C Extension Module for Python: Factorial Example

This project demonstrates how to create a C extension module for Python that calculates the factorial of a number. The module is implemented in C for performance and integrated with Python using the Python/C API. Below is the detailed flow of creating, compiling, and using the extension, with instructions tailored for a Windows system using Python 3.11.9 and MinGW GCC 15.2.0 (via MSYS2).

Flow of the C Extension Module

- Write the C Code (factorial.c)
- Create the Setup Script (setup.py)
- Compile the Extension
- Test the Extension (test_factorial.py)
- Execution and Output

1. Write the C Code (factorial.c)

- Purpose: Define the core functionality (factorial calculation) in C for performance and integration with Python.
- Key Components:
- - Include Python.h to provide Python/C API functions.
- - Define a factorial function that validates input and computes factorial.
- - Map the function to Python using PyMethodDef.
- - Define module with PyModuleDef.
- - Initialize module with PyInit_factorial.

2. Create the Setup Script (setup.py)

- Purpose: Configure the build process to compile the C code into a Python-compatible shared library.
- Uses setuptools to define an Extension object specifying factorial.c.
- Provides metadata (name, version, description).

3. Compile the Extension

- Command: python setup.py build_ext --inplace --compiler=mingw32
- Builds factorial.cp311-win_amd64.pyd using MinGW GCC.
- Ensure MSYS2 bin directory is in PATH.

4. Test the Extension (test_factorial.py)

- Purpose: Verify the module works by importing and calling the C function from Python.
- Handles inputs like 5, 0, -1 and shows errors for invalid inputs.

5. Execution and Output

- factorial(5) → 120
- factorial(0) → 1
- factorial(-1) → Raises ValueError

Setup Instructions for Windows

- Python 3.11.9 installed in virtual environment.
- MinGW GCC 15.2.0 installed via MSYS2.
- Install setuptools with pip.
- Save files (factorial.c, setup.py, test_factorial.py).
- Compile extension with setup.py.
- Run test script.

Troubleshooting

- 'command gcc not found': Add MSYS2 bin directory to PATH.
- ModuleNotFoundError: Ensure .pyd file is in same directory.
- Compilation errors: check error messages.

Relevance to AI/ML

- Performance boost for computational tasks.
- Used in libraries like NumPy, TensorFlow, PyTorch.
- Custom ML operations can be implemented in C.

Summary of Flow

- Write factorial in C with Python/C API.
- Setup compilation via setup.py.
- Compile with MinGW to generate .pyd.
- Test via Python script.
- Use C's efficiency within Python's flexibility.