VisIt-tutorial-Python-scripting

From VisItusers.org

Contents

Command line interface (CLI) overview

VisIt has a command line interface that is Python based.

There are several ways to use the CLI:

1. Launch VisIt in a batch mode and run scripts
   - e.g.: `./visit -nowin -cli -s <script.py>`
2. Launch VisIt so that a visualization window is visible and interactively issue CLI commands
3. Use both the standard graphical user interface (GUI) and CLI simultaneously

Launching the CLI

We will focus on the use case where we have the graphical user interface and CLI simultaneously.

To launch the CLI from the graphical user interface:

1. Go to Controls->Launch CLI
2. This will launch a separate terminal that has the Python interface.

Note that you can also use VisIt's Command window to submit Python code to the CLI. The Command window provides a text editor with Python syntax highlighting and an *Execute* button that tells VisIt to execute the script. Finally, the Command window lets you record your GUI actions into Python code that you can use in your scripts.

A first action in the CLI

1. Open example.silo in the standard GUI if it not already open.
   - You can do this in the CLI, but it requires typing in absolute paths.
2. Type: AddPlot("Pseudocolor", "temp")
   - You will see the active plots list in the GUI update, since the CLI and GUI communicate.
3. Type: DrawPlots()
   - You should see your plot.

Tips about Python

1. Python is whitespace sensitive! This is a pain, especially when you are cut-n-pasting things.
2. Python has great constructs for control and iteration.
   - I frequently use:

```
for i in range(100):
   # use i
```

```
if (cond):
 # stmt
```

```
import sys
...
sys.exit()
```

Example Scripts

We will be using Python scripts in each of the following sections: You can get them by:

1. Cut-n-paste-ing it into your Python interpreter
2. Or copying it to a separate file and issuing: Source("/path/to/script/location/script.py")

For all of these scripts, make sure example.silo is currently open.

Setting attributes

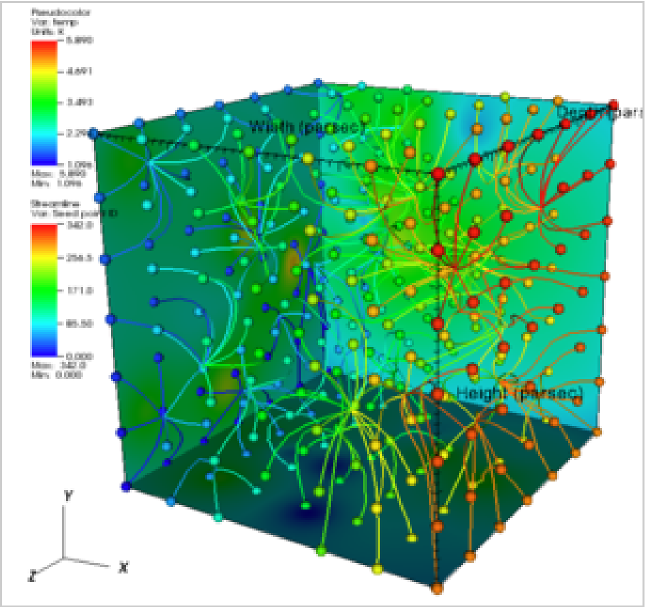
```
DeleteAllPlots()
AddPlot("Pseudocolor", "temp")
DrawPlots()
p = PseudocolorAttributes()
p.minFlag = 1
p.maxFlag = 1
p.min = 3.5
p.max = 7.5
SetPlotOptions(p)
```

Animating an isosurface

```
DeleteAllPlots()
AddPlot("Pseudocolor", "temp")
iso_atts = IsosurfaceAttributes()
iso_atts.contourMethod = iso_atts.Value
iso_atts.variable = "temp"
AddOperator("Isosurface")
DrawPlots()
for i in range(30):
   iso_atts.contourValue = (2 + 0.1*i)
   SetOperatorOptions(iso_atts)
   # For moviemaking, you'll need to save off the image
   # SaveWindow()
```

Using all of VisIt's Building Blocks

```
# clear any previous plots
DeleteAllPlots()
# Create a plot of the scalar field 'temp'
AddPlot("Pseudocolor","temp")
# Slice the volume to show only three
# external faces.
AddOperator("ThreeSlice")
tatts = ThreeSliceAttributes()
tatts.x = -10
tatts.y = -10
tatts.z = -10
SetOperatorOptions(tatts)
DrawPlots()
# Find the maximum value of the field 'temp'
Query("Max")
val = GetQueryOutputValue()
print "Max value of 'temp' = ", val
# Create a streamline plot that follows
# the gradient of 'temp'
DefineVectorExpression("g","gradient(temp)")
AddPlot("Streamline","g")
satts = StreamlineAttributes()
satts.sourceType = satts.SpecifiedBox
satts.sampleDensity0 = 7
satts.sampleDensity1 = 7
satts.sampleDensity2 = 7
satts.coloringMethod = satts.ColorBySeedPointID
SetPlotOptions(satts)
DrawPlots()
```



Streamlines

Animating the camera

See here.

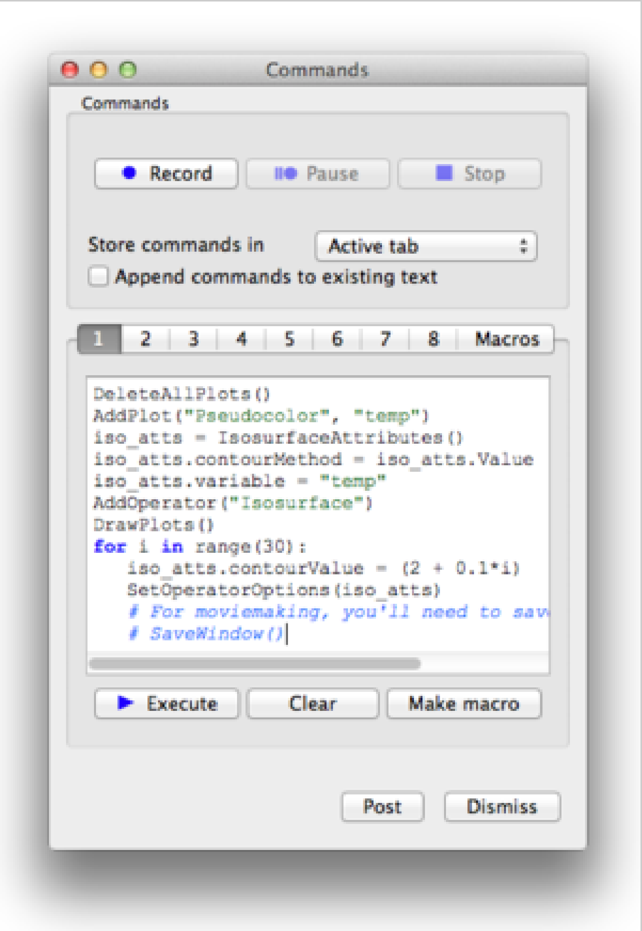
Automating data analysis

See here.

Learning the CLI

Here are some tips to learn the CLI better:

1. From within VisIt's python CLI, you can type "dir()" to see the list of all commands.
    - Sometimes, the output from "dir()" within VisIt's python CLI is a little hard to look through. So, a useful thing on Linux to get nicer list of methods is the following shell command (typed from *outside* VisIt's python CLI)...
        - `echo "dir()" | visit -cli -nowin -forceinteractivecli | tr ',' '\n' | tr -d " '" | sort`
    - Or, if you are looking for CLI functions having to do with a specific thing...
        - `echo "dir()" | visit -cli -nowin -forceinteractivecli | tr ',' '\n' | tr -d " '" | grep -i material</li>`
2. You can learn the syntax of a given method by typing "help(MethodName)"
    - (Type "help(AddPlot)")
3. You can have the GUI translate GUI actions into Python commands:
    1. Bring up Controls->Command…
    2. Press "Record"
    3. Perform GUI actions
    4. Click Stop
    5. The equivalent Python script will be placed in the Commands window.
    - Note that the scripts are very verbose and contain some unnecessary commands, which can be edited out.
4. The *WriteScript* function will create a python script that describes all of your current plots
    - For more details, see WriteScript
5. When you have a Python object, you can see all of its attributes by entering its name
    - (Type "s = SliceAttributes()", enter, followed by "s" and then enter again)



Store commands in Active tab

☐ Append commands to existing text

```
DeleteAllPlots()
AddPlot("Pseudocolor", "temp")
iso_atts = IsosurfaceAttributes()
iso_atts.contourMethod = iso_atts.Value
iso_atts.variable = "temp"
AddOperator("Isosurface")
DrawPlots()
for i in range(30):
    iso_atts.contourValue = (2 + 0.1*i)
    SetOperatorOptions(iso_atts)
    # For moviemaking, you'll need to sav
    # SaveWindow()
```

Command window

Tips for searching for help

1. Instead of working on the command line when looking for something useful in the long list of functions reported by dir(), can use a helpers functions to limit the scope of your search. From VisIt's cli, one such helper is the 'lsearch' command in the visit_utils module:

```
from visit_utils.common import lsearch
lsearch(dir(),"Material")
```

You can also start your VisIt session using the following script:

```
#!/usr/bin/python -i   # 1

from visit import *    # 2
```

```
Launch()

global_dir = dir()
def find(s):          # 3
 global global_dir
 for f in global_dir:
  if f.find(s)>-1:
   print f
#
```

Make the script executable and run it. The '-i' option passed to the python interpreter on line 1 will make sure that after finishing the script you will be taken to the Python commandline, just as if you had just started the interpreter with a single 'python'-command. Then lines 2 will start up VisIt for you, which you normally would have to do manually. Then on line 3 the script defines a special function for you, that will look for a specific string in the global listing of functions. After starting the script you could do for example:

```
>>> find('ActivePlots')
ChangeActivePlotsVar
DeleteActivePlots
HideActivePlots
SetActivePlots
```

Of course you can create variations on this find-function as complicated as you wish. Adapt the find-function or create other functions that perform more specialized searches.

Advanced features

1. You can set up your own buttons in the VisIt gui using the CLI. See here.
2. You can set up callbacks in the CLI that get called whenever events happen in VisIt. See here.
3. You can create your own custom Qt GUI that uses VisIt for plotting. See here.

Retrieved from "http://visitusers.org/index.php?title=VisIt-tutorial-Python-scripting"

Category: VisIt Tutorial

- This page was last modified 17:59, 13 December 2013.