AWR Scripting in Python

It is possible to write scripts for AWRDE in Python.

AWRDE is a COM automation server and any programming language that can perform as a COM client can be used. This includes Python. With Python you will need to install the win32com.client library.

Why use Python with AWR

By far the most common question we get regarding Python is when to use Python vs. Visual Basic. Each language has advantages and disadvantages and while most scripting tasks can be performed in either, some tasks are much easier in one vs the other.

When to use Visual Basic

- You already know VB and don't know Python
- You don't know any language and what to use whatever is easiest
- Your script will need a form or other UI to interact with the user
- You want to call your script from a menu or toolbar in MWO
- You want to save your script in your project file so other users can run it

When to use Python

- You know Python and don't want to learn Visual Basic
- You need to use libraries that only run in Python
- You want to create scripts that run from a command line
- You want to do analysis in a Jupyter notebook

If you decide to use Python this page will help get you setup. Currently, we assume that you already have Python installed and, if desired, setup a virtual environment to use AWR with. The steps below will go through installing the COM interface (win32com) and the AWR COM wrapper that will support intellisense (pyawr).

Installing win32com (pypiwin32)

Note it is recommended that you use Python version 3.7 or later with AWRDE although earlier versions may work.

The easiest way to do this is to use pip.

```
pip install pypiwin32
```

```
dane@PC-DANE C:\Python38\Scripts
> .\pip install pypiwin32
Collecting pypiwin32
  Using cached https://files.pythonhosted.org/packages/d0/1b/2f292bbd74
Collecting pywin32>=223 (from pypiwin32)
  Downloading https://files.pythonhosted.org/packages/a3/ca/d4011eb7f4c
3MB)
     || 9.3MB 6.4MB/s
Installing collected packages: pywin32, pypiwin32
Successfully installed pypiwin32-223 pywin32-300
```

This will install the package into your library. Once this is done the type library for AWRDE must be build. This is done by running the **makepy.py** script.

In a command prompt, perform the following...

```
cd [python install directory]\Lib\site-packages\win32com\client
python makepy.py
```

When prompted select the "*AWR Microwave Office 15 (1.0)*" entry in the menu. This script will then create a set of definitions that will be loaded automatically when you connect to AWRDE.

```
dane@PC-DANE C:\Python38\Lib\site-packages\win32com\client
> python makepy.py
Generating to C:\Users\dane\AppData\Local\Temp\gen_py\3.8\5AE2215C-270C-470B-A2B6-609A964E53A2x0x15x0.py
Building definitions from type library...
Generating...
Importing module
```

Testing your installation

Here is a very simple python program that connects to AWRDE and writes out all the models in the software:

```
# Simple script to test win32com installation

import win32com.client

awrde = win32com.client.Dispatch('AWR.MWOffice')  # connect to AWRDE, load AWRDE types

for model in awrde.Models:
        print(model.Name)
```

Raw Connections

Connecting to a Specific Version of AWRDE

When you use this line:

```
# Simple script to test interface to AWRDE
import win32com.client
awrde = win32com.client.Dispatch('AWR.MWOffice')  # connect to AWRDE, latest installed
```

It will connect to the most recent version of the AWR Design Environment installed, said another way, the version of the last AWR Design Environment Installer run.

To call a specific version, you will add the major version number such as:

```
# Simple script to test interface to AWRDE
import win32com.client
awrde = win32com.client.Dispatch('AWR.MWOffice.13.0')  # connect to AWRDE version 13.0
```

Connecting to a Specific Instance of AWRDE

If you have multiple instances of the AWR Design Environment running at the same time, you can follow the following procedure on how to pick which instance to connect.

First, in the AWR Design Environment, you need to go into the VB scripting and run

```
 Sub Main
   Debug.Print MWOffice.InstanceCLSID
End Sub
```

to get the CLSID of the instance you want to connect to. This will be something like:

`62F49D56-070F-4E6C-8AB9-25845CB94B9A`

Then, from within python, you use:

```
import win32com.client
obj = win32com.client.Dispatch("{CLSID}")
```

where you replace CLSID with the id you want to use.

For example

```
import win32com.client
obj2 = win32com.client.Dispatch("{62F49D56-070F-4E6C-8AB9-25845CB94B9A}")
```

Note the braces are required.

Connecting with pyawr

The pywin32 package contains everything needed to connect to AWRDE from Python but since Python is not a strongly typed language, editors do not have the information they need to provide advanced functionality such as IntelliSense. To solve this problem, AWR has created an interface layer on top of win32com which add the type information for the AWR COM interface. With this interface, editors such as Visual Studio Code (using the Visual Studio IntelliCode and Pylance plugins), can provide type hints.

The pyawr package is available at: https://github.com/danecollins/pyawr. See the README for instructions on installation.

Install using pip

Here is an example of installing pyawr using pip from a command prompt.

```
> pip install git+https://github.com/danecollins/pyawr

Collecting git+https://github.com/danecollins/pyawr
  Cloning https://github.com/danecollins/pyawr to c:\users\dane\appdata\local\temp\pip-req-build-9vh7xc7x
  Running command git clone -q https://github.com/danecollins/pyawr 'C:\Users\dane\AppData\Local\Temp\pip-req-
build-9vh7xc7x'
Building wheels for collected packages: pyawr
  Building wheel for pyawr (setup.py) ... done
  Created wheel for pyawr: filename=pyawr-0.11-py3-none-any.whl size=150587
sha256=65a1040b99b78b28d817d0c8c10a96f41985cfa2f56e8d4c02948915a819da0b
  Stored in directory: C:\Users\dane\AppData\Local\Temp\pip-ephem-wheel-cache-
yydfdc8k\wheels\3b\75\a7\33446d1f3729be5ed4e7d0ac513b9a19aac85bfc1784618dc7
Successfully built pyawr
Installing collected packages: pyawr
Successfully installed pyawr-0.11
```

Using pyawr

When using the mwoffice.py API wrapper layer, connecting to AWRDE will be done a little differently. To connect use the CMWOffice class.

```
import pyawr.mwoffice as mwo
awrde = mwo.CMWOffice()
```

This will create an object of type pyawr.mwoffice.CMWOffice which has type information defined for it.

To allow connections to different versions or a specific instance of AWRDE, CMWOffice has the following arguments:

- version which is the version of AWRDE such as 13.0, 14.0 or 15.0
    - If not specified then the last installed version will be invoked
- clsid which is the CLSID string as described above.

Helper Functions

pyawr also contains some helper functions which we have found quite useful when using python. The include:

- open_example() - a function to open any of the standard examples in the installer
- vbrange() - a function which returns a VB style index range. For example vbrange(2) returns 1, 2 (vb indexes start at 1 rather than 0)
- as_list() - a function which takes a VB collection and enumerates it into a list
- meas_from_graph() - returns the measurement objects in a graph
- class AwrMeas - returns an object from an MWO measurement including a pandas DataFrame of the data