**STEGANOGRAPHY**

****

**LE livre ultime sur notre projet de stéganographie**

# 1.Introduction à la Stéganographie

**Qu'est-ce que la stéganographie?**

La stéganographie est l’art de cacher un message ou une information à l’intérieur d’un support (comme une image, un audio ou une vidéo) sans que personne ne puisse le détecter. Contrairement au cryptage, qui protège les données en les transformant, la stéganographie masque les données pour qu’elles passent inaperçues.

## Pourquoi est-ce utile?

Confidentialité – Protéger des informations sensibles sans éveiller de soupçons.

Cybersécurité – Transmettre des messages cachés entre agents ou entreprises.

Art et Technologie – Certains artistes utilisent la stéganographie dans leurs œuvres numériques!

**Exemples modernes de stéganographie numérique**

Dans le monde numérique, on peut cacher des messages dans:

Images (PNG, JPEG) – Modification des pixels pour insérer des données.

Vidéos – Masquage des données dans des frames invisibles.

Documents – Ajout de texte invisible dans un fichier PDF ou Word.

**backend**

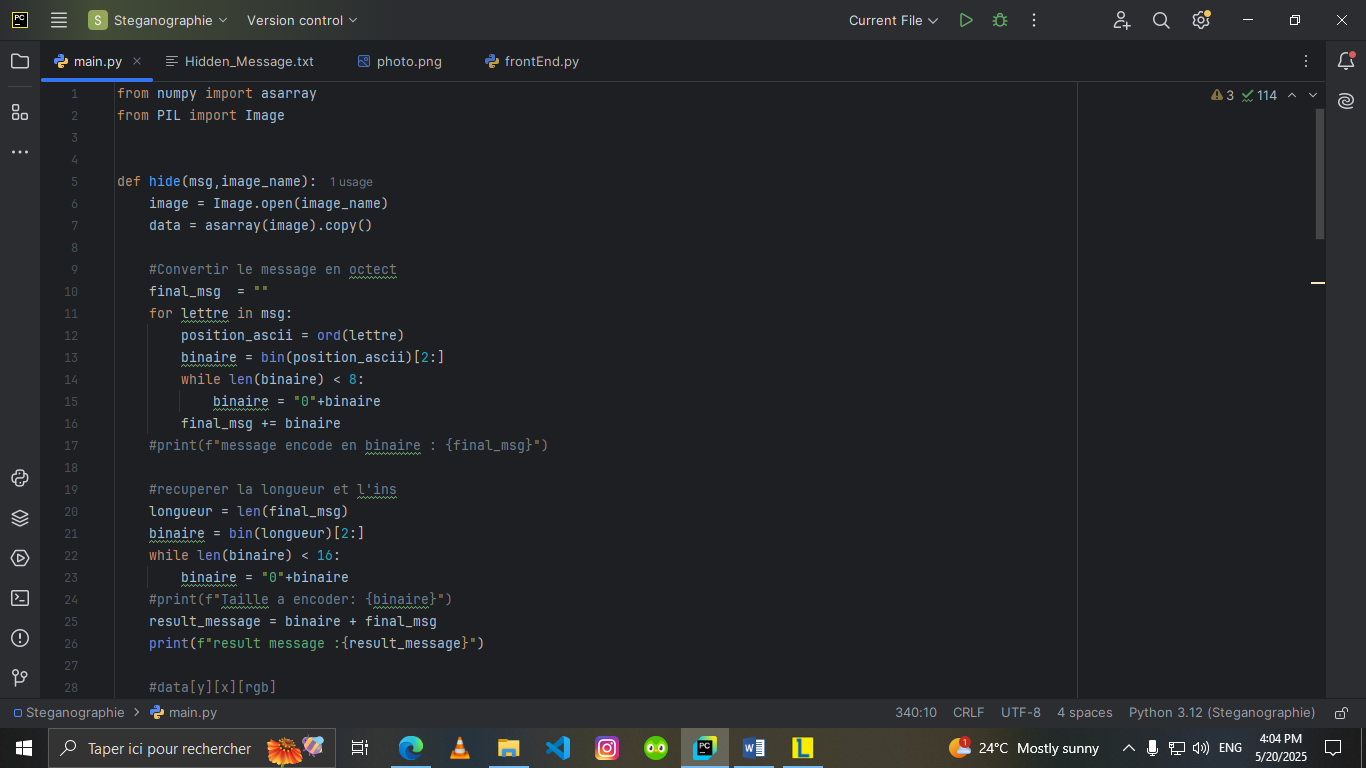
# 2.Explication complète du code

1. Fonction hide() - Cacher un message dans une image

Objectif: Transformer un message en binaire et l’insérer dans les pixels de l’image.

Méthode utilisée: Modification du bit de poids faible (LSB) de chaque pixel.

backend



Ce que fait ce code:

Charge l’image et la transforme en tableau numpy

Convertit le message en binaire

Ajoute la taille du message pour faciliter l’extraction

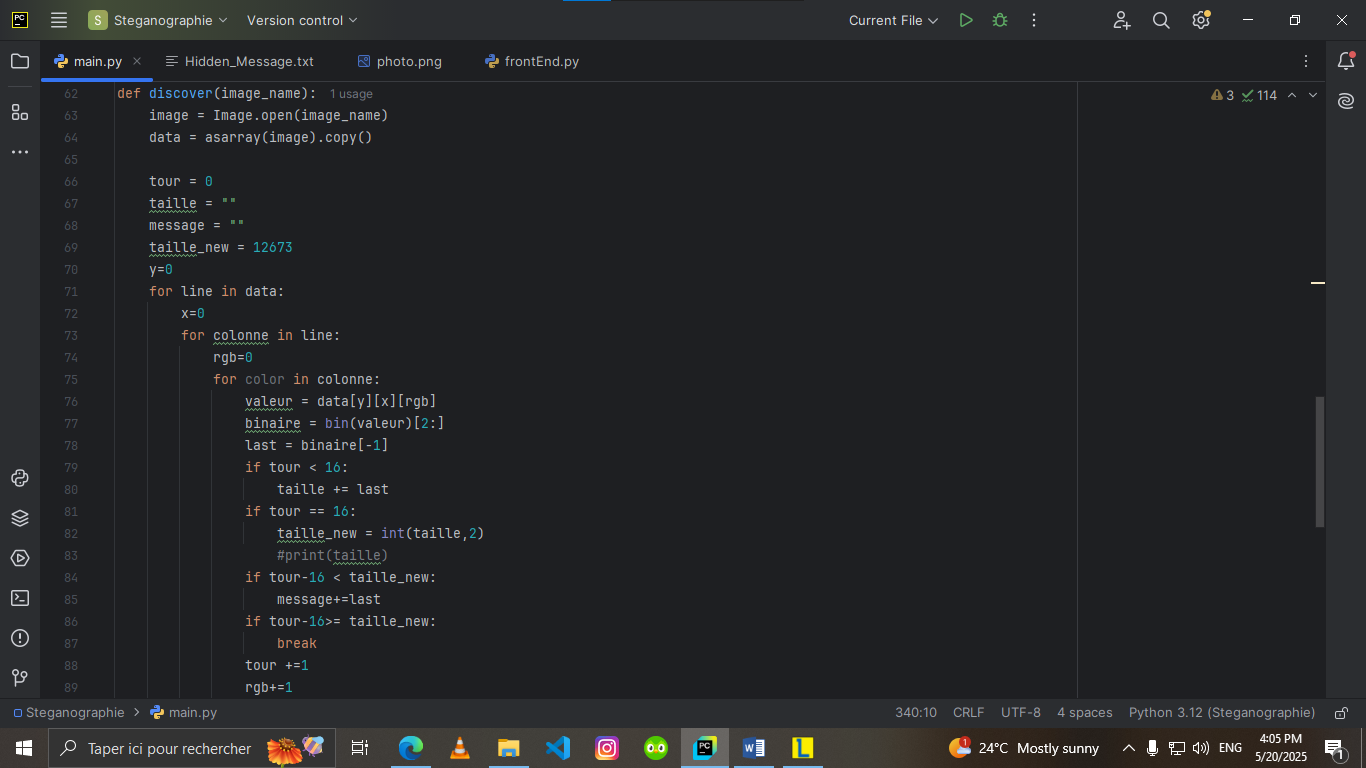
Modifie le dernier bit des pixels pour cacher les données

Sauvegarde l’image avec le message caché

**2. Fonction discover() - Extraire le message caché**

Objectif: Lire l’image modifiée et récupérer le message caché bit par bit.

Méthode utilisée: Lecture des bits modifiés et reconstruction du texte.



Ce que fait ce code:

Charge l’image et la transforme en tableau numpy

Lit les bits modifiés pour extraire la taille du message

Reconstruit le message caché en convertissant les bits en caractères

Sauvegarde le message dans Hidden\_Message.txt

**3. Intégration avec l’interface Tkinter**

Dans notre GUI améliorée, on ajoute des boutons et des champs de texte pour encoder/décoder les messages de manière visuelle!

Sélection automatique d’image

Champ de texte pour entrer le message

Affichage des résultats directement dans la fenêtre

# Création de l’interface Tkinter

🔹 Objectif de l’interface

✅ Interaction facile – Plus besoin de lancer le script manuellement.

✅ Champ d’entrée pour écrire le message – Simple et intuitif!

✅ Boutons pour cacher et extraire – Juste un clic pour exécuter les fonctions!

✅ Affichage du message extrait directement sur l’écran.

**UI**

**Installation de Tkinter**

Tkinter est déjà intégré à Python, donc pas besoin d’installation!

Si tu veux installer Pillow (pour gérer les images), utilise:

bash

pip install pillow

Construction de l’interface graphique

On commence avec un design simple avant d'ajouter effets et styles!

1️⃣ Initialisation de la fenêtre principale

import tkinter as tk

from tkinter import filedialog, messagebox

from PIL import Image

import numpy as np

# Création de la fenêtre principale

root = tk.Tk()

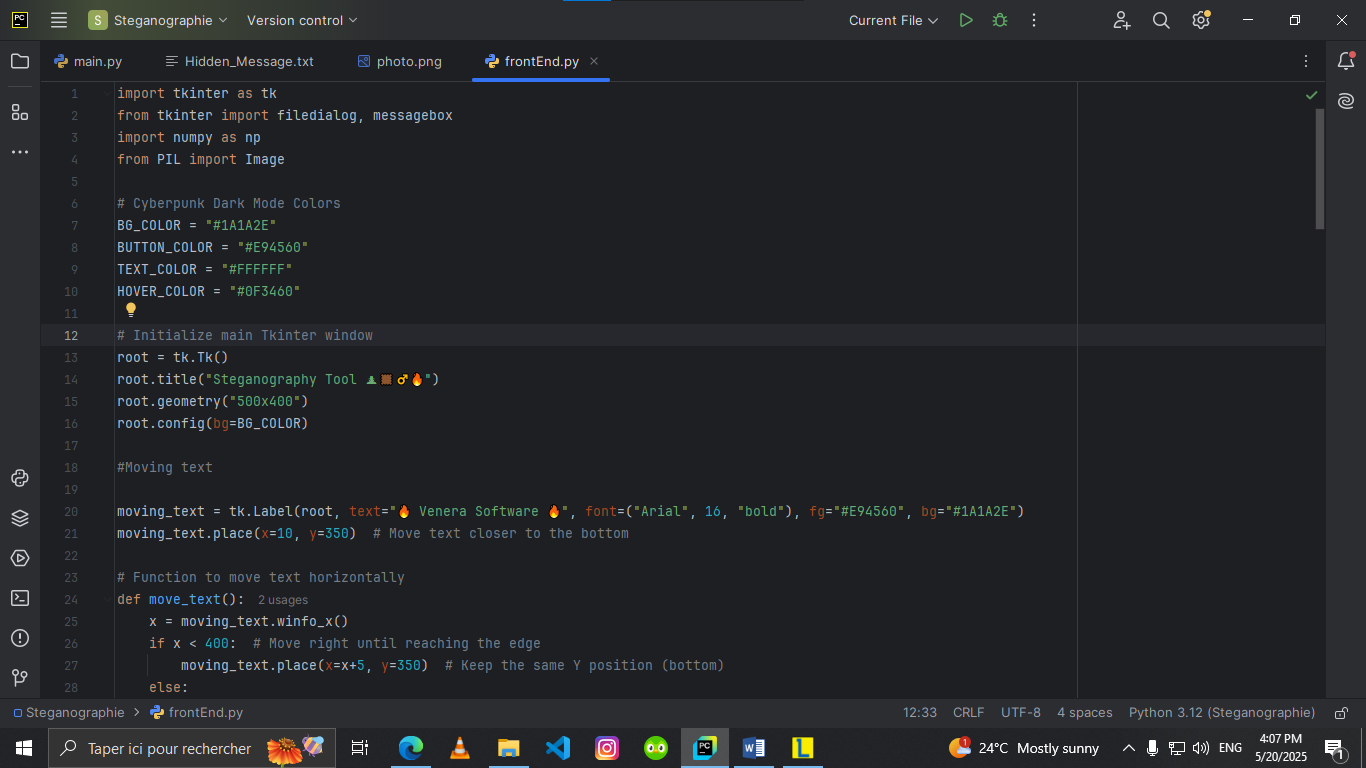
root.title("Steganographie Interface ")

root.geometry("500x300")

root.config(bg="#1A1A2E") # Couleur de fond sombre

✅ On crée une fenêtre Tkinter avec un titre et une taille adaptée

✅ On applique un fond sombre pour le style cyber



2️⃣ Ajout des champs et boutons

python

tk.Label(root, text="Entrez votre message:", fg="white", bg="#1A1A2E", font=("Arial", 12)).pack(pady=5)

entry\_message = tk.Entry(root, width=40, font=("Arial", 12))

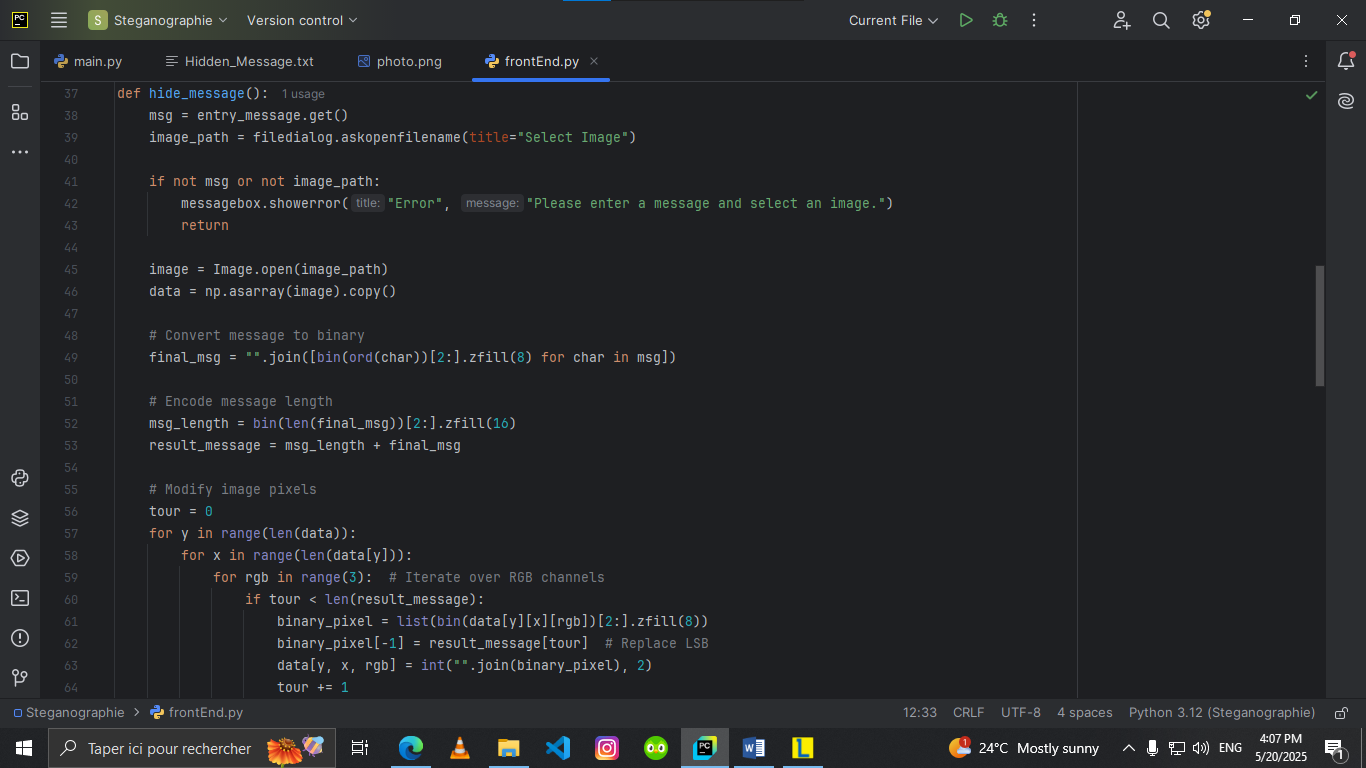
entry\_message.pack(pady=5)

btn\_hide = tk.Button(root, text="Cacher le Message ", font=("Arial", 12, "bold"), bg="red", fg="white")

btn\_hide.pack(pady=10)

btn\_discover = tk.Button(root, text="Découvrir le Message ", font=("Arial", 12, "bold"), bg="blue", fg="white")

btn\_discover.pack(pady=10)

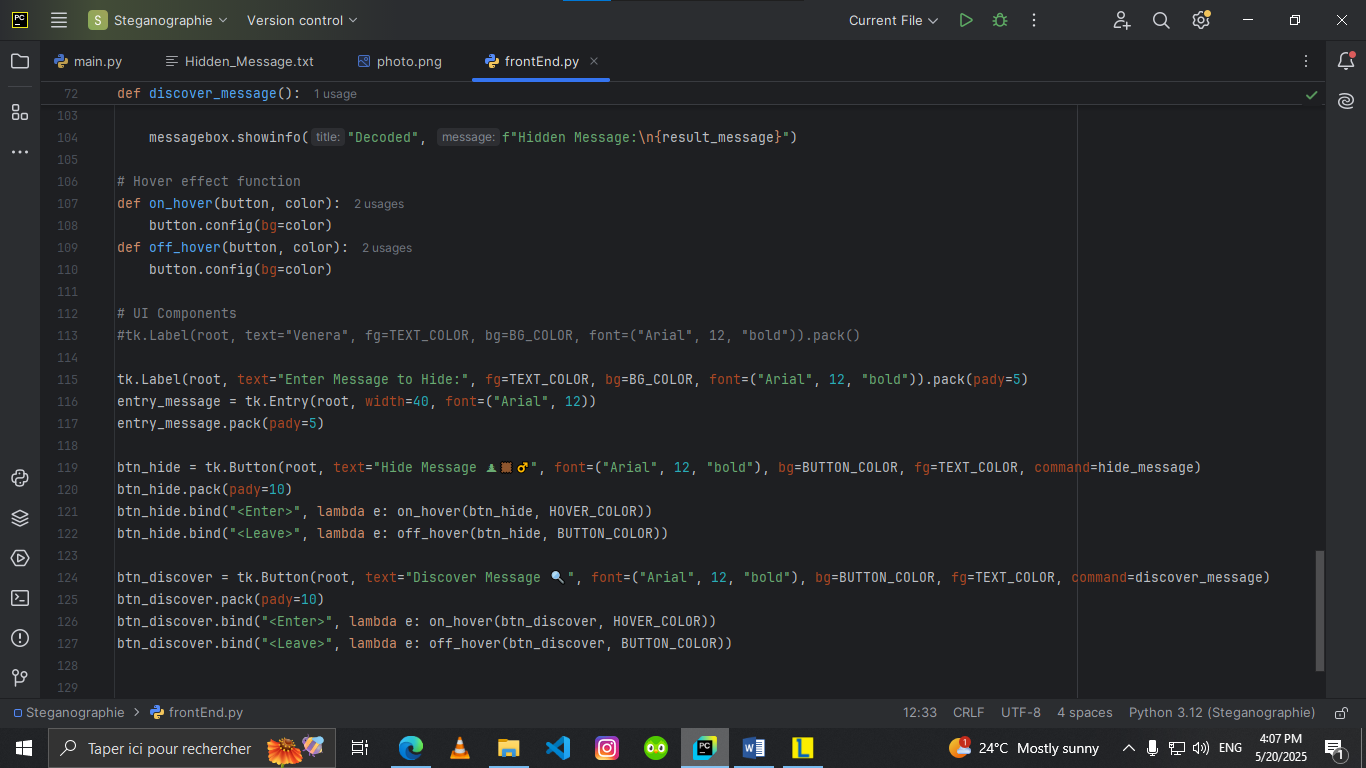


**Champ de texte pour écrire le message**

Boutons interactifs pour encoder/décoder

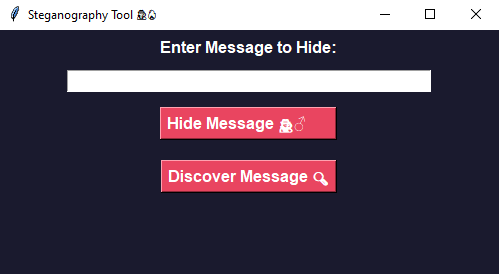
cette étape, notre interface fonctionne mais les boutons ne sont pas encore connectés aux fonctions!

La prochaine étape consiste à lier les boutons aux fonctions hide() et discover()!



# Les améliorations avancée

1. Ajout d’un système de sélection d’image



Actuellement, l’utilisateur doit entrer manuellement le nom du fichier, ce qui n’est pas pratique!

Solution: Ajouter un sélecteur de fichier avec filedialog.askopenfilename()

Mise à jour du code pour la sélection d’image

python

from tkinter import filedialog

def select\_image():

filepath = filedialog.askopenfilename(title="Sélectionner une image", filetypes=[("PNG Files", "\*.png"), ("JPEG Files", "\*.jpg")])

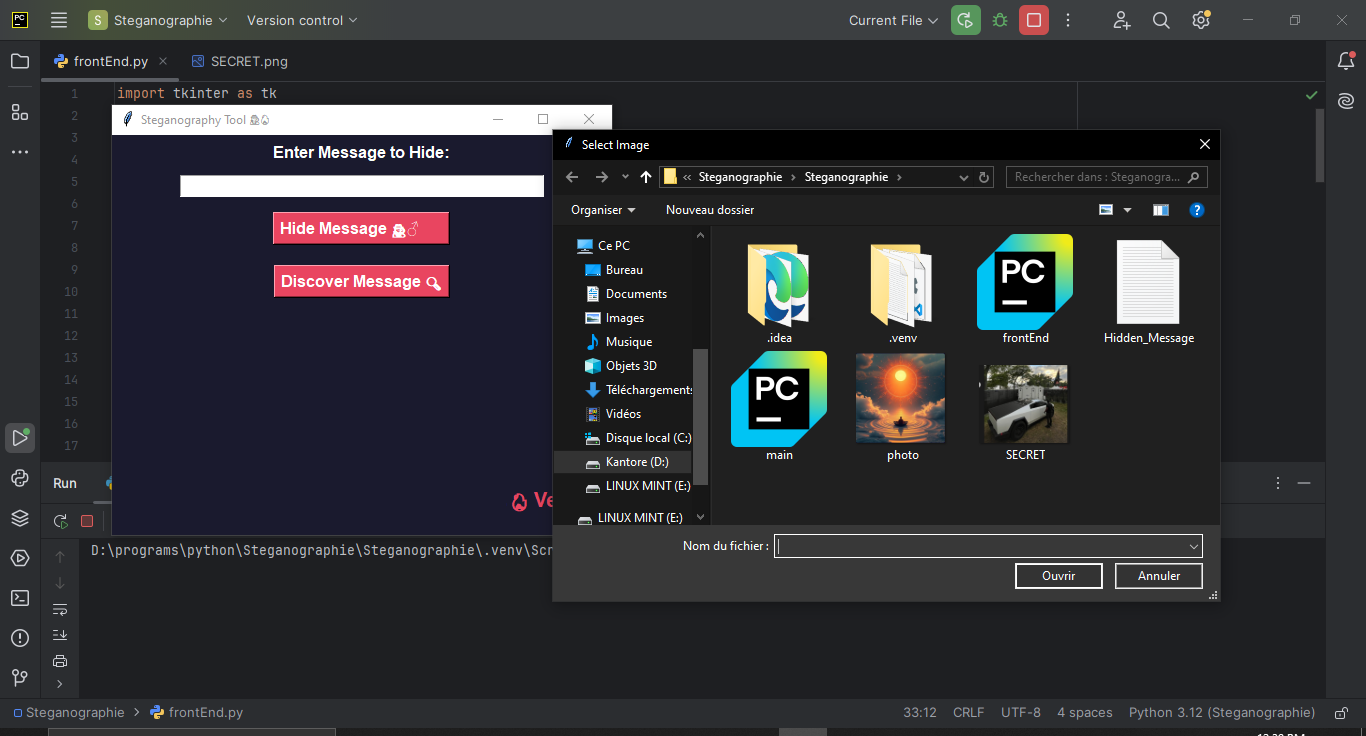
if filepath:

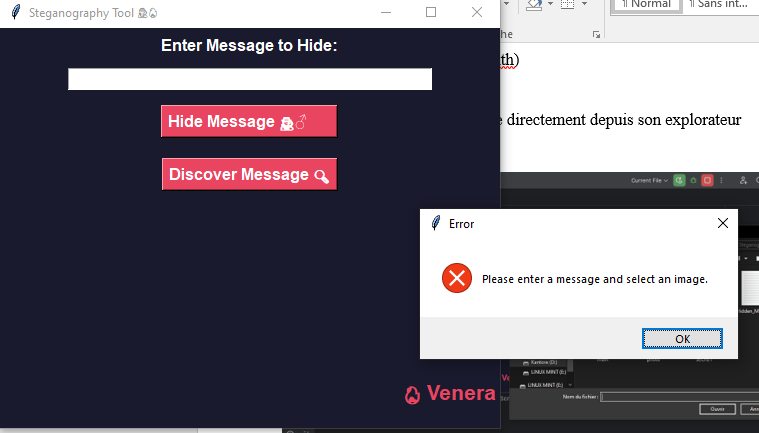
entry\_image.delete(0, tk.END)

entry\_image.insert(0, filepath)

L’utilisateur peut choisir l’image directement depuis son explorateur

Aucune erreur de saisie possible



****

**2. Ajout d’un champ d’affichage du message caché**

Actuellement, le message extrait est uniquement affiché dans une boîte de dialogue.

Solution: Afficher le message directement dans l’interface après extraction!

**Mise à jour du code pour afficher le message extrait**

def discover\_message():

image\_path = filedialog.askopenfilename(title="Sélectionner l'image à décoder")

if not image\_path:

messagebox.showerror("Erreur", "Veuillez sélectionner une image.")

return

result\_message = extraire\_message(image\_path) # Fonction qui récupère le message

text\_display.config(state="normal") # Activer l’édition du champ

text\_display.delete(1.0, tk.END) # Effacer le texte précédent

text\_display.insert(tk.END, result\_message) # Ajouter le message extrait

text\_display.config(state="disabled") # Désactiver l’édition

Le message caché s’affiche directement dans un champ Tkinter!

**3. Amélioration du design (animations & couleurs)**

Ajout d’un effet de survol sur les boutons

Solution: Modifier la couleur lorsqu’on passe la souris dessus!

def on\_hover(button, color):

button.config(bg=color)

def off\_hover(button, color):

button.config(bg=color)

btn\_hide.bind("<Enter>", lambda e: on\_hover(btn\_hide, "darkred"))

btn\_hide.bind("<Leave>", lambda e: off\_hover(btn\_hide, "red"))

Les boutons changent de couleur au survol pour un effet interactif

**4. Ajout d’une barre de progression**

Affichage d’une barre de progression lors du traitement

Solution: Utiliser ttk.Progressbar pour montrer l’état d’avancement.

python

from tkinter.ttk import Progressbar

progress = Progressbar(root, orient="horizontal", length=300, mode="determinate")

progress.pack(pady=5)

def update\_progress(value):

progress["value"] = value

root.update\_idletasks()

L’utilisateur voit l’avancement du processus en temps réel

# Utilisation réelle

**1. Sécurité des communications**

Problème: Les gouvernements et entreprises utilisent des canaux sécurisés, mais ceux-ci peuvent être surveillés.

Solution: La stéganographie permet de masquer des messages à l’intérieur d’images ou de fichiers audio, évitant ainsi d’éveiller les soupçons!

✅ Cas réel: Des journalistes et activistes utilisent la stéganographie pour échanger des informations sans attirer l’attention dans des pays où la censure est forte.

**2. Protection des données personnelles**

Problème: Les cyberattaques ciblent souvent les informations sensibles stockées en ligne.

Solution: Les entreprises peuvent cacher les données sensibles dans des fichiers multimédias pour qu’elles passent inaperçues.

Cas réel: Certains logiciels de cybersécurité utilisent la stéganographie pour protéger les mots de passe et clés d’accès, empêchant leur détection par les hackers.

**3. Authentification et lutte contre la fraude**

Problème: Les faux documents et images trafiquées sont monnaie courante sur Internet.

Solution: Des systèmes de protection intégrant des signatures cachées permettent de certifier les documents et images authentiques.

Cas réel: Les banques et institutions financières utilisent des techniques stéganographiques pour cacher des signatures invisibles sur les documents numériques, rendant leur falsification plus difficile.

**4. Stockage clandestin d’informations**

Problème: Certains gouvernements interdisent l’accès à des documents critiques ou des contenus protégés.

Solution: La stéganographie permet de stocker des fichiers entiers à l’intérieur d’autres fichiers (exemple: une vidéo qui contient un document secret).

Cas réel: Des chercheurs en cybersécurité ont démontré qu’il est possible de cacher un programme entier dans une image, le rendant indétectable aux antivirus!