Assignment 1

How to run the code
1.3) Q1_part3.ipynb
1.4) python ./PS0Q1.py (run the script)
2.) python ./PS0Q2.py (run the script)

Question 1

Question 1:

- Went through the documentation

Question 2:

Describe the following commands

```
a.) x = np.random.permutation(1000)
# 1000 random numbers  0-999 will be generated in random order

b.) a = np.array([1,2,3], [4,5,6], [7,8,9])
# 3x3 matrix will be created whose 1st row is 1,2,3 and 2nd row is 4,5,6 and 3rd row i
s 7,8,9
```

```
c.) a[2,:]
# 3rd row will be printed [7,8,9]

d.) f = np.random.randn(5,1) # 5x1 matrix will be created whose elements are random nu
mbers from normal distribution with mean 0 and variance 1
 f[f>0] # all the elements of f which are greater than 0 will be printed

e.)
x = np.zeros(10) + 0.5 #create an array of 10 zeros and add 0.5 to each element
y = 0.5*np.ones(len(x)) #create an array of 10 ones and multiply each element by 0.5
z = x+y #add the two arrays together  element by element resulting in an array of 10
 1's


f.) a = np.arange(1,100) # the numbers from 1 to 99 will be stored in a with a step si
ze of 1 and total length of a will be 99 with minimum value 1 and maximum value 99
b =a[::-1] #reverse the array from the last element to the first element starting from
99 to 1 in steps of -1
```

💡 Q1_part3.ipynb file has the code for Question 1 part 3

Question 3a:

Use numpy.random.rand to return the roll of a six sided die over N trials

```
def roll_dice(val):
    if val >= 0 and val < .166:
        return 1
    elif val >= .166 and val < .333:
        return 2
    elif val >= .333 and val < .5:
        return 3
    elif val >= .5 and val < .666:
        return 4
    elif val >= .666 and val < .833:
        return 5
    else:
        return 6

N = 1000
rolls = []
for i in range(N):
         rolls.append(roll_dice(np.random.rand()))
```

Question 3b:

```
y = np.array([1, 2, 3, 4, 5, 6])
z = y.reshape(3,2)
z

Ans:
array([[1, 2],
       [3, 4],
       [5, 6]])
```

Question 3c:

```
np.max(z) #Ans =6

r = np.where(z == np.max(z))[0][0]
c = np.where(z == np.max(z))[1][0]

#ans: (2, 1)
```

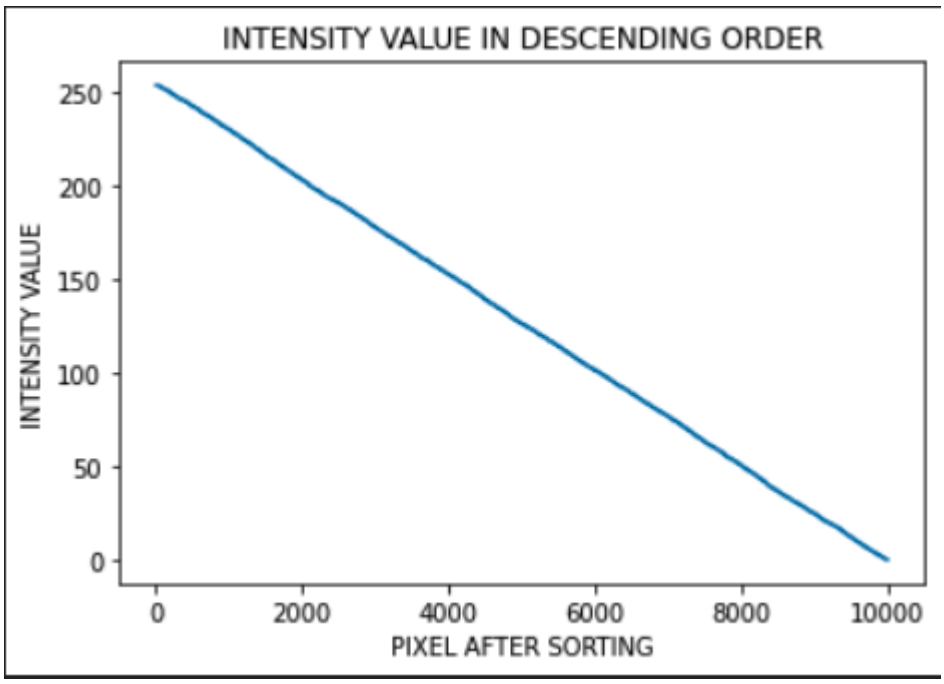
Question 3d:

```
v = np.array([1,8, 8,2,1,3,9,8])
x = len(np.where(v == 1)[0])
x

#ans 2
```

Question 4a : Saved in script `PS0Q1.py`

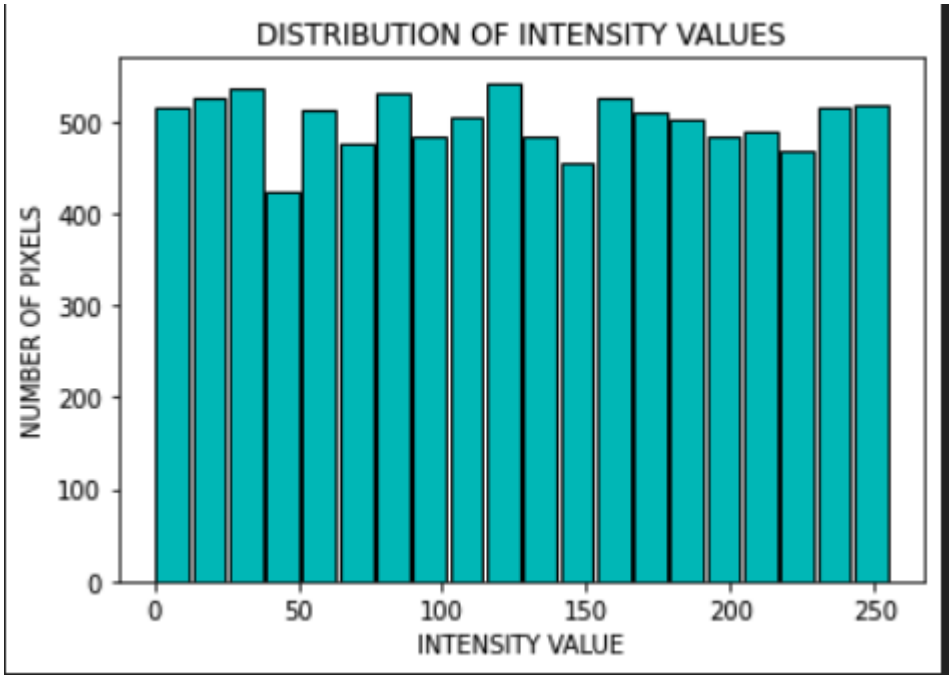
```
#create a 100 X 100 matrix
x = np.random.randint(0, 255, (100, 100))

#save the matrix to a file in npy format
np.save('inputAPS0Q1.npy', x)

#load the matrix from the file
A = np.load('inputAPS0Q1.npy')

#plot the intensity value in descending order

plt.plot(np.sort(A.flatten())[::-1])

#plot the intensity value in descending order

plt.plot(np.sort(A.flatten())[::-1])

#LABEL THE AXES
plt.xlabel('PIXEL AFTER SORTING')
```

```
plt.ylabel('INTENSITY VALUE')
#TITLE THE PLOT
plt.title('INTENSITY VALUE IN DESCENDING ORDER')
```



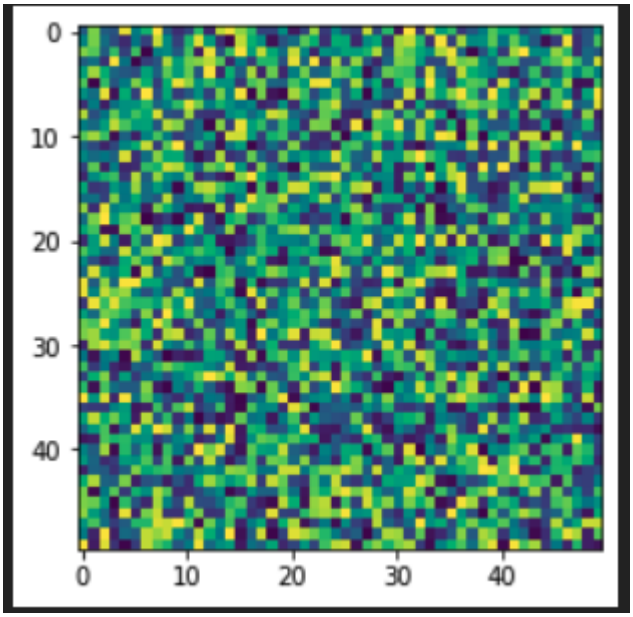
Question 4b

```
#plot the distribution of intensity values

plt.hist(A.flatten(), bins=20, range=(0, 255), color='c', edgecolor='k', linewidth=1.
0, rwidth=0.9)

#LABEL THE AXES
plt.xlabel('INTENSITY VALUE')
plt.ylabel('NUMBER OF PIXELS')
#TITLE THE PLOT
plt.title('DISTRIBUTION OF INTENSITY VALUES')
```



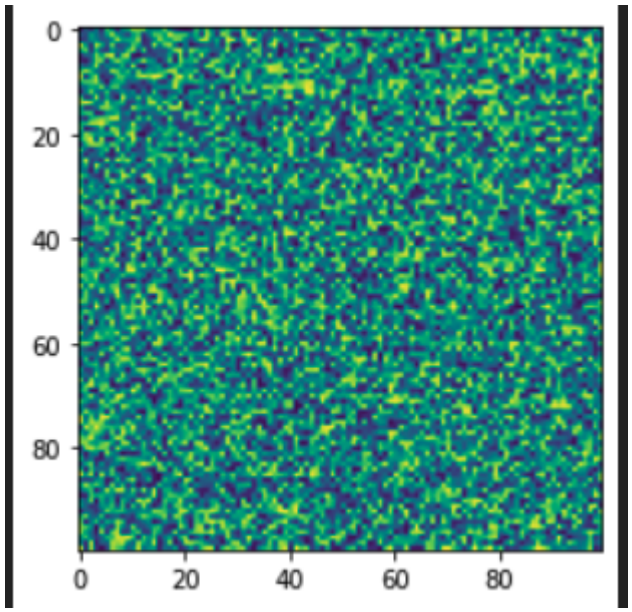
Question 4c

```
# create bottom left quadrant of A
X = A[50:, 0:50]
X.shape

#plot X as an image
plt.imshow(X)

#save X  in npy format
np.save('outputXPS0Q1.npy', X)
```



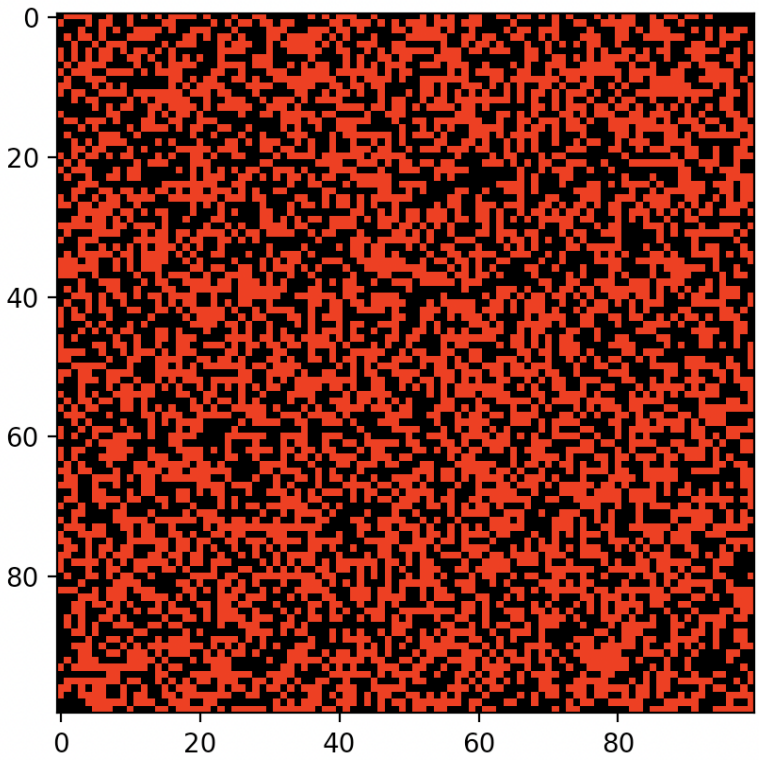
Question 4d

```
#Subtract mean of A from each element of A

Y = A - np.mean(A)

#plot Y as an image
plt.imshow(Y)

#save Y  in npy format
np.save('outputYPS0Q1.npy', Y)
```



Question 4e

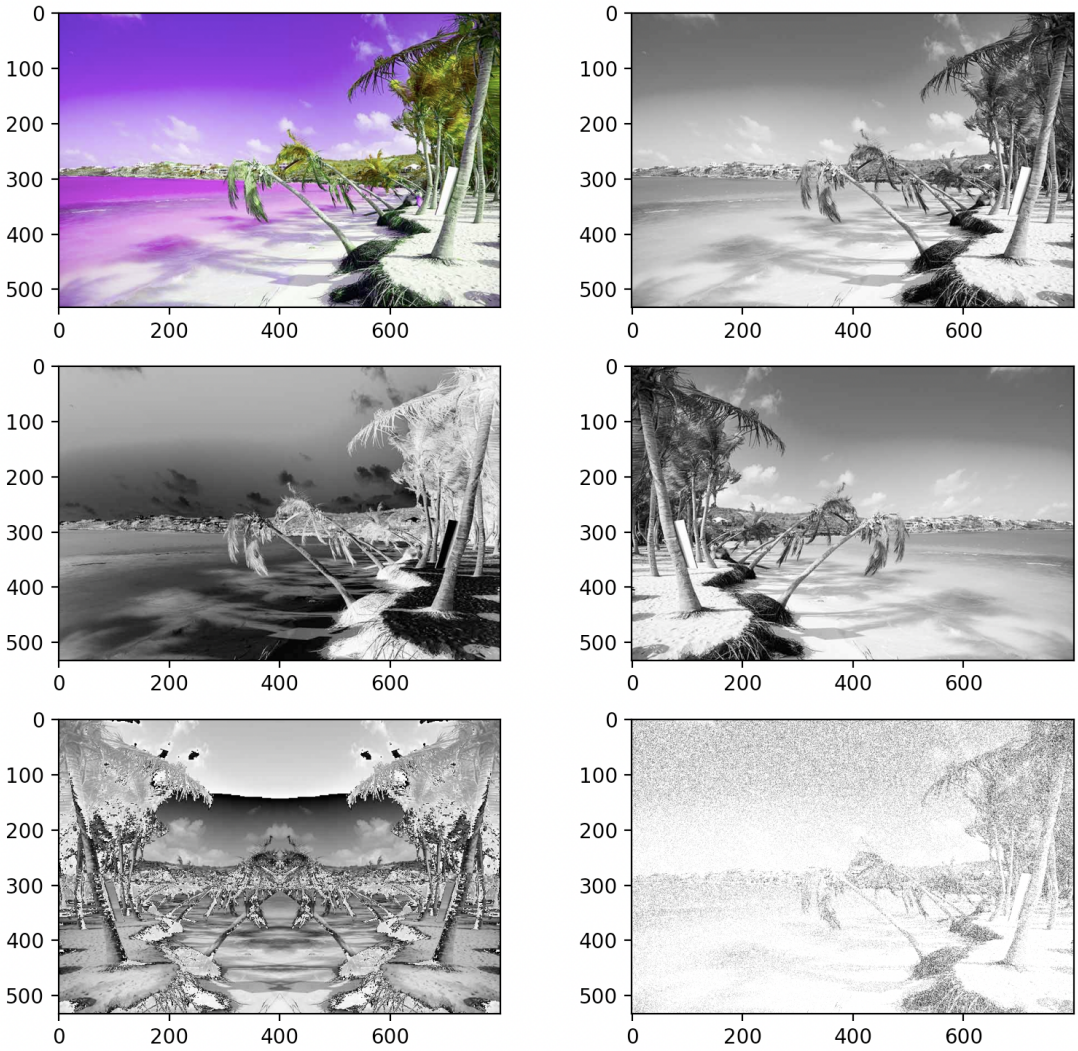
```
Z = np.zeros((100,100, 3))  # create a 100 X 100 X 3 matrix
t = np.mean(A)

# set the color channel to only red where the intensity is greater than the mean inten
sity of the image in every channel


Z[A > t] = [1, 0, 0]

Z[A <= t] = [0, 0, 0]

plt.imshow(Z)

#plot Z as an image
plt.show()

#save Z  image in png format
plt.imsave('outputZPS0Q1.png', Z)
```



Question 2 : `Saved in script PS0Q2.py`

Original Image

Transformed Images


0
100
200
300
400
500
0
200
400
600