dct

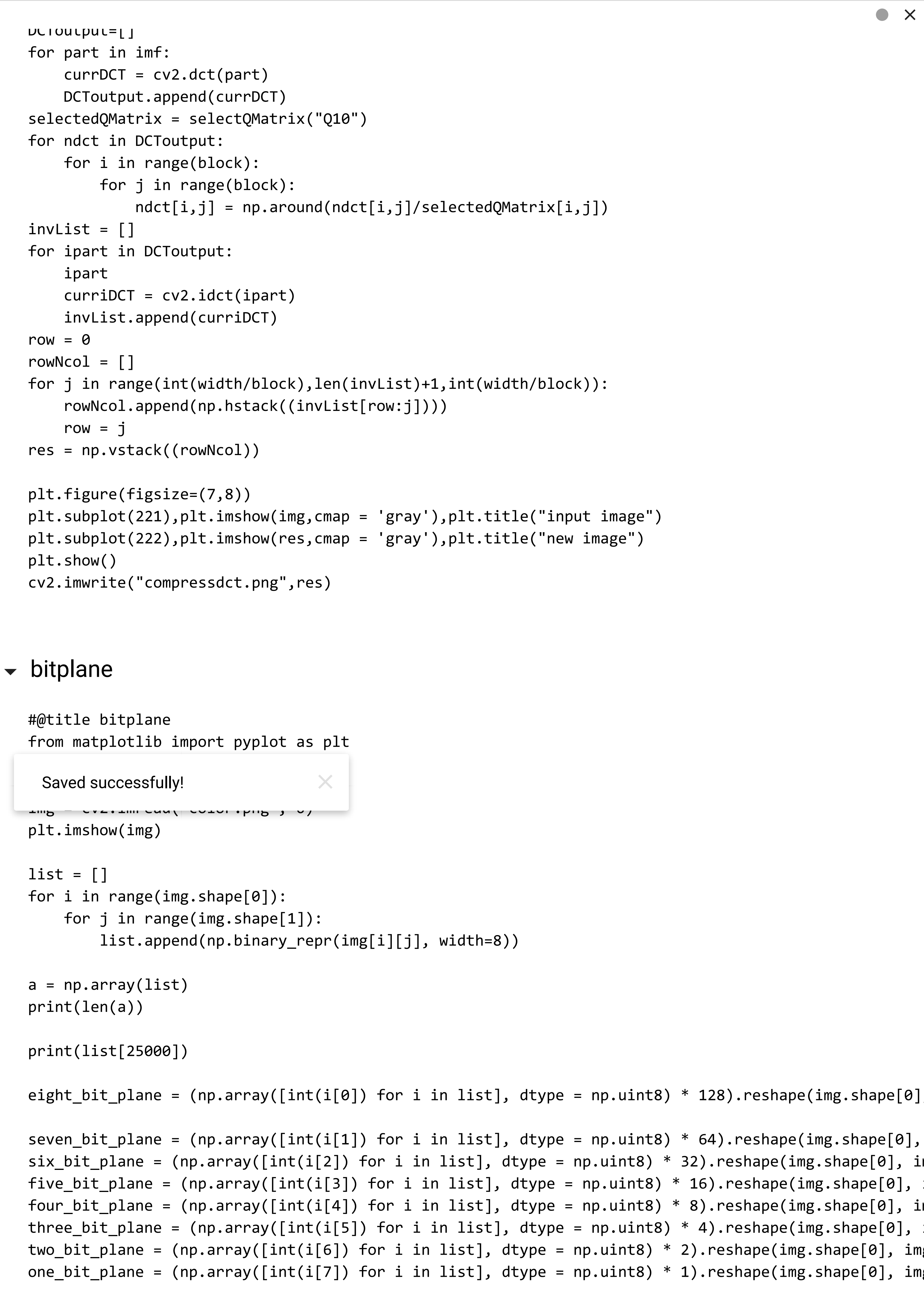
```
#@title dct

import numpy as np
import cv2
from matplotlib import pyplot as plt
img = cv2.imread("noise.png")
img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
def selectQMatrix(qName):
    Q10 = np.array([[80,60,50,80,120,200,255,255],
                    [55,60,70,95,130,255,255,255],
                    [70,65,80,120,200,255,255,255],
                    [70,85,110,145,255,255,255,255],
                    [90,110,185,255,255,255,255,255],
                    [120,175,255,255,255,255,255,255],
                    [245,255,255,255,255,255,255,255],
                    [255,255,255,255,255,255,255,255]])
    Q50 = np.array([[16,11,10,16,24,40,51,61],
                    [12,12,14,19,26,58,60,55],
                    [14,13,16,24,40,57,69,56],
                    [14,17,22,29,51,87,80,62],
                    [18,22,37,56,68,109,103,77],
                    [24,35,55,64,81,104,113,92],
                    [49,64,78,87,103,121,120,101],
                    [72,92,95,98,112,100,130,99]])
    Q90 = np.array([[3,2,2,3,5,8,10,12],
                    [2,2,3,4,5,12,12,11],
                    [3,3,3,5,8,11,14,11],
                    [3,3,4,6,10,17,16,12],
                    [4,4,7,11,14,22,21,15],
                    [5,7,11,13,16,12,23,18],
                                    24,24,21],
                                    20,20,20]])
```

Saved successfully!

```
        return Q10
    elif qName == "Q50":
        return Q50
    elif qName == "Q90":
        return Q90
    else:
        return np.ones((8,8))
height = len(img)
width = len(img[0])
sliced = []
block = 8
print("The image heigh is " +str(height)+", and image width is "+str(width)+" pixels")
currY = 0
for i in range(block,height+1,block):
    currX = 0
    for j in range(block,width+1,block):
        sliced.append(img[currY:i,currX:j]-np.ones((8,8))*128)
        currX = j
    currY = i

print("Size of the sliced image: "+str(len(sliced)))
print("Each elemend of sliced list contains a "+ str(sliced[0].shape)+ " element.")
```



```
DCToutput=[]
for part in imf:
    currDCT = cv2.dct(part)
    DCToutput.append(currDCT)
selectedQMatrix = selectQMatrix("Q10")
for ndct in DCToutput:
    for i in range(block):
        for j in range(block):
            ndct[i,j] = np.around(ndct[i,j]/selectedQMatrix[i,j])
invList = []
for ipart in DCToutput:
    ipart
    curriDCT = cv2.idct(ipart)
    invList.append(curriDCT)
row = 0
rowNcol = []
for j in range(int(width/block),len(invList)+1,int(width/block)):
    rowNcol.append(np.hstack((invList[row:j])))
    row = j
res = np.vstack((rowNcol))

plt.figure(figsize=(7,8))
plt.subplot(221),plt.imshow(img,cmap = 'gray'),plt.title("input image")
plt.subplot(222),plt.imshow(res,cmap = 'gray'),plt.title("new image")
plt.show()
cv2.imwrite("compressdct.png",res)
```

bitplane

```
#@title bitplane
from matplotlib import pyplot as plt
```

Saved successfully!

```
img = cv2.imread("color.png",0)
plt.imshow(img)

list = []
for i in range(img.shape[0]):
    for j in range(img.shape[1]):
        list.append(np.binary_repr(img[i][j], width=8))

a = np.array(list)
print(len(a))

print(list[25000])

eight_bit_plane = (np.array([int(i[0]) for i in list], dtype = np.uint8) * 128).reshape(img.shape[0]

seven_bit_plane = (np.array([int(i[1]) for i in list], dtype = np.uint8) * 64).reshape(img.shape[0],
six_bit_plane = (np.array([int(i[2]) for i in list], dtype = np.uint8) * 32).reshape(img.shape[0], i
five_bit_plane = (np.array([int(i[3]) for i in list], dtype = np.uint8) * 16).reshape(img.shape[0],
four_bit_plane = (np.array([int(i[4]) for i in list], dtype = np.uint8) * 8).reshape(img.shape[0], i
three_bit_plane = (np.array([int(i[5]) for i in list], dtype = np.uint8) * 4).reshape(img.shape[0],
two_bit_plane = (np.array([int(i[6]) for i in list], dtype = np.uint8) * 2).reshape(img.shape[0], im
one_bit_plane = (np.array([int(i[7]) for i in list], dtype = np.uint8) * 1).reshape(img.shape[0], im
plt.figure(figsize=(7,8))
plt.subplot(441),plt.imshow(one_bit_plane,cmap = 'gray'),plt.title("one bit")
plt.subplot(442),plt.imshow(two_bit_plane,cmap = 'gray'),plt.title("two bit")
plt.subplot(443),plt.imshow(three_bit_plane,cmap = 'gray'),plt.title("three bit")
plt.subplot(444),plt.imshow(four_bit_plane,cmap = 'gray'),plt.title("four bit")
plt.subplot(445),plt.imshow(five_bit_plane,cmap = 'gray'),plt.title("five bit")
plt.subplot(446),plt.imshow(six_bit_plane,cmap = 'gray'),plt.title("six bit")
plt.subplot(447),plt.imshow(seven_bit_plane,cmap = 'gray'),plt.title("seven bit")
plt.subplot(448),plt.imshow(eight_bit_plane,cmap = 'gray'),plt.title("eight bit")
plt.show()

new_img = eight_bit_plane + seven_bit_plane + six_bit_plane

plt.figure(figsize=(7,8))
plt.subplot(221),plt.imshow(img,cmap = 'gray'),plt.title("input image")
plt.subplot(222),plt.imshow(new_img,cmap = 'gray'),plt.title("new image")
plt.show()
```

all histogram

```
#@title all histogram


import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread("gray1.jpg",0)
hist,bins = np.histogram(img.ravel(),256,[0,255])
plt.xlim([0,255])
plt.plot(hist)
```

Saved successfully!

```
threshold = 115

#SIMPLE THRESHOLDING
r, final = cv2.threshold(img, threshold, 255, cv2.THRESH_BINARY)
cv2.imshow("SIMPLE THESH_BINARY", final)
cv2.waitKey(0)
cv2.destroyAllWindows();

r, final = cv2.threshold(img, threshold, 255, cv2.THRESH_BINARY_INV)
cv2.imshow("SIMPLE THRESH_BINARY_INV", final)
cv2.waitKey(0)
cv2.destroyAllWindows();

r, final = cv2.threshold(img, threshold, 255, cv2.THRESH_TOZERO)
cv2.imshow("SIMPLE THRESH_TOZERO", final)
cv2.waitKey(0)
cv2.destroyAllWindows();



r, final = cv2.threshold(img, threshold, 255, cv2.THRESH_TOZERO_INV)
cv2.imshow("SIMPLE THRESH_TOZERO_INV", final)
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()

#OSTU THRESHOLDING
r,final=cv2.threshold(img,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
cv2.imshow("OSTU NORMAL", final)
cv2.waitKey(0)
cv2.destroyAllWindows();

blurred = cv2.GaussianBlur(img,(5,5),0)
r,final = cv2.threshold(blurred,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
cv2.imshow("OSTU GAUSSIAN", final)
cv2.waitKey(0)
cv2.destroyAllWindows()


#ADAPTIVE THRESHOLDING
inal = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY,11,2)
cv2.imshow("ADAPTIVE MEAN", final)
cv2.waitKey(0)
cv2.destroyAllWindows()

final = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY,11,2)
cv2.imshow("ADAPTIVE GAUSSIAN", final)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

robert sobel perwit

```
#@title robert sobel perwit

#robert
```

Saved successfully! ×

```
img = cv2.imread("noise.png",0).astype('float64');
img/=255

mask_x = np.array([[1,0],[0,-1]]);
mask_y = np.array([[0,1],[-1,0]]);

robert_x = cv2.filter2D(img,-1,mask_x);
robert_y = cv2.filter2D(img,-1,mask_y);

robert = np.sqrt(np.square(robert_x) + np.square(robert_y))
robert*=255

cv2.imwrite("robert.jpg",robert)

#sobel

import cv2;
import numpy as np

img = cv2.imread("noise.png",0).astype('float64');
img/=255
```

```
mask_x = np.array([[-1,0,2],[-2,0,2],[-1,0,1]]);
mask_y = np.array([[1,2,1],[0,0,0],[-1,-2,-1]]);

sobel_x = cv2.filter2D(img,-1,mask_x);
sobel_y = cv2.filter2D(img,-1,mask_y);

sobel = np.sqrt(np.square(sobel_x) + np.square(sobel_y))
sobel*=255

cv2.imwrite("sobel.jpg",sobel)

#perwitt
import cv2;
import numpy as np


img = cv2.imread("noise.png",0).astype('float64');
img/=255

mask_x = np.array([[1,0,-1],[1,0,-1],[1,0,-1]]);
mask_y = np.array([[1,1,1],[0,0,0],[-1,-1,-1]]);

perwitt_x = cv2.filter2D(img,-1,mask_x);
perwitt_y = cv2.filter2D(img,-1,mask_y);

perwitt = np.sqrt(np.square(perwitt_x) + np.square(perwitt_y))
perwitt*=255

cv2.imwrite("perwitt.jpg",perwitt)

#point
import cv2;
import numpy as np
```

Saved successfully!

```
mask_p = np.array([[-1,-1,-1],[-1,8,-1],[-1,-1,-1]]);
point = cv2.filter2D(img,-1,mask_p)
point[point>t]=1
point[point<t]=0
cv2.imshow("img",point)
cv2.waitKey(0)

#line
import cv2;
import numpy as np

mask_h = np.array([[-1,-1,-1],[2,2,2],[-1,-1,-1]]);
mask_v = np.array([[-1,2,-1],[-1,2,-1],[-1,2,-1]]);
mask_45 = np.array([[2,-1,-1],[-1,2,-1],[-1,-1,2]]);
mask_45n = np.array([[-1,-1,2],[-1,2,-1],[2,-1,-1]]);

img = cv2.imread("noise.png",0).astype('float64');
img/=255

line_h = cv2.filer2D(img,-1,mask_h)
line_h*=255
cv2.imwrite("line_h.jpg",line_h)
```

```
line_v = cv2.filer2D(img,-1,mask_v)
line_v*=255
cv2.imwrite("line_v.jpg",line_v)

line_45 = cv2.filer2D(img,-1,mask_45)
line_45*=255
cv2.imwrite("line_45.jpg",line_45)

line_45n = cv2.filer2D(img,-1,mask_45n)
line_45n*=255
cv2.imwrite("line_45n.jpg",line_45n)
```