🧠 Today's Focus: **Chapter D1 — Reading & Understanding IQ Files**

This is the **first step in your IQ Data Skill Track**, and it unlocks:

- The **structure** of IQ binary files
- The **Pythonic way** to read, interpret, and visualize them
- The **DSP relevance** of the I and Q components

📘 Chapter D1: How to Read and Handle IQ Files

🧭 Goal of the Chapter:

- Build intuition for **what IQ files contain**
- Learn **how to read them in Python**
- Understand what `complex64`, `.view()`, `.reshape()` do
- Become confident in interpreting raw IQ signals visually

🔧 What Is an IQ File?

An IQ file contains samples of a **complex signal** over time, typically saved from SDR hardware. Each sample represents:

$$I(t) + jQ(t)$$

Where:

- **I** = In-phase signal (cosine-like)
- **Q** = Quadrature signal (sine-like)
- Together, they represent a **vector rotating in the complex plane** — encoding both **amplitude** and **phase**, i.e., **direction + strength** of a wave.

📦 How IQ Is Stored in Binary Files:

Most SDR tools save raw IQ data like this:

```
I0, Q0, I1, Q1, I2, Q2, ...
```

Each I and Q is typically:

- a 16-bit or 32-bit float
- stored **interleaved** (I-Q-I-Q...)

If 32-bit float:

- Each I and Q = 4 bytes
- So 1 complex sample = 8 bytes

🧠 The Key Concept — `np.fromfile()` + `.view()`

Step-by-step Python logic:

```
# Step 1: Read raw binary file into 32-bit float array
raw = np.fromfile("sample.iq", dtype=np.float32)

# Step 2: Reinterpret as complex64 (2 float32s → 1 complex sample)
iq = raw.view(np.complex64)

# Now iq is a NumPy array like: [I0+jQ0, I1+jQ1, I2+jQ2, ...]
```

🧩 Why `.view()`?

Because binary files store data as flat sequences of numbers — `.view()` tells NumPy: “interpret every **pair of float32** as a single `complex64` value.”

🔍 Visual Guide: How It All Fits

```
Binary IQ File (float32):

[0.25, -0.70,  0.50, -0.45,  0.95, 0.10, ...]

  ↓

NumPy `fromfile()` → array of float32:
[0.25       , -0.70,
 0.50       , -0.45,
 0.95       ,  0.10, ...]

  ↓

`.view(np.complex64)` → array of complex numbers:
[0.25 - 0.70j,
  0.50 - 0.45j,
  0.95 + 0.10j, ...]
```

🎯 How This Helps in DSP:

Python Tool	Skill It Enables	DSP Relevance
`np.fromfile()`	Load raw SDR output	Start of any IQ processing chain
`.view(np.complex64)`	Recombine I and Q	Recover phasor signal
`.real` / `.imag`	Access I and Q separately	Plot, analyze components
`.abs()`	Compute magnitude	Envelope / Power
`.angle()`	Extract phase	Demodulation