```python
# Arquivo: termo.py

# import the necessary packages
from tkinter import *
from PIL import Image
from PIL import ImageTk
import numpy as np
import tkinter.filedialog as fdlg
import cv2
from crivo import Crivo
import os

def select_image():
    global panelA
    global vermelhoImg
    progresso['text'] = 'Processando...'
    path = fdlg.askopenfilename()
    image = cv2.imread(path)
    crivo = Crivo(image)
    vrml = crivo.vermelho()
    lrj = crivo.laranja()
    amrl = crivo.amarelo()
    vrd = crivo.verde()
    cno = crivo.ciano()
    azl = crivo.azul()
    vlt = crivo.violeta()
    mgt = crivo.magenta()
    brc = crivo.branco()
    resultado = [vrml, lrj, amrl, vrd, cno, azl, vlt, mgt, brc]

    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image = Image.fromarray(image)
    image = ImageTk.PhotoImage(image)
    if panelA is None:
        panelA = Label(image=image)
        panelA.image = image
        panelA.pack(side="right", padx=10, pady=10)

    else:
        panelA.configure(image=image)
```

```
        panelA.image = image

    lb['text'] = 'Identificação: {}\nData:
{}\n============================\nVermelho: {:.2f}%\nLaranja:
{:.2f}%\nAmarelo: {:.2f}%\nVerde: {:.2f}%\nCiano: {:.2f}%\nAzul:
{:.2f}%\nVioleta: {:.2f}%\nMagenta: {:.2f}%\nBranco:
{:.2f}%'.format(nome.get(), data.get(), resultado[0][0], resultado[1][0],
resultado[2][0], resultado[3][0], resultado[4][0], resultado[5][0],
resultado[6][0], resultado[7][0], resultado[8][0])

    vermelho['text'] = 'Vermelho: {:.2f}%'.format(resultado[0][0])
    laranja['text'] = 'Laranja: {:.2f}%'.format(resultado[1][0])
    amarelo['text'] = 'Amarelo: {:.2f}%'.format(resultado[2][0])
    verde['text'] = 'Verde: {:.2f}%'.format(resultado[3][0])
    ciano['text'] = 'Ciano: {:.2f}%'.format(resultado[4][0])
    azul['text'] = 'Azul: {:.2f}%'.format(resultado[5][0])
    violeta['text'] = 'Violeta: {:.2f}%'.format(resultado[6][0])
    magenta['text'] = 'Magenta: {:.2f}%'.format(resultado[7][0])
    branco['text'] = 'Branco: {:.2f}%'.format(resultado[8][0])

    salvar.place(x=40, y=580, width=160, height=30)

    progresso['text'] = 'Concluído!'
    return resultado

def salvar():
    fileName = fdlg.asksaveasfilename()
    try:
        file = open(fileName, 'w')
        file.write(lb['text'])
    except:
        pass
    finally:
        file.close()

def sobre():
    root = Tk()
    root.wm_title("Sobre")
    texto=("TermoCrivo: Versão 0.1")
    textONlabel = Label(root, text=texto)
```

```
    textONlabel.pack()

janela = Tk()
janela.title('TermoCrivo v0.1')
janela["bg"] = "#333"
janela.geometry('1280x720+200+50')
canvas = Canvas(janela)
canvas.pack()
img = ImageTk.PhotoImage(Image.open("logo.png"))
canvas.create_image(35, 5, anchor=NW, image=img)
canvas.place(width='2000', height=50)
janela.iconbitmap(bitmap='icon.ico')

imgicon = PhotoImage(file='icon.ico')
janela.tk.call('wm', 'iconphoto', janela._w, imgicon)

panelA = None
nomelb = Label(text="Identificação:", fg='white')
nomelb.place(x=40, y=80)
nomelb["bg"] = "#333"
nome = Entry(janela)
nome.place(x=40, y=100, width=150, height=25)
nome["bg"] = "gray"

datalb = Label(text="Data (DD/MM/AA):", fg='white')
datalb.place(x=40, y=130)
datalb["bg"] = "#333"
data = Entry(janela)
data.place(x=40, y=150, width=100, height=25)
data["bg"] = "gray"

abrir = Button(janela, text="Abrir", command=select_image)
abrir.place(x=140, y=150, width=50, height=25)

lb = Label(janela, text='')

resultado = Label(janela, text='Resultado:')
resultado['bg'] = 'white'

vermelho = Label(janela, text='')
```

```
vermelho['bg'] = 'white'
vermelho.place(x=45, y=220, width=150, height=20)
barraVermelho = Label(janela)
barraVermelho['bg'] = 'red'
barraVermelho.place(x=40, y=220, width=10, height=20)

laranja = Label(janela, text='')
laranja['bg'] = 'white'
laranja.place(x=45, y=260, width=150, height=20)

barraLaranja = Label(janela)
barraLaranja['bg'] = 'orange'
barraLaranja.place(x=40, y=260, width=10, height=20)

amarelo = Label(janela, text='')
amarelo['bg'] = 'white'
amarelo.place(x=45, y=300, width=150, height=20)
barraAmarelo = Label(janela)
barraAmarelo['bg'] = 'yellow'
barraAmarelo.place(x=40, y=300, width=10, height=20)

verde = Label(janela, text='')
verde['bg'] = 'white'
verde.place(x=45, y=340, width=150, height=20)
barraVerde = Label(janela)
barraVerde['bg'] = 'green'
barraVerde.place(x=40, y=340, width=10, height=20)

ciano = Label(janela, text='')
ciano['bg'] = 'white'
ciano.place(x=45, y=380, width=150, height=20)
barraCiano = Label(janela)
barraCiano['bg'] = 'cyan'
barraCiano.place(x=40, y=380, width=10, height=20)

azul = Label(janela, text='')
azul['bg'] = 'white'
azul.place(x=45, y=420, width=150, height=20)
barraAzul = Label(janela)
barraAzul['bg'] = 'blue'
```

```
barraAzul.place(x=40, y=420, width=10, height=20)

violeta = Label(janela, text='')
violeta['bg'] = 'white'
violeta.place(x=45, y=460, width=150, height=20)
barraVioleta = Label(janela)
barraVioleta['bg'] = 'purple'
barraVioleta.place(x=40, y=460, width=10, height=20)

magenta = Label(janela, text='')
magenta['bg'] = 'white'
magenta.place(x=45, y=500, width=150, height=20)
barraMagenta = Label(janela)
barraMagenta['bg'] = 'magenta'
barraMagenta.place(x=40, y=500, width=10, height=20)

branco = Label(janela, text='')
branco['bg'] = 'white'
branco.place(x=45, y=540, width=150, height=20)
barraBranco = Label(janela)
barraBranco['bg'] = 'gray'
barraBranco.place(x=40, y=540, width=10, height=20)

salvar = Button(janela, text="Salvar", command=salvar)

progresso = Label(janela, text='Nada a processar...', bd=1, relief=SUNKEN,
anchor=W)
progresso.pack(side=BOTTOM, fill=X)

janela.mainloop()


# Arquivo: crivo.py

from PIL import Image
from PIL import ImageTk
import numpy as np
import cv2
```

```
class Crivo():
    def __init__(self, foto):
        self.foto = foto

    #VERMELHO
    def vermelho(self):
        total = self.foto.shape[1] * self.foto.shape[0]
        imagemHSV = cv2.cvtColor(self.foto, cv2.COLOR_BGR2HSV)
        lowVermelho = np.array([0,78,51])
        upVermelho = np.array([7, 255, 255])

        lowVermelho2 = np.array([173,78,51])
        upVermelho2 = np.array([179, 255, 255])

        maskVermelho =  cv2.inRange(imagemHSV, lowVermelho, upVermelho)
        maskVermelho2 =  cv2.inRange(imagemHSV, lowVermelho2, upVermelho2)
        outputVermelho= cv2.bitwise_and(imagemHSV, self.foto, mask =
maskVermelho + maskVermelho2)

        preto = 0
        for y in range(0, outputVermelho.shape[0], 1):
            for x in range(0, outputVermelho.shape[1], 1):
                (h, s, v) = outputVermelho[y, x]
                if (v==0):
                    preto = preto + 1

        vermelhoQtd = ((total - preto)/total)*100
        print('Vermelho: ', vermelhoQtd)
        # cv2.imshow('Vermelho', np.hstack([self.foto, outputVermelho]))
        vermelho = [vermelhoQtd, outputVermelho]
        return vermelho

    # LARANJA
    def laranja(self):
        total = self.foto.shape[1] * self.foto.shape[0]
        imagemHSV = cv2.cvtColor(self.foto, cv2.COLOR_BGR2HSV)
        lowLaranja = np.array([8, 78, 51])
        upLaranja = np.array([25, 255, 255])

        maskLaranja = cv2.inRange(imagemHSV, lowLaranja, upLaranja)
```

```
        outputLaranja = cv2.bitwise_and(imagemHSV, self.foto, mask=maskLaranja)

        preto = 0
        for y in range(0, outputLaranja.shape[0], 1):
            for x in range(0, outputLaranja.shape[1], 1):
                (h, s, v) = outputLaranja[y, x]
                if (v == 0):
                    preto = preto + 1

        laranjaQtd = ((total - preto)/total)*100
        print('Laranja: ', laranjaQtd)
        # cv2.imshow('Laranja', np.hstack([self.foto, outputLaranja]))
        laranja = [laranjaQtd, outputLaranja]
        return laranja

    #AMARELO
    def amarelo(self):
        total = self.foto.shape[1] * self.foto.shape[0]
        imagemHSV = cv2.cvtColor(self.foto, cv2.COLOR_BGR2HSV)
        lowAmarelo = np.array([26,78,51])
        upAmarelo = np.array([30,255,255])

        maskAmarelo =  cv2.inRange(imagemHSV, lowAmarelo, upAmarelo)
        outputAmarelo= cv2.bitwise_and(self.foto, imagemHSV, mask =
maskAmarelo)

        preto = 0
        for y in range(0, outputAmarelo.shape[0], 1):
            for x in range(0, outputAmarelo.shape[1], 1):
                (h, s, v) = outputAmarelo[y, x]
                if (v==0):
                    preto = preto + 1

        amareloQtd = ((total - preto)/total)*100
        print('Amarelo: ', amareloQtd)
        # cv2.imshow("Amarelo", np.hstack([self.foto, outputAmarelo]))
        amarelo = [amareloQtd, outputAmarelo]
        return amarelo

    #VERDE
```

```
def verde(self):
    total =  self.foto.shape[1] *  self.foto.shape[0]
    imagemHSV = cv2.cvtColor(self.foto, cv2.COLOR_BGR2HSV)
    lowVerde = np.array([31,78,51])
    upVerde = np.array([83,255,255])

    maskVerde =  cv2.inRange(imagemHSV, lowVerde, upVerde)
    outputVerde= cv2.bitwise_and(self.foto, imagemHSV, mask = maskVerde)

    preto = 0
    for y in range(0, outputVerde.shape[0], 1):
        for x in range(0, outputVerde.shape[1], 1):
            (h, s, v) = outputVerde[y, x]
            if (v==0):
                preto = preto + 1

    verdeQtd = ((total - preto)/total)*100
    print('Verde ', verdeQtd)
    # cv2.imshow("Verde", np.hstack([self.foto, outputVerde]))
    verde = [verdeQtd, outputVerde]
    return verde

#CIANO
def ciano(self):
    total =  self.foto.shape[1] *  self.foto.shape[0]
    imagemHSV = cv2.cvtColor( self.foto, cv2.COLOR_BGR2HSV)
    lowCiano = np.array([84,78,51])
    upCiano = np.array([98,255,255])

    maskCiano =  cv2.inRange(imagemHSV, lowCiano, upCiano)
    outputCiano= cv2.bitwise_and( self.foto, imagemHSV, mask = maskCiano)

    preto = 0
    for y in range(0, outputCiano.shape[0], 1):
        for x in range(0, outputCiano.shape[1], 1):
            (h, s, v) = outputCiano[y, x]
            if (v==0):
                preto = preto + 1

    cianoQtd = ((total - preto)/total)*100
```

```
        print('Ciano: ', cianoQtd)
        # cv2.imshow("Ciano", np.hstack([self.foto, outputCiano]))
        ciano = [cianoQtd, outputCiano]
        return ciano

    #AZUL
    def azul(self):
        total =  self.foto.shape[1] *  self.foto.shape[0]
        imagemHSV = cv2.cvtColor( self.foto, cv2.COLOR_BGR2HSV)
        lowAzul = np.array([99,78,51])
        upAzul = np.array([128,255,255])

        maskAzul =  cv2.inRange(imagemHSV, lowAzul, upAzul)
        outputAzul = cv2.bitwise_and( self.foto, imagemHSV, mask = maskAzul)

        preto = 0
        for y in range(0, outputAzul.shape[0], 1):
            for x in range(0, outputAzul.shape[1], 1):
                (h, s, v) = outputAzul[y, x]
                if (v==0):
                    preto = preto + 1

        azulQtd = ((total - preto)/total)*100
        print('Azul ', azulQtd)
        # cv2.imshow("Azul", np.hstack([self.foto, outputAzul]))
        azul = [azulQtd, outputAzul]
        return azul

    #VIOLETA
    def violeta(self):
        total =  self.foto.shape[1] *  self.foto.shape[0]
        imagemHSV = cv2.cvtColor( self.foto, cv2.COLOR_BGR2HSV)
        lowVioleta = np.array([129,78,51])
        upVioleta = np.array([143,255,255])

        maskVioleta =  cv2.inRange(imagemHSV, lowVioleta, upVioleta)
        outputVioleta = cv2.bitwise_and( self.foto, imagemHSV, mask =
maskVioleta)

        preto = 0
```

```
        for y in range(0, outputVioleta.shape[0], 1):
            for x in range(0, outputVioleta.shape[1], 1):
                (h, s, v) = outputVioleta[y, x]
                if (v==0):
                    preto = preto + 1

        violetaQtd = ((total - preto)/total)*100
        print('Violeta: ', violetaQtd)
        # cv2.imshow('Violeta', np.hstack([self.foto, outputVioleta]))
        violeta = [violetaQtd, outputVioleta]
        return violeta

    #MAGENTA
    def magenta(self):
        total =  self.foto.shape[1] *  self.foto.shape[0]
        imagemHSV = cv2.cvtColor( self.foto, cv2.COLOR_BGR2HSV)
        lowMagenta = np.array([144,78,51])
        upMagenta = np.array([172,255,255])

        maskMagenta =  cv2.inRange(imagemHSV, lowMagenta, upMagenta)
        outputMagenta = cv2.bitwise_and( self.foto, imagemHSV, mask =
maskMagenta)

        preto = 0
        for y in range(0, outputMagenta.shape[0], 1):
            for x in range(0, outputMagenta.shape[1], 1):
                (h, s, v) = outputMagenta[y, x]
                if (v==0):
                    preto = preto + 1

        magentaQtd = ((total - preto)/total)*100
        print('Magenta: ', magentaQtd)
        # cv2.imshow('Magenta', np.hstack([self.foto, outputMagenta]))
        magenta = [magentaQtd, outputMagenta]
        return magenta

    #BRANCO
    def branco(self):
        total =  self.foto.shape[1] *  self.foto.shape[0]
        imagemHSV = cv2.cvtColor( self.foto, cv2.COLOR_BGR2HSV)
```

```
        lowBranco = np.array([0,0,204])
        upBranco = np.array([179,77,255])

        maskBranco =  cv2.inRange(imagemHSV, lowBranco, upBranco)
        outputBranco = cv2.bitwise_and( self.foto, imagemHSV, mask =
maskBranco)

        preto = 0
        for y in range(0, outputBranco.shape[0], 1):
            for x in range(0, outputBranco.shape[1], 1):
                (h, s, v) = outputBranco[y, x]
                if (v==0):
                    preto = preto + 1

        brancoQtd = ((total - preto)/total)*100
        print('Branco: ', brancoQtd)
        # cv2.imshow("Branco", np.hstack([self.foto, outputBranco]))
        branco = [brancoQtd, outputBranco]
        return branco


# Especificações recomendadas:
# Sistema Operacional Windows 10;
# 2gb de memória RAM;
# 50mb livre de espaço;
# Processador Intel i3 2100;
```