📘 How-To: Data Types & Storage in IQ Signal Processing (with Signal Context)

Full Signal Flow section to include:

- Baseband signal generation
- Carrier signal representation (modulation concept)
- Sampling rate
- Small, clear code stages
- Print statements for step-by-step understanding
- Ideal for **practical stage-wise DSP-SDR practice**

🎯 Objective

Create a full understanding of IQ data handling — from baseband signal generation, to modulation, to interleaved IQ storage — using minimal, meaningful Python code, with annotated print statements for DSP/SDR learners.

🔁 Full Signal Flow: From DSP Array to IQ File to RF Chain

[Analog Signal → Sampling]
→ [Digital Array in Float64]
→ [Typed Conversion]
→ [I/Q Structuring]
→ [Storage in File]
→ [Reload for Transmission]
→ [DAC & Upconversion to RF]

🧩 Step-by-Step Code with Comments & Print Statements

import numpy as np

STEP 1: Define signal parameters

fs = 1000 # Sampling rate in Hz (how often we sample per second)

f_base = 50 # Baseband signal frequency in Hz

f_carrier = 200 # Carrier frequency in Hz (for modulation context)

t = np.arange(0, 1, 1/fs) # Time vector for 1 second

print("Time vector (t):", t[:10])

🔷 Step 2: Generate Baseband I and Q signals

In SDR, baseband IQ signals are derived from cos and sin

I = np.cos(2 * np.pi * f_base * t) # In-phase (cosine)

Q = np.sin(2 * np.pi * f_base * t) # Quadrature (sine)

print("\nBaseband I[:5]:", I[:5])

print("Baseband Q[:5]:", Q[:5])

🔷 Step 3: (Optional) Modulate onto a carrier (for RF simulation)

Simulating an RF carrier with the baseband signal

carrier = np.cos(2 * np.pi * f_carrier * t)

modulated_signal = I * carrier # AM-like modulation for illustration

print("\nCarrier[:5]:", carrier[:5])
print("Modulated signal I*carrier[:5]:", modulated_signal[:5])

🔷 Step 4: Convert to float32 (lightweight & SDR-standard)

I = I.astype(np.float32)
Q = Q.astype(np.float32)
print("\nI dtype:", I.dtype, "| Q dtype:", Q.dtype)

🔷 Step 5: Interleave I and Q for SDR storage

IQ = np.empty(I.size + Q.size, dtype=np.float32)
IQ[0::2] = I
IQ[1::2] = Q
print("\nInterleaved IQ[:10]:", IQ[:10]) # [I0, Q0, I1, Q1, ...]

🔷 Step 6: Save to binary file

IQ.tofile("iq_baseband_float32.bin")
print("\nSaved: iq_baseband_float32.bin (float32, interleaved I/Q)")

🧠 Optional Concepts for Learners:

- You can scale and convert to `int16` if file size or DAC format requires:
  IQ_int16 = (IQ * 32767).astype(np.int16)
  IQ_int16.tofile("iq_baseband_int16.bin")
  print("Also saved: iq_baseband_int16.bin (int16, scaled)")
  ✅ Stage-Wise Learning Summary

Stage	What Happens	Python Step
Signal Sampling	Create baseband cosine/sine	`I = cos(...)`, `Q = sin(...)`
Carrier Concept	Multiply with higher-freq wave	`modulated = I * carrier`
Type Conversion	Save space / SDR compatibility	`.astype(np.float32)`
I/Q Structuring	Interleaved storage format	`IQ[0::2]=I; IQ[1::2]=Q`
File Storage	Lightweight binary write	`.tofile("filename.bin")`