**Task 1 research Document**

**Extracting data:**

When I first started the task I was thinking of storing the image as a hex value as many images can be viewed in hex notation. However I have no idea how to do that in python.

So I started by using some basic python graphic library of turtle. In turtle I made a simple program that can draw a rectangle of any size using a function called coordinate generator which makes an array and appends tuples to it.

Using the turtle library documentation, we can also change the RGB colour values.

Here is the snippet of code that is responsible for the generation of the array.

Import turtle

def cordinates\_generator(x,y):

L = []

for i in range(y+1):

for j in range(x+1):

L.append((i,j))

return L

x = cordinates\_generator(50,50)

for i in x:

turtle.goto(i[0],i[1])

turtle.Screen()

From here I went to research on some more powerful image processing libraries in python. I came across Pillow as the fork of the discontinued Python Image Library. This library is able to process the image such that they will show the binary bit map of the image, giving us something to work from.

One of the few thing I came across is the idea of an image histogram. This image histogram is useful in highlighting the areas were the image is brighter and helps to construct an idea of what the image will be like.

Another area that I came across is the .getdata() method in the image class. This method extracts the RGB value of every pixel in the image. This can help to construct the colour values of the image back to where we would like.

The getdata() method is useful but it lacks the coordinates that we need to generate the new image.

Instead I looped though every pixel of the image and stored its data in the form of

(x-coordinate, y-coordinate ,redVal,greenVal,blueVal). This helps to reconstruct the image later and we can save this information simply to a text file which can be utilised by the second part of the programme

**Constructing the image:**

One way I came across the construction of an image is to use turtle graphic library in python. However, this is quite slow as it is mostly an O(n2) operation. Hence I went around to see if there are other ways to reproduce an image from it hex values and other data we extracted.

Another way to reproduce the image will be to again use PIL. By using .new() and assigning the data back into every pixel we can redraw the image. This is also a O(n2 ) operation. However this feels a lot faster and I am still unsure why.

This uses the load() method and just reassigns the new image the RGB data of the other image or anything that we input as a file.

Points for further research:

How do we use other ways of image representation such as hex value to generate the image?