🔄 IQ Signal Processing Chain (Time → Frequency → Noise → PSD) for Phase -1 Exercises

1. Generate the clean IQ signal

- Define `fs, f_signal, duration` .
- Create time vector `t = np.linspace(0, duration, N, endpoint=False)` .
- Generate `I = cos(2πft)` and `Q = sin(2πft)` .
- Combine: `iq_signal = I + 1j*Q` .

---

2. Frequency setup (FFT bins)

- Compute FFT of the clean signal:
  `iq_fft = np.fft.fft(iq_signal)` .
- Frequency vector:
  `freqs = np.fft.fftfreq(N, 1/fs)` .
- Shift both:
  `iq_fft_shifted = np.fft.fftshift(iq_fft)`
  `freqs_shifted = np.fft.fftshift(freqs)` .

---

3. Magnitude & Power of clean signal

- Magnitude: `mag = np.abs(iq_signal)` (time-domain amplitude).
- Signal power (RMS^2):
  `signal_power = np.mean(np.abs(iq_signal)**2)` .
- Power in dB: `10 * np.log10(signal_power)` .

---

4. Add Gaussian noise

- Define SNR: `SNR_dB = ...`.
- Convert to linear: `snr_linear = 10**(SNR_dB/10)`.
- Noise power: `noise_power = signal_power / snr_linear`.
- Create noise array (complex Gaussian):
  `noise = np.sqrt(noise_power/2) * (np.random.randn(N) + 1j*np.random.randn(N))`
  .
- Add to signal: `iq_noisy = iq_signal + noise`.

---

5. Compute noisy signal stats

- Compute mean power of noisy signal:
  `noisy_power = np.mean(np.abs(iq_noisy)**2)`.
- Noise power check: `np.mean(np.abs(noise)**2)` (should match desired).
- Noise dB = `10*log10(noise_power)`.

---

6. Frequency-domain analysis of noisy signal

- FFT: `iq_noisy_fft = np.fft.fft(iq_noisy)`.
- Shift: `iq_noisy_fft_shifted = np.fft.fftshift(iq_noisy_fft)`.
- PSD: `psd = np.abs(iq_noisy_fft_shifted)**2 / N`.
- PSD in dB: `10*np.log10(psd)`.

---

7. Plotting

- Subplot 1: Time-domain clean I/Q.
- Subplot 2: Time-domain noisy I/Q.
- Subplot 3: PSD of noisy IQ vs `freqs_shifted`.
