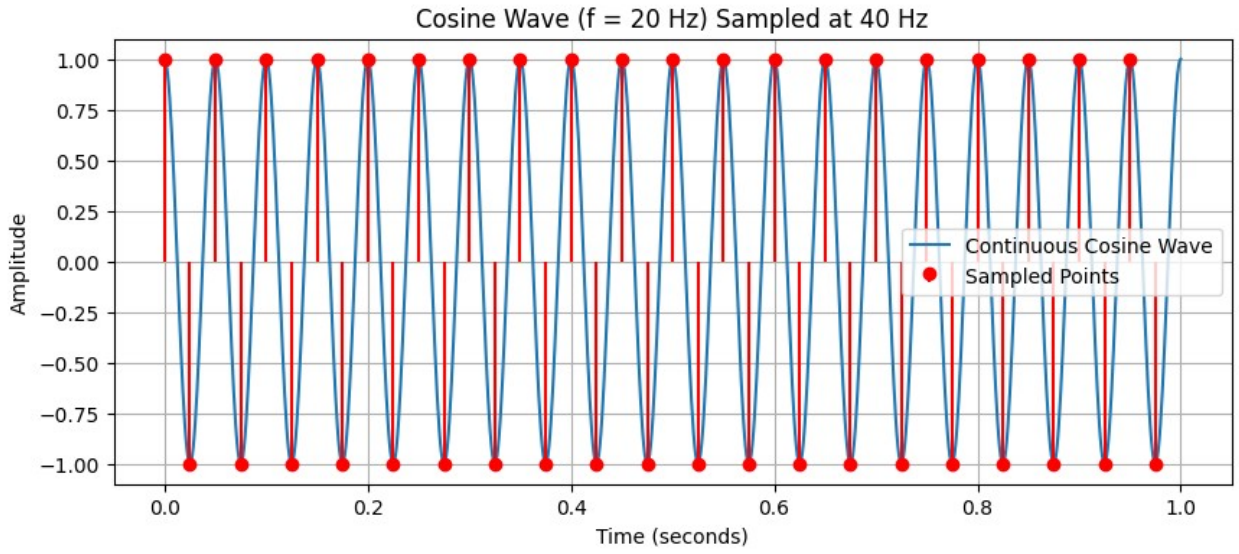
Solve the tasks for:

- sampling and reconstruction
- coding and decoding Variant 2

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

# Task 1: Sampling and Reconstruction, variant two
# Task: Analyze a cosine wave with frequency f = 20 Hz, sampled at fs
= 25 Hz.

f = 20
fs = 40
duration = 1

# Generate time vectors
t_continuous = np.linspace(0, duration, 1000)  # continuous time
points
t_sampled = np.arange(0, duration, 1/fs)  # sampled time points

# Generate signals
x_continuous = np.cos(2 * np.pi * f * t_continuous)  # continuous
signal
x_sampled = np.cos(2 * np.pi * f * t_sampled)  # sampled signal

plt.figure(figsize=(10, 4))
plt.plot(t_continuous, x_continuous, label='Continuous Cosine Wave')
plt.stem(t_sampled, x_sampled, linefmt='r', markerfmt='ro', basefmt='
', label='Sampled Points')
plt.title(f'Cosine Wave (f = {f} Hz) Sampled at {fs} Hz')
plt.xlabel('Time (seconds)')
plt.ylabel('Amplitude')
plt.legend()
plt.grid(True)
plt.show()
```




```
# Task 2: Delta Encoding and Decoding, variant two
# Task: Encode and decode the signal [8, 10, 12, 12, 14] using delta
coding.

signal = [8, 10, 12, 12, 14]

# Delta Encoding
encoded_signal = [signal[0]]
for i in range(1, len(signal)):
    encoded_signal.append(signal[i] - signal[i - 1])

print("Delta Encoded Signal:", encoded_signal)

# Delta Decoding
decoded_signal = [encoded_signal[0]]
for i in range(1, len(encoded_signal)):
    decoded_signal.append(decoded_signal[i - 1] + encoded_signal[i])

print("Delta Decoded Signal:", decoded_signal)

assert decoded_signal == signal, "Decoding failed: Decoded signal does
not match the original."

plt.figure(figsize=(10, 4))
plt.plot(encoded_signal, label='Encoded Signal')
plt.plot(decoded_signal, label='Decoded Signal')
plt.title('Original and Decoded Signals')
plt.xlabel('Sample Index')
plt.ylabel('Amplitude')
plt.legend()
plt.grid(True)
plt.show()
```

```
Delta Encoded Signal: [8, 2, 2, 0, 2]
Delta Decoded Signal: [8, 10, 12, 12, 14]
```