Project

ENIGMA MACHINE
WORLD WAR 2

Enigma machine developed in python.

INTRODUCTION

Today, I am delighted to stand before you to present a remarkable creation that showcases the intersection of history, cryptography, and computer science. It is my pleasure to introduce to you the Enigma Machine, implemented in Python.

In our modern age of digital communication, it is essential to understand the foundations upon which our secure communication systems are built. By recreating the Enigma Machine in Python, we have the opportunity to dive deep into the inner workings of this historic encryption device and gain valuable insights into the field of cryptography.

During this presentation, we will explore the various components of the Enigma Machine, including its rotors, reflector, and plugboard. We will discuss how these components work together to transform plain text into cipher text and vice versa. By examining the intricate mechanics behind the Enigma Machine, we can appreciate the challenges faced by cryptanalysts of the time and the immense computational power required to break its codes.

By understanding the historical significance of the Enigma Machine and its impact on cryptography, we can gain a deeper appreciation for the modern encryption algorithms that safeguard our digital communication today. This presentation aims to bridge the gap between past and present, demonstrating the evolution of cryptography and the enduring relevance of historical encryption techniques.

https://en.wikipedia.org/wiki/Enigma_machine

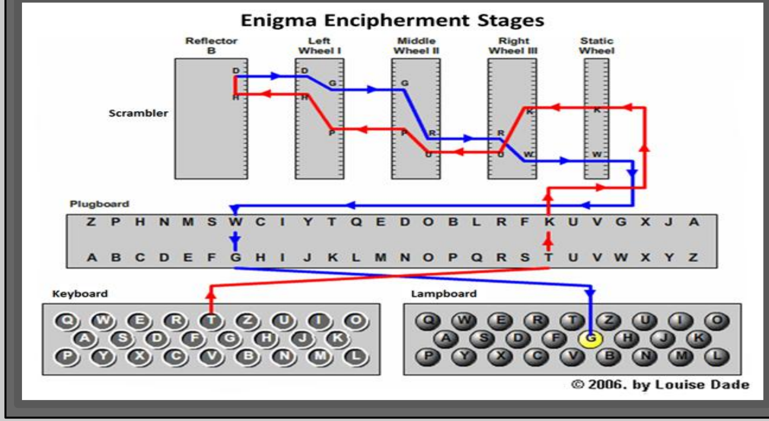
HISTORY

The Enigma Machine, invented by Arthur Scherbius in the early 20th century, became a pivotal tool in German military communication during World War II. Employing electromechanical components and a complex encryption mechanism, the Enigma Machine offered a seemingly impenetrable cipher. However, the codebreaking efforts at Bletchley Park, led by Alan Turing and a team of skilled cryptanalysts, successfully deciphered its encrypted messages. The intelligence gained from breaking Enigma codes, codenamed "Ultra," provided the Allies with a significant advantage and played a crucial role in shaping the outcome of the war. The Enigma Machine's impact on cryptography and its subsequent decryption showcased the power of innovation and teamwork, paving the way for advancements in modern encryption methods.

https://en.wikipedia.org/wiki/Enigma_machine

WORKING OF ENIGMA MACHINE

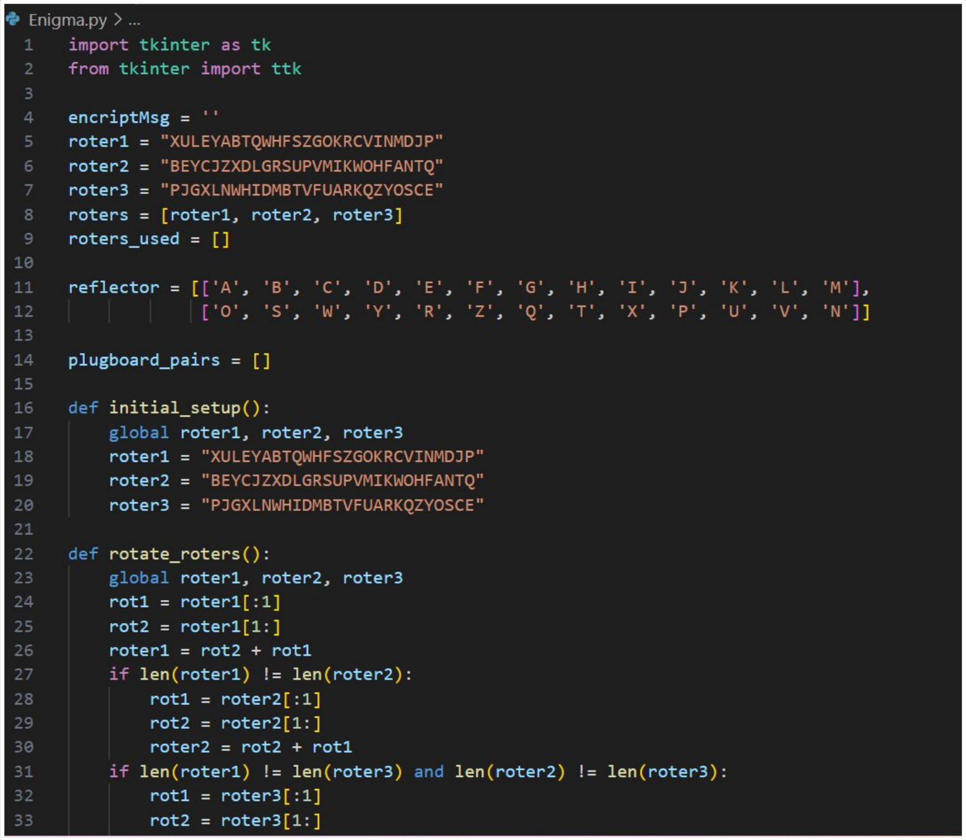
The Enigma Machine works by passing an electrical signal through a series of rotors, a reflector, and a plugboard. The rotors substituted letters as the signal passed through them, with the rightmost rotor rotating with each key press. The reflector reflected the signal back through the rotors, and the plugboard further scrambled the connections. The encryption process was reversible, allowing the same machine to be used for decryption. The complexity of the Enigma's design and rotor movement made it challenging to crack, but codebreakers like Alan Turing at Bletchley Park successfully deciphered its codes, significantly aiding the Allied war effort.



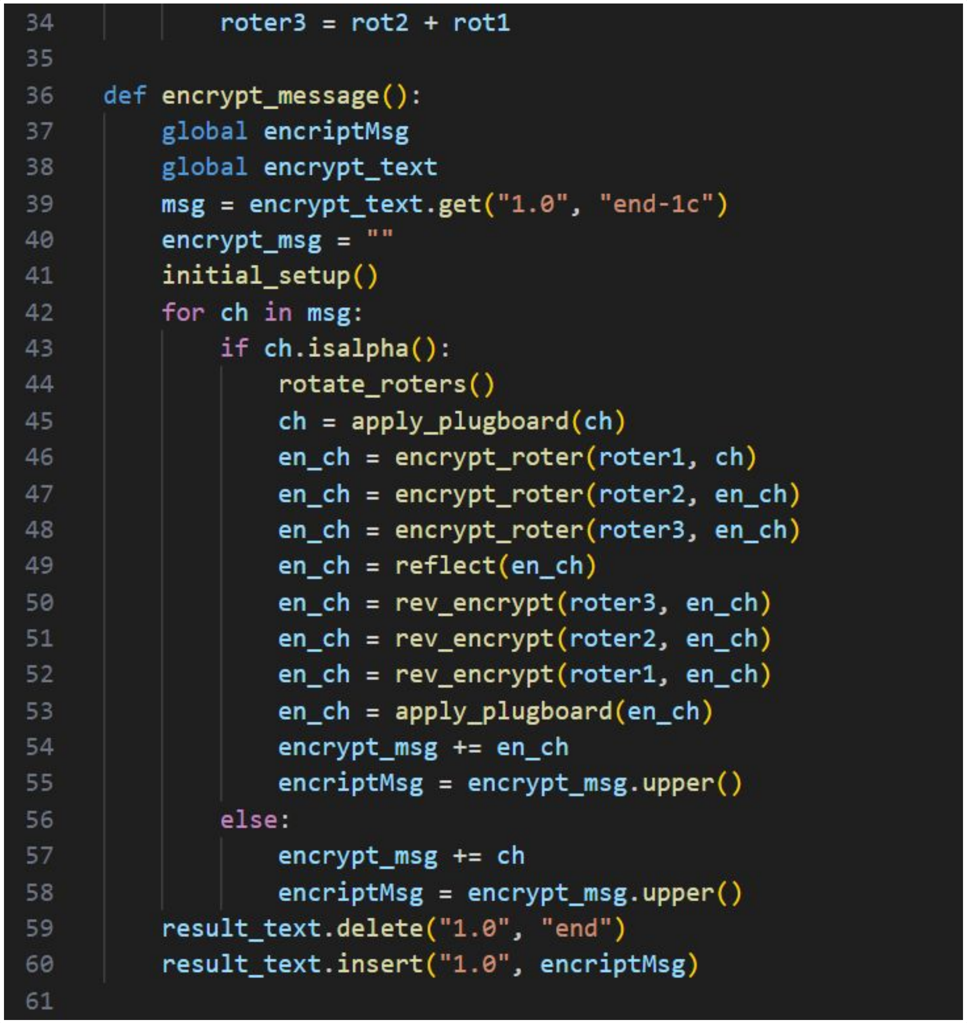
© 2006. by Louise Dade

https://en.wikipedia.org/wiki/Enigma_machine

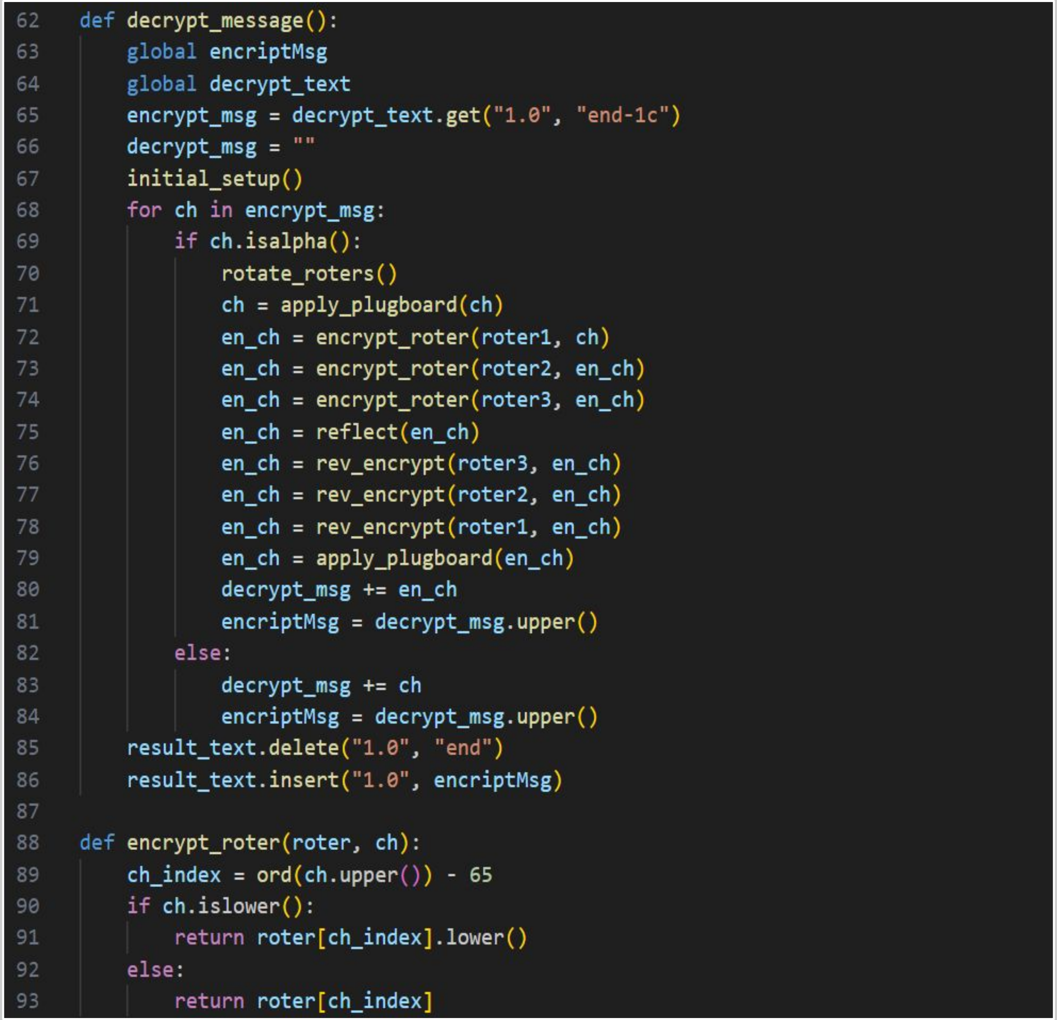
OUR CODE



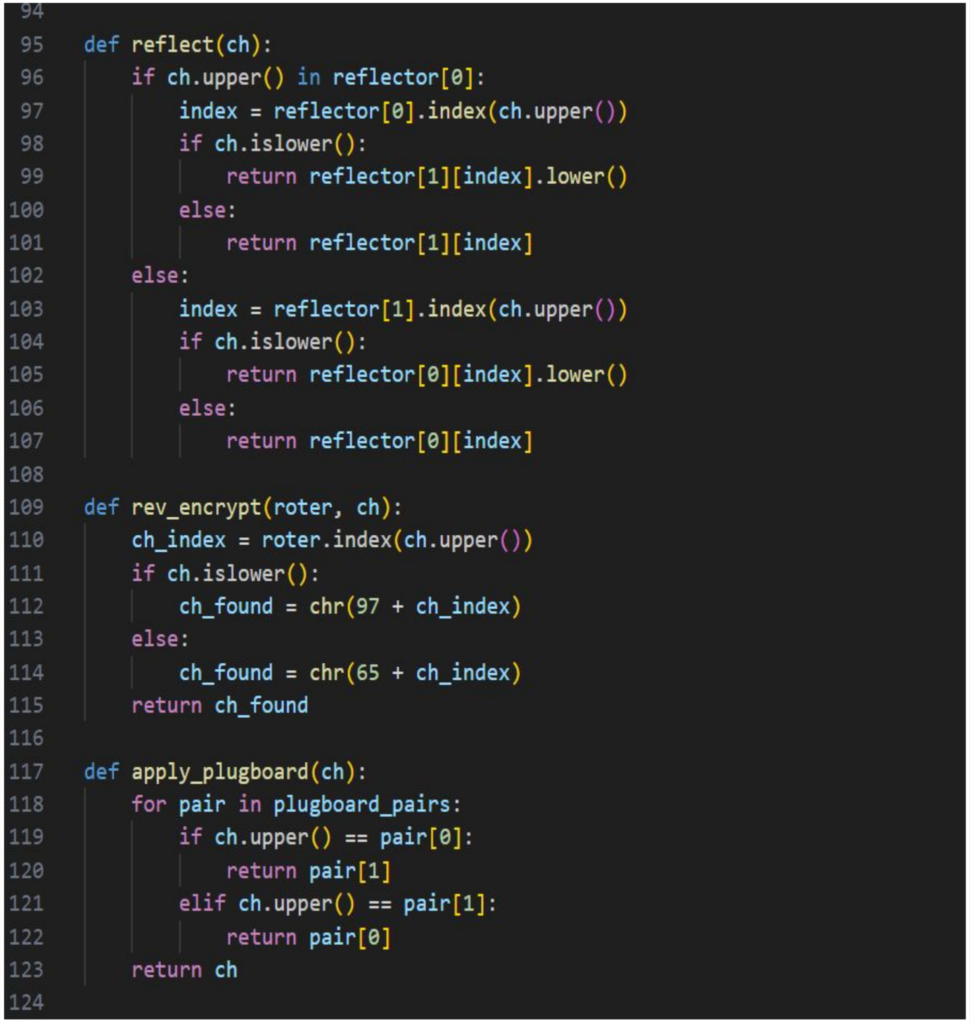
```python
import tkinter as tk
from tkinter import ttk

encriptMsg = ''
roter1 = "XULEYABTQWHFSZGOKRCVINMDJP"
roter2 = "BEYCJZXDLGRSUPVMIKWOHFANTQ"
roter3 = "PJGXLNWHIDMBTVFUARKQZYOSCE"
roters = [roter1, roter2, roter3]
roters_used = []

reflector = [['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M'],
             ['O', 'S', 'W', 'Y', 'R', 'Z', 'Q', 'T', 'X', 'P', 'U', 'V', 'N']]

plugboard_pairs = []

def initial_setup():
    global roter1, roter2, roter3
    roter1 = "XULEYABTQWHFSZGOKRCVINMDJP"
    roter2 = "BEYCJZXDLGRSUPVMIKWOHFANTQ"
    roter3 = "PJGXLNWHIDMBTVFUARKQZYOSCE"

def rotate_roters():
    global roter1, roter2, roter3
    rot1 = roter1[:1]
    rot2 = roter1[1:]
    roter1 = rot2 + rot1
    if len(roter1) != len(roter2):
        rot1 = roter2[:1]
        rot2 = roter2[1:]
        roter2 = rot2 + rot1
    if len(roter1) != len(roter3) and len(roter2) != len(roter3):
        rot1 = roter3[:1]
        rot2 = roter3[1:]
```



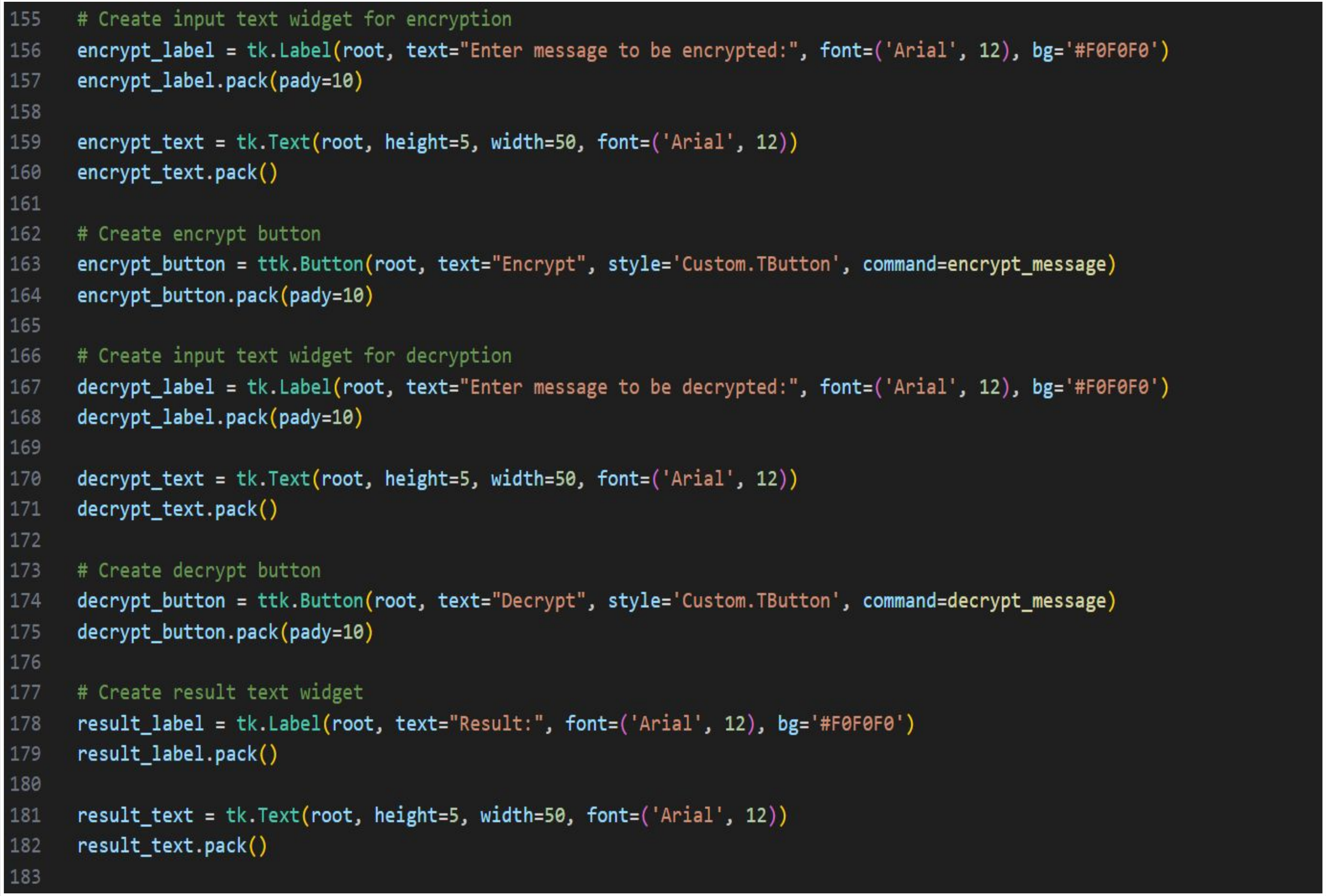
```python
        roter3 = rot2 + rot1

def encrypt_message():
    global encriptMsg
    global encrypt_text
    msg = encrypt_text.get("1.0", "end-1c")
    encrypt_msg = ""
    initial_setup()
    for ch in msg:
        if ch.isalpha():
            rotate_roters()
            ch = apply_plugboard(ch)
            en_ch = encrypt_roter(roter1, ch)
            en_ch = encrypt_roter(roter2, en_ch)
            en_ch = encrypt_roter(roter3, en_ch)
            en_ch = reflect(en_ch)
            en_ch = rev_encrypt(roter3, en_ch)
            en_ch = rev_encrypt(roter2, en_ch)
            en_ch = rev_encrypt(roter1, en_ch)
            en_ch = apply_plugboard(en_ch)
            encrypt_msg += en_ch
            encriptMsg = encrypt_msg.upper()
        else:
            encrypt_msg += ch
            encriptMsg = encrypt_msg.upper()
    result_text.delete("1.0", "end")
    result_text.insert("1.0", encriptMsg)

```



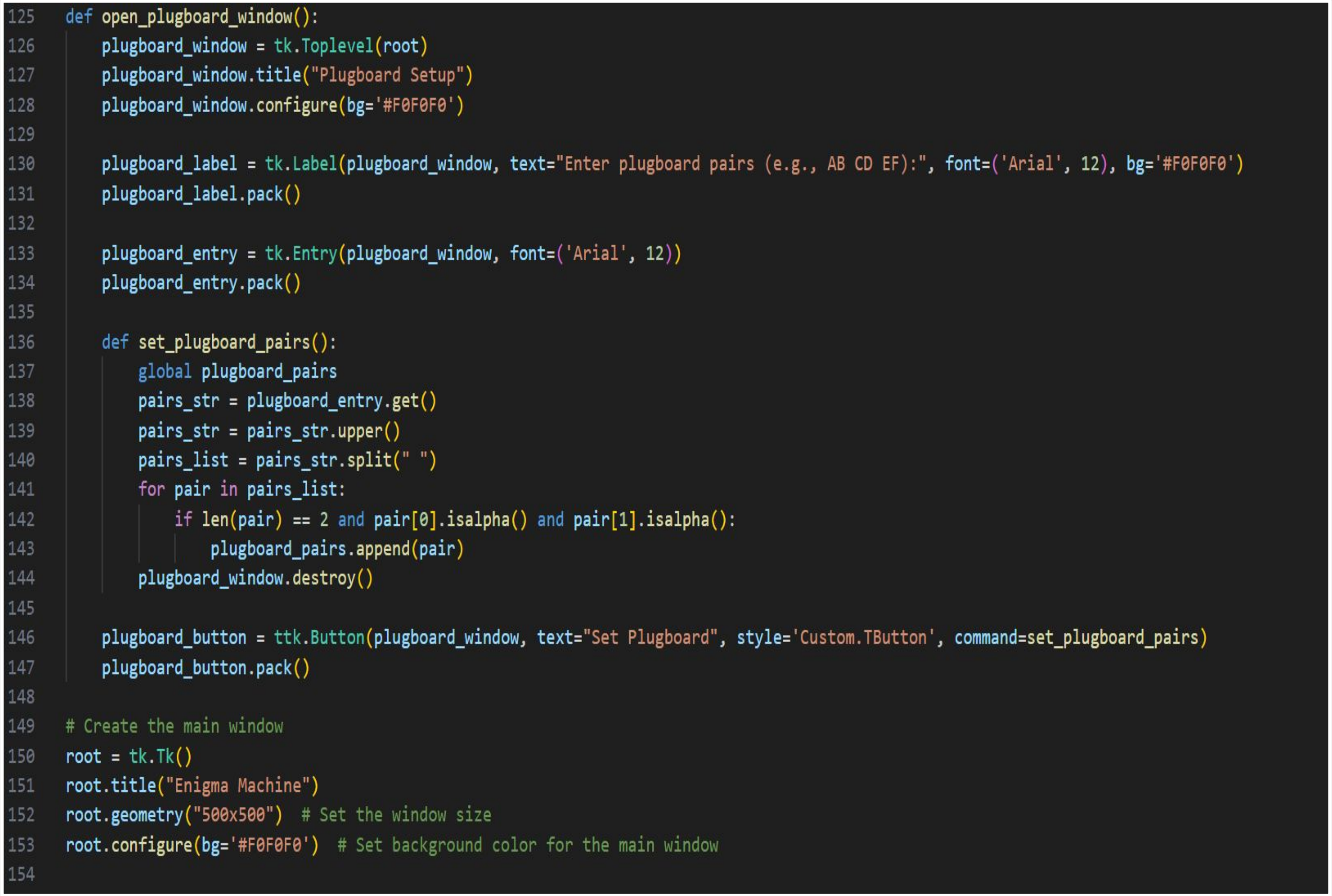
```python
def decrypt_message():
    global encriptMsg
    global decrypt_text
    encrypt_msg = decrypt_text.get("1.0", "end-1c")
    decrypt_msg = ""
    initial_setup()
    for ch in encrypt_msg:
        if ch.isalpha():
            rotate_roters()
            ch = apply_plugboard(ch)
            en_ch = encrypt_roter(roter1, ch)
            en_ch = encrypt_roter(roter2, en_ch)
            en_ch = encrypt_roter(roter3, en_ch)
            en_ch = reflect(en_ch)
            en_ch = rev_encrypt(roter3, en_ch)
            en_ch = rev_encrypt(roter2, en_ch)
            en_ch = rev_encrypt(roter1, en_ch)
            en_ch = apply_plugboard(en_ch)
            decrypt_msg += en_ch
            encriptMsg = decrypt_msg.upper()
        else:
            decrypt_msg += ch
            encriptMsg = decrypt_msg.upper()
    result_text.delete("1.0", "end")
    result_text.insert("1.0", encriptMsg)

def encrypt_roter(roter, ch):
    ch_index = ord(ch.upper()) - 65
    if ch.islower():
        return roter[ch_index].lower()
    else:
        return roter[ch_index]
```



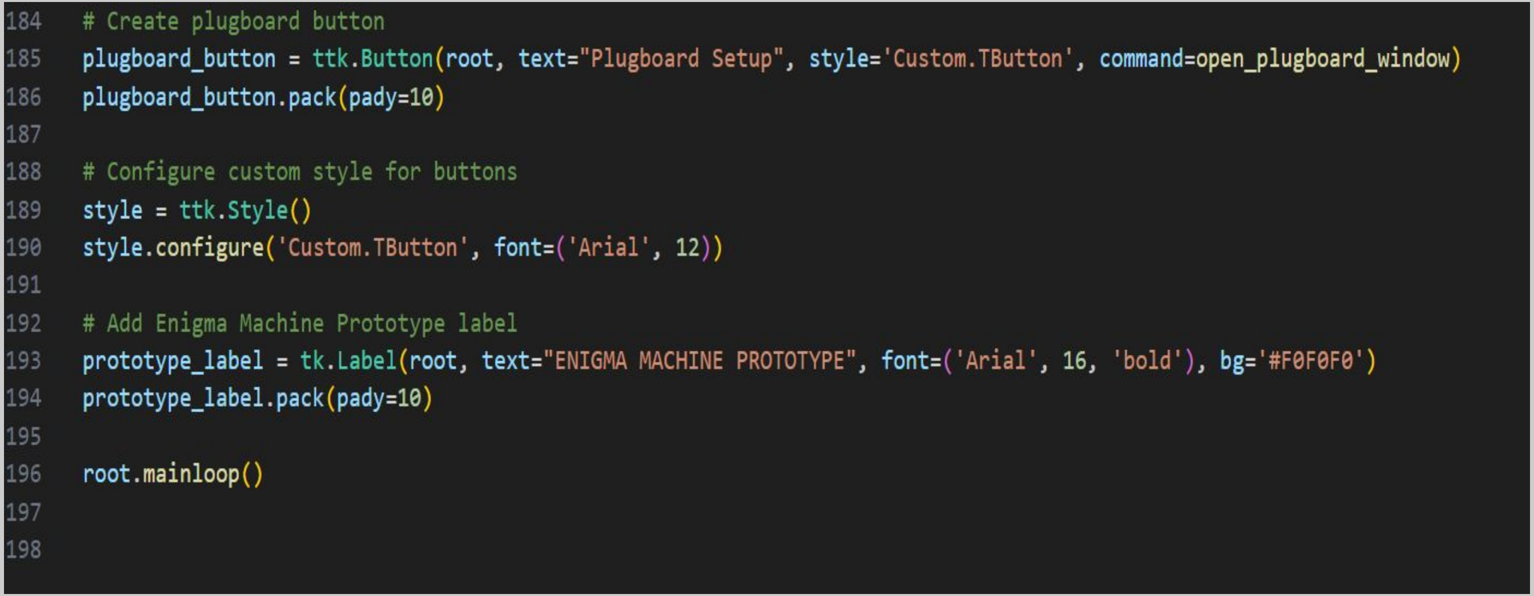
```python
def reflect(ch):
    if ch.upper() in reflector[0]:
        index = reflector[0].index(ch.upper())
        if ch.islower():
            return reflector[1][index].lower()
        else:
            return reflector[1][index]
    else:
        index = reflector[1].index(ch.upper())
        if ch.islower():
            return reflector[0][index].lower()
        else:
            return reflector[0][index]

def rev_encrypt(roter, ch):
    ch_index = roter.index(ch.upper())
    if ch.islower():
        ch_found = chr(97 + ch_index)
    else:
        ch_found = chr(65 + ch_index)
    return ch_found

def apply_plugboard(ch):
    for pair in plugboard_pairs:
        if ch.upper() == pair[0]:
            return pair[1]
        elif ch.upper() == pair[1]:
            return pair[0]
    return ch

```



```python
# Create input text widget for encryption
encrypt_label = tk.Label(root, text="Enter message to be encrypted:", font=('Arial', 12), bg='#F0F0F0')
encrypt_label.pack(pady=10)

encrypt_text = tk.Text(root, height=5, width=50, font=('Arial', 12))
encrypt_text.pack()

# Create encrypt button
encrypt_button = ttk.Button(root, text="Encrypt", style='Custom.TButton', command=encrypt_message)
encrypt_button.pack(pady=10)

# Create input text widget for decryption
decrypt_label = tk.Label(root, text="Enter message to be decrypted:", font=('Arial', 12), bg='#F0F0F0')
decrypt_label.pack(pady=10)

decrypt_text = tk.Text(root, height=5, width=50, font=('Arial', 12))
decrypt_text.pack()

# Create decrypt button
decrypt_button = ttk.Button(root, text="Decrypt", style='Custom.TButton', command=decrypt_message)
decrypt_button.pack(pady=10)

# Create result text widget
result_label = tk.Label(root, text="Result:", font=('Arial', 12), bg='#F0F0F0')
result_label.pack()

result_text = tk.Text(root, height=5, width=50, font=('Arial', 12))
result_text.pack()

```



```python
def open_plugboard_window():
    plugboard_window = tk.Toplevel(root)
    plugboard_window.title("Plugboard Setup")
    plugboard_window.configure(bg='#F0F0F0')

    plugboard_label = tk.Label(plugboard_window, text="Enter plugboard pairs (e.g., AB CD EF):", font=('Arial', 12), bg='#F0F0F0')
    plugboard_label.pack()

    plugboard_entry = tk.Entry(plugboard_window, font=('Arial', 12))
    plugboard_entry.pack()

    def set_plugboard_pairs():
        global plugboard_pairs
        pairs_str = plugboard_entry.get()
        pairs_str = pairs_str.upper()
        pairs_list = pairs_str.split(" ")
        for pair in pairs_list:
            if len(pair) == 2 and pair[0].isalpha() and pair[1].isalpha():
                plugboard_pairs.append(pair)
        plugboard_window.destroy()

    plugboard_button = ttk.Button(plugboard_window, text="Set Plugboard", style='Custom.TButton', command=set_plugboard_pairs)
    plugboard_button.pack()

# Create the main window
root = tk.Tk()
root.title("Enigma Machine")
root.geometry("500x500")  # Set the window size
root.configure(bg='#F0F0F0')  # Set background color for the main window

```



```python
# Create plugboard button
plugboard_button = ttk.Button(root, text="Plugboard Setup", style='Custom.TButton', command=open_plugboard_window)
plugboard_button.pack(pady=10)

# Configure custom style for buttons
style = ttk.Style()
style.configure('Custom.TButton', font=('Arial', 12))

# Add Enigma Machine Prototype label
prototype_label = tk.Label(root, text="ENIGMA MACHINE PROTOTYPE", font=('Arial', 16, 'bold'), bg='#F0F0F0')
prototype_label.pack(pady=10)

root.mainloop()


```

BIBLIOGRAPHY

1. https://en.wikipedia.org/wiki/Enigma_machine

2.https://chat.openai.com

3.www.google.com

4.https://www.cryptomuseum.com/crypto/enigma/working.htm

Made by: Nawang Dorjay
Rigzin Namgail
Jigmet Gyatso.