https://drive.google.com/file/d/1EjyP6oiyLX5-TAd2JJ0LKrKhnFboKrhh/view?usp=sharing

Sampling of Signals

Q1. Generate the following signals with the specified sampling rate and listen to it. Let t vary from $0 - 3$seconds. Let the sampling rate is 8000Hz or a sampling time period is $125 \mu s$ (A sample is taken at every integer multiple of $125 \mu s$)

1. $x(t) = sin(100\pi t)$ (100 Hz sinusoid)
2. $x(t) = cos(1000\pi t)$ (1kHz sinusoid)
3. $x(t) = sin(100 \pi t^2) $ (Called a chirp signal)

To listen to a signal, write the signal to a file using wave write command from scipy Please look up the documentation here (https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.io.wavfile.write.html). Then download the signal and listen to it.

In [1]:
```
import numpy as np
import scipy.io.wavfile as wav
import matplotlib.pyplot as plt
```

In [2]:
```
# Set the sampling rate
_RATE = 8000
# Genearate the t values use np.arange
_T = np.arange(0, 3, 1 / _RATE)
_T, _RATE
```

Out[2]:
```
(array([0.000000e+00, 1.250000e-04, 2.500000e-04, ..., 2.999625e+00,
        2.999750e+00, 2.999875e+00]),
 8000)
```

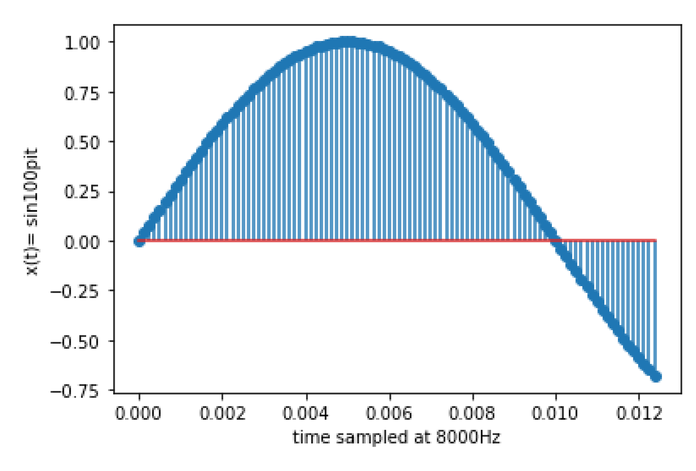
In [3]:
```
t = _T
```

1) $x(t) = sin(100\pi t)$ (50 Hz sinusoid)

In [4]:
```
#Generate signal
s1 = np.sin(100 * np.pi * t)
#Plot first 100 samples of the generated signal
plt.stem(t[:100], s1[:100])

plt.xlabel('time sampled at 8000Hz')
plt.ylabel('x(t)= sin100pit')
```

Out[4]:
```
Text(0, 0.5, 'x(t)= sin100pit')
```



In [5]:
```
import IPython.display as ipd
```

In [6]:
```
ipd.Audio(data=s1, rate=_RATE)
```

Out[6]: 0:00 / 0:03

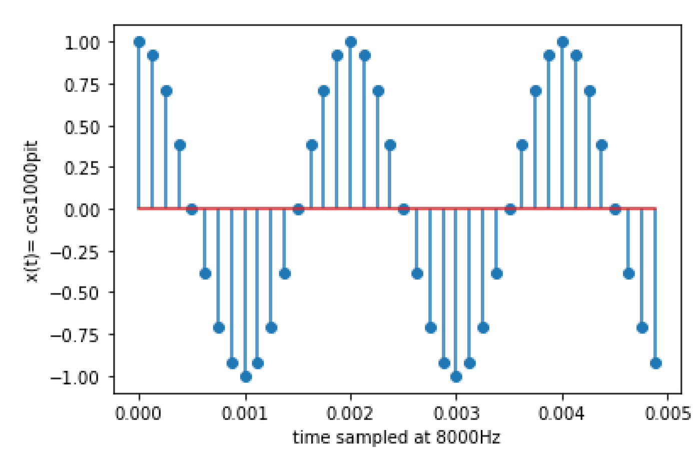
In [7]:
```
#Write thee signal
wav.write('sin100.wav', _RATE, s1)
```

2) $x(t)=cos(1000\pi t)$ (.5kHz sinusoid)

In [8]:
```
#Generate signal
s2 = np.cos(1000 * np.pi * t)
#Plot first 40 samples of the generated signal
plt.stem(t[:40], s2[:40])

plt.xlabel('time sampled at 8000Hz')
plt.ylabel('x(t)= cos1000pit')
```

Out[8]:
```
Text(0, 0.5, 'x(t)= cos1000pit')
```



In [9]:
```
ipd.Audio(data=s2, rate=_RATE)
```

Out[9]: 0:00 / 0:03

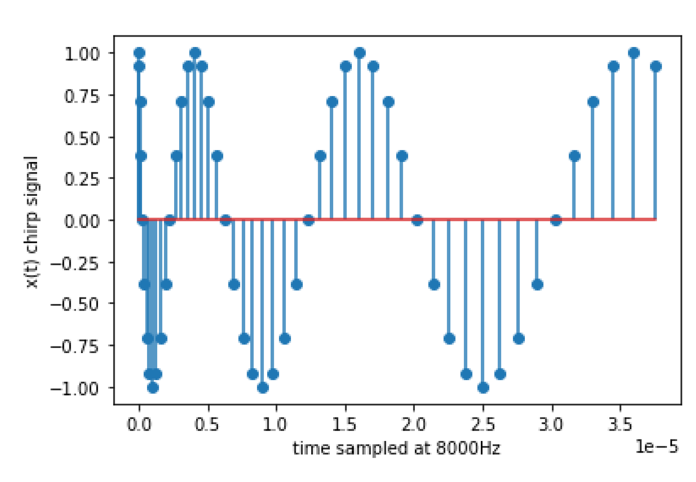
In [10]:
```
#Write thee signal
wav.write('cos1000.wav', _RATE, s2)
```

3) $x(t)=sin(100\pi t2)$ (Called a chirp signal)

In [11]:
```
t = _T**2
#Generate signal
s3 = np.sin(100 * np.pi * t)
#Plot first 40 samples of the generated signal
plt.stem(t[:50], s2[:50])


plt.xlabel('time sampled at 8000Hz')
plt.ylabel('x(t) chirp signal')
```

Out[11]:
```
Text(0, 0.5, 'x(t) chirp signal')
```



In [12]:
```
ipd.Audio(data=s3, rate=_RATE)
```

Out[12]: 0:00 / 0:03

In [13]:
```
#Write thee signal
wav.write('chirp100.wav', _RATE, s2)
```

Q2. Sample the signal $cos (10 \pi t)$ using the following frequencies (a) 20 Hz (b) 7.5 Hz (c) 5 Hz (d) 2.5 Hz. In each case, plot the signal and determine its period.

(a) 20 Hz

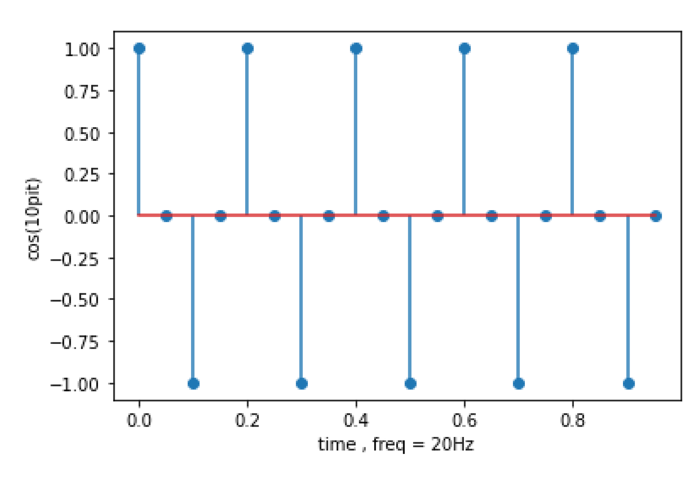
In [14]:
```
# Set the sampling rate
_RATE = 20
# Genearate the t values use np.arange
t = np.arange(0, 1, 1 / _RATE)
print(_T, _RATE)
#Generate signal
s1 = np.cos(10 * np.pi * t)
#Plot the generated signal
plt.stem(t,s1)

plt.xlabel('time , freq = 20Hz')
plt.ylabel('cos(10pit)')

#sampling time period =  0.05
```

```
[0.000000e+00 1.250000e-04 2.500000e-04 ... 2.999625e+00 2.999750e+00
 2.999875e+00] 20
```

Out[14]:
```
Text(0, 0.5, 'cos(10pit)')
```



(b) 7.5 Hz

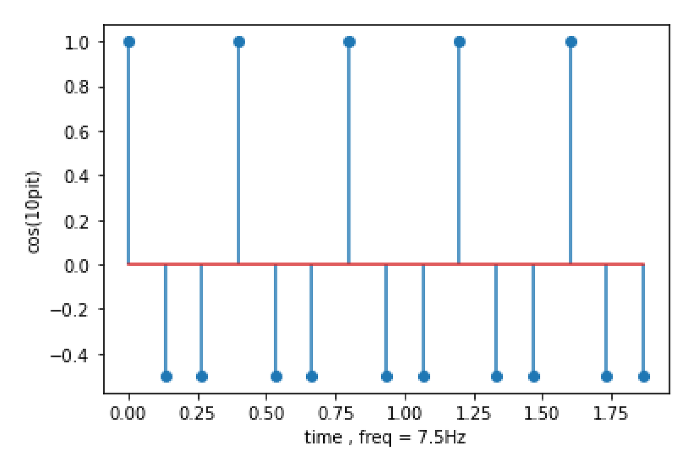
In [15]:
```
# Set the sampling rate
_RATE = 7.5
# Genearate the t values use np.arange
t = np.arange(0, 2, 1 / _RATE)
print(_T, _RATE)
#Generate signal
s2 = np.cos(10 * np.pi * t)
#Plot the generated signal
plt.stem(t,s2)

plt.xlabel('time , freq = 7.5Hz')
plt.ylabel('cos(10pit)')

#sampling time period = 0.134 sec
```

```
[0.000000e+00 1.250000e-04 2.500000e-04 ... 2.999625e+00 2.999750e+00
 2.999875e+00] 7.5
```

Out[15]:
```
Text(0, 0.5, 'cos(10pit)')
```



(c) 5 Hz

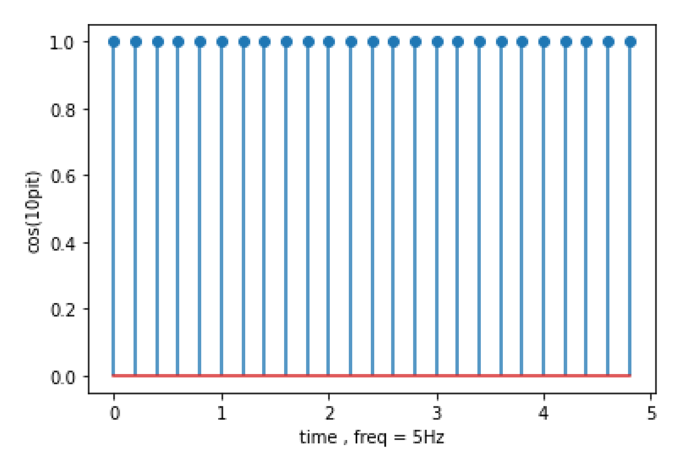
In [16]:
```
# Set the sampling rate
_RATE = 5
# Genearate the t values use np.arange
t = np.arange(0, 5, 1 / _RATE)
print(_T, _RATE)
#Generate signal
s3 = np.cos(10 * np.pi * t)
#Plot the generated signal
plt.stem(t,s3)

plt.xlabel('time , freq = 5Hz')
plt.ylabel('cos(10pit)')

#sampling time period = 0.2 sec
```

```
[0.000000e+00 1.250000e-04 2.500000e-04 ... 2.999625e+00 2.999750e+00
 2.999875e+00] 5
```

Out[16]:
```
Text(0, 0.5, 'cos(10pit)')
```



(d) 2.5 Hz

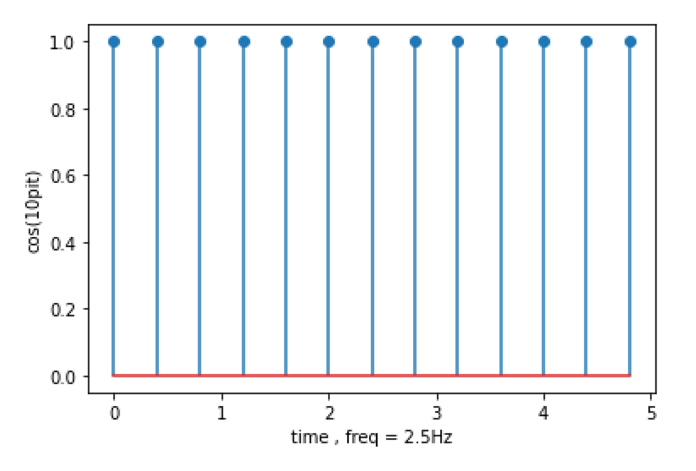
In [17]:
```
# Set the sampling rate
_RATE = 2.5
# Genearate the t values use np.arange
t = np.arange(0, 5, 1 / _RATE)
print(_T, _RATE)
#Generate signal
s4 = np.cos(10 * np.pi * t)
#Plot the generated signal
plt.stem(t,s4)

plt.xlabel('time , freq = 2.5Hz')
plt.ylabel('cos(10pit)')

#sampling time period = 0.4sec
```

```
[0.000000e+00 1.250000e-04 2.500000e-04 ... 2.999625e+00 2.999750e+00
 2.999875e+00] 2.5
```

Out[17]:
```
Text(0, 0.5, 'cos(10pit)')
```



Q3. In DTMF dialling a number is represented by a dual frequency tone. Do a web search and find the frequencies of each digit. Generate DTMF tones corresponding to the telephone number 08242474040 by sampling the sum of sinusoids at the required frequencies at Fs = 8192 Hz. Concatenate the signals by putting 100 zeros between each signal (to represent silence) and listen to the signal. (Must sound like tone dialling the number from a phone)

In [18]:
```
_RATE = 8192
```

https://rfmw.em.keysight.com/rfcomms/refdocs/cdma2k/cdma2000_meas_dtmf_desc.html image.png

In [19]:
```
row = [1209, 1336, 1447, 1663]
col = [697, 770, 852, 941]
dtmf_freq = dict()
dtmf_freq[0] = (row[1], col[-1])
r = 0
c = -1
for i in range(9):
    if i % 3 == 0:
        c += 1
    dtmf_freq[i + 1] = (row[r], col[c])
    r = (r + 1) % 3
dtmf_freq
```

Out[19]:
```
{0: (1336, 941),
 1: (1209, 697),
 2: (1336, 697),
 3: (1447, 697),
 4: (1209, 770),
 5: (1336, 770),
 6: (1447, 770),
 7: (1209, 852),
 8: (1336, 852),
 9: (1447, 852)}
```

In [20]:
```
dtmf_tone = dict()
for n, (f1, f2) in dtmf_freq.items():
    t = np.arange(0, .25, 1 / _RATE)
    s = np.sin(2 * np.pi * f1 * t) +  np.sin(2 * np.pi * f2 * t)
    dtmf_tone[n] = np.concatenate((s, np.zeros(100)))
```

In [21]:
```
number = '08242474040'
number_tone = np.concatenate(([dtmf_tone[int(n)] for n in number]))
number_tone
ipd.Audio(data=number_tone, rate=_RATE)
```

Out[21]: 0:00 / 0:02

https://drive.google.com/file/d/17vsa89cRQsyJtHNWqFuTjFfwXR0wXiMM/view?usp=sharing

When we press a key ,which corresponds to a number or symbol—the phone generates a tone that simultaneously combines the high-frequency signal from the column that key is in with the low-frequency signal of the row it's in. This unique signal pair is then transmitted over telephone wires to the local phone exchange, where the two signals are decoded to determine which numbers you are dialing. So when you press the "5" key on your phone's keypad, for example, a combined signal tone of 1336 Hz and 770 Hz is sent to the phone company, which then knows that you've just pressed "5".Once they receive the full number that you dialed, they can automatically route your call to it.

Q4. Record your own voice saying vowel /a/ as in cat (It can be done using a sound recording program in the computer). Please save in *.wav format, nd read it using wavread command in Python for further processing). Please look https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.io.wavfile.read.html for documentation

(a) Find the sampling rate used by the recorder
(b) Just zoom and plot only the middle 100 milliseconds of the data
(c) Find the mean (average value) and variance of the entire signal. Use np.mean and np.var commands

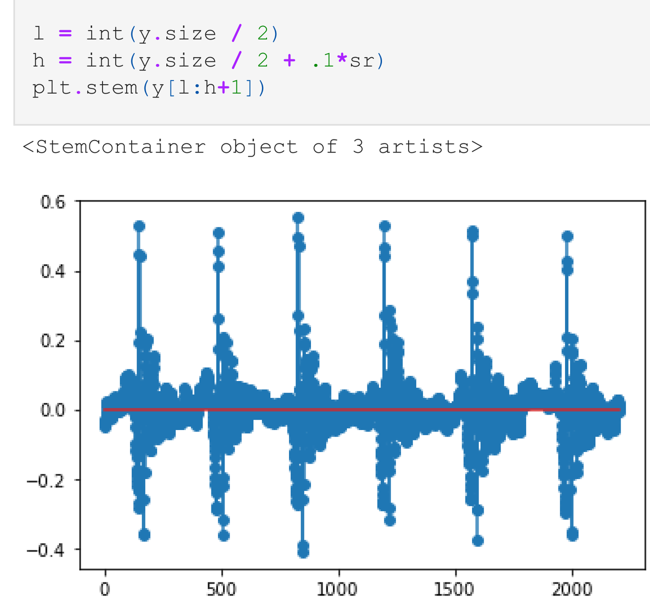
In [22]:
```
import librosa
```

In [23]:
```
y, sr = librosa.load('a.ogg')
print(f"The sample rate of the recorder is {sr} Hz.")
ipd.Audio(data=y, rate=sr)
print(type(sr))
```

```
The sample rate of the recorder is 22050 Hz.
<class 'int'>
```

In [24]:
```
l = int(y.size / 2)
h = int(y.size / 2 + .1*sr)
plt.stem(y[l:h+1])
```

Out[24]:
```
<StemContainer object of 3 artists>
```



In [25]:
```
mean = np.mean(y)
variance = np.var(y)
print(f"The mean of the signal is {mean} and the variance is {variance}.")
```

```
The mean of the signal is -2.2171128875925206e-05 and the variance is 0.00824511144310236.
```