

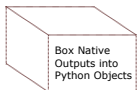
numbafied_library.py

```
1 @numba.njit(cache=True)
2 def foo(a, b):
3     ...
```

Python Interpreter

```
>>> foo(a, b)
```

Hand Back Control
To Python Interpreter



Run Optimized Machine Code
For Your Hardware: ARM, AMD,
Intel, PTX etc

numbafied_library.pyc

Python Byte Code

Hand Control To
Numba Runtime



Numba Front End

1) Analyze Python Bytecode To Create Numba's
Internal 'Intermediate Representation'

2) Perform Type Inference From Function's
Input Types

Numba Cache

Find/Store Function With
This Type Signature to Avoid
Re-Compilation

Numba Back End

Combine Type Information with Intermediate
Representation and Use LLVM to Create Optimized
Machine Code For Your Architecture

