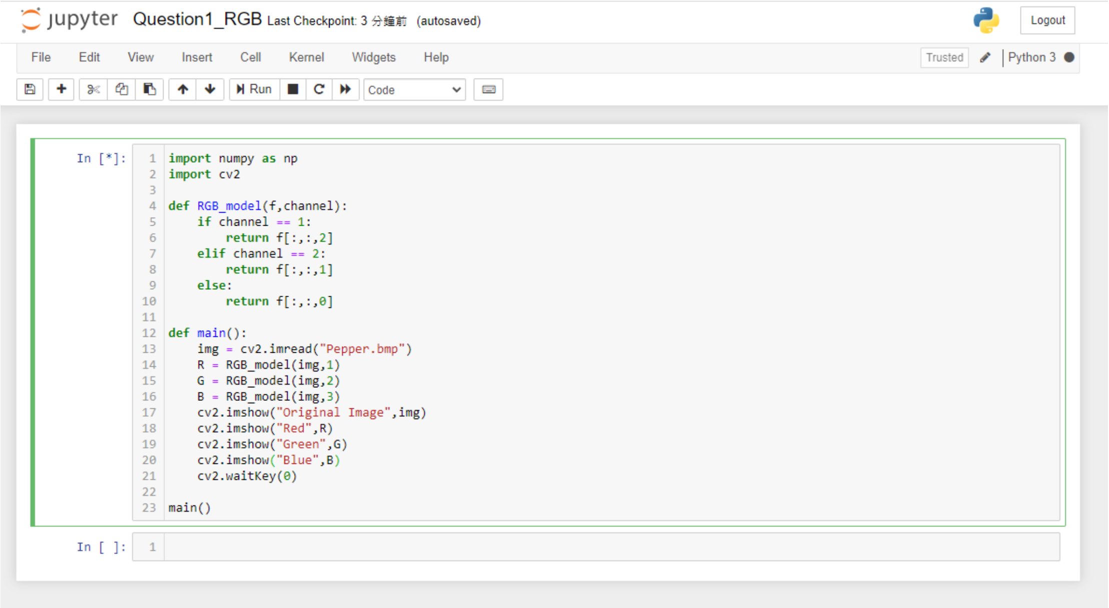
CBB110213 李明發 影像處理導論 期末考繳交

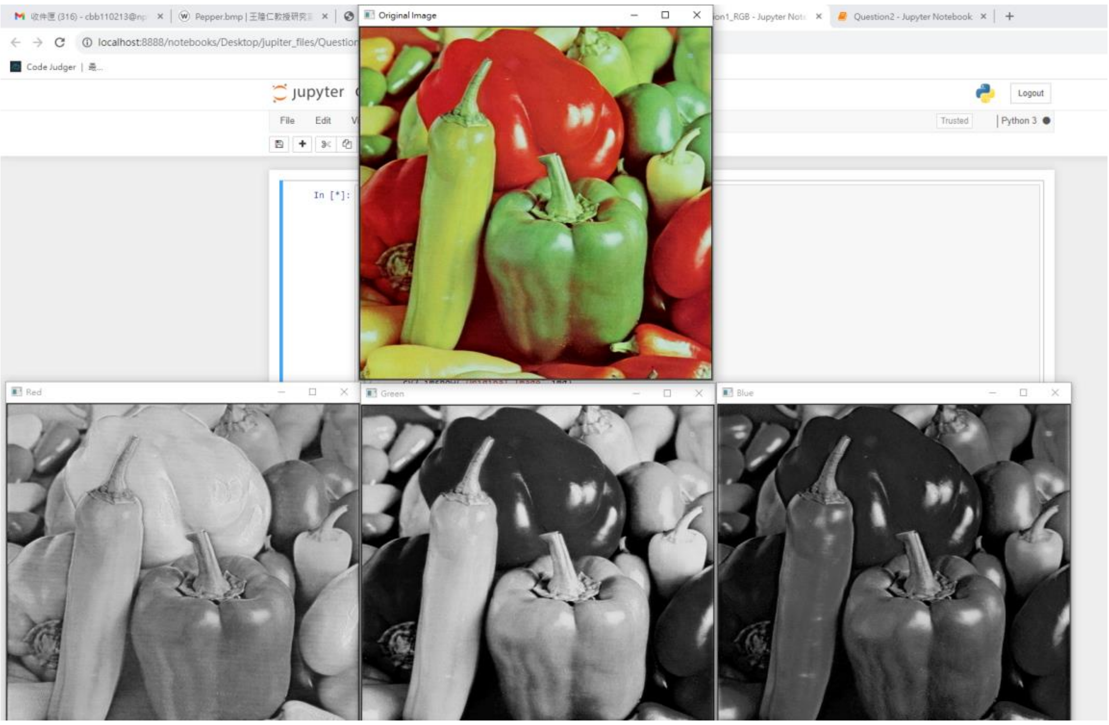
第一題:

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()
```

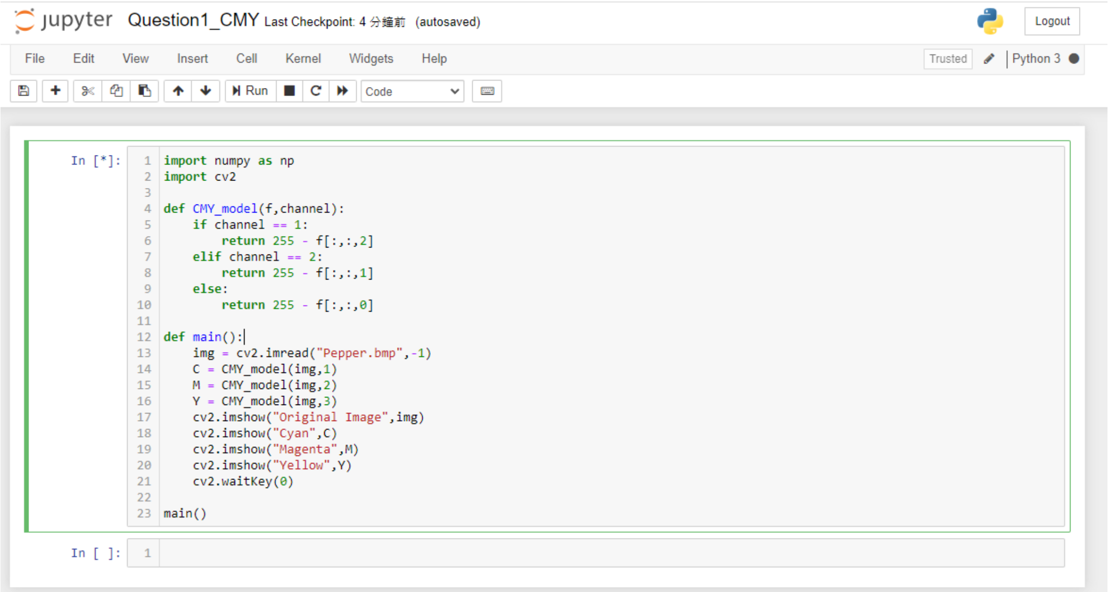


```
import numpy as np
import cv2

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()
```

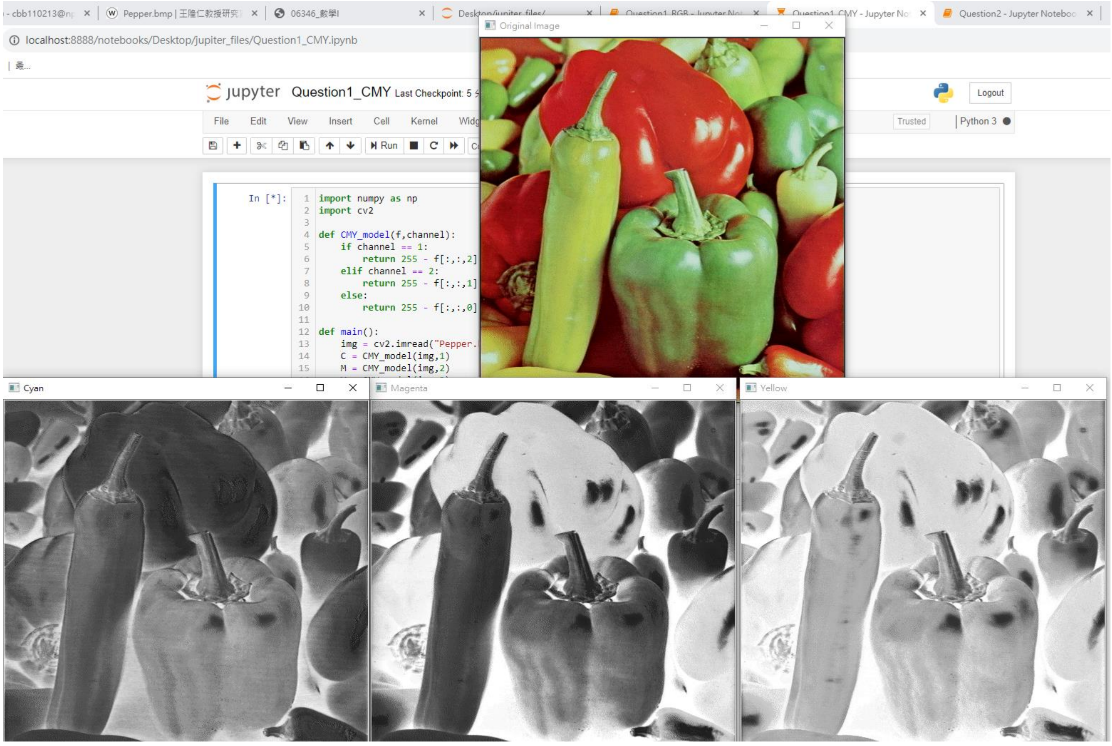

Original Image
Red
Green
Blue
localhost:8888/notebooks/Desktop/jupiter_files/Question
Question2 - Jupyter Notebook
jupyter
Logout
Trusted
Python 3
In [*]:

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()
```



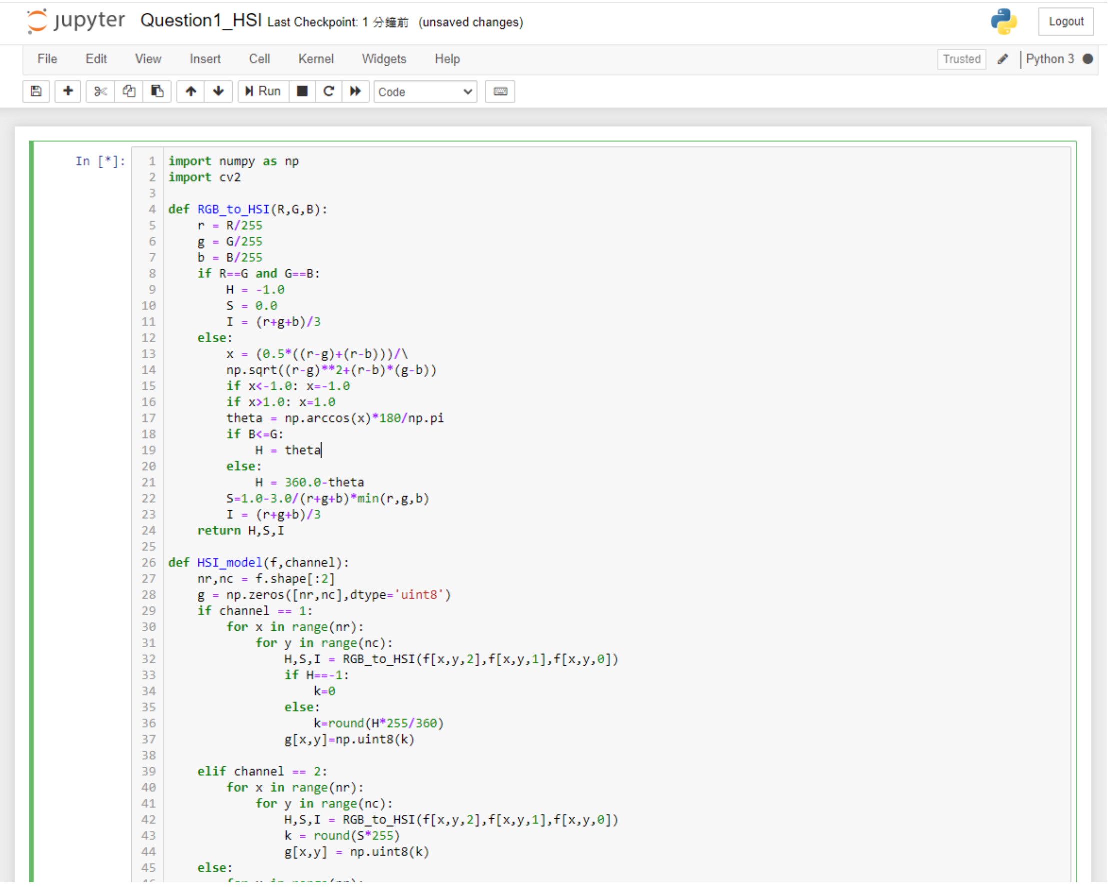
```
import numpy as np
import cv2

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()
```


localhost:8888/notebooks/Desktop/jupiter_files/Question1_CMY.ipynb
jupyter Question1_CMY Last Checkpoint: 5
File Edit View Insert Cell Kernel
Run
Logout
Trusted
Python 3
Original Image
Cyan
Magenta
Yellow

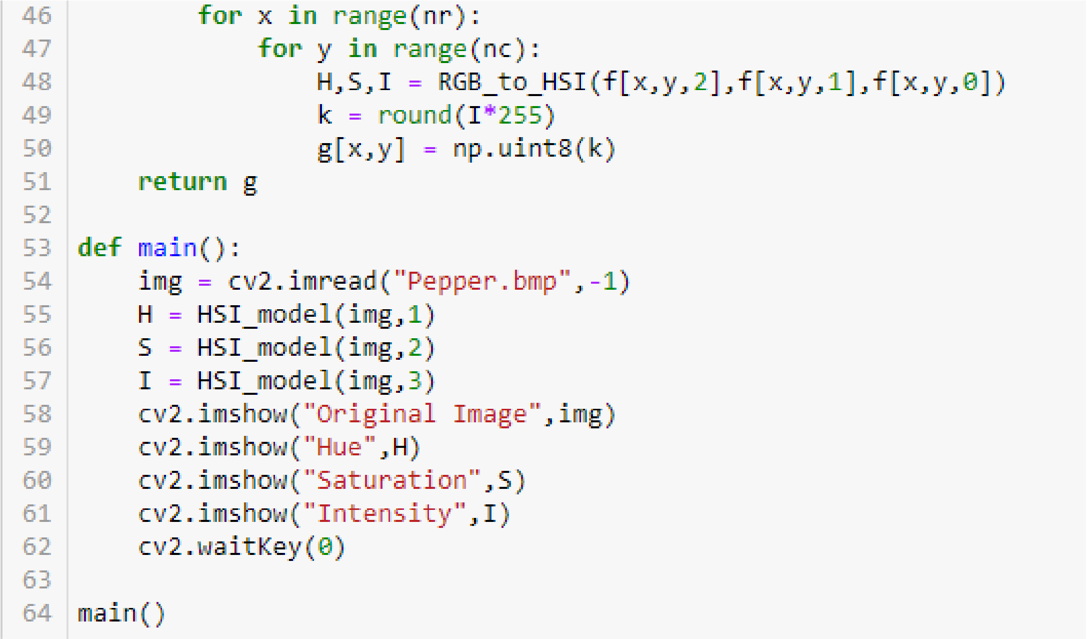
```
In [*]:
import numpy as np
import cv2

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.
    C = CMY_model(img,1)
    M = CMY_model(img,2)
```

HSI MODEL:

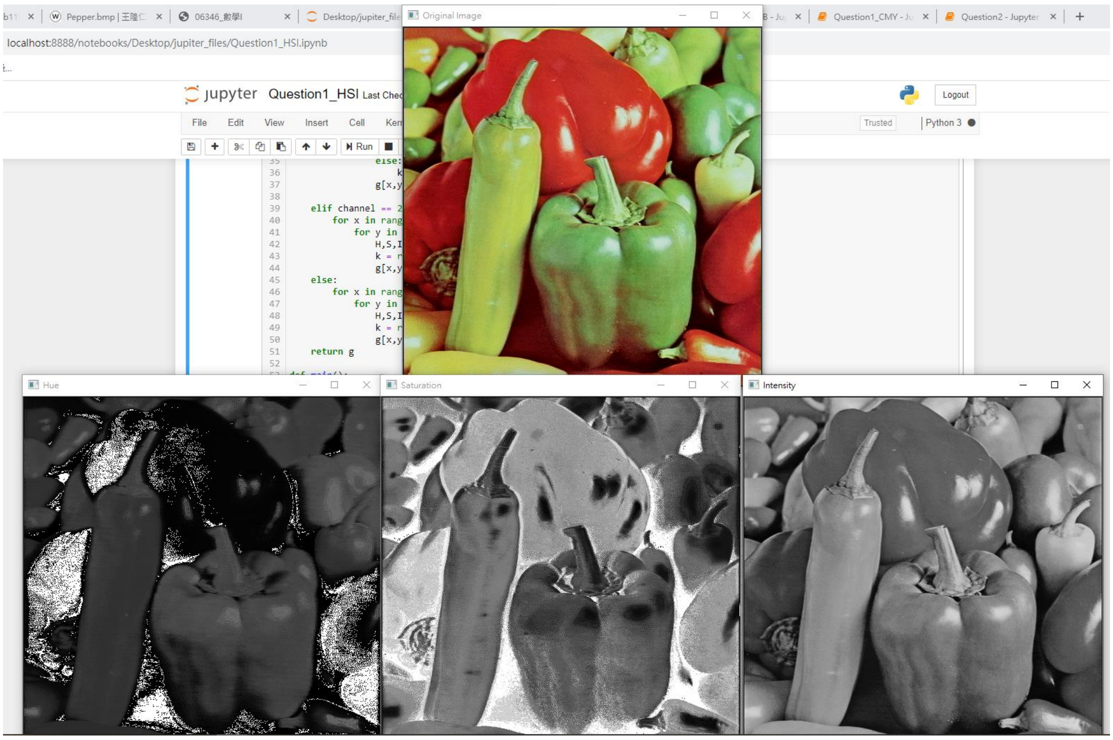
```
import numpy as np
import cv2

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()
```



```
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 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()
```



localhost:8888/notebooks/Desktop/jupiter_files/Question1_HSI.ipynb

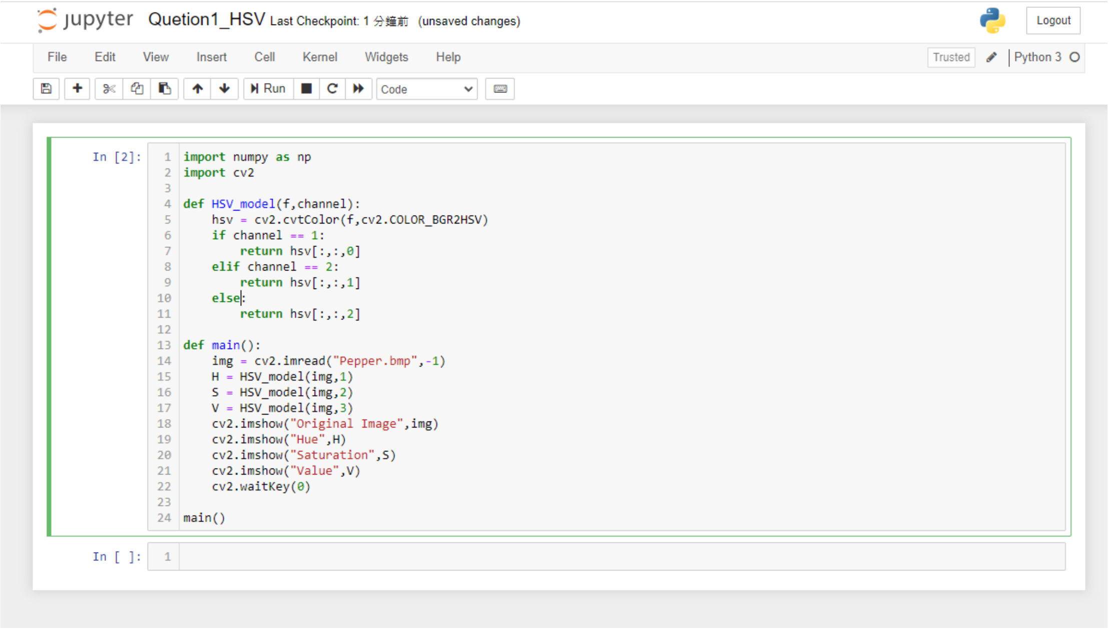
```
            else:
                k
                g[x,y
    elif channel == 2
        for x in rang
            for y in
                H,S,I
                k = r
                g[x,y
    else:
        for x in rang
            for y in
                H,S,I
                k = r
                g[x,y
    return g
```

HSV MODEL:

import numpy as np
import cv2

def HSV_model(f,channel):

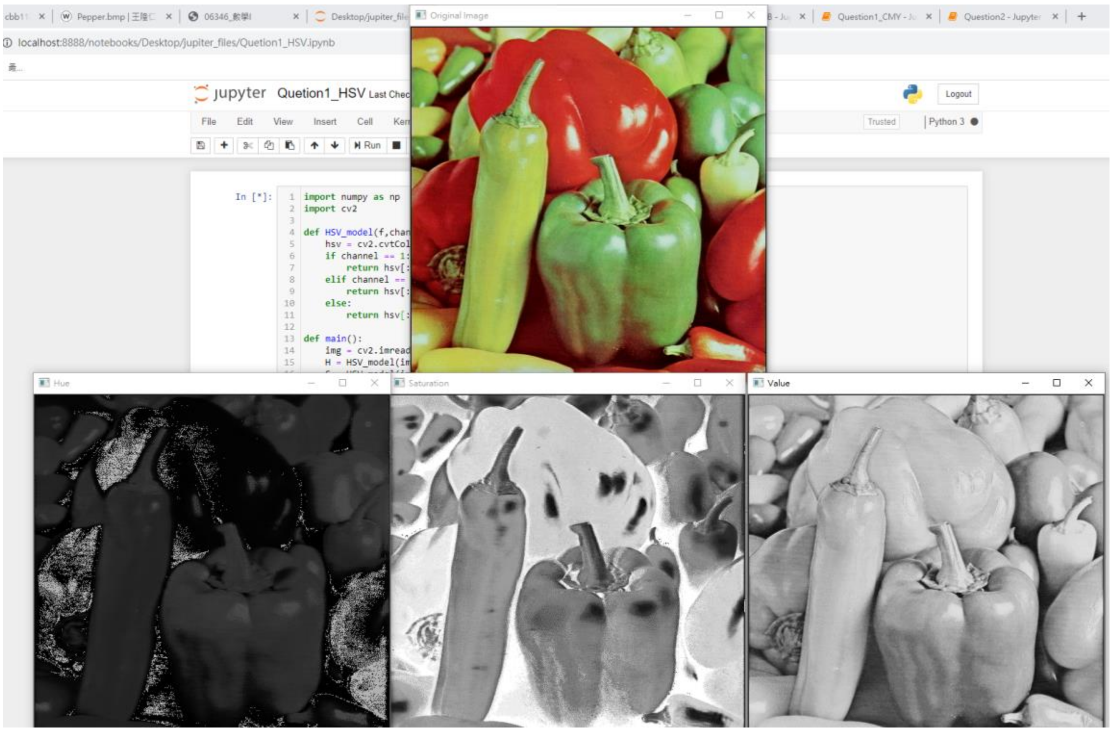
```
    hsv = cv2.cvtColor(f,cv2.COLOR_BGR2HSV)
    if channel == 1:
        return hsv[:,:,0]
    elif channel == 2:
        return hsv[:,:,1]
    else:
        return hsv[:,:,2]

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()
```



In [2]:

```
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()
```

In [ ]:

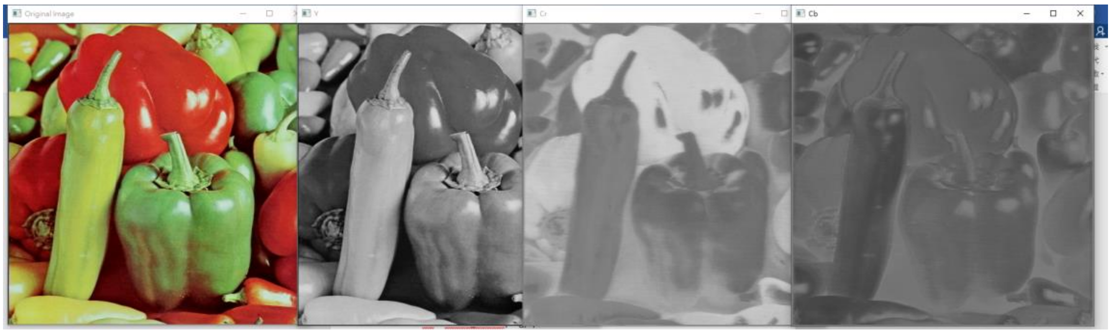


```
In [*]:
import numpy as np
import cv2

def HSV_model(f,chan
    hsv = cv2.cvtCol
    if channel == 1:
        return hsv[:
    elif channel ==
        return hsv[:
    else:
        return hsv[:

def main():
    img = cv2.imread
    H = HSV_model(im
```

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)
```

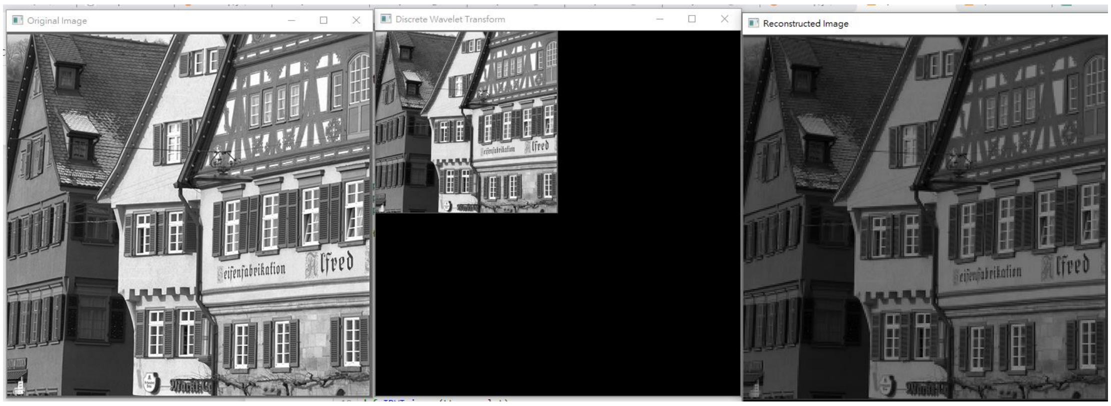
```
    cv2.imshow("Cr",Cr)
    cv2.imshow("Cb",Cb)
    cv2.waitKey(0)

main()
```

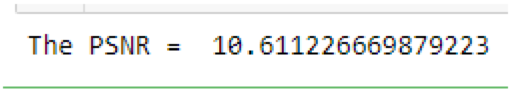


第二題:

```
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()
```



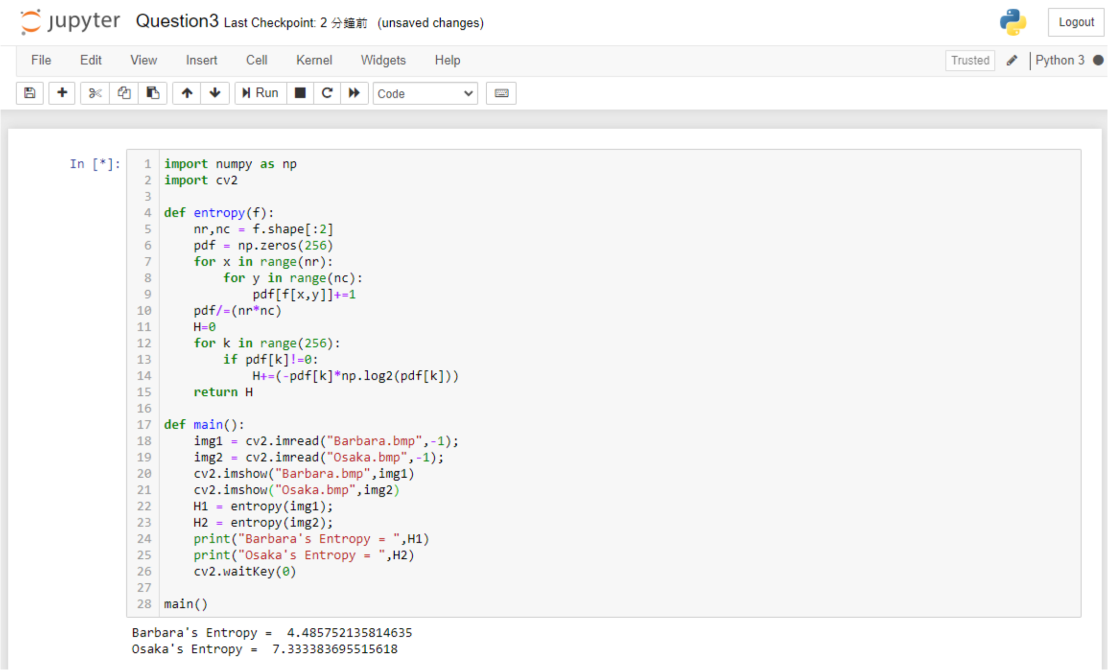


```
The PSNR =  10.611226669879223
```

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()
```



```python
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()
```

```
Barbara's Entropy =  4.485752135814635
Osaka's Entropy =  7.333383695515618
```

第四題:

CNN:

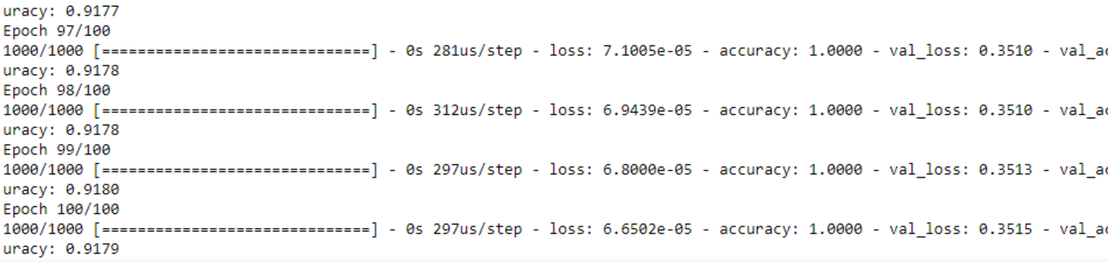
```
from keras.models import Sequential
from keras.layers.core import Dense, Dropout
from keras.layers import Conv2D, MaxPooling2D, Flatten
from keras.utils import np_utils
from keras.datasets import mnist

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'])
```

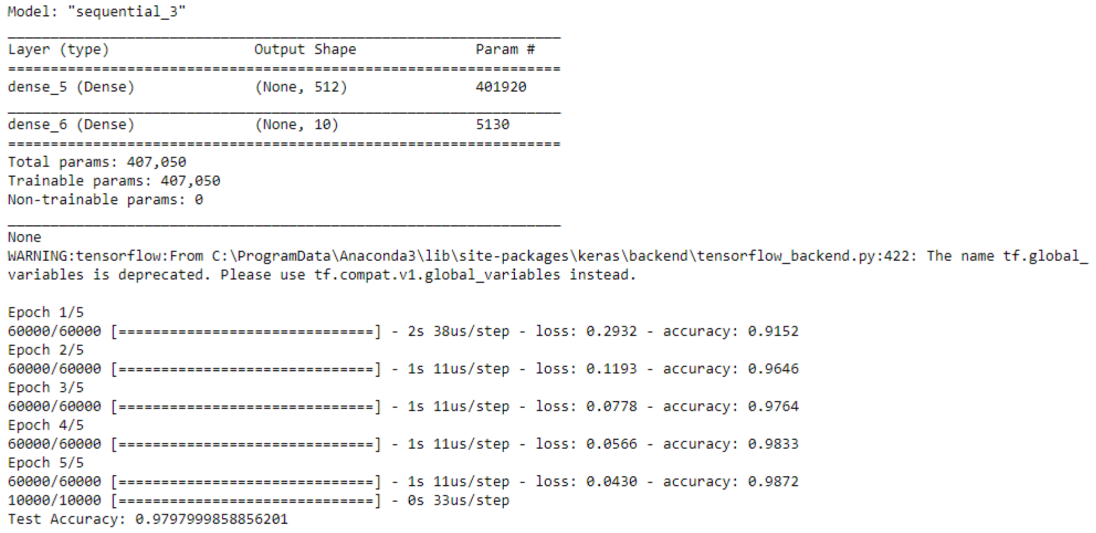
```
print(network.summary())

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:



```
uracy: 0.9177
Epoch 97/100
1000/1000 [==============================] - 0s 281us/step - loss: 7.1005e-05 - accuracy: 1.0000 - val_loss: 0.3510 - val_a
uracy: 0.9178
Epoch 98/100
1000/1000 [==============================] - 0s 312us/step - loss: 6.9439e-05 - accuracy: 1.0000 - val_loss: 0.3510 - val_a
uracy: 0.9178
Epoch 99/100
1000/1000 [==============================] - 0s 297us/step - loss: 6.8000e-05 - accuracy: 1.0000 - val_loss: 0.3513 - val_a
uracy: 0.9180
Epoch 100/100
1000/1000 [==============================] - 0s 297us/step - loss: 6.6502e-05 - accuracy: 1.0000 - val_loss: 0.3515 - val_a
uracy: 0.9179
```

ANN:



```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_5 (Dense)              (None, 512)               401920    
_________________________________________________________________
dense_6 (Dense)              (None, 10)                5130      
=================================================================
Total params: 407,050
Trainable params: 407,050
Non-trainable params: 0
_________________________________________________________________
None
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:422: The name tf.global_
variables is deprecated. Please use tf.compat.v1.global_variables instead.

Epoch 1/5
60000/60000 [==============================] - 2s 38us/step - loss: 0.2932 - accuracy: 0.9152
Epoch 2/5
60000/60000 [==============================] - 1s 11us/step - loss: 0.1193 - accuracy: 0.9646
Epoch 3/5
60000/60000 [==============================] - 1s 11us/step - loss: 0.0778 - accuracy: 0.9764
Epoch 4/5
60000/60000 [==============================] - 1s 11us/step - loss: 0.0566 - accuracy: 0.9833
Epoch 5/5
60000/60000 [==============================] - 1s 11us/step - loss: 0.0430 - accuracy: 0.9872
10000/10000 [==============================] - 0s 33us/step
Test Accuracy: 0.9797999858856201
```