"it" Scripting Tips and Tricks

"it" is not your usual python interpreter so there are some things to know about adding scripts.

Reloading a Script

When a custom command is loaded via a .ini file you would typically put and file called it.ini in a folder alongside the python file containing your code. The it.ini file would look like this:

```
LoadExtension python acmd.py

# It's a good idea to log the presence of the extension so we know
# what we have on our RMS_SCRIPT_PATHS
LogMsg NOTICE "Registered acmd extension"
```

Then in the python file we might have something like:

```
import it

def getCatalog(name):
    ''' return a catalog by name, if not found
        then create a new one.
    '''
    model = it.app.GetModel()
    for cat in model.GetCatalogs():
        if cat.GetLabel() == name:
            return cat
    cat = it.app.NewCatalog(name)
    return cat
```

If you editing acmd.py heavily as you develop your code you might not want to keep quitting "it" to get it to reload acmd.py. We can't use python's *reload()* c ommand because, for various reasons, acmd.py is not loaded as a python module. We can fix this by not loading acmd.py directly and instead using a intermediate loading script that does import acmd.py as a module.
First modify you it.ini file to point to the intermediate file:

```
LoadExtension python loader.py

# It's a good idea to log the presence of the extension so we know
# what we have on our RMS_SCRIPT_PATHS
LogMsg NOTICE "Registered loader extension"
```

Now in this new file we set things up in python so we can load acmd.py as a module. In reload.py we have:

```
import sys
import os

#
# this file is exec'd so python's __file__ is not accurate, we want to add the current
# dir to python's import path. Fortunately "it" passes us the correct file name:
moduledir = os.path.dirname(os.path.abspath(__it_extension_file__))
sys.path.append(moduledir)

# import modules. Note we could loop over all .py files found in moduledir if
# wanted to do something clever
import acmd
it.app.Notice("imported acmd")
```

Now when you launch "it" acmd will be loaded and if you edit acmd.py, save it and then in the Console window you can type:

```
py> reload(it.extensions.acmd)
```

Note that extensions are always loaded into it.extensions to keep them from colliding with python built-ins or "it" built-ins.