numbafied_library.py

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

numbafied_library.pyc

Python Byte Code

Numba Front End

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

Python Interpreter

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

Hand Control To Numba Runtime

2) Perform Type Inference From Function's Input Types

Hand Back Control To Python Interpreter

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

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