🔍 The Moving Average Filter — Explained from First Principles

🎯 Objective:
To reduce **noise** in a signal using a **simple low-pass filter** — specifically, a **moving average filter**.
A **moving average filter** works by replacing each sample in a signal with the **average of its neighbors**.
Think of it like:

> "Instead of trusting one value, I'll look at a few values around it and smooth things out."
> This is great for:

- Smoothing out short-term noise
- Keeping the overall shape of the signal intact
- Acting as a **low-pass filter** (because it removes rapid changes = high frequency)
  🔍 What's Under the Hood?
  Let's say we have this tiny signal:
  Signal: [2, 4, 6, 8, 10]
  If we apply a 3-point moving average filter, we use this kernel:
  Kernel: [1/3, 1/3, 1/3]
  Then each new value is:
  New[1] = (2 + 4 + 6)/3 = 4
  New[2] = (4 + 6 + 8)/3 = 6
  New[3] = (6 + 8 + 10)/3 = 8
  So the output is smoothed:
  Filtered: [4, 6, 8]
  🔄 This Operation Is Called **Convolution**
  We "slide" the kernel over the signal and compute dot products at each step. This is the core of **convolution filtering**.
  In Python, we use:
  np.convolve(signal, kernel, mode='same'):
  * `signal`: your noisy signal
  * `kernel`: the filter (moving average here)
  * `mode='same'`: keeps the output size equal to the input size
  Why 'valid' mode?

*'valid': Keeps only the parts where the kernel fully overlaps (result is shorter).
*'same': Pads to maintain original length (useful for real-time filtering).
*'full': Even includes the parts with partial overlap (longer result).