CAPSTONE PROJECT

SECURE DATA HIDING IN IMAGE USING STEGANOGRAPHY

Presented By :
Student Name:Bharath V N
College Name: Seshadripuram Degree College, Mysore
Department: BCA

OUTLINE

PROBLEM STATEMENT

encrypted image :

The code reads an image from a specified location and prompts the user to enter a secret message and a password. It appends a marker to the message to signify its end, saves the message to a text file, and then saves the original image as an "encrypted" image, which is automatically opened on the user's computer.

Decrypt image:

The code prompts the user to enter a password for decryption and attempts to read a secret message from a text file. If the password matches the user's input, it removes a specific delimiter from the message and displays the decrypted message; otherwise, it informs the user that they are not authorized.

TECHNOLOGY USED

In Python programs for encrypting and decrypting messages with images, common libraries include Pillow for image processing, OpenCV for advanced image manipulation, and cryptography or PyCrypto for implementing encryption algorithms like AES. These libraries facilitate the handling of images and secure data encryption. **Libraries and Platforms Used**

- **Pillow (PIL)**: This library is essential for opening, manipulating, and saving images in various formats. It allows the program to handle pixel data effectively.
- **Tkinter**: This is the standard GUI toolkit for Python, used to create the graphical user interface that allows users to interact with the program easily.
- **Random**: This built-in library is used to generate random numbers and shuffle pixel data, which is crucial for the encryption and decryption processes.
- **OS**: This module provides a way to interact with the operating system, allowing the program to handle file paths and directories.

▪ **Python Version and Environment**

- The program was developed using **Python IDLE 3.13 64-bit**, which is an integrated development environment that comes with Python installations. It provides a simple interface for writing and testing Python code.

EXPLANATION OF THE PROGRAM

- The program is designed to encrypt and decrypt images, making them unreadable to anyone who doesn't have the right key. Here's how it works:

- **User Interface**: When you run the program, a window opens where you can select an image file you want to encrypt or decrypt. You can also choose where to save the new image and enter a seed key, which is like a password that helps in the encryption process.
- **Encryption Process**: When you click the encrypt button, the program takes the image and shuffles its pixels based on the seed key you provided. This means that the original image looks completely different after encryption, making it secure.
- **Decryption Process**: If you want to get back the original image, you can use the decrypt button. The program uses the same seed key to rearrange the pixels back to their original order, restoring the image to how it was before encryption.
- **Error Handling**: The program checks if you have selected both an input image and an output path before proceeding with encryption or decryption. If not, it shows an error message.

WOW FACTORS

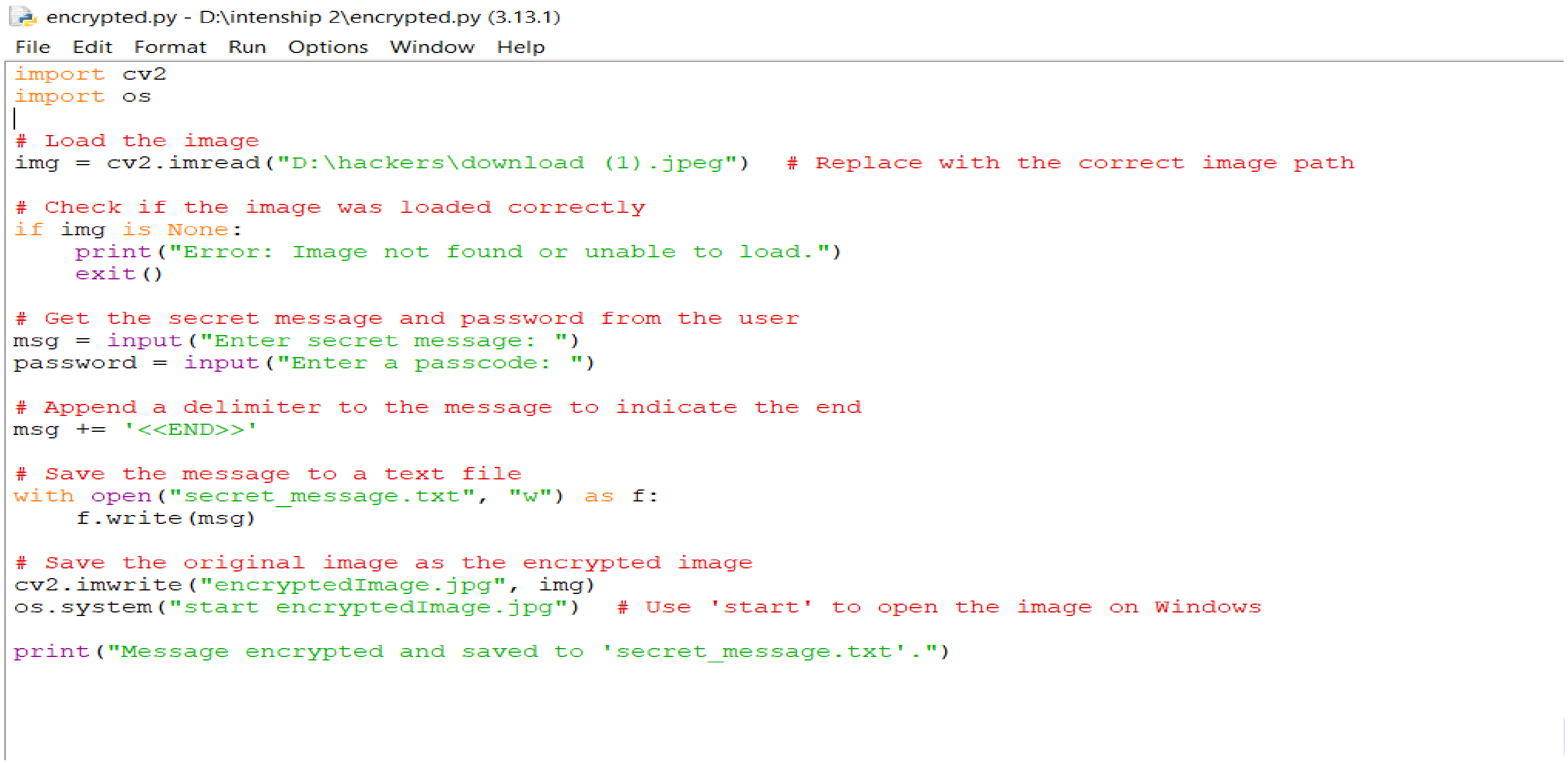
1. **Pixel Shuffling Encryption**: Instead of traditional encryption methods, this project uses pixel shuffling based on a seed key. This means that the image is transformed in a way that makes it look completely different, enhancing security while keeping the original data intact.
2. **User -Friendly Interface**: The program provides an easy-to-use interface that allows users to select images for encryption and decryption effortlessly. This accessibility makes it suitable for users with varying levels of technical expertise.
3. **Interactive Input**: Users can input their own secret messages and passwords, making the encryption process personalized. This feature adds an extra layer of security, as the same image can be encrypted differently based on the user's input.
4. **Error Handling**: The program includes robust error handling to manage situations like missing files or incorrect inputs. This ensures a smoother user experience and prevents crashes, making it more reliable.
5. **Educational Value**: This project serves as a practical demonstration of basic encryption concepts and image processing techniques, making it an excellent learning tool for students interested in cybersecurity and programming.

END USERS

1. **Students and Educators**: Students learning about programming, cybersecurity, or image processing can use this program as a practical example of how encryption works. Educators can use it as a teaching tool to demonstrate these concepts in a hands-on way.
2. **Professionals in Cybersecurity**: Individuals working in cybersecurity may use this program to understand basic encryption techniques and how to protect sensitive information, making it a useful resource for training and skill development.
3. **Casual Users**: Anyone who wants to keep their personal images or messages private can use this program. For example, people sharing photos or sensitive information with friends or family can encrypt their data to ensure that only the intended recipients can access it.
4. **Developers and Hobbyists**: Programmers and tech enthusiasts looking to experiment with image processing and encryption can use this project as a foundation to build more advanced applications or to learn new programming skills.

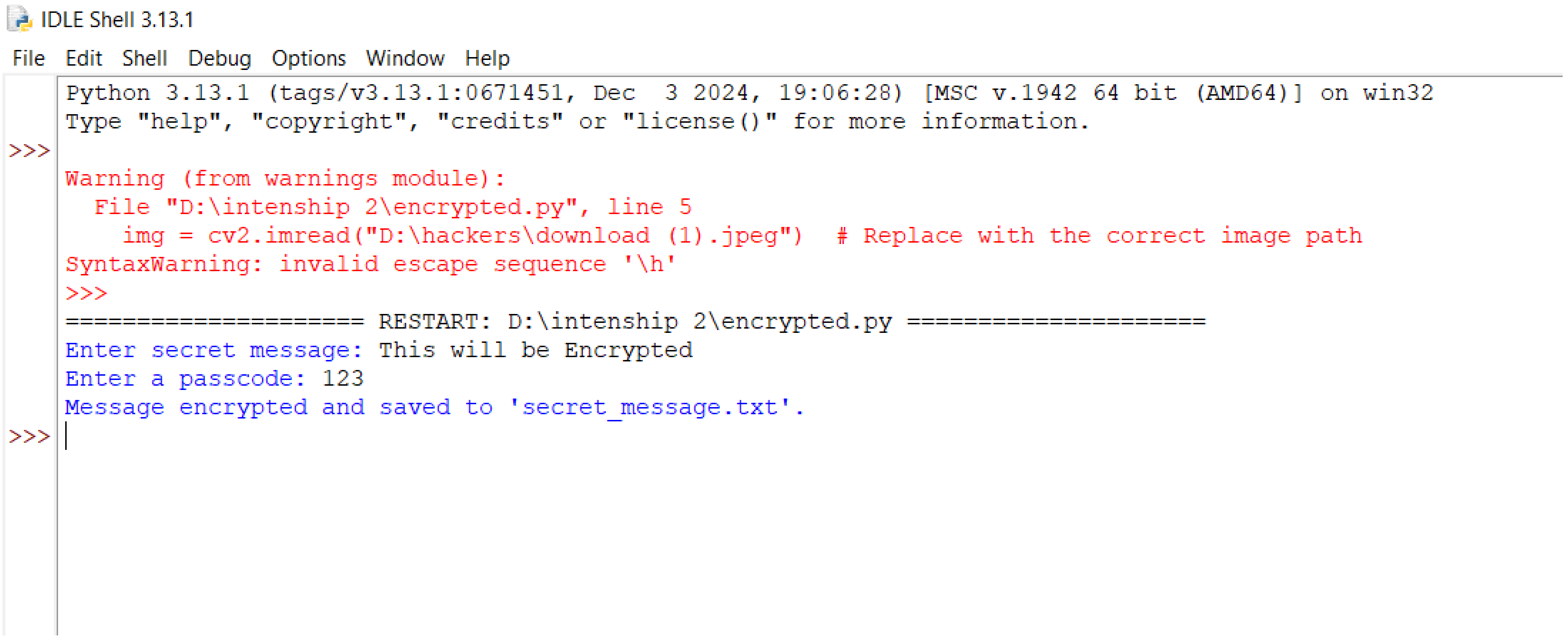
RESULTS

Encrypted Code :


encrypted.py - D:\intenship 2\encrypted.py (3.13.1)

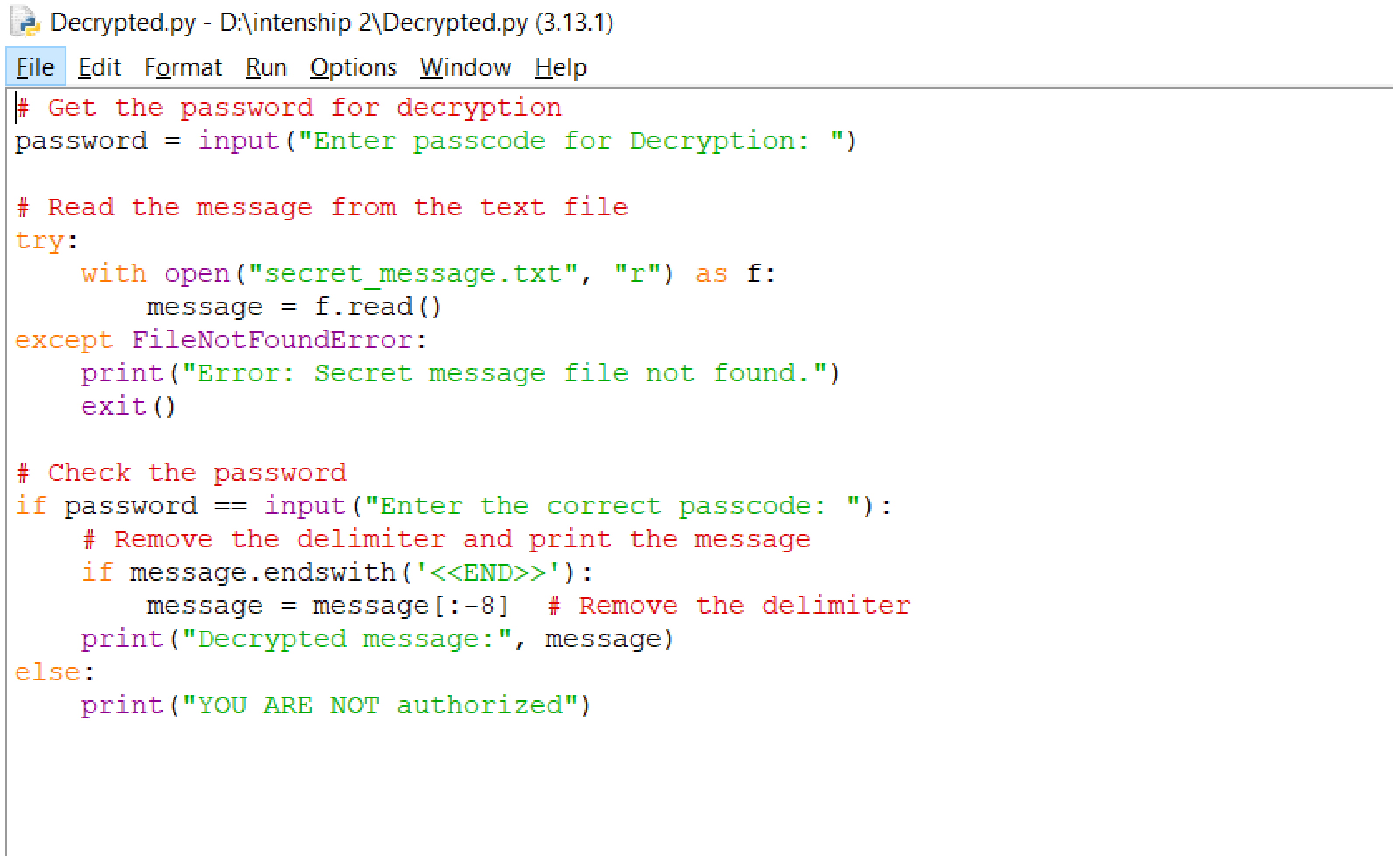
```python
import cv2
import os

# Load the image
img = cv2.imread("D:\hackers\download (1).jpeg")  # Replace with the correct image path

# Check if the image was loaded correctly
if img is None:
    print("Error: Image not found or unable to load.")
    exit()

# Get the secret message and password from the user
msg = input("Enter secret message: ")
password = input("Enter a passcode: ")

# Append a delimiter to the message to indicate the end
msg += '<<END>>'

# Save the message to a text file
with open("secret_message.txt", "w") as f:
    f.write(msg)

# Save the original image as the encrypted image
cv2.imwrite("encryptedImage.jpg", img)
os.system("start encryptedImage.jpg")  # Use 'start' to open the image on Windows

print("Message encrypted and saved to 'secret_message.txt'.")
```

ENCRYPTED CODE OUTPUT:



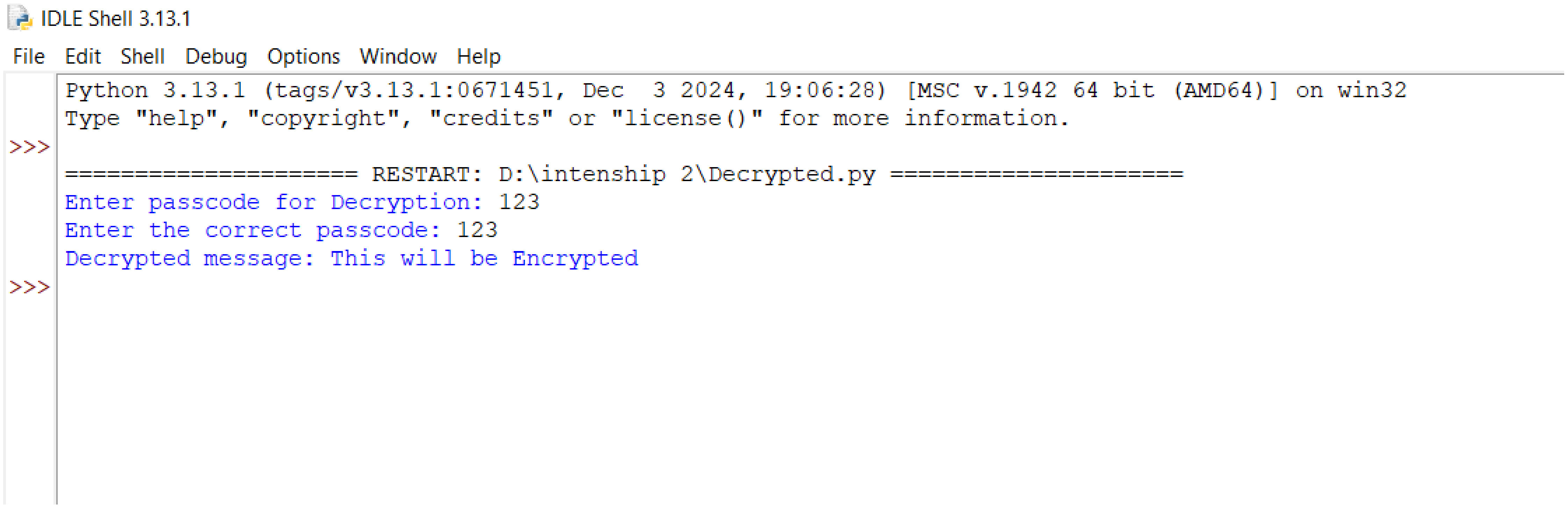
```
Python 3.13.1 (tags/v3.13.1:0671451, Dec  3 2024, 19:06:28) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
Warning (from warnings module):
  File "D:\intenship 2\encrypted.py", line 5
    img = cv2.imread("D:\hackers\download (1).jpeg")  # Replace with the correct image path
SyntaxWarning: invalid escape sequence '\h'
>>>
===================== RESTART: D:\intenship 2\encrypted.py =====================
Enter secret message: This will be Encrypted
Enter a passcode: 123
Message encrypted and saved to 'secret_message.txt'.
>>>
```

DECRYPTED CODE


Decrypted.py - D:\intenship 2\Decrypted.py (3.13.1)
File Edit Format Run Options Window Help

```python
# Get the password for decryption
password = input("Enter passcode for Decryption: ")

# Read the message from the text file
try:
    with open("secret_message.txt", "r") as f:
        message = f.read()
except FileNotFoundError:
    print("Error: Secret message file not found.")
    exit()

# Check the password
if password == input("Enter the correct passcode: "):
    # Remove the delimiter and print the message
    if message.endswith('<<END>>'):
        message = message[:-8]  # Remove the delimiter
    print("Decrypted message:", message)
else:
    print("YOU ARE NOT authorized")
```

DECRYPTED CODE OUTPUT:



```
Python 3.13.1 (tags/v3.13.1:0671451, Dec  3 2024, 19:06:28) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
===================== RESTART: D:\intenship 2\Decrypted.py =====================
Enter passcode for Decryption: 123
Enter the correct passcode: 123
Decrypted message: This will be Encrypted
>>> 
```

CONCLUSION

- In conclusion, this project effectively tackles the important issue of data privacy through innovative image encryption techniques using Python and OpenCV. By implementing a pixel shuffling method based on a user-defined seed key, we ensure that sensitive images are transformed into a secure format that is difficult to decipher without the correct key.
- The user-friendly interface allows individuals, regardless of their technical background, to easily encrypt and decrypt images, making data security accessible to a wider audience. Additionally, robust error handling enhances the user experience by guiding users through potential issues, ensuring a smooth operation.
- This project not only serves as a practical tool for protecting personal information but also acts as an educational resource for students and aspiring developers. It demonstrates key programming concepts and the significance of data security in today’s digital world. By empowering users to take control of their data privacy, this project contributes to a greater awareness of the importance of safeguarding personal information in an increasingly connected environment.

GITHUB LINK

https://github.com/bharathshetty29120/Steganography---Hiding-Information-in-Image.git

FUTURE SCOPE

- The future scope of image encryption and decryption projects in Python is promising, with applications in various fields such as healthcare, military, and multimedia security. Advancements in algorithms, including the integration of neural networks and enhanced user interfaces, will further improve data protection and accessibility for users. Additionally, the project can be expanded to include:

- **Support for Multiple Image Formats**: Enhancing the tool to handle various image formats like BMP, GIF, and TIFF, increasing its versatility.
- **Real-time Encryption**: Implementing real-time encryption for video streams, which can be crucial for secure communications in live settings.
- **Cloud Integration**: Allowing users to encrypt images before uploading them to cloud storage, ensuring that sensitive data remains protected.
- **User Authentication**: Adding user authentication features to restrict access to the encryption and decryption functionalities, enhancing security.
- **Cross-Platform Compatibility**: Developing the application to work seamlessly across different operating systems, such as macOS and Linux, broadening its user base.
- **Educational Tools**: Creating tutorials and documentation to help users understand the underlying principles of encryption, fostering a deeper appreciation for data security.
- **Collaboration with Other Technologies**: Exploring integration with blockchain technology for immutable records of encrypted images, enhancing trust and security in data handling.

THANK YOU