Conversor de imagem

Nomes:	RGMs:
Eduarda Fernandes	29204356
Davi Santos de Andrade	31075550
Johnatan Caetano	30087155
Everman	30333717
Daniel Medeiros	29381169

- ## Objetivo:

O objetivo do projeto é extrair as cores da imagem selecionada e transformá-las em valores RGB singulares, de cada pixel da imagem. E por fim ter a saída em Excel de todos os pixels da imagem e seus valores RGB nas respectivas posições nas células do Excel.

- ## Equipamentos/ferramentas utilizados:

Para desenvolver este conversor de imagem para matriz, utilizamos um notebook Samsung Core i5 de décima geração como plataforma principal. Optamos pelo Visual Studio Code devido à sua interface intuitiva e fácil manipulação. A linguagem de programação escolhida foi o Python, devido à sua versatilidade e eficiência. Para garantir uma organização adequada do projeto, decidimos utilizar o GitHub para armazenar e colaborar no código, permitindo que todos os membros do grupo pudessem contribuir de forma eficiente.

- ## Procedimento experimental:

➔ Para este projeto, utilizamos a linguagem Python e começamos a fazer as importações das bibliotecas necessárias para que o código seja executado corretamente.

```
1  from PIL import Image
2  import os
3  from openpyxl import Workbook
4  from openpyxl.styles import PatternFill
5  from openpyxl.utils import get_column_letter
6  import time
7
```

➔ Na segunda parte do código, desenvolvemos uma função que carrega uma imagem a partir do arquivo especificado.

Esta função é crucial para realizar a busca da imagem no diretório onde o projeto está sendo executado. Se a imagem for encontrada, ela será retornada; caso contrário, uma mensagem de erro será gerada.

```
8   def carregar_imagem(caminho):
9       if not os.path.exists(caminho):
10          raise FileNotFoundError(f"Arquivo '{caminho}' não encontrado.")
11
12      try:
13          imagem = Image.open(caminho)
14          return imagem
15      except Exception as e:
16          print("Ocorreu um erro ao carregar a imagem:", e)
17
```

➔ No terceiro trecho do código, elaboramos uma função para pegar as dimensões, altura e largura, da imagem selecionada. Em seguida, declaramos uma matriz cujas dimensões são definidas com base no tamanho da imagem.

```
18  def imagem_para_matriz(imagem):
19      largura, altura = imagem.size
20      matriz = []
21
22      for y in range(altura):
23          linha = []
24          for x in range(largura):
25              pixel = imagem.getpixel((x, y))
26              linha.append(pixel)
27          matriz.append(linha)
28
29      return matriz
30
```

➔ No quarto trecho do código, desenvolvemos uma função que cria um arquivo Excel para receber a imagem já convertida em uma matriz. Em cada célula do Excel, serão armazenados os valores RGB de cada pixel da imagem.

```python
def matriz_para_excel(matriz, caminho_imagem):
    wb = Workbook()
    ws = wb.active

    for y, linha in enumerate(matriz, start=1):
        for x, pixel in enumerate(linha, start=1):
            valor_pixel = f"({pixel[0]}, {pixel[1]}, {pixel[2]})"
            cell = ws.cell(row=y, column=x, value=valor_pixel)
            cell.alignment = cell.alignment.copy(wrapText=True)

    for coluna in ws.columns:
        max_length = 0
        coluna_letra = coluna[0].column_letter
        for cell in coluna:
            try:
                if len(str(cell.value)) > max_length:
                    max_length = len(cell.value)
            except:
                pass
        adjusted_width = (max_length + 2) * 1.2
        ws.column_dimensions[coluna_letra].width = adjusted_width

    nome_arquivo_original = os.path.splitext(os.path.basename(caminho_imagem))[0]
    nome_arquivo_convertido = nome_arquivo_original + '_converted.xlsx'
    caminho_arquivo = os.path.join(os.path.dirname(__file__), nome_arquivo_convertido)
    wb.save(caminho_arquivo)

    print(f"Matriz salva com sucesso no arquivo: {caminho_arquivo}")

```

➔ E por fim, desenvolvemos a função principal que executa os pedidos e cada função do código em ordem. Inicia-se então pedindo para o usuário colocar o nome do arquivo sem o formato, então ele executa as funções para converter a imagem em matriz no Excel, e por fim, exibe a mensagem de que a matriz foi salva com sucesso.

```python
def main():
    print("Imagens .png & .jpg são aceitas.")
    time.sleep(1.5)
    nome_arquivo = input("Digite o nome do arquivo de imagem (sem o formato): ")

    caminho_png = nome_arquivo + ".png"
    caminho_jpg = nome_arquivo + ".jpg"

    if os.path.exists(caminho_png):
        caminho_imagem = caminho_png
    elif os.path.exists(caminho_jpg):
        caminho_imagem = caminho_jpg
    else:
        print("Arquivo não encontrado.")
        return

    imagem = carregar_imagem(caminho_imagem)

    if imagem:
        print("Imagem carregada com sucesso!")
        matriz_pixels = imagem_para_matriz(imagem)
        print("Matriz de pixels criada.")
        print("Dimensões da matriz:", len(matriz_pixels), "x", len(matriz_pixels[0]))

        matriz_para_excel(matriz_pixels, caminho_imagem)


if __name__ == "__main__":
    main()

```

- Análise de resultados:

  A análise do resultado mostra que cada célula do Excel representa um pixel da imagem original, os valores do pixels são organizados em linhas e colunas, refletindo a estrutura bidimensional da imagem original, o script fornece uma representação detalhada e organizada dos pixels da imagem no formato de uma planilha Excel, isso facilita a análise e a interpretação das cores apresentadas da imagem original.

- Considerações finais:

  Foi bastante interessante essa experiência de criar um conversor que transforma uma imagem em uma matriz, e ver como a imagem é construída, pixel por pixel. O que nos impressiona, é saber que o processamento de imagem é muito rápido, mesmo quando a matriz é de grande escala.