Controlling APx500 Using Python

When writing a script to control the APx500 software, the APx500 API must be referenced so that API commands can successfully compile at run-time. Because the API libraries are packaged into .NET assemblies (dll files) that can only be executed in a .NET runtime environment, the standard Python language cannot be used. Support for .NET libraries must be added using a custom version of the language (IronPython) or an add-on package (Pythonnet).

While Both IronPython and Pythonnet are workable solutions, Pythonnet may be preferred in situations where using standard Python or a newer version of Python is required. As of December 2020, IronPython is still based on Python 2.7, whereas Pythonnet supports multiple versions, up to version 3.8.

Note that although Python is a cross-platform language, API control of APx500 is only available on operating systems supported by the APx500 application. A list of supported operating systems can be found on our website at the link below.

AP Software Windows Version Compatibility

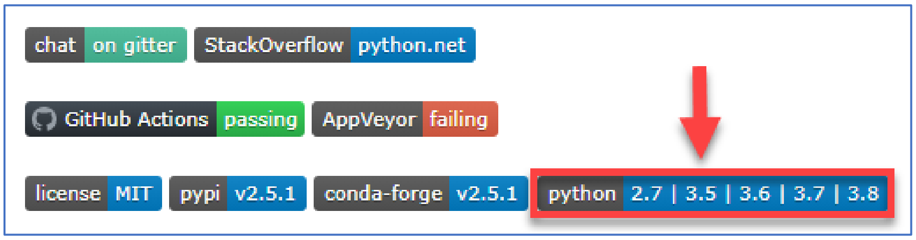
To get started controlling APx500 using Python and Pythonnet, follow the steps below.

Step 1: Install Python

Before installing Python, the list of versions supported by Pythonnet should be referred to using the Pythonnet GitHub repository available at:

Pythonnet GitHub Repository

Scroll down and look for a section indicating the list of supported Python versions. As of December 2020, the list of supported versions (as shown in the image below) are 2.7, 3.5, 3.6, 3.7, and 3.8.



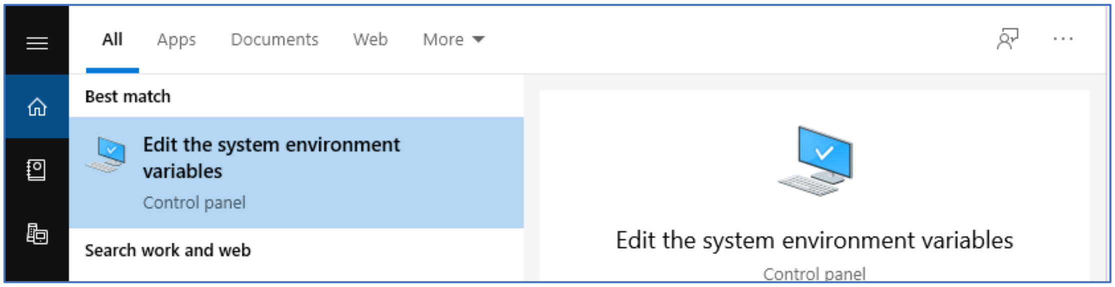
Once compatibility has been confirmed, download and install the relevant version of Python from:

www.Python.org

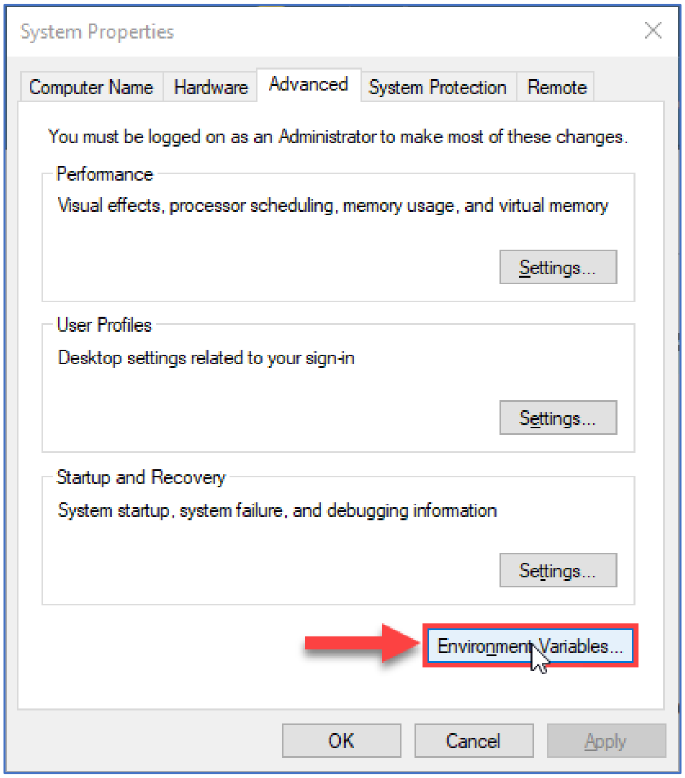
Step 2: Add Python to the Windows Path

Once Python has been installed, the Windows Path must be modified to allow Python commands to be executed from the Windows command prompt. To do so (in Windows 10):

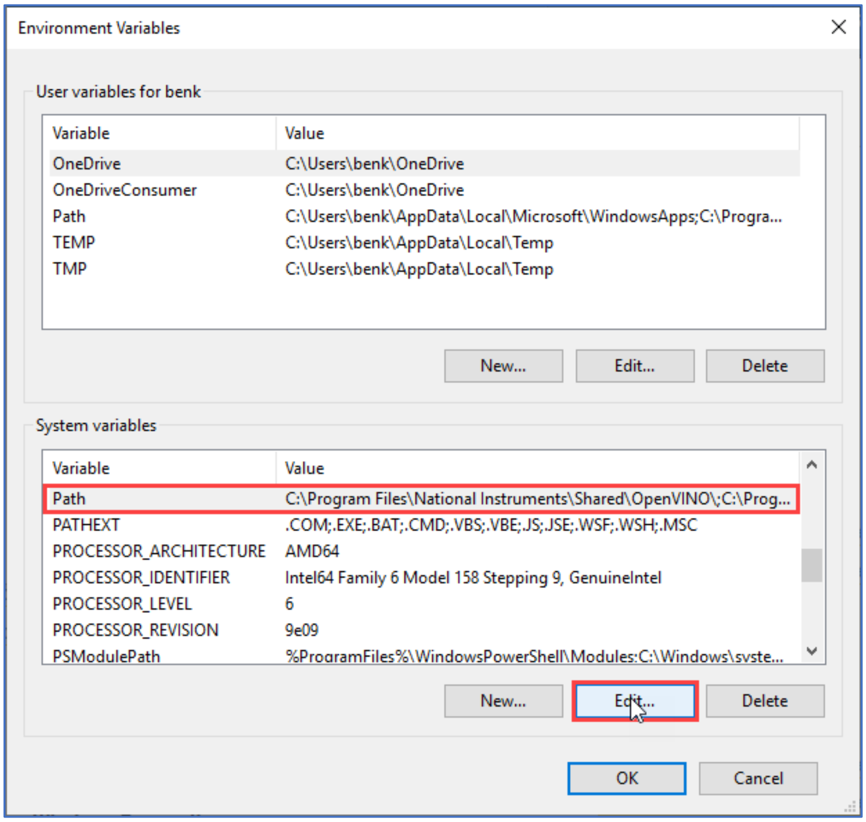
1. Use the Windows taskbar to search “Edit the system environment variables”. Press enter to Open the System Properties dialog.



2. Click the “Environment Variables...” button.


System Properties
Computer Name
Hardware
Advanced
System Protection
Remote
You must be logged on as an Administrator to make most of these changes.
Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...
User Profiles
Desktop settings related to your sign-in
Settings...
Startup and Recovery
System startup, system failure, and debugging information
Settings...
Environment Variables...
OK
Cancel
Apply

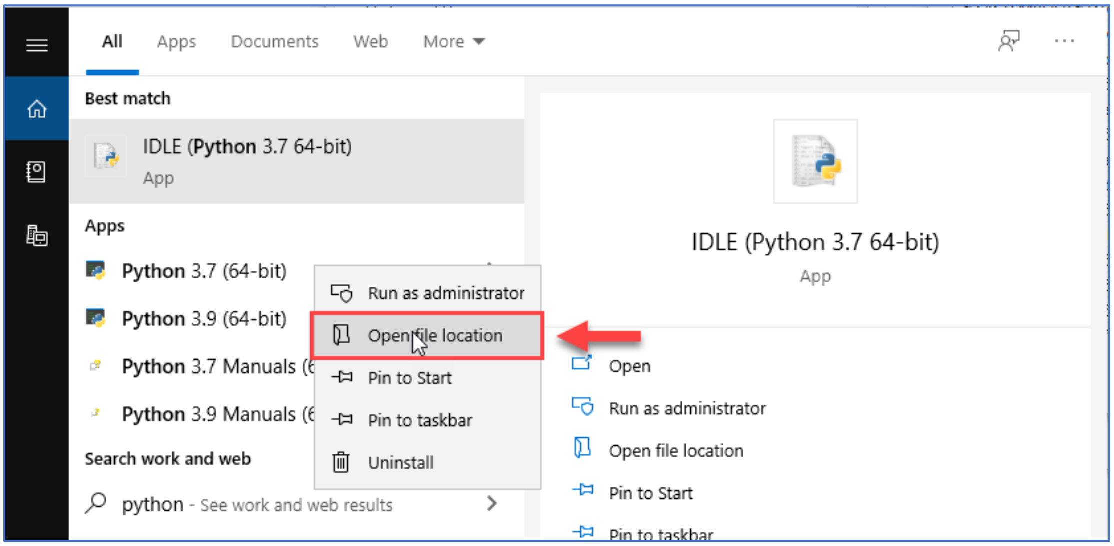
3. Under “System Variables”, find “Path” and click Edit…”.



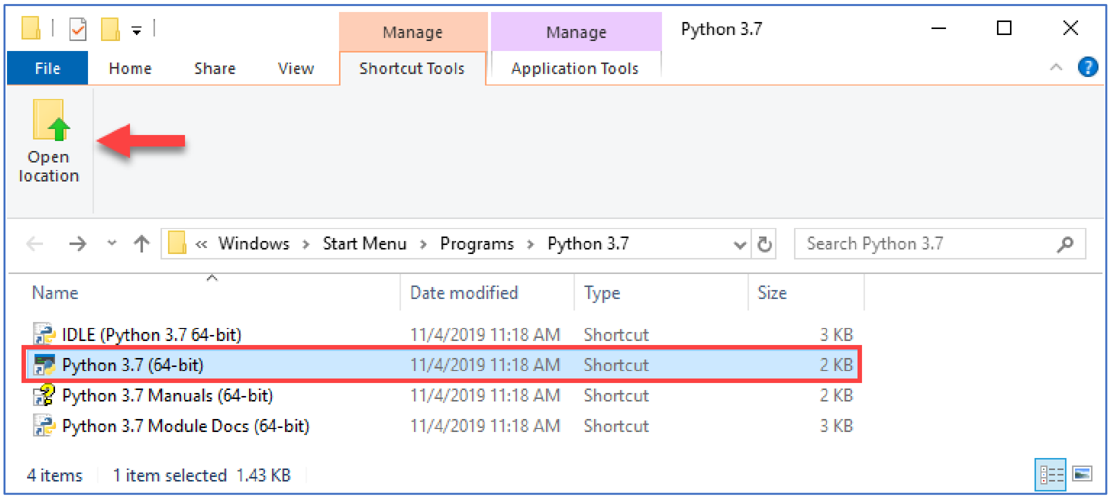
Variable	Value
OneDrive	C:\Users\benk\OneDrive
OneDriveConsumer	C:\Users\benk\OneDrive
Path	C:\Users\benk\AppData\Local\Microsoft\WindowsApps;C:\Progra...
TEMP	C:\Users\benk\AppData\Local\Temp
TMP	C:\Users\benk\AppData\Local\Temp

Variable	Value
Path	C:\Program Files\National Instruments\Shared\OpenVINO\;C:\Prog...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 158 Stepping 9, GenuineIntel
PROCESSOR_LEVEL	6
PROCESSOR_REVISION	9e09
PSModulePath	%ProgramFiles%\WindowsPowerShell\Modules;C:\Windows\syste...

4. To find the Python installation directory that will be added to the Path: using the Windows taskbar, search for “Python”. Right-click the version of Python you intend to use and select “Open file location”.



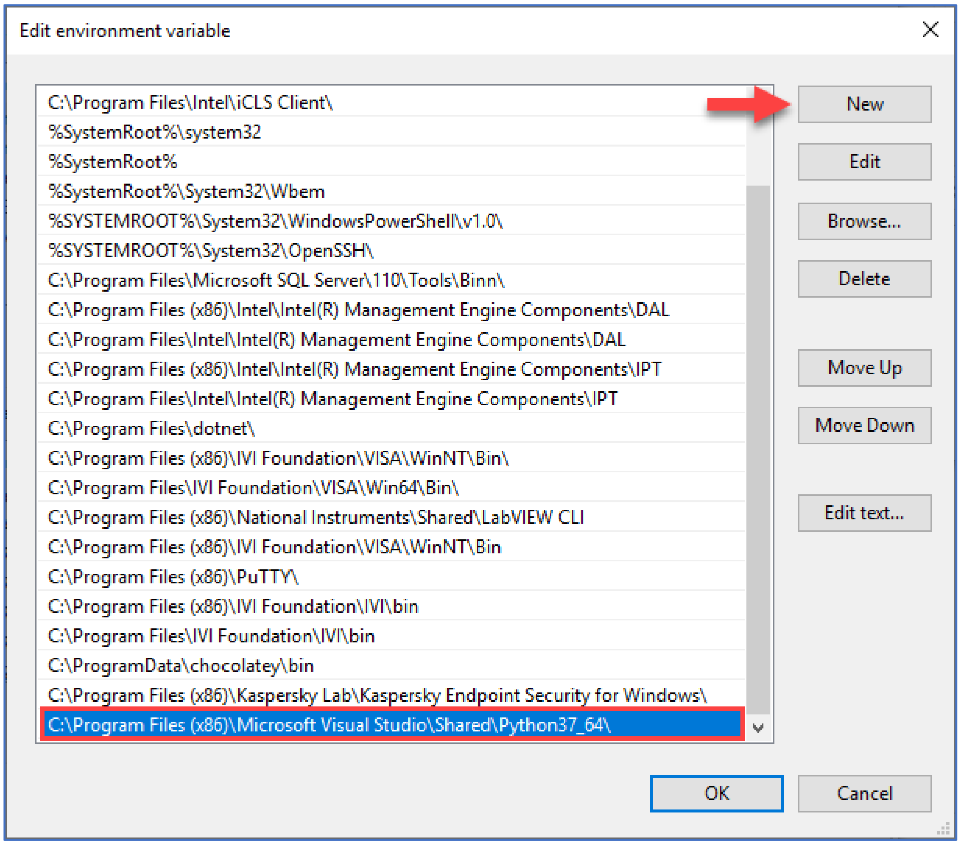
5. Select the shortcut to Python and click the “Open location” button on the “Manage Shortcut Tools” tab of the Windows file explorer.



Name	Date modified	Type	Size
IDLE (Python 3.7 64-bit)	11/4/2019 11:18 AM	Shortcut	3 KB
Python 3.7 (64-bit)	11/4/2019 11:18 AM	Shortcut	2 KB
Python 3.7 Manuals (64-bit)	11/4/2019 11:18 AM	Shortcut	2 KB
Python 3.7 Module Docs (64-bit)	11/4/2019 11:18 AM	Shortcut	3 KB

4 items 1 item selected 1.43 KB

6. Copy the full file path from the Explorer address bar.
7. Go back to the “Edit environment variable” dialog and click “New”.
8. Paste the copied Python path into the new path entry.



C:\Program Files\Intel\iCLS Client\
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\Microsoft SQL Server\110\Tools\Binn\
C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\DAL
C:\Program Files\Intel\Intel(R) Management Engine Components\DAL
C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\IPT
C:\Program Files\Intel\Intel(R) Management Engine Components\IPT
C:\Program Files\dotnet\
C:\Program Files (x86)\IVI Foundation\VISA\WinNT\Bin\
C:\Program Files\IVI Foundation\VISA\Win64\Bin\
C:\Program Files (x86)\National Instruments\Shared\LabVIEW CLI
C:\Program Files (x86)\IVI Foundation\VISA\WinNT\Bin
C:\Program Files (x86)\PuTTY\
C:\Program Files (x86)\IVI Foundation\IVI\bin
C:\Program Files\IVI Foundation\IVI\bin
C:\ProgramData\chocolatey\bin
C:\Program Files (x86)\Kaspersky Lab\Kaspersky Endpoint Security for Windows\
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python37_64\

9. Click “New” again, paste the copied path and append “\Scripts\” to the end.
10. Click OK to confirm changes in both the “Edit environment variable” and “Environment Variables” windows.

To confirm that Python was successfully added to the path, open a Windows Command Prompt and type “python –version” (two dashes). The Command Prompt should print out the version of Python added to the Windows path (e.g., Python 3.7.4).

Step 3: Install Pythonnet using a package manager

Once python has been installed and added to the Windows path, the Pythonnet package can be installed using pip, a package manager for Python. Pip is automatically installed with newer versions of Python.

To install Pythonnet using pip:

1. Open a Windows Command Prompt.
2. Type “pip install pythonnet”.

And that’s it! You are now ready to write code to control APx500 using Python and Pythonnet.

Tips on API usage

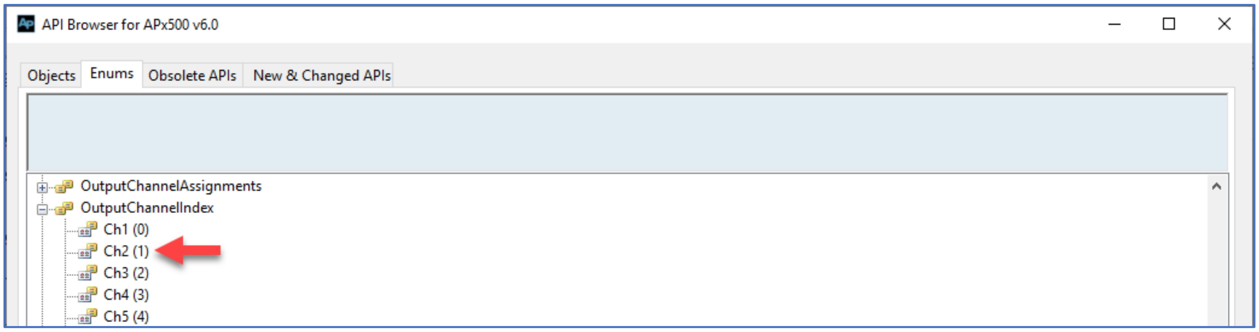
When calling a method that takes an argument of type Enum, the integer value of the enum should be used. For example, the following method:

```
APx.BenchMode.Generator.Levels.SetValue(ch as OutputChannelIndex, level as Double)
```

…should be called using the value “1” as the first argument to set the generator level on OutputChannelIndex Ch2:

```
APx.BenchMode.Generator.Levels.SetValue(1, 1.0)
```

The index values associated with an Enum can be looked up using the Enums tab of the API Browser (see image below).



When calling a method that takes a Double as an argument, the argument should be specified using a decimal separator so that the number is not interpreted as an integer. See second argument in “SetValue” example, above.

Example Script

Below is an example Python script written to control APx500 version 6.0 using Python (3.7.4) and Pythonnet (2.5.1). This example demonstrates how to reference the Audio Precision API, initiate the APx500 software and set it to visible, load a project file, run a sequence, and get data acquired in the sequence. The project file and source code used in this example can be found in a .zip file on our website. For more Python code examples, please check the Audio Precision website at www.ap.com or contact the Technical Support Department at techsupport@ap.com.

```
import sys, clr

clr.AddReference("System.Drawing")
clr.AddReference("System.Windows.Forms")

# Add a reference to the APx API
clr.AddReference(r"C:\Program Files\Audio Precision\APx500
6.0\API\AudioPrecision.API2.dll")
clr.AddReference(r"C:\Program Files\Audio Precision\APx500
6.0\API\AudioPrecision.API.dll")

from AudioPrecision.API import *
from System.Drawing import Point
from System.Windows.Forms import Application, Button, Form, Label
from System.IO import Directory, Path

class Container(Form):

    def __init__(self): # Create a form with a label and four buttons
        self.Text = 'APx Python Example'
        self.Height = 220
        self.Width = 200

        self.label = Label()
        self.label.Text = "Python APx Control"
        self.label.Location = Point(20, 20)
        self.label.Height = 20
        self.label.Width = 200

        self.count = 0

        bLoad = Button()
        bLoad.Text = "Load APx Software"
        bLoad.Location = Point(20, 50)
        bLoad.Width = 150
        bLoad.Click += self.APxLoad

        bProj = Button()
        bProj.Text = "Open Sample Project"
        bProj.Location = Point(20, 80)
        bProj.Width = 150
        bProj.Click += self.APxProject

        bMeas = Button()
        bMeas.Text = "Run Sequence..."
        bMeas.Location = Point(20, 110)
        bMeas.Width = 150
        bMeas.Click += self.APxMeasurement
        bData = Button()
        bData.Text = "Get Data"
        bData.Location = Point(20, 140)
        bData.Width = 150
        bData.Click += self.APxGetData

        self.Controls.Add(self.label)
        self.Controls.Add(bLoad)
        self.Controls.Add(bProj)
        self.Controls.Add(bMeas)
        self.Controls.Add(bData)

    def APxLoad(self, sender, args): # Initialize the software and set it to Visible
        APx = APx500_Application()
        APx.Visible = 1

    def APxProject(self, sender, args): # Load the example project file
        filename = "SampleProject.approjx"
        directory = Directory.GetCurrentDirectory()
        fullpath = Path.Combine(directory, filename)
        APx = APx500_Application()
        APx.OpenProject(fullpath)

    def APxMeasurement(self, sender, args): # Run the Frequency Response measurement
        APx = APx500_Application()
        APx.Sequence.Run()

    def APxGetData(self, sender, args): # Get results acquired in the last sequence run
        APx = APx500_Application()

        # Get Frequency Response RMS Level XY values by result name (string)
        xvalues = APx.Sequence[0]["Frequency Response"].SequenceResults["RMS
Level"].GetXValues(InputChannelIndex.Ch1, VerticalAxis.Left, SourceDataType.Measured, 1)
        yvalues = APx.Sequence[0]["Frequency Response"].SequenceResults["RMS
Level"].GetYValues(InputChannelIndex.Ch1, VerticalAxis.Left, SourceDataType.Measured, 1)

         # Get Frequency Response Gain XY values by result type
        xvalues = APx.Sequence[0]["Frequency
Response"].SequenceResults[MeasurementResultType.GainVsFrequency].GetXValues(InputChannel
Index.Ch1, VerticalAxis.Left, SourceDataType.Measured, 1)
        yvalues = APx.Sequence[0]["Frequency
Response"].SequenceResults[MeasurementResultType.GainVsFrequency].GetYValues(InputChannel
Index.Ch1, VerticalAxis.Left, SourceDataType.Measured, 1)

form = Container()
Application.Run(form)
```