stanCode

標準程式教育機構

Assignment 3

This assignment is based on the Assignment 3 of CS106AP at Stanford University

在這份作業的前半部內容，我們將帶同學更熟悉我們在 Python 習得的第一個資料結構「List」！後半的部分，我們則是會挑戰一個軟體工程師常常會遇到的工作 - 「接續他人撰寫的程式碼」，並將整個程式給完成！

請在開始之前，觀看作業三說明影片：

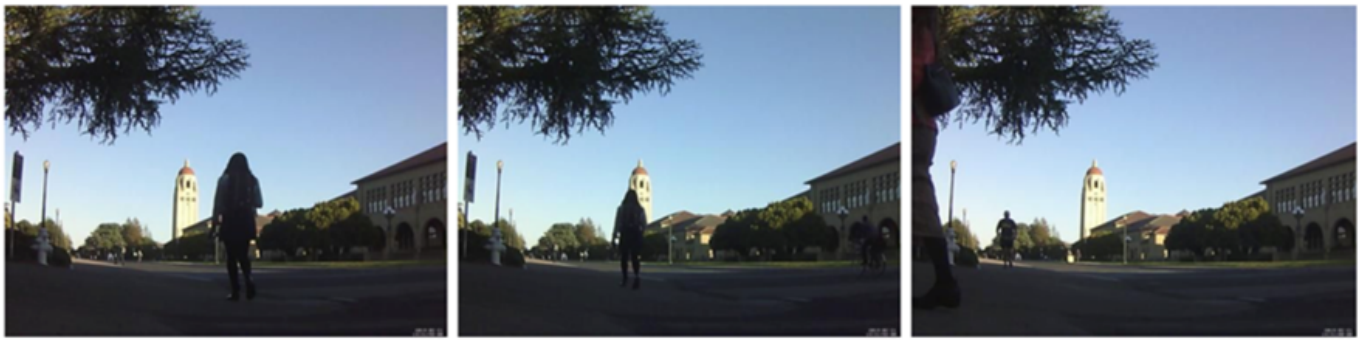
https://youtu.be/BXZ7XeN9Eys

點此下載作業檔案

如果過程卡關歡迎各位向助教詢問！也非常鼓勵同學們互相討論作業之概念，但請勿直接把 **code** 分享給同學看，這很可能會剝奪他獨立思考的機會，並讓他的程式碼與你的極度相似，使防抄襲軟體認定有抄襲嫌疑。

stanCodoshop.py

stanCode 打算推出一個全新 APP -「stanCodoshop」，讓使用者可以將美麗風景照中亂入的路人移除！（觀光景點拍照人擠人的話，真的很困擾啊><）



若使用者匯入如上方所示的三張圖片 -「位置相同，但有隨機路人出沒的史丹佛大學風景照 」到我們神奇的 「stanCodoshop」APP，將得到乾淨的風景照（如下圖）

Algorithm

假設使用者輸入了好幾張同樣大小的照片，而且照片中隨機路人出現的位置都不太一樣。這時我們將所有照片中，在「相同位置」上的 pixels 取出來一一比較，選擇一顆「沒有被路人干擾」的 pixel，並把它放到一張「與照片相同大小的空白圖片」的「相同位置」上。當我們的空白圖片的每個位置都被我們選擇的 pixel 填滿時，我們要的乾淨風景照就產生了！

然而，我們要如何從 N 張照片中相同位置 (x, y) 的 pixels 中，挑選出那顆「沒有被路人干擾」的 pixel 呢？

假設使用者輸入了四張照片（N = 4），這些照片在某相同位置 (x, y) 上的 (r, g, b) 數值為：

pixel_pic1：(1, 1, 2)　　pixel_pic2：(1, 1, 1)

pixel_pic3：(1, 2, 2)　　pixel_pic4：(28, 27, 29)

從他們的數值可以發現，pixel_pic4 的 RGB 數值看起來明顯地與其他三顆 pixels 不在同個範圍。而這也代表了，在第四張照片的 (x, y) 位置上，很有可能有路人經過（或狗狗）！

Color Distance

雖然從上面的例子中，我們可以很輕易地「感覺」出那顆奇怪的 pixel ，但有沒有一條比較一致的通則，可以用來選出我們要的 pixel 呢？

這邊，我們要借助一個叫做「color distance」的演算法來「公平地比較」這些數值。試想：若我們將一顆 pixel 的 RGB 數值填入我們熟知的 (x, y, z) 直角坐標系中，就會在三維空間中產生一個點。同理，我們也可以將剩下的三顆 pixels 也都填進去，所以目前總共是四個點（如下頁 圖1.）。而 點與點間的「直線距離」，就是我們所謂的「color distance」。

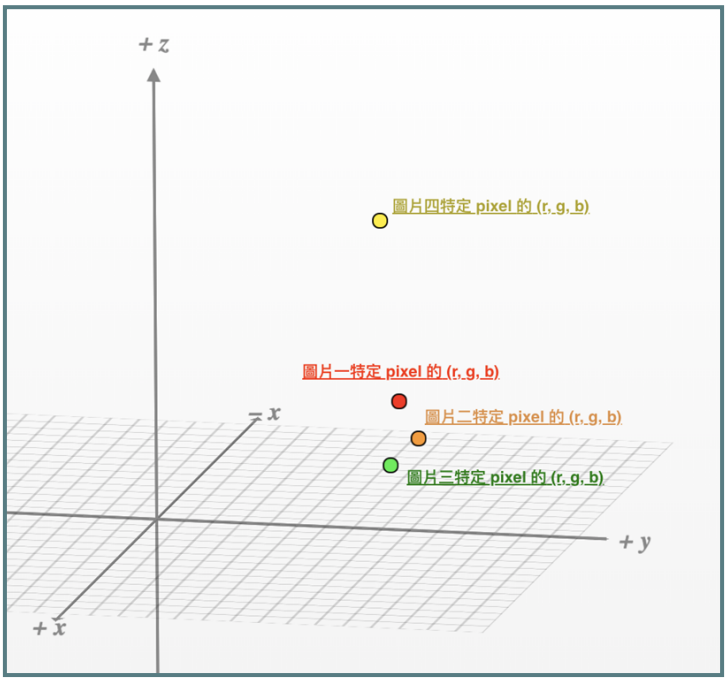


圖1. 某特定位置的不同 pixels 之 (r, g, b) 座標點在三維空間之示意圖

假設我們對所有 N 個 pixels 取一個 平均的 RGB 值，並得到：red_avg、green_avg 和 blue_avg，這在空間中也會佔據一個點，簡稱「avg_point」。
我們可以用下列公式計算某顆 pixel 與這個 avg_point 的 color distance：

$$color\_distance = \sqrt{(red\_avg - pixel.red)^2 + (green\_avg - pixel.green)^2 + (blue\_avg - pixel.blue)^2}$$

Milestone 1 - get_pixel_dist(pixel, red, green, blue)

說明

請完成 stanCodoshop.py 檔案裡名為 get_pixel_dist(pixel, red, green, blue) 之 function，並 return「pixel 與 avg_point 間的 color distance」。red、green 和 blue 分別為 avg_point 的 RGB 數值。

測試

請在「def solve(images): 」裡，加入下圖中紅色框框裡的三行程式碼，來檢查這個 function 的功能是否正確：

```
def solve(images):
    """
    Given a list of image objects, compute and display a Ghost solution image
    based on these images. There will be at least 3 images and they will all
    be the same size.

    Input:
        images (List[SimpleImage]): list of images to be processed
    """
    width = images[0].width
    height = images[0].height
    result = SimpleImage.blank(width, height)
    ######## YOUR CODE STARTS HERE #########
    # Write code to populate image and create the 'ghost' effect
    green_im = SimpleImage.blank(20, 20, 'green')
    green_pixel = green_im.get_pixel(0, 0)
    print(get_pixel_dist(green_pixel, 5, 255, 10))
    ######## YOUR CODE ENDS HERE ###########
    print("Displaying image!")
    result.show()
```

點擊並打開 PyCharm 下方的「Terminal」視窗：

macOS 請輸入： python3 stanCodoshop.py hoover

Windows 請輸入： py stanCodoshop.py hoover

如果看到下方字樣就代表 Milestone 1 完成！

```
Loading hoover/200-500.jpg
Loading hoover/158-500.jpg
Loading hoover/156-500.jpg
11.180339887498949
Displaying image!
```

Milestone 2 - get_average(pixels)

說明

請完成 stanCodoshop.py 檔案裡名為 get_average(pixels) 的 function，
並 return 一個包含 平均 RGB 值 的 Python list：[red, green, blue]。

括號內的 pixels（這個傳進來的 parameter）是一個 Python List，裡面裝著所有 N 張圖片在某相同位置 (x, y) 上的 pixels（共 N 顆）。

請從 pixels 中，取出所有 pixels 的 RGB 數值，並算出它們的 RGB 平均值，存在 red(紅色平均值)、green(綠色平均值) 及 blue(藍色平均值) 中。然後將三個數值存進一個 Python List，再 return 出去。

測試

請在「def solve(images): 」裡，將程式碼改成如下圖中紅色框框裡的四行程式碼，來檢查這個 function 的功能是否正確：

```
def solve(images):
    """
    Given a list of image objects, compute and display a Ghost solution image
    based on these images. There will be at least 3 images and they will all
    be the same size.

    Input:
        images (List[SimpleImage]): list of images to be processed
    """
    width = images[0].width
    height = images[0].height
    result = SimpleImage.blank(width, height)
    ######## YOUR CODE STARTS HERE #########
    # Write code to populate image and create the 'ghost' effect
    green_pixel = SimpleImage.blank(20, 20, 'green').get_pixel(0, 0)
    red_pixel = SimpleImage.blank(20, 20, 'red').get_pixel(0, 0)
    blue_pixel = SimpleImage.blank(20, 20, 'blue').get_pixel(0, 0)
    print(get_average([green_pixel, green_pixel, green_pixel, blue_pixel]))
    ######## YOUR CODE ENDS HERE ###########
    print("Displaying image!")
    result.show()
```

點擊並打開 PyCharm 下方的「Terminal」視窗：

macOS 請輸入： python3 stanCodoshop.py hoover

Windows 請輸入： py stanCodoshop.py hoover

如果看到下方字樣就代表 Milestone 2 完成！

```
Loading hoover/200-500.jpg
Loading hoover/158-500.jpg
Loading hoover/156-500.jpg
[0, 191, 63]
Displaying image!
```

Milestone 3 - get_best_pixel(pixels)

說明

請使用 Milestone 1 及 Milestone 2 所撰寫的指令，完成 get_best_pixel(pixels) 這個 function，並 return「與 avg_point 距離最近的 pixel」（也就是我們要放上空白圖片的那顆 pixel！）

測試

請在「def solve(images): 」裡，將程式碼改成如下圖中紅色框框裡的五行程式碼，來檢查這個 function 的功能是否正確：

```
def solve(images):
    """
    Given a list of image objects, compute and display a Ghost solution image
    based on these images. There will be at least 3 images and they will all
    be the same size.

    Input:
        images (List[SimpleImage]): list of images to be processed
    """
    width = images[0].width
    height = images[0].height
    result = SimpleImage.blank(width, height)
    ######## YOUR CODE STARTS HERE #########
    # Write code to populate image and create the 'ghost' effect
    green_pixel = SimpleImage.blank(20, 20, 'green').get_pixel(0, 0)
    red_pixel = SimpleImage.blank(20, 20, 'red').get_pixel(0, 0)
    blue_pixel = SimpleImage.blank(20, 20, 'blue').get_pixel(0, 0)
    best1 = get_best_pixel([green_pixel, blue_pixel, blue_pixel])
    print(best1.red, best1.green, best1.blue)
    ######## YOUR CODE ENDS HERE ###########
    print("Displaying image!")
    result.show()
```

點擊並打開 PyCharm 下方的「Terminal」視窗：

macOS 請輸入： python3 stanCodoshop.py hoover

Windows 請輸入： py stanCodoshop.py hoover

如果看到下方字樣就代表 Milestone 3 完成！

```
Loading hoover/200-500.jpg
Loading hoover/158-500.jpg
Loading hoover/156-500.jpg
0 0 255
```

Milestone 4 - solve(images)

最後，請同學將上方輸入過的 所有紅色框框的測試程式刪除，並開始建造我們的偉大的「stanCodoshop」APP！

def solve(images) 傳入的 images 為一個 Python List，儲存使用者在同一個景點拍攝的多張大小相同的圖片。您的工作就是在 images 裡，找到每個位置上需要的 pixel 並填在空白圖片 result 與之相同的位置上。

我們在 def solve(images) 的最後寫了 result.show()。因此若您成功完成這份作業，在 30 秒內「只有景點、沒有路人的照片」就會出現在您的螢幕上。

為了測試程式是否正確，我們提供了四套圖組（皆為史丹佛大學知名的景點！）
請在 Terminal 輸入下方指令：

macOS/

1. python3 stanCodoshop.py clock-tower
2. python3 stanCodoshop.py hoover
3. python3 stanCodoshop.py math-corner
4. python3 stanCodoshop.py monster

Windows/

1. py stanCodoshop.py clock-tower
2. py stanCodoshop.py hoover
3. py stanCodoshop.py math-corner
4. py stanCodoshop.py monster

評分標準

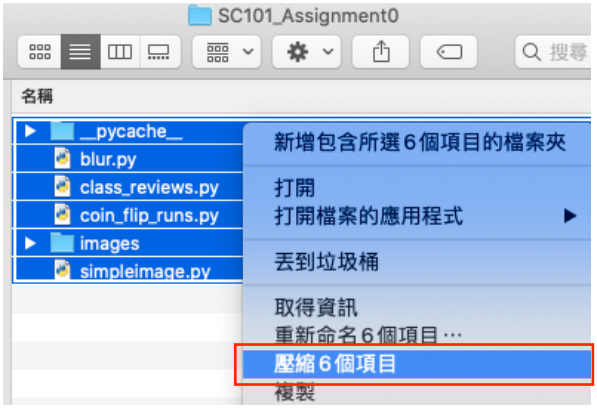
Functionality - 程式是否有通過我們的基本要求？程式必須沒有 bug 、能順利完成指定的任務、並確保程式沒有卡在任何的無限環圈（Infinite loop）之中。

Style - 好的程式要有好的使用說明，也要讓人一目瞭然，這樣全世界的人才能使用各位的 code 去建造更多更巨大更有趣的程式。因此請大家寫精簡扼要的使用說明、function敘述、單行註解。

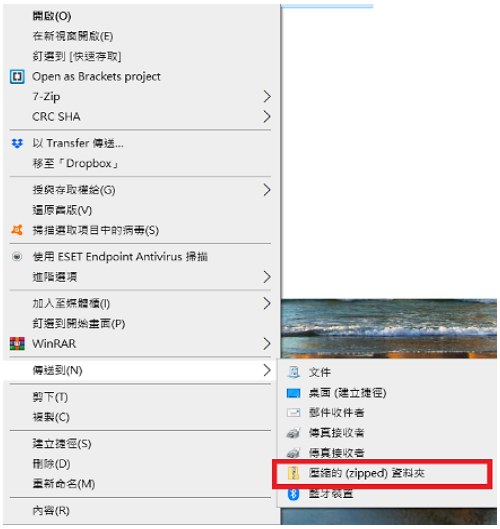
作業繳交

1. 以滑鼠「全選」作業資料夾內的所有檔案，並壓縮檔案。請見下圖說明。

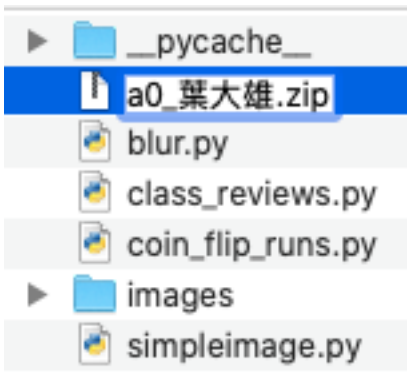
macOS：按右鍵選擇「壓縮n個項目」



Windows：按右鍵選擇「傳送到」→「壓縮的(zipped)資料夾」



2. 將壓縮檔(.zip)重新命名為「a(n)_中文姓名」。如：
   assignment 0命名為a0_中文姓名;
   assignment 1命名為a1_中文姓名; …



3. 將命名好的壓縮檔(.zip)上傳至Google Drive（或任何雲端空間）

   1) 搜尋「google drive」
   2) 登入後，點選左上角「新增」→ 「檔案上傳」→ 選擇作業壓縮檔(.zip)

4. 開啟連結共用設定，並複製下載連結

   1) 對檔案按右鍵，點選「共用」
   2) 點擊「變更任何知道這個連結的使用者權限」後，權限會變為「可檢視」
   3) 點選「複製連結」

5. 將連結上傳至臉書社團的作業貼文提供的「作業提交表單」