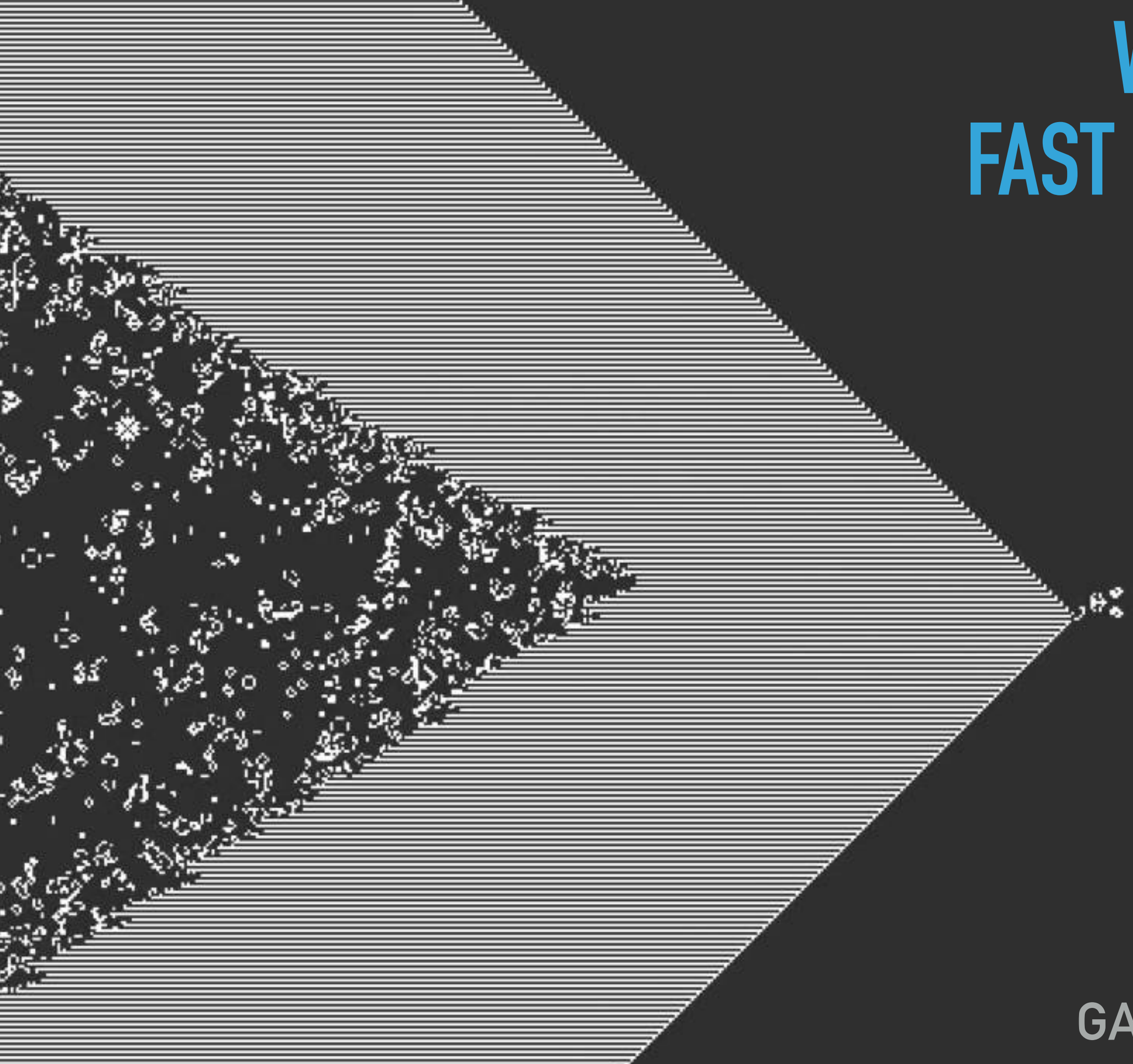
WRITING FAST PYTHON



CONWAY'S GAME OF LIFE

WHY IS PURE PYTHON SLOW?

- Interpreted language:
  - “Compiles” as it goes
- Dynamic typing (duck typing):
  - Doesn’t know the type of data until it checks
- Memory inefficient:
  - Internal memory structures need to be complex to provide type flexibility
- Inherently single threaded:
  - Global Interpreter Lock (GIL) prevents posix multithreading (not that important though)
- See e.g. http://jakevdp.github.io/blog/2014/05/09/why-python-is-slow/ for more detail

HOW TO MAKE PYTHON FASTER

- Use the best algorithm for the job… (Most important by far)
- Compile to byte code:
  - Python already partly does this for you, see *.pyc and *.pyo files
  - Can also use Cython, Pysco, ShedSkin, etc. but not always trivial
  - Can use JIT compiler
- Use the right tools:
  - arrays = Numpy/Numba/Numexpr, tables = Pandas
  - Need to do some ML, use Keras or Tensorflow
- Use libraries that interface to hardware accelerators:
  - See PyCuda, PyOpenCl, scikit.cuda etc.
- Embed a faster language in Python:
  - See Weave, PyCuda, Ctypes

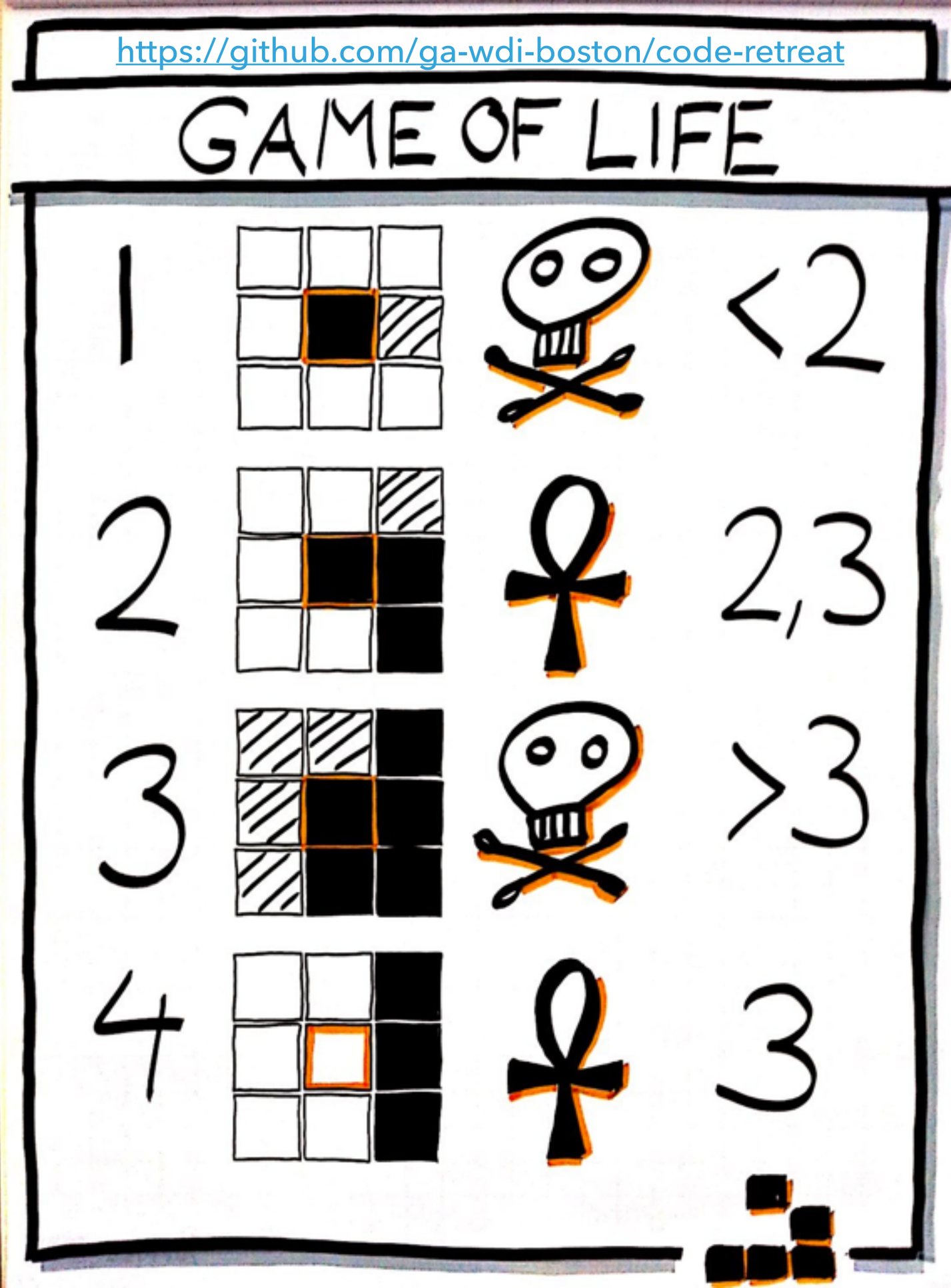
IN THIS TUTORIAL...

Weave

Numba

CONWAY'S GAME OF LIFE

- Cellular automaton developed by John Horton Conway 1970
- Four rules:
  - 1. If a cell has less than 2 neighbours it "dies"
  - 2. If a cell has 2 or 3 neighbours it "survives"
  - 3. If a cell has more than 3 neighbours it "dies"
  - 4. If a "dead" cell has 3 live neighbours, it comes back to life




jupyter