

Remote Debugger

http://pydev.org/manual_adv_remote_debugger.html

In PyDev you can debug a remote program (a file that is not launched from within Eclipse).

The steps to debug an external program are:

- Start the remote debugger server
- Go to the debug perspective
- Start the external program with the file 'pydevd.py' in its pythonpath
- Call `pydevd.settrace()`

Let's see a simple 'step-by-step' example on how this works:

1. Start the remote debugger server: To start the remote debugger server, you have to click the green button pointed by '1' in the image below. After doing that, it will show a message in the console (indicated by '2') to confirm that the server is listening for incoming connections.

Note: Those buttons should be present at the debug perspective and they can be enabled in other perspectives through Window > Customize perspective > Command groups availability > PyDev debug.

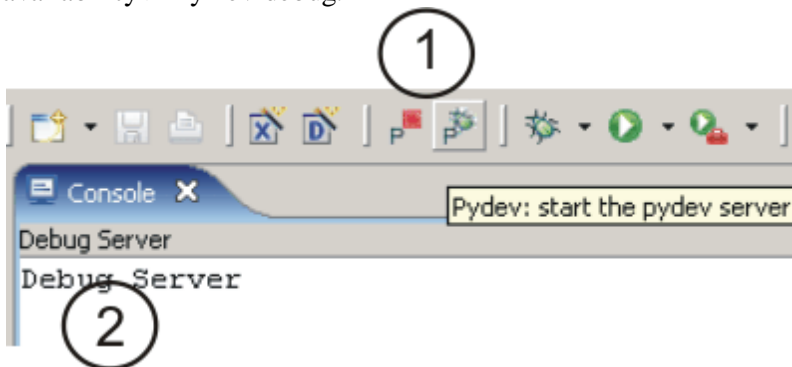


Image: Remote Debugger Server

2. Go to the debug perspective: This is needed because it has no actual 'signal' that it reached a breakpoint when doing remote debugging. So, if you already have it open, just cycle to it with **Ctrl+F8**. Otherwise, go to the menu: window > Open Perspective > Other > Debug.

Note that there should appear a process named 'Debug Server' in the debug view (see '1' in the image below).

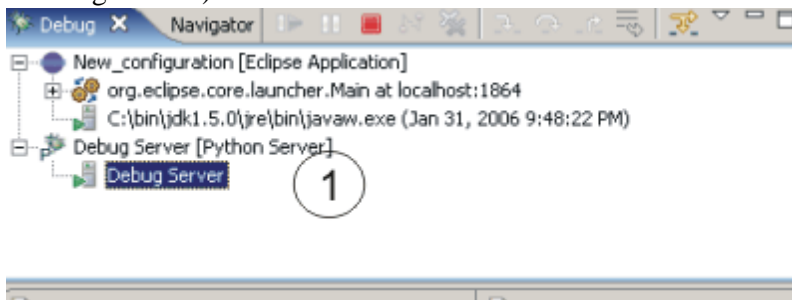


Image: Debug perspective

3. Make sure pydevd.py is in your pythonpath: This file is included in the org.python.pydev plugin. So, you'll have to add it to the pythonpath. Its exact location will depend upon the eclipse location and the plugin version, being something like:

eclipse/plugins/org.python.pydev_x.x.x/pysrc/pydevd.py

(so, the container folder must be in your pythonpath). If you choose to execute it from another machine, you need to copy all the files within that folder to the target machine in order to be able to debug it (if the target machine does not have the same paths as the client machine, the file **pydevd_file_utils.py** must be edited to properly translate the paths from one machine to the other - see comments on that file).

4. Call pydevd.settrace(): Now that the pydevd.py module is already on your pythonpath, you can use the template provided: 'pydevd' to make the call: `import pydevd;pydevd.settrace()`. When that call is reached, it will automatically suspend the execution and show the debugger.

Remote Maya Python Debugging

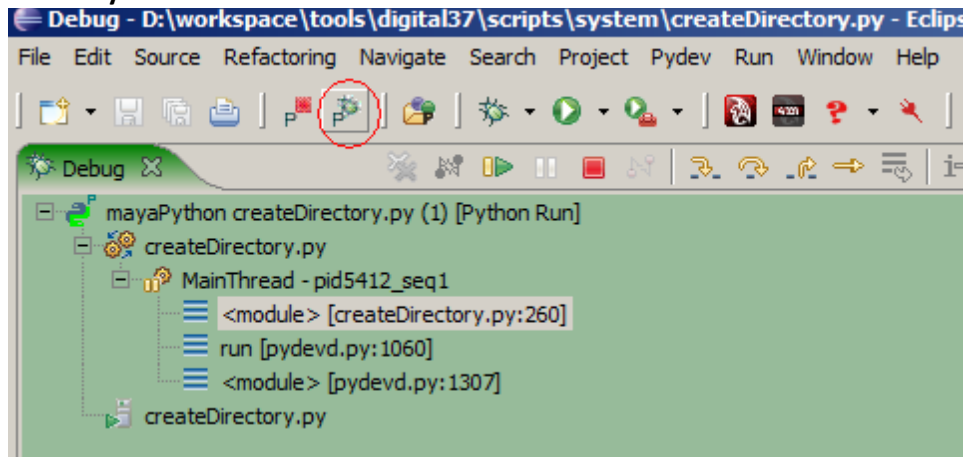
1.Setup pythonpath:

Firstly, you'll need the path to the pydevd.py module in your Maya session's pythonpaths. Follow the instructions in the link above to find out where this path is (something like: eclipse/plugins/org.python.pydev.debug_x.x.x/pysrc/).

Example(Maya.env):

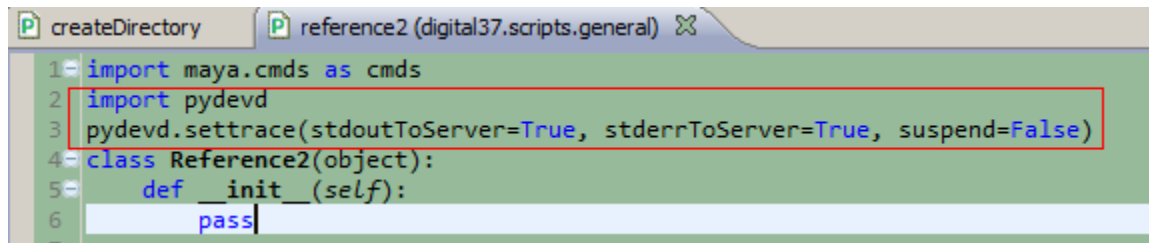
```
PYTHONPATH = ..\eclipse\..\plugins\org.python.pydev.debug_2.3.0.2011121518\pysrc
```

2.Start Pydev server:



3.Add below lines to your scripts:

```
import pydevd  
pydevd.settrace(stdoutToServer=True, stderrToServer=True, suspend=False)
```



```
1= import maya.cmds as cmds  
2= import pydevd  
3= pydevd.settrace(stdoutToServer=True, stderrToServer=True, suspend=False)  
4= class Reference2(object):  
5=     def __init__(self):  
6=         pass
```

4.Call your function in Maya script editor: