# Instrument Automation with Python

## Installation Requirements

Before starting programming the automation, make sure to have all the software and packages below installed first.

### Python

**Step 1: Download Python**

1. **Visit the Official Python Website**:
   * Go to [python.org](https://www.python.org/downloads/) to find the latest version of Python.
2. **Download the Installer**:
   * Click on the "Download Python" button (the latest version will be recommended for your operating system).

**Step 2: Install Python**

1. **Run the Installer**:
   * Locate the downloaded installer and double-click it to run.
2. **Select Installation Options**:
   * **Important**: Check the box that says, "Add Python to PATH" before clicking "Install Now". This step is crucial as it makes Python accessible from the command line.
   * Follow the prompts to complete the installation.
3. **Verify Installation**:
   * Open the Command Prompt (Windows) or Terminal (macOS/Linux) and type: *python --version*
   * You should see the version of Python you installed.

**Step 3: Install PyVISA**

1. **Open Command Prompt or Terminal**:
   * Make sure you have internet access.
2. **Install PyVISA Using pip**:
   * Run the following command: *pip install pyvisa*
   * This command will download and install the PyVISA library.

### Visual Studio Code

For writing and running Python scripts, you can use a simple text editor or an Integrated Development Environment (IDE). Here are some popular options:

1. **Visual Studio Code (VS)**:
   * A lightweight and powerful code editor with extensions for Python. Download it from [code.visualstudio.com](https://code.visualstudio.com/).

### Jupyter Notebook

**Jupyter Notebook**:

* An interactive environment ideal for data analysis and visualization. You can install it via pip: *pip install notebook*
* Make sure to close and open VS before continuing

### NI-VISA

PyVISA needs a VISA implementation. This usually requires a backend to communicate with your instruments. The backend could either be a driver from National Instruments (NI-VISA) or the Python-based pyvisa-py. Let’s use NI-VISA because it is a standard driver for communication with instruments over USB, GPIB, and other protocols. It is widely used and supported, making it the most reliable choice.

* **Download NI-VISA**:
  + Go to the National Instruments website and download the latest version of [NI-VISA](https://www.ni.com/en/support/downloads/drivers/download.ni-visa.html).
* **Install NI-VISA**:
  + Run the installer and follow the prompts to install it. Make sure to restart your computer after installation.
* **NI-VISA Driver Wizard:**

### Testing Installation

Run the following code:

*import pyvis*

*rm = pyvisa.ResourceManager()*

*print(rm.list\_resources())*

## Example Code

import pyvisa

import numpy as np

import matplotlib.pyplot as plt

import logging

import sys

import time

# Set up logging

logging.getLogger('pyvisa').setLevel(logging.WARNING)

# Create a resource manager with logging

rm = pyvisa.ResourceManager()

# List available resources

print("Available Resources:")

resources = rm.list\_resources()

for resource in resources:

    print(resource)

# Connect to the instrument (replace 'YOUR\_INSTRUMENT\_ADDRESS' with the actual address)

instrument\_address = resource #'USB0::0x5345::0x1235::7320141::INSTR'

instrument = rm.open\_resource(instrument\_address)

# Send the SCPI command to get the ID

idn = instrument.query("\*IDN?")

# Print the identification string

print(f"Instrument ID: {idn}")

# Set the channel to be active (if needed)

instrument.write(":CH1:DISP ON")  # Turn on Channel 1 display

# Allow time for the instrument to configure

time.sleep(0.1)

instrument.write(":DATA:WAVE:SCREen:HEAD?")  # Send command to request header

raw\_header\_info = instrument.read\_raw()  # Read the raw response

# Allow time for the instrument to configure

time.sleep(0.1)

# Try decoding the raw header info

try:

    header\_info = raw\_header\_info.decode('utf-8')  # Attempt to decode using UTF-8

except UnicodeDecodeError:

    header\_info = raw\_header\_info.decode('latin-1')  # Fallback to Latin-1 if UTF-8 fails

print("Header Info:", header\_info)

# Step 2: Fetch the waveform data for Channel 1

instrument.write(":DATA:WAVE:SCREen:CH1?")  # Send command to request waveform data

# Read the binary data

raw\_data = instrument.read\_raw()  # Read the raw binary data

# Step 3: Convert the raw data from little-endian format

data\_points = np.frombuffer(raw\_data, dtype=np.uint16)  # Read as 16-bit unsigned integers

# Step 4: Convert to 12-bit values (mask the upper 4 bits)

processed\_data = data\_points & 0x0FFF  # Mask to get only the lower 12 bits

# Step 5: Plot the data

plt.figure(figsize=(10, 5))

plt.plot(processed\_data)

plt.title('Channel 1 Waveform')

plt.xlabel('Sample Number')

plt.ylabel('Amplitude (12-bit)')

plt.grid()

plt.show()

# Close the connection

instrument.close()