執行環境:
Windows 10、Anaconda、Python 3.10

環境安裝方式:
$ conda create --name dip **python=3.10**
$ pip install opencv-python

- **For requirements 1 & 2, you need to show**
  - **which function you use or implement**
    匯入 cv2 (opencv 套件) 之後，呼叫 cv2.resize() 函式的同時，在其中的 interpolation 引數當中，使用「cv2.INTER_LINEAR」以及「cv2.INTER_CUBIC」，來進行 resize。
  - **how does your program work**
    在主程式當中，先定義幾個縮放比例 [0.2, 5, 32]，再迭代將各個比例的值整合自訂的 scale_img() 函式，在自訂函式裡頭呼叫 cv2.resize() 後，再回傳圖片物件，最後在主程式中，透過 cv2.imwrite() 儲存圖片到指定路徑。
  - **how to use your program**
    $ conda create --name dip python=3.10
    (安裝過程…)
    $ conda activate dip
    $ pip install opencv-python
    $ cd d12944007
    $ python code/run.py
- **For requirements 3 & 4, you need to provide**
  - **Resulted images for comparison**
    在自拍照縮放後進行觀察，0.2x 的 bilinear 會比較平滑，bicubic 會比較銳利；5x 的 bilinear 與 bicubic 似乎看不太出來差異；32x 的 bilinear 會比較模糊，bicubic 依然銳利。
  - **Explanation**
    - bilinear
      - 由 2*2 共 4 個附近的 pixel 座標來進行內插，每個格子，上方兩點做一次內插、下方兩點做一次內插，得到兩點再做一次內插；縮小的時候效果應該較好，放大的時候可能會有模糊的現象，運算花費時間比 bicubic 少。
      - 每一個 pixel 的時間複雜度為 O(1)，假設有 N 個 pixel，那時間複雜度為 O(N)。
    - bicubic

- 由 4*4 共 16 個附近的 pixel 座標來進行內插，每個格子，上方兩點做三次內插、下方兩點做三次內插，得到兩點再做三次內插；無論放大或縮小，都會有銳利化的效果，運算花費時間比 bilinear 還要多。
- 每一個 pixel 的時間複雜度為 O(1)，假設有 N 個 pixel，那時間複雜度為 O(N)。