

VERSI 2.1
DESEMBER 2025



PEMROGRAMAN FUNGSIONAL

**MODUL 6 - Functional Programming Implementation:
Image Processing**

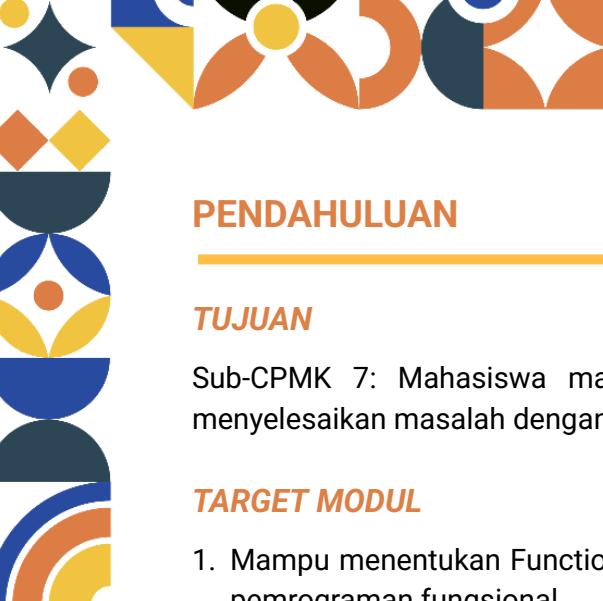
DISUSUN OLEH:

FERA PUTRI AYU L., S.KOM., M.T.

ALFI AULIA AZZAHRA

RAHMATUN NIKMAH

**TIM LABORATORIUM INFORMATIKA
UNIVERSITAS MUHAMMADIYAH MALANG**



PENDAHULUAN

TUJUAN

Sub-CPMK 7: Mahasiswa mampu mendesain program dengan teknik yang tepat untuk menyelesaikan masalah dengan menggunakan paradigma pemrograman fungsional (P6).

TARGET MODUL

1. Mampu menentukan Function yang ada dalam prosesing citra sederhana berdasarkan teknik pemrograman fungsional.

PERSIAPAN

1. Komputer/Laptop
2. Python, Google Collab/VS Code/Jupyter Notebook
3. [Source Code Colab Modul 6](#)
4. [Drive Foto Modul 6](#)

KEYWORDS

Image Processing, Matplotlib, Pillow

TABLE OF CONTENTS

PENDAHULUAN.....	2
TUJUAN.....	2
TARGET MODUL.....	2
PERSIAPAN.....	2
KEYWORDS.....	2
TABLE OF CONTENTS.....	2
1. WHAT IS IMAGE PROCESSING.....	4
2. IMAGE PROCESSING IN PYTHON.....	5
3. IMAGE PROCESSING WITH MATPLOTLIB.....	6
3.1. OPEN IMAGE IN VS CODE.....	6
3.2. OPEN IMAGE IN GOOGLE COLAB.....	7
3.2.1. LOCAL TO COLAB.....	7
3.2.2. IMAGE FROM GOOGLE DRIVE.....	8
3.3. MATPLOTLIB FEATURES.....	8
3.4. SIMPLE IMAGE PROCESSING USING NUMPY.....	14
4. IMAGE PROCESSING WITH PILLOW.....	16
4.1. OPEN IMAGE FILE IN PILLOW.....	16
4.2. RESIZE: IMPERATIF VS FUNGSIONAL.....	17
4.3. PIXEL MANIPULATION USING HIGH-ORDER FUNCTIONS.....	19



4.4. PILLOW FEATURES UNTUK MANIPULASI GAMBAR.....	21
4.4.1. CROPPING IMAGES.....	21
4.4.2. ROTATING IMAGES.....	22
4.4.3. FLIPPING IMAGES.....	23
4.4.4. IMAGES ENCHANCER.....	24
4.4.5. FILTERING IMAGES.....	25
4.5. MERGING IMAGES/OVERLAY IMAGES.....	26
CODELAB.....	29
CODELAB 1.....	29
CODELAB 2.....	30
TUGAS.....	31
TUGAS 1.....	32
TUGAS 2.....	33
NIM GENAP.....	33
NIM GANJIL.....	33
RUBIK PENILAIAN.....	34



Halo! Tidak terasa kita sudah sampai di akhir semester dan modul terakhir Pemrograman Fungsional. Setelah melalui berbagai tantangan tugas modul dan pembelajaran sepanjang semester, pada materi terakhir ini kita akan membahas Image Processing dengan Python, tentu sambil menerapkan konsep fungsional yang sudah dipelajari. Yuk, kita mulai!

1. WHAT IS IMAGE PROCESSING

Pemrosesan Citra (Image Processing) adalah sebuah proses pengolahan sinyal di mana inputnya berupa gambar digital yang kemudian diubah menjadi gambar lain sebagai *output*-nya menggunakan teknik tertentu.

Tujuan utama dari proses ini adalah untuk **memperbaiki kesalahan data gambar** yang mungkin terjadi selama proses transmisi dan akuisisi sinyal, serta untuk meningkatkan kualitas visual gambar agar lebih mudah diinterpretasikan, baik oleh sistem penglihatan manusia maupun oleh mesin. Sederhananya, ini adalah teknik untuk **mengedit atau memodifikasi** gambar digital agar menjadi lebih jelas atau memiliki kualitas yang lebih baik. Contoh modifikasi ini bisa termasuk memperbaiki warna, menghilangkan *noise* (bintik/gangguan), memperbesar atau memperkecil gambar, dan banyak lagi.

Secara fundamental, proses ini sangat erat kaitannya dengan konsep **paradigma pemrograman fungsional**. Mengapa? Karena pemrosesan citra dapat dilihat sebagai rangkaian **fungsi transformasi**. Input gambar (*sebagai data*) dimasukkan ke dalam suatu fungsi (*proses*) yang kemudian menghasilkan output gambar baru (*data yang ditransformasi*).

Beberapa langkah dasar dalam Pemrosesan Citra yang sesuai dengan konsep ini meliputi:

1. **Image Acquisition (Akuisisi Citra)**: Menangkap atau mengimpor gambar awal melalui perangkat keras seperti kamera atau pemindai. Ini adalah data input kita.
2. **Preprocessing (Pra-pemrosesan)**: Mengubah gambar sebelum dianalisis (misalnya, mengubah ukuran, rotasi, atau penghapusan noise). Ini adalah **fungsi transformasi awal**.
3. **Feature Detection (Deteksi Fitur)**: Mengidentifikasi piksel atau area yang menarik, seperti tepi, sudut, atau objek. Fungsi ini **mengekstraksi informasi** dari data.
4. **Analysis (Analisis)**: Mengekstraksi informasi yang lebih bermakna dari gambar menggunakan fitur yang terdeteksi.
5. **Manipulation (Manipulasi/Transformasi Akhir)**: Mengubah gambar berdasarkan informasi yang telah diekstrak (misalnya, *filtering* atau *morphing*). Ini adalah **fungsi transformasi utama** yang menghasilkan *output* akhir.

Nahhh, kabar baiknya, Kalian **tidak perlu** membuat semua fungsi transformasi ini dari awal. Dalam dunia pemrograman modern, terutama yang mengadopsi prinsip fungsional, kita dapat memanfaatkan library atau pustaka yang sudah tersedia.

Inilah yang membuat Pemrosesan Citra menjadi efisien: **kita cukup memilih atau menentukan fungsi yang sesuai dengan 'apa' yang kita inginkan**, tanpa perlu tahu detail rumit tentang bagaimana proses di baliknya (*how*). Kalian ingat kan dengan statement ini?

Yap, hal ini sejalan dengan konsep **abstraksi dan modularitas**, di mana fokus kita beralih dari **prosedur (bagaimana melakukannya)** ke **fungsi (apa hasilnya)**. Ini adalah inti dari paradigma fungsional: Anda hanya perlu tahu nama fungsi grayscale(image) untuk mengubah gambar menjadi hitam putih, dan **library** akan menangani semua perhitungan piksel di dalamnya!

2. IMAGE PROCESSING IN PYTHON

Python merupakan salah satu bahasa pemrograman yang memiliki banyak library yang mendukung Pemrosesan Citra yang mana dirangkum sebagai berikut:

1. PILLOW : [PILLOW](#) adalah versi modern dari Python Imaging Library (PIL) yang memungkinkan Anda untuk membuka, memanipulasi, dan menyimpan berbagai format gambar. Ini menyediakan fungsi dasar untuk operasi seperti pengubahan ukuran, pemotongan, rotasi, dan pengubahan warna gambar.
2. Matplotlib : [Matplotlib](#) adalah pustaka untuk membuat grafik dan visualisasi data di Python. Gambar yang dihasilkan oleh Matplotlib biasanya berupa grafik, plot, diagram, atau visualisasi data lainnya, termasuk data gambar/image.
3. OpenCV : [OpenCV](#) (Open Source Computer Vision Library) adalah pustaka yang kuat untuk pemrosesan gambar dan visi komputer. Ini mencakup berbagai alat untuk deteksi objek, pengenalan wajah, segmentasi gambar, penghapusan noise, deteksi tepi, dan banyak lagi.
4. scikit-image : [scikit-image](#) adalah pustaka yang berfokus pada pemrosesan gambar berbasis ilmiah, dibangun di atas NumPy dan SciPy. Ini menyediakan fungsi untuk segmentasi, transformasi geometris, analisis gambar, dan deteksi fitur.
5. imageio : [imageio](#) adalah pustaka yang digunakan untuk membaca dan menulis gambar dalam berbagai format. Ini sangat berguna untuk menangani berbagai jenis file gambar, seperti GIF, JPEG, PNG, dan juga video.
6. SimpleCV : [SimpleCV](#) adalah pustaka yang dirancang untuk memudahkan pemrosesan gambar dan visi komputer. Dengan antarmuka yang sederhana, SimpleCV memudahkan pemula untuk melakukan tugas-tugas seperti pengenalan bentuk, pelacakan objek, dan pengolahan gambar dasar.
7. Mahotas : [Mahotas](#) adalah library untuk pengolahan citra berbasis perhitungan matematika dan statistik. Ini menyediakan alat untuk ekstraksi fitur, analisis morfologi, segmentasi, dan transformasi lainnya.
8. Pytorch dan Tensorflow : [Pytorch](#) dan [TensorFlow](#) adalah pustaka yang kuat untuk pembelajaran mesin dan pembelajaran mendalam (deep learning). Keduanya sering digunakan dalam pemrosesan gambar untuk membangun dan melatih model neural network, termasuk model convolutional neural network (CNN) untuk tugas-tugas seperti pengenalan gambar dan deteksi objek

Dari banyaknya library diatas, di modul ini kita akan mempelajari image-processing sederhana menggunakan Matplotlib image dan PILLOW.



3. IMAGE PROCESSING WITH MATPLOTLIB

Pada modul sebelumnya, kita menggunakan *library Matplotlib* di Python untuk visualisasi data (membuat *plot*, grafik, dan *chart*). Selain itu, Matplotlib juga memiliki kemampuan dasar untuk **menampilkan** dan melakukan operasi I/O (Input/Output) sederhana pada gambar/citra digital.

Matplotlib sendiri **bukanlah library utama** untuk pemrosesan citra yang kompleks (seperti pengenalan objek atau transformasi geometris tingkat lanjut), melainkan sering digunakan sebagai **alat bantu visualisasi** hasil dari pemrosesan citra yang sebenarnya.

Nah, untuk package yang akan digunakan dari *library* ini adalah:

```
from matplotlib import pyplot as plt  
from matplotlib import image as im
```

dimana `plt` digunakan untuk menampilkan hasil gambar yang sudah kita modifikasi, dan `im` digunakan untuk proses modifikasi gambarnya.

3.1. OPEN IMAGE IN VS CODE

Ada berbagai metode untuk membuka file gambar dengan matplotlib, salah satunya adalah dengan memanfaatkan fungsi-fungsi yang tersedia pada library tersebut. Contohnya, kita dapat menggunakan `imread` dari matplotlib untuk membaca file gambar yang tersimpan di penyimpanan lokal. Dan `imshow` untuk menampilkan gambarnya.

```
#pastikan package (alias) matplotlib plt dan im sudah didownload dan  
di-import di file kalian!  
  
## Ini hanya contoh dan hanya bisa dijalankan di local yaa  
akses_file = im.imread(r'C:\Users\Rahma\Downloads\umm_new.png')  
  
plt.imshow(akses_file)  
plt.axis('off')  
plt.show()
```

Output:



Dalam contoh diatas kita terlebih dahulu harus mengimport library matplotlib dan sub-library/modul apa yang ingin kita gunakan disini kita menggunakan `image` untuk



mengakses file lokal dan pyplot untuk 'canvas' atau figur ketika kita menjalankan program. Untuk mengakses file lokal pastikan 'path' file sudah benar jika mengalami error unicode coba menambahkan 'r' untuk menghindari escape character.

3.2. OPEN IMAGE IN GOOGLE COLAB

Ada sedikit perbedaan saat ingin mengakses image menggunakan google collab karena google collab berbasis cloud based bukan software lokal seperti visual studio code. Simak beberapa contoh berikut :

3.2.1. LOCAL TO COLAB

Selain mengupload file di collab seperti di modul 5, kita juga bisa menggunakan fitur upload ketika kita melakukan run atau meletakkan fitur upload hanya di saat code kita dijalankan seperti di bawah ini:

```
from google.colab import files

# Mengunggah gambar dari perangkat lokal ke Google Collab
uploaded = files.upload()

# Mendapatkan nama file gambar yang diunggah
uploaded_filename = next(iter(uploaded.keys()))

# Membaca gambar menggunakan Matplotlib Image
img = im.imread(uploaded_filename)

plt.imshow(img)
plt.axis('off') # Tidak menampilkan sumbu koordinat
plt.show()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed

Saving umm_new.png to umm_new.png



3.2.2. IMAGE FROM GOOGLE DRIVE

Ketika kita ingin mengakses google drive dalam collab kita harus melakukan mounting dan memberikan akses untuk collab.

```
from google.colab import drive  
drive.mount('/content/drive')
```

Jika sudah pernah melakukan mounting, selanjutnya kita tidak perlu melakukan 'import drive' lagi karena proses mounting cukup 1x saja.

```
# hanya contoh, gunakan direktori drive pribadi untuk mengakses  
gambar_path = '/content/drive/MyDrive/Fungsional/umm_new.png'  
  
img = im.imread(gambar_path)  
  
plt.imshow(img)  
plt.axis('off')  
plt.show()
```

Output:



3.3. MATPLOTLIB FEATURES

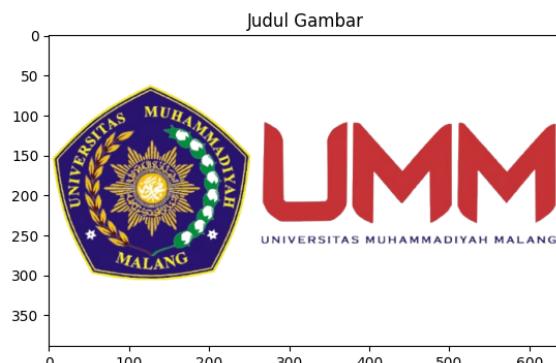
Beberapa fitur dalam pyplot yang sudah kita pelajari di modul 5 juga dapat kita implementasikan disini. Berikut beberapa fitur lain Matplotlib untuk Image Processing.

1. Menampilkan judul gambar dengan title().

```
#tampilkan gambar dengan menambahkan judul  
plt.imshow(image) #ganti variabel image dengan variabel imread  
#kalian  
plt.title("Judul Gambar")  
plt.show()
```



Output:



- Menampilkan atau menyembunyikan sumbu dengan axis().

```
# disini kita akan melakukan mounting dari drive yang sudah
disediakan

#Kalian bisa merubah/mengganti path file sesuai dengan lokasi
image kalian berada
gambar_kucing =
'/content/drive/MyDrive/Fungsional/kucing_duduk.jpg'

img = im.imread(gambar_kucing)

#tampilan output tanpa sumbu axis dengan menambahkan judul
plt.imshow(img)
plt.axis('off')
plt.title("Gambar Kucing Duduk")
plt.show()
```

Output:



3. Menambahkan teks dan anotasi ke gambar dengan `annotate()` dan `text()`.
Contoh di bawah ini kita memberikan anotasi ke dalam gambar yang kita upload.
Anotasi adalah memberikan informasi tambahan atau penjelasan tentang bagian tertentu dari gambar atau grafik (biasanya menggunakan panah).

```
# --- PLOTTING IMAGE ---
plt.figure(figsize=(12, 8))
plt.imshow(img)
plt.title("Gambar Logo Universitas Muhammadiyah Malang (UMM)  
dengan Anotasi", fontsize=16)

# Anotasi 1: Menunjuk ke Logo/Lambang UMM
plt.annotate(
    'Lambang Universitas ',
    xy=(70, 100),
    xytext=(120, 20),
    arrowprops=dict(facecolor='navy', shrink=0.05, width=2,
headwidth=8), # panah biru
    fontsize=14,
    color='navy',
    fontweight='bold',
    ha='center'
)

# Anotasi 2: Menunjuk ke Tulisan "MUHAMMADIYAH"
plt.annotate(
    'Tulisan "MUHAMMADIYAH"',
    xy=(500, 260),
    xytext=(500, 350),
    arrowprops=dict(facecolor='green', shrink=0.05, width=2,
headwidth=8), # panah hijau
    fontsize=14,
    color='green',
    fontweight='bold',
    ha='center'
)

plt.axis('off')
plt.show()
```



Output:



4. Menyimpan gambar.

```
# Menyimpan gambar
plt.imshow(image)
plt.axis('off')
plt.title("Gambar untuk Disimpan")

#kode untuk menyimpan gambar
plt.savefig('masukkan/path/untuk/disimpan/sini')

plt.show()
```

output : Perintah plt.savefig akan menyimpan gambar ke dalam direktori yang ditentukan.

Perlu diperhatikan bahwa di library ini, kita hanya menampilkan dan menambahkan anotasi. Yang artinya, Matplotlib tidak bisa digunakan untuk image processing (modifikasi gambar), jika ingin melakukan modifikasi kalian bisa menggunakan library tambahan lain yang sudah disediakan oleh Python.



📌 Latihan 0 - Preparation

Jalankan kedua cell kode dibawah ini!!! kode ini bertujuan untuk menampilkan hasil gambar kalian (jadi tidak perlu membuat kode plt yang berulang-ulang lagi- plt.imshow, plt.title, plt.axis, plt.show).

Disini kita manfaatkan closure dan high order function gaes.. Biar apa, biar kehidupan code kita jadi lebih simple-ga perlu lagi mengulang-ulang code plt=plt setiap mau nampilin gambar. Coba buktikan sendiri aja.

```
import matplotlib.pyplot as plt

def buat_penampil(cols=3, figsize=(12,4)):
    def tampilkan(img_list, titles=None, cmaps=None):
        rows = (len(img_list) + cols - 1) // cols
        plt.figure(figsize=figsize)

        for i, img in enumerate(img_list, 1):
            plt.subplot(rows, cols, i)

            # jika ada cmap untuk gambar
            if cmaps and i-1 < len(cmaps) and cmaps[i-1]:
                plt.imshow(img, cmap=cmaps[i-1])
            else:
                plt.imshow(img)

            plt.axis('off')

            if titles and i-1 < len(titles):
                plt.title(titles[i-1])

        plt.tight_layout()
        plt.show()
    return tampilkan

def tampil_foto(img, posisi=(1,1,1), title=None, cmap=None):
    plt.subplot(*posisi)
    if cmap:
        plt.imshow(img, cmap=cmap)
    else:
        plt.imshow(img)
    if title:
        plt.title(title)
```



```
plt.axis("off")

#Instance pemanggilan multi-gambar
penampil4cols = buat_penampil(cols=4)

#jika ingin menampilkan 1 gambar saja,
#silahkan langsung panggil fungsinya.
```

Selanjutnya, kode ini bertujuan untuk mengupload foto kalian yang nantinya akan digunakan untuk latihan selanjutnya.

```
# wadah galeri akan digunakan sebagai menyimpan hasil prosesing kalian di
Latihan selanjutnya
galeri_latihan = []

# Upload Gambar
print("Silakan upload gambar yang ingin Anda gunakan untuk praktikum ini.")
print("Disarankan menggunakan gambar ukuran sedang (misal: JPG/PNG).")
print("Sangat diasarankan untuk memilih foto yang lucu.")

uploaded = files.upload()

if uploaded:
    filename = next(iter(uploaded))

try:
    imgLat = im.imread(filename)
    print(f"\nSukses! Gambar '{filename}' berhasil dimuat.")
    print(f"Ukuran asli: {imgLat.size}")

    # Tampilkan gambar awal
    tampil_foto(imgLat, title="Gambar Asli")

    # Simpan gambar asli ke galeri sebagai pembanding
    galeri_latihan.append(("Gambar Asli", imgLat))

except Exception as e:
    print(f"Error: File yang diupload bukan gambar yang valid. ({e})")
else:
    print("Anda tidak mengupload file. Silakan jalankan ulang sel ini jika ingin mengupload.")
```

3.4. SIMPLE IMAGE PROCESSING USING NUMPY

Apakah kalian tahu jika **gambar (citra digital)** itu pada dasarnya merupakan **data numerik**?

Secara teknis, setiap gambar yang kita lihat di layar komputer atau ponsel adalah **matriks (atau array) bilangan** yang tersusun rapi. Setiap titik kecil dalam gambar (disebut **piksel**) memiliki nilai numerik yang mewakili warnanya.

- Pada **citra grayscale** (hitam-putih), setiap piksel direpresentasikan oleh satu angka (biasanya dari 0 hingga 255), di mana **0 adalah hitam pekat** dan **255 adalah putih pekat**.
- Pada **citra berwarna** (seperti RGB), setiap piksel direpresentasikan oleh **tiga angka** (atau lebih) yang mewakili intensitas warna **Merah (R)**, **Hijau (G)**, dan **Biru (B)**.

Oleh karena itu, untuk melakukan pemrosesan pada gambar, kita dapat menggunakan **NumPy**, library fundamental di Python yang dirancang khusus untuk bekerja dengan array multidimensi (matriks). Perhatikan proses gambar dibawah ini:

```
'''fungsi kode ini memuat gambar, mengonversinya menjadi grayscale,
dan menampilkannya dalam dua bentuk, gambar asli dan gambar
grayscale.

pre-cond: gambar yang dimuat harus memiliki format yang didukung oleh
matplotlib (misalnya, .webp, .jpg, .png);
           input gambar memiliki format RGB untuk proses konversi
menjadi grayscale yang sesuai.'''


import numpy as np

#kalian bisa mengganti gambar yang di read disini
img = im.imread('/content/drive/MyDrive/Fungsional/umm_new.png')

#Untuk membuat gambar grayscale
gray_image = np.dot(img[..., :3], [0.2989, 0.587, 0.114])
```

Kode ini memuat gambar dari file yang diberikan menggunakan `im.imread()` kemudian gambar tersebut dikonversi menjadi grayscale menggunakan rumus luminosity (`np.dot(img[..., :3], [0.2989, 0.587, 0.114])`) (Bisa kalian cari di internet), yang mengalikan setiap golongan warna RGB(merah, hijau, biru) dengan koefisien yang sesuai untuk menghasilkan intensitas kecerahan yang diinginkan (coba ubah nilainya dan perhatikan bedanya).

Selanjutnya di sel berikutnya kita akan menampilkannya dengan ukuran plot 10 x 5 inci. Pada subplot pertama kita tampilkan `img` bentuk asli (RGB). Gambar grayscale kemudian ditampilkan pada subplot kedua dengan pemetaan warna `gray(cmap)`. Kedua gambar tersebut (gambar asli dan grayscale) ditampilkan berdampingan dalam layout 1x3, dengan sumbu (axis) dinonaktifkan untuk tampilan yang lebih bersih.

```
# menampilkan gambar asli dan hasil pengolahan menggunakan plt
plt.figure(figsize=(10,5))

plt.subplot(1, 3, 1)
plt.imshow(img)
plt.title('Gambar Awal')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.imshow(gray_image, cmap=plt.get_cmap('gray'))
plt.title('Gambar Setelah')
plt.axis('off')
```

Output:

Gambar Awal



Gambar Setelah



Kalian bisa melihat bahwa untuk setiap gambar yang ingin ditampilkan, kita harus mengulang serangkaian kode plt seperti plt.subplot, plt.imshow, plt.title, dan plt.axis('off'). Jika kita memiliki puluhan langkah pemrosesan, kita harus mengulang kode ini berkali-kali—sebuah pendekatan yang tidak efisien dan rawan kesalahan.

Nah karena kita sudah menjalankan fungsi yang ada di latihan 0 tadi, kita hanya perlu memakai kembali fungsi tersebut dengan memanggil fungsi tersebut, lalu kita bisa tambahkan titlenya juga

```
#fungsi menampilkan foto
print("penampil menggunakan fungsi")
penampil([img, gray_image],
        ['Gambar Awal', 'Gambar Setelah'],
        [None, 'gray']) #tambahkan cmaps untuk gambar setelah agar
menampilkan grayscale
()
```

Output:

Gambar Awal



Gambar Setelah



4. IMAGE PROCESSING WITH PILLOW

PILLOW adalah library Python yang digunakan untuk memanipulasi gambar (image processing). Dengan PILLOW, kita dapat melakukan berbagai operasi pada gambar seperti membuka, menyimpan, mengedit, dan menggabungkan gambar. Library ini sangat berguna dalam pengembangan aplikasi yang melibatkan pemrosesan citra. Mari kita simak penerapan PILLOW dalam image processing :

4.1. OPEN IMAGE FILE IN PILLOW

Kita sudah mengetahui bagaimana cara untuk mengakses file menggunakan library matplotlib, untuk library PILLOW tidak jauh berbeda baik menggunakan text editor dan google collab yaitu kita bisa mengakses menggunakan path local, fitur upload dan mounting ke google drive. Perbedaan hanya pada function yang dipanggil saja dan kita perlu mengimport library PIL dan memilih sub-library yang ingin kita gunakan. Mari kita simak contoh berikut :

```
from PIL import Image

# Membuka gambar dari drive mount
img = Image.open('/content/drive/MyDrive/Fungsional/kucing_huh.jpg')

plt.imshow(img)
plt.axis('off')
plt.show()
```

```
from PIL import Image

# dengan cara upload file
uploaded = files.upload()

# Ambil nama file pertama yang di-upload
```



```
filename = list(uploaded.keys())[0]

img = Image.open(filename)

img.show()
```

Perlu dicatat bahwa saat menggunakan PILLOW pada google collab, kita perlu menggunakan matplotlib sebagai 'canvas' (GUI) untuk menampilkan gambar. Hal ini dikarenakan function show() dalam PILLOW tidak bisa tereksekusi tanpa GUI (biasanya mengikuti environments system). Jika ingin menggunakan pure library PILLOW kita bisa menggunakan text editor seperti vscode. Berikut contoh kode untuk digunakan pada vscode dalam mengakses dan menampilkan file gambar/image:

```
from PIL import Image

img = Image.open(r'C:\Users\Aku\Downloads\kucing.jpg')

img.show()
```

Output:



4.2. RESIZE: IMPERATIF VS FUNGSIONAL

Dalam paradigma Pemrograman Fungsional, terdapat konsep Immutability (tidak dapat berubah), di mana kita sebisa mungkin menghindari perubahan data pada objek aslinya (side-effect). Di dalam library Pillow, terdapat dua metode umum untuk mengubah ukuran gambar, yaitu thumbnail() dan resize(). Perbedaannya sangat mendasar dalam konteks paradigma pemrograman:

1. thumbnail() (Gaya Imperatif): Fungsi ini memodifikasi objek gambar asli secara langsung (in-place).



2. `resize()` (Gaya Fungsional): Fungsi ini tidak mengubah gambar asli, melainkan mengembalikan objek gambar baru yang sudah diubah ukurannya.

Mari kita cek perbedaannya secara langsung dengan cara mencobanya pada sel-sel kode berikut ini:

```
print(f"Ukuran Asli: {img.size}")
Output :
Ukuran Asli: (345, 396)
```

```
# --- PENDEKATAN 1: IMPERATIF (thumbnail) ---
# Kita harus meng-copy gambar dulu agar gambar asli tidak rusak untuk
# demo berikutnya
img_copy = img.copy()
img_thum = img_copy.thumbnail((100, 100)) # Fungsi ini tidak
# mengembalikan nilai (None), tapi merubah img_copy
print(f"Ukuran setelah thumbnail (Imperatif): {img_copy.size}")
print(img_thum)

Output:
Ukuran setelah thumbnail (Imperatif): (87, 100)
None
```

```
# --- PENDEKATAN 2: FUNGSIONAL (resize) ---
# Fungsi ini mengembalikan objek baru. Objek 'img' asli TIDAK
# BERUBAH.
new_img = img.resize((100, 100))

print(f"Ukuran gambar baru (resize Fungsional): {new_img.size}")
print(f"Ukuran gambar asli (tetap terjaga): {img.size}")

Output:
Ukuran gambar baru (resize Fungsional): (100, 100)
Ukuran gambar asli (tetap terjaga): (345, 396)
```

Dalam Pemrograman Fungsional, kita lebih menyukai pendekatan ke-2 (`resize`) karena data asli aman dan alur data lebih mudah dilacak.



Sebelum Kalian mengerjakan Latihan 1, silahkan kalian run kode ini:

```
img_lat = Image.fromarray(imgLat)
```

Kenapa harus dikonversi? Karena fungsi im.imread pada library Matplotlib digunakan untuk membaca dan menyimpan gambar sebagai NumPy Array. Sedangkan untuk bab 4 ini kita sudah melakukan manipulasi gambar menggunakan library Pillow dengan menggunakan objek gambar secara langsung (tanpa Array).

📌 Latihan 1 - Resize Gambar

Pada gambar yang sudah kalian upload pada latihan 0, buatlah versi gambar yang diperkecil menjadi ukuran 200x200 piksel. Simpan hasilnya ke variabel 'lat1_img'.

```
lat1_img = ...

# Tampilkan gambar menggunakan fungsi yang sudah disediakan
...

# --- SIMPAN HASIL (Jangan Diubah) ---
galeri_latihan.append(("Latihan 1: Resize", lat1_img))
print("Latihan 1 tersimpan!")
```

4.3. PIXEL MANIPULATION USING HIGH-ORDER FUNCTIONS

Kita sudah belajar tentang salah satu implementasi dari First-Class Function pada Paradigma Fungsional adalah penggunaan High-Order Functions, yaitu fungsi yang menerima fungsi lain sebagai argumennya. Pillow menyediakan method point() yang sangat *powerful*.

Method ini memungkinkan kita memanipulasi setiap piksel gambar dengan melemparkan sebuah fungsi (biasanya menggunakan *lambda*). Ini mirip dengan fungsi map() pada list, namun diterapkan pada data piksel gambar.

Contoh di bawah ini menunjukkan cara mengatur kecerahan (brightness) dan mengubah gambar menjadi biner (hitam-putih mutlak) menggunakan method point() dan ekspresi lambda:

```
# Gambar 1: Mencerahkan Gambar (Brightness adjustment)
# Setiap nilai piksel dikalikan 1.5 (jika > 255 otomatis di-clamp oleh Pillow)
bright_img = img.point(lambda p: p * 1.5)

# Gambar 2: Thresholding (Membuat gambar Biner Hitam-Putih)
# Jika nilai piksel > 128, ubah jadi 255 (putih), jika tidak jadi 0 (hitam)
```

```
# Ini sering digunakan dalam segmentasi citra
binary_img = img.convert("L").point(lambda p: 255 if p > 128 else 0)
```

```
# Menampilkan hasil menggunakan fungsi yang sudah kita deklarasikan
# sebelumnya (pada Latihan 0)
penampil(
    [img, bright_img, binary_img],
    ["Original", "Brightened (Lambda * 1.5)", "Binary Threshold
(Lambda If-Else)"],
    [None, None, 'gray']
)
```

Output:



📍 Latihan 2 - Resize Gambar

Pada gambar yang sudah kalian upload pada latihan 0, gunakan teknik Lambda/HOF untuk meningkatkan kecerahan gambar sebesar 2x lipat. Simpan hasilnya ke variabel 'lat2_img'.

```
lat2_img = ...

#Tampilkan gambar menggunakan fungsi yang sudah disediakan
...

# --- SIMPAN HASIL (Jangan Diubah) ---
galeri_latihan.append(("Latihan 2: Brightness", lat2_img))
print("Latihan 2 tersimpan!")
```

4.4. PILLOW FEATURES UNTUK MANIPULASI GAMBAR

PILLOW memiliki fitur yang lebih lengkap untuk image processing dibanding dengan matplotlib karena matplotlib lebih diunggulkan dengan visualisasi data saja. Berikut beberapa contoh fitur yang ada pada PILLOW :

4.4.1. CROPPING IMAGES

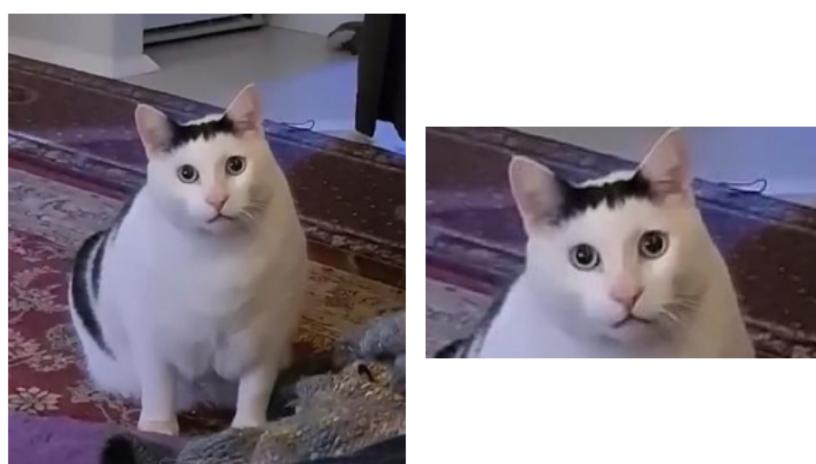
Cropping dengan code sedikit berbeda ketika kita menggunakan fitur crop pada smartphone, crop disini menggunakan parameter yang menangkap area mana saja yang harus di crop.

```
crop_area = (60, 60, 300, 200)
img_cropped = img.crop(crop_area)

#simpan hasil proses
img_cropped.save('croppedimage.jpg')

#tampil foto
#tampil_foto(img_cropped)
penampil(
    [img, img_cropped]
)
```

Output:



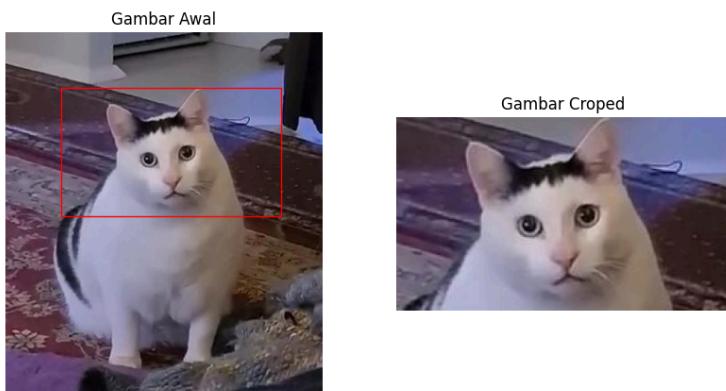
(60, 60, 300 , 200) berarti posisi crop akan membentuk persegi dengan rumus (left, upper, right, lower) dari tepian gambar asli dalam satuan pixels. maksudnya adalah kode akan memotong dari titik (60,60) hingga (300,200).



📌 Latihan 3 - Box Rezise

Pada gambar yang sudah kalian upload pada latihan 0, lakukan cropping pada posisi yang kalian inginkan. Lalu tampilkan hasilnya seperti contoh berikut (tambahkan sebuah kotak/rectangle untuk menandai posisi yang dicrop)! Simpan hasil crop kedalam variabel 'lat3_img'.

Contoh Output:



```
lat3_img = ....  
  
#Tampilkan gambar  
...  
  
# --- SIMPAN HASIL (Jangan Diubah) ---  
galeri_latihan.append(("Latihan 3: Box Resize", lat3_img))  
print("Latihan 3 tersimpan!")
```

4.4.2. ROTATING IMAGES

Rotating Images ada beberapa cara seperti code di bawah ini, karena biasanya saat image dirotasi bentuk images bisa berubah menyesuaikan sumbu gambar diputar, namun kita bisa memilih apakah kita ingin melakukan perubahan bentuk atau tidak.

```
img_rotated = img.rotate(30)  
  
tampil_foto(img_rotated)  
  
img_rotated.save('example_rotated.jpg')
```



Output:



```
Function sejenis
img_rotated = img.rotate(45, center=(100, 100)) -> Rotate
berdasarkan titik

img_rotated = img.rotate(45, expand=True) -> Rotate tapi
mempertahankan ukuran asli

img_rotated_90 = img.transpose(Image.ROTATE_90) -> 90 derajat
arah jarum jam
```

📍 Latihan 4 - Rotasi Gambar

Pada gambar yang sudah kalian upload pada latihan 0, putar gambar asli sebesar 45 derajat. Simpan hasilnya ke variabel 'lat4_img'.

```
lat4_img = ...

#Tampilkan gambar menggunakan fungsi yang sudah disediakan
...

# --- SIMPAN HASIL (jangan dihapus)---
galeri_latihan.append(("Latihan 4: Rotasi", lat4_img))
print("Latihan 4 tersimpan!")
```

4.4.3. FLIPPING IMAGES

Flipping image disini sama dengan membalik arah gambar dengan menggunakan fungsi transpose.

```
img_flipped = img.transpose(Image.FLIP_LEFT_RIGHT)

penampil(
    [img, img_flipped],
```



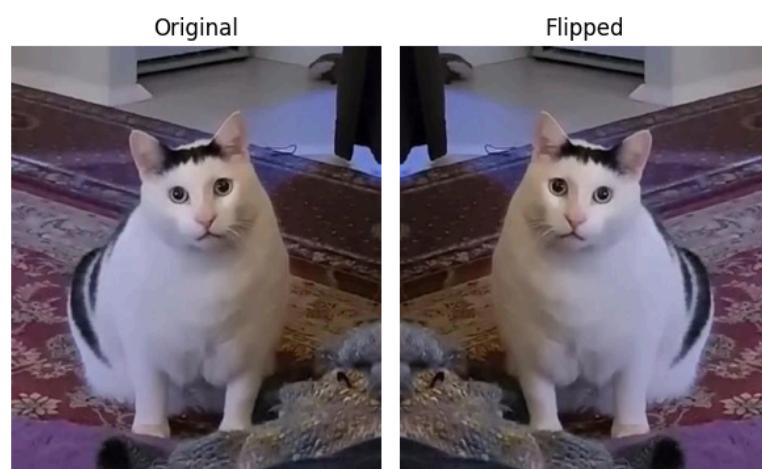
```

        ["Original", "Flipped"],
    )

img_flipped.save('example_flipped.jpg')

```

Output:



4.4.4. IMAGES ENCHANCER

Image enhancer dalam pillow memiliki beberapa fitur seperti mengatur brightness, colour, contrast, sharpness. Untuk contoh lebih lanjut bisa kamu pelajari di [ImageEnhance](#).

```

from PIL import ImageEnhance as enhanc

# Meningkatkan kecerahan
enhancer_brightness = enhanc.Brightness(img)
img_brightness = enhancer_brightness.enhance(2.0) # kelipatan
brightness 2 kali lipat

# Meningkatkan kontras
enhancer_contrast = enhanc.Contrast(img_brightness)
img_contrast = enhancer_contrast.enhance(2.0) # kelipatan
kontras 2 kali lipat

tampil_foto(img_contrast)

img_contrast.save('example_enhanced.jpg')

```



Output:



4.4.5. FILTERING IMAGES

Dalam PILLOW terdapat banyak function untuk melakukan [filtering](#), seperti yang tercakup di bawah ini :

- BLUR
- CONTOUR
- DETAIL
- EDGE_ENHANCE
- EDGE_ENHANCE_MORE
- EMBOSSED
- FIND_EDGES
- SHARPEN
- SMOOTH
- SMOOTH_MORE

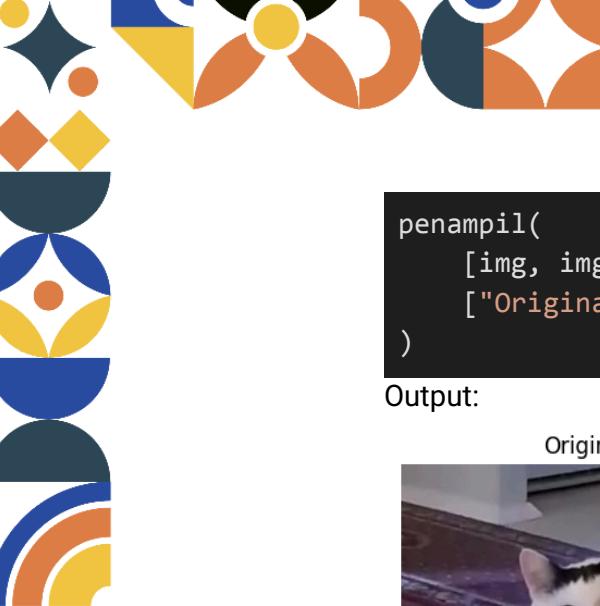
```
from PIL import ImageFilter

# Terapkan efek filter
img_blur = img.filter(ImageFilter.BLUR)
img_contour = img.filter(ImageFilter.CONTOUR)

# Menyimpan gambar hasil filter
img_blur.save('example_blur.jpg')
img_contour.save('/example_contour.jpg')

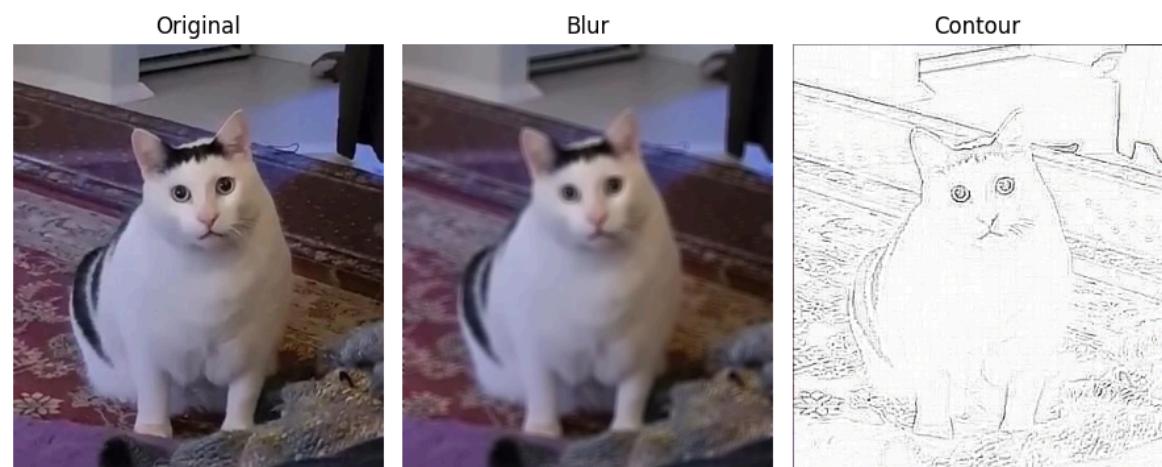
# Tampilkan gambar before and after (kiri: asli, kanan: filter)
```





```
penampil(  
    [img, img.blur, img.contour],  
    ["Original", "Blur", "Contour"],  
)
```

Output:



4.5. MERGING IMAGES/OVERLAY IMAGES

Merging atau overlay sama halnya dengan yang kita lakukan untuk mengedit foto, kita perlu 2 gambar awal untuk digabungkan menjadi sebuah overlay.

```
img2 = Image.open(r'umm_new.png')  
  
img2 = img2.resize(img.size)  
  
if img.mode != img2.mode:  
    img2 = img2.convert(img.mode)  
  
merged_image = Image.blend(img, img2, alpha=0.5)  
#merged_image.save('example_merged.jpg')  
  
tampil_foto(merged_image)
```



Output:



Seperti halnya library pada umumnya banyak fitur yang mungkin tidak akan lengkap dijelaskan disini seperti palette, path, sequence dan masih banyak lagi kalian bisa mengexplore sendiri pada tautan berikut.

```
from PIL import Image, ImageEnhance, ImageFilter

img = Image.open(r'/content/drive/MyDrive/Fungsional/cukurukuk.jpg')
enhancer = ImageEnhance.Brightness(img)
img_enhanced = enhancer.enhance(1.5)
img_blurred = img_enhanced.filter(ImageFilter.GaussianBlur(radius=3))
img_gray = img_blurred.convert('L')
img_gray.save('kucing.jpg')

tampil_foto(img_gray, cmap='gray')
```



Output:



Penjelasan :

1. Kita menggunakan variabel enhancer untuk memastikan bahwa ImageEnhance berfokus pada adjustment Brightness.
2. Selanjutnya melakukan peningkatan 1.5x lipat brightness dari semula
3. Terakhir kita menambahkan gaussian filter untuk gambar.



CODELAB

CODELAB 1

Lengkapi kode berikut agar kalian dapat menampilkan gambar yang sudah kalian proses pada latihan 1 sampai 4 di atas dalam satu grid layout.

```
import matplotlib.pyplot as plt

# 1. Cek jumlah Latihan yang sudah tersimpan
# TODO: Hitung panjang list 'galeri_latihan'
jumlah_latihan = len(__)

print(f"Menampilkan {jumlah_latihan} gambar dari galeri.")

# Siapkan Canvas
fig, axes = plt.subplots(1, jumlah_latihan, figsize=(5 * jumlah_latihan,
5))

# 3. Loop untuk menampilkan setiap gambar
# TODO: Gunakan enumerate untuk mendapatkan index (i) dan datanya
# Ingat: Setiap item di galeri berisi pasangan (judul, gambar)
for i, (judul_gambar, objek_gambar) in enumerate(__):

    # Menangani akses ke axes (jika cuma 1 gambar, axes bukan list)
    if jumlah_latihan > 1:
        ax = axes[i]
    else:
        ax = axes

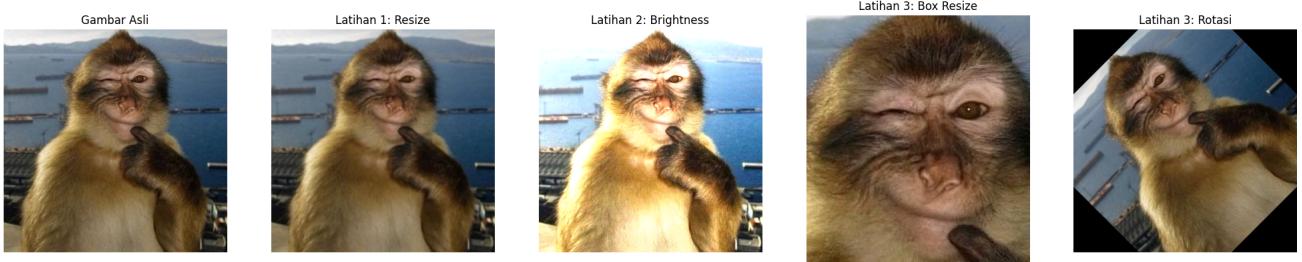
    # A. Set Judul Gambar
    # Ambil dari variabel judul yang sudah di-unpack di loop
    ax.set_title(__)

    # B. Matikan Axis
    ax.axis('__')

# Menampilkan hasil akhir
plt.tight_layout()
plt.show()
```



Contoh Output:



CODELAB 2

Program dibawah adalah sebuah Pipeline pemrosesan gambar yang terdiri dari tiga langkah berurutan: Grayscale, Rotate, dan Resize. Akan tetapi kode tersebut masih belum lengkap. Tugas kalian adalah melengkapi kode tersebut dan perlihatkan kepada asisten jika berhasil sesuai dengan output yang diberikan!

```
from functools import reduce
from PIL import Image

# Pastikan variabel 'img' sudah ter-load sebelumnya, bebas langsung upload atau mount dari drive
img = Image.open('____')

# Buatlah fungsi-fungsi kecil yang tidak mengubah data asli.

def to_grayscale(image):
    # TODO: Gunakan convert dengan parameter "L" untuk grayscale
    return image.___("L")

def rotate_90(image):
    # TODO: Gunakan rotate untuk memutar 90 derajat
    return image.___(90)

def resize_half(image):
    # TODO: Ambil ukuran gambar saat ini
    w, h = image.___
    # TODO: Return gambar yang sudah di-resize menjadi setengah ukuran
    return image.resize((____, ____))

# Susun fungsi-fungsi di atas ke dalam sebuah list.
```

```

transformation_pipeline = [
    __,           # 1. Fungsi Grayscale
    __,           # 2. Fungsi Rotate
    __,           # 3. Fungsi Resize
]

# Gunakan reduce untuk mengaplikasikan fungsi secara berantai.
def apply_transformation(image, func):
    # TODO: Terapkan fungsi 'func' ke 'image'
    return func(__)

# Jalankan reduce
# Hint: Argumen ke-3 adalah gambar awal (img)
processed_image = reduce(__, transformation_pipeline, __)

# Tampilkan Hasil (Jangan diubah)
import matplotlib.pyplot as plt

penampil(
    [image, processed_image],
    ["Original", "Hasil"],
    [None, 'gray']
)

```

Contoh Output:



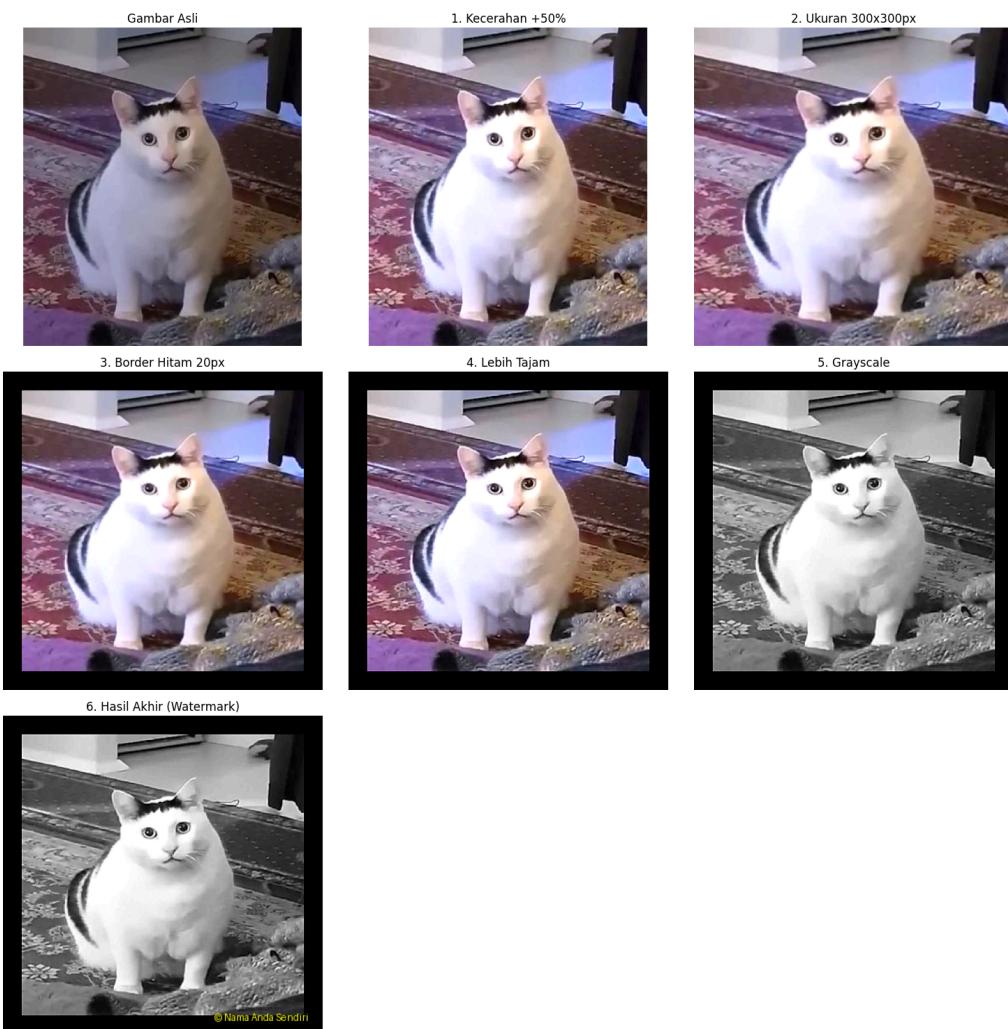
TUGAS

TUGAS 1

Carilah Gambar sesuai dengan tema kalian, lalu berikan:

1. Kecerahan gambar sebanyak 50%.
2. Ukuran 300x300 pixel agar konsisten
3. berikan border hitam dengan lebar 20 pixel agar lebih rapi
4. fotonya lebih tajam dari sebelumnya.
5. ubah warna gambar menjadi grayscale
6. Tambahkan watermark/annotate nama kalian di ujung kanan gambar. Gambar yang sudah diedit disimpan dalam perangkat pribadi
7. Tampilkan perbandingan antara gambar asli dan gambar hasil manipulasi (kreasikan bagian ini sekreatif mungkin dengan syarat masih menerapkan konsep paradigma fungsional, bisa berupa high order function, closure, atau decorator).

Contoh Output:



TUGAS 2

Ubahlah gambar before menjadi gambar after menggunakan image processing yang sudah kalian pelajari!

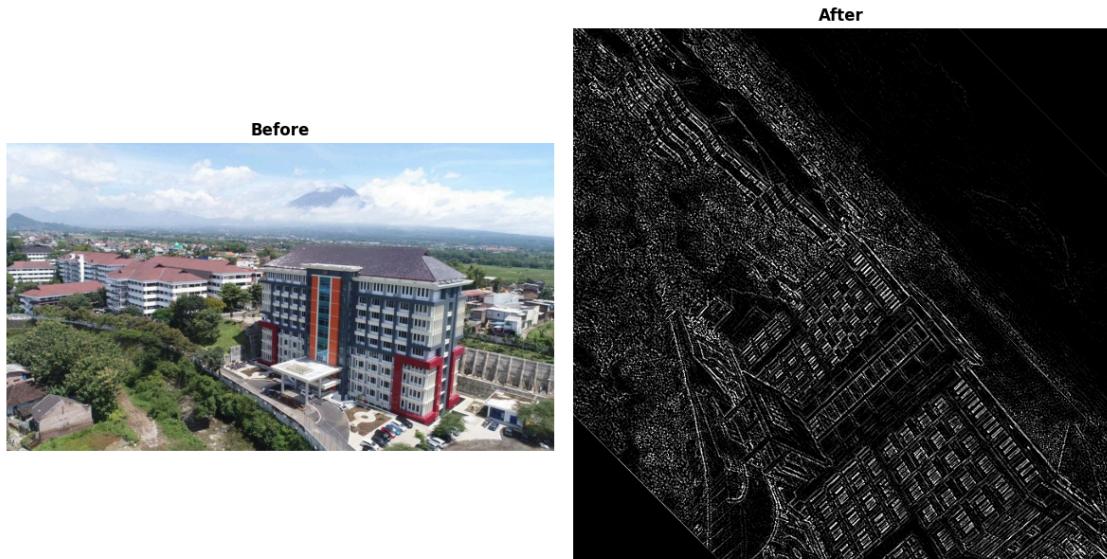
NIM GENAP



Link Image Asli:

https://drive.google.com/file/d/1wSnrFGvE9oPDEIKBu4TwxKckUKYMZbUp/view?usp=drive_link

NIM GANJIL



Link Image Asli:

https://drive.google.com/file/d/1Pm0TNyZYVJYSz7Y3cqhdHvGhT0fMTT7R/view?usp=drive_link



RUBIK PENILAIAN

Komponen Penilaian	Bobot (%)
Codelab	15
Tugas 1	
Kode / Kelengkapan Fitur	20
Kreativitas	10
Pemahaman	10
Tugas 2	
Kode / Kelengkapan Fitur	20
Kreativitas	10
Pemahaman	15
Total Akhir	100