



**Marwadi University**  
**Faculty of Engineering & Technology**  
**Department of Information and Communication Technology**

**Subject: Programming With Python (01CT1309)**

**Aim:** Practical based on Signal Processing using Scipy

**Experiment No: 12**

**Date:**

**Enrollment No: 92400133037**

**Aim:** Practical based on Signal Processing using Scipy

**IDE:**

What is SciPy?

SciPy is a free and open-source Python library used for scientific computing and technical computing. It is a collection of mathematical algorithms and convenience functions built on the NumPy extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data. As mentioned earlier, SciPy builds on NumPy and therefore if you import SciPy, there is no need to import NumPy.

**Generates a sine wave and a square wave with a frequency of 5 Hz and a sampling frequency of 500 Hz.**

```
import numpy as np  
  
import matplotlib.pyplot as plt  
  
from scipy import signal  
  
# Parameters  
  
fs = 500 # Sampling frequency  
  
f = 5 # Frequency of the signal  
  
t = np.linspace(0, 1, fs, endpoint=False) # Time array  
  
# Create a sine wave signal  
  
sine_wave = np.sin(2 * np.pi * f * t)  
  
# Create a square wave signal using scipy  
  
square_wave = signal.square(2 * np.pi * f * t)  
  
# Plot the signals  
  
plt.figure(figsize=(10, 5))  
  
plt.subplot(2, 1, 1)
```



**Subject: Programming With Python (01CT1309)**

**Aim:** Practical based on Signal Processing using Scipy

**Experiment No: 12**

**Date:**

**Enrollment No: 92400133037**

```
plt.plot(t, sine_wave)
```

```
plt.title('Sine Wave')
```

```
plt.xlabel('Time [s]')
```

```
plt.ylabel('Amplitude')
```

```
plt.subplot(2, 1, 2)
```

```
plt.plot(t, square_wave)
```

```
plt.title('Square Wave')
```

```
plt.xlabel('Time [s]')
```

```
plt.ylabel('Amplitude')
```

```
plt.tight_layout()
```

```
plt.show()
```



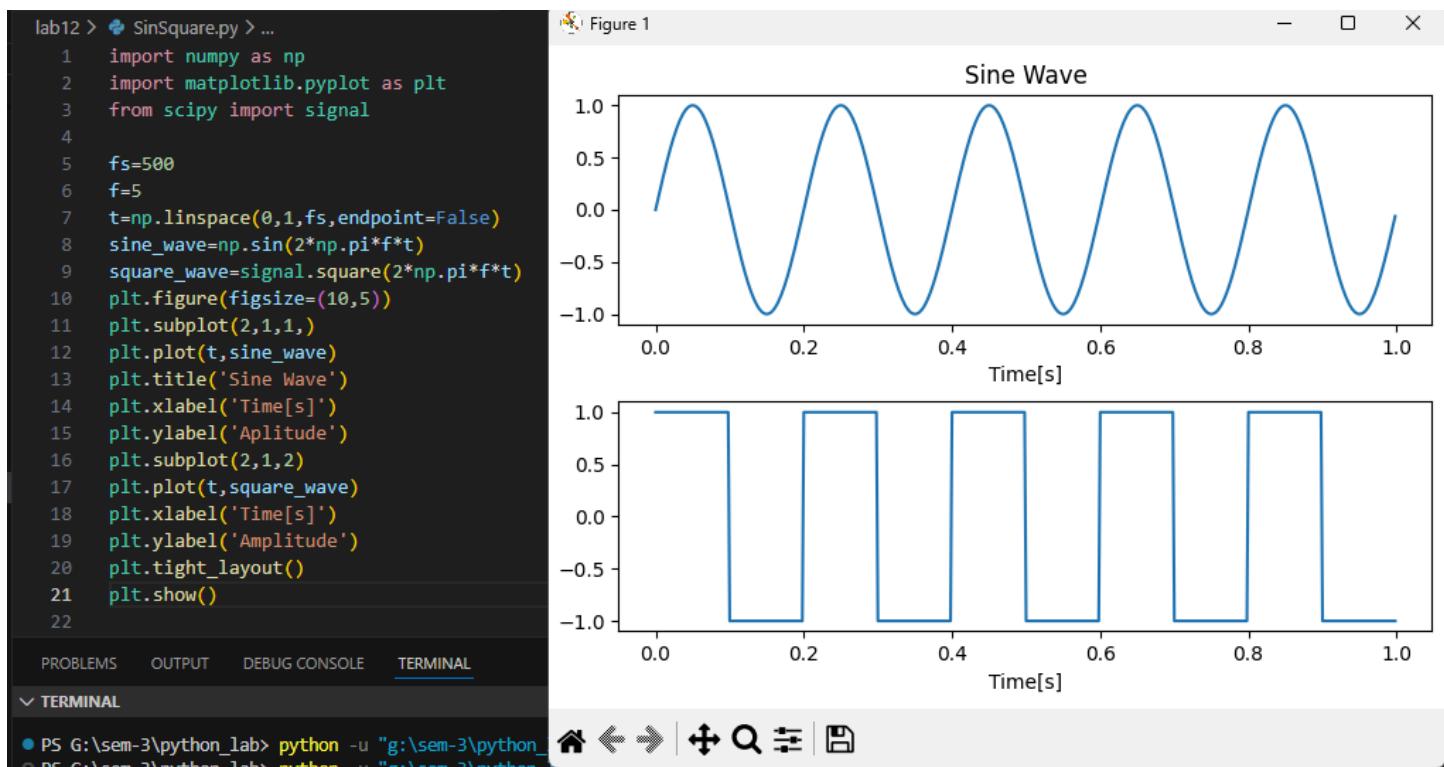
**Subject: Programming With Python (01CT1309)**

**Aim:** Practical based on Signal Processing using Scipy

**Experiment No: 12**

**Date:**

**Enrollment No: 92400133037**



### Triangular and Ramp signal

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
# Parameters
fs = 500 # Sampling frequency
f = 5 # Frequency of the signal
t = np.linspace(0, 1, fs, endpoint=False) # Time array
# Create a triangular wave signal using scipy
triangular_wave = signal.sawtooth(2 * np.pi * f * t, 0.5)
```



**Subject: Programming With Python (01CT1309)**

**Aim:** Practical based on Signal Processing using Scipy

**Experiment No: 12**

**Date:**

**Enrollment No: 92400133037**

```
# Create a ramp (sawtooth) signal using scipy
```

```
ramp_signal = signal.sawtooth(2 * np.pi * f * t)
```

```
# Plot the signals
```

```
plt.figure(figsize=(10, 5))
```

```
plt.subplot(2, 1, 1)
```

```
plt.plot(t, triangular_wave)
```

```
plt.title('Triangular Wave')
```

```
plt.xlabel('Time [s]')
```

```
plt.ylabel('Amplitude')
```

```
plt.subplot(2, 1, 2)
```

```
plt.plot(t, ramp_signal)
```

```
plt.title('Ramp Signal')
```

```
plt.xlabel('Time [s]')
```

```
plt.ylabel('Amplitude')
```

```
plt.tight_layout()
```

```
plt.show()
```



**Subject: Programming With Python (01CT1309)**

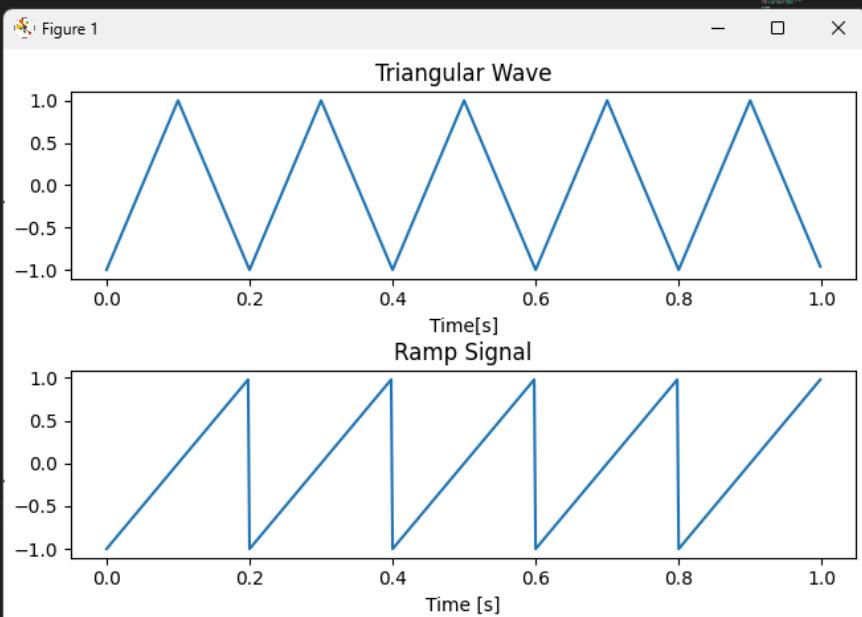
**Aim:** Practical based on Signal Processing using Scipy

**Experiment No: 12**

**Date:**

**Enrollment No: 92400133037**

```
lab12 > TrianRamp.py ...
 1  import numpy as np
 2  import matplotlib.pyplot as plt
 3  from scipy import signal
 4
 5  fs=500
 6  f=5
 7  t=np.linspace(0,1,fs,endpoint=False)
 8  triangular_wave=signal.sawtooth(2*np.pi*f*t,0.5)
 9
10 ramp_signal=signal.sawtooth(2*np.pi*f*t)
11 plt.figure(figsize=(10,5))
12 plt.subplot(2,1,1)
13 plt.plot(t,triangular_wave)
14 plt.title('Triangular Wave')
15 plt.xlabel('Time[s]')
16 plt.ylabel('Amplitude')
17 plt.subplot(2, 1, 2)
18 plt.plot(t, ramp_signal)
19 plt.title('Ramp Signal')
20 plt.xlabel('Time [s]')
21 plt.ylabel('Amplitude')
22 plt.tight_layout()
23 plt.show()
```



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Home ← → ⌂ Q ⌂ | ⌂

## #Elementary signals

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
from scipy import signal
```

```
# Parameters
```

```
fs = 500 # Sampling frequency
```

```
t = np.linspace(-1, 1, fs, endpoint=False) # Time array
```

```
# 1. Unit Step Signal
```

```
unit_step = np.heaviside(t, 1)
```

```
# 2. Unit Impulse Signal (Dirac Delta)
```



**Subject: Programming With Python (01CT1309)**

**Aim:** Practical based on Signal Processing using Scipy

**Experiment No: 12**

**Date:** **Enrollment No: 92400133037**

```
unit_impulse = np.zeros_like(t)
```

```
unit_impulse[fs//2] = 1 # Impulse at t=0
```

# 3. Ramp Signal

```
ramp_signal = signal.sawtooth(2 * np.pi * t, 1)
```

# 4. Sine Wave

```
f_sine = 5 # Frequency of the sine wave
```

```
sine_wave = np.sin(2 * np.pi * f_sine * t)
```

# 5. Cosine Wave

```
f_cosine = 5 # Frequency of the cosine wave
```

```
cosine_wave = np.cos(2 * np.pi * f_cosine * t)
```

# 6. Exponential Signal

```
exponential_signal = np.exp(t)
```

# 7. Triangular Wave

```
triangular_wave = signal.sawtooth(2 * np.pi * 5 * t, 0.5)
```

# 8. Square Wave

```
square_wave = signal.square(2 * np.pi * 5 * t)
```

# Plot the signals

```
plt.figure(figsize=(12, 12))
```

```
plt.subplot(4, 2, 1)
```

```
plt.plot(t, unit_step)
```

```
plt.title('Unit Step Signal')
```



**Subject: Programming With Python (01CT1309)**

**Aim:** Practical based on Signal Processing using Scipy

**Experiment No: 12**

**Date:**

**Enrollment No: 92400133037**

```
plt.xlabel('Time [s]')
plt.ylabel('Amplitude')
plt.subplot(4, 2, 2)
plt.plot(t, unit_impulse)
plt.title('Unit Impulse Signal')
plt.xlabel('Time [s]')
plt.ylabel('Amplitude')
plt.subplot(4, 2, 3)
plt.plot(t, ramp_signal)
plt.title('Ramp Signal')
plt.xlabel('Time [s]')
plt.ylabel('Amplitude')
plt.subplot(4, 2, 4)
plt.plot(t, sine_wave)
plt.title('Sine Wave')
plt.xlabel('Time [s]')
plt.ylabel('Amplitude')
plt.subplot(4, 2, 5)
plt.plot(t, cosine_wave)
plt.title('Cosine Wave')
plt.xlabel('Time [s]')
```



**Subject: Programming With Python (01CT1309)**

**Aim:** Practical based on Signal Processing using Scipy

**Experiment No: 12**

**Date:**

**Enrollment No: 92400133037**

```
plt.ylabel('Amplitude')
```

```
plt.subplot(4, 2, 6)
```

```
plt.plot(t, exponential_signal)
```

```
plt.title('Exponential Signal')
```

```
plt.xlabel('Time [s]')
```

```
plt.ylabel('Amplitude')
```

```
plt.subplot(4, 2, 7)
```

```
plt.plot(t, triangular_wave)
```

```
plt.title('Triangular Wave')
```

```
plt.xlabel('Time [s]')
```

```
plt.ylabel('Amplitude')
```

```
plt.subplot(4, 2, 8)
```

```
plt.plot(t, square_wave)
```

```
plt.title('Square Wave')
```

```
plt.xlabel('Time [s]')
```

```
plt.ylabel('Amplitude')
```

```
plt.tight_layout()
```

```
plt.show()
```



**Subject: Programming With Python (01CT1309)**

**Aim:** Practical based on Signal Processing using Scipy

**Experiment No: 12**

**Date:**

**Enrollment No: 92400133037**

```
lab12 > Elementary.py > ...
1  import numpy as np
2  import matplotlib.pyplot as plt
3  from scipy import signal
4  fs=500
5  t=np.linspace(-1, 1, fs, endpoint=False)
6  unit_step=np.heaviside(t, 1)
7  unit_impulse=np.zeros_like(t)
8  unit_impulse[fs//2]=1
9  ramp_signal=signal.sawtooth(2*np.pi*t,1)
10 f_sine=5
11 sine_wave=np.sin(2*np.pi*f_sine*t)
12 f_cosine=5
13 cosine_wave=np.cos(2*np.pi*f_cosine*t)
14 exponential_signal=np.exp(t)
15 triangular_wave=signal.sawtooth(2*np.pi*5*t,0.5)
16 square_wave = signal.square(2 * np.pi * 5 * t)
17 # Plot the signals
18 plt.figure()
19 plt.subplot(4, 2, 1)
20 plt.plot(t, unit_step)
21 plt.title('Unit Step Signal')
22 plt.xlabel('Time [s]')
23 plt.ylabel('Amplitude')
```



**Subject: Programming With Python (01CT1309)**

**Aim:** Practical based on Signal Processing using Scipy

**Experiment No: 12**

**Date:**

**Enrollment No: 92400133037**

```
lab12 > Elementary.py > ...
24 plt.subplot(4, 2, 2)
25 plt.plot(t, unit_impulse)
26 plt.title('Unit Impulse Signal')
27 plt.xlabel('Time [s]')
28 plt.ylabel('Amplitude')
29 plt.subplot(4, 2, 3)
30 plt.plot(t, ramp_signal)
31 plt.title('Ramp Signal')
32 plt.xlabel('Time [s]')
33 plt.ylabel('Amplitude')
34 plt.subplot(4, 2, 4)
35 plt.plot(t, sine_wave)
36 plt.title('Sine Wave')
37 plt.xlabel('Time [s]')
38 plt.ylabel('Amplitude')
39 plt.subplot(4, 2, 5)
40 plt.plot(t, cosine_wave)
41 plt.title('Cosine Wave')
42 plt.xlabel('Time [s]')
43 plt.ylabel('Amplitude')
44 plt.subplot(4, 2, 6)
45 plt.plot(t, exponential_signal)
46 plt.title('Exponential Signal')
47 plt.xlabel('Time [s]')
```



**Subject: Programming With Python (01CT1309)**

**Aim:** Practical based on Signal Processing using Scipy

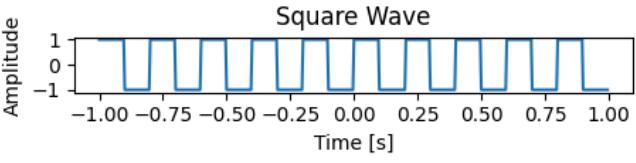
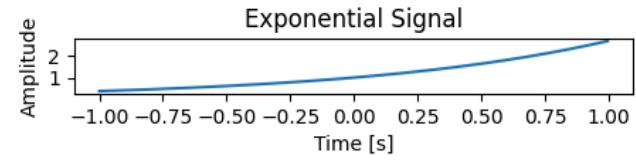
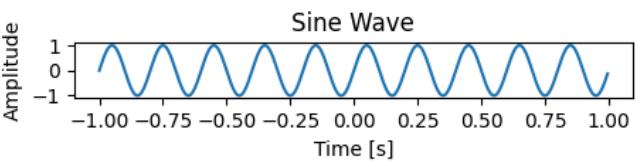
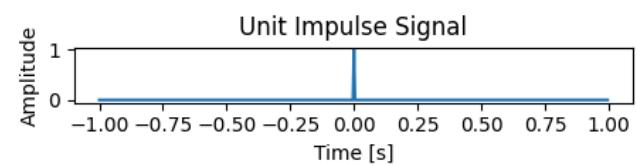
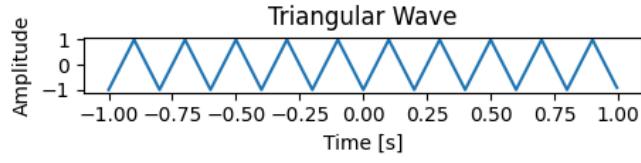
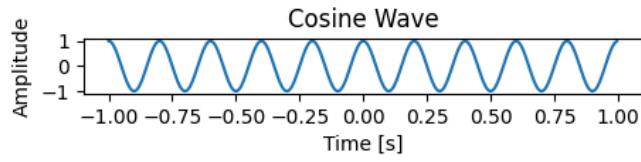
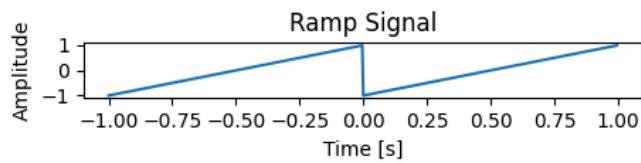
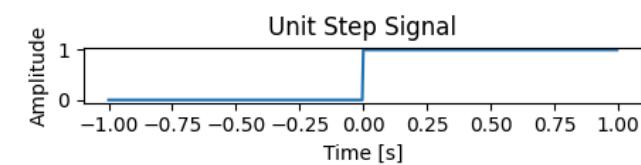
**Experiment No: 12**

**Date:**

**Enrollment No: 92400133037**

```
lab12 > Elementary.py > ...
...
48 plt.ylabel('Amplitude')
49 plt.subplot(4, 2, 7)
50 plt.plot(t, triangular_wave)
51 plt.title('Triangular Wave')
52 plt.xlabel('Time [s]')
53 plt.ylabel('Amplitude')
54 plt.subplot(4, 2, 8)
55 plt.plot(t, square_wave)
56 plt.title('Square Wave')
57 plt.xlabel('Time [s]')
58 plt.ylabel('Amplitude')
59 plt.tight_layout()
60 plt.show()
```

Figure 1





**Marwadi University**  
**Faculty of Engineering & Technology**  
**Department of Information and Communication Technology**

**Subject: Programming With Python (01CT1309)**

**Aim:** Practical based on Signal Processing using Scipy

**Experiment No: 12**

**Date:** **Enrollment No: 92400133037**

### Signal Classification

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
# Parameters
```

```
fs = 20 # Sampling frequency for discrete-time signal
```

```
t_continuous = np.linspace(0, 1, 1000) # Time array for continuous signals
```

```
t_discrete = np.arange(0, 1, 1/fs) # Discrete time array
```

```
# Generate a continuous-time sine wave
```

```
f = 5 # Frequency of the signal
```

```
continuous_signal = np.sin(2 * np.pi * f * t_continuous)
```

```
# Generate a discrete-time sine wave (sampled)
```

```
discrete_time_signal = np.sin(2 * np.pi * f * t_discrete)
```

```
# Discretize the amplitude (quantization) for the continuous-time signal
```

```
num_levels = 4 # Number of quantization levels
```

```
discrete_amplitude_signal = np.round(continuous_signal * (num_levels / 2)) / (num_levels / 2)
```

```
# Discretize both time and amplitude
```



**Marwadi University**  
**Faculty of Engineering & Technology**  
**Department of Information and Communication Technology**

**Subject: Programming With Python (01CT1309)**

**Aim:** Practical based on Signal Processing using Scipy

**Experiment No: 12**

**Date:** **Enrollment No: 92400133037**

```
discrete_time_amplitude_signal = np.round(discrete_time_signal * (num_levels / 2)) / (num_levels / 2)
```

```
# Plot the signals
```

```
plt.figure(figsize=(12, 10))
```

```
# Continuous-Time Signal
```

```
plt.subplot(4, 1, 1)
```

```
plt.plot(t_continuous, continuous_signal)
```

```
plt.title('Continuous-Time Signal (Sine Wave)')
```

```
plt.xlabel('Time [s]')
```

```
plt.ylabel('Amplitude')
```

```
# Discrete-Time Signal
```

```
plt.subplot(4, 1, 2)
```

```
plt.stem(t_discrete, discrete_time_signal, use_line_collection=True)
```

```
plt.title('Discrete-Time Signal (Sampled Sine Wave)')
```

```
plt.xlabel('Time [s]')
```

```
plt.ylabel('Amplitude')
```

```
# Discrete-Amplitude Signal
```

```
plt.subplot(4, 1, 3)
```



**Subject: Programming With Python (01CT1309)**

**Aim:** Practical based on Signal Processing using Scipy

**Experiment No: 12**

**Date:**

**Enrollment No: 92400133037**

```
plt.plot(t_continuous, discrete_amplitude_signal, drawstyle='steps-pre')
```

```
plt.title('Discrete-Amplitude Signal (Quantized Sine Wave)')
```

```
plt.xlabel('Time [s]')
```

```
plt.ylabel('Amplitude')
```

```
lab12 > ⚡ signalPara.py > ...
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  fs=20
5  t_continuous=np.linspace(0,1,1000)
6  t_discrete=np.arange(0,1,1/fs)
7
8  f=5
9  continuous=np.sin(2*np.pi*f*t_continuous)
10 discrete=np.sin(2*np.pi*f*t_discrete)
11
12 #quantisation
13 level=4
14 dis_amp_signal=np.round(continuous*(level/2))/(level/2)
15 dis_time_ampl_signal=np.round(discrete*(level/2))/(level/2)
16
17 plt.figure()
18 plt.subplot(3, 1, 1)
19 plt.plot(t_continuous, continuous)
20 plt.title('Continuous-Time Signal (Sine Wave)')
21 plt.xlabel('Time [s]')
22 plt.ylabel('Amplitude')
23
24 plt.subplot(3, 1, 2)
25 plt.stem(t_discrete, discrete)
26 plt.title('Discrete-Time Signal (Sampled Sine Wave)')
27 plt.xlabel('Time [s]')
28 plt.ylabel('Amplitude')
29
30 plt.subplot(3, 1, 3)
31 plt.plot(t_continuous, dis_amp_signal, drawstyle='steps-pre')
```



**Subject: Programming With Python (01CT1309)**

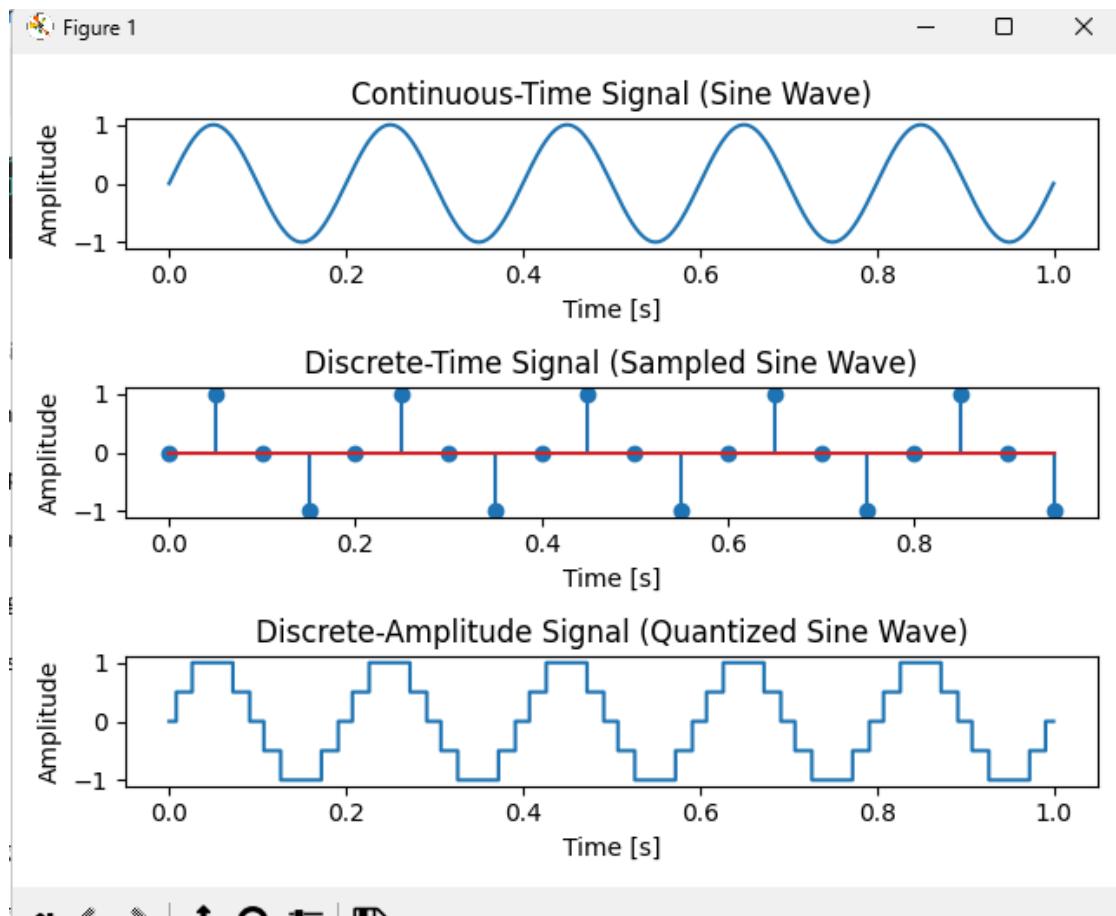
**Aim:** Practical based on Signal Processing using Scipy

**Experiment No: 12**

**Date:**

**Enrollment No: 92400133037**

```
lab12 > signalPara.py > ...
32 plt.title('Discrete-Amplitude Signal (Quantized Sine Wave)')
33 plt.xlabel('Time [s]')
34 plt.ylabel('Amplitude')
35 plt.tight_layout()
36 plt.show()
37
```



### # Discrete signal operation

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```



**Subject: Programming With Python (01CT1309)**

**Aim:** Practical based on Signal Processing using Scipy

**Experiment No: 12**

**Date:**

**Enrollment No: 92400133037**

# Parameters

```
n = np.arange(0, 20) # Discrete time array (0 to 19)
```

```
signal = np.sin(0.2 * np.pi * n) # Example discrete-time signal (sine wave)
```

# Delay the signal by 3 samples

```
delay = 3
```

```
delayed_signal = np.zeros_like(signal)
```

```
delayed_signal[delay:] = signal[:-delay]
```

# Advance the signal by 3 samples

```
advance = 3
```

```
advanced_signal = np.zeros_like(signal)
```

```
advanced_signal[:-advance] = signal[advance:]
```

# Plot the original and shifted signals

```
plt.figure(figsize=(12, 8))
```

# Original Signal

```
plt.subplot(3, 1, 1)
```

```
plt.stem(n, signal, use_line_collection=True)
```

```
plt.title('Original Signal')
```

```
plt.xlabel('n (Discrete Time)')
```

```
plt.ylabel('Amplitude')
```

# Delayed Signal

```
plt.subplot(3, 1, 2)
```



**Subject: Programming With Python (01CT1309)**

**Aim:** Practical based on Signal Processing using Scipy

**Experiment No: 12**

**Date:**

**Enrollment No: 92400133037**

```
plt.stem(n, delayed_signal, use_line_collection=True)

plt.title(f'Delayed Signal (by {delay} samples)')

plt.xlabel('n (Discrete Time)')

plt.ylabel('Amplitude')

# Advanced Signal

plt.subplot(3, 1, 3)

plt.stem(n, advanced_signal, use_line_collection=True)

plt.title(f'Advanced Signal (by {advance} samples)')

plt.xlabel('n (Discrete Time)')

plt.ylabel('Amplitude')

plt.tight_layout()

plt.show()
```



**Subject: Programming With Python (01CT1309)**

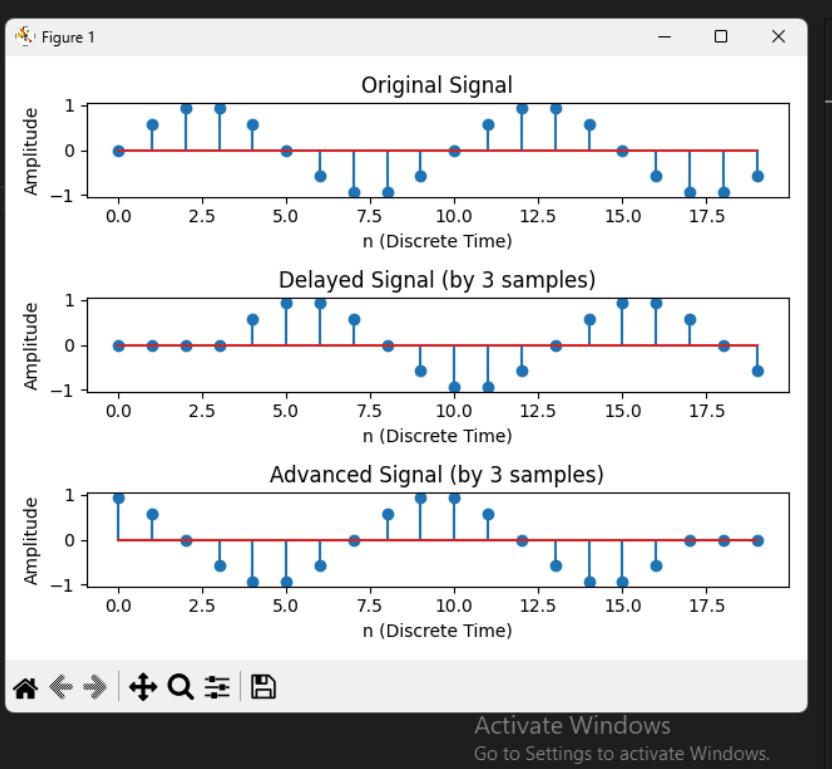
**Aim:** Practical based on Signal Processing using Scipy

**Experiment No: 12**

**Date:**

**Enrollment No: 92400133037**

```
lab12 > DiscreteOp.py > ...
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 n = np.arange(0, 20)
5 signal = np.sin(0.2 * np.pi * n)
6 delay = 3
7 delayed_signal = np.zeros_like(signal)
8 delayed_signal[delay:] = signal[:-delay]
9 advance = 3
10 advanced_signal = np.zeros_like(signal)
11 advanced_signal[:advance] = signal[advance:]
12
13 plt.figure()
14 plt.subplot(3, 1, 1)
15 plt.stem(n, signal)
16 plt.title('Original Signal')
17 plt.xlabel('n (Discrete Time)')
18 plt.ylabel('Amplitude')
19
20 plt.subplot(3, 1, 2)
21 plt.stem(n, delayed_signal)
22 plt.title(f'Delayed Signal (by {delay} samples)')
23 plt.xlabel('n (Discrete Time)')
24 plt.ylabel('Amplitude')
25
26 plt.subplot(3, 1, 3)
27 plt.stem(n, advanced_signal)
28 plt.title(f'Advanced Signal (by {advance} samples)')
29 plt.xlabel('n (Discrete Time)')
30 plt.ylabel('Amplitude')
31 plt.tight_layout()
32 plt.show()
```



### Post Lab Exercise:

- Generate two sine wave signals with frequencies of 5 Hz and 10 Hz, both sampled at 1000 Hz for 1 second. Add the two signals together and plot the result.
- Generate a 5 Hz sine wave and a 10 Hz cosine wave, both sampled at 500 Hz for 2 seconds. Multiply the two signals element-wise and plot the resulting signal.
- Generate a 5 Hz sine wave signal and shift it in time by 0.1 seconds. Plot the original and shifted signals on the same graph for comparison.
- Generate a 10 Hz sine wave and scale its amplitude by a factor of 3. Plot the original and scaled signals together.
- Generate a 5 Hz sine wave and reverse it in time. Plot the original and reversed signals on the same graph.



**Subject: Programming With Python (01CT1309)**

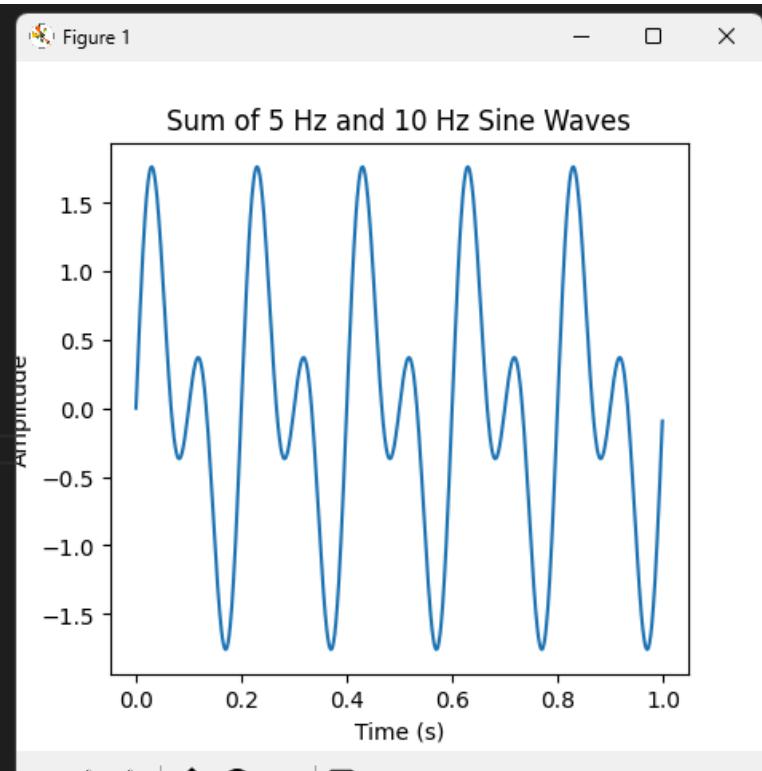
**Aim:** Practical based on Signal Processing using Scipy

**Experiment No: 12**

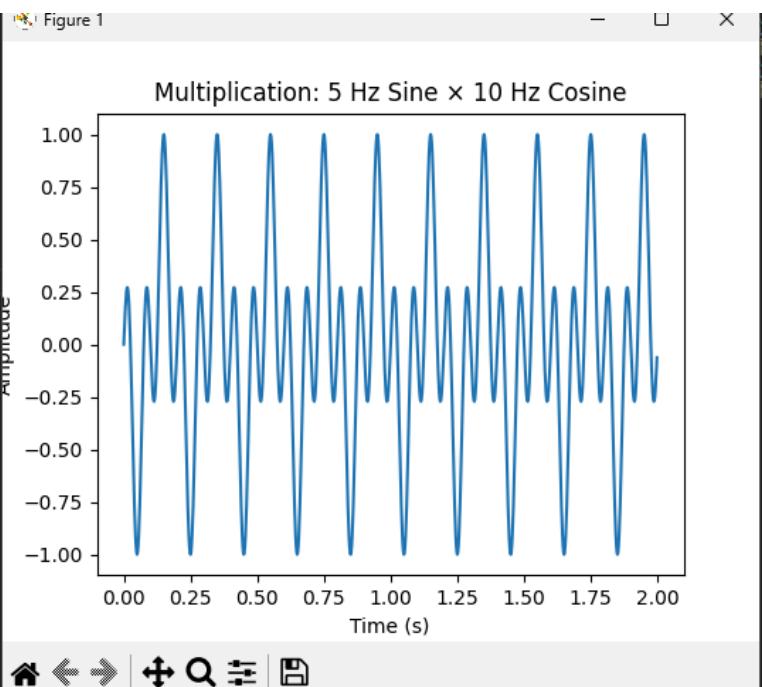
**Date:**

**Enrollment No: 92400133037**

```
lab12 > PostLab.py > ...
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  # Sampling
5  fs = 1000
6  t = np.linspace(0, 1, fs, endpoint=False)
7
8  x1 = np.sin(2*np.pi*5*t)
9  x2 = np.sin(2*np.pi*10*t)
10 sum=x1+x2
11
12 plt.figure()
13 plt.plot(t,sum)
14 plt.title("Sum of 5 Hz and 10 Hz Sine Waves")
15 plt.xlabel("Time (s)")
16 plt.ylabel("Amplitude")
17 plt.show()
```



```
19  #b
20  fs = 500
21  t = np.linspace(0, 2, 2*fs, endpoint=False)
22
23  x1 = np.sin(2*np.pi*5*t)
24  x2 = np.cos(2*np.pi*10*t)
25  mul=x1*x2
26
27  plt.figure()
28  plt.plot(t,mul)
29  plt.title("Multiplication: 5 Hz Sine × 10 Hz Cosine")
30  plt.xlabel("Time (s)")
31  plt.ylabel("Amplitude")
32  plt.show()
```





**Subject: Programming With Python (01CT1309)**

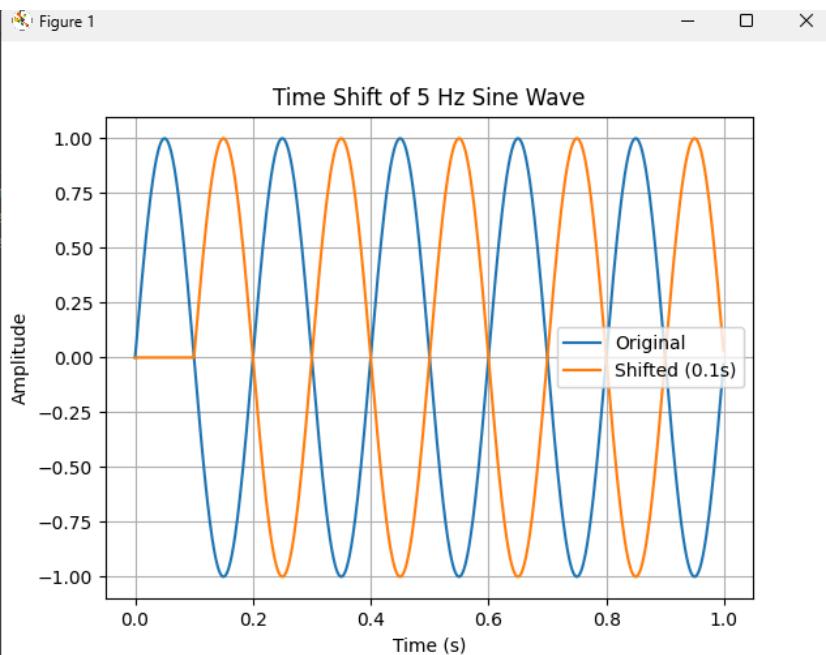
**Aim:** Practical based on Signal Processing using Scipy

**Experiment No: 12**

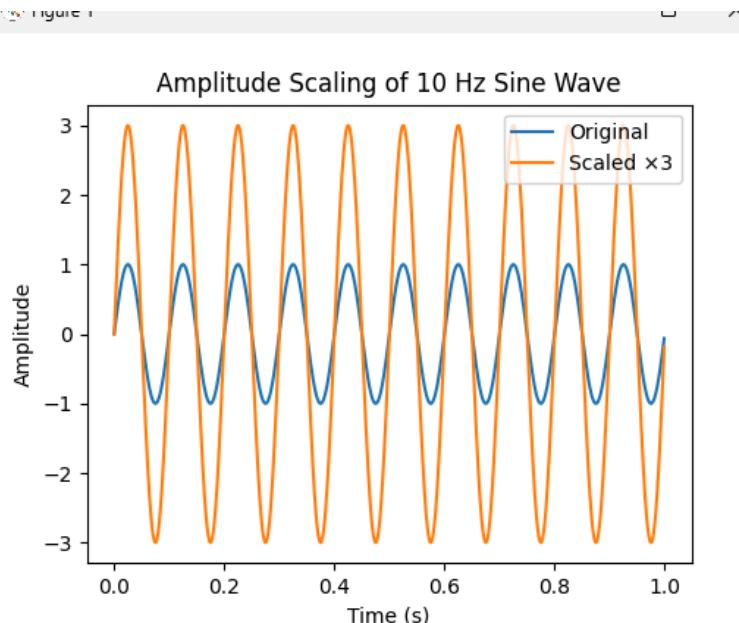
**Date:**

**Enrollment No: 92400133037**

```
lab12 > PostLab.py > ...
34 #c
35 fs = 1000
36 t = np.linspace(0, 1, fs, endpoint=False)
37 x = np.sin(2*np.pi*5*t)
38 timeShift = 0.1
39 #Convert the time shift into number of samples.
40 #Shifting should be in samples because array is
41 #multiplied the shift time by the sampling frequency
42 #how many samples the signal needs to be shifted
43 shift_samples = int(timeShift * fs)
44 #length of x
45 x_shift= np.zeros_like(x)
46 x_shift[shift_samples:] = x[:-shift_samples]
47
48 plt.figure()
49 plt.plot(t, x, label="Original")
50 plt.plot(t, x_shift, label="Shifted (0.1s)")
51 plt.title("Time Shift of 5 Hz Sine Wave")
52 plt.xlabel("Time (s)")
53 plt.ylabel("Amplitude")
54 plt.legend()
55 plt.show()
```



```
lab12 > PostLab.py > ...
50
51 #d
52 fs = 1000
53 t = np.linspace(0, 1, fs, endpoint=False)
54
55 x = np.sin(2*np.pi*10*t)
56 scale= 3 * x
57 plt.figure()
58 plt.plot(t, x, label="Original")
59 plt.plot(t, scale, label="Scaled 3x")
60 plt.title("Amplitude Scaling of 10 Hz Sine Wave")
61 plt.xlabel("Time (s)")
62 plt.ylabel("Amplitude")
63 plt.legend()
64 plt.show()
```





**Subject: Programming With Python (01CT1309)**

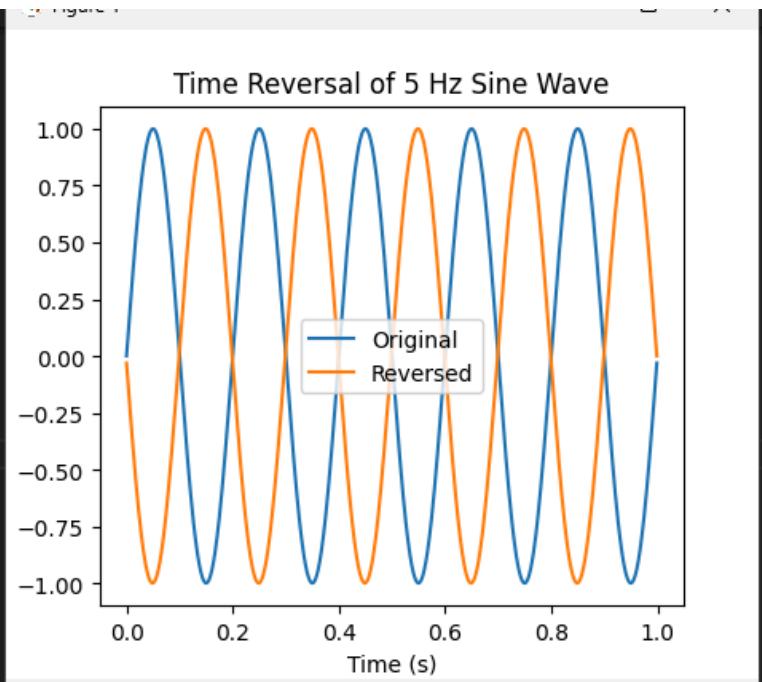
**Aim:** Practical based on Signal Processing using Scipy

**Experiment No: 12**

**Date:**

**Enrollment No: 92400133037**

```
lab12 / PostLab.py ...  
71  
72     #e  
73     fs = 1000  
74     t = np.linspace(0, 1, fs, endpoint=False)  
75  
76     x = np.sin(2*np.pi*5*t)  
77     x_rev = x[::-1]  
78  
79     plt.figure()  
80     plt.plot(t, x, label="Original")  
81     plt.plot(t, x_rev, label="Reversed")  
82     plt.title("Time Reversal of 5 Hz Sine Wave")  
83     plt.xlabel("Time (s)")  
84     plt.ylabel("Amplitude")  
85     plt.legend()  
86     plt.show()
```



**GITHUB LINK:**

[https://github.com/Heer972005/Python\\_Lab](https://github.com/Heer972005/Python_Lab)