🎲 NumPy Random API Mastery for DSP & Communication Systems

Overview

In signal processing and communication simulations, generating random symbols, bits, and noise is essential. This hands-on workshop focuses on the **most-used NumPy random functions** tailored to typical DSP tasks.

🔧 Part 1: Core NumPy Random APIs You'll Use Most

1. `np.random.choice` – Random Selection from a List

Use this to generate modulation symbols (e.g., BPSK, QPSK).

```
symbols = np.random.choice([-1, 1], size=200)
```

✅ **Use case**: BPSK/QPSK symbol generation
🔍 **Tip**: Pass a list of complex numbers for QPSK or higher-order modulations.

---

2. `np.random.randint` – Random Integers in a Range

Great for binary sequences or multi-level PAM/QAM.

```
bits = np.random.randint(0, 2, size=200)
```

✅ **Use case**: Bitstream or quantized level generation
🎲 **Range rule**: Upper bound is exclusive → `randint(0, 2)` gives 0 or 1.

---

3. `np.random.rand` – Uniform Distribution [0, 1)

Generates random floats uniformly distributed.

```
noise = 0.1 * np.random.rand(6400)
```

✅ **Use case**: Uniform noise, random scaling
🎯 **Scale** as needed to desired range (e.g., `[-0.5, 0.5]`).

---

4. `np.random.randn` – Gaussian Distribution (μ=0, σ=1)

Ideal for simulating **AWGN** channels.

```
awgn = 0.05 * np.random.randn(6400)
```

✅ **Use case**: Additive White Gaussian Noise
🔧 **Control noise power** via scaling (e.g., `0.1 * randn(...)`).

---

5. `.astype(float)` – Type Conversion

Ensure numerical precision for signal processing operations.

```
symbols = symbols.astype(float)
```

✅ **Use case**: Avoid type mismatch in DSP filters/multipliers

---

📌 Quick Reference Table

Task	NumPy Call
Random ±1 symbols	`np.random.choice([-1, 1], size=N)`
Random 0/1 bits	`np.random.randint(0, 2, size=N)`
Uniform noise	`np.random.rand(N)`

Task	NumPy Call
Gaussian noise (AWGN)	`np.random.randn(N)`
Type conversion	`arr.astype(float)`

🔷 Part 2: Guided Exercises – Learn by Doing

🎯 Activity 1: BPSK Symbol Generation & Visualization

```
import numpy as np
import matplotlib.pyplot as plt

symbols = np.random.choice([-1, 1], size=1000)
print("Proportion of +1:", np.mean(symbols == 1))
print("Proportion of -1:", np.mean(symbols == -1))

plt.stem(symbols[:50], use_line_collection=True)
plt.title("First 50 BPSK Symbols")
plt.xlabel("Index"); plt.ylabel("Amplitude")
plt.grid(True); plt.show()
```

🔁 **Challenge**: Modify to generate **QPSK** symbols

💡 *Hint:* Use `np.random.choice()` with a list of complex symbols like `[(1+1j)/√2, ...]` and plot a constellation diagram.

🔷 Activity 2: Binary Bitstream

```
bits = np.random.randint(0, 2, size=200)
print("First 20 bits:", bits[:20])
```

🔁 **Challenge**: Generate 4-PAM levels → Map `{0,1,2,3} → {-3V, -1V, +1V, +3V}`

🔷 Activity 3: Uniform Noise

```
uniform_noise = 0.5 * (2 * np.random.rand(1000) - 1)

plt.hist(uniform_noise, bins=30, edgecolor='black', density=True)
plt.title("Uniform Noise Histogram [-0.5, 0.5]")
plt.xlabel("Amplitude"); plt.ylabel("Density")
plt.grid(True); plt.show()
```

📈 Activity 4: Gaussian Noise (AWGN)

```
std = 0.1
noise = std * np.random.randn(1000)

plt.hist(noise, bins=30, edgecolor='black', density=True)
plt.title(f"Gaussian Noise Histogram (σ={std})")
plt.xlabel("Amplitude"); plt.ylabel("Density")
plt.grid(True); plt.show()
```

🔷 Activity 5: Type Conversion

```
int_syms = np.random.choice([-1, 1], size=10)
print(int_syms, int_syms.dtype)

float_syms = int_syms.astype(float)
print(float_syms, float_syms.dtype)
```

📡 Part 3: Simulating a BPSK Link Over AWGN

🔁 End-to-End Simulation (with BER)

```
num_bits = 10000
noise_std = 0.5

# 1. Random bits
bits = np.random.randint(0, 2, num_bits)

# 2. BPSK modulation (0 → -1, 1 → +1)
symbols = 2 * bits - 1  # or np.where(bits == 1, 1.0, -1.0)

# 3. Add AWGN
received = symbols + noise_std * np.random.randn(num_bits)

# 4. Demodulation
demodulated = np.where(received > 0, 1, 0)

# 5. BER calculation
errors = np.sum(bits != demodulated)
ber = errors / num_bits
print(f"BER: {ber:.4f} ({errors} errors out of {num_bits})")
```

📈 **Optional**: Plot received vs transmitted scatter for 200 symbols

🔁 **Challenge**: Simulate for multiple `noise_std` and plot BER vs noise

---

🔑 Bonus: Reproducibility with Seeds

```
# Legacy global seed
np.random.seed(42)
print("Global seed:", np.random.randint(0, 10, 5))

# Modern generator
rng = np.random.default_rng(42)
print("Modern RNG:", rng.integers(0, 10, 5))
```

✅ **Tip**: Always use `default_rng()` for clean, reusable, thread-safe code.

🎓 Workshop Summary: What You've Learned

API	Purpose
`np.random.choice`	Modulation symbols (BPSK, QPSK, etc.)
`np.random.randint`	Bitstreams, quantized data
`np.random.rand`	Uniform random floats
`np.random.randn`	Gaussian (AWGN) noise
`.astype(float)`	Type safety in DSP
`np.random.seed()` / `default_rng()`	Reproducibility

📣 Final Takeaway

> Once you master these few APIs, you'll never struggle to generate test signals, symbols, or noise in DSP projects again.

This is your **go-to toolkit** for:

- Simulations
- Modulation/demodulation
- BER testing
- AWGN channels
- Eye diagram and filtering experiments
