Extending and Embedding the Python Interpreter

This document describes how to write modules in C or C++ to extend the Python interpreter with new modules. Those modules can not only define new functions but also new object types and their methods. The document also describes how to embed the Python interpreter in another application, for use as an extension language. Finally, it shows how to compile and link extension modules so that they can be loaded dynamically (at run time) into the interpreter, if the underlying operating system supports this feature.

This document assumes basic knowledge about Python. For an informal introduction to the language, see :ref:`tutorial-index`. :ref:`reference-index` gives a more formal definition of the language. :ref:`library-index` documents the existing object types, functions and modules (both built-in and written in Python) that give the language its wide application range.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]index.rst, line 15); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]index.rst, line 15); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]index.rst, line 15); *backlink*

Unknown interpreted text role "ref".

For a detailed description of the whole Python/C API, see the separate :ref:`c-api-index`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]index.rst, line 21); *backlink*

Unknown interpreted text role "ref".

Recommended third party tools

This guide only covers the basic tools for creating extensions provided as part of this version of CPython. Third party tools like Cython, cffi, SWIG and Numba offer both simpler and more sophisticated approaches to creating C and C++ extensions for Python.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]index.rst, line 35)

Unknown directive type "seealso".

```
.. seealso::

   `Python Packaging User Guide: Binary Extensions <https://packaging.python.org/guides/packaging-binary-ext
      The Python Packaging User Guide not only covers several available
      tools that simplify the creation of binary extensions, but also
      discusses the various reasons why creating an extension module may be
      desirable in the first place.
```

Creating extensions without third party tools

This section of the guide covers creating C and C++ extensions without assistance from third party tools. It is intended primarily for creators of those tools, rather than being a recommended way to create your own C extensions.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]index.rst, line 52)

Unknown directive type "toctree".

```
.. toctree::
   :maxdepth: 2
   :numbered:

   extending.rst
   newtypes_tutorial.rst
   newtypes.rst
   building.rst
   windows.rst
```

Embedding the CPython runtime in a larger application

Sometimes, rather than creating an extension that runs inside the Python interpreter as the main application, it is desirable to instead embed the CPython runtime inside a larger application. This section covers some of the details involved in doing that successfully.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]index.rst, line 70)

Unknown directive type "toctree".

```
.. toctree::
   :maxdepth: 2
   :numbered:

   embedding.rst
```