在 Windows 系统上安装 Boost.Python 库可以按照以下步骤进行：

### 步骤 1：下载 Boost 库

* 访问 [Boost 官方下载页面](https://www.boost.org/users/download/)。
* 在该页面中找到适合你系统的 Boost 版本（通常建议选择最新的稳定版本），下载对应的 boost\_<版本号>.7z 或 boost\_<版本号>.zip 文件。例如，如果你选择的是 Boost 1.82.0 版本，文件可能是 boost\_1\_82\_0.7z。

### 步骤 2：解压文件

* 使用解压工具（如 7-Zip）将下载的压缩文件解压到一个指定的目录，例如 C:\boost\_1\_82\_0 。解压后的文件夹包含了 Boost 库的所有源代码和相关文件。

### 步骤 3：配置编译环境

* 确保你已经安装了合适的编译器，例如 Visual Studio 中的 MSVC 编译器。Boost 库支持多种编译器，但 MSVC 是 Windows 平台上较为常用的选择。
* 打开 “适用于 VS 的 x64 本机工具命令提示”（如果你需要编译 64 位版本的库）或 “适用于 VS 的 x86 本机工具命令提示”（如果你需要编译 32 位版本的库）。这些命令提示可以在开始菜单中找到，它们会自动配置好编译器所需的环境变量。

### 步骤 4：编译 Boost.Python

#### 生成编译工具

* 打开上述提到的命令提示窗口，使用 cd 命令切换到解压后的 Boost 文件夹，例如：

收起

plaintext

cd C:\boost\_1\_82\_0

* 运行以下命令来生成编译工具 b2（以前称为 bjam）：

收起

plaintext

bootstrap.bat

执行该命令后，会在当前目录下生成 b2.exe 文件。

编辑 project-config.jam

文件末尾加入

using python : 3.10 ;

#### 编译 Boost.Python

* 运行 b2 命令来编译 Boost.Python 库。以下是一个基本的编译命令示例，用于编译支持 Python 3.10 的 64 位版本的 Boost.Python 库：

收起

plaintext

b2 --with-python toolset=msvc-<编译器版本> address-model=64 python=3.10 link=shared threading=multi runtime-link=shared

toolset=msvc-<编译器版本>：指定使用的 MSVC 编译器版本。例如，如果你使用的是 Visual Studio 2022，编译器版本可能是 14.3 。

address-model=64：表示编译 64 位版本的库；如果需要 32 位版本，可以将其改为 address-model=32 。

python=3.10：指定要支持的 Python 版本，你需要根据自己实际安装的 Python 版本进行修改。

link=shared：表示编译为动态链接库；如果需要静态链接库，可以将其改为 link=static 。

threading=multi：表示支持多线程。

runtime-link=shared：表示运行时库使用共享模式。

编译过程可能需要一些时间，具体取决于你的计算机性能。编译完成后，生成的库文件会位于 stage\lib 文件夹中，头文件则位于 boost 文件夹中。

### 步骤 5：配置开发环境

#### ~~配置头文件路径~~

* ~~如果你使用的是 Visual Studio 开发 C++ 项目，打开项目属性页（右键点击项目 -> 属性）。~~
* ~~选择 “配置属性” -> “C/C++” -> “常规”，在 “附加包含目录” 中添加 Boost 头文件所在的目录，即 C:\boost\_1\_82\_0 。~~

#### ~~配置库文件路径~~

* ~~在项目属性页中，选择 “配置属性” -> “链接器” -> “常规”，在 “附加库目录” 中添加编译生成的库文件所在的目录，即 C:\boost\_1\_82\_0\stage\lib 。~~

#### ~~配置链接器输入~~

* ~~选择 “配置属性” -> “链接器” -> “输入”，在 “附加依赖项” 中添加所需的 Boost.Python 库文件，例如 boost\_python310-vc<编译器版本>-mt-x64-<版本号>.lib （具体文件名可能会根据编译选项和版本有所不同）。~~

添加环境变量BOOST\_PATH和PYTHON\_PATH分别为你的BOOST以及PYTHON主目录

图形用户界面, 文本, 应用程序

AI 生成的内容可能不正确。

### 步骤 6：测试安装

* 编写一个简单的 C++ 程序来测试 Boost.Python 是否安装成功。以下是一个示例代码：

收起

cpp

#include <iostream>#include <boost/python.hpp>

namespace py = boost::python;

int main() {

Py\_Initialize();

try {

py::object main\_module = py::import("\_\_main\_\_");

py::object main\_namespace = main\_module.attr("\_\_dict\_\_");

py::exec("print('Hello from Python using Boost.Python!')", main\_namespace);

}

catch (py::error\_already\_set const&) {

PyErr\_Print();

}

Py\_Finalize();

return 0;}

* 编译并运行该程序，如果能够正常输出 Python 代码的执行结果，则说明 Boost.Python 库安装成功。

C++ 17 标准

