

MIRA Converter

ソースコード読解編

HEIC → JPG 一括変換アプリを「何が・なぜ・どの順番で動くのか」から読む

このPDFの目的

PythonやStreamlitをまだ完全に理解していなくても、MIRA Converterのソースを上から順に追い、各行の役割と処理の流れをつかむための入門書です。

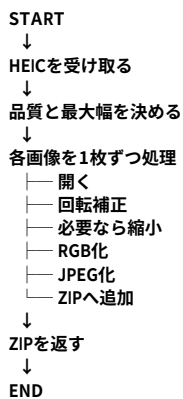
コードを暗記するのではなく、入力 → 判定 → 変換 → 圧縮 → 出力というアルゴリズムの骨格を見つけることを目標にします。

初版 / MIRA Converter

1. まず全体像

このアプリは、ブラウザ画面でHEIC/HEIF写真を受け取り、必要なら縮小し、JPEGへ変換し、最後にZIPへまとめて返します。コードは長く見えても、役割で分けると5ブロックです。

1 準備	HEICを読めるようにし、必要ライブラリを読み込む。
2 画面	Streamlitでアップロード、品質、横幅のUIを作る。
3 条件	ファイル選択とボタン押下を確認する。
4 変換	回転補正、縮小、RGB化、JPEG保存。
5 出力	JPEG群をZIPへ入れ、ダウンロードさせる。



2. import文を読む

```
import io
import zipfile
from pathlib import Path
```

```
import streamlit as st
from PIL import Image, ImageOps
from pillow_heif import register_heif_opener
```

io	メモリ上に仮想ファイルを作る。
zipfile	複数JPEGをZIPへまとめる。
Path	ファイル名を安全に扱う。
streamlit	Web画面を作る。
Pillow	画像を開く・加工・保存する。
pillow-heif	PillowでHEIC/HEIFを扱えるようにする。

Streamlitは画面、Pillowは画像処理。この役割分担を押さえると一気に読みやすくなります。

3. HEIC対応とページ設定

```
register_heif_opener()
```

この1行でImage.open()がHEIC/HEIFを扱えるようになります。

```
st.set_page_config(  
    page_title="HEIC → JPG 一括変換",  
    page_icon="camera",  
    layout="centered"  
)
```

page_title	ブラウザタブのタイトル。
page_icon	ブラウザタブ用の簡易アイコン。
layout	中央寄せの読みやすい画面幅。

4. アップロードと品質

```
uploaded_files = st.file_uploader(  
    "HEIC / HEIFファイルを選択",  
    type=["heic", "heif"],  
    accept_multiple_files=True  
)
```

選ばれたファイル群がそのままuploaded_files変数へ入ります。UIとPython変数が直結しています。

```
quality = st.slider("JPEG品質", 70, 100, 92)
```

初期値92。ユーザーが動かした値がqualityへ入ります。

5. 最大横幅と0の意味

```
max_width = st.selectbox(
    "最大横幅",
    [0, 2560, 1920, 1600, 1280],
    format_func=lambda x: "元サイズ" if x == 0 else f"{x}px"
)
```

0は「幅0px」ではなく、リサイズしないという特別な合図です。画面表示だけ「元サイズ」に変えています。

6. if文が処理の門番

```
if uploaded_files:  
    if st.button("JPGへ変換", type="primary"):
```

ファイルがあり、かつ変換ボタンが押されたときだけ本処理へ進みます。ページを開いただけでは変換しません。

7. ZIPをメモリ上に作る

```
zip_buffer = io.BytesIO()
```

```
with zipfile.ZipFile(  
    zip_buffer,  
    "w",  
    zipfile.ZIP_DEFLATED  
    ) as zf:
```

ZIPはディスク上の一時ファイルではなく、RAM上に作ります。途中ファイルを散らかさない設計です。

8. 画像を1枚ずつ処理

```
for uploaded_file in uploaded_files:
    image = Image.open(uploaded_file)
    image = ImageOps.exif_transpose(image)
```

for文で1枚ずつ処理します。exif_transpose()はiPhone写真などの向き情報を実画像へ反映し、縦写真が横向きになる事故を防ぎます。

9. リサイズ計算

```
if max_width and image.width > max_width:
    ratio = max_width / image.width
    image = image.resize(
        (max_width, int(image.height * ratio)),
        Image.Resampling.LANCZOS
    )
```

小さい画像は拡大せず、大きい画像だけ縮小します。縦横比を保つためratioを使います。

```
例: 4032 x 3024 を最大幅1920へ
ratio = 1920 / 4032    0.47619
新しい高さ = 3024 x 0.47619    1440
結果 = 1920 x 1440
```

10. RGB化とJPEG保存

```
if image.mode != "RGB":  
    image = image.convert("RGB")  
  
jpg_buffer = io.BytesIO()  
image.save(  
    jpg_buffer,  
    format="JPEG",  
    quality=quality,  
    optimize=True  
)
```

拡張子だけ変更しているのではなく、本当にJPEGへ再エンコードしています。qualityは画面のスライダー値です。

11. ファイル名とZIP追加

```
jpg_name = Path(uploaded_file.name).stem + ".jpg"  
zf.writestr(jpg_name, jpg_buffer.getvalue())
```

```
IMG_4115.HEIC  
↓ stem  
IMG_4115  
↓ + ".jpg"  
IMG_4115.jpg
```

変換したJPEGをZIPへ直接追加します。

12. ダウンロード

```
zip_buffer.seek(0)
st.success("変換完了")
st.download_button(
    "JPGをZIPでダウンロード",
    zip_buffer,
    "converted_jpg.zip",
    "application/zip"
)
```

seek(0)でZIPの読み取り位置を先頭へ戻し、ブラウザのダウンロードボタンへ渡します。

13. ソース全体

ここまで読んだあとで全体を見ると、役割の塊として追えるようになります。

```
import io
import zipfile
from pathlib import Path

import streamlit as st
from PIL import Image, ImageOps
from pillow_heif import register_heif_opener

register_heif_opener()

st.set_page_config(
    page_title="HEIC → JPG 一括変換",
    page_icon="[camera]",
    layout="centered"
)

st.title("HEIC → JPG 一括変換")
st.write("HEIC / HEIF画像をまとめてJPGへ変換します。")

uploaded_files = st.file_uploader(
    "HEIC / HEIFファイルを選択",
    type=["heic", "heif"],
    accept_multiple_files=True
)

quality = st.slider("JPEG品質", 70, 100, 92)

max_width = st.selectbox(
    "最大横幅",
    [0, 2560, 1920, 1600, 1280],
    format_func=lambda x: "元サイズ" if x == 0 else f"{x}px"
)

if uploaded_files:
    if st.button("JPGへ変換", type="primary"):
        zip_buffer = io.BytesIO()
        with zipfile.ZipFile(zip_buffer, "w", zipfile.ZIP_DEFLATED) as zf:
            for uploaded_file in uploaded_files:
                image = Image.open(uploaded_file)
                image = ImageOps.exif_transpose(image)

                if max_width and image.width > max_width:
                    ratio = max_width / image.width
                    image = image.resize(
                        (max_width, int(image.height * ratio)),
                        Image.Resampling.LANCZOS
                    )

                if image.mode != "RGB":
                    image = image.convert("RGB")

                jpg_buffer = io.BytesIO()
                image.save(
                    jpg_buffer,
                    format="JPEG",
                    quality=quality,
                    optimize=True
                )

                jpg_name = Path(uploaded_file.name).stem + ".jpg"
                zf.writestr(jpg_name, jpg_buffer.getvalue())

            zip_buffer.seek(0)
            st.success("変換完了")
            st.download_button(
                "JPGをZIPでダウンロード",
                zip_buffer,
                "converted_jpg.zip",
                "application/zip"
            )
```

14. run.ps1の役割

Set-Location "C: \ ProjectG \ NOTE \ heic-to-jpg-web"

python -m streamlit run app.py

run.ps1はアプリ本体ではなく、Windowsからアプリを起動する発射スイッチです。

Set-Location	アプリフォルダへ移動。
python -m	現在のPython環境から実行。
streamlit run	Webサーバー起動。
app.py	アプリ本体。

デスクトップアイコン → run.ps1 → Streamlit → app.py、という4段構造です。

15. GitHubにある各ファイル

app.py	アプリ本体。
requirements.txt	必要ライブラリ一覧。
run.ps1	起動スクリプト。
push.ps1	GitHub更新補助。
mira-converter.ico	デスクトップ用ミラアイコン。
.gitignore	GitHubへ送らないものを指定。
README.md	説明書。

GitHubは保管庫であると同時に、アプリがどう変化したかを残す履歴帳です。

16. この1本で学べたPython基礎

import	他のライブラリを借りる。
変数	値に名前を付ける。
関数	Image.open()などと呼ぶ。
if	条件分岐。
for	繰り返し。
with	安全な開閉。
計算	ratioで縦横比維持。
変換	HEIC → RGB → JPEG → ZIP。
UI連携	画面操作とPython値を直結。

小さなアプリですが、Pythonの主要要素がかなり詰まっています。

17. 次の改造候補

PNG / WEBP対応	file_uploaderのtypeを増やす。
個別DL	JPEGごとのdownload_buttonを作る。
ファイル名変更	連番・日付・prefixを追加。
プレビュー	st.image()で表示。
容量比較	元サイズと変換後サイズを比較。
MIRA画面	ミラのキービジュアルを画面へ組み込む。

まず覚える3行

```
uploaded_files = st.file_uploader(...) # 入力
image.save(..., format="JPEG")      # 変換
st.download_button(...)              # 出力
```

この3点を軸に見ると、MIRA Converterは「入力を受け、加工し、返す」プログラムだと理解できます。