Discrete Fourier Transform

In [1]:

```
import numpy as np
import scipy.linalg as la
import matplotlib.pyplot as plt
```

Basic example

In [2]:

```
z = np.array([1,3,7,4,2,5,9,6])
```

Use the function numpy.fft.fft to compute the DFT.

In [3]:

```
np.fft.fft(z)
```

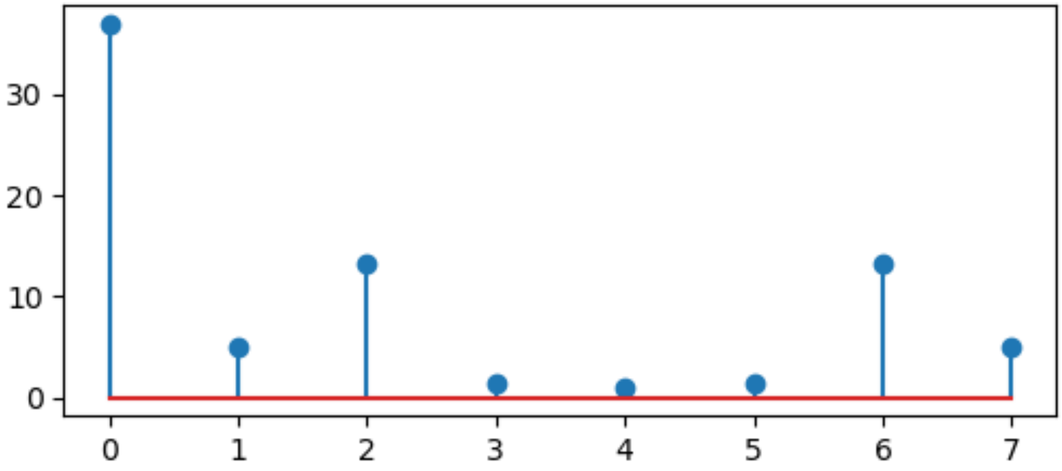
Out[3]:

```
array([ 37.+0.j        ,  -1.+4.82842712j, -13.+2.j        ,
        -1.+0.82842712j,   1.+0.j        ,  -1.-0.82842712j,
       -13.-2.j        ,  -1.-4.82842712j])
```

Note the conjugate symmetry between entries in the DFT. We will always get this when the input signal is a real vector.

In [4]:

```
plt.figure(figsize=(6,2.5))
plt.stem(np.abs(np.fft.fft(z)))
plt.show()
```



To get the inverse of the DFT, we use numpy.fft.ifft.

In [5]:

```
np.fft.ifft(np.fft.fft(z))
```

Out[5]:

```
array([1.+0.j, 3.+0.j, 7.+0.j, 4.+0.j, 2.+0.j, 5.+0.j, 9.+0.j, 6.+0.j])
```

The result after inverting will always be a complex vector regardless of whether the input signal was real or not. In this case we know that the output should be real, so we can use numpy.real to write the entries as real numbers instead.

In [6]:

```
print(np.real(np.fft.ifft(np.fft.fft(z))))
```

```
[1. 3. 7. 4. 2. 5. 9. 6.]
```

Filtering

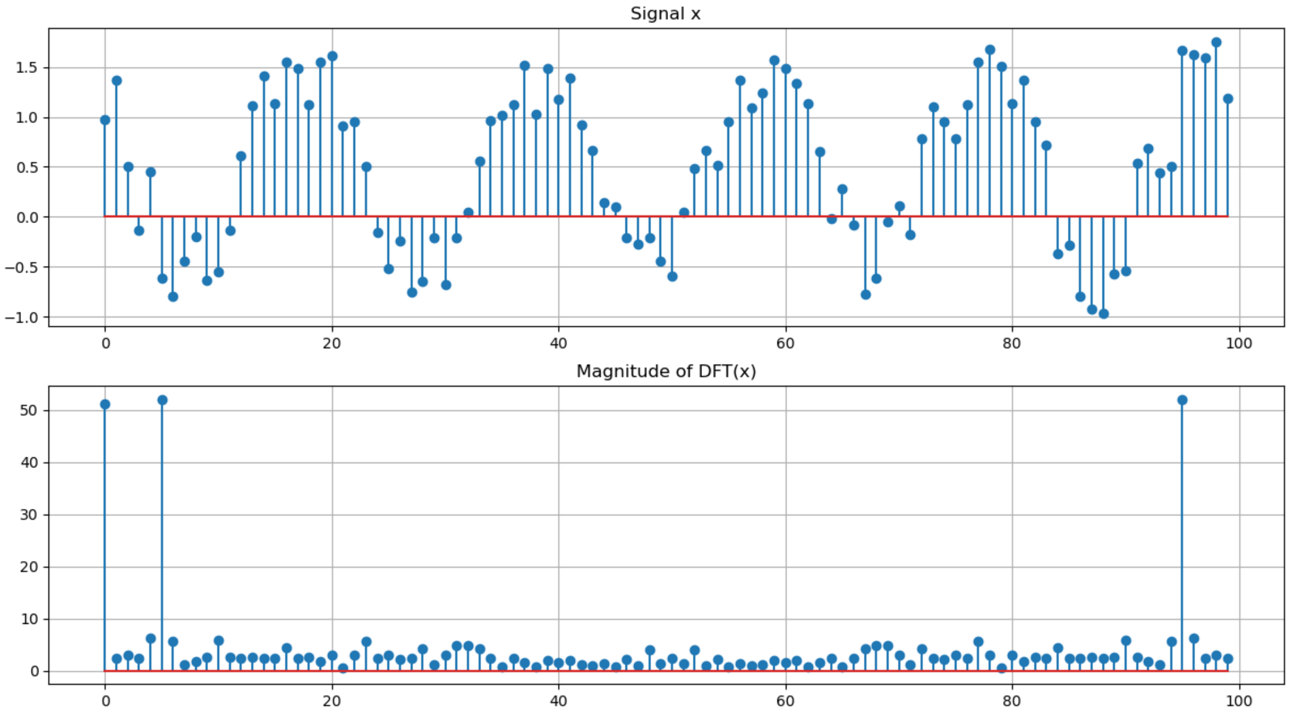
In [7]:

```
N = 100
k = 5
n = np.arange(0,N)
phi = np.pi/4
x = np.cos(2*np.pi*k*n/N + phi) + np.random.random(N)
y = np.fft.fft(x)

plt.figure(figsize=(15,8))
plt.subplot(2,1,1)
plt.stem(x)
plt.title('Signal x'); plt.grid(True);

plt.subplot(2,1,2)
plt.stem(np.abs(y))
plt.title('Magnitude of DFT(x)'); plt.grid(True);

plt.show();
```



The signal is roughly a sinusoid of frequency 5, but there is a lot of noise. We can filter out this noise by modifying the DFT, writing the value 0 for any entry that is of magnitude less than a certain threshold, say 10. Then we use the inverse transform to get the new filtered signal.

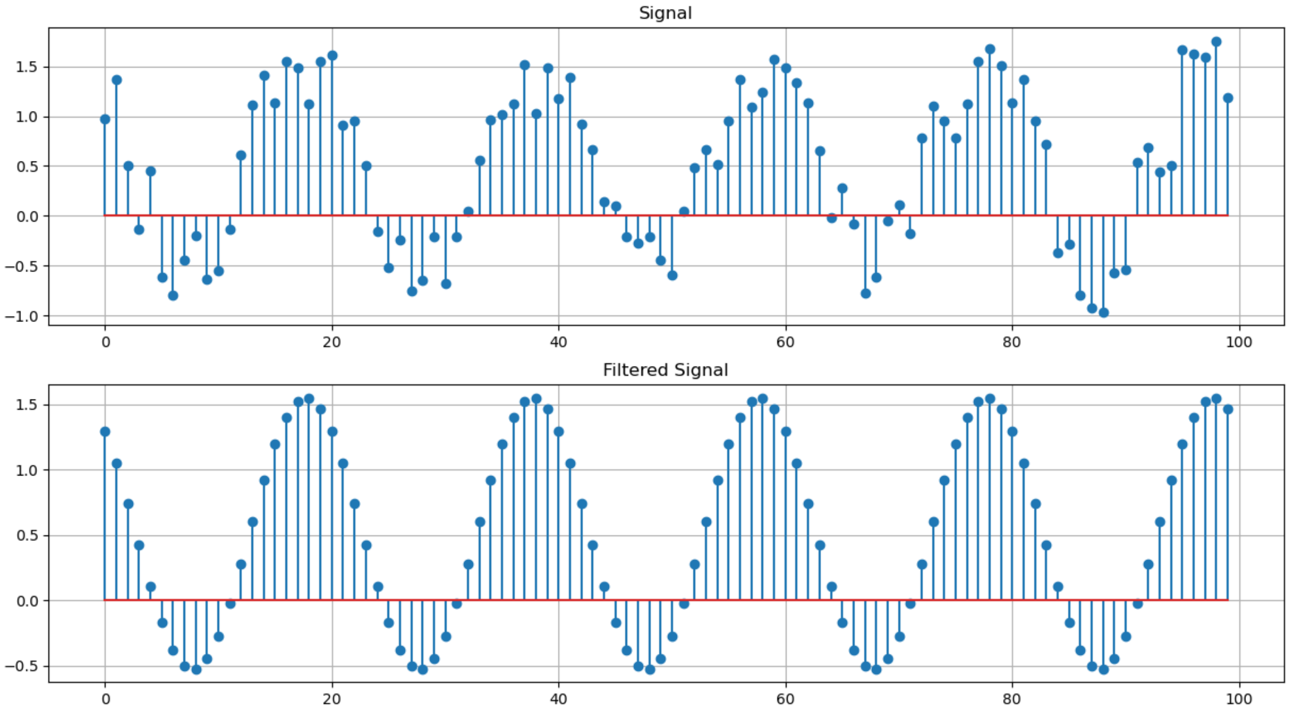
In [8]:

```
y_filtered = y.copy()
y_filtered[y < 10] = 0.0
plt.figure(figsize=(15,8))
x_filtered = np.fft.ifft(y_filtered).real

plt.subplot(2,1,1)
plt.stem(x)
plt.title('Signal'); plt.grid(True);

plt.subplot(2,1,2)
plt.stem(x_filtered)
plt.title('Filtered Signal'); plt.grid(True);

plt.show();
```



Signal

Filtered Signal

In [ ]: