.

A screen shot of a computer code

Description automatically generated

Figure

- `decode\_image\_gui()`: This function creates a GUI window with a button to choose a steganography image file. After selecting the steganography image file, it decodes and displays the hidden text if present.

## Main Function:

The `main()` function is the entry point of the script:

A screenshot of a computer program

Description automatically generated

Figure

A screen shot of a computer program

Description automatically generated

Figure

It asks the user to choose between CLI mode or GUI mode:

- If the user chooses CLI mode, it presents a menu with two options: hide text in an image or decode hidden text from a steganography image. Based on the user's choice, it calls the respective CLI functions.

- If the user chooses GUI mode, it creates a main window with buttons for hiding text and decoding text. When the user clicks on the buttons, it calls the corresponding GUI functions.

# Script Execution:

The script checks if it is being executed directly (not imported as a module) using the following block of code:

A black background with white text

Description automatically generated

Figure

If the script is executed directly (e.g., `python stegg.py`), it calls the `main()` function to start the application.

To use the script:

1. Run the script from the command line or terminal.

2. It will ask you to choose between CLI or GUI mode.

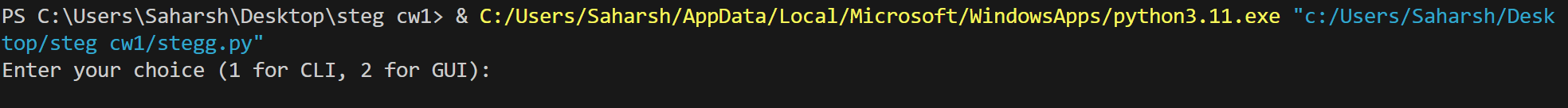
3. Based on your choice, follow the instructions to either hide text in an image or decode hidden text from a steganography image.

## The working: Execution

Sure, let's go through how the code executes based on the user's input:

1. Choosing Mode:

When you run the script, it will prompt you to choose between two modes: CLI mode (Command-Line Interface) or GUI mode (Graphical User Interface). It will display a menu like this:



Figure

2. Selecting Mode:

- If you enter `1` and press Enter, it means you've chosen CLI mode. It will then present you with another menu:

A screenshot of a computer

Description automatically generated

Figure

```

- If you enter `2` and press Enter, it means you've chosen GUI mode. The script will proceed to open a GUI window with buttons for hiding text and decoding text.

3. CLI Mode - Hiding Text:

If you chose to run in CLI mode and then selected option `1` to hide text in an image, it will prompt you to enter the phrase to hide:

A black background with white text

Description automatically generated

Figure

You will type the phrase you want to hide and press Enter. It will then prompt you to enter the path to the image file:

A black background with white text

Description automatically generated

Figure

You should provide the path to the image file (e.g., `C:\path\to\image.jpg`) and press Enter. The script will load the image, hide the text in the image using steganography, and save the new steganography image.

Finally, it will ask you to enter the new file name for the steganography image:

A black background with white text

Description automatically generated

Figure

You can provide the desired file name for the steganography image (e.g., `steganography\_image.png`) and press Enter. The steganography image will be saved with the hidden text, and the process will be complete.

4. CLI Mode - Decoding Text:

If you chose to run in CLI mode and then selected option `2` to decode hidden text from a steganography image, it will prompt you to enter the path to the steganography image file:

A screenshot of a computer

Description automatically generated

Figure

You should provide the path to the steganography image file (e.g., `C:\path\to\steganography\_image.png`) and press Enter. The script will load the steganography image, decode the hidden text, and display it on the command line.

5. GUI Mode - Hiding Text:

If you chose to run in GUI mode and selected the "Hide text in an image" button, a graphical window will appear. In the window, you will see a text entry field where you can type the phrase to hide. There will be a button labeled "Hide Text in Image." After typing the phrase, you can click the button to choose the image file from your system using a file dialog. Once the image is chosen, the script will hide the text in the image and save the steganography image with the hidden text.

A screenshot of a computer

Description automatically generated

Figure

6. GUI Mode - Decoding Text:

If you chose to run in GUI mode and selected the "Decode hidden text from a steganography image" button, a graphical window will appear. In the window, there will be a button labeled "Decode Image." Clicking this button will allow you to choose a steganography image file from your system using a file dialog. Once the image is chosen, the script will decode the hidden text and display it in a message box.

A computer screen shot of a computer

Description automatically generated

Figure

That's how the code works in both CLI and GUI modes. Depending on the user's input and choices, it either prompts for input in the command line or displays a graphical user interface for interactions.

# Conclusion

In this coursework, we explored the concept of steganography and developed a steganography application using Python. Steganography involves hiding secret information within seemingly innocuous data, such as images, without raising suspicion. Our application focused on employing the least significant bit (LSB) technique to embed text within images.

The implementation utilized essential Python libraries, such as `PIL` for image manipulation and `stegano` for LSB steganography. Users could interact with the application in two modes: Command-Line Interface (CLI) and Graphical User Interface (GUI). The CLI mode prompted users to input the text to be hidden and the image file to be used for steganography. In contrast, the GUI mode provided a user-friendly window for text input and image selection.

Throughout the project, we gained valuable insights into image processing, user interface design, and the fundamental principles of steganography. However, it is crucial to note that our implementation represents a basic application and may not be suitable for high-security scenarios.

In conclusion, this coursework successfully introduced steganography and demonstrated a practical Python-based application. It serves as a stepping stone for further exploration of information concealment techniques. By combining Python's versatility and relevant libraries, we created an interactive steganography tool with potential applications in data privacy, digital watermarking, and secure communication.