Présentation Projet : Analyseur d’Image en Python avec Interface Graphique

1. Introduction

Ce projet propose une application simple en Python permettant de :

- Charger une image depuis ton disque dur
- Afficher l’image dans une fenêtre graphique
- Extraire et afficher plusieurs paramètres utiles concernant l’image, notamment :
- Sa taille en mémoire
- Sa résolution (largeur et hauteur en pixels)
- Son format (JPEG, PNG, etc.)
- Le taux de compression approximatif
- Sa plage dynamique de luminosité (différence entre intensité minimale et maximale)

L’interface est réalisée avec Tkinter, la bibliothèque graphique standard de Python.
Le traitement d’image est réalisé avec la bibliothèque Pillow (PIL fork).

2. Structure du projet et installation

Fichiers utilisés

- **`image_analyzer_python_gui.py`** : script Python principal

Dépendances

- Python 3.x installé sur la machine
- Pillow (PIL fork) : installation via **`pip install pillow`**

Lancement

- Depuis un terminal, place-toi dans le dossier contenant le script
- Lance la commande :

```
python image_analyzer_python_gui.py
```

- L’interface graphique s’ouvre automatiquement

3. Explications détaillées du code

```
Python :


import os


from tkinter import Tk, Button, Label, filedialog, Canvas, Scrollbar, Text,
BOTH, RIGHT, Y, LEFT, Frame, TOP
```

```
from PIL import Image, ImageTk, ImageStat...
```

- Import des modules nécessaires :
- `os` pour accéder à la taille du fichier sur disque
- `tkinter` et ses composants pour créer l'interface
- `Pillow` pour ouvrir, manipuler et analyser l'image

```python
def human_readable_size(size_bytes):
    if size_bytes == 0:
        return "0 B"
    size_name = ("B", "Ko", "Mo", "Go", "To")
    i = 0
    p = 1024
    while size_bytes >= p and i < len(size_name)-1:
        size_bytes /= p
        i +=1
    return f"{size_bytes:.2f} {size_name[i]}". . .
```

- Fonction utilitaire pour convertir et afficher des tailles en lecture humaine (octets, kilo-octets, méga-octets…)

```python
class ImageAnalyzerApp:
    def __init__(self, master):
        self.master = master
        master.title("Analyseur d'image Python")
        master.geometry("800x600")
        master.resizable(False, False). . .
```

- Création de la fenêtre principale avec Tkinter, taille fixe 800x600px

```python
```

```
        # Bouton pour charger une image

        self.load_btn = Button(master, text="Charger une image depuis le disque",
command=self.load_image)

        self.load_btn.pack(side=TOP, pady=10). . .
```

- Création d’un bouton placé en haut, qui déclenche la sélection et chargement d’image

```
python7 lines

        # Frame pour contenir image + scrollbar si besoin

        self.image_frame = Frame(master, width=400, height=400)

        self.image_frame.pack(side=LEFT, padx=10, pady=10)



        # Canvas pour afficher l'image

        self.canvas = Canvas(self.image_frame, width=400, height=400, bg="#eee")

        self.canvas.pack(). . .
```

- Zone d’affichage de l’image, sous forme de canvas de 400x400 px à gauche

```
python11 lines

        # Frame pour afficher les paramètres

        self.info_frame = Frame(master, width=380, height=580)

        self.info_frame.pack(side=RIGHT, fill=BOTH, expand=True, padx=10, pady=10)



        self.info_text = Text(self.info_frame, wrap="word", state="disabled", width=45)

        self.info_text.pack(side=LEFT, fill=BOTH, expand=True)



        # Scrollbar

        self.scrollbar = Scrollbar(self.info_frame, command=self.info_text.yview)

        self.scrollbar.pack(side=RIGHT, fill=Y)
```

```
        self.info_text.config(yscrollcommand=self.scrollbar.set)...
```

- Zone texte avec ascenseur pour afficher les paramètres à droite de la fenêtre

```python
    def load_image(self):
        file_path = filedialog.askopenfilename(
            title="Choisissez une image",
            filetypes=[("Images", "*.jpg *.jpeg *.png *.bmp *.gif *.tiff *.webp"), ("Tous les fichiers", "*.*")]
        )
        if not file_path:
            return
        try:
            self.img = Image.open(file_path)
        except Exception as e:
            self.show_info(f"Erreur lors de l'ouverture de l'image : {e}")
            return
        # Afficher l'image redimensionnée dans le canvas
        self.display_image()
        # Analyser et afficher les paramètres
        self.analyze_image(file_path)...
```

- Ouvre la boîte de dialogue pour choisir une image
- Ouvre l’image avec Pillow
- Lance l’affichage et l’analyse

```python
    def display_image(self):
        max_size = (400, 400)
        img_copy = self.img.copy()
```

```
        img_copy.thumbnail(max_size, Image.ANTIALIAS)

        self.img_tk = ImageTk.PhotoImage(img_copy)

        self.canvas.delete("all")

        self.canvas.create_image(200, 200, image=self.img_tk). . .
```

- Affiche une copie redimensionnée de l’image dans le canvas, centrée

```
python23 lines

    def analyze_image(self, file_path):

        try:

            file_size = os.path.getsize(file_path)

            width, height = self.img.size

            format_img = self.img.format if self.img.format else "Inconnu"

            raw_size_bytes = width * height * 3

            compression_ratio = raw_size_bytes / file_size if file_size > 0 else 0

            gray_img = self.img.convert("L")

            stat = ImageStat.Stat(gray_img)

            min_lum, max_lum = stat.extrema[0]

            dynamic_range = max_lum - min_lum

            info = f"""

Nom du fichier : {os.path.basename(file_path)}

Taille fichier : {human_readable_size(file_size)}

Format : {format_img}

Dimensions : {width} x {height} pixels

Taille brute pixels : {human_readable_size(raw_size_bytes)}

Taux de compression approximatif : {compression_ratio:.2f} (taille brute / taille fichier)

Plage dynamique (luminosité) : {dynamic_range} (min {min_lum} - max {max_lum})
```

```
            """

            self.show_info(info.strip())

        except Exception as e:

            self.show_info(f"Erreur lors de l'analyse de l'image : {e}")...
```

- Analyse les paramètres via Pillow et os
- Calcule dynamique sur image convertie en niveaux de gris
- Formate l’affichage textuel

```
python5 lines

    def show_info(self, text):

        self.info_text.config(state="normal")

        self.info_text.delete(1.0, "end")

        self.info_text.insert("end", text)

        self.info_text.config(state="disabled")...
```

- Méthode pour afficher/mettre à jour le texte dans la zone d’infos

4. Instructions d’utilisation

- Installe Python 3 depuis https://python.org (si nécessaire)
- Installe Pillow avec :
  `pip install pillow`
- Place le script dans un dossier accessible
- Lance-le avec :
  `python image_analyzer_python_gui.py`
- Clique sur le bouton pour sélectionner et charger une image
- Le visuel s’affiche à gauche, les détails à droite

5. Conclusion

Cette application simple permet d’explorer visuellement et techniquement les images sur votre ordinateur sans installer de logiciel lourd.
Le code est facile à étendre et à personnaliser selon vos besoins (ajout de nouveaux paramètres, capture caméra, export des résultats…).