

# Python/C API Reference Manual

This manual documents the API used by C and C++ programmers who want to write extension modules or embed Python. It is a companion to [Extending and Embedding the Python Interpreter](#), which describes the general principles of extension writing but does not document the API functions in detail.

- [Introduction](#)
  - [Include Files](#)
  - [Objects, Types and Reference Counts](#)
  - [Exceptions](#)
  - [Embedding Python](#)
  - [Debugging Builds](#)
- [Stable Application Binary Interface](#)
- [The Very High Level Layer](#)
- [Reference Counting](#)
- [Exception Handling](#)
  - [Printing and clearing](#)
  - [Raising exceptions](#)
  - [Issuing warnings](#)
  - [Querying the error indicator](#)
  - [Signal Handling](#)
  - [Exception Classes](#)
  - [Exception Objects](#)
  - [Unicode Exception Objects](#)
  - [Recursion Control](#)
  - [Standard Exceptions](#)
  - [Standard Warning Categories](#)
- [Utilities](#)
  - [Operating System Utilities](#)
  - [System Functions](#)
  - [Process Control](#)
  - [Importing Modules](#)
  - [Data marshalling support](#)
  - [Parsing arguments and building values](#)
  - [String conversion and formatting](#)
  - [Reflection](#)
  - [Codec registry and support functions](#)
- [Abstract Objects Layer](#)
  - [Object Protocol](#)
  - [Number Protocol](#)
  - [Sequence Protocol](#)
  - [Mapping Protocol](#)
  - [Iterator Protocol](#)

- Buffer Protocol
  - Old Buffer Protocol
- Concrete Objects Layer
  - Fundamental Objects
  - Numeric Objects
  - Sequence Objects
  - Container Objects
  - Function Objects
  - Other Objects
- Initialization, Finalization, and Threads
  - Initializing and finalizing the interpreter
  - Process-wide parameters
  - Thread State and the Global Interpreter Lock
  - Sub-interpreter support
  - Asynchronous Notifications
  - Profiling and Tracing
  - Advanced Debugger Support
- Memory Management
  - Overview
  - Raw Memory Interface
  - Memory Interface
  - Object allocators
  - Customize Memory Allocators
  - The pymalloc allocator
  - Examples
- Object Implementation Support
  - Allocating Objects on the Heap
  - Common Object Structures
  - Type Objects
  - Number Object Structures
  - Mapping Object Structures
  - Sequence Object Structures
  - Buffer Object Structures
  - Async Object Structures
  - Supporting Cyclic Garbage Collection
- API and ABI Versioning