Student’s Details :

Name - Aman Singh

SkillsBuild Email id - singh4444aman@gmail.com

College Name- Rungta College of Engineering and Technology, Bhilai

College State - Chhattisgarh

Internship Domain/ Start and End Date - **CyberSecurity / 13 October**2023 to **26 November** 2023

CYBERSECURITY AND IMAGE STEGANOGRAPHY

Hiding message in an image

Efficiency, productivity, and better results

Agenda

Technology's positive, lasting impact on businesses and the workplace

01 Project Overview

02 Who are the end users?

03 Solution

04 How did you customize the project and make it your own

05 Modeling

06 Results

07 Links

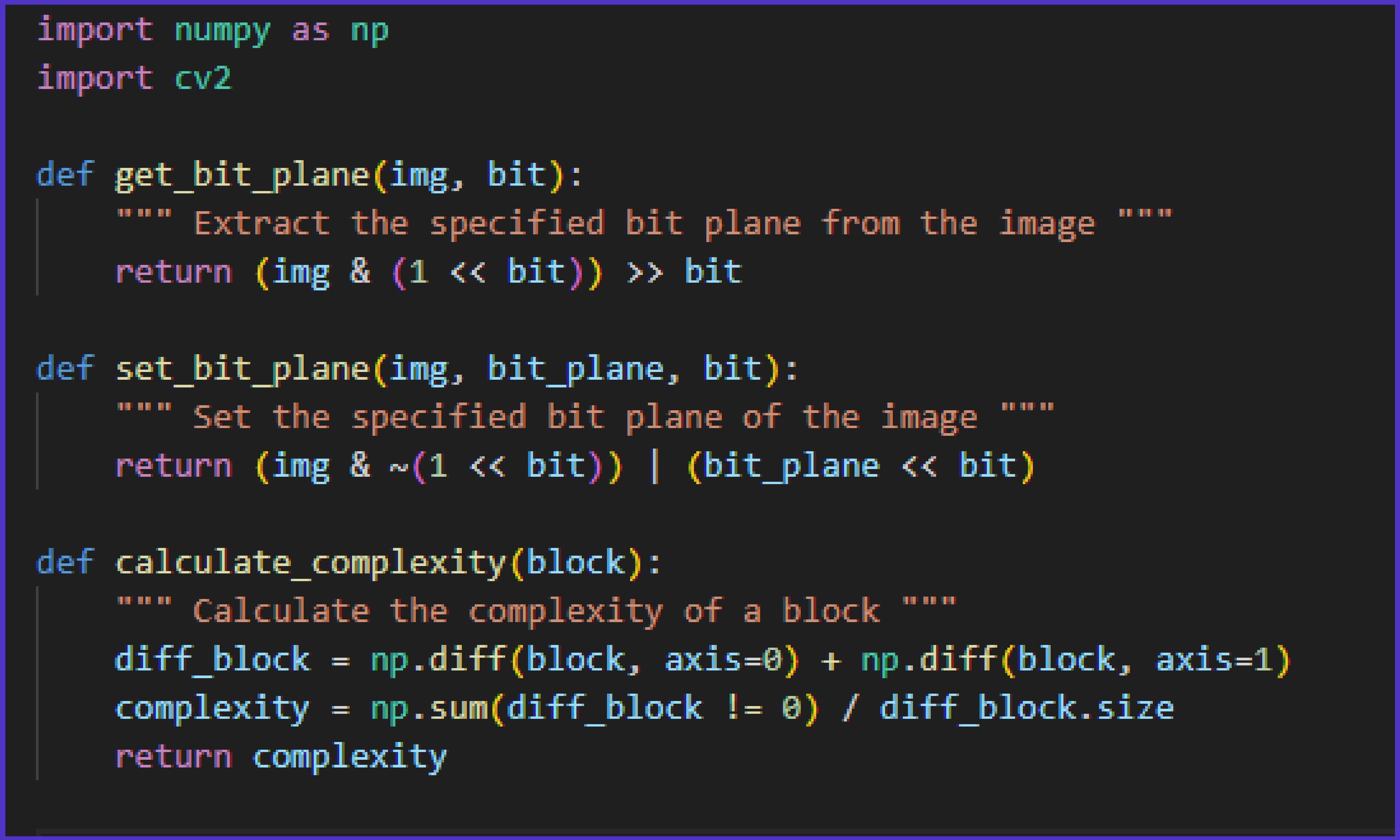
Project Overview

Steganography, the art of hiding information in plain sight, is a field that has gained significant attention in the digital age. Bit-Plane Complexity Segmentation (BPCS) Steganography is a sophisticated method in this domain, offering a robust way to conceal large amounts of data within digital images with minimal perceptual distortion. This project aims to develop and implement a BPCS steganography system, exploring its potential and limitations in digital communication and data security.

Who are the End Users

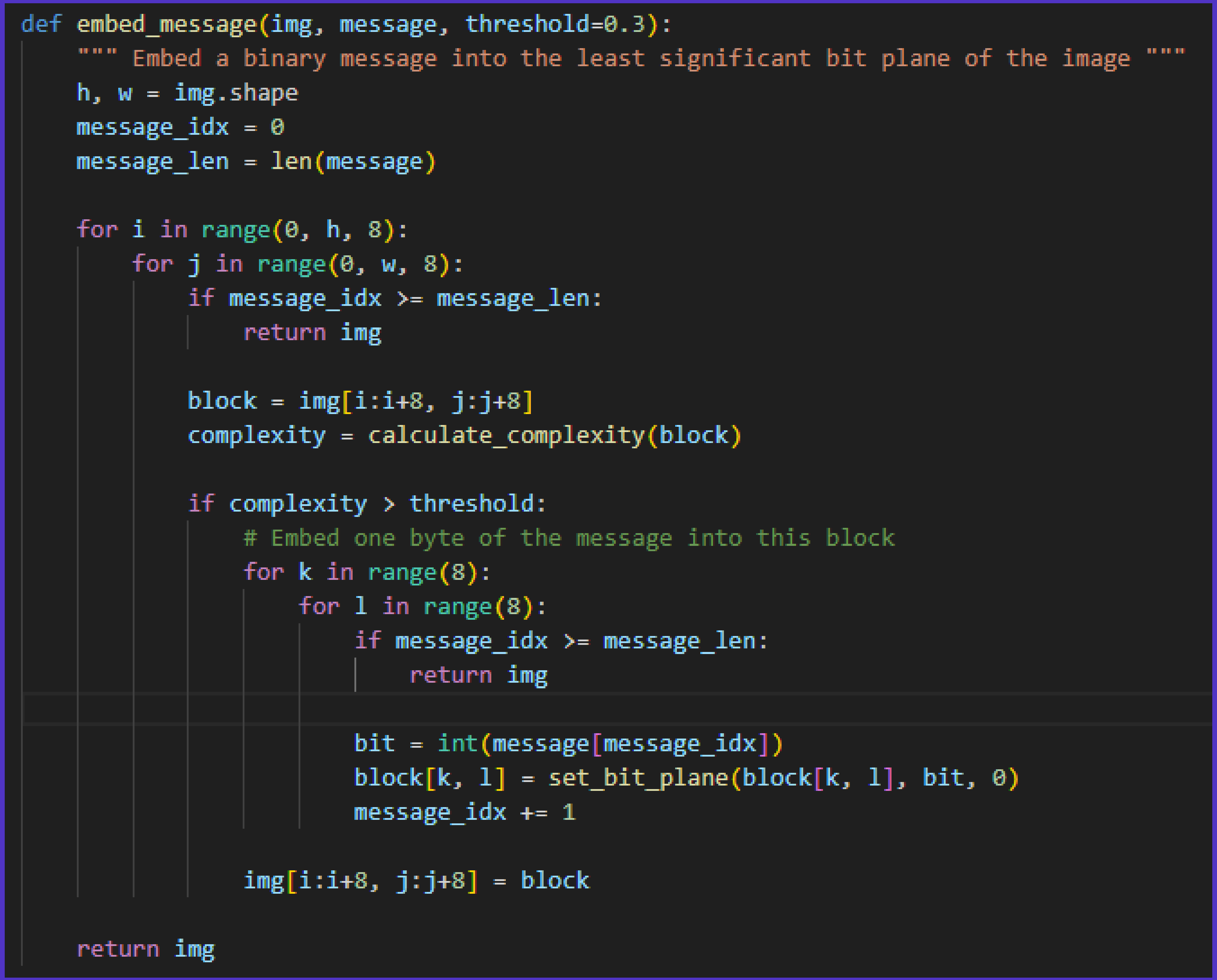
- **General Public**: Suitable for individuals seeking enhanced privacy in personal communications.
- **Corporate Sector:** Employed for confidential business communications and digital watermarking of intellectual property
- **Data Security Professionals**: Utilized by cybersecurity firms and forensic analysts for secure data transfer and cybercrime investigations.
- **Government and Defense**: Applied in military and intelligence operations for covert communications and counterintelligence.
- **Academic and Research Institutions:** Beneficial for researchers in digital security and educational use in computer science and digital forensics.

Solutions



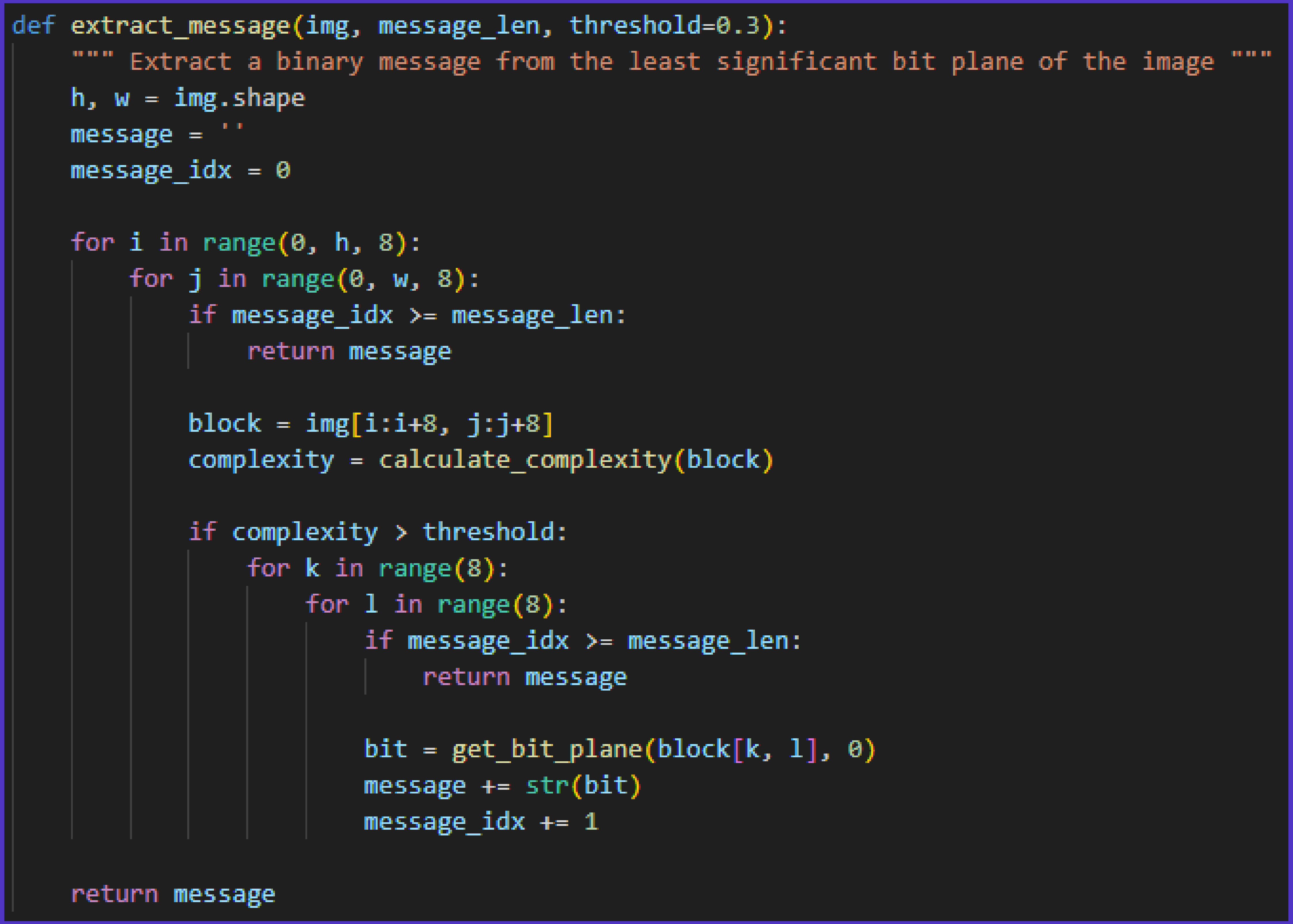
```python
import numpy as np
import cv2

def get_bit_plane(img, bit):
    """ Extract the specified bit plane from the image """
    return (img & (1 << bit)) >> bit

def set_bit_plane(img, bit_plane, bit):
    """ Set the specified bit plane of the image """
    return (img & ~(1 << bit)) | (bit_plane << bit)

def calculate_complexity(block):
    """ Calculate the complexity of a block """
    diff_block = np.diff(block, axis=0) + np.diff(block, axis=1)
    complexity = np.sum(diff_block != 0) / diff_block.size
    return complexity
```

Solutions



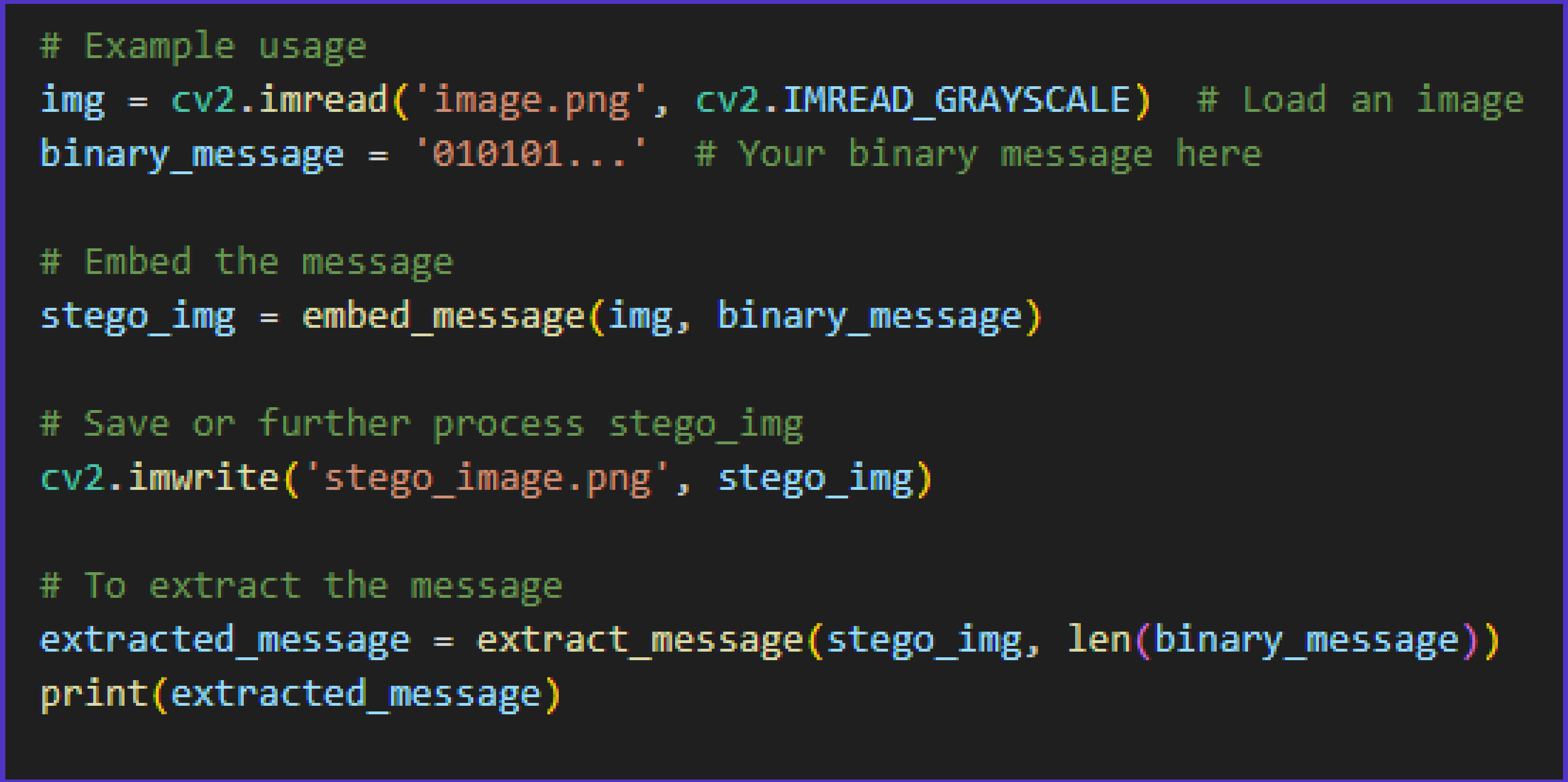
```python
def embed_message(img, message, threshold=0.3):
    """ Embed a binary message into the least significant bit plane of the image """
    h, w = img.shape
    message_idx = 0
    message_len = len(message)

    for i in range(0, h, 8):
        for j in range(0, w, 8):
            if message_idx >= message_len:
                return img

            block = img[i:i+8, j:j+8]
            complexity = calculate_complexity(block)

            if complexity > threshold:
                # Embed one byte of the message into this block
                for k in range(8):
                    for l in range(8):
                        if message_idx >= message_len:
                            return img

                        bit = int(message[message_idx])
                        block[k, l] = set_bit_plane(block[k, l], bit, 0)
                        message_idx += 1

                img[i:i+8, j:j+8] = block

    return img
```

Solutions



```python
def extract_message(img, message_len, threshold=0.3):
    """ Extract a binary message from the least significant bit plane of the image """
    h, w = img.shape
    message = ''
    message_idx = 0

    for i in range(0, h, 8):
        for j in range(0, w, 8):
            if message_idx >= message_len:
                return message

            block = img[i:i+8, j:j+8]
            complexity = calculate_complexity(block)

            if complexity > threshold:
                for k in range(8):
                    for l in range(8):
                        if message_idx >= message_len:
                            return message

                        bit = get_bit_plane(block[k, l], 0)
                        message += str(bit)
                        message_idx += 1

    return message
```

Solutions



```
# Example usage
img = cv2.imread('image.png', cv2.IMREAD_GRAYSCALE)  # Load an image
binary_message = '010101...'  # Your binary message here

# Embed the message
stego_img = embed_message(img, binary_message)

# Save or further process stego_img
cv2.imwrite('stego_image.png', stego_img)

# To extract the message
extracted_message = extract_message(stego_img, len(binary_message))
print(extracted_message)
```

How did you customize the project and make it your own

01 Developed a unique variant of image steganography using the novel Bit-Plane Complexity Segmentation (BPCS) Steganography technique

02 Implemented the project in multiple development environments:

- Visual Studio Code
- PyCharm
- Google Colab

03 Shared the project on Discord for community engagement and feedback.

Deployed the project using GitHub, ensuring accessibility and version control.

04

Modeling

Presenting live not your thing? No worries! Record your Canva Presentation your audience can watch at their own pace.

Don't forget to delete or hide this page before presenting.

01 **Literature Review:** Conduct a comprehensive review of existing literature on steganography, with a focus on BPCS techniques, to build a strong theoretical foundation.

02 **Tool Development:**

- **Design:** Outline the architecture of the BPCS steganography tool, including the user interface, embedding and extraction algorithms, and error handling mechanisms.
- **Implementation:** Code the tool using a suitable programming language (e.g., Python) and libraries (e.g., OpenCV, NumPy).
- **Testing:** Perform unit testing and integration testing to ensure reliability and accuracy.

03 **Application Scenarios and Ethical Discussion:** Explore real-world scenarios where BPCS steganography could be beneficial, and discuss ethical considerations, particularly regarding privacy and security.

Modeling

Methodologies

04

Experimental Analysis:

- Capacity Testing: Determine the maximum amount of data that can be hidden without noticeable image degradation.
- Robustness Testing: Assess the system's performance against image compression, scaling, cropping, and other common image processing operations.
- Steganalysis Resistance: Evaluate the tool's ability to withstand detection by common steganalysis techniques.

Results

01 **Enhanced Understanding of Steganography:** The project has deepened our understanding of digital steganography, particularly the BPCS method, which is a sophisticated approach to hiding data within images.

02 **Advancement in Data Hiding Techniques:** By implementing a unique variant of BPCS, the project contributes to the advancement of data hiding techniques, showcasing the potential for high-capacity and secure information embedding in digital media.

03 **Tool Versatility and Accessibility:** The development across multiple platforms (Visual Studio Code, PyCharm, Google Colab) demonstrates the tool's versatility and ensures broader accessibility for different user groups

04 **Community Engagement and Feedback:** Sharing the project on platforms like Discord has facilitated community engagement, allowing for valuable feedback, which is crucial for iterative improvement and real-world applicability.

Once you're done, download your Canva Presentation in MP4

Results

5 **Innovation in Secure Communication**: The project underscores the importance and potential of steganography in secure communications, especially in scenarios where conventional encryption might draw undue attention.

6 **Ethical and Legal Implications:** By bringing attention to the capabilities of steganography, the project also highlights the need for ethical considerations and legal frameworks governing its use, especially in matters of privacy and security.

Links

GitHub link : https://github.com/mvpamansingh/Image-Steganography

Google Drive link : https://drive.google.com/drive/folders/1blCMraEURyGooeCGmlP1lj5biPNTraX8?usp=sharing

Resource link : https://en.wikipedia.org/wiki/BPCS-steganography#:~:text=BPCS%2Dsteganography%20(Bit%2DPlane,cover%2C%20or%20dummy%20data%22.

https://ieeexplore.ieee.org/document/9633914

THANK YOU

:)