Objective:
We’ll convolve a sinusoidal signal:
$x[n]$=sin $(2\pi fn)$
with the filter
$h[n]$=[1,−1]

This filter gives the discrete difference, so:
Where the sine is increasing, the output will be positive
At peaks/troughs, the output will be near zero
What You'll See:
Input: smooth blue sinusoid
Output: red plot showing positive spikes when sine is rising, and negative dips when it's falling
Around the sine wave’s peaks and troughs, the output approaches zero (since the slope is near flat)

⍰ Physical Interpretation:
This is like measuring:

Slope

Speed of change

Also called discrete derivative

Used in:

FM demodulation

Instantaneous frequency estimation

Detecting fast vs. slow variation in IQ signals.
import numpy as np
import matplotlib.pyplot as plt

Time vector

n = np.arange(0, 50)

```
f = 0.05 # normalized frequency (cycles/sample)
x = np.sin(2 * np.pi * f * n)
```

Difference filter

```
h = np.array([1, -1])
y = np.convolve(x, h, mode='full')
```

Time axis for y[n]

```
n_y = np.arange(len(y))
```

Plotting

```
plt.figure(figsize=(10, 4))
plt.stem(n, x, basefmt=" ", linefmt='b-', markerfmt='bo', label='Input: sin(2πfn)')
plt.stem(n_y, y, basefmt=" ", linefmt='r-', markerfmt='ro', label='Output: Difference')
plt.title("Effect of Convolution with [1, -1] on a Sinusoid")
plt.xlabel("Sample index n")
plt.ylabel("Amplitude")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

Optional: print first few values

```
print("First 10 output values y[n]:", np.round(y[:10], 3))
```