NumPyro is a library for probabilistic inference built on JAX that has the same interface for model specification and inference as Pyro.

JAX is a high-level tracing library for program transformations of Python and NumPy functions. e.g. automatic differentiation (grad), JIT compilation (jit), vectorization (vmap), and parallelization (pmap). Inference subroutines in NumPyro use effect handlers to inspect and modify program behavior and freely compose with JAX transformations resulting in significant speedup via parallelization and JIT compilation.

Effect handlers allow to modify the behavior of the program, hence enable more advanced inference mechanism such as enumeration to marginalize out the discrete latent variable \( z \). In particular, effect handlers allow us to run the program in two modes: one in which discrete latent variables are sampled and one in which they are enumerated. The first mode can be used to inspect the model structure and the second mode is used to compute the joint probability of the model.

**Effect Handlers**

- Effect handlers provide a way to injecct or extract inference utilities by combining effect handlers like seed, condition and trace with JAX transformations like vmap and jit.

It is easy to write fast vectorized inference utilities by combining effect handlers with seed, condition and trace with JAX transformations like vmap and jit.

```
def logistic_regression(x, y=None):
    ndias = np.shape(x)[-1]
    a = numpyro.sample("a", Normal(0, 1).expand([ndias]))
    b = numpyro.sample("b", Normal(0, 1))
    return numpyro.sample("y", Bernoulli(logits=x @ a + b), obs=y)
```

```
def predict_fn(rng_key, param, *args):
    conditioned_model = condition(logistic_regression, param)
    return seed(conditioned_model, rng_key="arg")
```

```
def gmm(data, K):
    """K-Gaussian Mixture Density under a Plate Notation"""
    plate("N", len(data), dim=-1)
    plate("K", K, dim=-1)
    phi = sample("phi", Categorical(1/K))
    mu = sample("mu", Normal(np.orange(K), 1))
    with plate("K", K, dim=-1):
        z = sample("z", Categorical(phi))
    return sample("obs", Normal(mu[z], 1), obs=data)
```

---

**Conclusion**

- NumPyro is a library for doing probabilistic inference. It is batteries included with modules for distributions, bijective transforms, and effect handlers.
- NumPyro uses JAX transformations under the hood for hardware acceleration, automatic differentiation, and vectorization.
- NumPyro’s effect handlers are composable with JAX’s transformations. This compositability allows us to:
  - offer the same modeling language as Pyro with features such as automatic enumeration of discrete latent variables.
  - leverage JAX transformations to parallelize and JIT compile static inference subroutines for significant speed ups.