
📍 Target Audience

- Learners entering DSP from practical fields (e.g., SDR, counter-drone systems)
- Those who want strong *foundations* before diving into Python or hardware

---

⏱ Duration

90 minutes (Can be split into 2 × 45-min sessions)

⟐ Session Structure

Session	Topic	Focus
Part 1	Foundations	What FIR filters are and how they work
Part 2	Interpretation	How to connect FIR to convolution, frequency, and design

---

⟐ PART 1 — FIR Filters: What, Why, and How

✓ 1. What is an FIR Filter?

FIR = Finite Impulse Response

> A filter where the output depends only on a finite number of past input samples.

🗏 FIR Equation:

$$y[n] = \sum_{k=0}^{N-1} h[k] \cdot x[n-k]$$

Where:

- $h[k]$: filter coefficients (impulse response)
- $x[n]$: input signal
- $y[n]$: output signal
- N: number of taps (filter length)

🔍 Why “Impulse Response”?

- Apply a delta input (1 followed by 0s)
- Output = the **coefficients themselves** → hence the name

✅ 2. How Does It Work? (Sliding Dot Product)

- At each time n, you **slide** $h[k]$ over input $x[n]$
- Take a **dot product** to compute one output value
- This is **convolution**

💬 Lyons Analogy:

> Like a moving weighted average with memory — each output is a blend of recent inputs

---

✅ 3. Key Properties of FIR Filters

Feature	Description
Linear Phase	Symmetric $h[n]$ → no phase distortion (important in comms/

Feature	Description
	audio)
Always Stable	No feedback = no runaway output
Easy to Design	Especially via windowing
Good for Multirate Systems	Used in up/down-sampling pipelines

---

✅ 4. Types of FIR Filters by Coefficient Shape

Filter Type	Coefficients Shape
Low-pass	Smooth sinc-like
High-pass	Alternating signs
Band-pass	Band-limited sinc
Notch	Impulse + negative bump
Moving Average	All ones

🔷 PART 2 — FIR Filtering via Convolution & Frequency Domain

✅ 5. Convolution = Core Operation

$$y[n] = x[n] * h[n]$$

- $*$: convolution operator
- Intuitively: **blending input with filter shape**
- Mathematically: **sum of weighted, time-shifted inputs**

✅ 6. Convolution ↔ Frequency Multiplication

Lyons emphasizes:

> "Filtering in time = shaping the spectrum in frequency."

Time Domain	Frequency Domain
Convolution	Multiplication
FIR filter	Windowed sinc = frequency gate
Apply $h[n]$	Suppress/pass spectral components

🌐 Design Insight:

- Want low-pass? Use sinc (ideal LPF) → window it → truncate to finite taps

---

✅ 7. Practical Design Flow (as per Lyons)

Step-by-step FIR Low-pass Design:

1. Choose sampling rate f_s
2. Pick cutoff frequency f_c
3. Build ideal sinc function:

$$h[n] = \text{sinc}\left(\frac{2f_c}{f_s}(n - \frac{N-1}{2})\right)$$

4. Apply window (e.g., Hamming)
5. Normalize (so gain = 1 at DC)

✅ 8. What to Remember Going Forward

Concept	Insight
FIR = convolution	Dot product of input and impulse response
Shape of $h[n]$	Defines the spectral behavior
Number of taps	More taps → sharper cutoff but more delay
Symmetry in $h[n]$	Ensures linear phase
Windowing	Balances leakage vs sharpness
Design with `firwin()`	Automates steps above using Python (e.g., SciPy)

🧪 Suggested Hands-on Activities (with Python or Paper)

1. **Visualize convolution** using small signals and filter coefficients
2. Plot time-domain response of FIR low-pass filter
3. Use `freqz()` to inspect frequency response
4. Compare filters with 21, 51, 101 taps
5. Try filtering real signal (e.g., 500 Hz + 2000 Hz sine mix)

📚 Suggested Reading (from Lyons)

Section	What It Teaches
Ch. 5: FIR Filters	Core filter structure, convolution mechanics
Ch. 6: Windows	Why and how to shape filters

Section	What It Teaches
Ch. 9: Frequency Domain	How to interpret filter behavior in freq domain
Appendix: Complex filters	Advanced FIR cases like Hilbert or matched filters

---

✅ Output of This Workshop

By the end, you should be able to:

- Explain what FIR filters are and why they matter
- Design a basic FIR filter and apply it in Python
- Understand convolution both intuitively and mathem