Dan Boschen Workshop for the 2024 DSP Online Conference

Miniconda Installation Instructions

Dan Boschen

Last Updated: October 20, 2024

Installs:

Python 3.12

Jupyter Notebook 7.1.0

(The original document for this file is located in .MyCourses/2024 DSP Online Conf/)

Table of Contents

Indemnification 4
Disclaimer of Warranties, Limitations and Exclusions of Liability 4
Overview 5
Formatting Conventions 5
STEP 1: Install Miniconda Distribution 6
STEP 2: Install the Microsoft C++ Build Tools (for Windows Users Only) 8
STEP 3: Create Local Directory for Workshop Files 10
STEP 4: Install the Python Environment 10
STEP 5: Accessing Environments from Jupyter Notebook 12
STEP 6: Confirm Installation 13
STEP 7: Running the Notebook 16
Appendix A: Interoperating with an existing Python installation 18
Appendix B: Troubleshooting 21
  DLL Load failed related to scipy.signal 21
  Unhandled Exception in event loop 22
  DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. 22
  Firewall Issues Loading packages 23
  CondaSSLError: OpenSSL appears to be unavailable on this machine. OpenSSL is required to download and install packages. 24
  Solving environment: failed with initial frozen solve. Retrying with flexible solve. 25
Appendix C: File Sharing 26
Appendix D: Installing Python without Miniconda 27
Appendix E: Exporting / Restoring Environments 28
Appendix F: Changing Default Directory for Jupyter Notebooks 29

Appendix G: Updating Miniconda 30

Appendix H: Uninstalling Miniconda 30

Appendix I: Removing Environments 31

MORE DSP AND PYTHON with Dan! 32

Indemnification

By proceeding with the instructions in this document and associated workshop material, you are proceeding at your own risk in the use of the material and instructions provided and agree to defend, indemnify and hold completely harmless Dan Boschen, publisher and distribution partners from and against any and all claims, damages, costs and expenses arising from or related to your use of this document and the software tools referenced.

Disclaimer of Warranties, Limitations and Exclusions of Liability

This document provides instructions on installing open-source tools and software applications. Dan Boschen has provided these instructions as a resource as part of a course or workshop and in exchange for providing that resource, you expressly understand and agree that:

(a) Your use of these tools, software, instructions and the associated Jupyter Notebook files and all course or workshop material is completely at your own risk. These are provided on an "AS IS" and "AS AVAILABLE" basis and Dan Boschen disclaims all warranties of any kind, whether expressed or implied, including but not limited to the implied warranties of fitness for a particular purpose, merchantability, non-infringement, quality, performance, non-interference with information, and accuracy of packages or tools. There is no warranty that the packages, tools or files will fulfill any of your particular purposes or needs.

(b) Any files downloaded or otherwise obtained through these instructions is done at your own discretion and risk and you will be solely responsible for any damage to your computer system or loss of data that results from the download of any packages or tools. Dan Boschen does not warrant or make any representations concerning the accuracy, likely results or reliability.

You further expressly understand and agree that neither Dan Boschen, publisher, distribution partners or any affiliates shall be liable for any direct, indirect, incidental, special, consequential or exemplary damages, including but not limited to, damages for loss of profits, goodwill, use, data or other intangible losses (even if Dan Boschen or affiliates have been advised of the possibility of such damages), resulting from the use or the inability to use the material provided herein. If you are dissatisfied with the tools, application and instruction, your sole and exclusive remedy is to discontinue your use. Notwithstanding the foregoing, Dan Boschen and any affiliates liability to you in total sum shall in no case exceed any fees you may have paid to them to obtain this document. You further agree to not join in any lawsuit with another person or serve as a class representative of any class action lawsuit against Dan Boschen and any affiliates arising out of use of this document and the associated course or workshop material.

Overview

The following is a procedure for installing Miniconda which includes Python and Jupyter Notebooks used to demonstrate workshop material. Allow yourself at least one hour to complete this procedure.

To ensure the notebooks provided run seamlessly, please install Miniconda and associated python packages according to these instructions. The following steps were done on a Windows platform, but Miniconda can be installed on Windows, MacOS or Linux.

If you are an experienced Python user with a version of Python installed and do not want to interfere with your current set-up, proceed to "Appendix A: Interoperating with an existing Python installation" to create a separate environment for the workshop. This will allow you to continue to use what is already installed on your machine for other purposes beyond the workshop. If you have a version of Miniconda or Anaconda installed that is not used elsewhere and you want to do a clean install, completely uninstall the current version following the procedure in Appendix H: Uninstalling Miniconda.

Anaconda vs Miniconda:

Anaconda is a larger download that includes all libraries and modules commonly used in data science applications. We will instead use Miniconda which is a free minimal installer for Conda (an open-source package and environment manager) where any additional libraries needed, now or in the future, can be downloaded and installed as needed. This installation guide will include the installation of all libraries and modules needed for the workshop. Further, installing Miniconda and working in individual environments using only the packages needed for any project is recommended for recreating a development platform beyond the workshop, rather than installing the full Anaconda distribution, given Miniconda's smaller footprint and ease of portability. Additionally, the smaller number of overall packages facilitates problem-free subsequent updates and adding more tools in the future.

Formatting Conventions

Throughout this document the following conventions will be use:

Text indicating code you should enter verbatim will be shown using the following font:

```
jupyter notebook
```

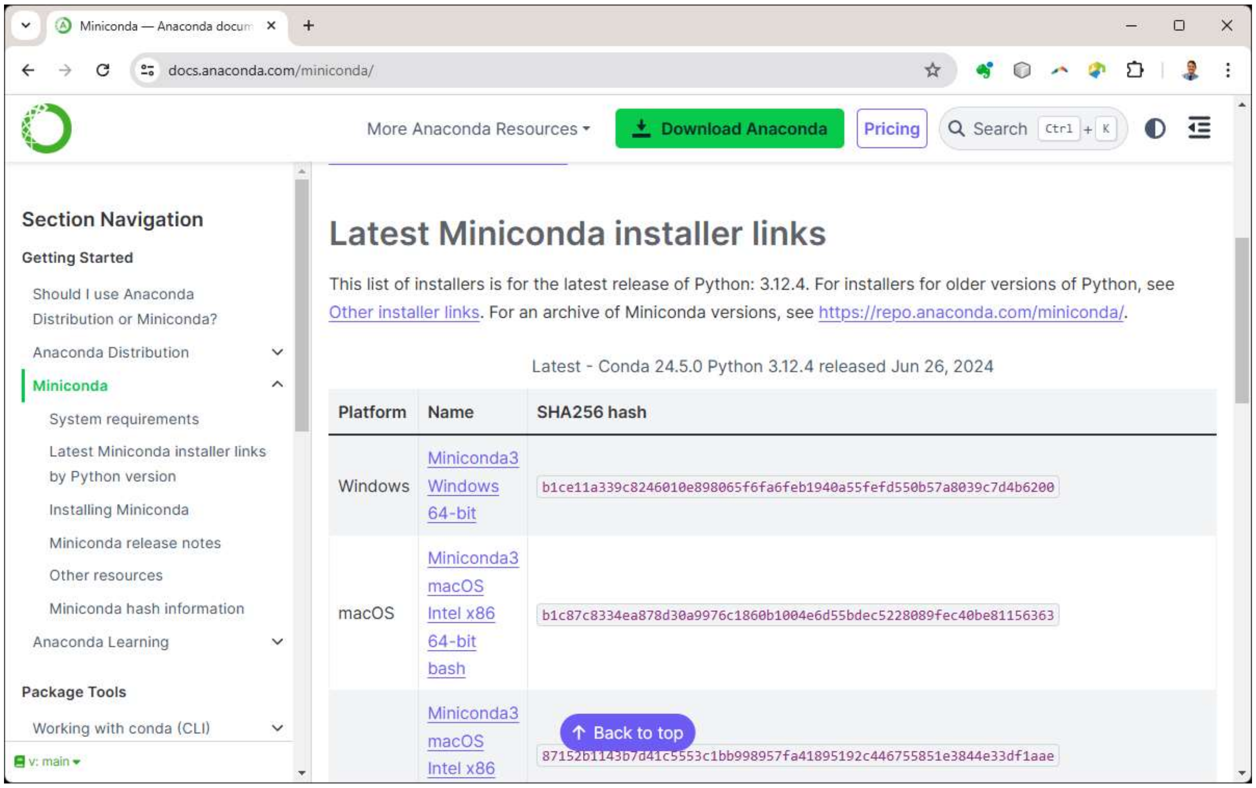
Text indicating generic code that should be replaced will be colored in red and enclosed in <>:

```
conda activate <env-name>
```

STEP 1: Install Miniconda Distribution

Follow the installation instructions for the latest version of Miniconda by going to the link below and scrolling down to the installer for your operating system. Do not be concerned if the Python version installed with Miniconda does not match what we will use for this workshop, as conda will support using different Python versions in the environment we will later create.

https://docs.conda.io/en/latest/miniconda.html



Latest Miniconda installer links

This list of installers is for the latest release of Python: 3.12.4. For installers for older versions of Python, see Other installer links. For an archive of Miniconda versions, see https://repo.anaconda.com/miniconda/.

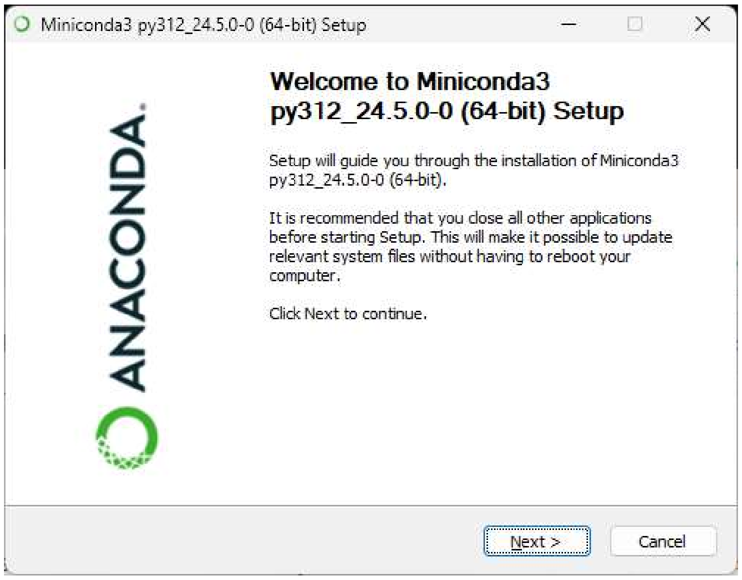
Latest - Conda 24.5.0 Python 3.12.4 released Jun 26, 2024

Platform	Name	SHA256 hash
Windows	Miniconda3 Windows 64-bit	b1ce11a339c8246010e898065f6fa6feb1940a55fefd550b57a8039c7d4b6200
macOS	Miniconda3 macOS Intel x86 64-bit bash	b1c87c8334ea878d30a9976c1860b1004e6d55bdec5228089fec40be81156363
	Miniconda3 macOS Intel x86	87152b1143b7d41c5553c1bb998957fa41895192c446755851e3844e33df1aae

Figure 1: Download Miniconda3 installer for your specific operating system and processor

The following details the expected installation process for a Windows 64-bit operating system specifically:

Once downloaded, launch the installer and press Next as shown in Figure 2.


Miniconda3 py312_24.5.0-0 (64-bit) Setup

Welcome to Miniconda3 py312_24.5.0-0 (64-bit) Setup

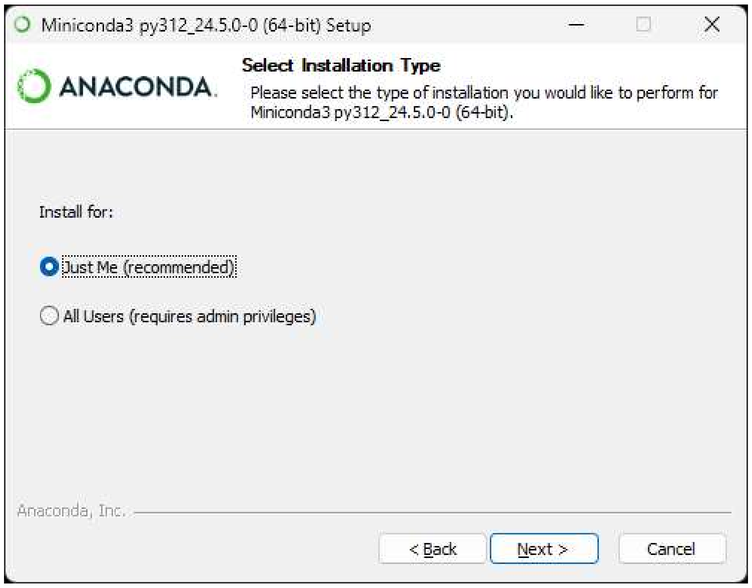
Setup will guide you through the installation of Miniconda3 py312_24.5.0-0 (64-bit).

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Figure 2: Miniconda installer first screen.

Press "I Agree" on the next License Agreement window and then select "Just Me (recommended)" as shown in Figure 3. **(Be sure to select “Just Me …” otherwise there will be installation issues later.)**


Miniconda3 py312_24.5.0-0 (64-bit) Setup

Select Installation Type

Please select the type of installation you would like to perform for Miniconda3 py312_24.5.0-0 (64-bit).

Install for:

[✓ Just Me (recommended)]

All Users (requires admin privileges)

Figure 3: Installation Type

Press Next to accept the default installation location (**unless there are spaces in the path names; Anaconda recommends for correct operation not installing into paths that contain spaces such as C:\Program Files, or that include Unicode characters outside the 7-bit ASCII character set**), and then press Install to accept the default Advanced Installation Options and complete the installation. Once the installation is complete, press Next and then press Finish on the final window.

STEP 2: Install the Microsoft C++ Build Tools (for Windows Users Only)

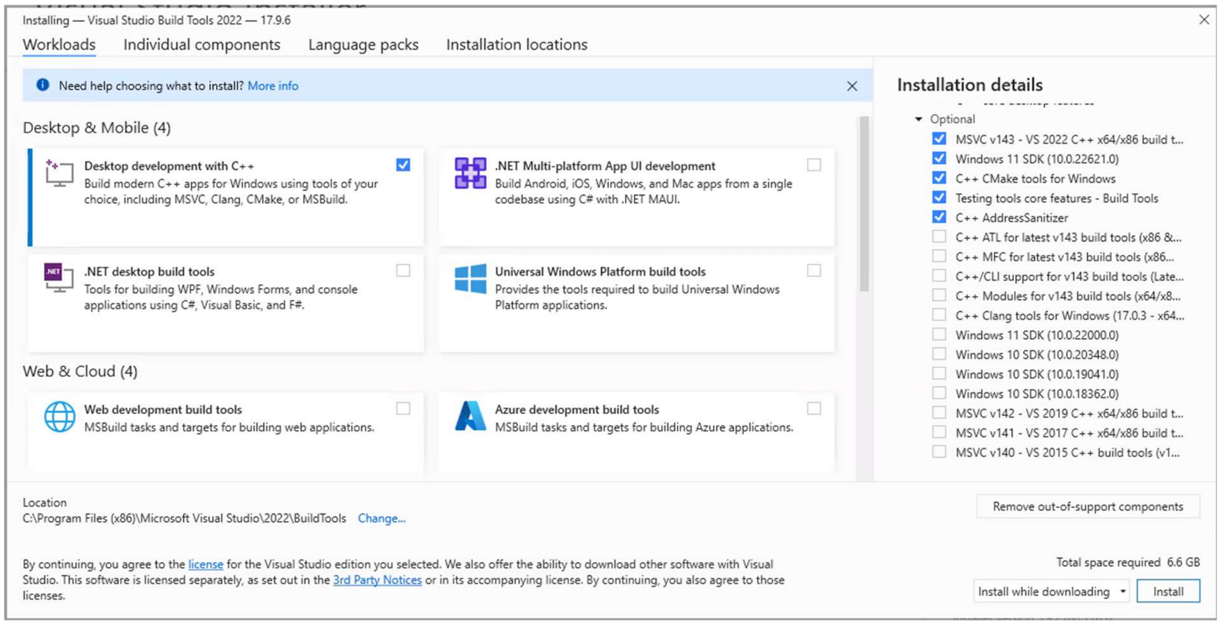
The following applies to Windows users only; if you are using MacOS or Linux, please proceed to "STEP 3: Create Local Directory for Workshop Files".

The Microsoft C++ Build Tools are necessary for Windows platforms in order to compile certain packages that may need to be installed using pip. Install the Build Tools by going to this link below and clicking on "Download Build Tools":

https://visualstudio.microsoft.com/visual-cpp-build-tools/

Proceed in downloading vs_BuildTools.exe, once downloaded, run the executable to start the installation.

A screen should appear similar to what is shown below in Figure 4. On this screen be sure to check off "Desktop development with C++". This will install the compiler needed by pip for distributions.



Installing — Visual Studio Build Tools 2022 — 17.9.6

Workloads | Individual components | Language packs | Installation locations

Need help choosing what to install? More info

Desktop & Mobile (4)

[✓] **Desktop development with C++** — Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.

[ ] **.NET Multi-platform App UI development** — Build Android, iOS, Windows, and Mac apps from a single codebase using C# with .NET MAUI.

[ ] **.NET desktop build tools** — Tools for building WPF, Windows Forms, and console applications using C#, Visual Basic, and F#.

[ ] **Universal Windows Platform build tools** — Provides the tools required to build Universal Windows Platform applications.

Web & Cloud (4)

[ ] **Web development build tools** — MSBuild tasks and targets for building web applications.

[ ] **Azure development build tools** — MSBuild tasks and targets for building Azure applications.

Installation details

Optional
- [✓] MSVC v143 - VS 2022 C++ x64/x86 build t...
- [✓] Windows 11 SDK (10.0.22621.0)
- [✓] C++ CMake tools for Windows
- [✓] Testing tools core features - Build Tools
- [✓] C++ AddressSanitizer
- [ ] C++ ATL for latest v143 build tools (x86 &...
- [ ] C++ MFC for latest v143 build tools (x86...
- [ ] C++/CLI support for v143 build tools (Late...
- [ ] C++ Modules for v143 build tools (x64/x8...
- [ ] C++ Clang tools for Windows (17.0.3 - x64...
- [ ] Windows 11 SDK (10.0.22000.0)
- [ ] Windows 10 SDK (10.0.20348.0)
- [ ] Windows 10 SDK (10.0.19041.0)
- [ ] Windows 10 SDK (10.0.18362.0)
- [ ] MSVC v142 - VS 2019 C++ x64/x86 build t...
- [ ] MSVC v141 - VS 2017 C++ x64/x86 build t...
- [ ] MSVC v140 - VS 2015 C++ build tools (v1...

Location
C:\Program Files (x86)\Microsoft Visual Studio\2022\BuildTools Change...

Remove out-of-support components

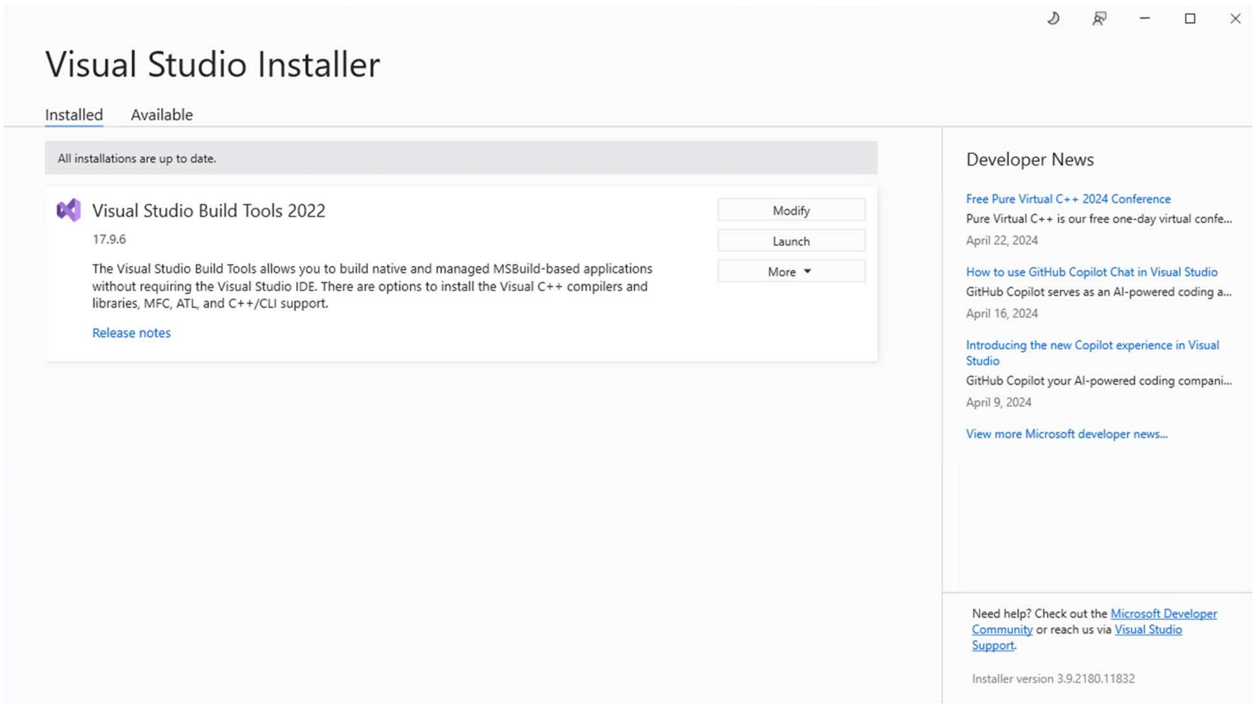
By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

Total space required 6.6 GB

Install while downloading | Install

Figure 4: Microsoft Visual Studio.

Press Install to proceed. Once complete the Visual Studio Installer screen will show the status similar to Figure 5 below. The installation is complete and this screen can now be closed.


Visual Studio Installer

All installations are up to date.

Visual Studio Build Tools 2022

17.9.6

The Visual Studio Build Tools allows you to build native and managed MSBuild-based applications without requiring the Visual Studio IDE. There are options to install the Visual C++ compilers and libraries, MFC, ATL, and C++/CLI support.

Developer News

Free Pure Virtual C++ 2024 Conference
Pure Virtual C++ is our free one-day virtual confe...
April 22, 2024

How to use GitHub Copilot Chat in Visual Studio
GitHub Copilot serves as an AI-powered coding a...
April 16, 2024

Introducing the new Copilot experience in Visual Studio
GitHub Copilot your AI-powered coding compani...
April 9, 2024

View more Microsoft developer news...

Need help? Check out the Microsoft Developer Community or reach us via Visual Studio Support.

Installer version 3.9.2180.11832

Figure 5: Conclusion screen for installation.

STEP 3: Create Local Directory for Workshop Files

Using File Explorer (the Finder on a Mac), navigate to your top-level home directory (for example on a windows machine: C:\Users\bosch) and create a directory called 'workspace' where we will keep all the files used for the workshop. Please use this directory specifically as that will be the default set-up for where Jupyter Notebook opens.

Unzip the files distributed for the workshop and place below the workspace folder preserving the same directory structure. The files should include:

16QAM.ipynb
4096QAM.ipynb
ADC Example.ipynb
ADEV Example.ipynb
(Folder) src

(The src folder contains boschen_tools.py. Additional files may be added by the time of the workshop).

STEP 4: Install the Python Environment

Open the "Anaconda Prompt". On a Windows machine, this can be launched from the Windows Start Menu from the Miniconda3 folder, or just typing "Anaconda" in the "Type Here to Search" field next to the Start button in the lower left-hand corner of the screen (as shown in Figure 6). Do NOT select "Anaconda Powershell Prompt (miniconda3)". On a macOS and Linux machine, open a terminal window and type `conda activate base` to have the equivalent of the 'Anaconda Prompt'.

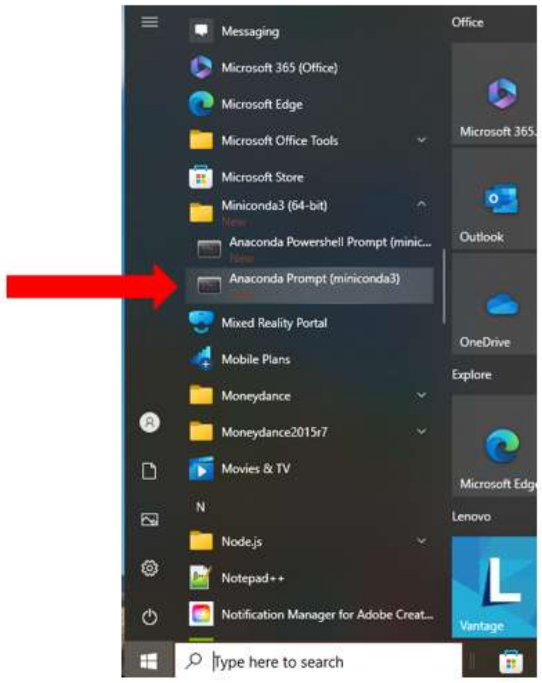


Figure 6: Opening "Anaconda Prompt" on Windows

The Anconda Prompt should open in your ~home directory and you should see (base) at the command prompt.

Enter the following in the Anaconda Prompt. Be careful about copying and pasting from this document, Windows may render the “—" incorrectly). The following will create an environment name ‘dspconf2024’ which you can replace with any other name desired. If you do use another name, be sure to be consistent with that change later in the document wherever you see ‘dspconf2024’ appear. Be sure to install the specific versions as shown below as they have been confirmed to work well together.

```
conda config --add channels conda-forge
conda create –-name dspconf2024 python=3.12.2
conda activate dspconf2024
conda install scipy=1.12.0 matplotlib=3.8.3 notebook=7.1.0
conda install ipympl=0.9.3
conda install allantools=2024.6
pip install komm==0.9.1
pip install fpbinary==1.5.8
```

STEP 5: Accessing Environments from Jupyter Notebook

When Jupyter Notebook is launched from within an environment, it does not by default access that environment. Environments must be registered with Jupyter by installing ipykernel in the environment and then adding that environment to Jupyter's list of kernels. This section explains how to add an environment so that it can be selected from within Jupyter once a notebook is opened.

To add an environment to Jupyter Notebooks, from the Anaconda Prompt, first activate the environment if not already active by typing the following:

```
    conda activate dspconf2024
```

Add the environment to Jupyter Notebook by typing the following in the console with the selected environment activated (Important! Confirm the environment name appears in parenthesis to the left of the prompt, otherwise it is not activated and you are working in a different environment!). The --name will set the name that you see in the list of Kernels within Jupyter Notebooks. This can be any text desired but it is recommended to use the environment name to minimize confusion.

```
    ipython kernel install --user --name=dspconf2024
```

You may see a “Debugger warning:” appear with a message about that. This can be ignored as we will not be debugging any of the modules that were installed (so they can remain frozen).

If you actually read this far then you deserve to step back and enjoy a DSP Joke:

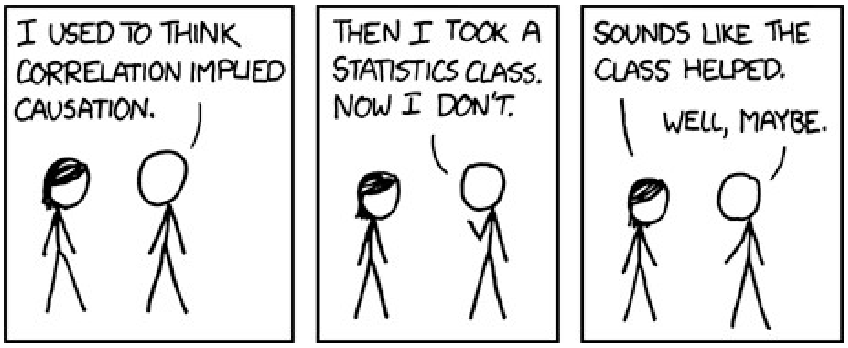

I USED TO THINK CORRELATION IMPLIED CAUSATION.
THEN I TOOK A STATISTICS CLASS. NOW I DON'T.
SOUNDS LIKE THE CLASS HELPED.
WELL, MAYBE.

Figure 7: Momentary Distraction. Credit Randall Munroe, xkcd

Ok, back to task… The environment will now appear in Jupyter Notebook under Kernel-Change Kernel once a notebook is opened. This will be demonstrated in STEP 7: Running the Notebook.

An environment can later be removed from the list of Kernels in Jupyter Notebook by typing in the Anaconda Prompt console from any environment, and added later by repeating command above (replacing <env-name> with actual name used):

```
    jupyter kernelspec uninstall <env-name>
```

STEP 6: Confirm Installation

Perform the following steps to confirm everything is working properly.

Open the Anaconda Prompt as explained in “STEP 4: Install the Python Environment”

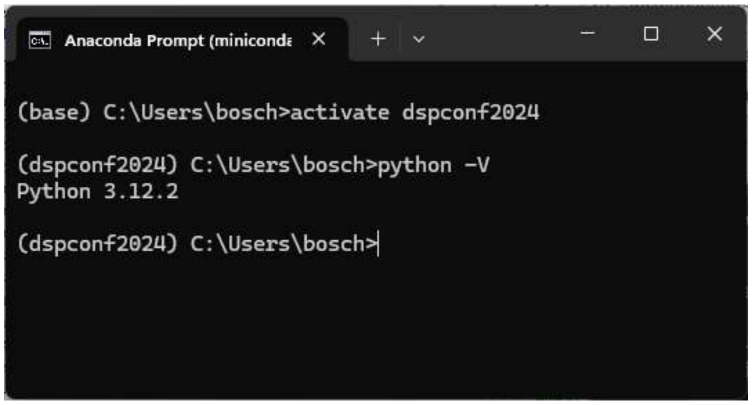
View a list of all environments by typing:

```
conda env list
```

This should include the base environment in addition to the environment installed. Note the name of the environment and activate it if not currently active by typing the following, replacing <*env-name*> with the actual environment name used:

```
conda activate <env-name>
```

From the Anaconda Prompt console with the environment activated, type `python -V` and press enter. Confirm the response with the Python version as shown in the screen capture below (be sure to use a capital V! Lower case v means "verbose"):



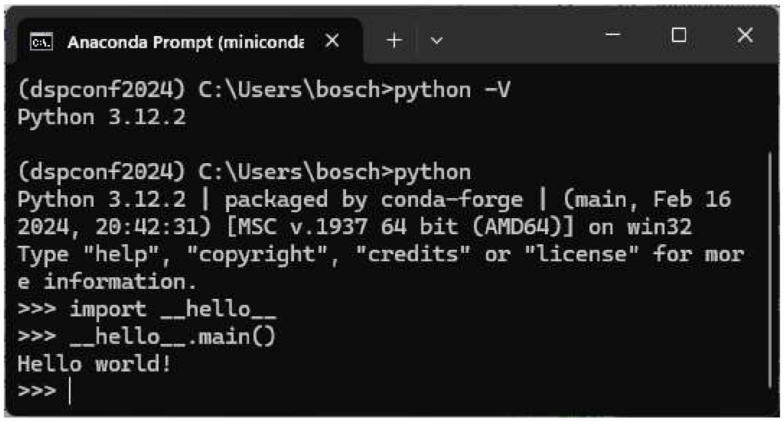
```
(base) C:\Users\bosch>activate dspconf2024

(dspconf2024) C:\Users\bosch>python -V
Python 3.12.2

(dspconf2024) C:\Users\bosch>
```

Figure 8: Displaying Python version installed

To confirm installation and operation, the following are some of the Easter Eggs to try that are buried in Python. Start python by typing `python` in the Anaconda Prompt after activating the workshop environment as directed above, and then from the Python command prompt, type

```
import __hello__
__hello__.main()
```

(Note that there are two underscores in a row before and after hello) and confirm that you get the “Hello world!” response as shown below.



```
(dspconf2024) C:\Users\bosch>python -V
Python 3.12.2

(dspconf2024) C:\Users\bosch>python
Python 3.12.2 | packaged by conda-forge | (main, Feb 16
2024, 20:42:31) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for mor
e information.
>>> import __hello__
>>> __hello__.main()
Hello world!
>>> 
```

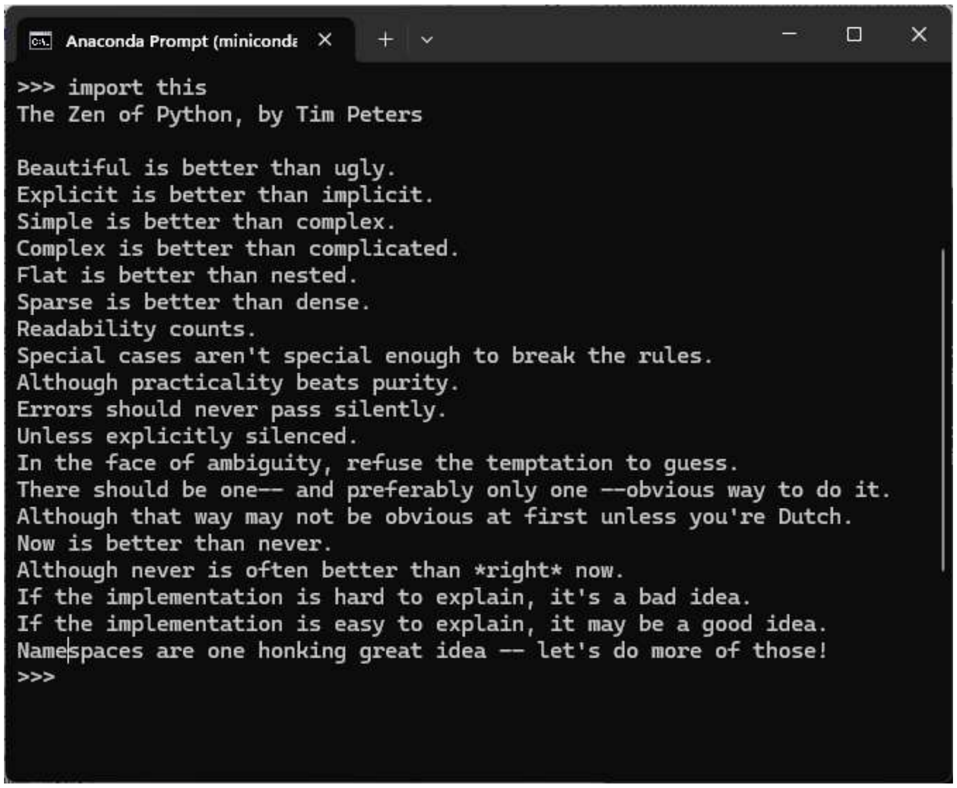
Figure 9: Hello World!

Congratulations if that was your first Python program!

Next, type `import this`

- The Zen of Python, a poem by Tim Peters should print in the console.

Hey! If you are reading this and actually doing all this then please e-mail me at boschen @ loglin dot com. I applaud your interest and want to remember you!



```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

Figure 10: Figure 10: Result of import this

Next, type `import antigravity`

- A browser should open displaying an xkcd comic about Python

Confirm that the Jupyter Notebook is properly installed by running Jupyter Notebook. Type `exit()` if still in the Python interpreter, and then type `jupyter notebook` from the Anaconda Prompt in the activated environment.

If successful, a browser window like the screenshot below will appear with a file listing from the default starting directory (this should be the ~home directory described earlier where the workspace folder from Dropbox was copied over). The file listing is identical to what you would see through File Explorer using Windows or The Finder on a Mac; the File Tab through Jupyter Notebooks is just another view of your file system. Importantly if the above procedure was followed, you should see a 'workspace' folder which has the course files. If a different default start-up directory is later desired, this is detailed in "Appendix F: Changing Default Directory for Jupyter Notebooks".

The Anaconda Prompt will display several server messages and can be minimized but must not be closed or the Jupyter server now running in the background will stop. You may see warning messages in the console "Config option `template path` not recognized by `LenvsLatexExporter` - ignore this as it has no impact on anything we will do. There is a fix online that suggests downgrading the nbconvert module; Don't do this! That will resolve the warning messages but cause things we do use to no longer work.

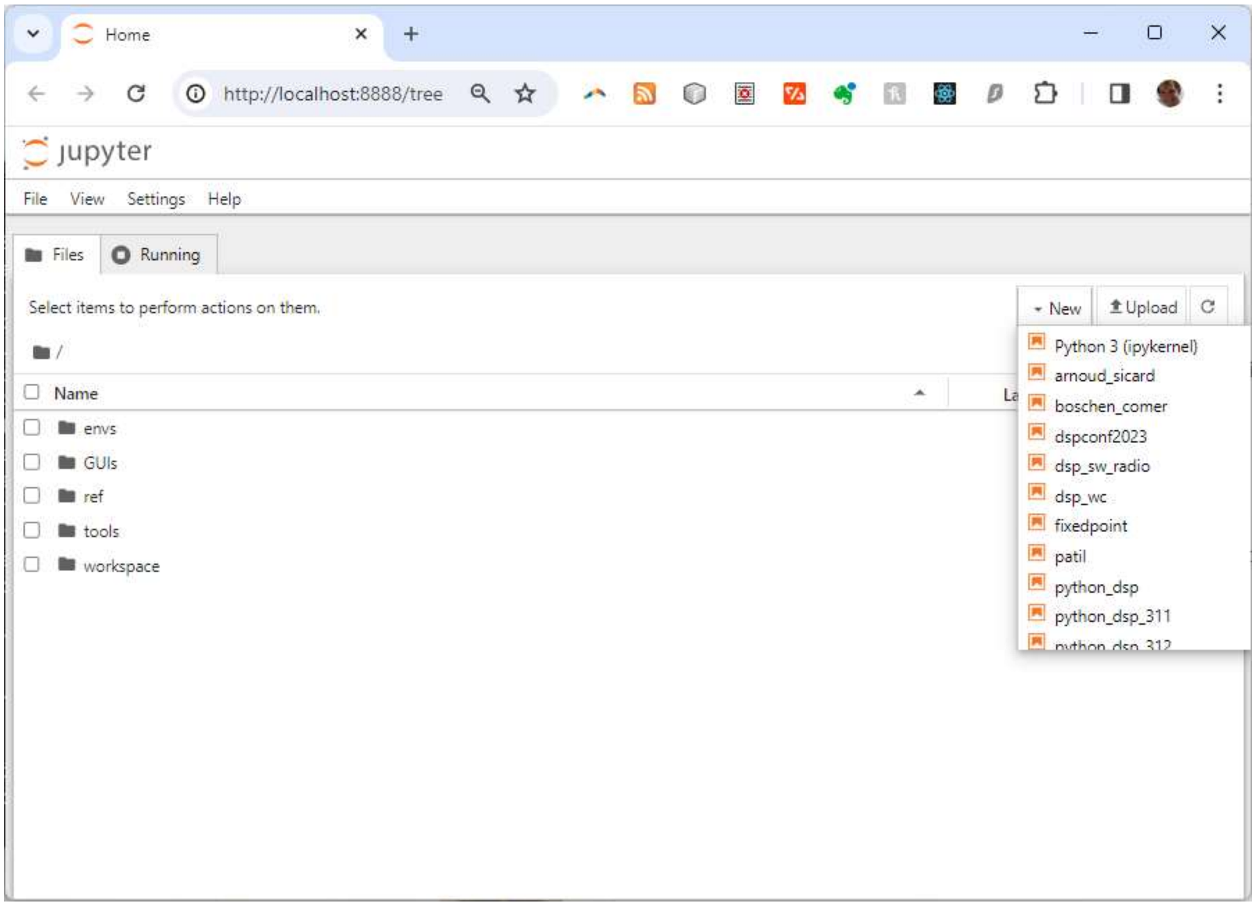


Figure 11: Jupyter Notebook main file window

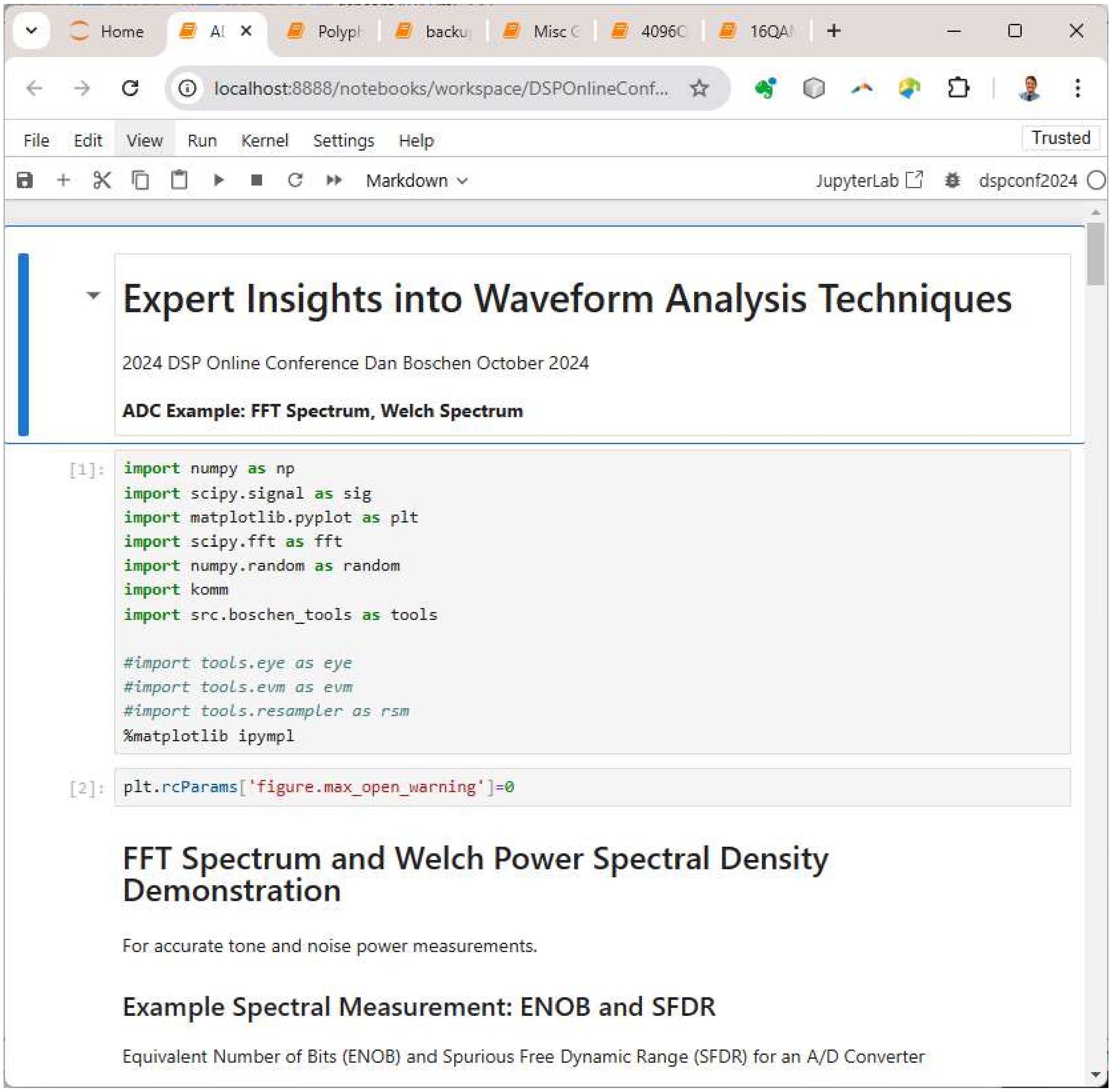
STEP 7: Running the Notebook

Launch Jupyter Notebooks if not already running as explained at the end of “STEP 6: Confirm Installation” and click on the workspace folder and navigate to the files distributed with the workshop (that should have been placed below workspace) Click on one of the notebooks for the workshop <notebook name>.pynb to open it. If 'Kernel Error' displays in red across the top bar, or if “dspconf2024” is not displayed in the upper right and corner as depicted in Figure 12; first associate the linked environment to the Notebook under menu item Kernel-Change Kernel which will bring up the Select Kernel window. From here, select the environment used for the course as set up in “STEP 5: Accessing Environments from Jupyter Notebook”. If you don’t see the Kernel listed (with the environment name if following my suggestion), follow the previous instructions in “STEP 5: Accessing Environments from Jupyter Notebook” for adding the environment as a selection item in the Notebook.

Note: Clicking on the environment name listed in the upper right hand corner of the notebook will bring up the Select Kernel window as well from which dspconf2024 can be selected.

If “Not Trusted” appears above the environment name in the upper right hand corner, click on that to select “Trust”.

Either run the entire notebook by selecting the rerun menu button (⏩), (and select Restart when prompted to restart the Kernel) or click on the top cell in the notebook and press the run button (▶) in the top menu bar to execute each cell (after pressing run the cell will execute and position will advance to the next cell). For larger notebooks I recommend this latter approach as otherwise the interactive plots may not properly show up, and may not be as clear if a particular cell is holding things up. Importantly confirm that the final plot displays completely similar to the graphic shown below (Note, to see the handy Table of Contents, this can be enabled from the menu View – Table of Contents).


Home
localhost:8888/notebooks/workspace/DSPOnlineConf...
File Edit View Run Kernel Settings Help
Trusted
Markdown
JupyterLab
dspconf2024

Expert Insights into Waveform Analysis Techniques

2024 DSP Online Conference Dan Boschen October 2024

ADC Example: FFT Spectrum, Welch Spectrum

```
[1]: import numpy as np
     import scipy.signal as sig
     import matplotlib.pyplot as plt
     import scipy.fft as fft
     import numpy.random as random
     import komm
     import src.boschen_tools as tools

     #import tools.eye as eye
     #import tools.evm as evm
     #import tools.resampler as rsm
     %matplotlib ipympl
```

```
[2]: plt.rcParams['figure.max_open_warning']=0
```

FFT Spectrum and Welch Power Spectral Density Demonstration

For accurate tone and noise power measurements.

Example Spectral Measurement: ENOB and SFDR

Equivalent Number of Bits (ENOB) and Spurious Free Dynamic Range (SFDR) for an A/D Converter

Figure 12: Example notebook result

Once complete, close the notebook by selecting "File – Close and Shut Down Notebook" from the top-level menu in the open Notebook to quit the Notebook that was open (and press OK), and then from the main Jupyter page select File – Shut Down to close the Jupyter Notebook server and then the browser window can be closed normally. The console window may also be closed at this time. (I often simply close the browser and close the Anaconda Prompt without consequence that I am aware of, but the above would be the “graceful” way to close out).

This completes the installation and test of Anaconda sufficient for the workshop.

Appendix A: Interoperating with an existing Python installation

If you have Python installed and need to continue using it, this can be done using Conda Environments to avoid any conflicts. I also highly recommend working in an environment to test out major updates rather than updating the base environment except for individual package updates. Please read this entire section before proceeding to first understand the possible options to doing this. This section details how to duplicate the existing environment, while "**Error! Reference source not found.**" details creating a new environment with a minimum installation needed for the workshop.

If Miniconda or Anaconda is not already installed: Create a file that will list the current libraries used (and version). From the command prompt in a terminal window enter:

```
pip freeze > legacy_env.txt
```

Then uninstall everything related to the current Python installation (except your own source code files) so that we can run everything without conflict within the conda environment management system.

Proceed with the instructions from the beginning of this document to install Miniconda, and then continue in this Appendix A at the header “Creating a new environment from Miniconda”.

If Anaconda is already installed:

If we want to replicate the existing installation in a new environment, this can be done from the Anaconda Prompt in Windows (or terminal in macOS or Linux) by entering the following Conda command (this can also be utilized as a roll-back option). This is doing the same thing as the pip freeze command given above:

```
conda list --export > legacy_env.txt
```

Note: If exporting and importing environments across different machines or operating systems, instead follow the procedure outlined in Appendix E: Exporting / Restoring Environments.

Creating a new environment from Miniconda:

Once Miniconda is installed, all of the following commands listed are done from the Anaconda Prompt console. Create a virtual environment for the legacy Python as follows: (please be patient as this may take a while)

```
conda update -n base conda
conda create -n <environment name> --file legacy_env.txt
```

Where <environment name> is replaced with your desired name for the environment. You may need to add the full path to the file name legacy_env.txt or move the file to the directory location where the conda command is being entered.

The above will install all the packages that were previously installed. Optionally you can just install a bare

Python version and then from that new environment install the packages needed. For example, entering the following installs Python version 3.5 in an environment names py35:

```
conda create -n py35 python=3.5
```

Once an environment is created, you can activate the new environment using:

```
conda activate py35
```

(Replacing py35 with whatever environment name is used.) To deactivate type the following:

```
conda deactivate
```

To see a list of environments, use:

```
conda env list
```

The environment name that is currently active will be indicated in the prompt itself from the console.

To update Miniconda in a new environment:

Create the new environment as detailed above with any environment name of choice (such as py38 below)

```
conda update conda
conda create -n py38
```

Activate the environment

```
conda activate py38
```

Install the latest version of minconda

```
conda install miniconda
```

Or, alternatively install only the minimum packages needed for the workshop by following the instructions to create an environment for the course as detailed further in STEP 4: Install the Python Environment.

You may get a PackagesNotFoundError if any of the packages from legacy_env.txt are not located within the channels loaded for conda. The available channels for a package can be found by going to https://anaconda.org and type the package name in the “Search Anaconda Cloud” search bar at the top of the page. From this you can locate available channels for the missing packages. Add the channels to your conda channel list by entering from the Anaconda Prompt:

```
conda config --append channels <channel>
```

Which adds the channel to the bottom of the list of accessed channels. With this added, reenter the command that resulted in the error. Optionally you can install a package from a channel directly if using the conda install command:

```
conda install -c <channel> <package name>
```

If duplicating an existing environment using the process with the legacy_env.txt file described above, and the packages listed as not being found are not actually needed / used by you, then you can optionally make a copy of legacy_env.txt with those lines removed, and repeat

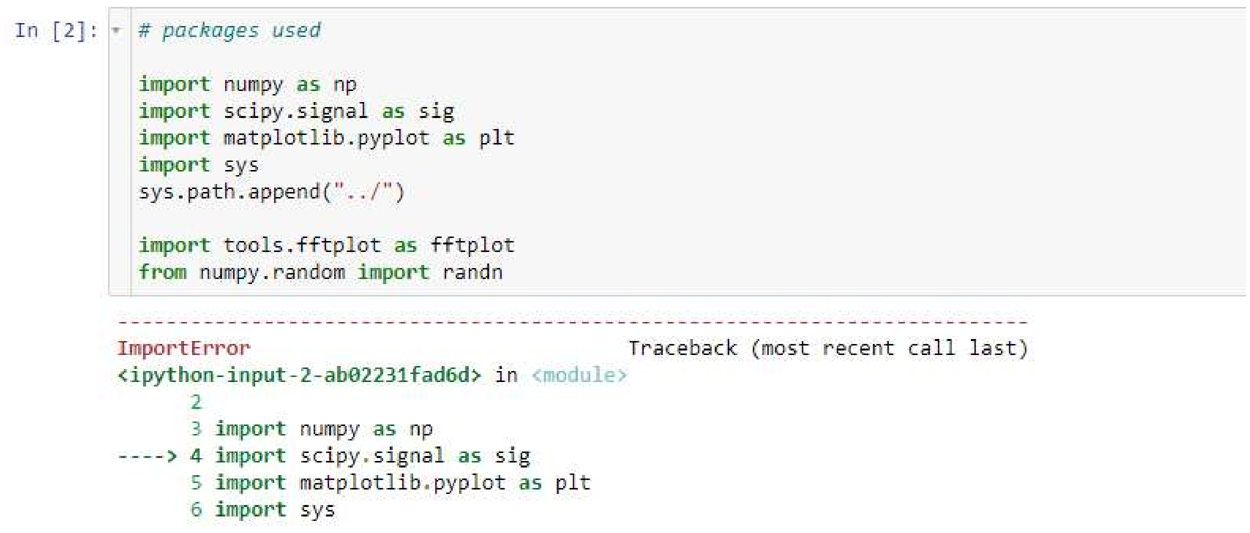
```
conda create -n <environment name> --file legacy_env_copy.txt
```

Appendix B: Troubleshooting

The following lists possible issues and known solutions.

DLL Load failed related to scipy.signal

This should only occur if you had a previous version of Python installed, or may occur after installing the control package that is used in the workshop. If when running the notebook files, you encounter an import error after running the import block, starting as captured below and ending with "ImportError: DLL Load failed: The specified procedure could not be found":



```
In [2]: # packages used

        import numpy as np
        import scipy.signal as sig
        import matplotlib.pyplot as plt
        import sys
        sys.path.append("../")

        import tools.fftplot as fftplot
        from numpy.random import randn

---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
<ipython-input-2-ab02231fad6d> in <module>
      2
      3 import numpy as np
----> 4 import scipy.signal as sig
      5 import matplotlib.pyplot as plt
      6 import sys
```

This solution worked for me: https://stackoverflow.com/questions/55201924/scikit-learn-dll-load-failed-in-anaconda

Specifically disable any of the following listed DLL's that are in the Windows/system32 directory by changing the suffix from .dll to .old as they are conflicting with the Anaconda installation:

mkl_core.dll, mkl_def.dll, mkl_intel_thread.dll, libiomp5md.dll, libmmd.dll

Unhandled Exception in event loop

If the following error occurs in Ipython through the Ananconda Prompt on a Windows machine:

```
Unhandled exception in event loop:
  File "C:\Users\bosch\anaconda3\lib\asyncio\proactor_events.py", line
768, in _loop_self_reading
    f.result()  # may raise
  File "C:\Users\bosch\anaconda3\lib\asyncio\windows_events.py", line
808, in _poll
    value = callback(transferred, key, ov)
  File "C:\Users\bosch\anaconda3\lib\asyncio\windows_events.py", line
457, in finish_recv
    raise ConnectionResetError(*exc.args)

Exception [WinError 995] The I/O operation has been aborted because of
either a thread exit or an application request
Press ENTER to continue...
```

The solution according to https://github.com/ipython/ipython/issues/12049 is to downgrade the prompt toolkit to 2.0.10 using conda by typing the following from the Anaconda Prompt from the activated environment:

```
conda install -c esrf-bcu prompt_toolkit=2.0.10
```

(if package not found, search on anaconda.org for alternate channel that has 2.0.10 version and replace with esrf-bcu above)

DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future.

If this warning appears when running cells in Jupyter Notebook, it is just a warning so can be ignored or suppressed. Updating the ipython kernel (ipykernel) resolved this for me by typing the following in the Anaconda Prompt:

```
conda update ipykernel
```

Alternatively, to suppress deprecation warnings altogether, execute the following code lines in the Jupyter notebook:

```
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

Firewall Issues Loading packages

If you get a connection failed error such as the message shown below when attempting to install a package, this may be due to firewall issue.

```
CondaHTTPError: HTTP 000 CONNECTION FAILED for url
<https://conda.anaconda.org/conda-forge/win-64/current_repodata.json>
```

Try again in case there was a bad http connection, and otherwise what follows are two options, one to implement a temporary work around and the other on how to add the firewall proxy servers to the conda configuration.

Option 1: **Temporary workaround.** This option disables SSL certificate verification and allows traffic to continue. (You are advised NOT to do this on work computers without the expressed permission of your corporate IT department).

Open the Anaconda Prompt and from that terminal enter:

```
conda config
```

This will create a file labeled .condarc in the top level data directory (where you placed your workspace directory). Open this file with a standard text editor; it should only contain two brackets: {}

Delete the two brackets and replace with the following as a single line and save the file:

```
ssl_verify: False
```

After saving the updated .condarc file, return to the Anaconda Prompt and install the package that was attempted (for example here is shown installing numpy):

```
conda install seaborn=0.13.2
```

Assuming the installation proceeded successfully (following the prompts to replace packages as needed), then go back to the .condarc file with a text editor and change ssl_verify to True to reenable SSL certificate verification:

```
ssl_verify: True
```

Option 2: Add firewall proxy servers. This option will add the proxy server for the firewall to the conda configuration, assuming the server addresses are known (request this from your corporate IT department).

Follow the instructions above with Option 1 to create a .condarc file, open that file with a text editor and replace the brackets "{}" with the following using your actual proxy server address in place of myproxy:

```
proxy_servers:
    http: http://myproxy
    https: https://myproxy
```

CondaSSLError: OpenSSL appears to be unavailable on this machine. OpenSSL is required to download and install packages.

This appears to be resolved as of 6/26/2024 but can occur if there is any interruption during the conda installation or during a conda update. For latest status see: https://github.com/conda/conda/issues/11795.

Solution:

Copy existing dlls from \bin to \DLLs

Copy libcrypto-1_1-x64.dll and libssl-1_1-x64.dll

For Miniconda Installations:

from C:\Users\username\miniconda3\Library\bin

to C:\Users\username\miniconda3\DLLs

For Anaconda Installations:

from C:\Users\username\anaconda3\Library\bin

to C:\Users\username\anaconda3\DLLs

Solving environment: failed with initial frozen solve. Retrying with flexible solve.

If you see this message after entering a conda install command through the Anaconda prompt, AND the installation appears to be taking an unusually long time to proceed (several minutes), this may be due to an outdated conda installation (the conda solver was admittedly quite slow but the solver has since migrated to libmamba as of conda 23.10.0 which is much faster.

The conda version can be seen by typing the following in the anaconda prompt from any environment:

```
conda -V
```

Updating conda to the latest version may resolve this issue. To update conda, be sure you are in the base environment (enter “conda deactivate” if another environment is activated, and then type the following from the Anaconda prompt:

```
conda update conda
```

Appendix C: File Sharing

This section is not specific to Miniconda but contains debugging details depending no mechanism used to distribute the workshop files.

Unlocking all files recursively on macOS

The files once copied over may need to be unlocked. To do this efficiently, open the terminal and enter

```
    chflags -R nouchg /usr/bin/workspace
```

Where /usr/bin/workspace is replaced with the actual top-level directory of all files to unlock.

Appendix D: Installing Python without Miniconda

The manual installation procedure for all Python libraries used for the workshop without Miniconda follows.

Step 1: Install Microsoft Visual Studio 2022 (or later) If using a Windows OS. (see STEP 2: Install the Microsoft C++ Build Tools (for Windows Users Only)

Step 2: Download and install Python 3.12.2 from https://www.python.org/downloads/ (I have confirmed compatibility with this release and the other libraries installed, so choose this release specifically even if it isn't the latest available). Scroll down to "Looking for a specific release?" and select version 3.12.2. Then scroll down and click on the installer appropriate for your OS. If Windows, select either "Windows installer (32-bit)" or "Windows installer (64-bit)". If unsure if your system is 32 bit or 64 bit, on Windows open File Explorer and then right click on "This PC" and select "Properties".

Run the downloaded executable to start the installer. Check off "Add Python 3.x to Path" ("Install launcher for all users…" should already be checked), and then click on "Install Now" to proceed.

Step 3: Install Libraries

Open the command prompt and type the following:

```
    pip install --upgrade pip
```

It's possible when upgrading pip for it to uninstall and then error before the updated installation is completed. If this happens, pip can be restored by typing from the console:

```
    python -m ensurepip
```

Continue package installations by typing the following in the command prompt console:

```
    pip install wheel
    pip install –upgrade setuptools
    pip install scipy==1.12.0
    pip install matplotlib==3.8.3
    pip install notebook==7.1.0
    pip install ipympl==0.9.3
    pip install PyQtWebEngine==5.15.7
    pip install allantools==2024.6
    pip install komm==0.9.1
    pip install fpbinary==1.5.8
```

Appendix E: Exporting / Restoring Environments

When exporting and restoring environments using conda, it is recommended to use a YAML file as it is more robust for sharing environments across platforms and operating systems. A YAML file is a human-readable serialization language similar to XML and JSON. Another option not detailed here is a Spec List text file format which results in a simpler list of packages and versions, but is only recommended for archiving and restoring environments on one's own machine.

To get a list of all current environments used, type the following command from the Anaconda Prompt:

```
conda env list
```

For each environment that you wish to restore, first activate the environment from the Anaconda Prompt by opening the Anaconda Prompt and typing the following, replacing <env-name> with the actual environment name:

```
activate <env-name>
```

Then follow the instructions below for either using the YAML file to restore the environment. I recommend using an environment for each project, and with that approach, I recommend keeping the requirements file that is generated in the top-level folder of the project. This is done easily by navigating to the specific project directory and entering the subsequent commands from there.

Export and restore using YAML

To share an environment across platforms and operations systems enter the following from the workspace project directory to create an environment.yml file, in the Anaconda Prompt with the environment activated (using any name desired in place of <*env-name*> below; I recommend matching the environment name to reduce confusion):

```
conda env export --no-builds > <env-name.yml>
```

To recreate the environment from the environment.yml file, enter the following from the base environment:

```
conda env create -f <env-name.yml>
```

Appendix F: Changing Default Directory for Jupyter Notebooks

The following details how to change the default start-up directory for Jupyter Notebooks. Note that Jupyter must be installed in an environment for it to run, but running Jupyter from any environment will bring up the same instance; so, the following procedure only needs to be done once from any environment. For this same reason, the environments need to be linked to Jupyter Notebooks as detailed in STEP 5: Accessing Environments from Jupyter Notebook.

To change the default directory for Jupyter Notebooks, from the Anaconda Prompt within any environment where jupyter in installed type:

```
jupyter notebook --generate-config
```

This will create a jupyter_notebook_config.py file in the ~home/.jupyter directory. Locate this file and open it with your favorite text editor and locate the following line to change

```
# c.NotebookApp.notebook_dir = ''
```

To:

```
c.Notebook.App.notebook_dir='<path to folder>'
```

Where '*path to folder*' is the full directory path to the desired folder to be the default directory when Jupyter Notebook opens. Be sure to use only forward slashes in directory separations such as 'C:/user/workspace' or double backslashes such as 'C:\\user\\workspace' as the single backslash is interpreted as an escape character. Note the removal of the comment character '#' in this edit.

Appendix G: Updating Miniconda

The following outlines the procedure to update the base environment for Miniconda. It is highly recommended to not update the base environment for a major Python release (for example Python 3.11 to Python 3.12), but rather create a new environment for that release. Updating a major release will likely result in significant conflict resolution in package dependencies and could break the base installation.

If there is an issue with the base installation, attempting to update the base installation by entering the following command would be the first step prior to the complete uninstall outlined in Appendix H: Uninstalling Miniconda.

```
conda update conda
```

To check your current Conda version, type:

```
conda --version
```

Appendix H: Uninstalling Miniconda

Note that the following instructions apply to uninstalling Anaconda as well as Miniconda.

At some point you may corrupt your base Miniconda installation such that the recovery/update outlined in "Appendix G: Updating Miniconda" may either not work at all or take hours to resolve conflicts. In this case it is much easier and quicker to completely uninstall Miniconda and reinstall it.

Step 1: If **requirement files** were not previously created or may not be current, save a requirement file for each environment (for any environments you want to restore) as detailed in “Appendix E: Exporting / Restoring Environments”

Step 2: Uninstall Miniconda using the add/remove programs feature of your operating system (for windows enter “add or remove programs” in the “Type here to search” prompt in the task bar).

Important! Delete the following directories that may exist in your ~home directory:

.anaconda, .conda, .continuum, .ipynb_checkpoints, .ipython, .jupyter, .matplotlib, .spyder-py3,

Delete the following files if they exist in your ~home directory:

.condarc, .python_history

(for Windows, confirm in the ~home/AppData/Roaming/ directory there is no .anaconda folder, or if it exists, delete it.

Appendix I: Removing Environments

To delete an existing environment, enter the following command (replacing 'env-name' with actual environment name, to get a list of all environments, type 'conda env list'):

```
conda remove --name <env-name> --all
```

MORE DSP AND PYTHON with Dan!

If you want to participate in more DSP and Python training by Dan Boschen, please check out the link below where Dan's courses are routinely offered:

https://www.dsprelated.com/courses

What others are saying…

Dan Boschen is one of those few people in the world of wireless communication who have strong backgrounds in both theory and practice. In addition, he has a rare gift for teaching that is reflected in his enjoyable lessons. In today's fast-paced economy where quickly learning a diverse set of skills is the only way forward, Dan helps through invaluable Python exercises at each step leading to a solid understanding of the ideas covered. Building from the fundamentals all the way up to the state-of-the-art in the field, this course is beneficial for everyone from the beginners to the seasoned engineers.
Qasim Chaudhari, Wireless Pi

Dan's Python course was ridiculously GREAT. I learned quite a bit about Python and found the presentations/material to be far better than any Python training I have ever seen. It's a bargain for the price. Not being an accomplished Python programmer, my background is in machine language->FORTRAN->C->C++->.NET, etc. I am also experienced in IC design, and embedded systems. I have a strong background in object-oriented language concepts. Taking this course with Dan is really interesting and has pushed me forward into Python. I can't tell you how impressed I am with Dan's presentation, knowledge and teaching skills. Dan uses a combination of pre-recorded videos, live workshops, and code examples with excellent content. My knowledge of Python, DSP & communications has been taken to the next level.
Dan being a "hardware engineer" is very impressive with what he has accomplished in software. Dan's enthusiasm is key to the learning experience!
David Comer

Dan Boschen's Course "DSP for Wireless Communications" is an excellent class for both new engineers learning the material and for senior personnel getting a refresher course. The lecture material is clear and concise and the verbal presentation is excellent. Most engineers obtain a basic understanding of the Fourier and Laplace transforms in their undergraduate courses. However, the z-transform is key to digital processing. Dan does an excellent job introducing and explaining the power of the z-transform. I highly recommend this course both to junior and senior engineers working modern communication systems
Dr. Alan Palevsky - Raytheon Engineering Fellow (retired), Keysight ADS Certified Expert

Table 1: Revision History

Date	Author	Description
9/13/2024	Dan Boschen	Document creation extracted content from installation guides from Dan's DSP and Python courses.
10/20/2024	Dan Boschen	Updated for use with the 2024 DSP Online Conference