In [ ]:

```python
import matplotlib.pyplot as plt;
import skimage.data as data
import numpy as np
```

In [ ]:

```python
def normalize(img):
    img = (img - img.min())/(img.max() - img.min())
    img = img*255
    img = img.astype(np.uint8)
    return img

def log_normalize(img):
    return 20*np.log(abs(img) + 1)
```

Utils

In [ ]:

```python
import cmath
from math import log, ceil
```

In [ ]:

```python
def omega(p, q):
    ''' The omega term in DFT and IDFT formulas'''
    return cmath.exp((2.0 * cmath.pi * 1j * q) / p)

def pad(lst):
    '''padding the list to next nearest power of 2 as FFT implemented is radix 2'''
    k = 0
    while 2**k < len(lst):
        k += 1
    return np.concatenate((lst, ([0] * (2 ** k - len(lst)))))

def pad2(x):
    m, n = np.shape(x)
    M, N = 2 ** int(ceil(log(m, 2))), 2 ** int(ceil(log(n, 2)))
    F = np.zeros((M,N), dtype = x.dtype)
    F[0:m, 0:n] = x
    return F
```

In [ ]:

```python
## FFT - 1D
def fft1d(x):
    ''' FFT of 1-d signals
    usage : X = fft(x)
    where input x = list containing sequences of a discrete time signals
    and output X = dft of x '''

    n = len(x)
    if n == 1:
        return x

    Feven, Fodd = fft(x[0::2]), fft(x[1::2])
    combined = [0] * n
    for m in range(n//2):
        combined[m] = Feven[m] + omega(n, -m) * Fodd[m]
        combined[m + n//2] = Feven[m] - omega(n, -m) * Fodd[m]
    return combined
```

In [ ]:

```python
## FFT - 2D
def fft(f):
    f = pad2(f)
    return np.transpose(fft1d(np.transpose(fft1d(f))))

def inverse_fft(F):
```

```
    M, N = np.shape(F)

    f = fft(np.conj(F))
    f = np.matrix(np.real(np.conj(f)))/(M*N)
    return f[0:M, 0:N]
```

Numpy FFT Functions

def fft(img): fft = np.fft.fft2(img) return fft

def inverse_fft(img): ifft = np.fft.ifft2(img) return ifft

8.1)

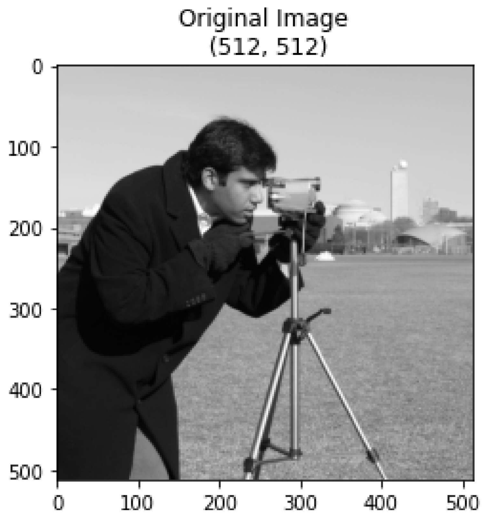
- a) 512 x 512 8bit bir imgeyi gösterin.
    - İmgenin 2 Boyutlu Fourierini alın.
    - Genlik ve fazlarını 0-255 arasına çekerek (log alabilirsiniz) gösterin.

In [ ]:

```
## Read in data file and transform
img_org = data.camera()
plt.imshow(img_org, cmap='gray')
plt.title(f'Original Image\n {img_org.shape}')
```

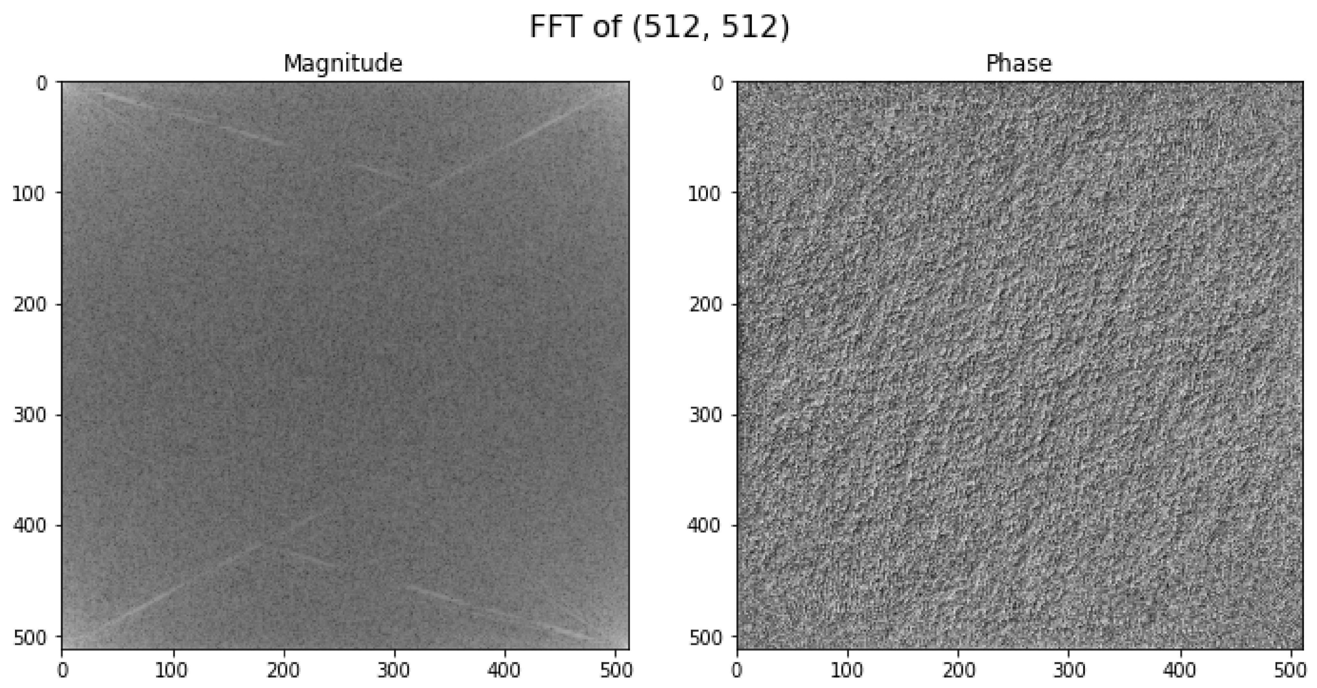
Out[ ]:

```
Text(0.5, 1.0, 'Original Image\n (512, 512)')
```



In [ ]:

```
img_fft = fft(img_org)

img_fft_mag = log_normalize(np.real(img_fft))
img_fft_phase = np.angle(img_fft)


# Plot the Fourier transform
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))
fig.suptitle(f'FFT of {img_org.shape}', fontsize=16)

ax1.imshow(img_fft_mag, cmap='gray')
ax1.set_title('Magnitude')

ax2.imshow(img_fft_phase, cmap='gray')
ax2.set_title('Phase')

fig.tight_layout()
plt.show()
```



8.1 b)

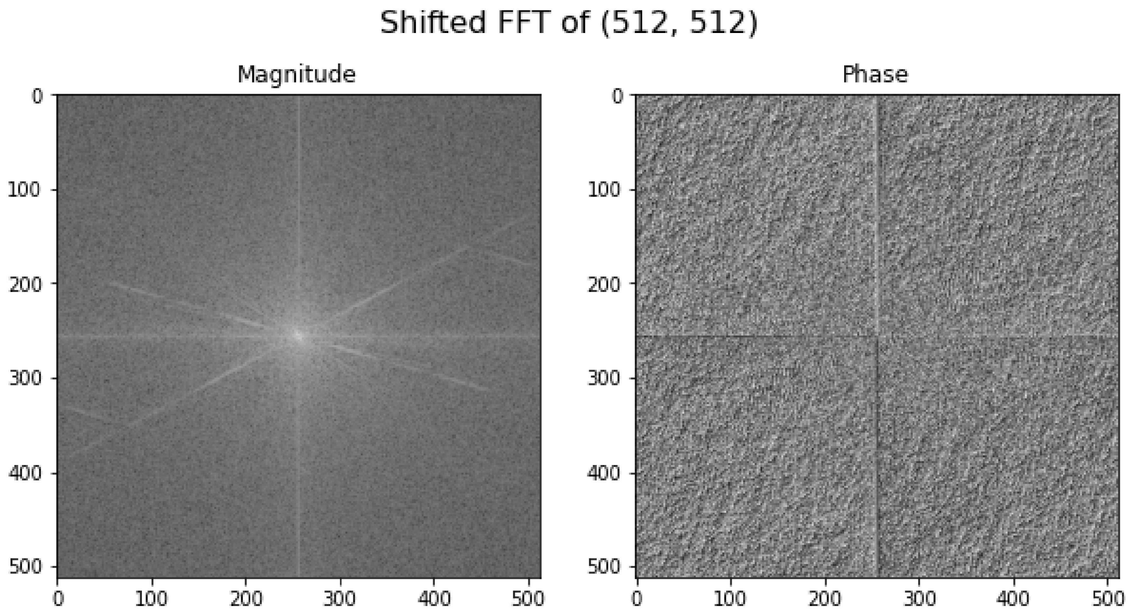
- Resmi (-1)^(x+y) ile çarparak tekrar FFT sini alıp genlik ve fazları gösterin (0-255 aralığına çekerek)

In [ ]:

```
def fft_shift(img):
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            img[i][j] = img[i][j] * pow(-1, (i+j))
    return img

img_fft_shifted = fft(fft_shift(img_org))

img_fft_shifted_mag = log_normalize(np.real(img_fft_shifted))
img_fft_shifted_phase = np.angle(img_fft_shifted)


# plot the shifted fft
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))
fig.suptitle(f'Shifted FFT of {img_fft.shape}', fontsize=16)

ax1.imshow(img_fft_shifted_mag, cmap='gray')
ax1.set_title('Magnitude')

ax2.imshow(img_fft_shifted_phase, cmap='gray')
ax2.set_title('Phase')
```

Out[ ]:

```
Text(0.5, 1.0, 'Phase')
```

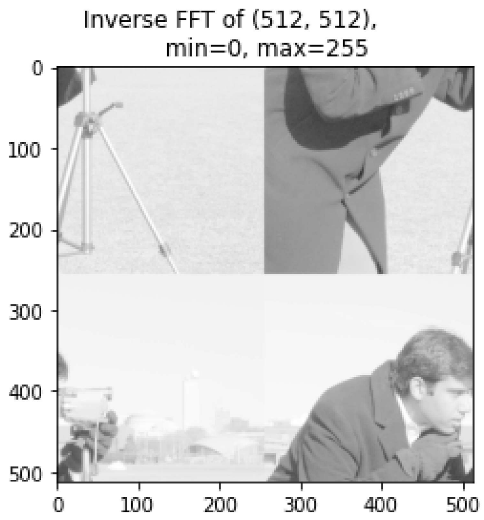


8.2

- İmgenin fft sini (f(u,v)) matrisini (-1)^(u+v) ile çarpıp ters fourierini alın. Sonucu gösterin ve tartışın

In [ ]:

```
img_fft_shifted = fft_shift(img_fft)

img_fft_shifted_ifft = inverse_fft(img_fft_shifted)
img_fft_shifted_ifft_norm = normalize(log_normalize(img_fft_shifted_ifft))

# plot the result
plt.imshow(img_fft_shifted_ifft_norm, cmap='gray')
plt.title(f'Inverse FFT of {img_fft_shifted_ifft_norm.shape}, \
        \n min={img_fft_shifted_ifft_norm.min()}, max={img_fft_shifted_ifft_norm.max()}')
plt.show()
```



In [ ]: