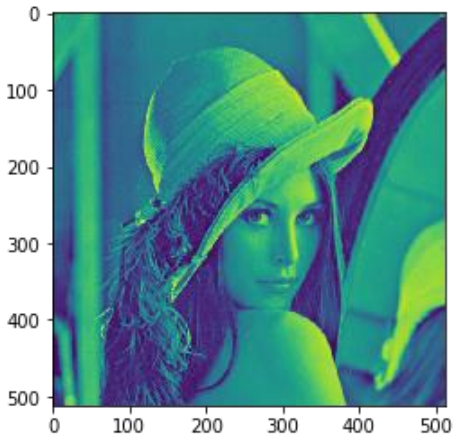
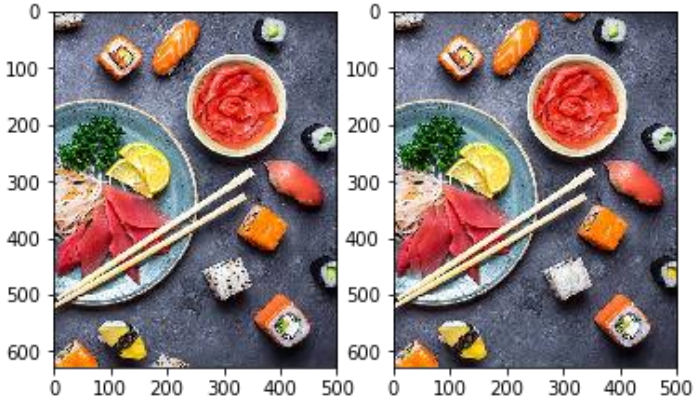
Інструменти мови Python для роботи з зображеннями

4) ***1,5 бала*** Завантажте та виведіть на екран чорно-біле зображення lena.png. Без додаткових налаштувань кольору воно може бути таким



Тому потрібно додатково передавати в imshow() параметр cmap, значення якого буде 'gray' або plt.cm.gray. Додайте до звіту код та скриншот візуалізації з порівнянням коректно та некоректно відображеної «Лени».

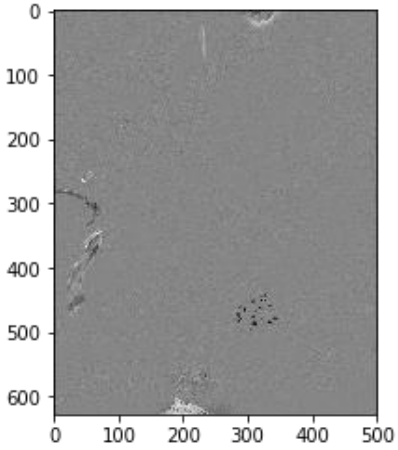
Далі займемось порівнянням кольорових зображень. Для початку спробуйте знайти відмінності самостійно.



Першим кроком для знаходження відмінностей стає усунення альфа-каналу із зображень (залишаємо тільки RGB значення в масиві).

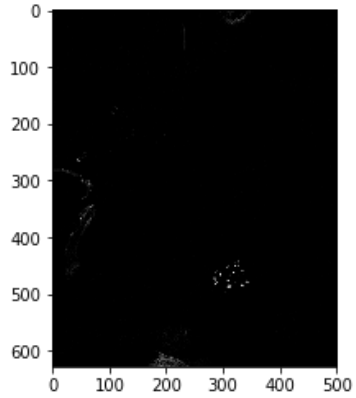
Далі швидко знебарвіть зображення, виконуючи усереднення по RGB каналах за допомогою функції numpy.mean(). Зверніть увагу, що зображення є двовимірним об’єктом, тому також потрібно задати параметр axis.

Отримавши чорно-білі зображення, знайдіть їх різницю та виведіть на екран її зображення:

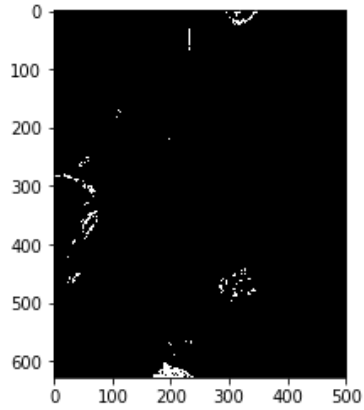


Додайте відповідний код та результати до звіту.

Уже з’явились перші контури об’єктів, що відрізняються на вище згаданих зображеннях. Проте варто зауважити, що різниця між пікселями може призвести до появи від’ємних значень. Виведіть мінімальне та максимальне значення пікселя для даного зображення та помітите це. Для відображення всіх значень у проміжок [0; 1] піднесемо їх усі до квадрату. Тоді отримаємо такий результат:

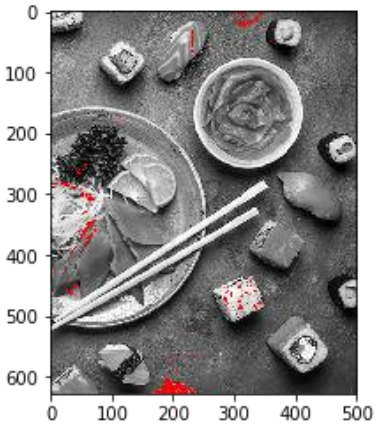


Яскравіше виділити області зображень, які відрізняються, можна за рахунок додавання порогового значення, яке відкине всі близькі до 0 значення пікселів. Наприклад, візьмемо поріг 0.05 (5%). Тоді пікселі, значення яких відрізняються для двох зображень більш суттєво, стануть білими (True), а менше порогу – стануть чорними (False):



Наступний крок – позначити відмінності власне на зображеннях. Для цього потрібно нанести додатковий прошарок на початкове зображення, зробивши його, по суті, тривимірним. Цього можна добитись, наприклад, застосувавши функцію numpy.reshape() (розміри по x та y залишаються, проте з'явиться третій вимір - np.reshape(зображення, (x-розмір, y-розмір, 1)). Далі потрібно буде накласти отриману на останньому зображенні маску на кожний канал зображення. В цьому допоможе функція numpy.tile():

різницяRGB = np.tile(3d-зображення, (1, 1, 3))



Побудуйте отриманий результат разом із початковими зображеннями суші. Врешті решт, має з'явитись подібна візуалізація:

Якщо розміри візуалізації Вас не задовольняють, можете задати їх при побудові підграфіків, наприклад, plt.subplots(1, 3, figsize=(20, 7)). *Додайте у звіт напрацьований код та скриншоти*.

5) ***0,5 бала*** (Проста анімація) Розгляньте анімацію з імітацією дощу. *Детально опишіть код та доповніть звіт посиланнями на інформацію щодо основних функцій, використаних для створення анімації. Зробіть скриншот запуску та збережіть його в звіті*.