# Procedures - more details

* how to add 4 floats - three ways

* diff. between user-created procs and built-in Maya procs

* MEL scrips: ground rules
The overall idea is to create a PERSISTENT collection of procs (and non-procs) in the form of a text file [with .mel extension], store it in a 'standard' place to be accessed whenever necessary - no need to enter procs/statements into the Script Editor window each time. Procs contain MEL statements which are made of flow control syntax, expressions, operators and variables. Procs also commonly make use of (call) other procs to get their work [and your work!] done.

The 'source' command sources (executes) a file containing MEL commands (procs and non-procs). In the case of sourcing a proc, all we're doing is committing that proc's definition to Maya's memory. This is NOT the same as running the proc. Running it is a separate, subsequent step.

MAYA_SCRIPT_PATH is an environment variable that specifies the 'standard' set of directories (locations) in which to place MEL scripts. The 'getenv' command retrieves the existing value (a colon-separated string) of $MAYA_SCRIPT_PATH, and 'putenv' (re)sets it to a new supplied value.

*If* a MEL script called runMe.mel (for example) is placed in a 'standard' location, *and if* it contains a global proc also named runMe(), *then* that proc will be automatically executed (run) by Maya when the proc name is specified in the Script Editor window or in another script. NO NEED TO MANUALLY SOURCE runMe.mel, that is done automagically.

In some cases, a MEL script might not have a global proc inside it with the same name as the file name. Thi is common in 'utility' collections of procs. Eg. a file called 'cinesiteUtils.mel' need not contain a global proc called cinesiteUtils(). So to make use of the various global procs in such a file, the file needs to be sourced first, by saying:
source cinesiteUtils; // cinesiteUtils.mel is assumed to be in a 'standard' location
After the file is sourced, global procs inside it can be called (used).

If a MEL script is NOT in a standard location, it needs to be sourced by specifying its full path, eg.
source "/tmp/tst.mel";
Simplying using the name of the file is not sufficient.

Once a MEL script is sourced automagically or manually, ANY global proc inside that script can be invoked on the Script Editor window or in another script.

When a proc is sourced, its definition is stored in (volatile) memory. So if a change is made to the text version on disk by editing it (as is common practice during code development), you MUST source the file again, to force MEL to update its online proc definition. So, during a typical MEL edit-run-debug cycle, if you change your source code, be sure to source, THEN re-run your program.

Say there are two MEL files A.mel and B.mel, in a 'standard' directory. If A.mel simply contains 'source B;' and if B.mel reciprocally contains 'source A;', what would happen you executed A.mel by saying 'A;'? Would Maya go into an infinite loop, alternately sourcing A.mel and B.mel? NO, since the 'source' stmt, is not a MEL command (it is a 'directive'). The way to FORCE MEL to get into an infinite loop, however, is to use the 'eval' stmt., ie. make A.mel to be 'eval "source B";' and B.mel to be 'eval "source A";'. The eval will always execute its input string, so the 'source' stmts. will run endlessly.

If you place A.mel in a standard location and another A.mel (maybe a completely different program in /tmp, to run the standard version you can simply say 'A;'. To run the alternate version, however, you need to do 'source "/tmp/A.mel";A;', ie, source and then run. What if you make a cange in /tmp/A.mel and want to rerun it? You can't just say 'source A;A;' and expect the 'source' stmt. to source the alternate version. It will in fact source the 'standard' version. For the alternate version, you need an explicit source stmt., ie. 'source "/tmp/A.mel"'.