MEL to Python Cheat Sheet

Procedures vs. Functions

MEL

```
global proc myproc( string $arg1, int $arg2 )
{
    print ($arg1 + " " + arg2 + "\n" );
}
```

Python

```
def myfunc( arg1, arg2 ):
    print arg1, arg2
```

Scripts vs. Modules

MEL

```
source "myScript.mel";
```

Python

```
import myModule
```

In MEL, procedures are often readily available to use, even if you have not explicitly sourced the MEL script that contains it. This automatic availability is only achieved if a script is on the `MAYA_SCRIPT_PATH` and it contains a global procedure with the same name as the script. Many people are not aware of this caveat and it is a source of great confusion and inconsistent results.

Python has a more explicit set of rules. Any python script with a .py extension found on the `PYTHONPATH` is considered a module and *must* be imported inside every script that utilizes it. This usually occurs at the very top of a file. Think of it as a way of telling python and anyone else who looks at your module exactly what other modules it depends on.

Module Namespaces

The primary feature that conceptually distinguishes a module from a MEL

script is that a module by default serves as its own namespace. In other words, once the module is imported, any functions contained inside it must be referenced with the module's name as a prefix (without the .py extension):

```
import myModule
myModule.myfunc()
```

The concept of namespaces should be familiar to anyone who has used referencing in Maya. In Maya, namespaces provide organization and help avoid conflicts between the contents of each referenced file. And like Maya, python provides the ability to modify the default namespace when importing.

Here's how you can change the namespace:

```
import myModule as mod
mod.myfunc()
```

or you can get rid of the namespace all together:

```
from myModule import *
myfunc()
```

Environment Variables

MEL	Python
MAYA_SCRIPT_PATH	PYTHONPATH

Vectors

MEL	Python
<<0,1,2>>	Vector(0,1,2)

tokenize

MEL

```
string $buf1[];
tokenize( "chad dombrova", $buf1 );
string $buf2[];
tokenize( "joint_01_left_leg", $buf2, "_" );
```

Python

::

buf1 = ‘chad dombrova’.split() buf2 = ‘joint_01_left_leg’.split(‘_’)