Discrete Fourier Transform (DFT)

The **Discrete Fourier Transform (DFT)** is a mathematical technique used to analyze the frequency content of discrete-time signals. Unlike the **DTFT**, which provides a continuous frequency spectrum, the DFT offers a discrete set of frequency points and is suitable for computational purposes, especially when working with digital signals.

Relationship to DTFT

While the DTFT is continuous and evaluates the signal at all frequencies, the DFT samples the DTFT at equally spaced frequency intervals. This allows the DFT to represent the signal's frequency content with a finite number of points, making it easier to implement in computers. In practice, the DFT is used more often due to its compatibility with digital systems and limited computational resources.

DFT in Practice

The DFT is typically computed using the **Fast Fourier Transform (FFT)**, an efficient algorithm that reduces the computational complexity of performing the DFT. The FFT allows us to compute the DFT quickly for large datasets, which is crucial in applications like digital signal processing and real-time analysis.

Visualization

The DFT results in a **discrete frequency spectrum** that shows how much of each frequency is present in the original signal. By plotting the magnitude and phase of the DFT, we can see which frequencies dominate the signal. This information is useful in tasks such as:

- Signal compression,
- Noise reduction,
- Signal reconstruction.

The DFT can be computed and visualized using the FFT algorithm in Python, as shown in the earlier code, to display the discrete frequency spectrum of signals.

Discrete-Time Fourier Transform (DTFT)

The Discrete-Time Fourier Transform (DTFT) is a mathematical tool used to analyze signals in the frequency domain. It is the frequency domain representation of a discrete-time signal and provides a continuous frequency spectrum of the signal. The DTFT is particularly useful in digital signal processing for understanding how different frequency components make up a signal.

Relationship to DFT

The DTFT provides a continuous frequency spectrum, but in practice, we often compute the Discrete Fourier Transform (DFT), which is a sampled version of the DTFT at specific frequency points. While the DTFT evaluates the signal at all frequencies, the DFT samples it at evenly spaced intervals, suitable for computational purposes.

DTFT in Practice

In digital signal processing, the DTFT is typically computed using the Fast Fourier Transform (FFT), which efficiently calculates a sampled version of the DTFT. The FFT provides a finite set of frequency components (as in the DFT), but it gives an approximation of the continuous frequency spectrum.

Visualization

By plotting the magnitude and phase of the DTFT, we can visualize how the energy of the signal is distributed across different frequency components. This is useful for tasks like:

- Filtering specific frequencies,
- Analyzing periodicity,
- Detecting hidden patterns in signals.

The DTFT can be computed using FFT algorithms in Python (as shown in the code provided earlier) to visualize the frequency content of signals.

Question 1

(a) Generate a basic sinusoidal signal in the time domain.

(For example, you may generate a sine wave with a frequency of 5Hz, sampled at 1000Hz.)

(b) Plot the time-domain waveform of the signal.

(c) Compute the Discrete-Time Fourier Transform (DTFT) and plot the continuous frequency spectrum.

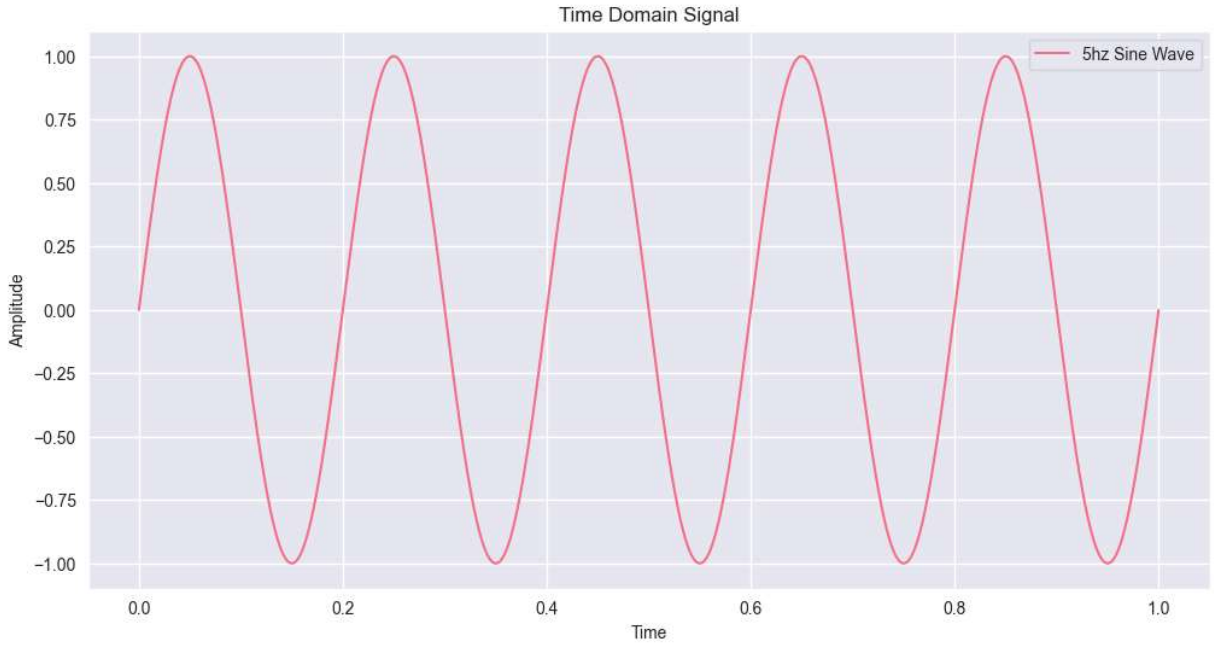
(d) Compute the Discrete Fourier Transform (DFT) and plot the discrete frequency spectrum.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

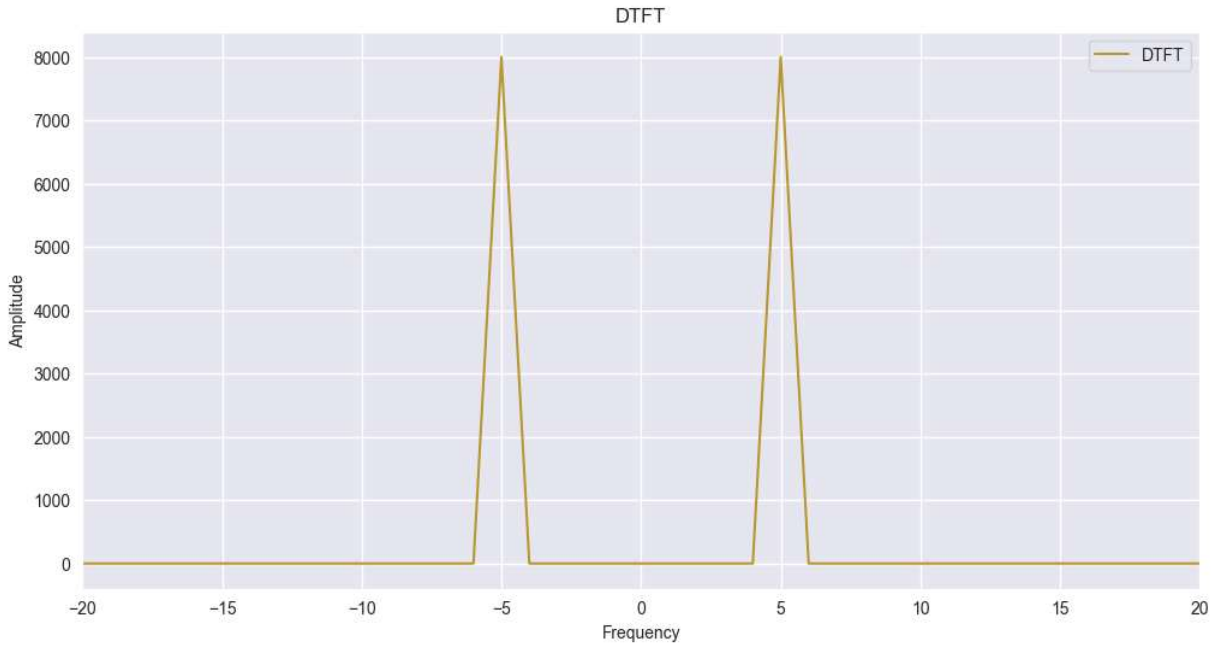
In [2]:
```
frequency = 5
sampling_rate = 16000
duration = 1
```

In [3]:
```
t = np.arange(0, duration, 1/sampling_rate)
signal = np.sin(2*np.pi*frequency*t) #generate sinusodial signals with frequency 5H
```

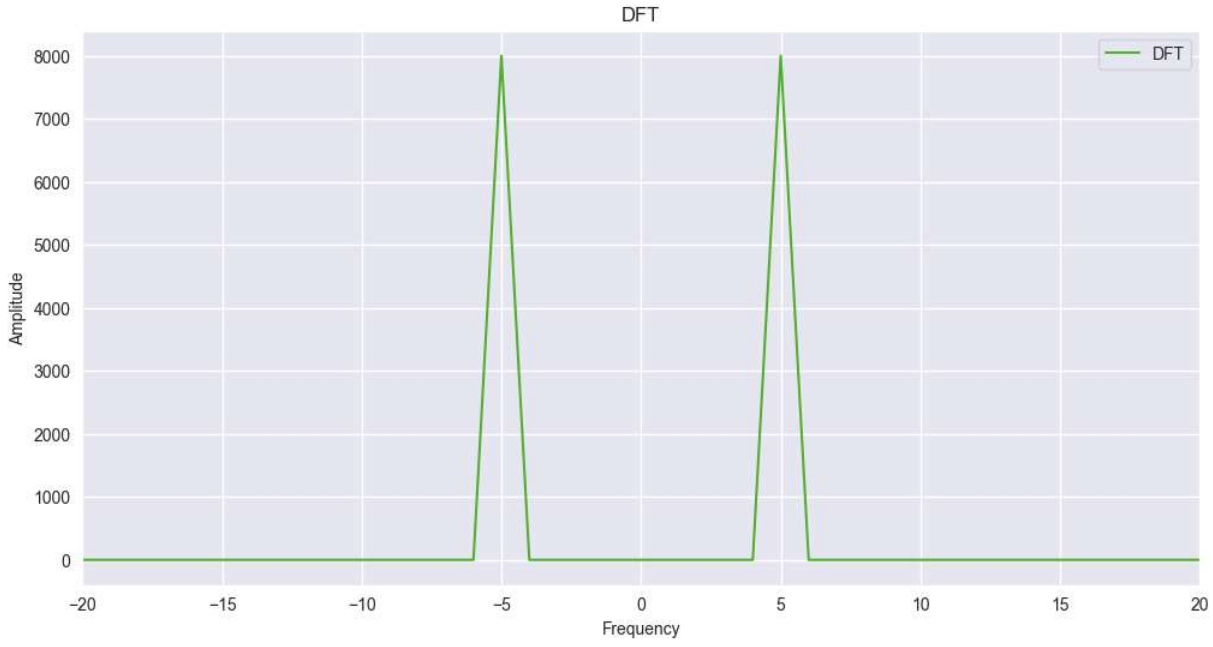
In [4]:
```
sns.set_style('darkgrid')
colors = sns.color_palette('husl')

plt.figure(figsize=(12, 6))
sns.lineplot(x = t, y= signal, label = "5hz Sine Wave", color = colors[0])
plt.title("Time Domain Signal")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.legend()
plt.show()
```



In [5]:
```
# Discrete-Time Fourier Transform (DTFT)

frequencies = np.fft.fftfreq(len(t), d = 1/sampling_rate)
dtft = np.fft.fft(signal)
plt.figure(figsize=(12, 6))
sns.lineplot(x = frequencies, y = np.abs(dtft), label = "DTFT", color = colors[1])
plt.title("DTFT")
plt.xlabel("Frequency")
plt.ylabel("Amplitude")
plt.xlim(-20, 20)
plt.legend()
plt.show()
```



In [6]:
```
# Discrete Fourier Transform

dft = np.fft.fft(signal)
dft_frequencies = np.fft.fftfreq(len(signal), d = 1/sampling_rate)

plt. figure(figsize=(12, 6))
sns.lineplot(x = dft_frequencies, y = np.abs(dft), label = "DFT", color = colors[2]
plt.title("DFT")
plt.xlabel("Frequency")
plt.ylabel("Amplitude")
plt.xlim(-20, 20)
plt.legend()
plt.show()
```



Question 2

(a) Generate a composite signal by adding two or more sinusoidal signals of different frequencies and amplitudes.

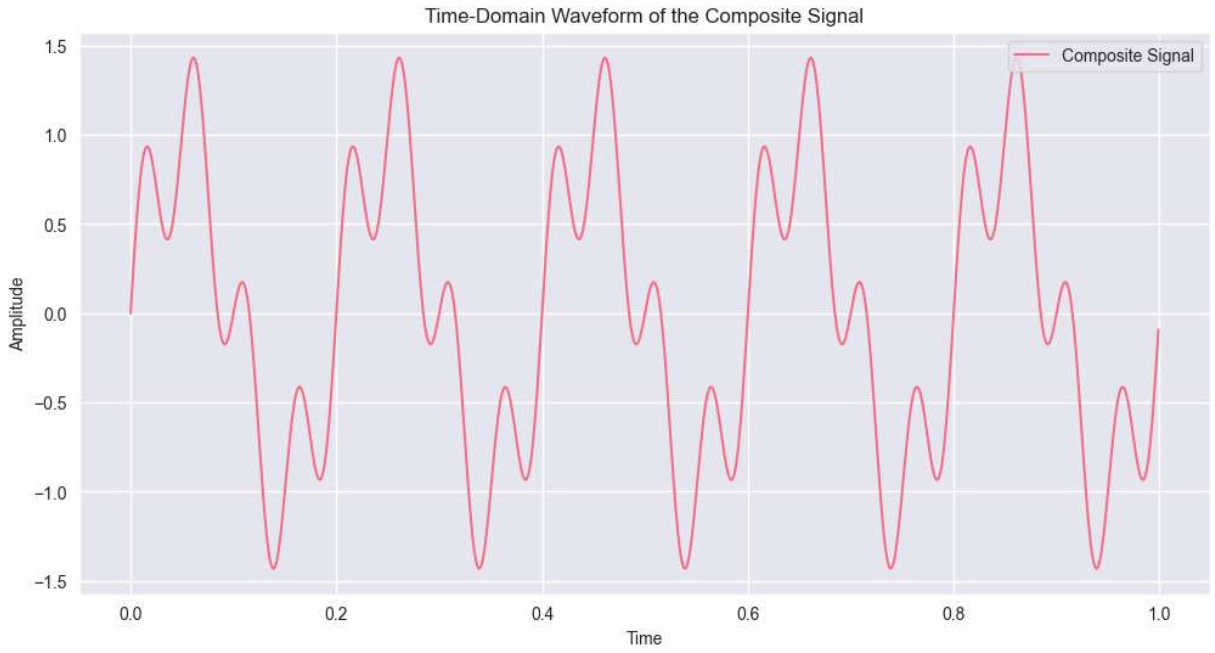
(b) Plot the time-domain waveform of the composite signal.

(c) Compute the Discrete-Time Fourier Transform (DTFT) and plot the continuous frequency spectrum.

(d) Compute the Discrete Fourier Transform (DFT) and plot the discrete frequency spectrum.

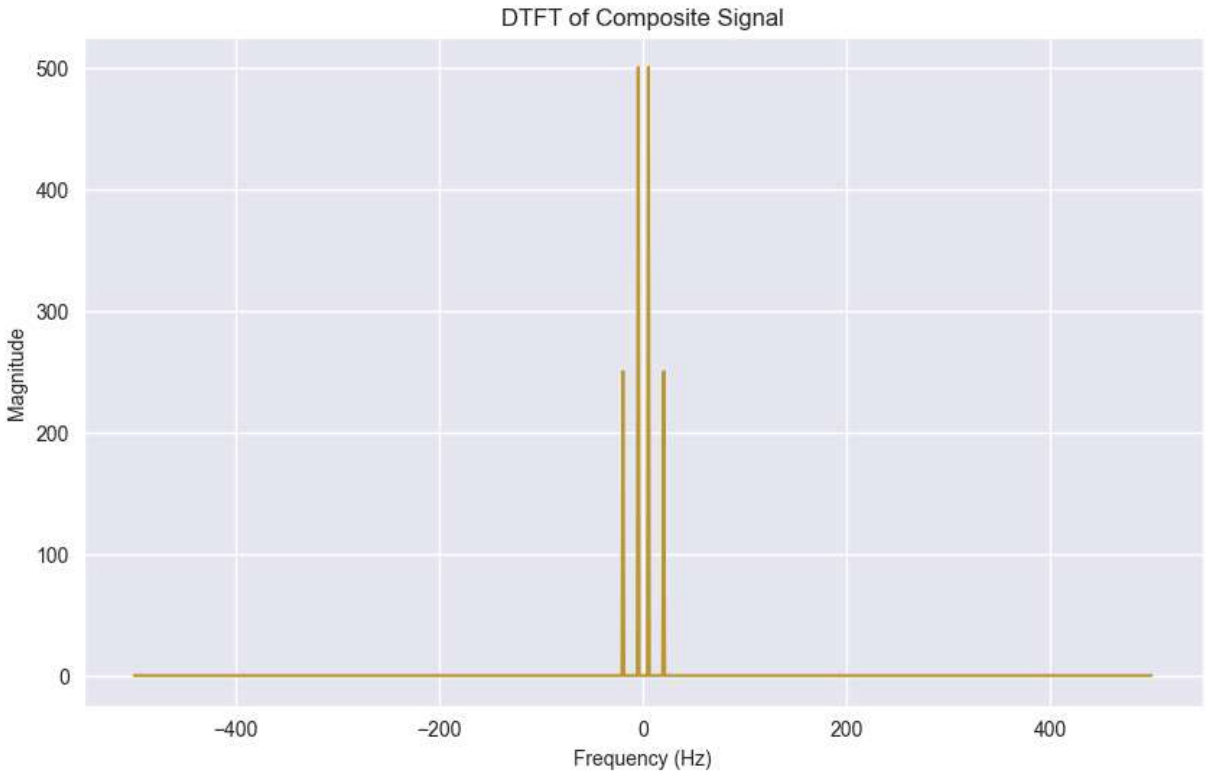
In [7]:
```
fs = 1000
t = np.linspace(0, 1, fs, endpoint=False)

f1 = 5
f2 = 20
A1 = 1
A2 = 0.5

signal1 =A1*np.sin(2*np.pi*f1*t)
signal2 = A2*np.sin(2*np.pi*f2*t)

composite_signal = signal1 + signal2
```

In [11]:
```
sns.set_style("darkgrid")
color = sns.color_palette("hsv")

plt.figure(figsize=(12, 6))
sns.lineplot(x = t, y = composite_signal, label = "Composite Signal", color = color
plt.title("Time-Domain Waveform of the Composite Signal")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.legend()
plt.show()
```

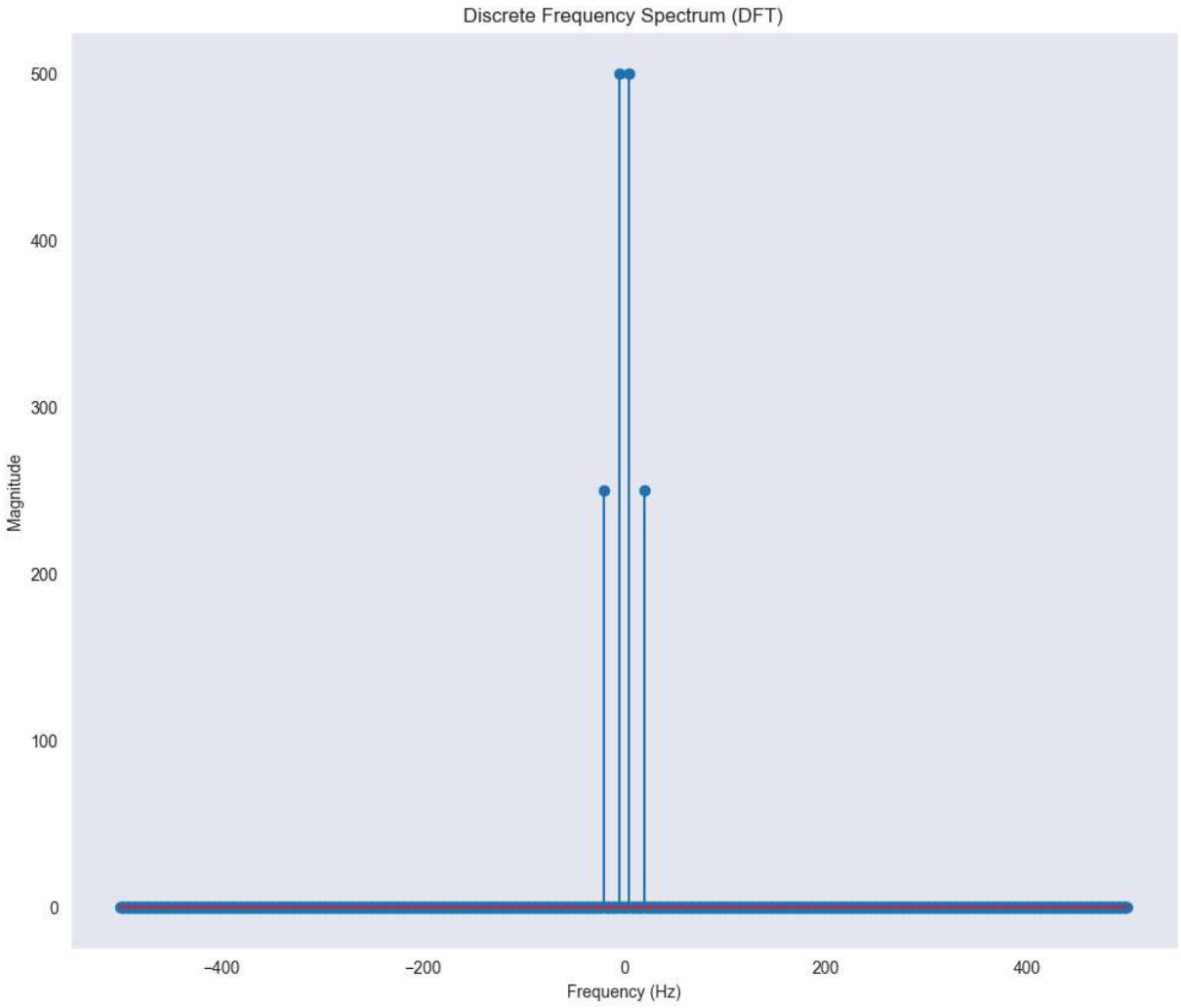


In [12]:
```
# Compute DTFT

N = len(composite_signal)
f = np.fft.fftfreq(N, d = 1/fs)

dtft = np.fft.fft(composite_signal)
```

In [13]:
```
plt.figure(figsize=(10, 6))
sns.lineplot(x = f, y = np.abs(dtft), color = colors[1])
plt.xlabel('Frequency (Hz)')
plt.ylabel('Magnitude')
plt.title('DTFT of Composite Signal')
plt.show()
```



In [19]:
```
dft = np.fft.fft(composite_signal)

plt.figure(figsize=(12,10))
plt.stem(f, np.abs(dft))
plt.title('Discrete Frequency Spectrum (DFT)')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Magnitude')
plt.grid()
plt.show()
```



(3) Question 3

(a) Generate an exponentially decaying signal. (b) Plot the time-domain waveform.

(c) Compute the Discrete-Time Fourier Transform (DTFT) and plot the con- tinuous frequency spectrum.

(d) Compute the Discrete Fourier Transform (DFT) and plot the discrete frequency spectrum.

(e) Analyze the relationship between the time-domain waveform and the frequency-domain representation.

An **exponentially decaying signal** is a signal that decreases in amplitude over time, typically following the mathematical form:

$$x(t) = A \cdot e^{-\alpha t}$$

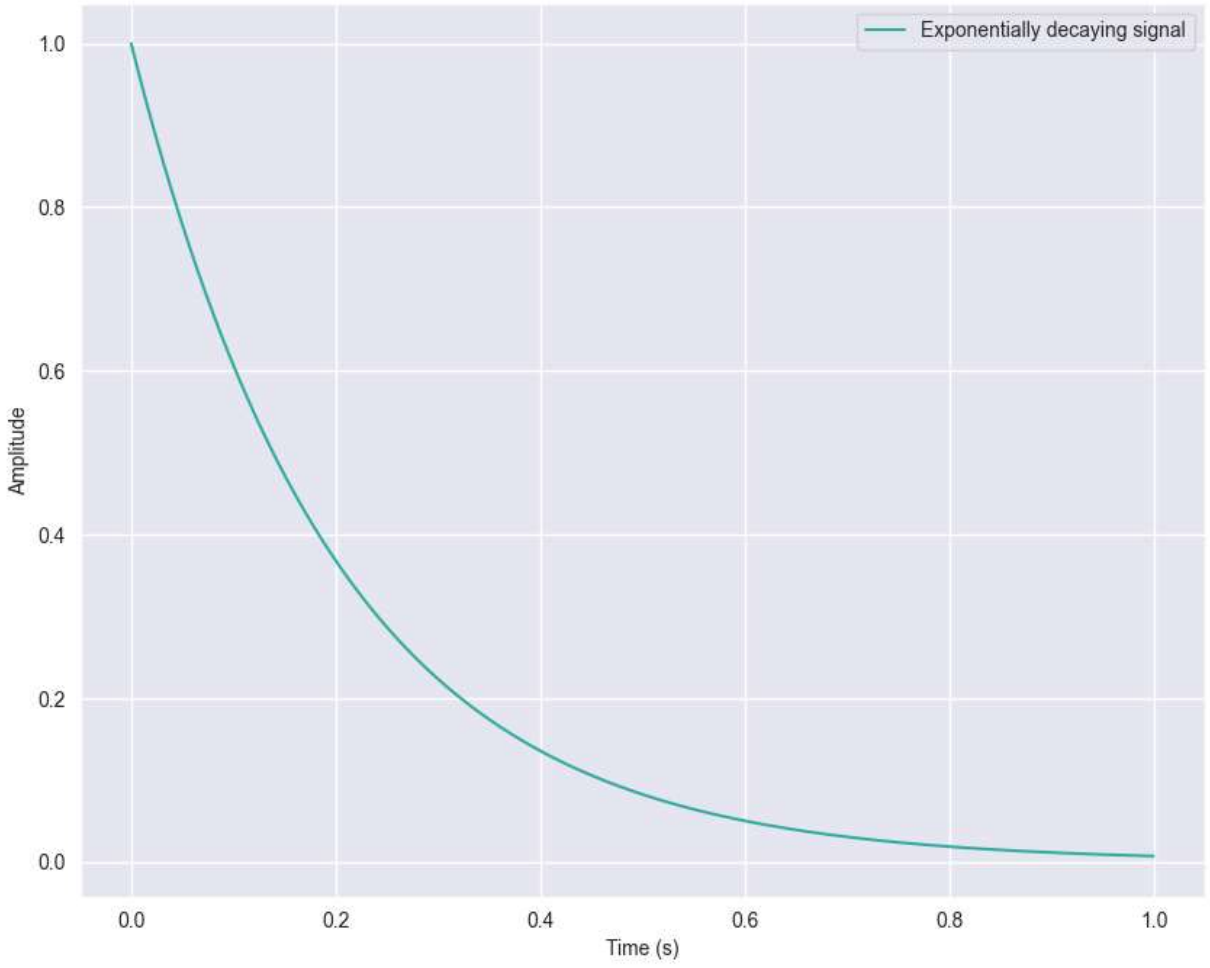
Where:

- (A) is the initial amplitude,
- (alpha) is the decay rate, which controls how fast the signal decreases,
- (t) is time.

As time progresses, the signal's value reduces exponentially, approaching zero. These signals are commonly encountered in natural processes such as electrical circuits, radioactive decay, and damping systems. In the frequency domain, exponentially decaying signals have a broad spectrum, with more high-frequency components as the decay rate increases.

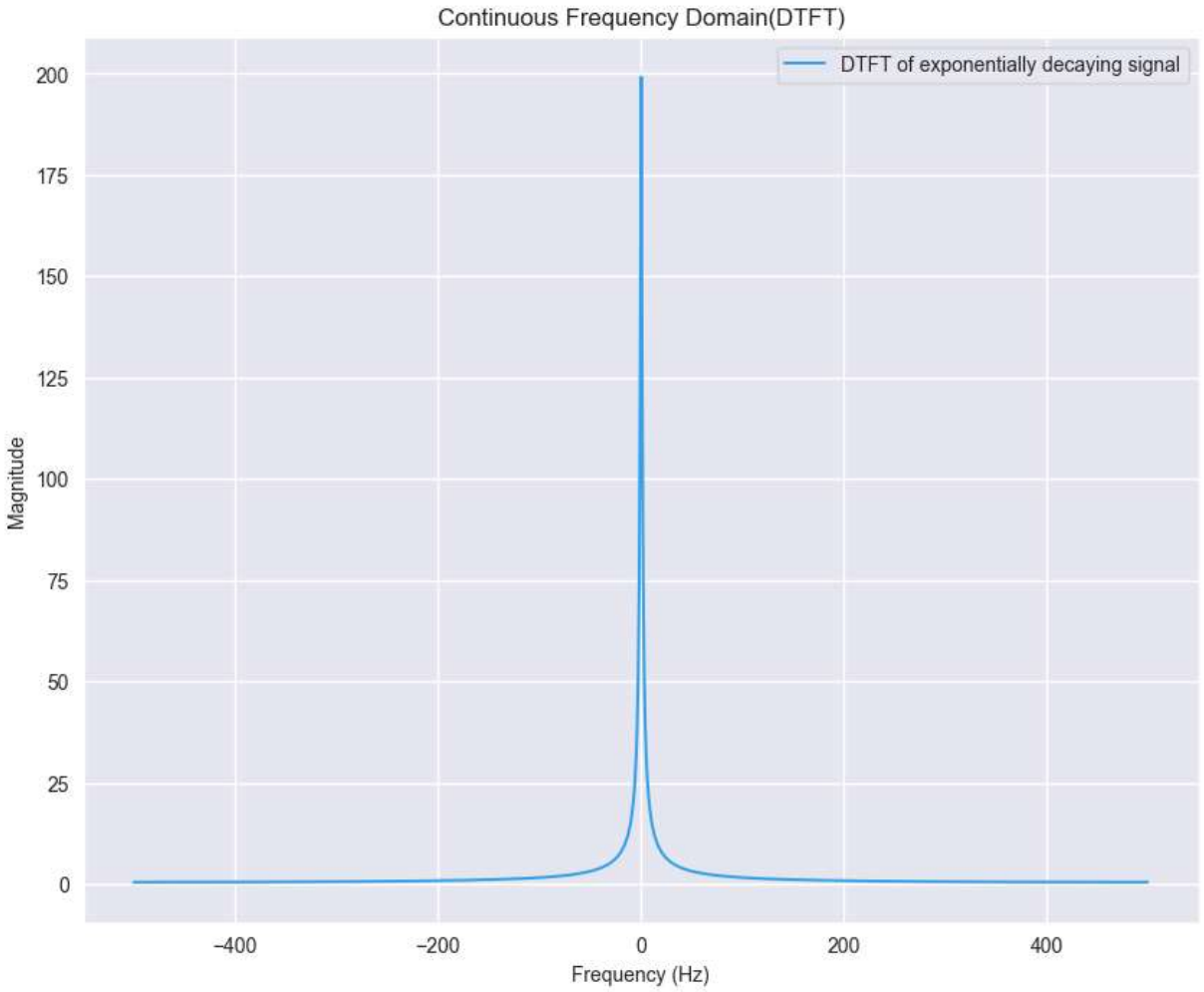
In [21]:
```
fs = 1000
t = np.linspace(0,1, fs, endpoint=False)
decay_rate = 5
exp_signal = np.exp(-decay_rate * t)
```

In [26]:
```
plt.figure(figsize=(10,8))
sns.lineplot(x = t, y = exp_signal, label = "Exponentially decaying signal", color
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.legend()
plt.show()
```

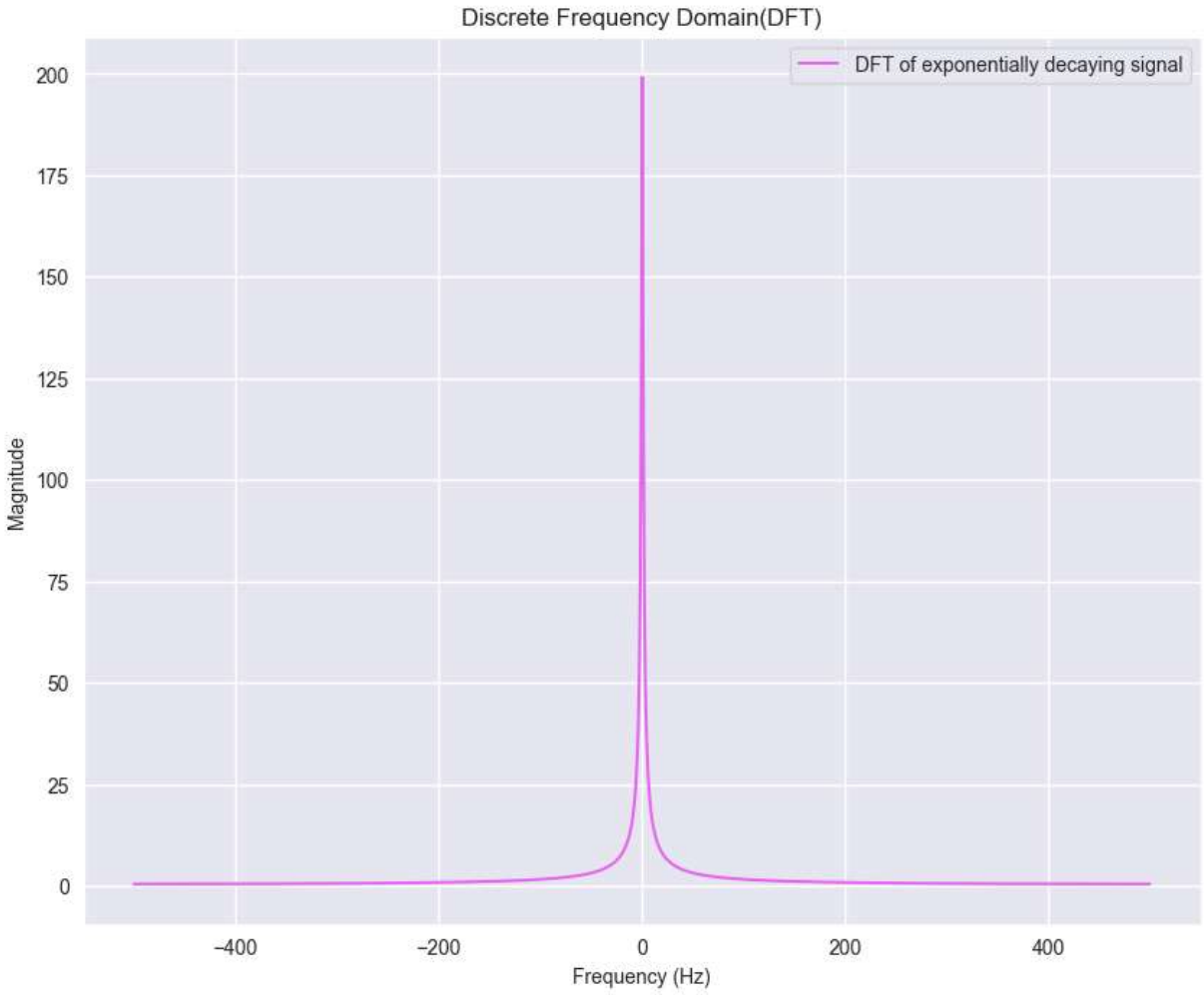


In [29]:
```
# DTFT

N = len(exp_signal)
```

```
f = np.fft.fftfreq(N, d=1/fs)
X = np.fft.fft(exp_signal)

plt.figure(figsize=(10,8))
sns.lineplot(x = f, y = np.abs(X), label = "DTFT of exponentially decaying signal",
plt.title('Continuous Frequency Domain(DTFT)')
plt.xlabel("Frequency (Hz)")
plt.ylabel("Magnitude")
plt.legend()
plt.show()
```



In [30]:
```
# DFT

dft = np.fft.fft(exp_signal)


plt.figure(figsize=(10,8))
sns.lineplot(x = f, y = np.abs(dft), label = "DFT of exponentially decaying signal"
plt.title('Discrete Frequency Domain(DFT)')
plt.xlabel("Frequency (Hz)")
plt.ylabel("Magnitude")
plt.legend()
plt.show()
```



Relationship Between Time-Domain and Frequency-Domain Representations

The exponentially decaying signal in the time domain represents a signal whose amplitude decreases over time. In the frequency domain:

- The DTFT and DFT show a broad range of frequency components. This is due to the decaying nature of the signal, which contains both low and high-frequency components.
- The DTFT has smoother, continuous spectral components, whereas the DFT provides discrete frequency points. The decay rate in the time domain affects the spread in the frequency domain.
- A faster decay corresponds to a broader frequency spectrum, as more frequency components are needed to capture the sharp decay.

(4) Question 4

(a) Generate an exponentially decaying signal.

(b) Plot the time-domain waveform.

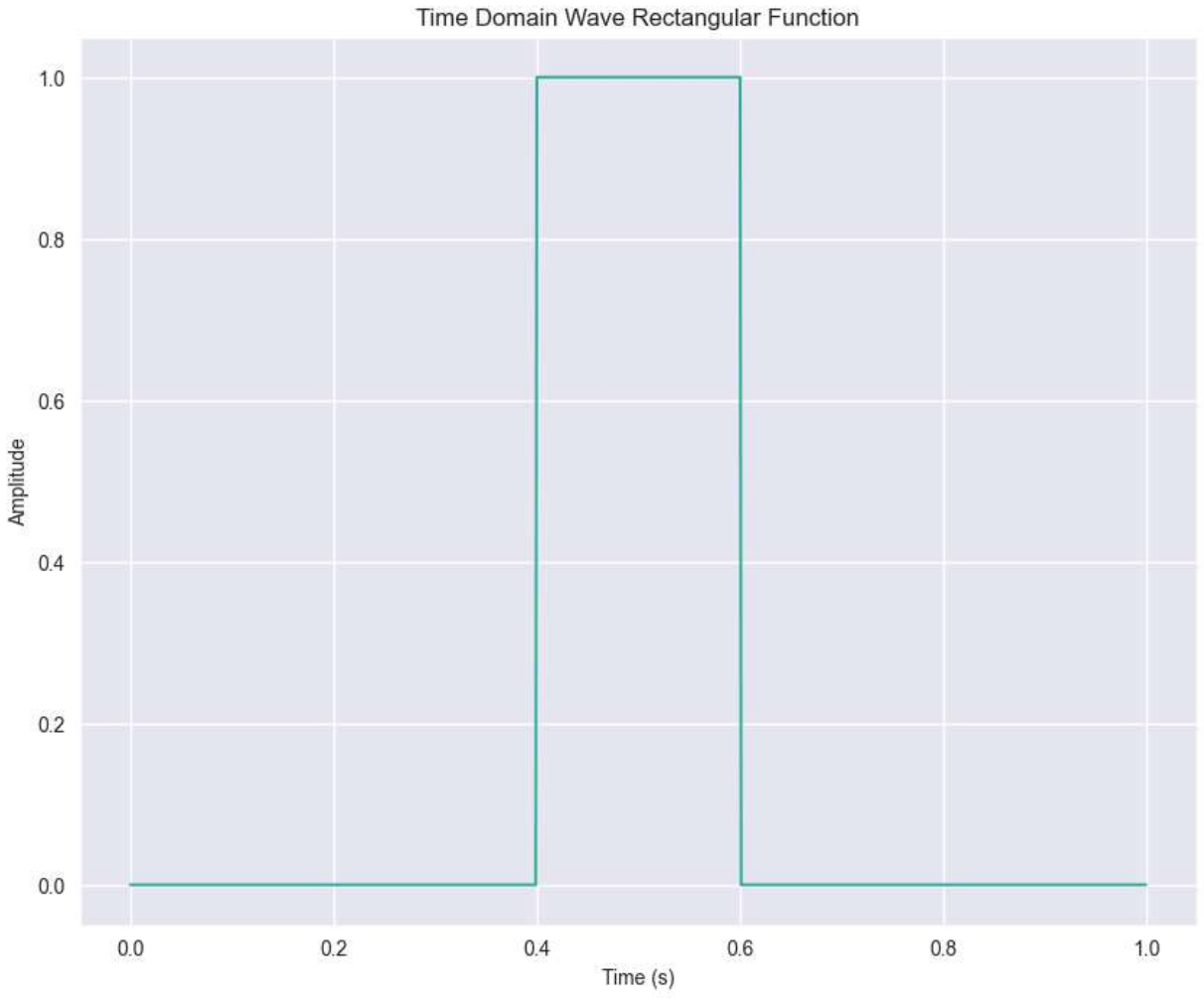
(c) Compute the Discrete-Time Fourier Transform (DTFT) and plot the con- tinuous frequency spectrum.

(d) Compute the Discrete Fourier Transform (DFT) and plot the discrete frequency spectrum.

(e) Analyze the relationship between the time-domain waveform and the frequency-domain representation.

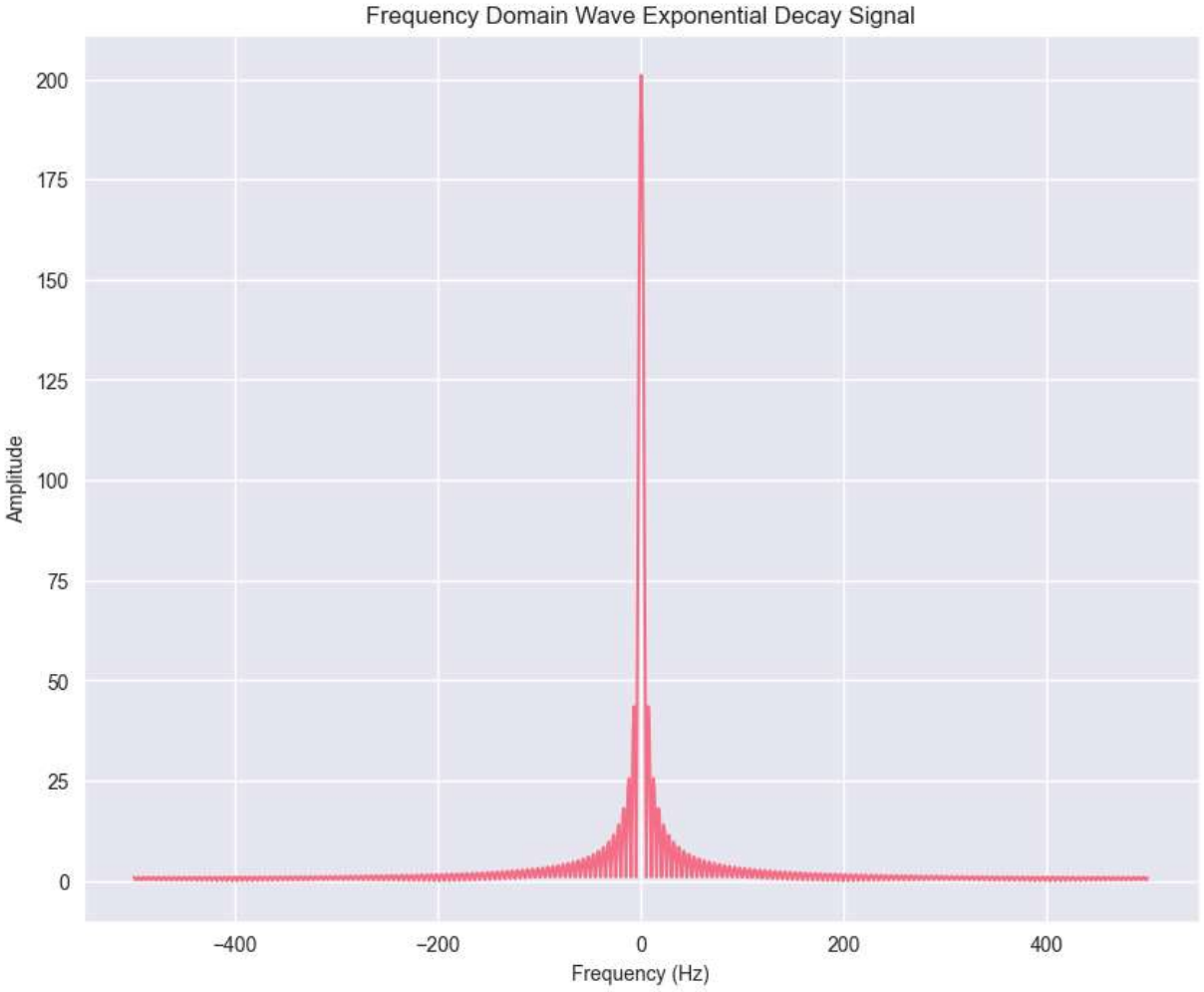
In [35]:
```
fs = 1000  # Sampling frequency (Hz)
t = np.linspace(0, 1, fs, endpoint=False)  # Time vector of 1 second
rect_width = 0.2  # Width of the rectangular pulse
rect_function = np.where((t >= (0.5 - rect_width/2)) & (t <= (0.5 + rect_width/2)),
```

In [39]:
```
plt.figure(figsize=(10, 8))
sns.lineplot(x = t, y = rect_function, color = colors[3])
plt.title("Time Domain Wave Rectangular Function")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.show()
```

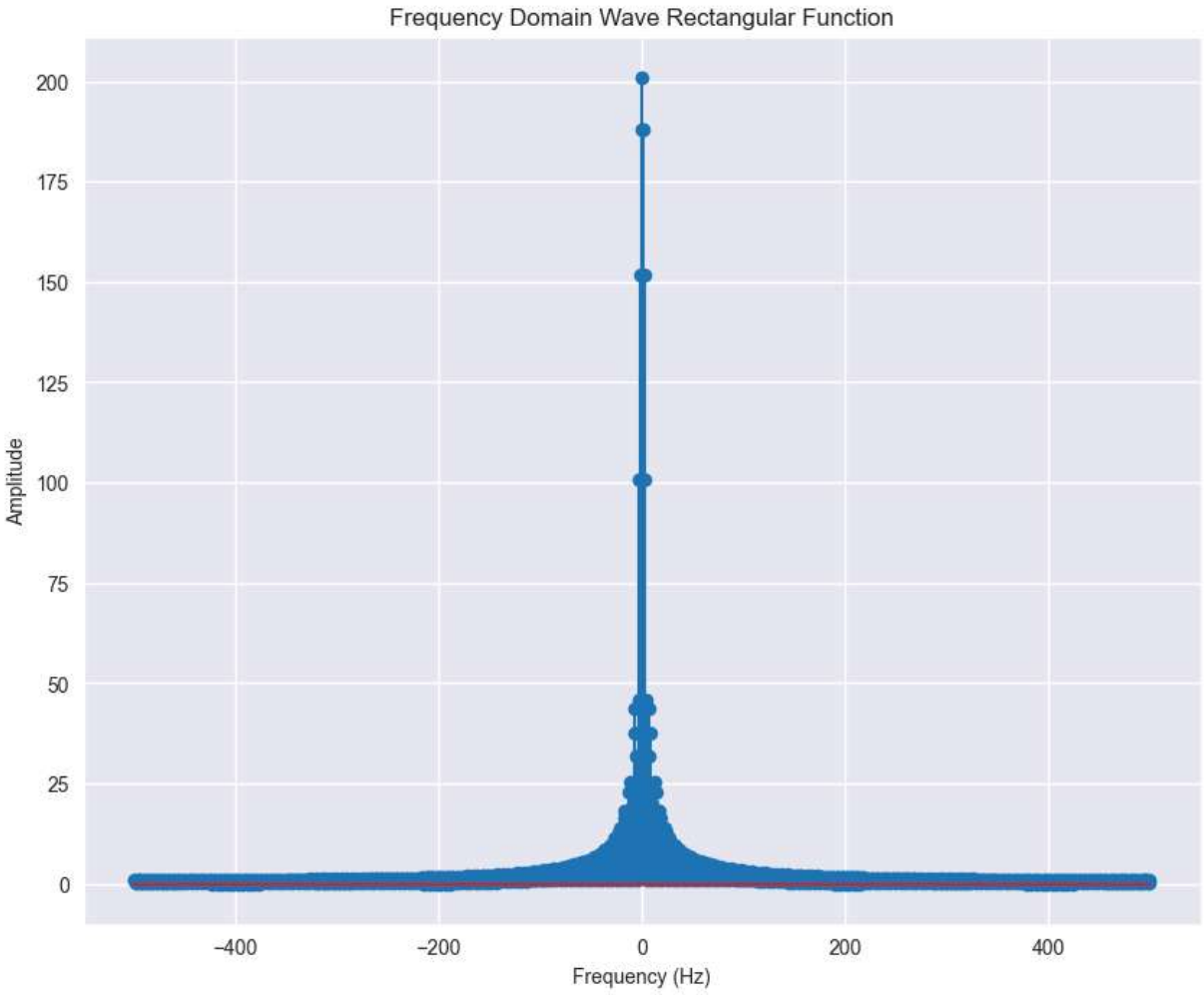


In [40]:
```
# DTFT

N = len(rect_function)
X = np.fft.fft(rect_function)
f = np.fft.fftfreq(N, d = 1/fs)
```

In [43]:
```
plt.figure(figsize=(10, 8))
sns.lineplot(x = f, y = np.abs(X), color = colors[0])
plt.title("Frequency Domain Wave Exponential Decay Signal")
plt.xlabel("Frequency (Hz)")
plt.ylabel("Amplitude")
plt.show()
```



In [47]:
```
dft = np.fft.fft(rect_function)

plt.figure(figsize=(10, 8))
plt.stem(f, np.abs(dft))
plt.title("Frequency Domain Wave Rectangular Function")
plt.xlabel("Frequency (Hz)")
plt.ylabel("Amplitude")
plt.show()
```



Inference on Frequency Domain Analysis of Signals Using DTFT and DFT

Frequency domain analysis is a powerful tool for understanding the characteristics of various signals, including rectangular functions, exponentially decaying signals, sinusoidal signals, and composite signals. The Discrete-Time Fourier Transform (DTFT) and the Discrete Fourier Transform (DFT) provide insights into how signals behave across different frequency components.

1. Rectangular Function

- **DTFT Analysis**: The frequency spectrum of a rectangular function exhibits a sinc-like shape, characterized by a main lobe and diminishing side lobes. This indicates that a finite-duration rectangular pulse contains a broad range of frequency components, with the width of the main lobe inversely related to the pulse duration.
- **DFT Analysis**: The DFT samples the sinc function at discrete frequencies, reinforcing the relationship between time-domain duration and frequency spread. The DFT representation provides practical insight into the frequency components at specific intervals, useful for digital signal processing applications.

2. Exponentially Decaying Signal

- **DTFT Analysis**: The frequency spectrum of an exponentially decaying signal is typically broader, indicating that the signal contains both low and high-frequency components. The decay rate directly affects the spectral spread, with faster decays resulting in wider frequency distributions.
- **DFT Analysis**: The DFT of an exponentially decaying signal confirms this by presenting a frequency spectrum that captures the rapid decay, highlighting the presence of higher frequencies. The DFT's discrete representation allows for easier implementation in digital systems.

3. Sinusoidal Signal

- **DTFT Analysis**: A pure sinusoidal signal results in a frequency spectrum that is sharply peaked at the sinusoid's frequency. The DTFT emphasizes the signal's periodic nature and its concentration of energy at specific frequencies.
- **DFT Analysis**: The DFT captures this characteristic as well, showing distinct spikes at the frequency of the sinusoidal signal. This representation is crucial for applications such as frequency analysis and filtering.

4. Composite Signal

- **DTFT Analysis**: The DTFT of a composite signal, which combines multiple sinusoidal components, reveals multiple peaks in the frequency spectrum corresponding to the frequencies of the individual sinusoids. The superposition principle allows for the examination of how different frequency components interact.
- **DFT Analysis**: The DFT displays these peaks at the same discrete frequencies, making it an effective tool for analyzing the frequency content of complex signals. It provides a practical representation that can be directly utilized in digital signal processing applications, such as modulation and demodulation.

Conclusion

The analysis of signals in the frequency domain using DTFT and DFT reveals critical insights into their spectral characteristics. Each type of signal exhibits unique frequency behaviors that can be understood through these transforms. The DTFT provides a continuous spectrum, ideal for theoretical insights, while the DFT offers a discrete spectrum, making it suitable for practical digital implementations. Understanding these relationships is essential for effective signal processing, allowing for the design of filters, communication systems, and various applications in engineering and science.