Marwadi University (Marwadi Chandarana Group), NAAC A+	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?

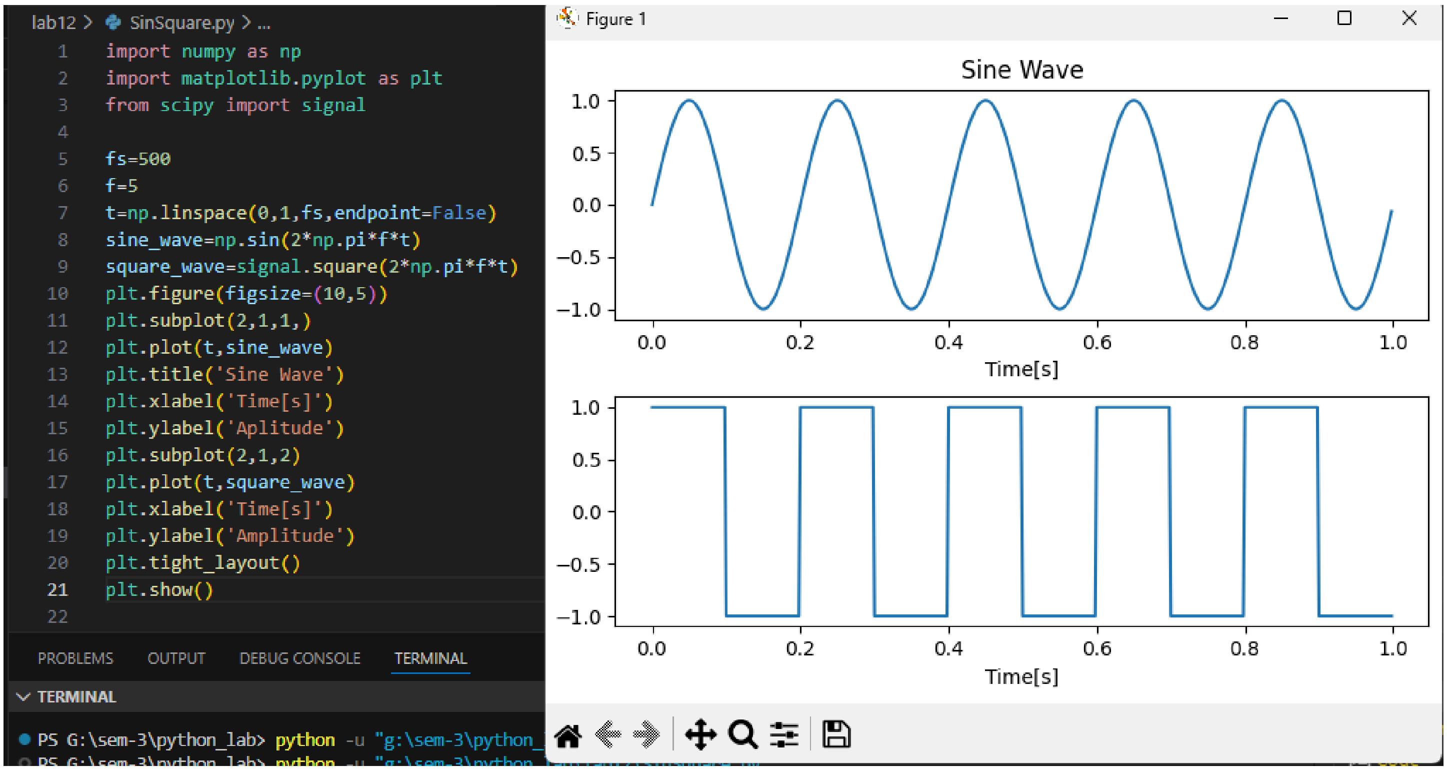
SciPy is a free and open-source Python library used for scientific computing and technical computing. It is a collection of mathematical algorithms and convenience functions built on the NumPy extension of Python. It adds significant power to the interactive Python session by providing the user with 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)
```

```
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()
```

Marwadi University	**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**



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

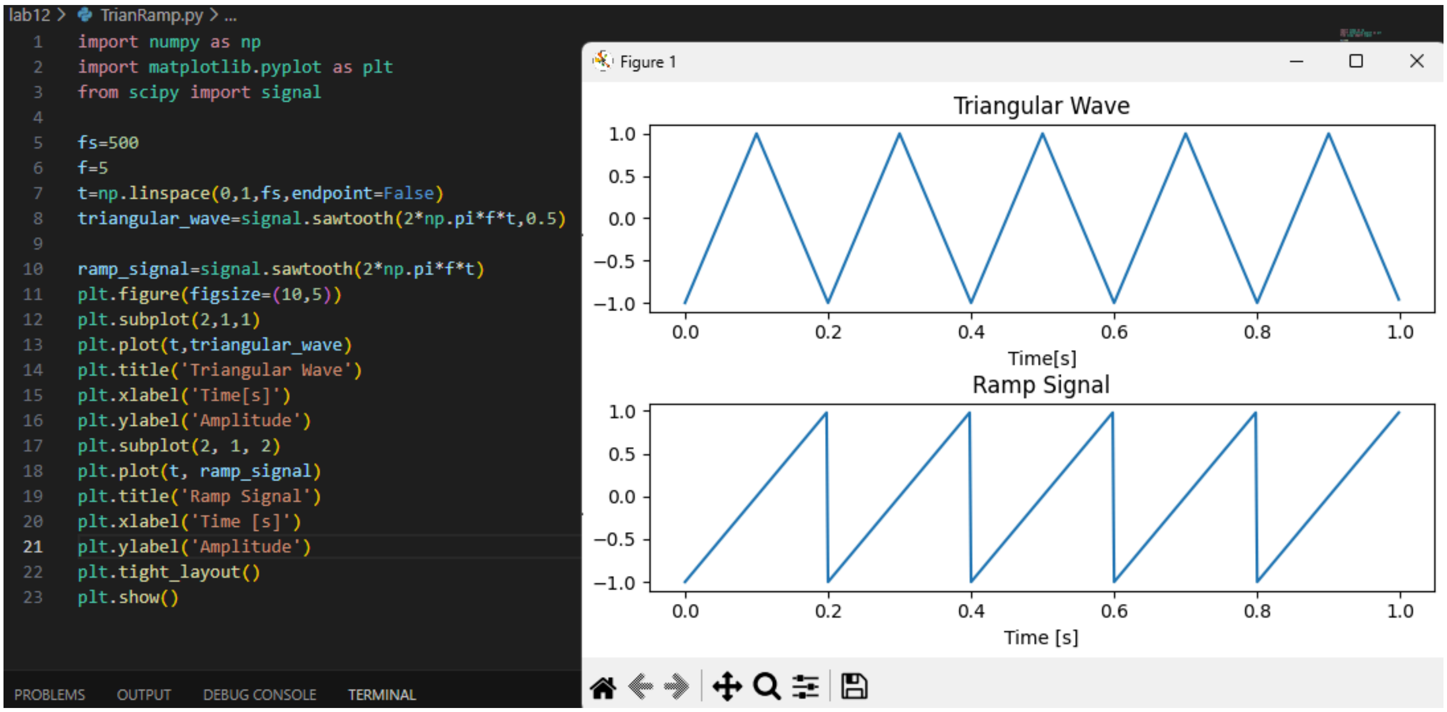
t = np.linspace(0, 1, fs, endpoint=False) # Time array

Create a triangular wave signal using scipy

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

```
# 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()
```

	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**



#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)

```
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')
```

```
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]')
```

```
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()
```

Marwadi University Marwadi Chandarana Group NAAC A+	**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**

lab12 > Elementary.py > ...

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
fs=500
t=np.linspace(-1, 1, fs, endpoint=False)
unit_step=np.heaviside(t, 1)
unit_impulse=np.zeros_like(t)
unit_impulse[fs//2]=1
ramp_signal=signal.sawtooth(2*np.pi*t,1)
f_sine=5
sine_wave=np.sin(2*np.pi*f_sine*t)
f_cosine=5
cosine_wave=np.cos(2*np.pi*f_cosine*t)
exponential_signal=np.exp(t)
triangular_wave=signal.sawtooth(2*np.pi*5*t,0.5)
square_wave = signal.square(2 * np.pi * 5 * t)
# Plot the signals
plt.figure()
plt.subplot(4, 2, 1)
plt.plot(t, unit_step)
plt.title('Unit Step Signal')
plt.xlabel('Time [s]')
plt.ylabel('Amplitude')
```

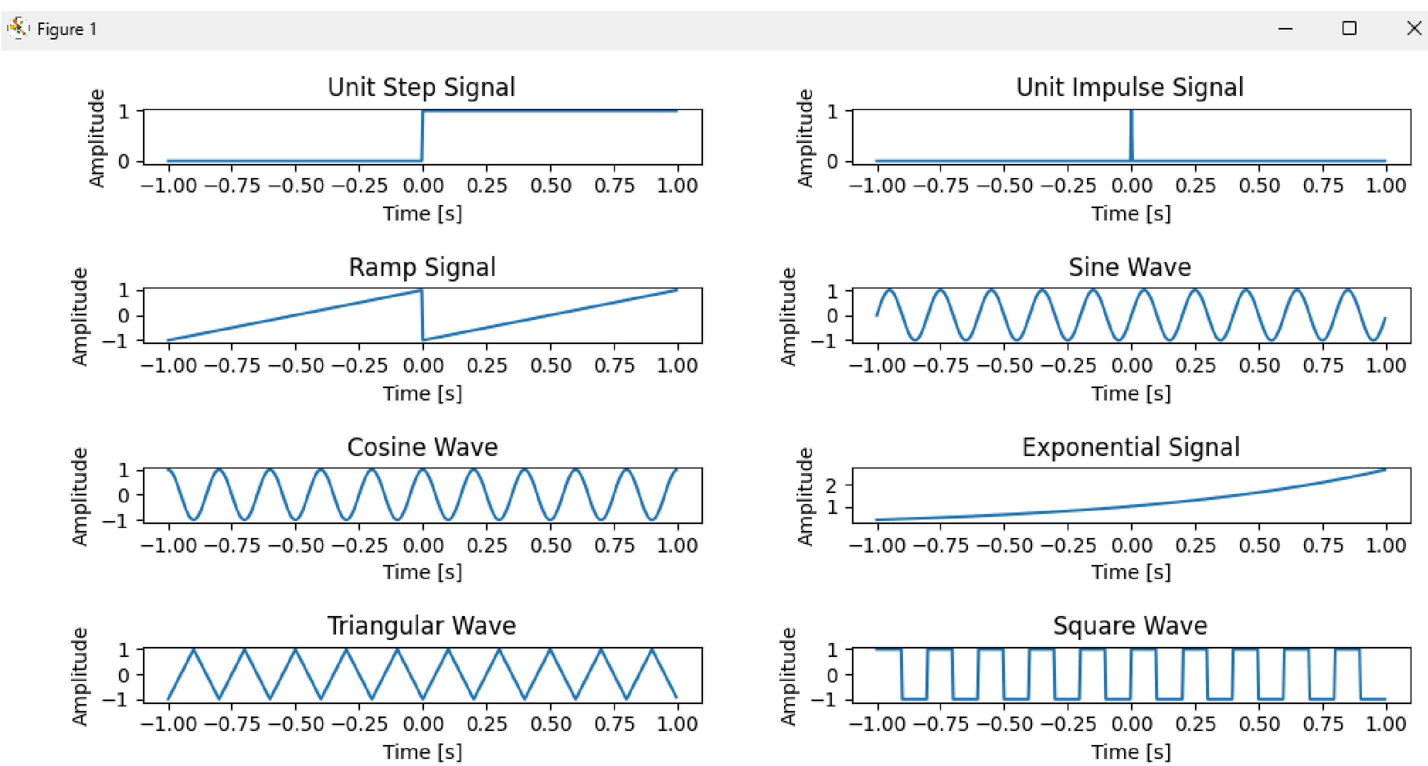
lab12 > Elementary.py > ...

```
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]')
plt.ylabel('Amplitude')
plt.subplot(4, 2, 6)
plt.plot(t, exponential_signal)
plt.title('Exponential Signal')
plt.xlabel('Time [s]')
```

Marwadi University	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**

lab12 > Elementary.py > ...

```
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()
```



Marwadi University Marwadi Chandarana Group NAAC A+	**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
```

```
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)
```

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

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

plt.xlabel('Time [s]')

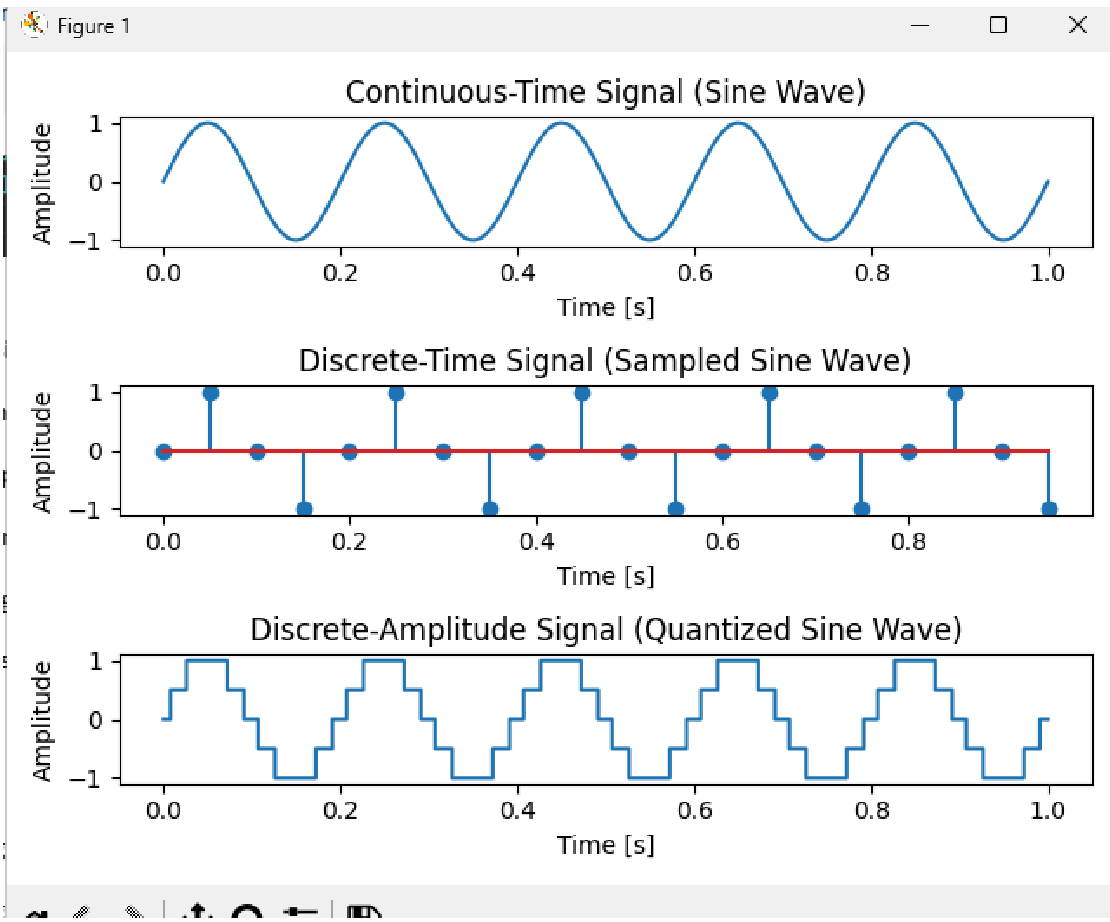
plt.ylabel('Amplitude')

lab12 > signalPara.py > ...

```python
import numpy as np
import matplotlib.pyplot as plt

fs=20
t_continuous=np.linspace(0,1,1000)
t_discrete=np.arange(0,1,1/fs)

f=5
continuous=np.sin(2*np.pi*f*t_continuous)
discrete=np.sin(2*np.pi*f*t_discrete)

#quantisation
level=4
dis_amp_signal=np.round(continuous*(level/2))/(level/2)
dis_time_ampl_signal=np.round(discrete*(level/2))/(level/2)

plt.figure()
plt.subplot(3, 1, 1)
plt.plot(t_continuous, continuous)
plt.title('Continuous-Time Signal (Sine Wave)')
plt.xlabel('Time [s]')
plt.ylabel('Amplitude')

plt.subplot(3, 1, 2)
plt.stem(t_discrete, discrete)
plt.title('Discrete-Time Signal (Sampled Sine Wave)')
plt.xlabel('Time [s]')
plt.ylabel('Amplitude')

plt.subplot(3, 1, 3)
plt.plot(t_continuous, dis_amp_signal, drawstyle='steps-pre')
```

lab12 > signalPara.py > ...

```
plt.title('Discrete-Amplitude Signal (Quantized Sine Wave)')
plt.xlabel('Time [s]')
plt.ylabel('Amplitude')
plt.tight_layout()
plt.show()

```



 # Discrete signal operation

import numpy as np

import matplotlib.pyplot as plt

```
# 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)
```

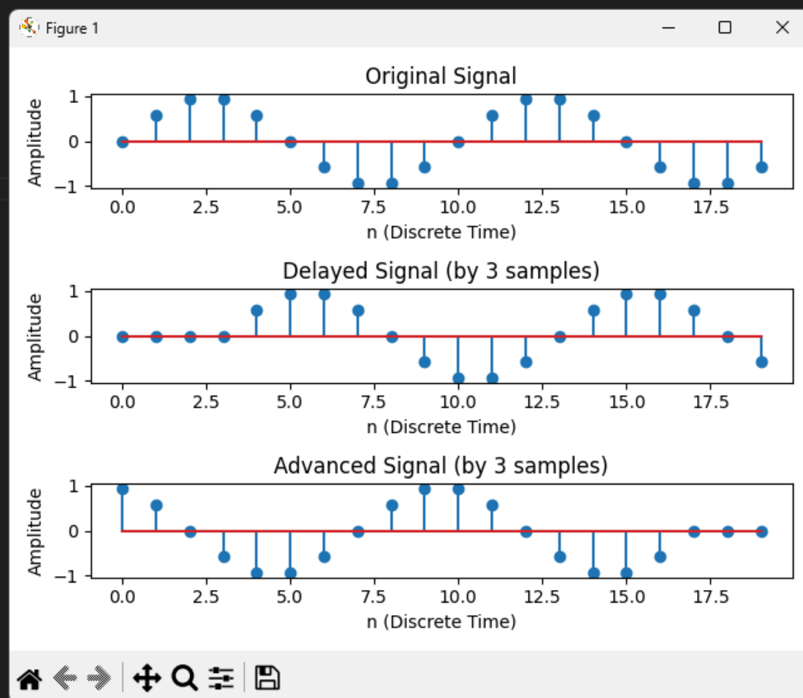
```
plt.stem(n, delayed_signal, use_line_collection=True)

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()
```

	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**

lab12 > DiscreteOp.py > ...

```python
import numpy as np
import matplotlib.pyplot as plt

n = np.arange(0, 20)
signal = np.sin(0.2 * np.pi * n)
delay = 3
delayed_signal = np.zeros_like(signal)
delayed_signal[delay:] = signal[:-delay]
advance = 3
advanced_signal = np.zeros_like(signal)
advanced_signal[:-advance] = signal[advance:]

plt.figure()
plt.subplot(3, 1, 1)
plt.stem(n, signal)
plt.title('Original Signal')
plt.xlabel('n (Discrete Time)')
plt.ylabel('Amplitude')

plt.subplot(3, 1, 2)
plt.stem(n, delayed_signal)
plt.title(f'Delayed Signal (by {delay} samples)')
plt.xlabel('n (Discrete Time)')
plt.ylabel('Amplitude')

plt.subplot(3, 1, 3)
plt.stem(n, advanced_signal)
plt.title(f'Advanced Signal (by {advance} samples)')
plt.xlabel('n (Discrete Time)')
plt.ylabel('Amplitude')
plt.tight_layout()
plt.show()
```



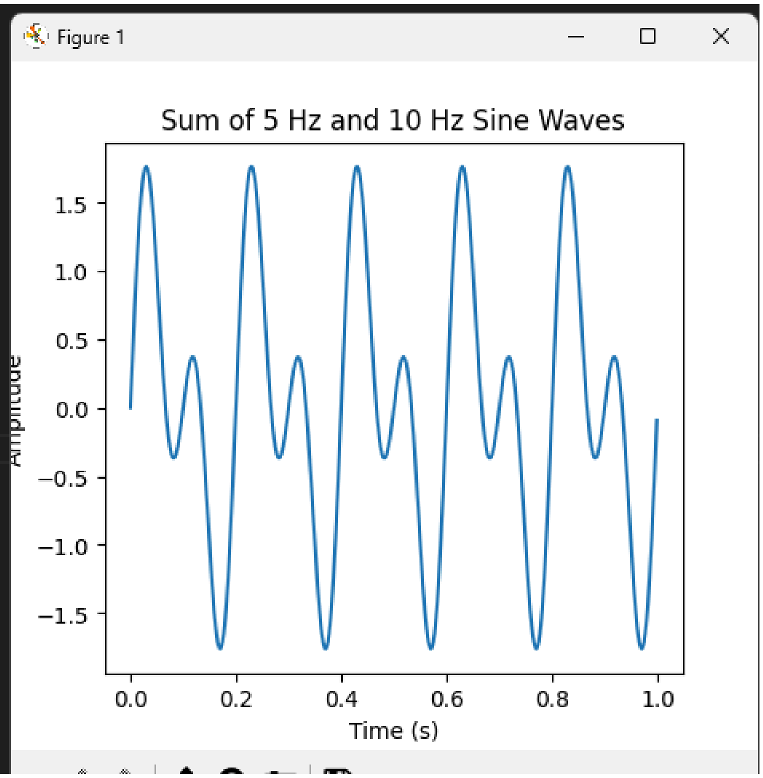
Post Lab Exercise:

a. 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.
b. 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.
c. 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.
d. Generate a 10 Hz sine wave and scale its amplitude by a factor of 3. Plot the original and scaled signals together.
e. Generate a 5 Hz sine wave and reverse it in time. Plot the original and reversed signals on the same graph.

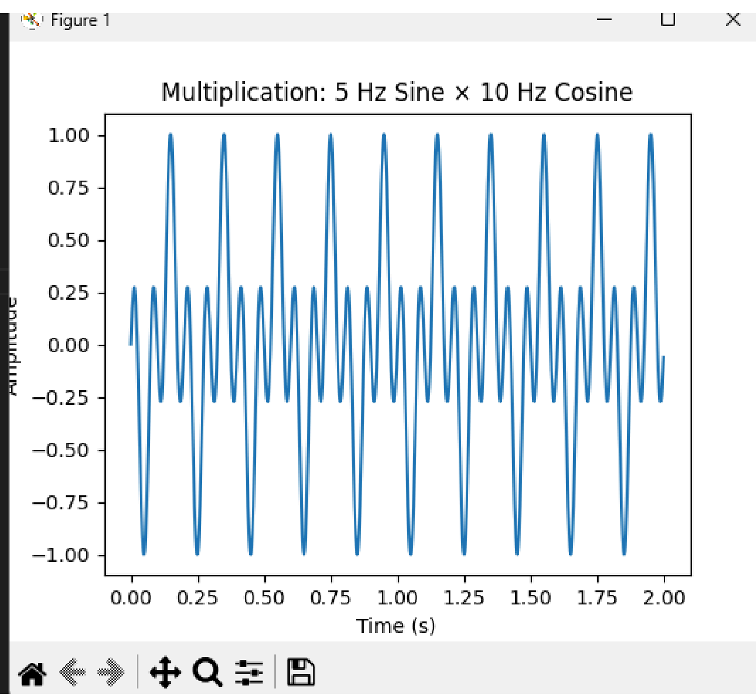
Marwadi University, Marwadi Chandarana Group, NAAC A+	Marwadi University Faculty of Engineering & Technology Department of Information and Communication Technology	
Subject: Programming With Python (01CT1309)	**Aim:** Practical based on Signal Processing using Scipy	
Experiment No: 12	**Date:**	**Enrollment No: 92400133037**

lab12 > PostLab.py > ...

```python
import numpy as np
import matplotlib.pyplot as plt

# Sampling
fs = 1000
t = np.linspace(0, 1, fs, endpoint=False)

x1 = np.sin(2*np.pi*5*t)
x2 = np.sin(2*np.pi*10*t)
sum=x1+x2

plt.figure()
plt.plot(t,sum)
plt.title("Sum of 5 Hz and 10 Hz Sine Waves")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.show()
```



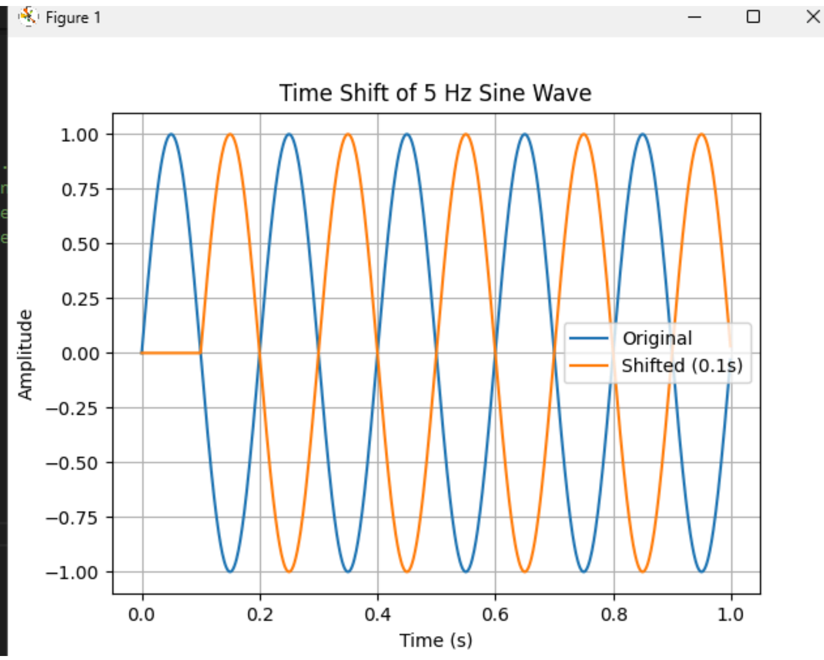

```python
#b
fs = 500
t = np.linspace(0, 2, 2*fs, endpoint=False)

x1 = np.sin(2*np.pi*5*t)
x2 = np.cos(2*np.pi*10*t)
mul=x1*x2

plt.figure()
plt.plot(t,mul)
plt.title("Multiplication: 5 Hz Sine × 10 Hz Cosine")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.show()
```



Marwadi University Marwadi Chandarana Group NAAC A+	**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**

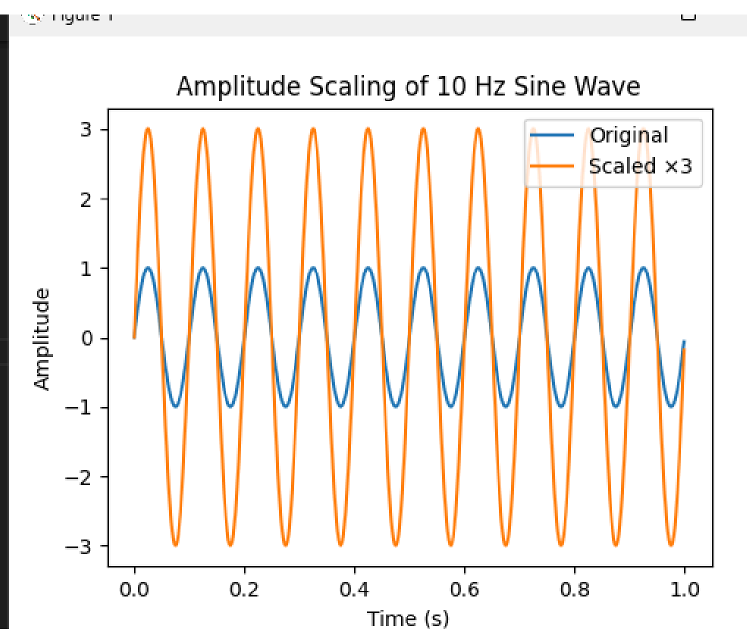
lab12 > PostLab.py > ...

```python
#c
fs = 1000
t = np.linspace(0, 1, fs, endpoint=False)
x = np.sin(2*np.pi*5*t)
timeShift = 0.1
#Convert the time shift into number of samples.
#Shifting should be in samples because array in
#multiply the shift time by the sampling freque
#how many samples the signal needs to be shifte
shift_samples = int(timeShift * fs)
#length of x
x_shift= np.zeros_like(x)
x_shift[shift_samples:] = x[:-shift_samples]

plt.figure()
plt.plot(t, x, label="Original")
plt.plot(t, x_shift, label="Shifted (0.1s)")
plt.title("Time Shift of 5 Hz Sine Wave")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.legend()
plt.show()
```

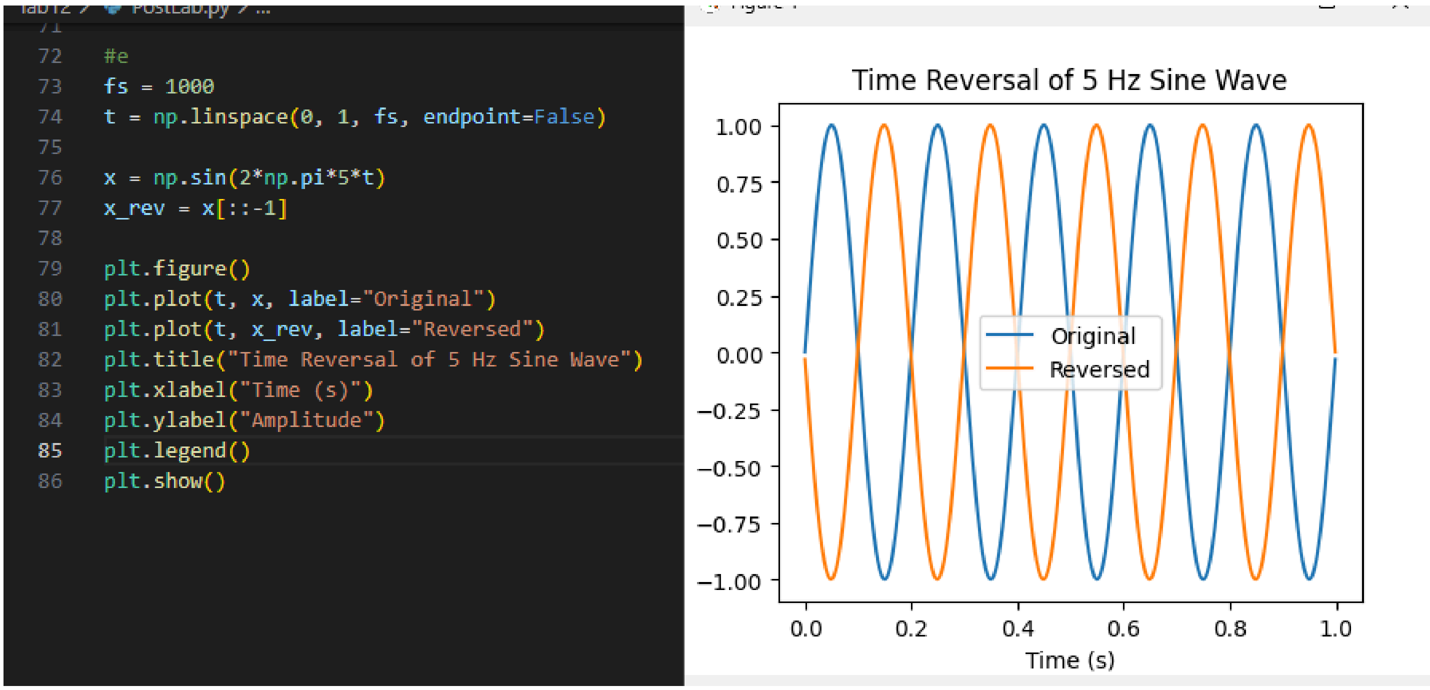


lab12 > PostLab.py > ...

```python
#d
fs = 1000
t = np.linspace(0, 1, fs, endpoint=False)

x = np.sin(2*np.pi*10*t)
scale= 3 * x
plt.figure()
plt.plot(t, x, label="Original")
plt.plot(t, scale, label="Scaled ×3")
plt.title("Amplitude Scaling of 10 Hz Sine Wave")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.legend()
plt.show()
```



	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**



GITHUB LINK:
https://github.com/Heer972005/Python_Lab