JAX NumPy Quick Reference

Core Concepts & API

`import jax.numpy as jnp`: The primary way to access the JAX NumPy API. Designed to feel familiar to NumPy users.

Immutability: JAX arrays (`jnp.ndarray`) are **immutable**. Operations do *not* modify arrays in-place; they return *new* arrays.

Copies vs. Views: Operations like `jnp.transpose()`, `jnp.reshape()`, slicing typically return **copies**, not views.

Performance & Execution

`jax.jit(func)` / `@jax.jit`: Just-In-Time (JIT) compilation using **XLA**

- **Tracing:** Runs `func` once with placeholder shapes/types to record operations.
- **Optimization/Compilation:** XLA optimizes and compiles the trace into fast machine code for CPU/GPU/TPU.
- **Execution:** Subsequent calls with compatible inputs use the fast compiled code.

Hardware Acceleration (CPU, GPU, TPU): JAX runs seamlessly on different hardware via XLA.

- No code changes needed (`jax.jit` handles optimization for the target device).
- Major performance benefit for large computations.

Array Manipulation (Immutable Style)

- **`array.at[index].set(value)`**: The **required** way to perform indexed updates. Returns a **new array** with the update.

Random Number Generation (Explicit State)

- **`from jax import random`**: JAX's random module.
- **`key = random.PRNGKey(seed)`**: Creates an initial random state (PRNG key). **Must** be explicitly managed.

- **`key, subkey = random.split(key)`**: **Crucial step!** Splits a key to generate a new key for future use and a subkey for the current random operation. Ensures reproducibility.
- **`random.normal(key, ...)`, etc.**: All JAX random functions require an explicit `key` (usually a `subkey` from `split`) as the *first* argument.

Function Transformations

- **`jax.vmap(func, in_axes=(...))`**: Auto-vectorization. Transforms `func` (written for single data points) to operate efficiently over batches or axes.
- **`jax.grad(func)`**: Automatic differentiation. Returns a *new function* that computes the gradient of `func` w.r.t. its first argument (or specified args). Foundational for ML.
- **`jax.experimental.shard_map(...)` / `shmap`** : (**Experimental API**) Explicit, manual control over distributing computation across multiple devices (SPMD).

Debugging Utilities

- **`jax.debug.visualize_array_sharding(array)`**: Shows how an array is distributed across devices when using features like `shard_map`.

More Information

- JAX AI Stack - https://jaxstack.ai
- JAX - https://jax.dev
- Flax - https://flax.readthedocs.io