1. **Github链接**

**二、任务描述（blog05-image01.py 的核心操作）**

blog05-image01.py 的核心任务是**对 “lena-hd.png” 图像执行两种不同的加法运算**，并展示原图与运算结果的差异，具体在原图形上的修改逻辑如下：

1. 读取原始图像 “lena-hd.png”，将其赋值给img和test（即test与原图完全一致，无任何修改）；
2. 分别通过**Numpy 原生加法**（img + test）和**OpenCV 加法函数**（cv2.add(img, test)）对原图进行 “原图 + 原图” 的叠加运算；
3. 用cv2.imshow()创建三个独立窗口，分别显示原图、Numpy 加法结果、OpenCV 加法结果。

**三、数据描述（处理的图像与中间数据）**

本次处理的核心数据是**经典测试图像 “lena-hd.png”**，相关数据特征如下：

* **原始图像数据**：Lena 图是图像处理领域常用的标准测试图，格式为BGR（OpenCV 默认读取格式），像素值类型为np.uint8（范围 0-255），包含 RGB 三个颜色通道的空间像素矩阵；
* **中间数据**：
  + blog05-image01.py 中，test是原图的直接副本（像素值、尺寸、通道数与原图完全相同）；
  + 两种加法结果数据：均为与原图尺寸、通道数一致的np.uint8类型矩阵，差异仅源于加法运算逻辑；
* **数据本质**：所有图像数据均为 “像素值矩阵”，运算过程本质是对矩阵中每个像素点的数值进行加法计算。

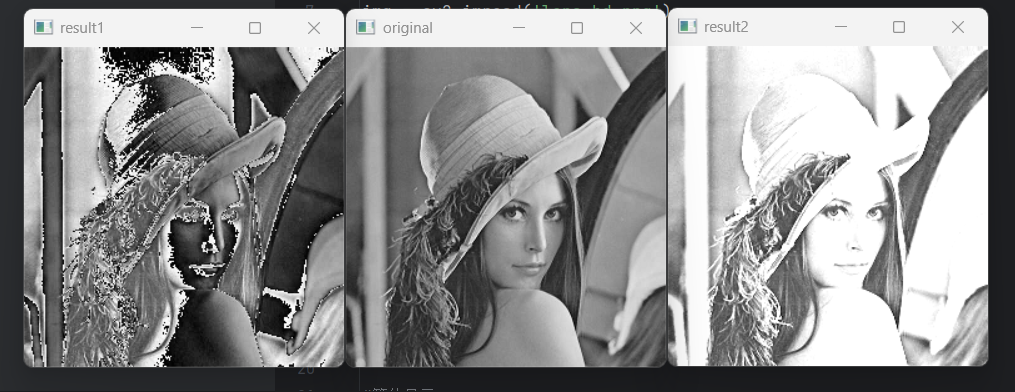
**四**、**代码修改描述（blog.change.py 对比 blog05-image01.py 的图形处理修改）**

blog.change.py 在**图形处理逻辑、显示方式、鲁棒性**三方面对 blog05-image01.py 进行了关键修改，具体如下：

| **修改维度** | **blog05-image01.py（原代码）** | **blog.change.py（修改后代码）** |
| --- | --- | --- |
| 1. 测试图（test）生成逻辑 | test = img：test 与原图完全一致，无亮度调整 | test = cv2.multiply(img, 0.8).astype(np.uint8)：通过cv2.multiply将原图亮度降低 20%（像素值 ×0.8），避免 “原图 + 原图” 叠加导致的过度饱和 |
| 2. Numpy 加法处理 | 直接计算result1 = img + test：未显式类型转换（虽结果仍为 uint8，但逻辑不明确） | 显式添加astype(np.uint8)：result\_numpy = (img + test).astype(np.uint8)，明确像素值类型约束，避免潜在类型溢出风险 |
| 3. 图像显示方式 | 用cv2.imshow()创建 3 个独立窗口：未处理 BGR→RGB 转换，导致 matplotlib 显示时颜色偏蓝 | 用matplotlib.pyplot创建 1 行 3 列子图：  ① 强制cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)转换，解决 OpenCV（BGR）与 matplotlib（RGB）的颜色通道差异，显示正常颜色；  ② 用plt.subplot(131/132/133)将原图、两个结果图合并到一个画布，便于对比 |
| 4. 显示标注与可读性 | 窗口标题为 “original”“result1”“result2”：未说明加法逻辑差异 | 子图标题为 “原图”“Numpy 加法（模运算）”“OpenCV 加法（饱和运算）”：直接标注两种加法的本质，明确结果差异原因 |
| 5. 鲁棒性与环境配置 | 无异常处理：若图片路径错误，代码直接崩溃；无中文字体配置 | ① 增加if img is None: raise FileNotFoundError：图片读取失败时抛出明确异常，便于排查路径问题；  ② 配置中文字体（plt.rcParams["font.family"]）：解决标题中文乱码；  ③ 切换 matplotlib 后端（plt.switch\_backend('TkAgg')）：避免 “tostring\_rgb 属性错误” |

**五、结果对比**

**blog05-image01：**

****

**blog.change：**

****