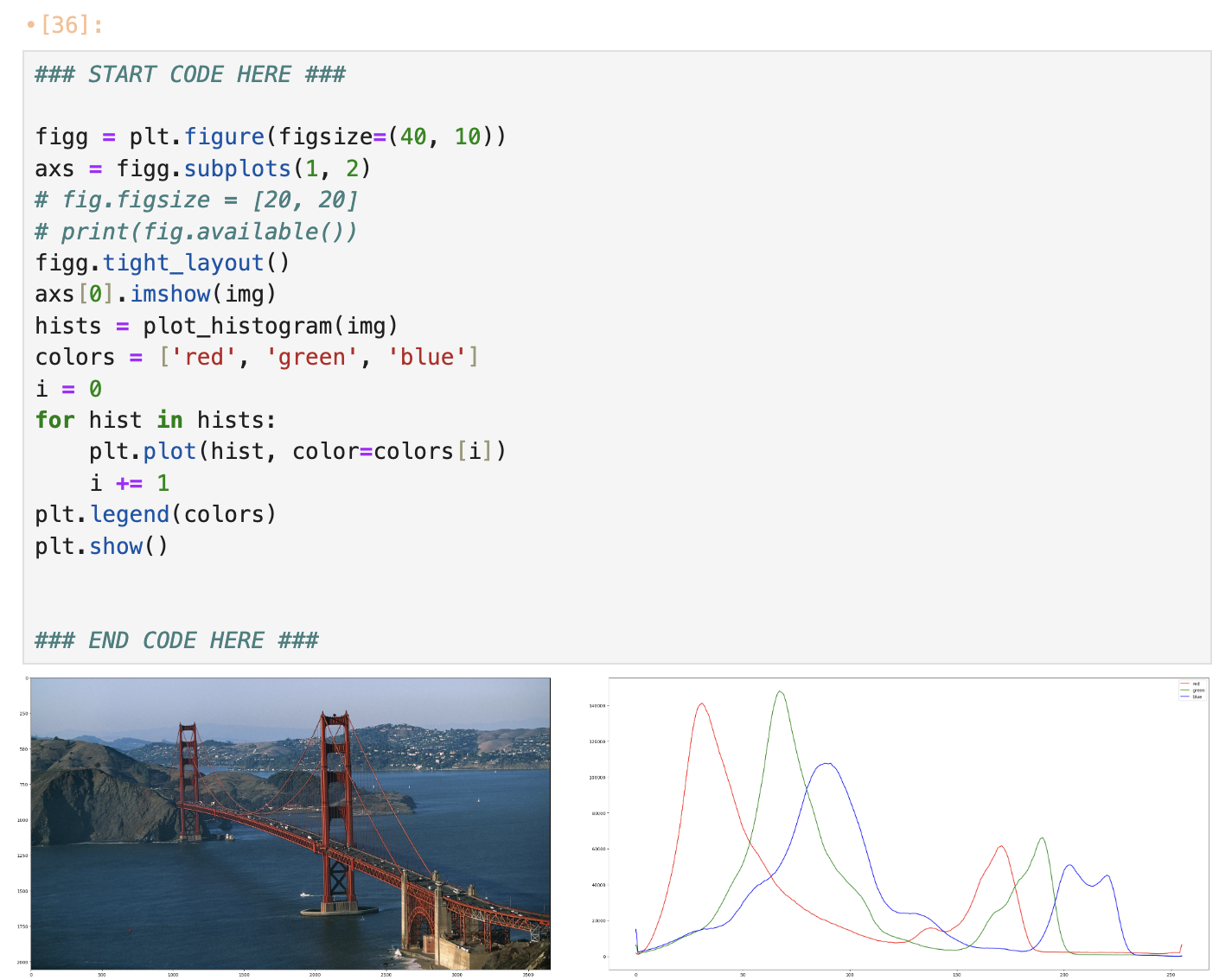
**Lab2-2 : Image Enhancement with Statistical Operation**

**Histogram Equalization**

****

* Return ออกมาเป็น List ของ Histogram ที่มีขนาดเป็น 3 โดยจะเป็น Histogram ของสีแดง สีเขียว สีฟ้า (RGB) ตามลำดับ
* ใช้ฟังชั่น cv2.calcHist() โดยจะรับ parameter เป็น image และ index ของสีที่สนใจ
* นอกจากนี้ยังรับ mask ที่เราไม่ได้ต้องการใช้ และก็เป็น size และ range ของ dimension สีของรูปภาพ

**Before:**

****

* นำ list ของ Histogram data จากฟังชั่นข้างต้นมาแสดงทีละอัน โดยใช้สีที่แตกต่างกัน โดยใช้ฟังชั่น plt.ploy(hist, color=colors[i]) โดย hist คือ iterator ของ list และ colors = [‘red’, ‘green’, ‘blue’ ]

**After:**

**A screenshot of a computer

Description automatically generated**

* นำรูปภาพเดิม(ซึ่งเป็น array 3 มิติ) มาแยกเป็น array 2 มิติจำนวน 3 อัน เป็นของ channel R, G และ B ตามลำดับ จากนั้นนำแต่ละ Channel มาเข้าเป็นพารามิเตอร์ของฟังก์ชัน cv2.equalizeHist() เพื่อกระจายเฉดสีแต่ละสี

**Image Histogram Matching**

**Load image:**

**A white background with red and blue text

Description automatically generated**

* โหลดรูปภาพโดยใช้ฟังชั่น cv2.imread() แล้วนำมา convert จากที่ได้เป็น BGR ให้เป็น RGB ด้วยฟังก์ชัน cv2.cvtColor()

**Display Histogram matching:**

**A screenshot of a computer

Description automatically generated**

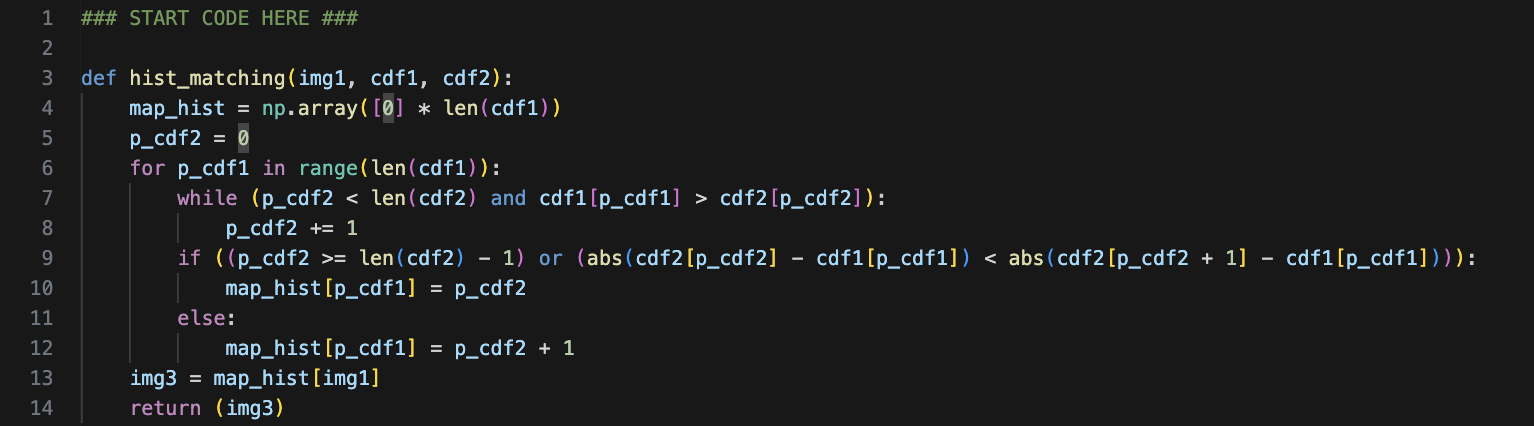
* ในการ display PDF เราสามารถ plot ได้โดยตรงจาก histrogram data ข้อก่อนหน้าได้เลย แต่ว่าในแนวแกน y เราต้องเป็นความน่าจะเป็น เราจึงต้องหารผลรวมของ histogram นั้นด้วย
* ส่วนในการ display CDF เราจำเป็นต้องนำมาเข้าฟังชั่น cumsum() เพื่อซึ่งฟังก์ชันนี้จะ return ออกมาเป็น list ใหม่ซึ่งคำนวณออกมาเป็น cumulative แล้ว ยกตัวอย่างเช่นหาก input เป็น [1,2,3,4] จะได้ output เป็น [1, 3, 6, 10]

**Histrogram Matching**

**ผลลัพธ์:**

**A collage of graphs and diagrams

Description automatically generated**

****

* โดยฟังก์ชั่นนี้จะเป็นฟังก์ชันหลักในการทำ histogram matching มีหลักการทำคือเราจำรับ cdf ทั้งของ input และ template มา และเราจะ loop cdf ของ input ไปเรื่อยๆ ว่า output ควรออกมาเป็นอะไร
* โดยทุกครั้งที่ loop เราจะเทียบกับตัว template ให้มันห่างกับ template น้อยสุด เช่น
  + หากเราลูปอยู่ที่ 0.23 แล้ว template เป็น [0.1, 0.2, 0.3, …] เราก็จะเลือกตัวที่มีความห่างน้อยที่สุด ก็คือ 0.2
* Pseudo code
  + สร้างตัวแปรไว้เก็บค่า output ของ matching hist
  + Loop ทุก iterator ของ input cdf
    - ลูปเพื่อทำให้ iterator ของ template cdf เป็นค่ามากสุดที่ยังไม่เกิน iterator ของ input cdf
    - เช็คว่า iterator ของ template cdf กับ iterator + 1 (ตัวถัดไป) อันไหนมีความห่างกับ iterator ของ input cdf น้อยกว่ากัน แล้วเอาตัวนั้น
  + บรรทัดที่ 13 จะเป็นการนำ map มาลงที่ images โดย img1 จะทำหน้าที่เป็น index ซึ่งในกรณีนี้จะเป็นค่าสี และนำมาทำการ map กับ map\_hist ซึ่งก็จะเป็นการอาแต่ละค่าสีของรูปจริง มาเทียบกับ map\_hist แล้วเอาสีจาก map\_hist นั้นไปแทน

A computer screen shot of a program code

Description automatically generated

* พล็อต 2 แถวแรก เหมือนข้อก่อนหน้า แต่ไม่ได้ทำเป็นลูปแล้ว

A screen shot of a computer

Description automatically generated

* รับรูปภาพที่ผ่านการทำ histogram matching มาโดยเราจะ slice ทีละ Channel เนื่องจากฟังก์ชั่นที่เราใช้ matching เราทำให้มันสามารถรับได้แค่ 2 มิติ และรวมกันด้วย np.array() ซึ่งเป็น contructor ของ numpy array
* พอเรารับมาแล้วนำมารวมกัน มันจะได้ shape เป็น ​(COLOR, HEIGHT, WIDGTH) เราจึงต้องนำมา transpose ด้วย np.transpose และใช้พารามิเตอร์เป็น (1,2,0) เพื่อให้ได้ shape เป็น (HEIGHT, WIDGTH, COLOR)
* เนื่องจาก img3 มี depth ที่ไม่ตรงกับที่ plt.imshow() กับ calcHist() รับได้ เราจึงต้องปรับ img ก่อนโดยใช้ฟังก์ชัน np.clip() และ cv2.convertScaleAbs()