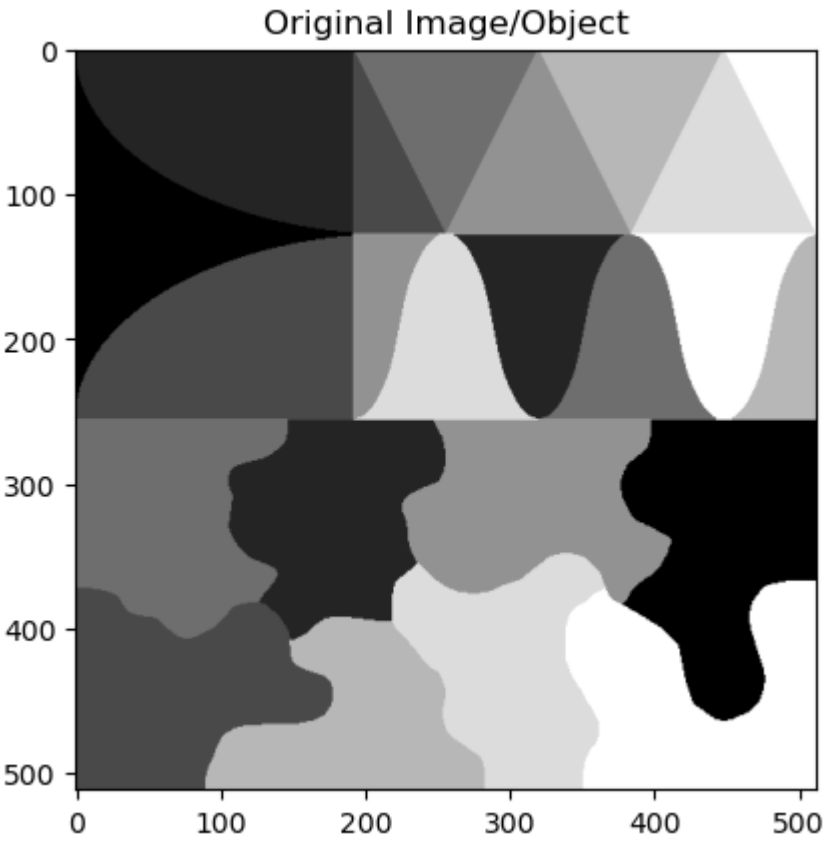
Homework 1 - Angular Spectrum

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color
```

Importing and scaling the object (image)

In [2]:
```
g = io.imread("texmos3.s512.tiff") # Imports the image

if g.ndim == 3:
    g = color.rgb2gray(g) # If the image is in rgb format, converts the image fr

beta = np.pi/(2*255) # Scaling factor
g = g*beta  # Scales the pixel valus of the image in range [0, pi/2]

# Display the image
plt.figure()
plt.imshow(g, cmap = "gray")
plt.title("Original Image/Object")
plt.show()
```



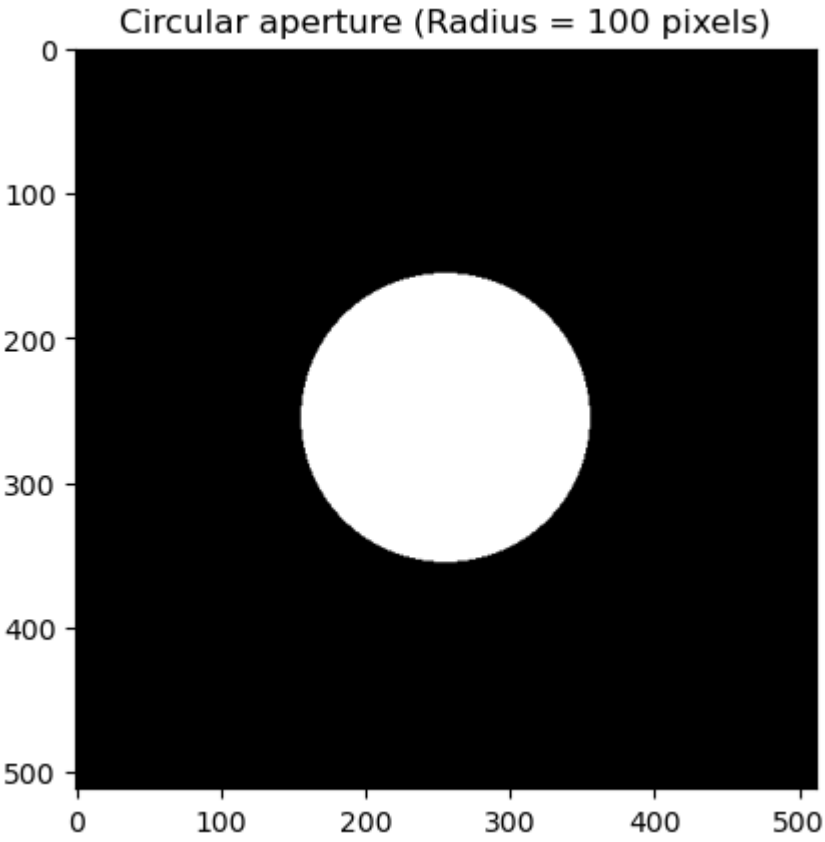
Defining Parameters

In [3]:
```
N = g.shape[0] # stores the height of the image as N
p = 5e-6 # pixel size
wavelength = 0.5e-6 # wavelength
k = 2*np.pi/wavelength # wave number
z = 100e-6 # propagation distance
```

```
# Creating a meshgrid
x0 = np.linspace(-N/2, N/2, N, endpoint = True)
x, y = np.meshgrid(x0, x0)
x = x*p
y = y*p
f0 = x0/N
fx, fy = np.meshgrid(f0, f0)
fx = fx/p
fy = fy/p
```

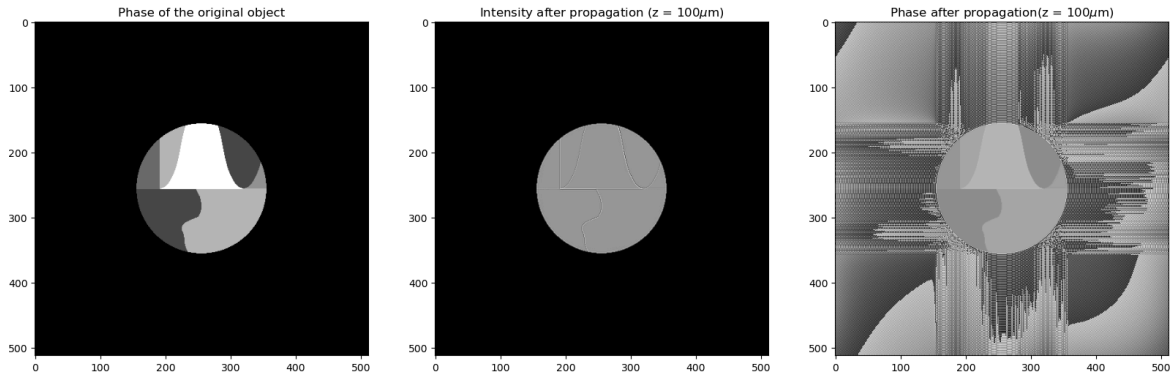
Making a circular aperture

In [4]:
```
circ = np.zeros((N,N))
circ[x**2 + y**2 <= (100*p)**2] = 1 # creates a circular aperture of radius 100
plt.figure()
plt.imshow(circ, cmap = "gray")
plt.title("Circular aperture (Radius = 100 pixels)")
plt.show()
```



In [5]:
```
# Turning the image into a phase object
u = np.exp(1j*g)*circ

# Defining parameters for propagation
alpha = np.sqrt(k**2 - 4*np.pi**2*(fx**2 + fy**2))
H = np.exp(1j*alpha*z) # Transfer function

# Low pass filter in the fourier domain for proper sampling
lp_f = (1/wavelength) * 1 / (np.sqrt(1 + (2 * z / (N * p))**2))
filtr = np.zeros((N, N))
filtr[(fx**2 + fy**2 < lp_f**2)] = 1

# Fourier transform the object
```

```
U = np.fft.fftshift(np.fft.fft2(np.fft.ifftshift(u)))

# Apply filter and transfer function
U_prop = U*H*filtr # Fourier spectrum of the field at distance z

# Inverse transform to get the field at distance z
U_out = np.fft.fftshift(np.fft.ifft2(np.fft.ifftshift(U_prop)))
Intensity = np.abs(U_out)**2 # Intensity of the propagated field

# Display the results
plt.figure(figsize = (20,10))
plt.subplot(1,3,1)
plt.imshow(np.angle(u), cmap = "gray")
plt.title("Phase of the original object")
plt.subplot(1,3,2)
plt.imshow(Intensity, cmap = "gray")
plt.title(r"Intensity after propagation (z = 100$\mu$m)")
plt.subplot(1,3,3)
plt.imshow(np.angle(U_out), cmap = "gray")
plt.title(r"Phase after propagation(z = 100$\mu$m)")
plt.show()
```

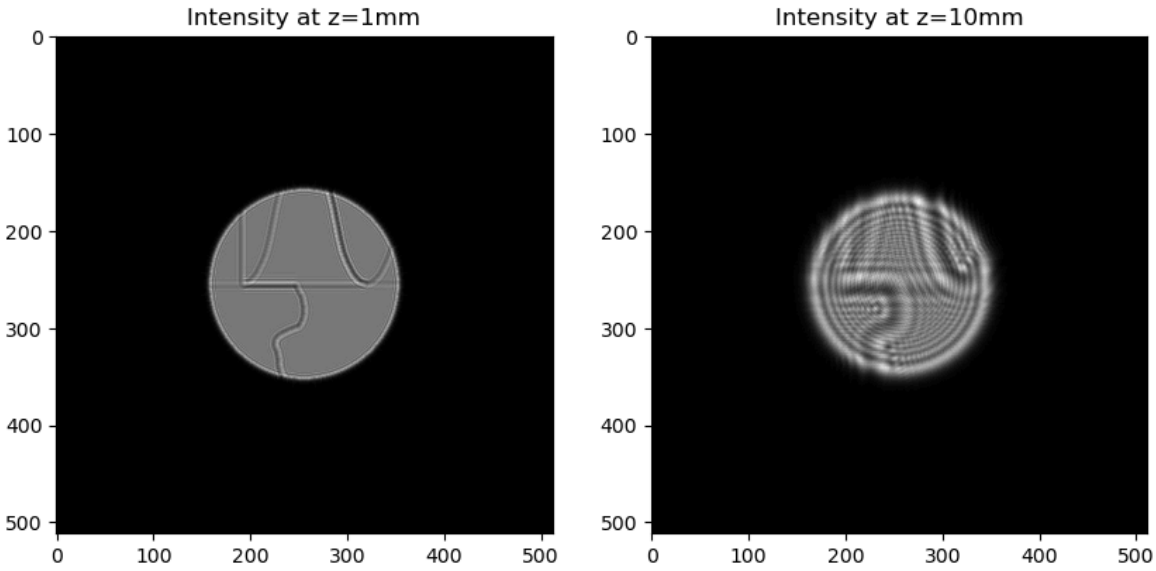


In the image of intensity of the field after propagating 100μm distance, we see very few details. This can be due to the propagation distance being too small such that the frequencies captured are limited. We can increase the propagation distance and check whether there is any improvement in the result.

Increasing the propagation distance

In [6]:
```
z1 = 1e-3 # Propagation distance is 1mm
H1 = np.exp(1j*alpha*z1)
U_z1 = U*H1*filtr
U_out_z1 = np.fft.fftshift(np.fft.ifft2(np.fft.ifftshift(U_z1)))
Intensity_z1 = np.abs(U_out_z1)**2

z2 = 10e-3 # Propagation distance is 10mm
H2 = np.exp(1j*alpha*z2)
U_z2 = U*H2*filtr
U_out_z2 = np.fft.fftshift(np.fft.ifft2(np.fft.ifftshift(U_z2)))
Intensity_z2 = np.abs(U_out_z2)**2

# Display the intensities
plt.figure(figsize = (10,10))
plt.subplot(1,2,1)
plt.imshow(Intensity_z1, cmap = "gray")
plt.title("Intensity at z=1mm")
```

```
plt.subplot(1,2,2)
plt.imshow(Intensity_z2, cmap = "gray")
plt.title("Intensity at z=10mm")
plt.show()
```



For propagation distance of 1mm there are more details in the intensity of the propagated field. However, increasing the propagation distance to 10mm leads to distortion in the image, possibly due to the diffraction.