Tektronix®

使用tm_devices 简化Python测试自动化

操作指南

目录

引言

许多行业的工程师都使用自动化来扩展其测试仪器的功能，而大多是选择免费的编程语言——Python 来完成。作为适用于自动化的主要编程语言，Python 具备许多显著优势：

- 多功能性
- 易于教学
- 代码可读性
- 广泛可用的知识库和模块

自动化包含两种主要使用情形：

- 模拟人类行为以自动化前面板并节省时间的例程，例如自动化合规性测试。每次需要测试新零件时，工程师不会呆坐在示波器前，添加相应的测量值并记下结果，而是开发一个脚本来完成所有工作并显示最终结果。
- 扩展仪器功能的用途，例如：测量记录、验证或质量保证。自动化操作支持工程师执行复杂的测试，且能够规避测试过程中的许多固有缺点。操作人员无需设置示波器和手动记录结果，且每次都可以以相同的方式进行测试。

本技术简介将涵盖使用 Python 进行编程所需的内容，包括编程接口的基础知识以及下载和运行方法的示例。

编程接口的定义

编程接口 (PI) 是两个计算系统之间的一个或多个边界，可以通过编程来执行特定行为。就本文而言，编程接口是运行每台泰克测试设备的计算机和最终用户编写的应用程序之间的桥梁。为了进一步缩小范围，编程接口在此处定义为一组可以远程发送到仪器的命令，仪器会处理这些命令并执行相应的任务。PI 堆栈 (图 1) 显示了从主机控制器到仪器的信息流。最终用户编写的应用程序代码用于定义目标仪器的行为。最终用户通常用业内流行的开发平台编写代码，如 Python、MATLAB、LabVIEW、C++ 或 C#。这些应用程序将使用可编程仪器标准命令 (SCPI) 格式——大多数测试和测量设备都支持的标准格式——发送数据。SCPI 命令通常通过虚拟仪器软件架构 (VISA) 层发送，VISA 层可通过为增强通信协议的鲁棒性 (例如错误检查) 促进数据传输。在某些情况下，应用程序可能会调用驱动程序，然后驱动程序会向 VISA 层发送一个或多个 SCPI 命令。

tm_devices 驱动程序包的内容

泰克的 tm_devices 是泰克自行开发的设备管理包，支持用户使用编程语言 Python 控制和自动化泰克和吉时利产品的测试过程。使用 Python 的软件包管理系统 pip 可快速安装 tm_devices。Python 软件包包含大量命令和功能，可帮助用户轻松地对泰克和吉时利产品进行自动化测试。其可在最流行的 Python IDE 中使用，且支持代码补全辅助。此外，Python 软件包还可以帮助任何水平软件技能的工程师简单轻松地进行编码和测试自动化。

环境设置

本节将指导您完成使用 tm_ devices 进行开发工作的先决条件和安装。我们特意在 Python (venvs) 中提供了支持虚拟环境的指令，因为我们相信这样可以帮助您更轻松地管理和维护项目，特别是在使用 Python 软件包之前，您只是简单地试用了一下的情况下。

安装与先决条件概述

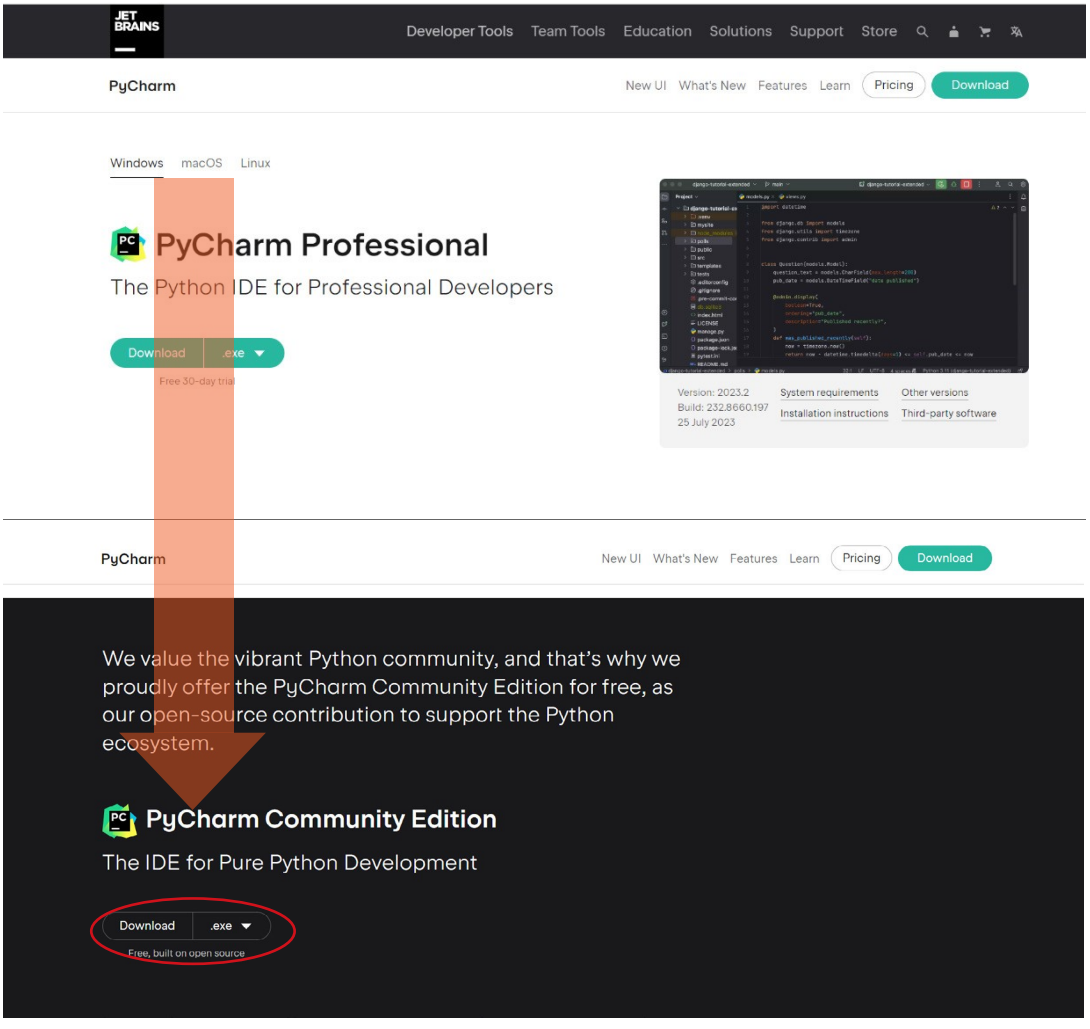
1. 安装 Python.
   a. Python >=3.8
2. PyCharm——PyCharm 安装、启动项目和 tm_devices 安装
3. VSCode——VSCode 安装、启动项目和 tm_devices 安装

PyCharm 社区版（免费）

PyCharm 是一款各行各业软件开发人员广泛使用的 Python IDE。PyCharm 有一个集成的单元测试器，支持用户按文件、类、方法的运行测试，或直接运行文件夹中的所有测试。与大多数现代 IDE 一样，PyCharm 拥有代码补全形式，和基本的文本编辑器相比，可以大大加快开发速度。

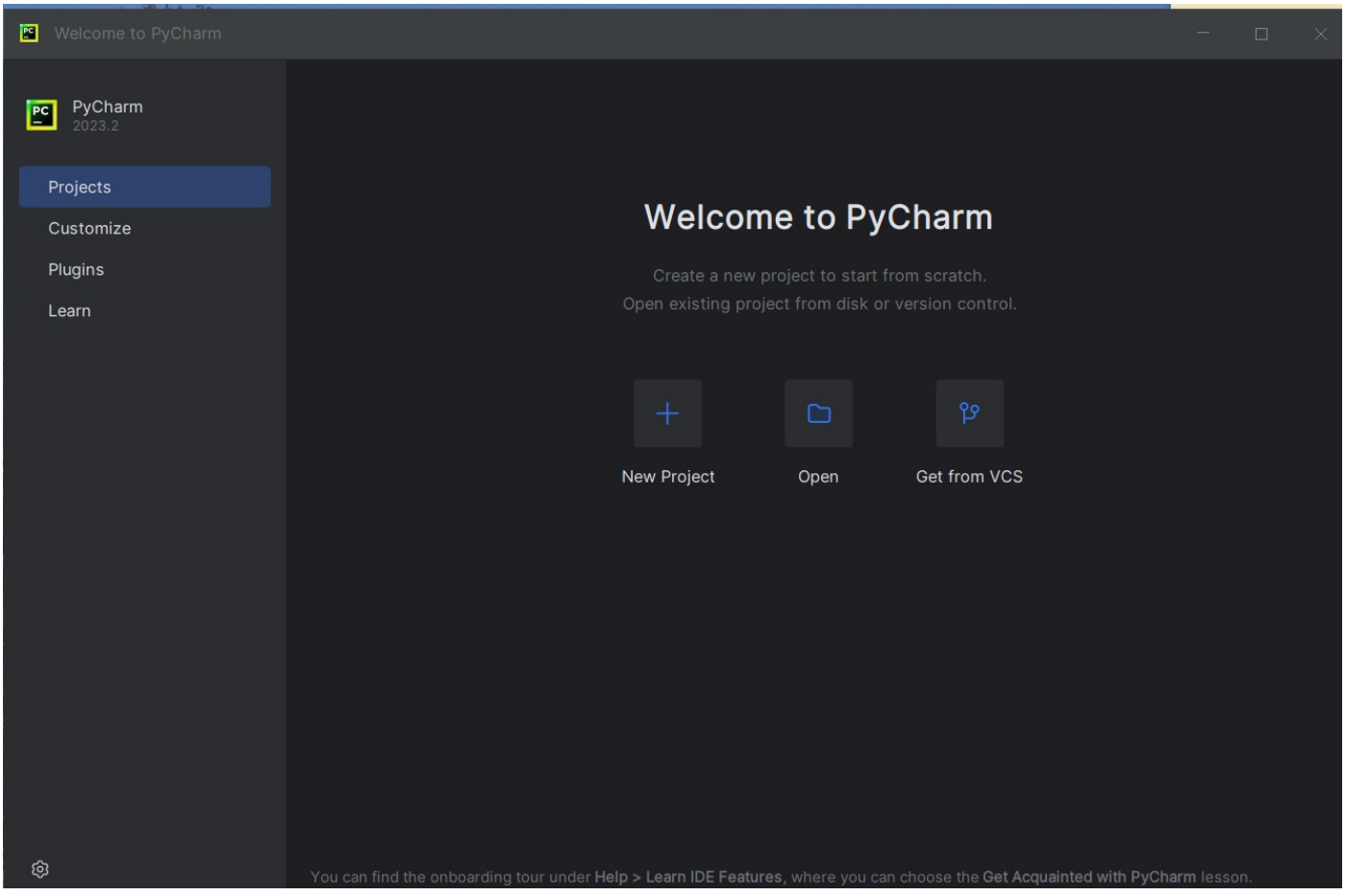
接下来，我们将为您介绍 PyCharm 社区版（免费）的安装过程，然后在 IDE 中安装 tm_devices 并设置一个虚拟环境进行开发。

1. 访问 https://www.jetbrains.com/pycharm/.
2. 跳过 PyCharm 专业版，下拉至 PyCharm 社区版，然后点击下载。



3. 请继续执行默认的安装步骤。无需进行任何特殊操作。

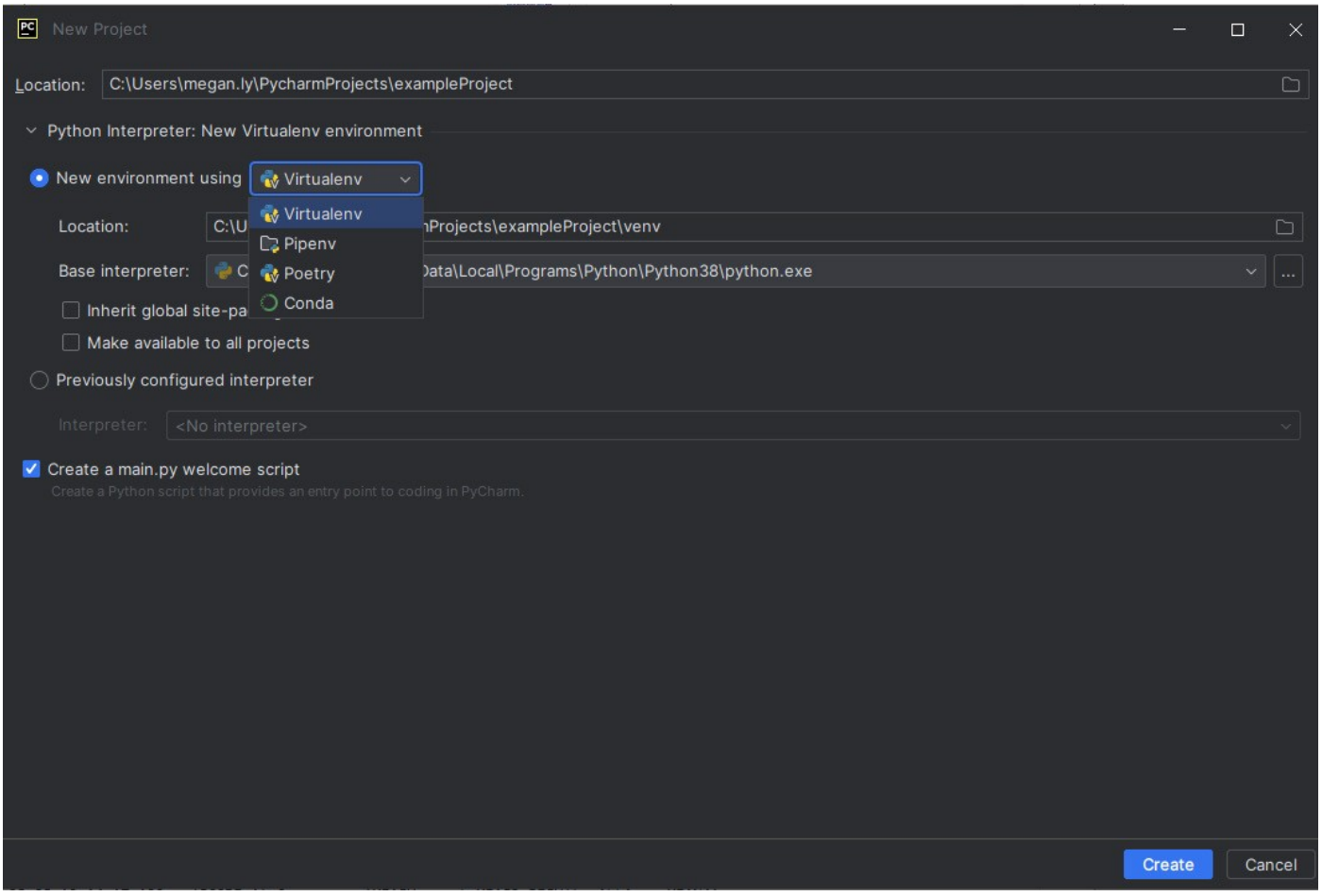
4. 欢迎使用 PyCharm！



在 PyCharm 中创建新项目并设置虚拟环境

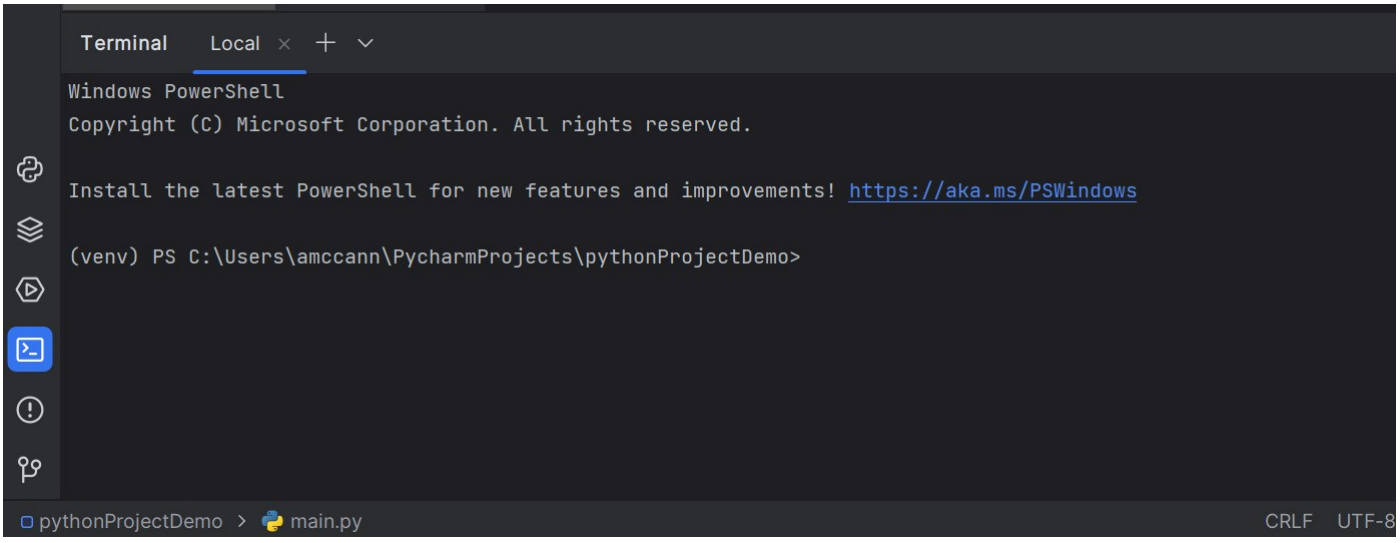
5. 点击“新建项目”。

6. 确认项目路径，确保选择 “Virtualenv”。



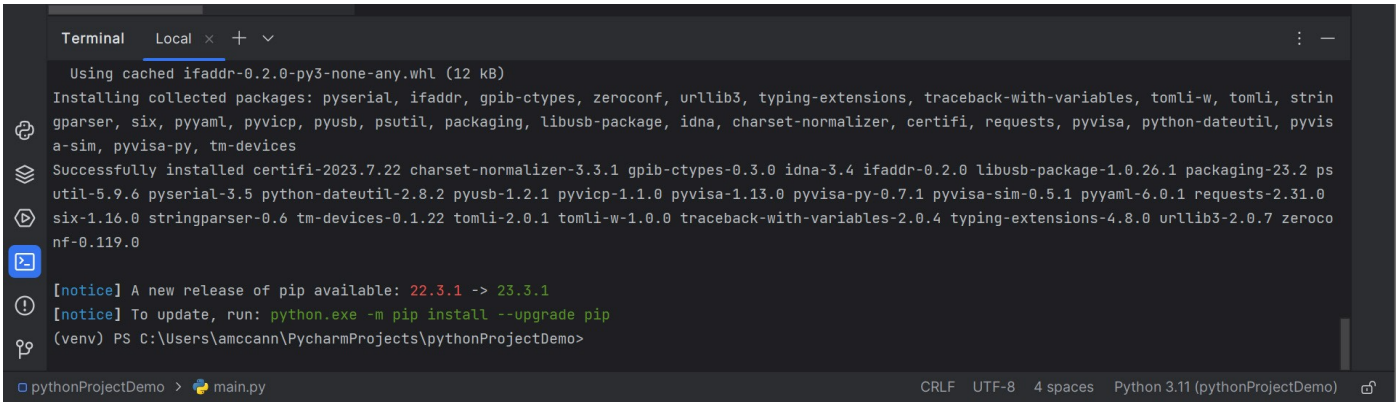
7. 打开一个终端。如果您的视图底部没有带标签的按钮，请查看以下内容：

8. 在终端提示符出现之前进行检查 (venv)，确认虚拟环境已设置。



9. 从终端安装驱动程序。

键入：pip install tm _ devices



10. 确保终端没有出错！开始使用吧！

Visual Studio Code

Visual Studio Code 是另一款各行各业的软件开发人员都在使用的免费 IDE。其适用于大多数语言，且具有针对大多数语言的扩展包，因此用户能够在此 IDE 中快捷高效地进行编码。Visual Studio Code 可提供 IntelliSense。IntelliSense 是非常实用的开发工具，有助于代码补全、参数信息以及其他有关对象和类的信息。tm_devices 支持对描述对象和类的命令树快速进行代码补全。

我们可提供一份关于 Python 和 Visual Studio Code 安装的操作指南，其中包括关于虚拟环境设置的信息，详情请点击此处。

示例代码

在本节中，我们将逐步介绍一个简单的代码示例，并重点介绍一些有效使用 tm_ devices 的必要组件。

导入

```
from tm_devices import DeviceManager
from tm_devices.drivers import MSO4B
```

这两行对于 tm_devices 的有效使用至关重要。在第一行中，我们导入 DeviceManager。这将处理多个设备类的样板连接和断开。

在第二行中，我们导入一个特定的驱动程序（在本例中是 MSO4B）。

我们用 DeviceManager 设置一个上下文管理器：

```
with DeviceManager(verbose=True) as device_manager:
```

然后，当我们同时使用设备管理器和驱动程序时：

```
    scope: MSO4B = device_manager.add_scope("192.168.1.1")
```

我们可以用与其型式匹配的特定命令集来实例化仪器。输入您仪器的 ip 地址（其他 VISA 地址也可）即可。

补全以上四行代码后，我们就可以开始为 MSO4B 编写特定的自动化代码了！

代码片段

接下来我们来看看几个简单的操作：

将触发类型设置为边沿触发

```
    # Setting Trigger A to Edge
    scope.commands.trigger.a.type.write("EDGE")
```

以下是添加和查询 CH1 峰 - 峰值测量值的方法：

```
    # Specifying source as Channel 1
    scope.commands.display.select.source.write("CH1")
    # Identifying pk2pk as the measurement we would like to make
    scope.commands.measurement.addmeas.write('PK2Pk')
    # Make sure the operation is complete using the opc command
    scope.commands.opc.query()
    # Store the value locally before we print
    ch1pk2pk = float(scope.commands.measurement.meas[1].results.allacqs.mean.query())
    # Printing the value onto the console
    print(f'Channel 1 pk2pk: {ch1pk2pk}')
```

如果您想对 CH2 进行幅值测量：

```
# Specifying source as channel 2
scope.commands.display.select.source.write("CH2")
# Identifying amplitude as the measurement we would like to make
scope.commands.measurement.addmeas.write('AMPLITUDE')
# Make sure the operation is complete using the opc command
scope.commands.opc.query()
# Store the value locally before we print
ch2amp = float(scope.commands.measurement.meas[2].results.allacqs.mean.query())
# Print the value onto the console
print(f'amplitude: {ch2amp}')
```

使用 IntelliSense/ 代码补全

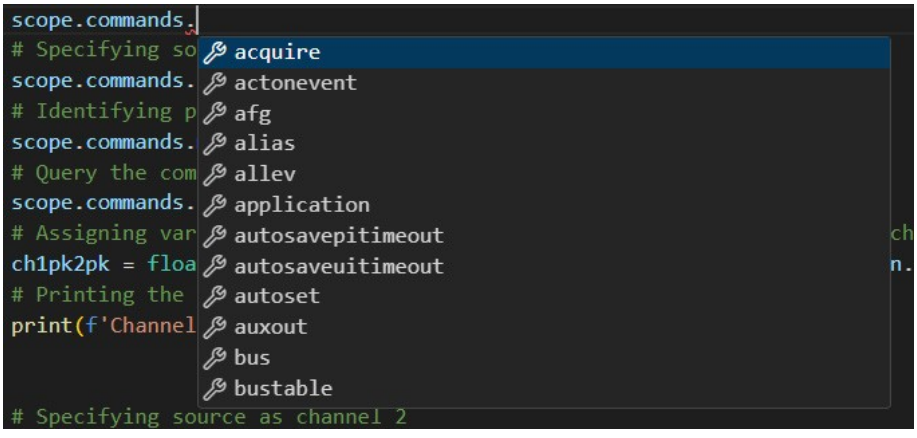
IntelliSense——微软的代码补全系统，也是我们在尽可能利用的 IDE 的一个强大功能。

SCPI 命令集是测试和测量设备自动化面临一大障碍。这是一种较为过时的结构，其语法在开发社区中并未得到广泛支持。

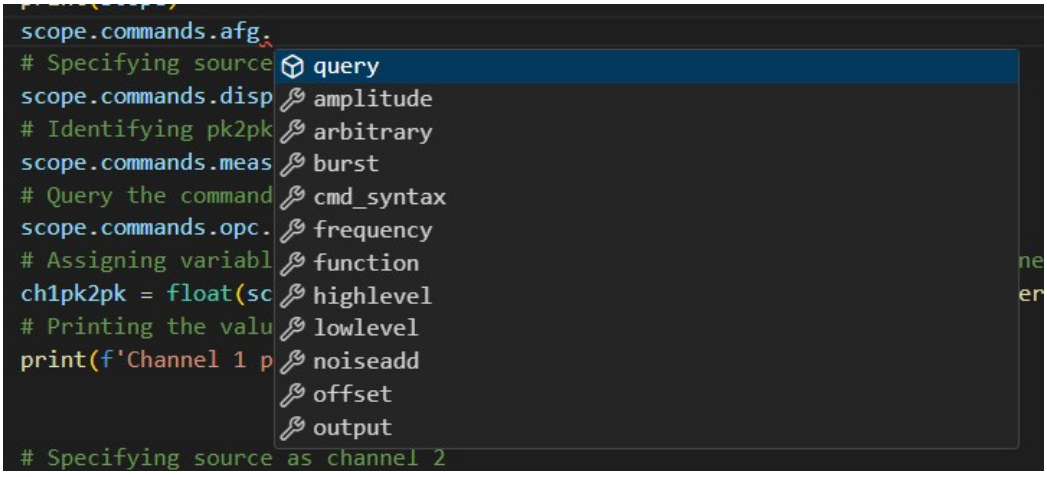
我们使用 tm_devices 所做的是为每个 SCPI 命令创建一组 Python 命令。我们可以从现有的命令语法中生成 Python 代码，避免手动开发驱动程序，并创建现有 SCPI 用户熟悉的结构。tm_devices 还可映射在程序创建期间可能需要特意调试的底层代码。Python 命令的结构模仿了 SCPI（或在某些吉时利案例中的 TSP）命令结构，因此如果您熟悉 SCPI，您就知道这些结构。

以下是关于“IntelliSense 如何显示先前键入的命令”的所有可用命令示例。

在 Scope 上的圆点后出现的可滚动列表中，我们可以看到按字母顺序排列的 Scope 命令类别列表：



选择 afg，我们可以看到 afg 类别列表。

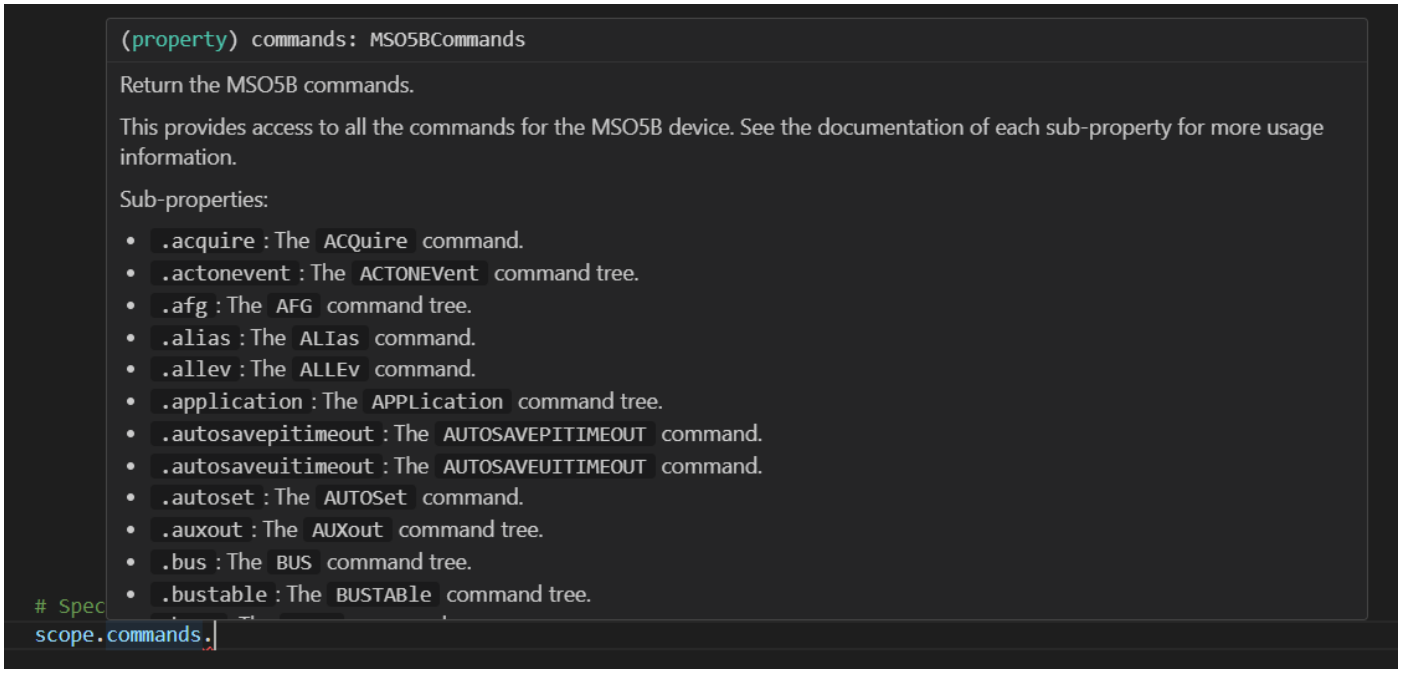


在 IntelliSense 的帮助下编写的最终命令：

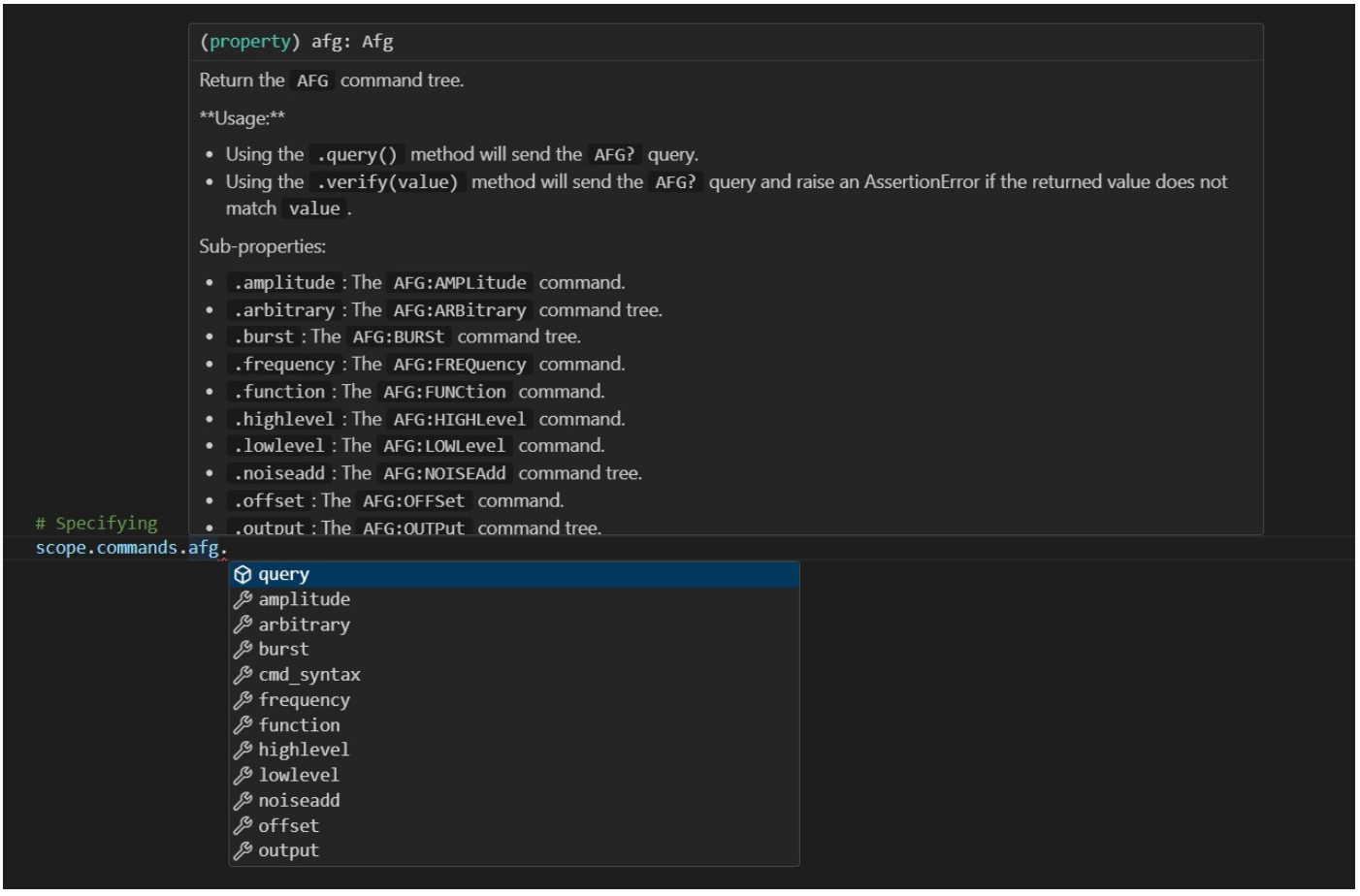
```
scope.commands.afg.amplitude.write(10e6)
```

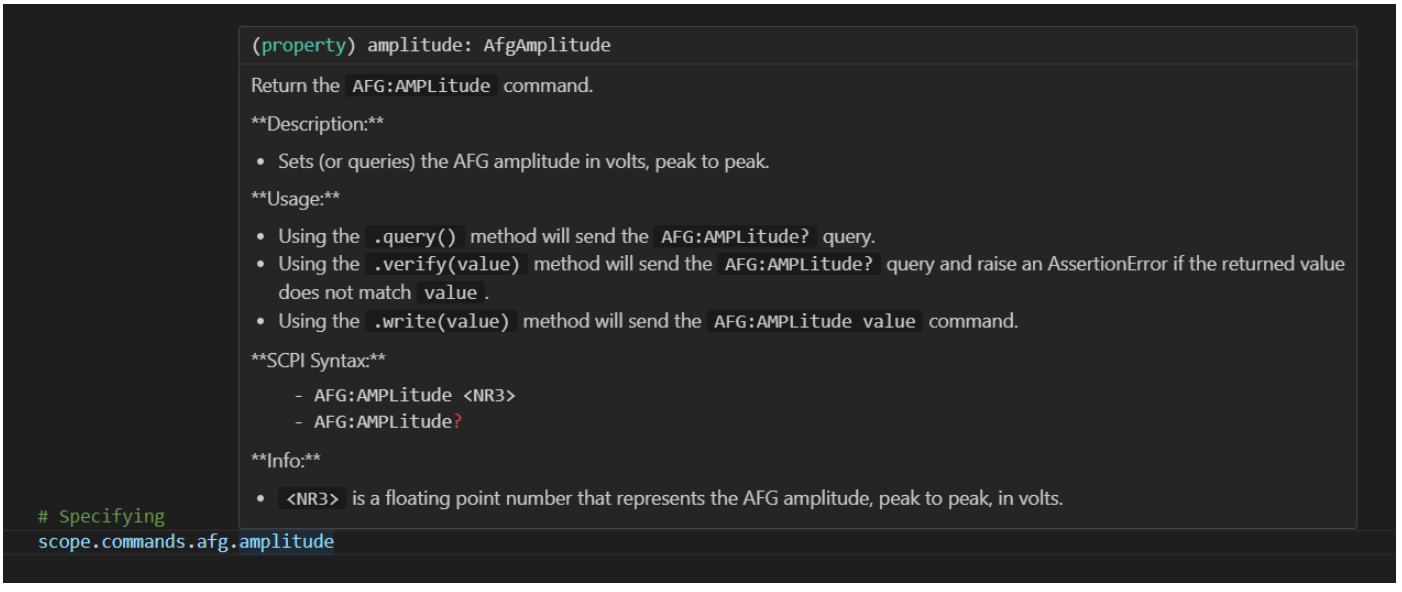
文档字符串帮助

编写代码之时，或者查看其他人的代码时，您可以将鼠标悬停在语法的不同部分上，以获得该级别的特定帮助文档。越接近完整的命令语法，其内容就越具体。



如果您想对 CH2 进行幅值测量：





阅读本指南后，您可以了解泰克 Python 驱动程序包 tm_devices 的部分优势，接下来便可以开始您的自动化之旅了。通过简单的设置、代码补全和内置帮助，您无需离开 IDE 就能学习，进而可以加快开发速度，并以更高的信心进行编码。

如果您希望改进软件包，Github repo 中有编辑指南。在文档和 examples 文件夹的软件包内容中突出显示了大量更高级的示例。

其他资源

tm_devices · PyPI——软件包驱动程序下载和信息

tm_devices Github——源代码、问题跟踪、编辑

https://github.com/tektronix/tm_devices#documentation ——在线文档

故障排除

通常情况下，故障排除的第一步是升级 pip：

在您的终端键入：Python.exe -m pip install -upgrade pip

错误：whl 看起来像文件名，但文件不存在或此平台不支持 whl。

```
(.venv) PS C:\pythonProject> python -m pip install pythonProject/tm_devices/tm_devices-0.1.0-py3-non-any.whl
WARNING: Requirement 'pythonProject/tm_devices/tm_devices-0.1.0-py3-non-any.whl' looks like a filename, but the file does not exist
ERROR: tm_devices-0.1.0-py3-non-any.whl is not a supported wheel on this platform.
(.venv) PS C:\pythonProject>
```

解决方案：在 pip 中安装 wheel，使其能够识别文件格式。

在您的终端键入：pip install wheel

如果您需要离线安装 wheel，您可以遵循与附件 A 类似的说明，但其需要下载 tar.gz 格式文件而非 .whl 文件。

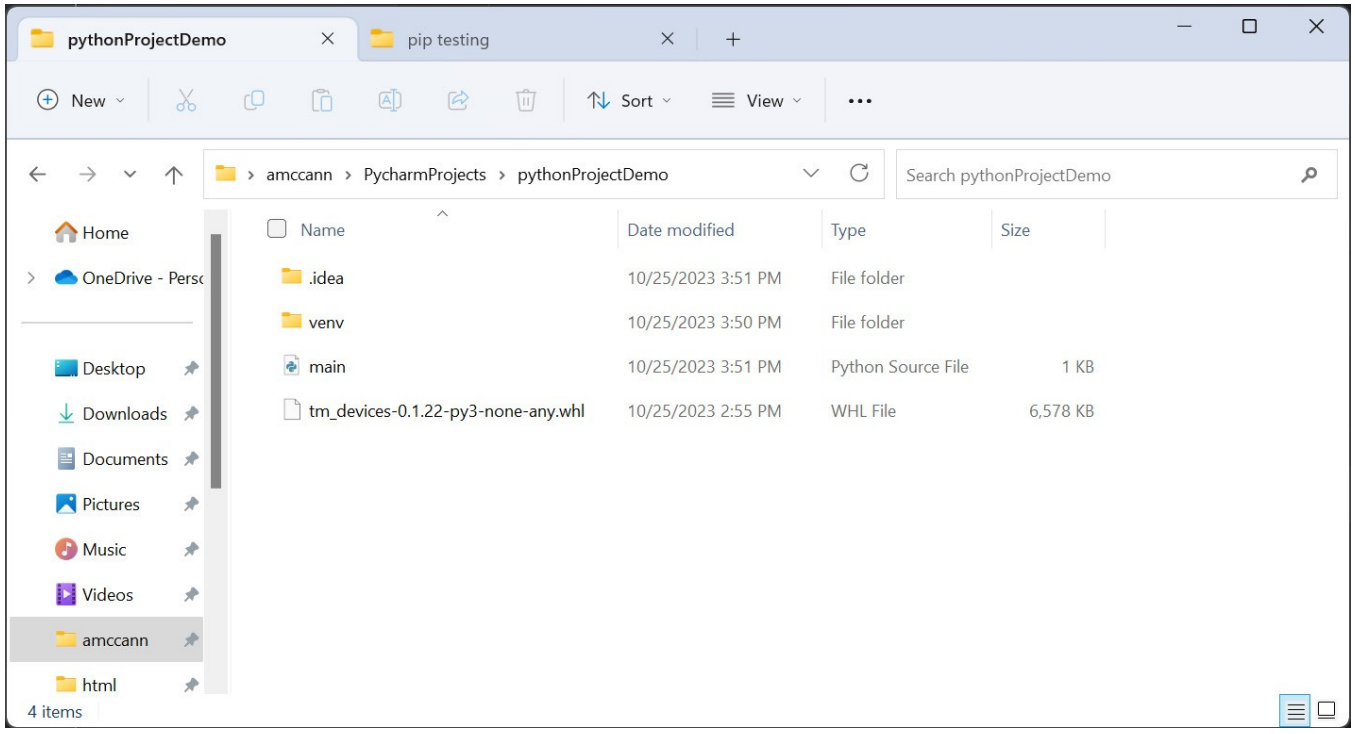
附件 A——tm_devices 的离线安装

在 PyCharm 中执行本地安装

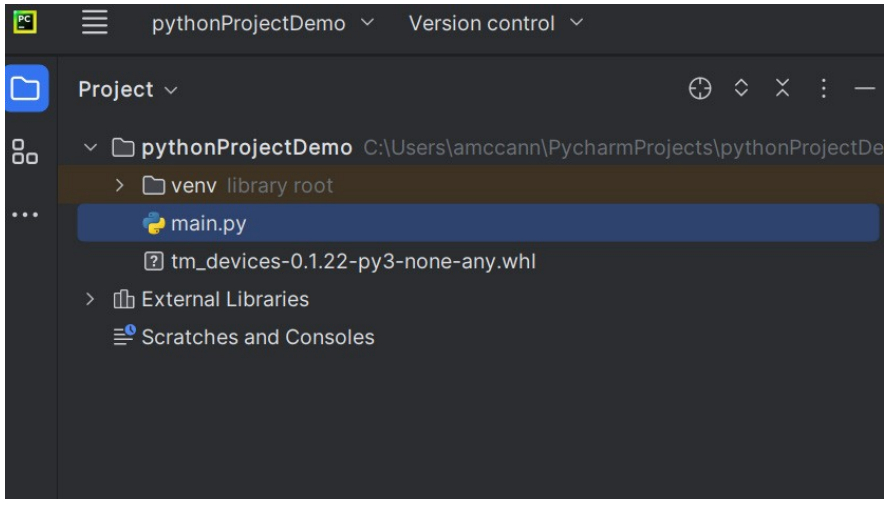
11. 在可访问互联网的计算机上，从以下位置下载最新的 tm_devices 驱动程序包：

    a. tm-devices · PyPI

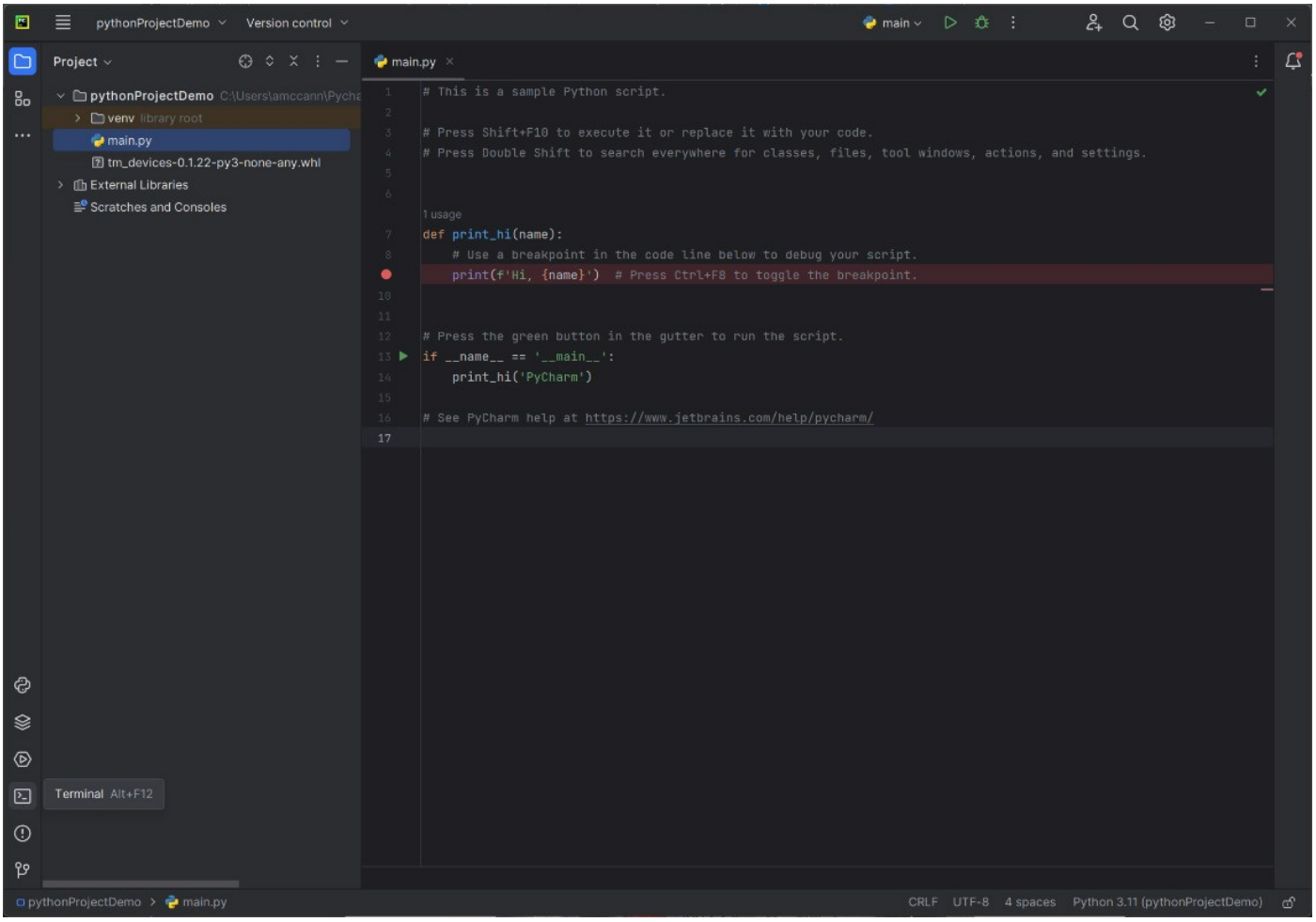
12. 将 .whl 文件移动到项目文件夹。



13. 返回 PyCharm，确保 .whl 文件在项目文件夹中。

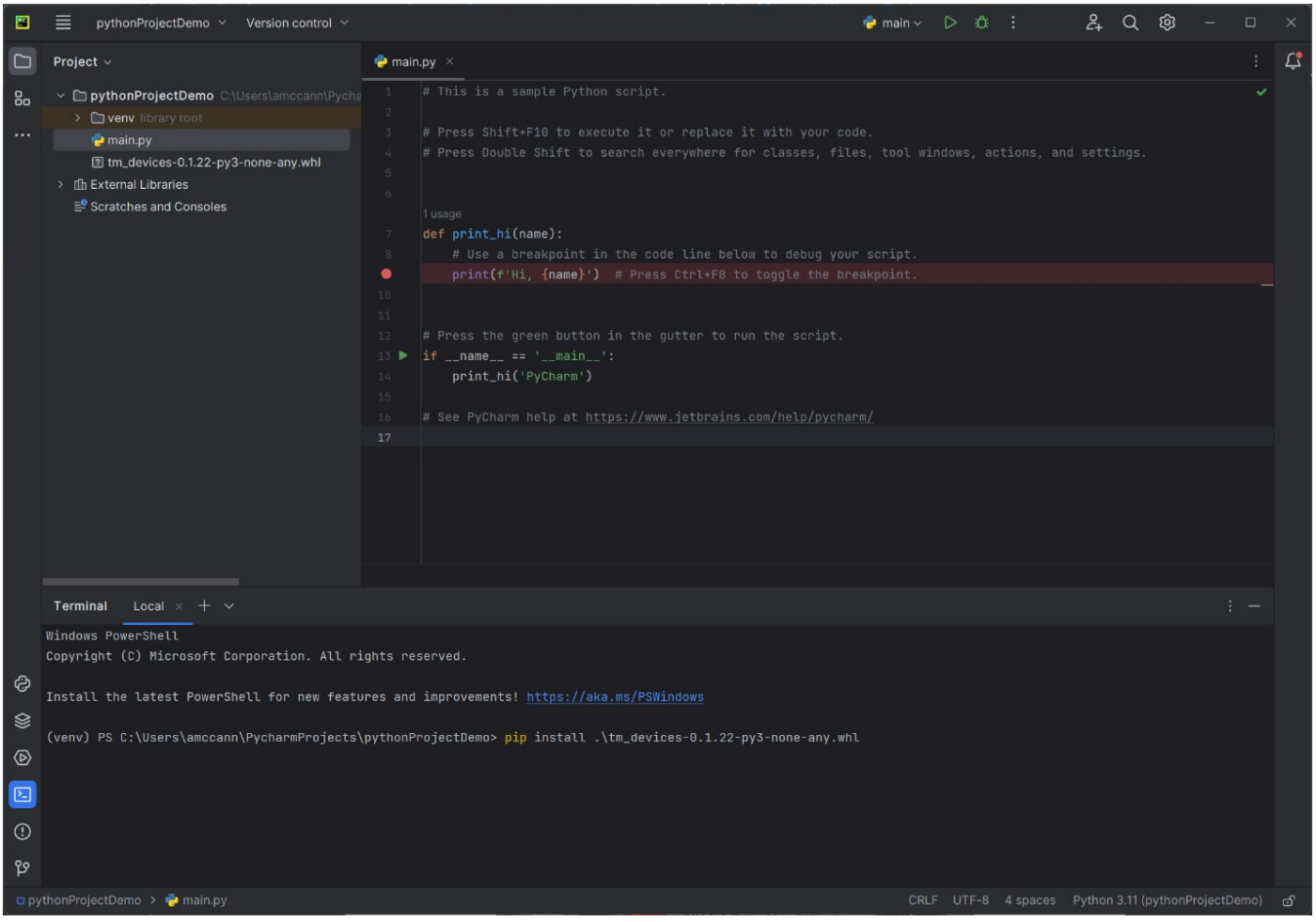


14. 打开一个终端。如果您的视图底部没有带标签的按钮，请查看以下内容：



15. 在您的终端键入：pip install <specific filename and version of tm_devices>.whl

    其应该类似于：tm_devices-1.0.0-py3-none-any.whl



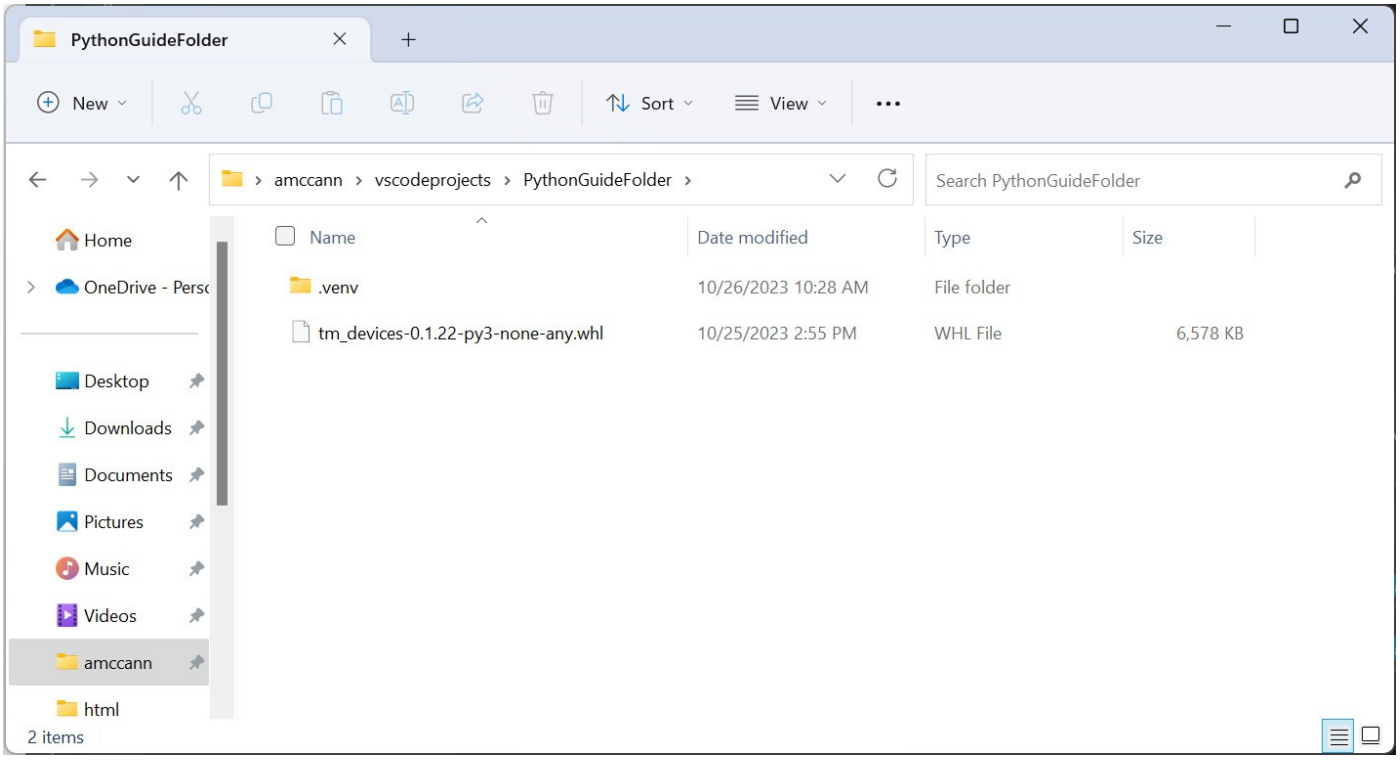
在 Visual Studio Code 执行本地安装

在本节中，我们将详细介绍如何在没有本地互联网连接的情况下安装 tm_devices 驱动程序包。

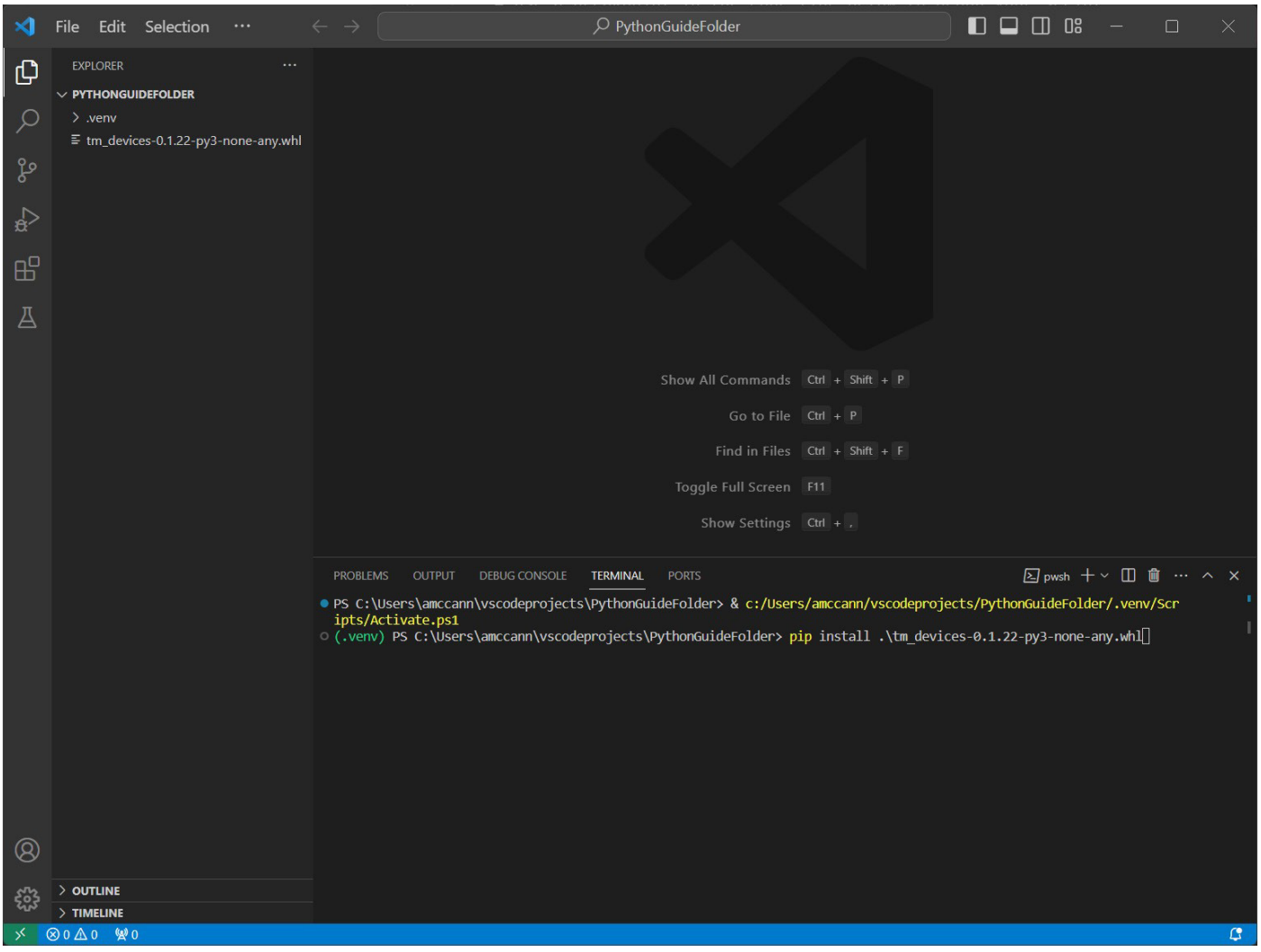
1. 在可访问互联网的计算机上，从以下位置下载最新的 tm_devices 驱动程序包：

    a. tm-devices · PyPI

2. 将 .whl 文件移动至您正在使用的项目文件夹中。



3. 在 Visual Studio Code 中导航至终端 (Ctrl+Shift+P-> 创建新终端)。本例使用的是虚拟环境，因此如果不是虚拟环境，其看起来可能会有所不同。



4. 确保此文件位于您的工作目录中，并键入：

   pip install <specific filename and version of tm_devices>.whl

   其应该类似于：tm_devices-1.0.0-py3-none-any.whl

5. 如果 tm_devices 导入过程没有出错，则安装成功。