Python in the VE

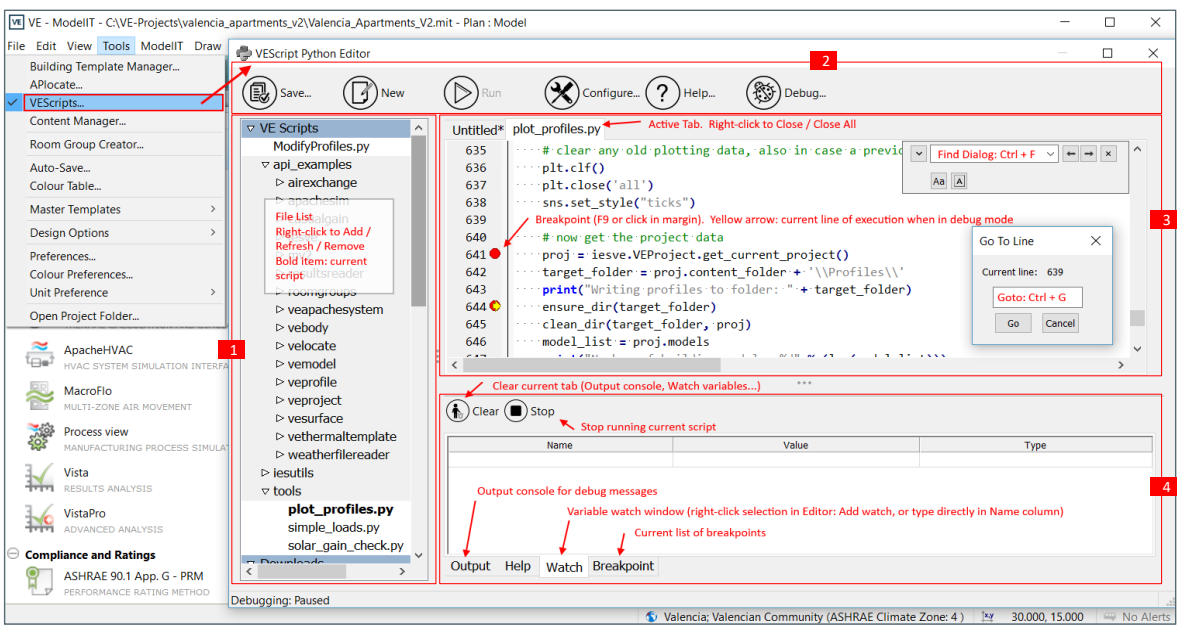
VE Scripts Editor (or IDE … *Integrated development environment*)

What? A summary of the VE Scripts Editor pointing out key features

Why? VE Scripts can be created in any editor; however they must be executed from inside the VE, so this article explains the VE Scripts Editor so you can get the most out of it when we dive in to more VE Script examples.

Scripted profiles can also be created in this editor, but must be loaded into a scripted profile package in ApPro before you can use them. Trying to 'run' a scripted profile in this editor will produce an error because the Apache module is only recognised by the Python Engine in Apache. We recommend testing the pure Python part of any scripted profile as a script before you embed it into scripted profile code; this will make testing easier

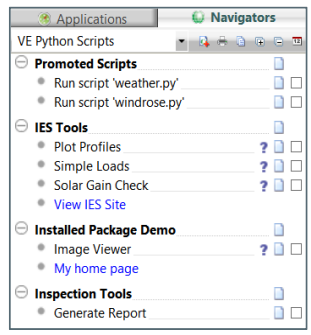
Editor: The editor is an integrated editor / debugger / console solution for developing VEScripts from inside the VE. This Python editor is modeless so activating it will not stop you from working with the rest of the VE.



1. Folder & file list:

- Right click menu to add, remove & refresh folders
- The default VE Scripts folder has an api_examples folder that contains example scripts organised by the API utilised
- The default VE Scripts folder has an ies_utils folder that contains example scripts for useful utilities like Filepicker
- The default VE Scripts folder has a tools folder that contains example tool scripts …
  The examples demonstrate using the VE Python APIs and som key Python APIs, for example creating dialogs etc
  This includes utilities for help creating free form profiles (profiles without the need to specify daily, weekly patterns) from a csv data file and creating a csv file for this purpose (so you can just insert the data and not worry about the file structure)

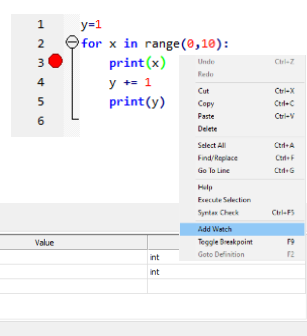
- Right click menu to a promote a script to navigator– this makes a script accessible from the VE Navigators feature ……



2. Toolbar:

- Save and new do as expected
- Configure allows you setup preferences in the editor and actions
- Run executes the VEscript
- Help accesses online VE help for the Python API including troubleshooting errors and Python documentation
- Debug gives you controls for the debugging mode:
  1. add breakpoints to your script in the edit window (left click next to the line number) ……
  2. add variables that you want to watch to the watch window (select variable, right click & add watch)
  3. on debug run debugger
  4. when the execution reaches the breakpoint it will update the values of the watched variables in the watch tab & stop
  5. press debug and continue to proceed; the code will pause each time it reaches a breakpoint

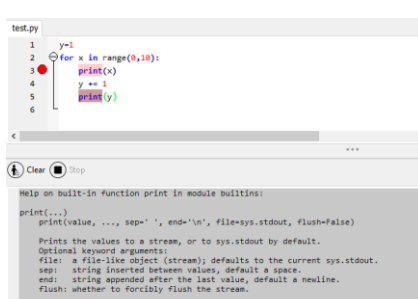
Try a small example to see how it works ……
As we increment y after the breakpoint y =x+1 as y starts at 1 & x starts at 0



6. Press debug stop to stop execution

3. Edit window:

- This is where you create and edit scripts and add breakpoints (see debugging above)
- Multiple tabs allow for more than script to be open at once
- Use Ctrl C & Ctrl V to copy and paste between scripts
- Use Ctrl F for find or find & replace
- Double click on a variable or keyword will highlight all instance of it in the script
- Ctrl G options opens a goto line number dialog
- Double click highlighting a Python keyword then right click & help will print in-line help to the output console ……



4. Output console:

- Output tab is where the 'print' output is displayed
- Double click on an *error < line ##>* and the editor will jump to that line in the code ……
- Help tab is where in-line help is displayed
- Watch tab is where watched variables and their value are shown (see debugging above)
- Breakpoint tab is where breakpoints are listed; double clicking will take the cursor to the line in the edit window
- Stop to terminate the script
- Clear to clear the output window