CBB110213 李明發 影像處理導論 期末考繳交

第一題:

RGB MODEL:

import numpy as np

import cv2

def RGB\_model(f,channel):

if channel == 1:

return f[:,:,2]

elif channel == 2:

return f[:,:,1]

else:

return f[:,:,0]

def main():

img = cv2.imread("Pepper.bmp")

R = RGB\_model(img,1)

G = RGB\_model(img,2)

B = RGB\_model(img,3)

cv2.imshow("Original Image",img)

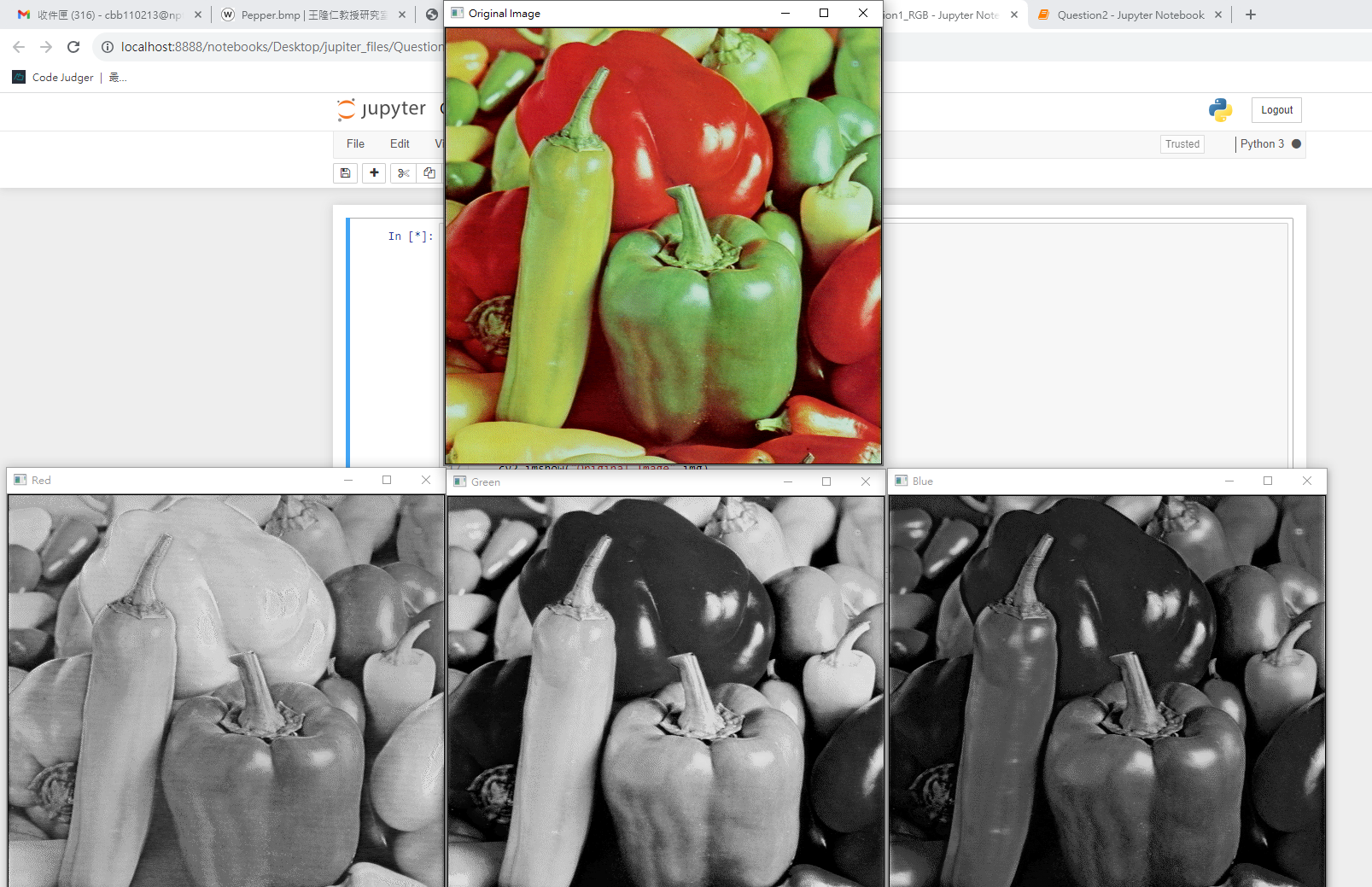
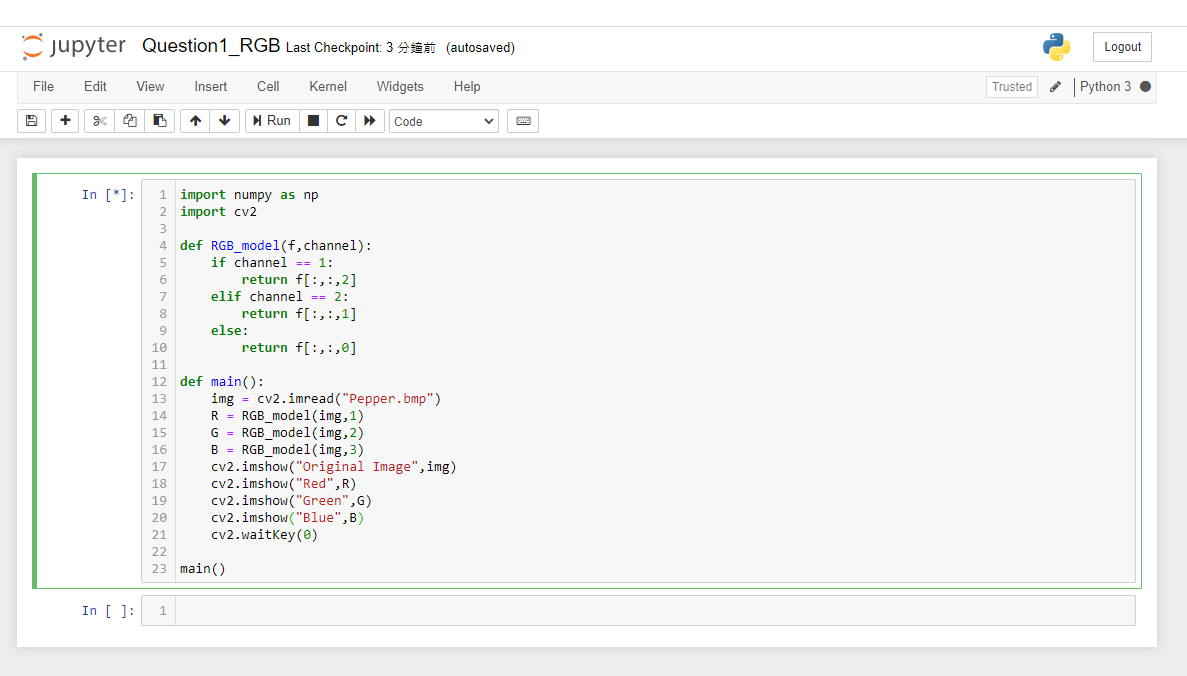
cv2.imshow("Red",R)

cv2.imshow("Green",G)

cv2.imshow("Blue",B)

cv2.waitKey(0)

main()



CMY MODEL:

import numpy as np

import cv2

def CMY\_model(f,channel):

if channel == 1:

return 255 - f[:,:,2]

elif channel == 2:

return 255 - f[:,:,1]

else:

return 255 - f[:,:,0]

def main():

img = cv2.imread("Pepper.bmp",-1)

C = CMY\_model(img,1)

M = CMY\_model(img,2)

Y = CMY\_model(img,3)

cv2.imshow("Original Image",img)

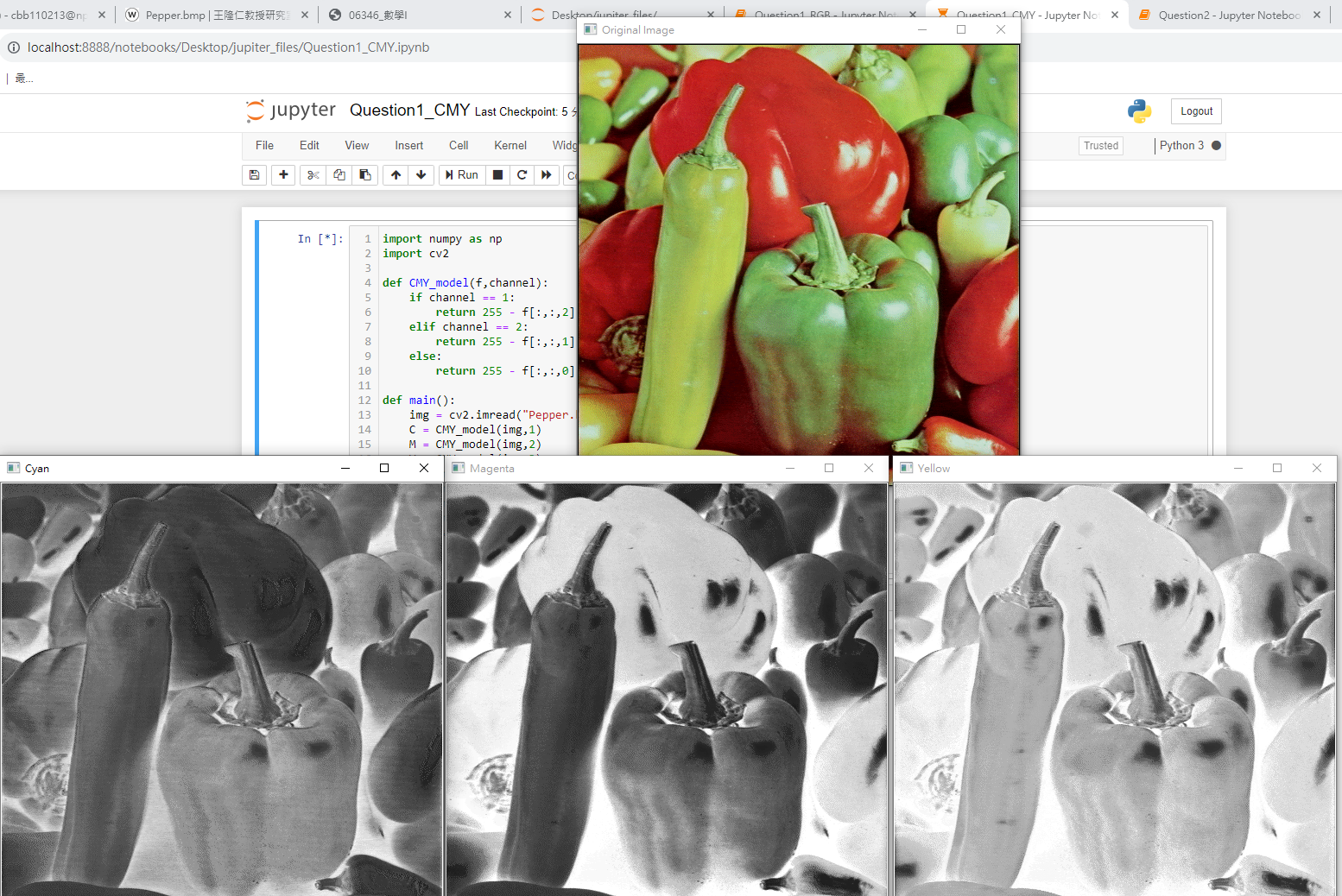
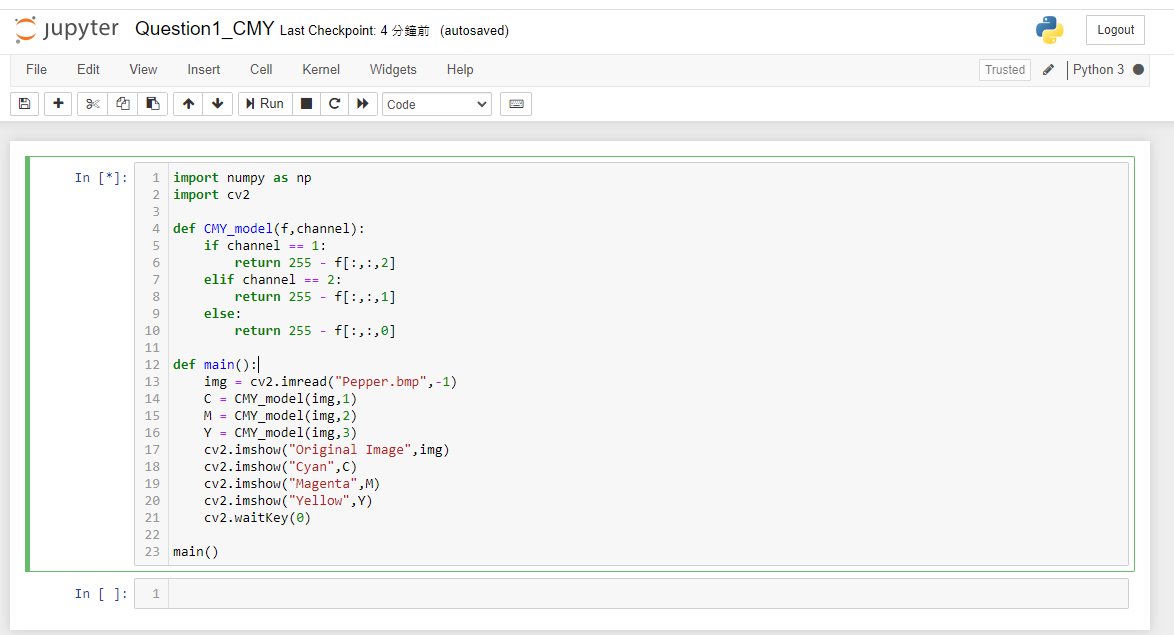
cv2.imshow("Cyan",C)

cv2.imshow("Magenta",M)

cv2.imshow("Yellow",Y)

cv2.waitKey(0)

main()



HSI MODEL:

import numpy as np

import cv2

def RGB\_to\_HSI(R,G,B):

r = R/255

g = G/255

b = B/255

if R==G and G==B:

H = -1.0

S = 0.0

I = (r+g+b)/3

else:

x = (0.5\*((r-g)+(r-b)))/\

np.sqrt((r-g)\*\*2+(r-b)\*(g-b))

if x<-1.0: x=-1.0

if x>1.0: x=1.0

theta = np.arccos(x)\*180/np.pi

if B<=G:

H = theta

else:

H = 360.0-theta

S=1.0-3.0/(r+g+b)\*min(r,g,b)

I = (r+g+b)/3

return H,S,I

def HSI\_model(f,channel):

nr,nc = f.shape[:2]

g = np.zeros([nr,nc],dtype='uint8')

if channel == 1:

for x in range(nr):

for y in range(nc):

H,S,I = RGB\_to\_HSI(f[x,y,2],f[x,y,1],f[x,y,0])

if H==-1:

k=0

else:

k=round(H\*255/360)

g[x,y]=np.uint8(k)

elif channel == 2:

for x in range(nr):

for y in range(nc):

H,S,I = RGB\_to\_HSI(f[x,y,2],f[x,y,1],f[x,y,0])

k = round(S\*255)

g[x,y] = np.uint8(k)

else:

for x in range(nr):

for y in range(nc):

H,S,I = RGB\_to\_HSI(f[x,y,2],f[x,y,1],f[x,y,0])

k = round(I\*255)

g[x,y] = np.uint8(k)

return g

def main():

img = cv2.imread("Pepper.bmp",-1)

H = HSI\_model(img,1)

S = HSI\_model(img,2)

I = HSI\_model(img,3)

cv2.imshow("Original Image",img)

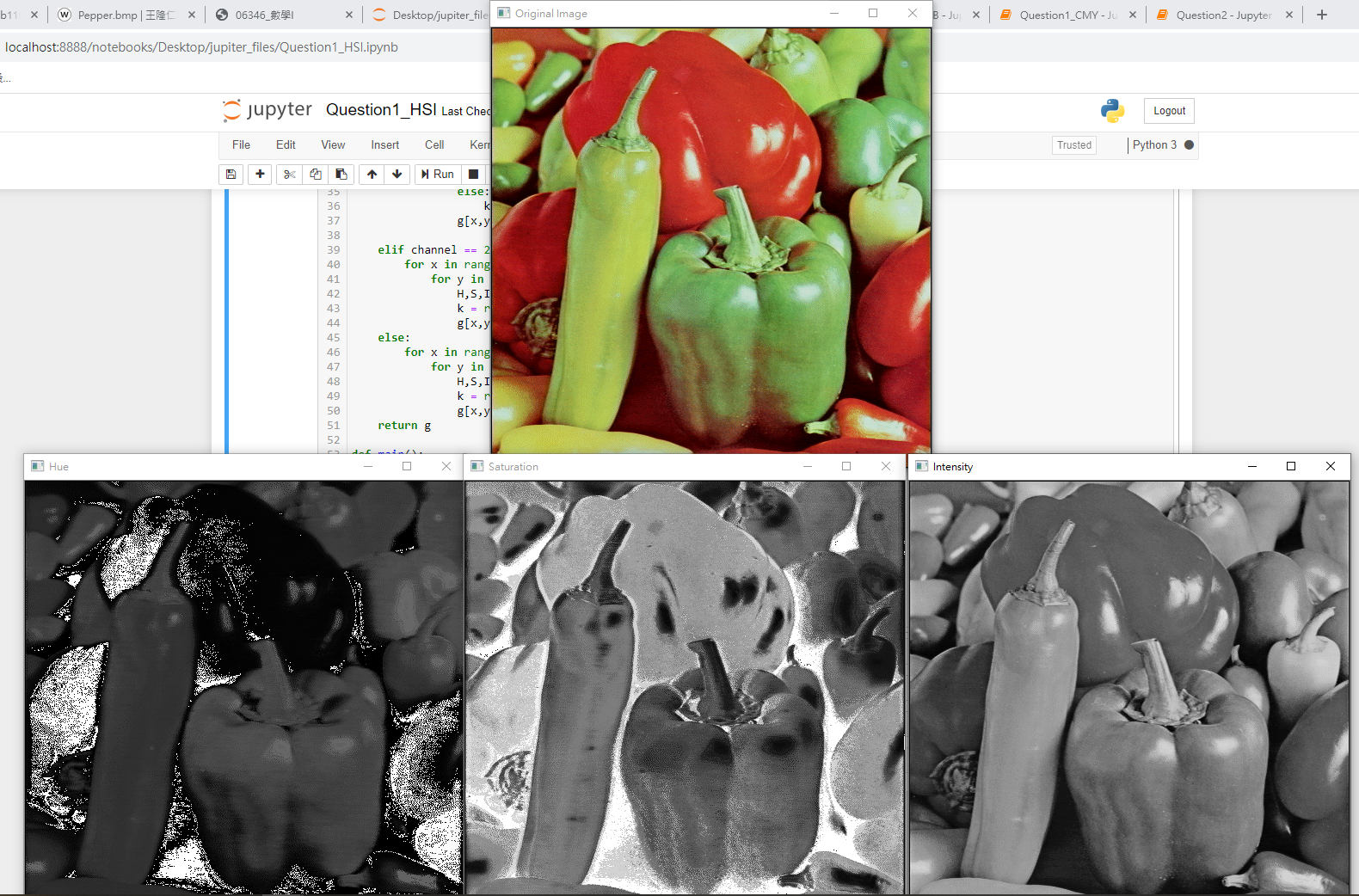
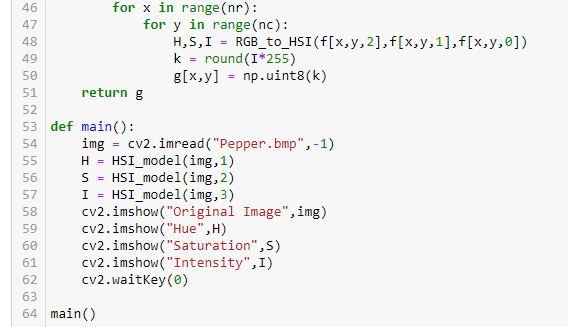
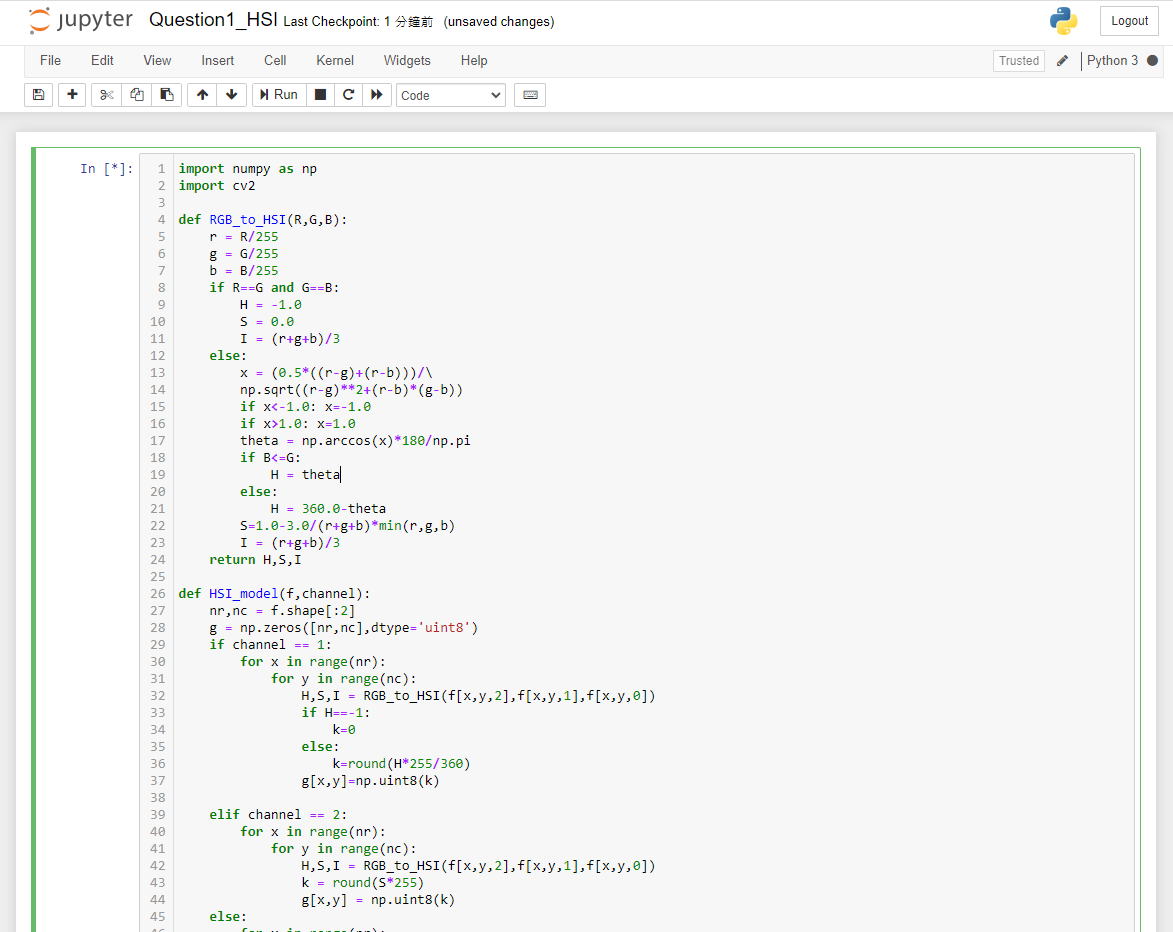
cv2.imshow("Hue",H)

cv2.imshow("Saturation",S)

cv2.imshow("Intensity",I)

cv2.waitKey(0)

main()



HSV MODEL:

import numpy as np

import cv2

def HSV\_model(f,channel):

hsv = cv2.cvtColor(f,cv2.COLOR\_BGR2HSV)

if channel == 1:

return hsv[:,:,0]

elif channel == 2:

return hsv[:,:,1]

else:

return hsv[:,:,2]

def main():

img = cv2.imread("Pepper.bmp",-1)

H = HSV\_model(img,1)

S = HSV\_model(img,2)

V = HSV\_model(img,3)

cv2.imshow("Original Image",img)

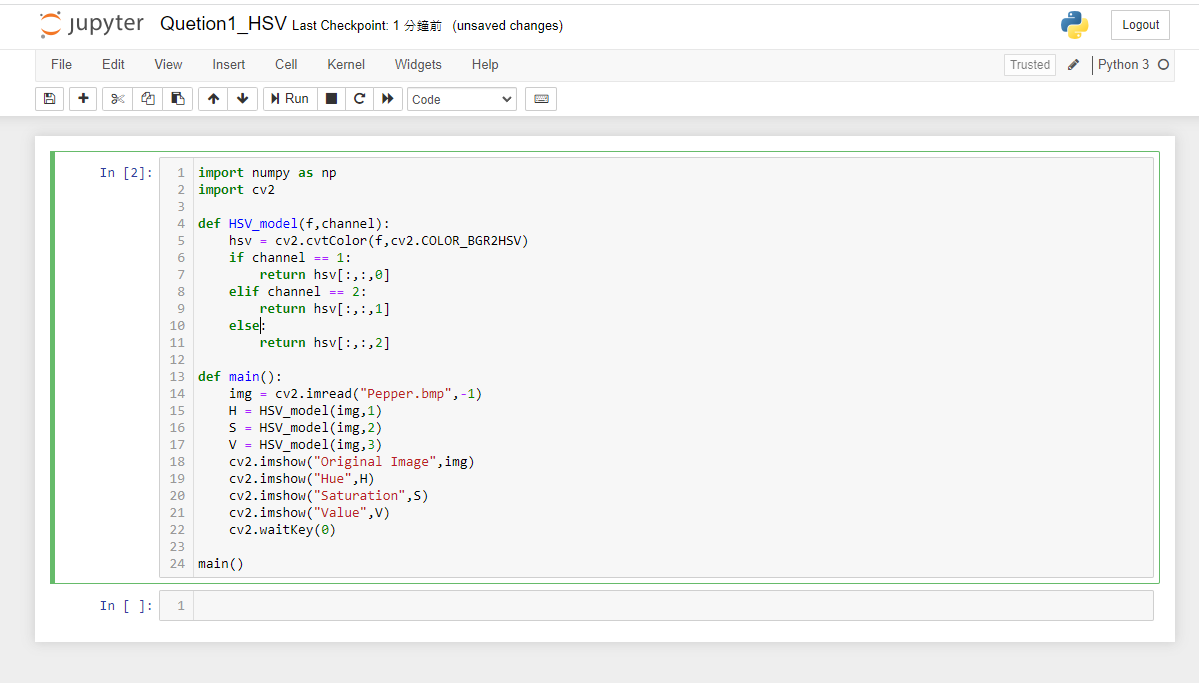
cv2.imshow("Hue",H)

cv2.imshow("Saturation",S)

cv2.imshow("Value",V)

cv2.waitKey(0)

main()



YCrCb:

import numpy as np

import cv2

def YCrCb\_model(f,channel):

ycrcb = cv2.cvtColor(f,cv2.COLOR\_BGR2YCrCb)

if channel == 1:

return ycrcb[:,:,0]

elif channel == 2:

return ycrcb[:,:,1]

else:

return ycrcb[:,:,2]

def main():

img = cv2.imread("Pepper.bmp",-1)

Y = YCrCb\_model(img,1)

Cr = YCrCb\_model(img,2)

Cb = YCrCb\_model(img,3)

cv2.imshow("Original Image",img)

cv2.imshow("Y",Y)

cv2.imshow("Cr",Cr)

cv2.imshow("Cb",Cb)

cv2.waitKey(0)

main()



第二題:

import numpy as np

import cv2

import pywt

def DWT\_image(f,wavelet):

nr,nc = f.shape[:2]

coeffs = pywt.dwt2(f,wavelet)

LL,(LH,HL,HH) = coeffs

nr1,nc1 = LL.shape[:2]

g = np.zeros([nr1\*2,nc1\*2],dtype='uint8')

LL\_normalized = np.zeros([nr1,nc1])

cv2.normalize(LL,LL\_normalized,0,255,cv2.NORM\_MINMAX)

g[0:nr1,0:nc1] = np.uint8(LL\_normalized[:,:])

return g

def IDWT\_image(LL, wavelet):

coeffs = LL, (None, None, None)

img = pywt.idwt2(coeffs, wavelet)

return np.uint8(img)

def PSNR(f,g):

nr,nc = f.shape[:2]

MSE = 0.0

for x in range(nr):

for y in range(nc):

MSE+=(float(f[x,y])-float(g[x,y]))\*\*2

MSE/=(nr\*nc)

PSNR = 10\*np.log10((255\*255)/MSE)

return PSNR

def main():

img1 = cv2.imread("House.bmp",-1)

img2 = DWT\_image(img1,'db1')

reconstructed\_img = IDWT\_image(img2,'db1')

cv2.imshow("Original Image",img1)

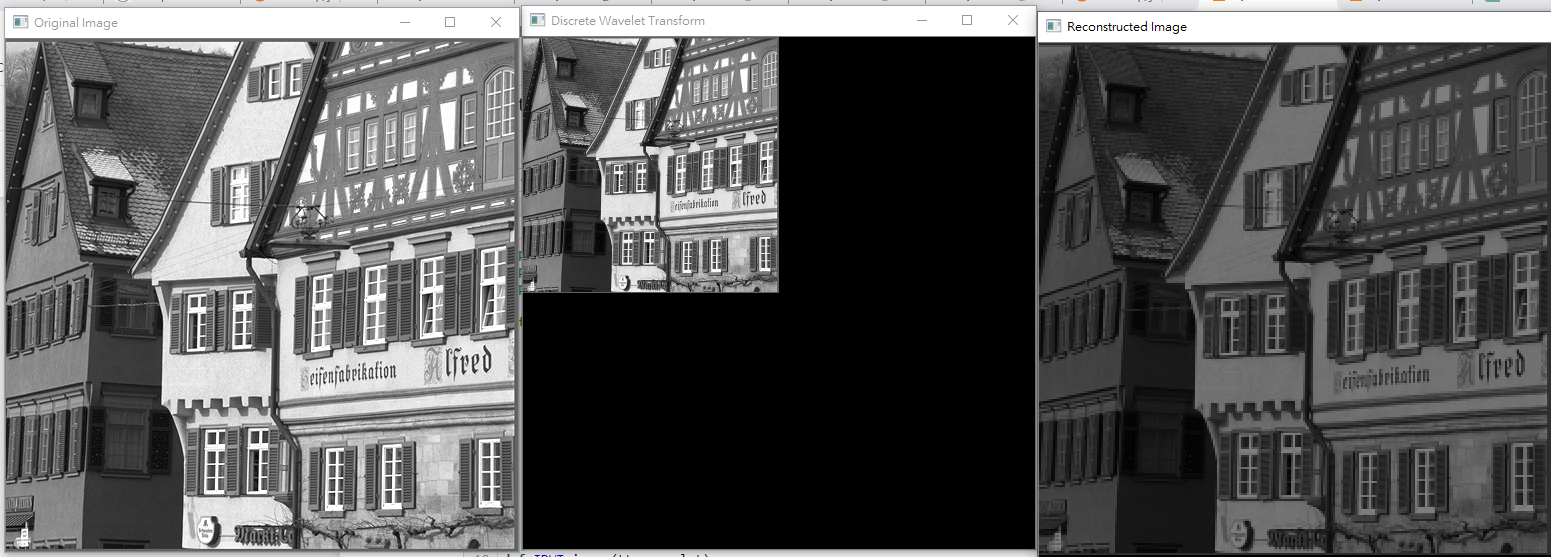
cv2.imshow("Discrete Wavelet Transform",img2)

cv2.imshow("Reconstructed Image",reconstructed\_img)

print("The PSNR = ",PSNR(img1,reconstructed\_img))

cv2.waitKey(0)

main()



第三題:

import numpy as np

import cv2

def entropy(f):

nr,nc = f.shape[:2]

pdf = np.zeros(256)

for x in range(nr):

for y in range(nc):

pdf[f[x,y]]+=1

pdf/=(nr\*nc)

H=0

for k in range(256):

if pdf[k]!=0:

H+=(-pdf[k]\*np.log2(pdf[k]))

return H

def main():

img1 = cv2.imread("Barbara.bmp",-1);

img2 = cv2.imread("Osaka.bmp",-1);

cv2.imshow("Barbara.bmp",img1)

cv2.imshow("Osaka.bmp",img2)

H1 = entropy(img1);

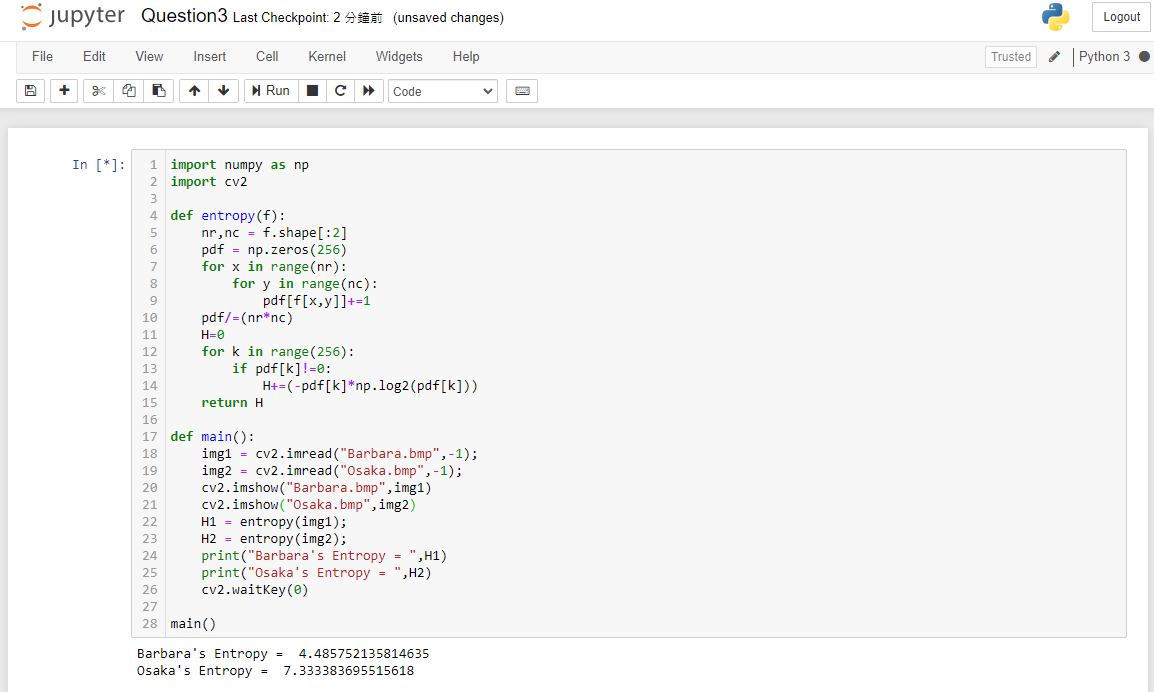
H2 = entropy(img2);

print("Barbara's Entropy = ",H1)

print("Osaka's Entropy = ",H2)

cv2.waitKey(0)

main()



第四題:

CNN:

from keras.models import Sequential

from keras.layers.core import Dense, Dropout

from keras.layers import Conv2D, MaxPooling2D, Flatten

from keras.utils import np\_utils

from keras.datasets import mnist

def load\_data():

(x\_train,y\_train),(x\_test,y\_test) = mnist.load\_data()

number = 1000

x\_train = x\_train[0:number]

y\_train = y\_train[0:number]

x\_train = x\_train.reshape(number,28,28,1)

x\_test = x\_test.reshape(x\_test.shape[0],28,28,1)

x\_train = x\_train.astype('float')

x\_test = x\_test.astype('float')

y\_train = np\_utils.to\_categorical(y\_train,10)

y\_test = np\_utils.to\_categorical(y\_test,10)

return (x\_train,y\_train),(x\_test,y\_test)

(x\_train,y\_train),(x\_test,y\_test)=load\_data()

model = Sequential()

model.add(Conv2D(30,(3,3),padding = 'same',input\_shape = (28,28,1),activation = 'relu'))

model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Conv2D(30,(3,3),padding='same',activation = 'relu'))

model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Flatten())

model.add(Dense(units=200,activation='relu'))

model.add(Dense(units=10,activation='softmax'))

model.compile(loss='categorical\_crossentropy',optimizer='adam',metrics=['accuracy'])

model.fit(x\_train,y\_train,batch\_size=100,epochs=100,validation\_data=(x\_test,y\_test),shuffle=True)

result - models.evaluate(x\_test,y\_test)

print('\n Test Acc:',result[1])

y\_pred = model.predict(x\_test)

y\_pred\_class = model.predict\_classes(x\_test)

ANN:

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Dense

from keras.utils import to\_categorical

(train\_images,train\_labels),(test\_images,test\_labels) = mnist.load\_data()

network = Sequential()

network.add(Dense(512,activation='relu',input\_shape=(784,)))

network.add(Dense(10,activation='softmax'))

network.compile(optimizer = 'rmsprop',loss = 'categorical\_crossentropy',metrics = ['accuracy'])

print(network.summary())

train\_images = train\_images.reshape((60000,28\*28))

train\_images = train\_images.astype('float32')/255

test\_images = test\_images.reshape((10000,28\*28))

test\_images = test\_images.astype('float32')/255

train\_labels = to\_categorical(train\_labels)

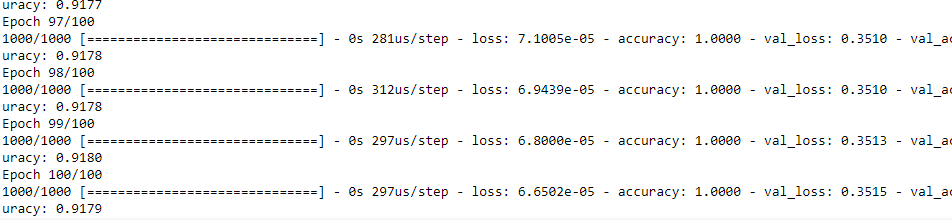
test\_labels = to\_categorical(test\_labels)

network.fit(train\_images,train\_labels,epochs = 5, batch\_size=200)

test\_loss,test\_acc = network.evaluate(test\_images,test\_labels)

print("Test Accuracy:",test\_acc)

CNN:



ANN:

