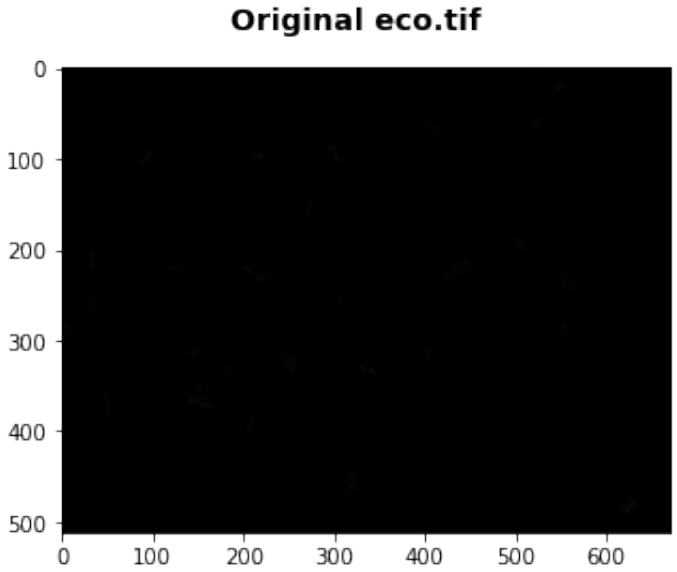
In [1]:

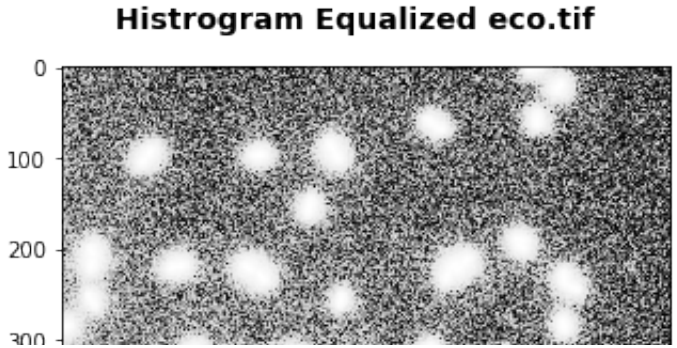
```
import numpy
from scipy import misc
import matplotlib.pyplot as plt
import copy
import imageio
# misc has no attribute imread, use imageio instead
```

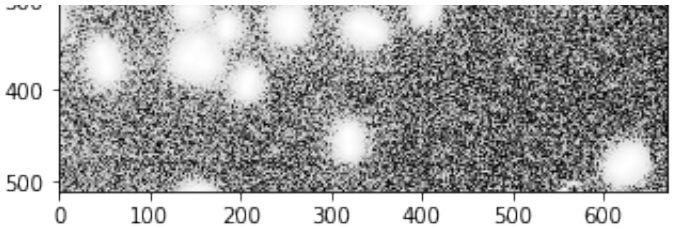
In [14]:
```
# ECE420 - Spring2017
# Lab6 - Part 1: Histogram Equilization


# Implement This Function
def histeq(pic):
    # Follow the procedures of Histogram Equalizaion
    # Modify the pixel value of pic directly

    L = 2**16 # uint16
    M,N = pic.shape

    hist = numpy.zeros(L)
    for r in range(M):
        for c in range(N):
            hist[pic[r,c]] += 1

    cdfmin = 0
    cdf = numpy.zeros(L)
    cdf[0] = hist[0]
    for i in range(1, L):
        cdf[i] = cdf[i-1] + hist[i]
        if cdf[i-1] == 0 and cdf[i] != 0:
            cdfmin = cdf[i]
    for r in range(M):
        for c in range(N):
            pic[r,c] = int( (cdf[pic[r,c]] - cdfmin) * (L-1) / (M*N-1) )

    return pic;
```

In [15]:
```
# Histogram Equilization
eco_origin = imageio.imread('eco.tif');
eco_histeq = copy.deepcopy(eco_origin);
# Call to histeq to perform Histogram Equilization
eco_histeq = histeq(eco_histeq);
```

```
eco_histeq = histeq(eco_histeq);
# Show the result in two windows
fig_eco_origin = plt.figure(1);
fig_eco_origin.suptitle('Original eco.tif', fontsize=14, fontweight='bold');
plt.imshow(eco_origin,cmap='gray',vmin = 0, vmax = 65535);
fig_eco_histeq = plt.figure(2)
fig_eco_histeq.suptitle('Histrogram Equalized eco.tif', fontsize=14, fontweight='bold');
plt.imshow(eco_histeq,cmap='gray',vmin = 0, vmax = 65535);
plt.show()
```






400
500
0
100
200
300
400
500
600

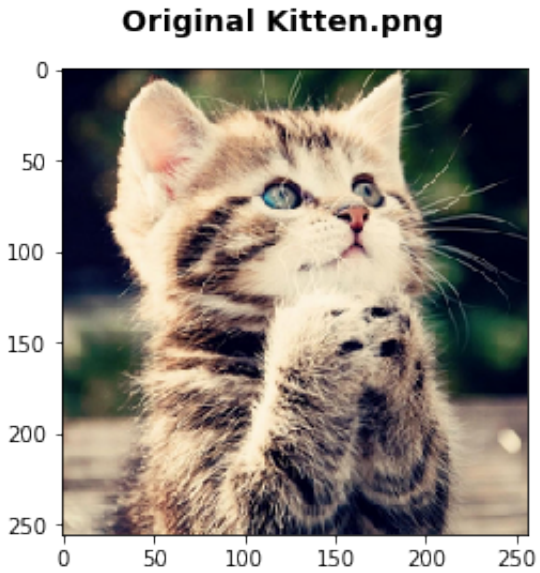
In [8]:
```
# ECE420 - Spring2017
# Lab6 - Part 2: 2D-Convolution

# Function Definition Here

# Implement This funtion
def conv2(pic,kernel):
    # Create a new pic with same size but float type
    pic_conv = numpy.zeros(numpy.shape(pic))
    # Perform 2-D Convolution with the given kernel

    row, col, channel = pic.shape
    ii, jj = kernel.shape
    h = int(ii/2)  # half the size of kernel

    for ch in range(channel):
        for r in range(row):
            for c in range(col):

                for i in range(ii):
                    for j in range(jj):
                        if (r-h+i<0) or (r-h+i>=row) or (c-h+j<0) or (c-h+j>=col):
                            continue
                        pic_conv[r,c,ch] += kernel[ii-i-1, jj-j-1] * pic[r-h+i, c-h+j, ch]

                #pic_conv[r,c,ch] = int(pic_conv[r,c,ch])
    return pic_conv.astype('uint8');


```

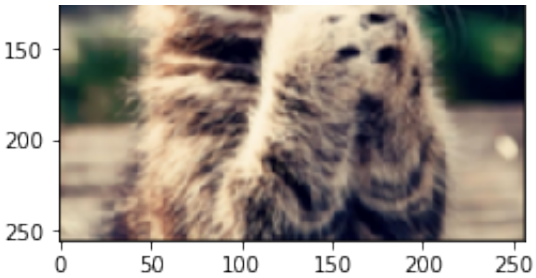
In [9]:
```
# Gaussian Kernel Following the Descriptiong: http://www.mathworks.com/help/images/ref/fspecial.html
def gengaussian(size=5,sigma=3.0):
    if size%2==0 or size<2:
        print('Size Not Valid');
        return None;
```

```
        return None;
    kernel = numpy.zeros((size,size));
    for x in range(size):
        for y in range(size):
            kernel[x][y] = numpy.exp(-((x-(size-1)/2)**2+(y-(size-1)/2)**2)/(2*sigma**2));
    kernel = kernel / numpy.sum(kernel);
    return kernel

# Edge Detection Kernel Source:https://alwaysbusycorner.com/2011/12/02/realbasic-canvas-tutorial-les
def genxkernel(flag=1):
    if flag == 1:
        kernel = numpy.array([[-1,0,1]]*3);
    else:
        kernel = numpy.array([[-1,0,1],[-2,0,-2],[-1,0,-1]]);
    return kernel

def genykernel(flag=1):
    if flag == 1:
        kernel = numpy.array([[-1,-1,-1],[0,0,0],[1,1,1]]);
    else:
        kernel = numpy.array([[-1,-2,-1],[0,0,0],[1,2,1]]);
    return kernel

# Merge Detected X-Edge and Y-Edge
def merge(picx,picy):
    picshape = numpy.shape(picx);
    if picshape != numpy.shape(picy):
        print('Pic Size Not Matched!');
        return picx;
    picmerge = numpy.zeros(picshape);
    for row in range(picshape[0]):
        for col in range(picshape[1]):
            for channel in range(picshape[2]):
                picmerge[row][col][channel] = numpy.sqrt((picx[row][col][channel]**2+picy[row][col]|
    picmerge = picmerge.astype(picx.dtype,copy=False);
    return picmerge;
```

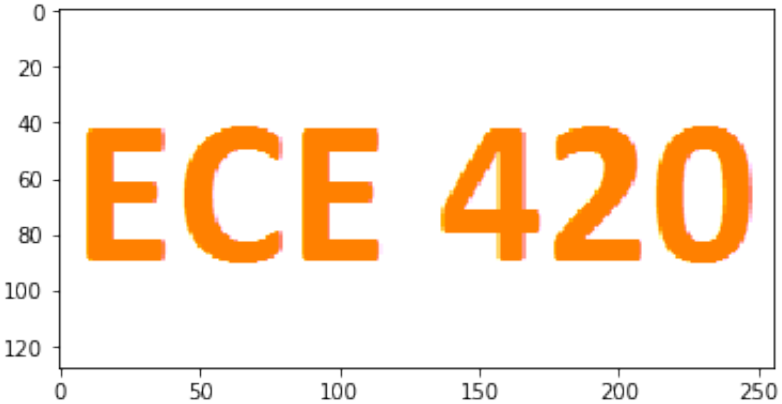
```
41
42 # Main Function Starts Here!!!
43 # Gaussian Blur Kernel
44 # Read Image and Display
45 kitten_origin = imageio.imread('kitten.png');
46 fig_kitten_origin = plt.figure(1);
47 fig_kitten_origin.suptitle('Original Kitten.png', fontsize=14, fontweight='bold');
48 plt.imshow(kitten_origin,vmin = 0, vmax = 255);
49 plt.show(block=False);
50 # Generate Kernel
51 kernel_blur = gengaussian(3);
52 # Apply Convolution
53 kitten_blur = conv2(kitten_origin,kernel_blur)
54 # Display Results
55 fig_kitten_blur = plt.figure(2);
56 fig_kitten_blur.suptitle('Blurred Kitten.png', fontsize=14, fontweight='bold');
57 plt.imshow(kitten_blur,vmin = 0, vmax = 255);
58 plt.show(block=False);
59
60 # Edge Detection Kernel
61 # Read Image and Display
62 logo_origin = imageio.imread('logo.png');
63 fig_logo_origin = plt.figure(3);
64 fig_logo_origin.suptitle('Original Logo.png', fontsize=14, fontweight='bold');
65 plt.imshow(logo_origin,vmin = 0, vmax = 255);
66 plt.show(block=False);
67 # X-Edge Detection
68 kernel_xedge = genxkernel();
69 logo_xedge = conv2(logo_origin,kernel_xedge)
70 fig_logo_xedge = plt.figure(4);
71 fig_logo_xedge.suptitle('X-Edge Detected Logo.png', fontsize=14, fontweight='bold');
72 plt.imshow(logo_xedge,vmin = 0, vmax = 255);
73 plt.show(block=False);
74 # Y-Edge Detection
75 kernel_yedge = genykernel();
76 logo_yedge = conv2(logo_origin,kernel_yedge)
```

```
fig_logo_yedge = plt.figure(5);
fig_logo_yedge.suptitle('Y-Edge Detected Logo.png', fontsize=14, fontweight='bold');
plt.imshow(logo_yedge,vmin = 0, vmax = 255);
plt.show(block=False);
# Merge Edges
logo_fulledge = merge(logo_xedge,logo_yedge);
fig_logo_fulledge = plt.figure(6);
fig_logo_fulledge.suptitle('Full-Edge Detected Logo.png', fontsize=14, fontweight='bold');
plt.imshow(logo_fulledge,vmin = 0, vmax = 255);
plt.show();
```

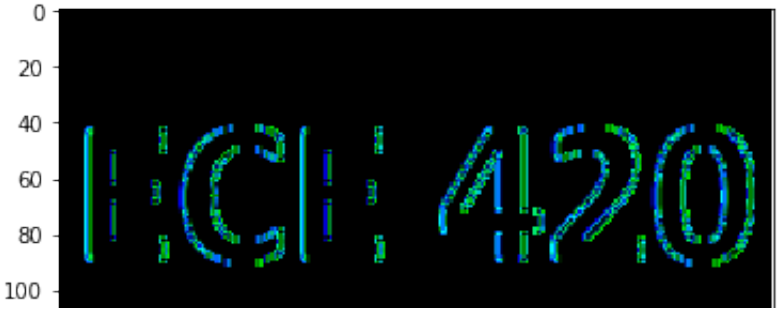


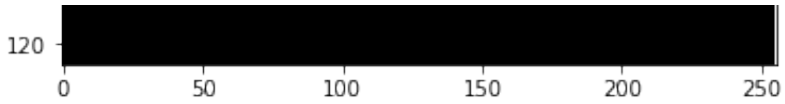


Original Logo.png

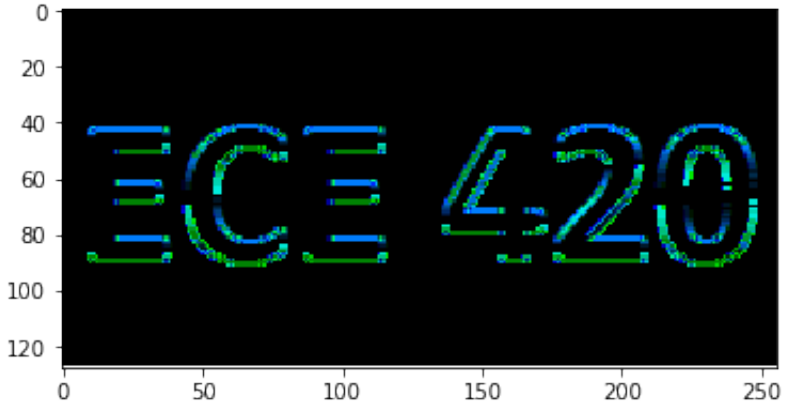


X-Edge Detected Logo.png





Y-Edge Detected Logo.png



Full-Edge Detected Logo.png