Transform Signals from Time Domain to Frequency Domain with FFT

Purpose:

- Use narray to generate sine-waveforms (signals) with different frequencies (i.e., 4Hz, 7Hz. & 9Hz)
- Plot these waveforms in (Amplitude vs Time) with each individual waveform in a separate graph using matplotlib
- Merge/combine the waveforms(signals) into one waveform(signal)
- Plot the combined waveform in (Amplitude vs Time) using matplotlib
- Transform the combined signal from time domain into frequency domain waveform using numpy's FFT
- Plot the transformed waveform in (Amplitude vs Frequency) using matplotlib

Key Concepts:

Forward and Inverse Fourier Transforms

Forward Fourier Transform: Analysis Equation

$$X(\omega) = \int_{-\infty}^{+\infty} x(t)e^{-j\omega t}dt$$

Inverse Fourier Transform: Synthesis Equation

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} X(\omega)e^{j\omega t}d\omega$$

The Discrete Fourier Transform Equations

Forward Discrete Fourier Transform (DFT):

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i\,2\pi\,k\,n\,/\,N}$$

Inverse Discrete Fourier Transform (IDFT):

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{i\,2\pi\,k\,n\,/\,N}$$

The Implementation of FFT

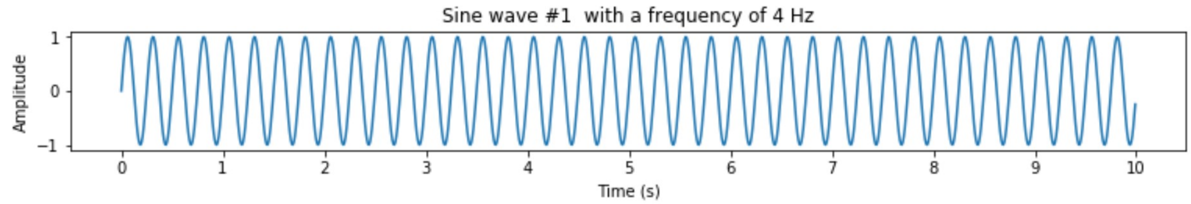
The FFT is the implementation of the DFT and IDFT based on the algorithm lay out in JW Cooley and John Tukey's [1965 paper (http://www.ams.org/journals/mcom/1965-19-090/S0025-5718-1965-0178586-1/)](http://www.ams.org/journals/mcom/1965-19-090/S0025-5718-1965-0178586-1/).

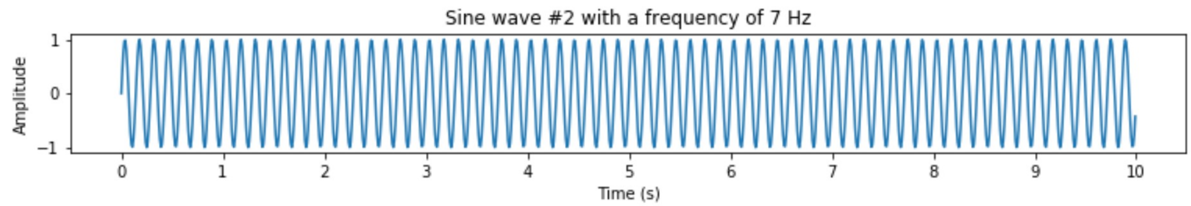
In [6]:
```
%matplotlib inline
# Python example - Fourier transform using numpy.fft method
import numpy as np
import matplotlib.pyplot as plotter
```

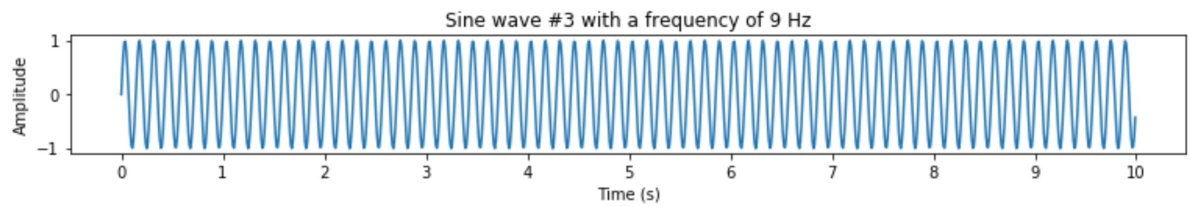
In [7]:
```
# How many time points are needed i,e., Sampling Frequency
samplingFrequency   = 100;

# At what intervals time points are sampled
samplingInterval       = 1 / samplingFrequency;

# Begin time period of the signals
beginTime           = 0;

# End time period of the signals
endTime             = 10;

# Frequency of the signals
signal1Frequency     = 4; #Sine wave 1
signal2Frequency     = 7; #Sine wave 2
signal3Frequency     = 9 #Sine wave 3

# Time points
time        = np.arange(beginTime, endTime, samplingInterval);
#print(time)
```

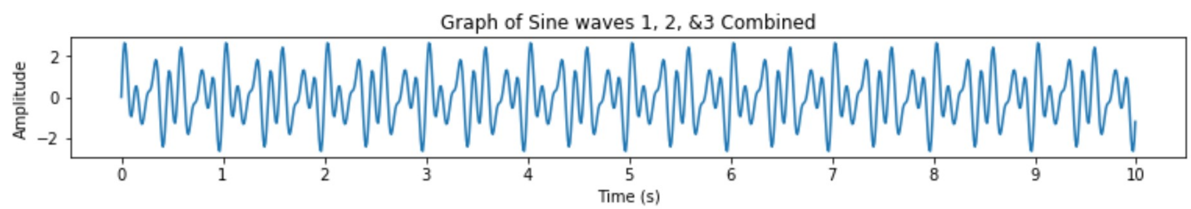
In [8]:
```
# Create two sine waves
amplitude1 = np.sin(2*np.pi*signal1Frequency*time)
amplitude2 = np.sin(2*np.pi*signal2Frequency*time)
amplitude3 = np.sin(2*np.pi*signal3Frequency*time)
#print(amplitude1)
```

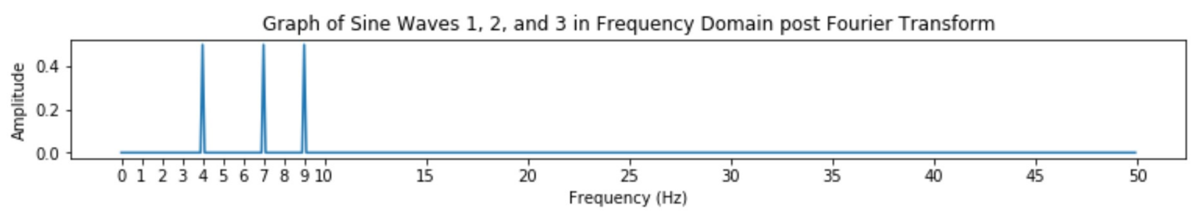
In [9]:
```
# Create subplot
figure, axis = plotter.subplots(5, 1)

figure.set_figwidth(12.8)
#figure.set_figheight(9.6)
figure.set_figheight(12.6)
plotter.subplots_adjust(hspace=1)

# Time domain representation for sine wave 1
axis[0].set_title('Sine wave #1  with a frequency of 4 Hz')
axis[0].set_xlabel('Time (s)')
axis[0].set_ylabel('Amplitude')
axis[0].set_xticks([0,1,2,3,4,5,6,7,8,9,10])


# Time domain representation for sine wave 2
axis[1].set_title('Sine wave #2 with a frequency of 7 Hz')
axis[1].set_xlabel('Time (s)')
axis[1].set_ylabel('Amplitude')
axis[1].set_xticks([0,1,2,3,4,5,6,7,8,9,10])

# Time domain representation for sine wave 3
axis[2].set_title('Sine wave #3 with a frequency of 9 Hz')
axis[2].set_xlabel('Time (s)')
axis[2].set_ylabel('Amplitude')
axis[2].set_xticks([0,1,2,3,4,5,6,7,8,9,10])

# Add the sine waves -- plot the 2 sine waves as one graph
amplitude = amplitude1 + amplitude2 + amplitude3

# Time domain representation of the resultant sine wave
#axis[2].set_title('Sine wave with multiple frequencies')
axis[3].set_title('Graph of Sine waves 1, 2, &3 Combined')
axis[3].set_xlabel('Time (s)')
axis[3].set_ylabel('Amplitude')
axis[3].set_xticks([0,1,2,3,4,5,6,7,8,9,10])

#--- Transform from time domain to frequency domain using FFT ---#

# Frequency domain representation
fourierTransform = np.fft.fft(amplitude)/len(amplitude)           # Normalize ampli
tude
fourierTransform = fourierTransform[range(int(len(amplitude)/2))] # Exclude samplin
g frequency
tpCount     = len(amplitude)
values      = np.arange(int(tpCount/2))
timePeriod  = tpCount/samplingFrequency
frequencies = values/timePeriod

# Plot Frequency domain graph
#axis[3].set_title('Fourier transform depicting the frequency components')
axis[4].set_title('Graph of Sine Waves 1, 2, and 3 in Frequency Domain post Fourier
Transform')
axis[4].set_xlabel('Frequency (Hz)')
axis[4].set_ylabel('Amplitude')
axis[4].set_xticks([0,1,2,3,4,5,6,7,8,9,10,15,20,25,30,35,40,45,50])

# Plot everything
axis[0].plot(time, amplitude1)
axis[1].plot(time, amplitude2)
axis[2].plot(time, amplitude2)
axis[3].plot(time, amplitude)
axis[4].plot(frequencies, abs(fourierTransform))
#print(frequencies)
```











In [10]:

```
realFourierTransform = abs(fourierTransform)
arrFreq = np.array([frequencies, realFourierTransform])
print(f'After transforming with FFT, found the following freq at: ')
for i, j in np.argwhere(arrFreq > 0.495):
    if i > 0:
        print(f'{j/10} Hz ') # since (end - start = 10), therefore freq = 1/t ==> 1
/10
```

```
After transforming with FFT, found the following freq at:
4.0 Hz
7.0 Hz
9.0 Hz
```