Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

High-Performance Python with Cython

Class Duration

14 hours of live training delivered over 2 days.

Student Prerequisites

- All students should have significant experience with Python
- Working knowledge of C is required for the sections on wrapping C libraries and using C-level types
- The class has no review of the Python or C programming languages
- If you need to learn C programming, please consider the C Programming for Python Developers course

Target Audience

This course is for experienced Python developers who need to make Python code dramatically faster with Cython - whether by compiling and typing existing Python code, optimizing numerical and NumPy-heavy workloads, wrapping existing C libraries, or writing parallel code that releases the GIL.

Description

This two-day course teaches Cython 3 as a serious tool for high-performance Python. Students start by compiling pure Python through Cython for an easy speedup, then progressively add static typing, optimize numerical code with typed memory views and NumPy interop (covering NumPy 2.x), and learn to wrap existing C libraries to expose them as Python modules. Day two covers parallel programming with nogil and prange, advanced patterns including extension types and fused types, distribution to PyPI and Conda (including free-threaded wheels via cibuildwheel), and debugging and profiling Cython code. The course also addresses how Cython fits into the officially-supported free-threaded Python world (PEP 703 / PEP 779, supported in Python 3.14).

Learning Outcomes

- Understand what Cython is and when to choose it over alternatives (PyBind11, nanobind, CFFI, Numba)

- Compile Python code with Cython and read the generated annotation output to find optimization opportunities
- Apply Cython 3 static typing using both classic syntax (cdef/cpdef) and Pure Python mode with type hints
- Optimize loops and numerical code using compiler directives, typed memory views, and NumPy interop (NumPy 2.x compatible)
- Wrap existing C libraries with Cython, handling structs, unions, enums, typedefs, error codes, and callbacks
- Write parallel Cython code that safely releases the GIL using nogil and prange
- Use advanced Cython features including extension types, fused types, and special methods
- Distribute Cython extensions to PyPI and Conda using modern build tooling (setuptools, meson-python, cibuildwheel) including free-threaded wheels
- Debug and profile Cython code effectively
- Understand how Cython interacts with officially-supported free-threaded Python (PEP 779, Python 3.14)

Training Materials

All students receive comprehensive courseware covering all topics in the course. Courseware is distributed via GitHub in the form of documentation and extensive code samples. Students practice the topics covered through challenging hands-on lab exercises.

Software Requirements

Students will need a free, personal GitHub account to access the courseware. Students will need permission to install Python, a C compiler toolchain, and Visual Studio Code on their computers. Also, students will need permission to install Python Packages and Visual Studio Code extensions. If students are unable to configure a local environment, a cloud-based environment can be provided.

Training Topics

Cython Overview

- What is Cython?
- Cython vs C Extensions, PyBind11, nanobind, CFFI, and Numba

- When to Reach for Cython
- Installing Cython 3.x and the Toolchain

Compiling Python with Cython

- Compiling Pure Python Code with Cython
- Build Configuration with setuptools, meson-python, and pyproject.toml
- Inspecting Generated C Code with HTML Annotation
- Establishing Performance Baselines
- Quick Compilation with Jupyter %%cython

Cython Static Typing

- Cython's Function Forms: def, cdef, cpdef, and @cython.cfunc / @cython.ccall (Pure Python Mode)
- C-Level Variables and Type Declarations
- Pure Python Mode with Type Hints (Recommended for New Projects)
- Cython 3 Migration: language_level=3str, Stricter x: int Semantics, noexcept and Automatic Exception Propagation, Arithmetic Special-Method Binding Changes
- Cython 3.1+ Features: cython.pointer[T], cython.address, Improved Pure-Python-Mode Typing
- Cython 3.2 Highlights: PEP 701 F-Strings, PEP 750 T-Strings, range as a Type, Inferred Builtin Exception Types
- Measuring the Impact of Typing

Implementation and Declaration Files

- .pyx vs .pxd Files
- Sharing Declarations Across Modules
- Importing Cython Modules from Cython
- Organizing Larger Cython Codebases
- Type Stubs (.pyi) for Cython Extension Types

Optimizing Loops and Numerical Code

- Loop Optimization Patterns
- C-Level Math Operations
- Pointer Arithmetic in Cython
- Compiler Directives: boundscheck, wraparound, cdivision, initializedcheck
- Inlining and Function-Level Performance Tuning

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

Cython with NumPy

- Typed Memory Views (the Recommended Modern Approach)
- Passing NumPy Arrays to Cython Functions
- NumPy 2.x Compatibility: C API Changes and Py_ssize_t Patterns
- Legacy np.ndarray[dtype, ndim] Syntax (Context Only)
- Fast Numerical Code with NumPy and Cython
- Performance Comparison with NumPy and Numba

Wrapping C Libraries

- Declaring and Wrapping External C Code
- C Structs, Unions, Enums, and Typedefs
- Wrapping C Functions and Linking
- Error Handling Across the Python/C Boundary
- C Callbacks Calling Back into Python
- Memory Ownership and Resource Management
- When to Pick PyBind11 or nanobind Instead for C++ Wrapping

Parallel Programming with Cython

- Releasing the GIL with nogil
- Parallel Loops with prange
- OpenMP Integration
- What nogil Means Under Free-Threaded Python vs Standard GIL Builds
- Thread-Safety of cdef class Instances Under Free-Threading (cython.critical_section and cython.pymutex in Cython 3.3+)
- Free-Threading Officially Supported in Python 3.14 via PEP 779 (Cython's nogil/prange Support Remains Experimental in 3.1+)
- Module-Level State and Thread Safety

Advanced Cython Patterns

- Extension Types (cdef class)
- Special Methods and Operator Overloading
- Inheritance Patterns for Extension Types
- Implementing the Buffer Protocol on cdef class
- Fused Types (Cython Generics)

Debugging and Profiling Cython

- Building Cython in Debug Mode
- Debugging with GDB and the Cython Debugger

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

- Profiling Cython Code with cProfile and py-spy
- Testing Patterns for Cython Code
- Common Pitfalls and How to Avoid Them

Distributing Cython Extensions

- Building Wheels for Multiple Platforms
- cibuildwheel for Cross-Platform Builds
- Building Free-Threaded Wheels with cibuildwheel: cp314t (Default in cibuildwheel 3.x) and cp313t (Opt-In via CIBW_ENABLE)
- Packaging Cython Extensions for PyPI
- Packaging Cython Extensions for Conda

Choosing Cython vs Other Approaches

- When to choose Cython over Numba, PyBind11/nanobind, or Rust extensions (see *High-Performance Python with C and C++* and *High-Performance Python with Rust*)

