Pixel Manipulation for Image Encryption Task-2E.py

```python
from PIL import Image
import os

def encrypt_image(input_path, output_path, key):
    try:
        image = Image.open(input_path)
        pixels = image.load()

        width, height = image.size

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

                # Encrypt by adding key and wrapping around using modulo 256
                r = (r + key) % 256
                g = (g + key) % 256
                b = (b + key) % 256

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

        image.save(output_path)
        print(f"✅ Image encrypted and saved as {output_path}")
    except Exception as e:
        print("Error during encryption:", e)

def decrypt_image(input_path, output_path, key):
    try:
        image = Image.open(input_path)
        pixels = image.load()

        width, height = image.size

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

                # Decrypt by subtracting key and wrapping around using modulo 256
                r = (r - key) % 256
                g = (g - key) % 256
                b = (b - key) % 256

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

        image.save(output_path)
        print(f"✅ Image decrypted and saved as {output_path}")
    except Exception as e:
        print("Error during decryption:", e)
```

```python

def main():
    print("=== Image Encryption Tool ===")
    print("1. Encrypt Image")
    print("2. Decrypt Image")

    choice = input("Enter your choice (1 or 2): ").strip()

    if choice not in ['1', '2']:
        print("Invalid choice.")
        return

    input_path = input("Enter input image path (e.g., input.jpg): ").strip()
    if not os.path.exists(input_path):
        print("File not found.")
        return

    output_path = input("Enter output image path (e.g., encrypted.png): ").strip()

    try:
        key = int(input("Enter encryption/decryption key (integer 1-255): "))
        if not (1 <= key <= 255):
            raise ValueError
    except ValueError:
        print("Invalid key. Must be an integer between 1 and 255.")
        return

    if choice == '1':
        encrypt_image(input_path, output_path, key)
    else:
        decrypt_image(input_path, output_path, key)

if __name__ == "__main__":
    main()

```