### Installation

* Install Anaconda: <https://www.anaconda.com/products/individual>
* Open the CMD.exe Prompt via the Anaconda Navigator
  + CMD.exe Prompt is the first tile in the application’s center panel
* Type this command in CMD.exe Prompt:

*conda create -n py38 python=3.8*

(“py38” is the name of the environment, you can rename this if desired)

When prompted, type “y” to proceed with the download.

**Using python 3.8 is critical. CoolProp will not work in 3.9, and other versions are untested.**

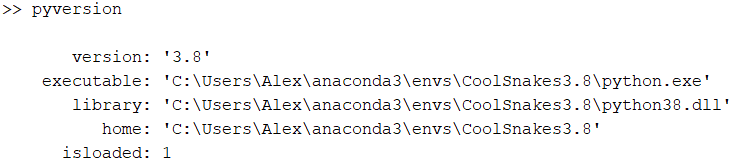
* Open Matlab
* Type this command in the Matlab command window:

*pyversion C:\Users\YourUserFolder\Anaconda3\envs\py38\python.exe*

(Find “*YourUserFolder*” and replace it with the actual folder name, i.e. the name of the windows user. If you used a different name than “*py38*”, you will have to replace that as well). For instance, my User folder is “*Alex*”, and I named the environment “*CoolSnakes3.8*”

If you get the “no executable found in that location” error, you will need to go into your files and find the directory address which links to the correct python.exe executable for your python 3.8 environment.

* To check that the right version of python is being used, type *pyversion* in Matlab. You should see something like this:



* If “version” is not 3.8, Matlab is using the wrong version.
* If “isloaded” is not 0, restart Matlab.
* Type the following in Matlab:

*[v,e] = pyversion; system([e,' -m pip install --user -U CoolProp'])*

* If this does not successfully install, troubleshooting is needed…

CoolProp will stay loaded whenever you open Matlab. If you update Matlab, you will not need to reinstall Anaconda or recreate the environment, but you will need to run the installation commands in Matlab again.

### Calling CoolProp in MATLAB

Once installed, CoolProp can be used to call a wide variety of thermochemical properties of a fluid within the library using the following command:

*py.CoolProp.CoolProp.PropsSI(“Out”, “In1”, In1Val, “In2”, In2Val, “FluidName”)*

This function can return a wide variety of outputs, and is extremely versatile for our purposes. In this case:

* *Out* is the string name of the variable representing the property to be returned
* *In1* is the string name of the first input variable
* *In1Val* is the value of the first input variable
* *In2* is the string name of the second input variable
* *In2Val* is the value of the second input variable
* *FluidName* is the name or chemical formula of the fluid considered (i.e. “C2H6”)
  + [All Fluids in CoolProp](http://www.coolprop.org/fluid_properties/PurePseudoPure.html#id401) [3]

Note: CoolProp will take inputs in either single or double quotes.

There are a handful of input values which can be employed for our purposes, most namely:

* “T”: Temperature, in Kelvins
* “P”: Pressure, in Pascals
* “Q”: Vapor quality, for a saturated liquid, in decimal value

There are dozens of possible output variables CoolProp can produce, tabulated here: [CoolProp Inputs/Outputs](http://www.coolprop.org/coolprop/HighLevelAPI.html#parameter-table) [2]. For example, this command…

*py.CoolProp.CoolProp.PropsSI(“Hmolar”, “P”, 101325, “T”, 300, “N2O”)*

…returns the molar enthalpy (note that for CoolProp enthalpy is relative to differing reference states for each fluid) of Nitrous Oxide at a pressure of 1 atmosphere and a temperature of 300 K.

**The Input table tells you what units you need to use for inputs, as well as what unit the output will be in.**

If the “Trivial” column on the [input table](http://www.coolprop.org/coolprop/HighLevelAPI.html#parameter-table) is TRUE (meaning the property you want to output isn’t dependent on the fluid’s Thermodynamic state), you do not need to input all 6 inputs, you only need 2 inputs. For instance, to find Water’s Critical Temperature, you only need the following:

*py.CoolProp.CoolProp.PropsSI(“TCRIT”,“H2O”)*

If you are using CoolProp regularly, bookmark the inputs table. It is an amazing tool which I use every single time I’m coding with CoolProp. The benefit of using CoolProp over other reference software is that you will not need to search tables to find the property you need. As long as the properties you input are possible, CoolProp will be able to output the property you are looking for. This also means that you are able to call CoolProp in a script.

One last tool that is incredibly useful is the CoolProp.m script. This script is just used to shorten the input from the long function name to:

*CoolProp(“Hmolar”, “P”, 101325, “T”, 300, “N2O”)*

The function is as follows:

function Output = CoolProp(OutputVar,InputVar1,InputVal1,InputVar2,InputVal2,Fluid)

if nargin == 2

Fluid = InputVar1;

Output = py.CoolProp.CoolProp.PropsSI(OutputVar,Fluid);

else

Output = py.CoolProp.CoolProp.PropsSI(OutputVar,InputVar1,InputVal1,InputVar2,InputVal2,Fluid);

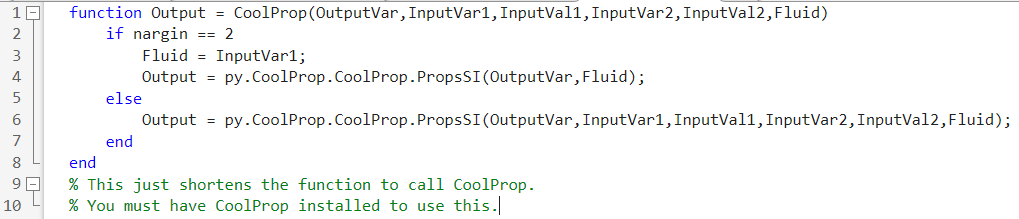
end

end

% This just shortens the function to call CoolProp.

% You must have CoolProp installed to use this.

If you copy/paste this function (rather than using a provided file of the function), you’ll probably have to reformat it to look like the screenshot below:



### References and Sources

[1] [CoolProp High Level Interface Documentation](http://www.coolprop.org/coolprop/HighLevelAPI.html)

[2] [CoolProp Inputs Table](http://www.coolprop.org/coolprop/HighLevelAPI.html#parameter-table) (This link takes you to partway down the page of source [1])

[3] [All Fluids Included in CoolProp](http://www.coolprop.org/fluid_properties/PurePseudoPure.html#id401) (You can click on the fluid name in the “List of Fluids” table to find a list of the aliases which you can use to reference that fluid. For instance, “Water” can also be referred to as “WATER”, “water”, “H2O”, “h2o”, or “R718” in the “FluidName” string.)