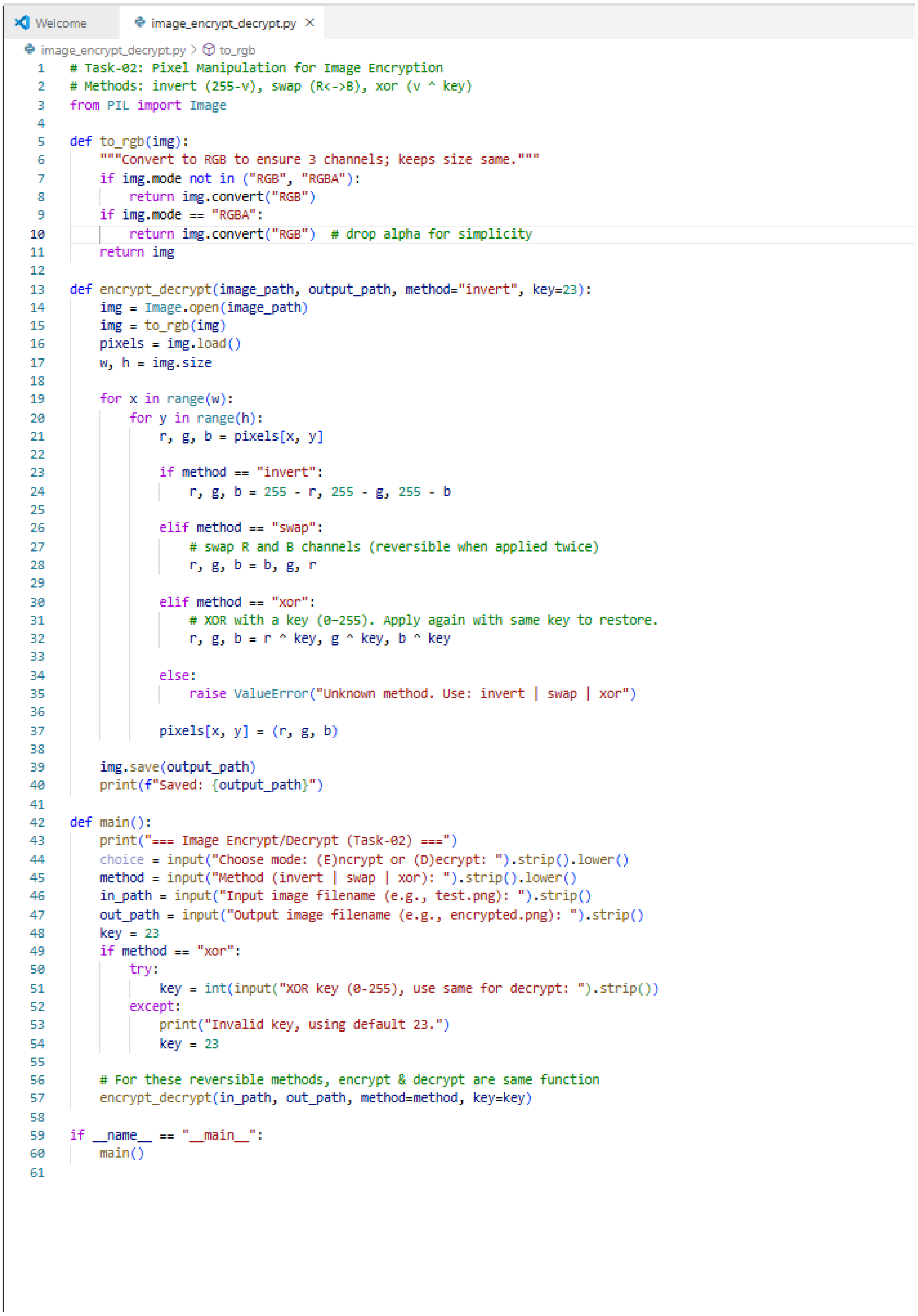
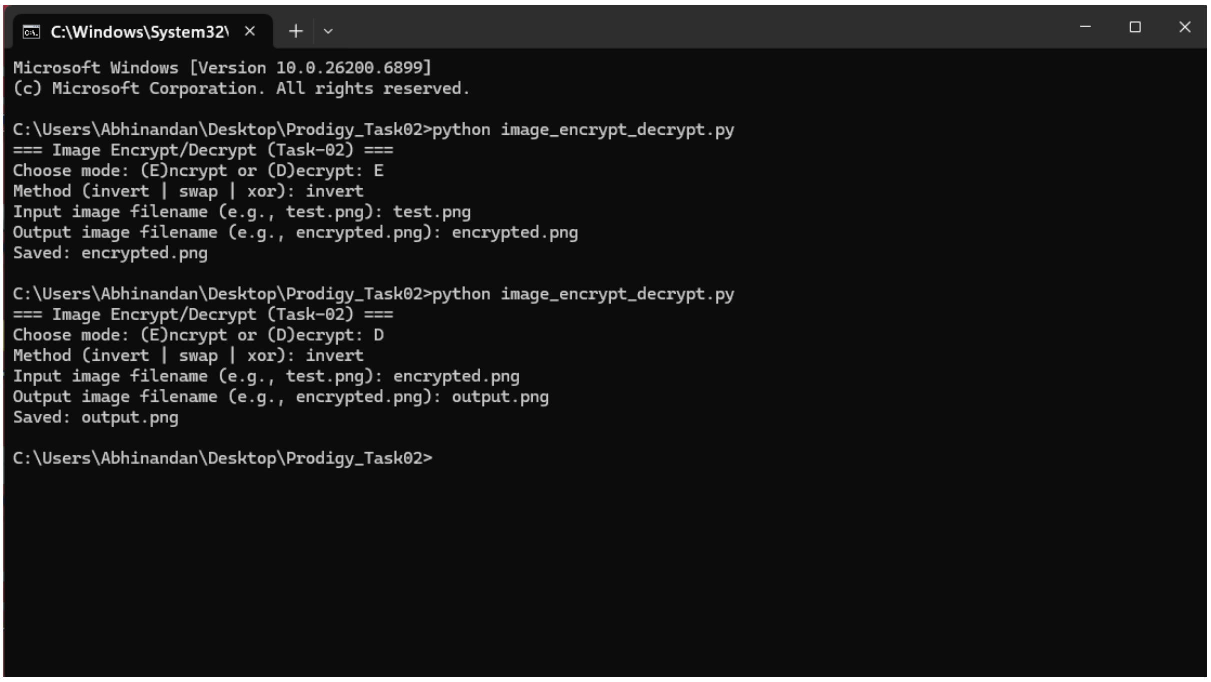
Title: Task-02 — Pixel Manipulation for Image Encryption (Python)
Objective: Build a simple image encryption/decryption tool using pixel operations.
Methods Used: invert (255−v), swap (R↔B), xor (v ^ key).
Tools: Python 3, Pillow.
Algorithm: Load image → loop through pixels → apply chosen operation → save.

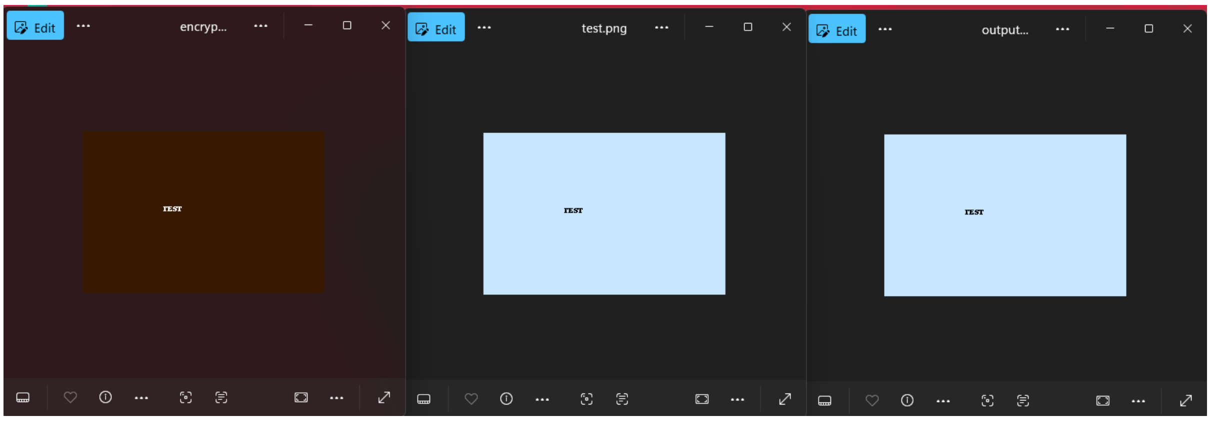
Screenshots:



```python
# Task-02: Pixel Manipulation for Image Encryption
# Methods: invert (255-v), swap (R<->B), xor (v ^ key)
from PIL import Image

def to_rgb(img):
    """Convert to RGB to ensure 3 channels; keeps size same."""
    if img.mode not in ("RGB", "RGBA"):
        return img.convert("RGB")
    if img.mode == "RGBA":
        return img.convert("RGB")  # drop alpha for simplicity
    return img

def encrypt_decrypt(image_path, output_path, method="invert", key=23):
    img = Image.open(image_path)
    img = to_rgb(img)
    pixels = img.load()
    w, h = img.size

    for x in range(w):
        for y in range(h):
            r, g, b = pixels[x, y]

            if method == "invert":
                r, g, b = 255 - r, 255 - g, 255 - b

            elif method == "swap":
                # swap R and B channels (reversible when applied twice)
                r, g, b = b, g, r

            elif method == "xor":
                # XOR with a key (0-255). Apply again with same key to restore.
                r, g, b = r ^ key, g ^ key, b ^ key

            else:
                raise ValueError("Unknown method. Use: invert | swap | xor")

            pixels[x, y] = (r, g, b)

    img.save(output_path)
    print(f"Saved: {output_path}")

def main():
    print("=== Image Encrypt/Decrypt (Task-02) ===")
    choice = input("Choose mode: (E)ncrypt or (D)ecrypt: ").strip().lower()
    method = input("Method (invert | swap | xor): ").strip().lower()
    in_path = input("Input image filename (e.g., test.png): ").strip()
    out_path = input("Output image filename (e.g., encrypted.png): ").strip()
    key = 23
    if method == "xor":
        try:
            key = int(input("XOR key (0-255), use same for decrypt: ").strip())
        except:
            print("Invalid key, using default 23.")
            key = 23

    # For these reversible methods, encrypt & decrypt are same function
    encrypt_decrypt(in_path, out_path, method=method, key=key)

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



```
Microsoft Windows [Version 10.0.26200.6899]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Abhinandan\Desktop\Prodigy_Task02>python image_encrypt_decrypt.py
=== Image Encrypt/Decrypt (Task-02) ===
Choose mode: (E)ncrypt or (D)ecrypt: E
Method (invert | swap | xor): invert
Input image filename (e.g., test.png): test.png
Output image filename (e.g., encrypted.png): encrypted.png
Saved: encrypted.png

C:\Users\Abhinandan\Desktop\Prodigy_Task02>python image_encrypt_decrypt.py
=== Image Encrypt/Decrypt (Task-02) ===
Choose mode: (E)ncrypt or (D)ecrypt: D
Method (invert | swap | xor): invert
Input image filename (e.g., test.png): encrypted.png
Output image filename (e.g., encrypted.png): output.png
Saved: output.png

C:\Users\Abhinandan\Desktop\Prodigy_Task02>
```



Conclusion: Images encrypted and restored successfully using reversible pixel manipulation.