Tutorial: Generated Code

(journal this)

We'll start with some simple code and make it progressively more complicated.

```
x = 5
cmd = "y = 6"
status = execute(cmd)
help,x,y,status,cmd
print,x,y,status,cmd
```

Follow so far? Now make a new `cmd` that creates an array from 0 to 4 of integers using `indgen` and execute that command.

When you've executed the command, make sure that it worked (make sure that `execute` returned `1`).

Now we'll compose a command from different strings

```
; determine if x is 5
test_cmd = "x" + " eq " + 5
```

Why didn't that work? What can you do to fix it?
Hint 1: use `string`.
Hint 2: `test_cmd` should be `"x eq 5"` when you've fixed it
Hint 3: What will happen if you `print,"5"`? `print,'5'`?

OK, now `print,test_cmd`. Is it valid code?
(if you're not sure, see what happens if you try to `execute` it: if `print,execute(test_cmd)` returns 0, that means it's not valid code; if it returns 1, then it is valid code)
What do you need to do to make it valid?
Hint: What would happen if you just typed:
`IDL> 1`
What would you need to do to make THAT a valid command?

If you followed all that, you should now have a `test_cmd` that looks something like `"test = x eq 5"` or `"print,x eq 5"`. Execute that. Does it work?

Answer these questions in the Journal

```
test_cmd = "test = x eq y"
print,execute(test_cmd)
```

What is `test_cmd`? What will be printed? What is the value of `test`?

```
test_cmd = "test = x eq "+ string(y)
print,execute(test_cmd)
```

What is `test_cmd`? What will be printed? What is the value of `test`?

```
test_cmd = "test = x eq "+ string(x)
print,execute(test_cmd)
```

What is `test_cmd`? What will be printed? What is the value of `test`?

Is this valid code?

(do this in a txt file, not a journal)

Hint: This is a great time to copy & paste from the pdf into `gvim` and then edit the text so that it's formatted reasonably.

Hint 2: This is also a great opportunity to try out `execute`, since `execute` will return 0 for invalid code and 1 for valid code

Hint 3: You could get this all done with a for loop and a list...

```
abs(x - y)
print,x-y lt 2
x-y eq 2
print,"25" + "30"
print,'25' + '30'
print,'b' + '30'
print,'b' + 30
print,'b'+string(30)
cmd = x - y eq -1
cmd = abs(x-y) lt 2
cmd = "abs(x-y)" lt 2
cmd = "abs(x-y) lt" 2
cmd = "abs(x-y) lt 2"
cmd = "abs(x-y) lt "+"2"
cmd = "abs(x-y) lt "+string(2)"
```

Break down this line into its components (split it onto several lines): `print,execute("test = abs( (x-y) * 2 ) lt 1e-8 ?`

Then, try this line after making some changes:

```
x = 0.5
y = 0.5d
print,execute("test = abs( x-y ) lt 1e-10 ? 'true' : 'false'")
print,test

x = 0.2
y = 0.2d
print,execute("test = abs( x-y ) lt 1e-10 ? 'True' : 'False'")
print,test

x = !pi
y = !dpi
print,execute("test = abs( x-y ) lt 1e-10 ? 'True' : 'False'")
print,test

x = 3.08567758e16
y = 3.08567758d16
print,execute("test = abs( x-y ) lt 1e-10 ? 'True' : 'False'")
print,test
print,x,y,format="(E25.15,' ',E25.15)"
```

Now you should be ready to finish the `test_mks_units` code.