1. Install Python
2. Install Required Packages (via Pip)
3. Add Exp Lab Matlab adapted packages
4. To properly learn how it works, use the MyMod.py script as a test

## Links

Python Homepage

<https://www.python.org/>

Matlab Python Documentation

<https://nl.mathworks.com/products/matlab/matlab-and-python.html>

Directory of packages that can be installed using PiP

<https://pypi.org/>

Information on using pip

<https://pip.pypa.io/en/stable/>

# Introduction

## What is Python?

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

## Why do we use Python?

Within the Experience Lab we have chosen Matlab as our primary coding environment. We develop our tools, and perform our core data cleaning inside of Matlab to preserve a singular coding environment, style, and method. Although this has worked in the past, many packages written by other universities have been developed using Python, being a free to use, open-source nature, allowing anyone who can run Python to use a package developed in Python. To allow our pipeline to integrate with these packages, we have explored, and implemented the Matlab Engine API for Python. Using this API, we can call Python packages inside of Matlab, and even transfer data between Matlab and Python, allowing for seamless integration of external functions inside of our Experience Lab Pipeline. This expansion will increase our future abilities of expanding the Experience Lab Pipeline, and create an environment where multiple coding environments could be used simultaneously.

# Installing Python on your Computer

## From a Tutorial

Use the following youtube tutorial to install Python (on Windows):

<https://www.youtube.com/watch?v=4Rx_JRkwAjY>

**Notes:**

* The tutorial installs python on a custom directory, this is not mandatory, I (Wilco) have Python installed in the default directory.
* You will not be using the IDLE for use in Matlab, altering the IDLE visuals is not required
* You can install multiple versions of Python, this should not break the Matlab Link
* Please follow the final tests from the video, which show how you can check if Python is properly working
* At 7:00 into the tutorial, it is explained how you can see if PiP (the package manager of python) is working, you need this later on, please make sure you check this as well
* Note down where you installed python, to more easily following the coming steps. My (wilco’s) location is: C:\Users\wilco\AppData\Local\Programs\Python\Python37\

# Installing Required Packages / EdaExplorer (Manually, and Using PiP)

A big strength of Python is in the availability of packages. There is an immense amount of official and home-made python packages and libraries available to use. These could be fundamental packages, such as numpy, used in many different packages for array computing, or end use packages such as the EdaExplorer, which are used as an end-package, utilizing those fundamental packages for their calculations.

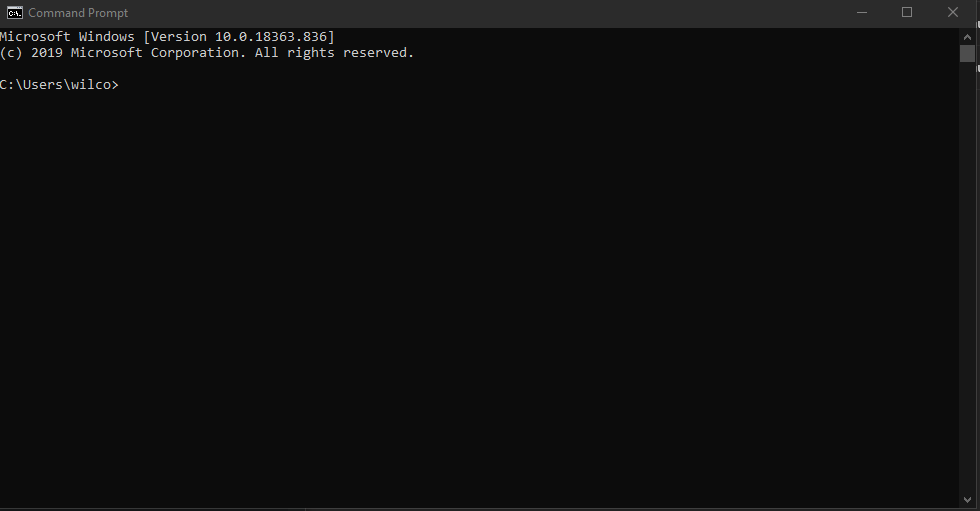
In this section I will go over the installation of the Experience Lab adapted version of EdaExplorer, one of the the packages used by the Experience Lab for detecting artifacts in EDA data. To do this, we will use both the Manual (copy Paste) as well as the automated method (using PiP).

First make sure you download the EdaExplorer folder from teams, this folder contains the full package, including the files used to run the package from Matlab.

## Installation using PiP

After downloading the EdaExplorer package, you see it will contain a file called Requirements.txt, open this to view all Python Packages which are required to run EdaExplorer in Python.

Installing packages via pip is done via the Windows Command Prompt, this can be accessed by searching for CMD or by searching for Command Prompt.

 My Windows Command Prompt

While installing Python, you would have seen the current version of PiP, you can always find this by typing pip –version in the windows command prompt. The latest version while I was writing this is version 20.1

To install the packages we have 3 options

1. Install the latest version

pip install PACKAGENAME

1. Install a specific version

pip install ‘PACKAGENAME’>=1.0.4'

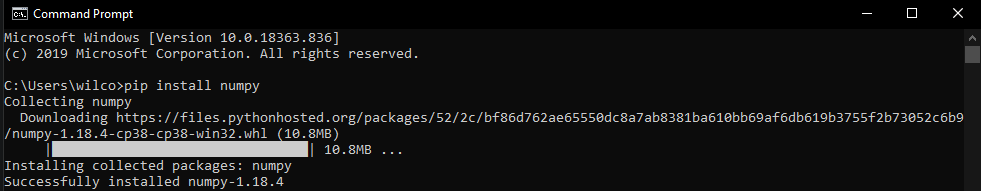
1. Install a version higher than

pip install PACKAGENAME ==1.0.4

The requirements.txt file in the EdaExplorer folder already has information on the versions required. When testing, all the latest versions of these packages working with EdaExplorer. To install these packages, type:

Pip install

Followed by the line name of the package. This will give you something like what you see below. Do this for all packages in the requirements file to install all packages.



Now you installed new packages in Python, using the automated package manager.

## Manually adding EdaExplorer

Since our addition of EdaExplorer is not available in the online library used by pip, we will need to manually paste our package into the python version we are using. To do so, navigate to the folder where Python was installed, for me that is:

C:\Users\wilco\AppData\Local\Programs\Python\Python37

Inside this directory, go into: Lib -> site-packages  
Your directory should now be:

C:\Users\wilco\AppData\Local\Programs\Python\Python37\Lib\site-packages

Copy the entire edaexplorer folder (not only the content, but the entire folder) into this directory. This allows python to recognize the package.

This is all we need to do to install the edaexplorer package inside of Python.

**Notes**:

* If you install a new version of Python, make sure you install the pip installed packages, as well as the edaexplorer folder, in this new version
* There is probably a new version of pip available, you can install it, but do not have to do this
* Many packages have a newever version than the one specified in the requirements.txt file, we tested, and assured that all versions available on 15/05/2020 work with the edaexplorer package.

# Adding Python to Matlab

Now that Python is installed, and we have added all packages we need, we can add our current version of python to Matlab. To do this run the following code inside of the Matlab Command Window.

This first line assigns the specified version of python as the version Matlab should use. Change the directory to the directory where you installed python.

pyenv('Version','C:\Users\wilco\AppData\Local\Programs\Python\Python38\python.exe')

This second section checks whether the python libraries have been indexed, and adds these if necessary

if count(py.sys.path,'') == 0

insert(py.sys.path,int32(0),'');

end

If you want to see the currently indexed python directory, run the following line in the command window:

py.sys.path

Python should now be added to Matlab, and ready to be used.

**Notes:**

* If you install and migrate to a new version of Python, you still need to assign the desired directory / version in matlab via the above specified method
* Make sure you add python.exe behind the directory, as you need to specify the location of the python executable

# Calling Python from Matlab

Calling python from Matlab is usually done with the pipeline / code provided by Wilco or Marcel, however, here is still a little explanation on how to call python from Matlab.

The first step is to import the module (the package) we want to use into matlab. The lines below first specify which library should be loaded, then reload the current libraries using the specified package.

mod = py.importlib.import\_module('edaexplorer.ArtifactDetection\_matlab');

py.importlib.reload(mod);

After importing the library, we can freely call functions from the module to send and retrieve data via the functions of that script, in the case of the EdaExplorer package, that is:

artifactData = py.edaexplorer.ArtifactDetection\_matlab.GetArtifactDataFromFolder(cfg.datafolder,cfg.artifactName,cfg.artifactType);

This code calls the GetArtifactDataFromFolder function, and provides several configurations, such as the folder, name of the data, and type. The output is then stored inside matlab in a variable called artifactData.

And those are the three lines required to run python packages inside matlab. More details and further functions can be found at:

<https://nl.mathworks.com/products/matlab/matlab-and-python.html>

# HELP

If anything fails, installing python, packages, or linking with Matlab, then feel free to send me a message ([boode.w@buas.nl](mailto:boode.w@buas.nl)). I haven’t encountered any issues while following these exact steps, but this can change per computer.