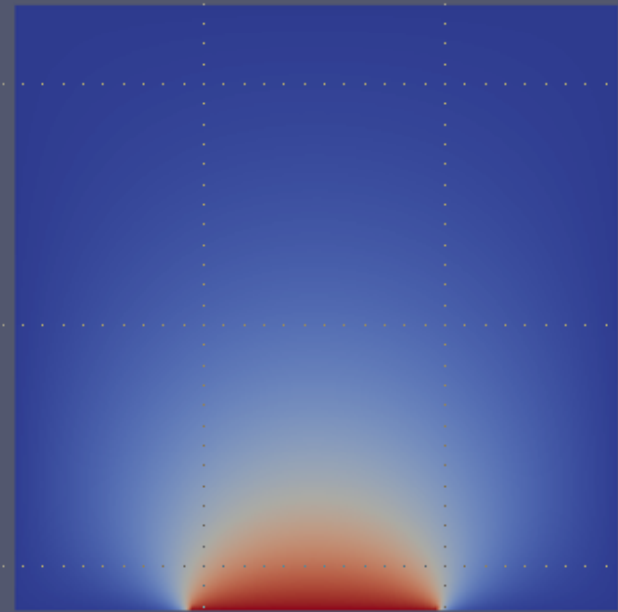
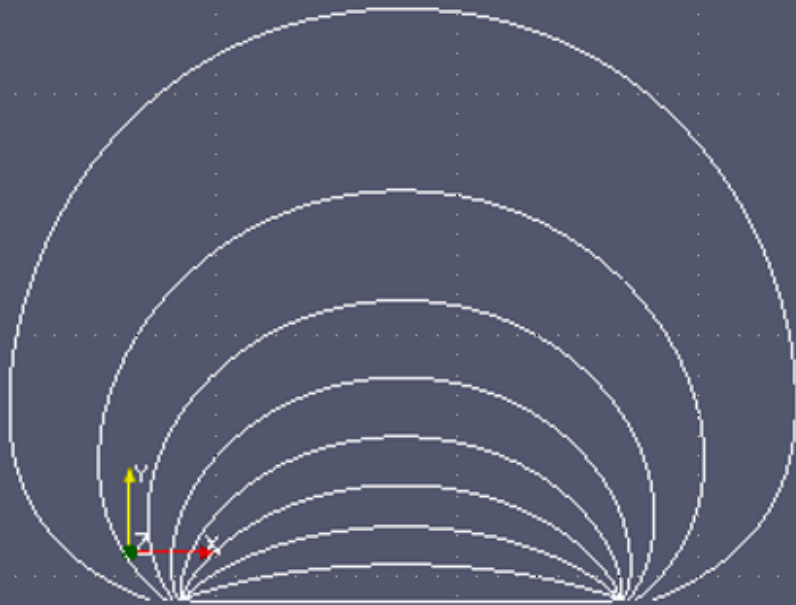




You should be here



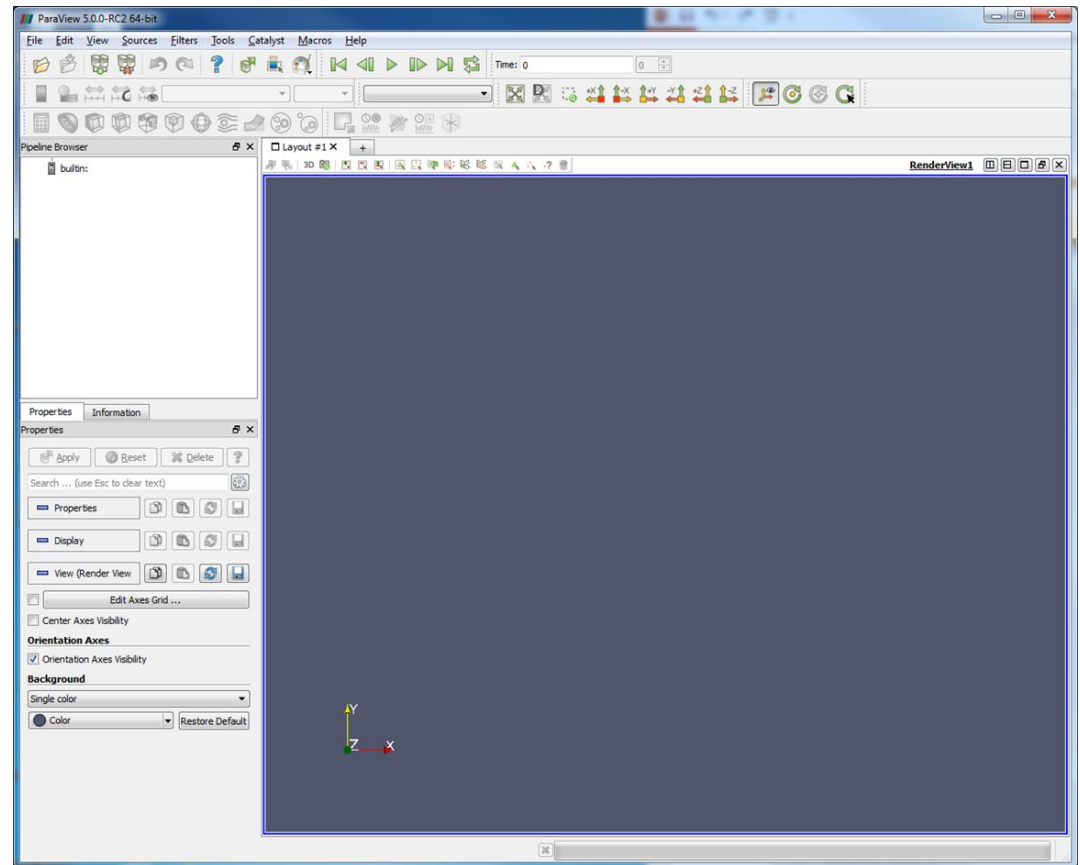
# Filters in ParaView

- Filters
- Alphabetical

# In preparation for the next section

You should be here

- Delete all objects in the Pipeline Browser
- Select an object in the Pipeline Browser
- Click the Delete button (or right click, then Delete)
- To select multiple objects press and hold the CTRL key while selecting objects



# ParaView/Python Scripting

**A short introduction to ParaView's Python Interface**



# PARAVIEW/PYTHON SCRIPTING

Rich scripting support through Python.

Available

- As part of the ParaView Client (ParaView)
- An MPI-enabled batch application (pvbatch)
- The ParaView python client (pvpython) or
- Any other Python-enabled application

Using Python, users and developers can gain access to the ParaView engine called **Server Manager**

# PARAVIEW/PYTHON SCRIPTING

## SERVER MANAGER

- Library
- Designed to make it easy to build distributed client-server applications

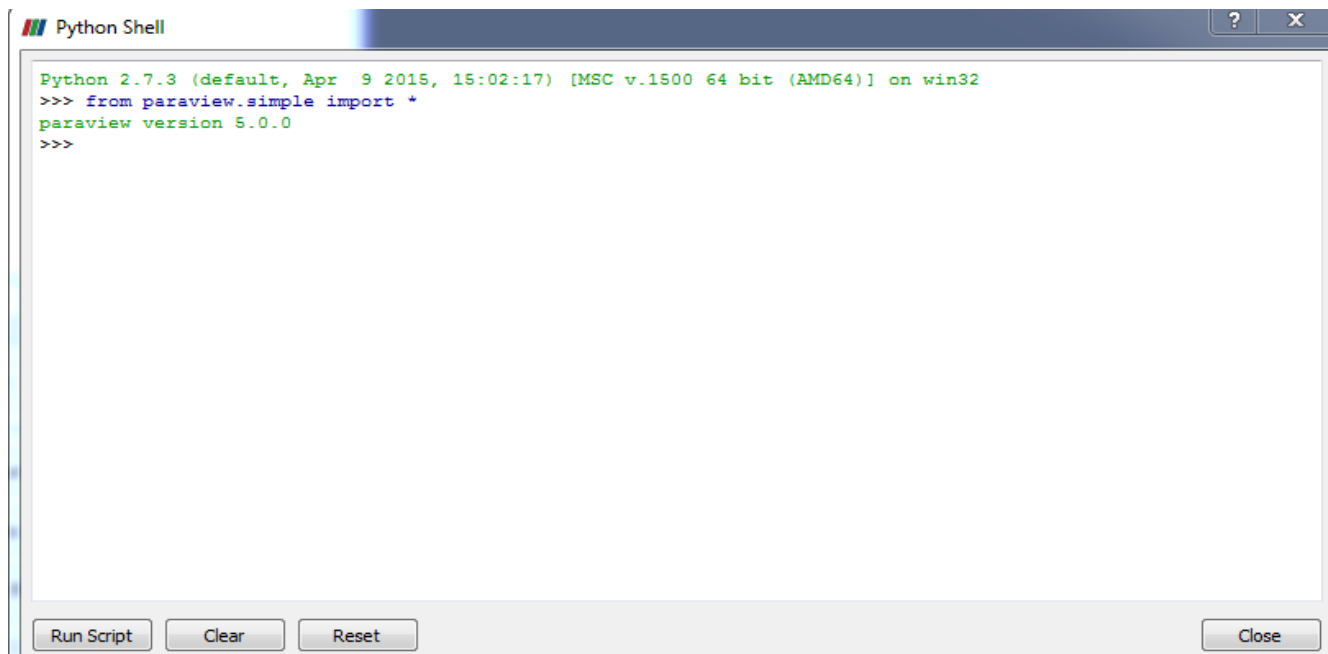
# GETTING STARTED

## PYTHON SHELL – USING PARAVIEW CLIENT

Open Python Shell:

→ Tools

→ Python Shell



```
Python 2.7.3 (default, Apr  9 2015, 15:02:17) [MSC v.1500 64 bit (AMD64)] on win32
>>> from paraview.simple import *
paraview version 5.0.0
>>>
```

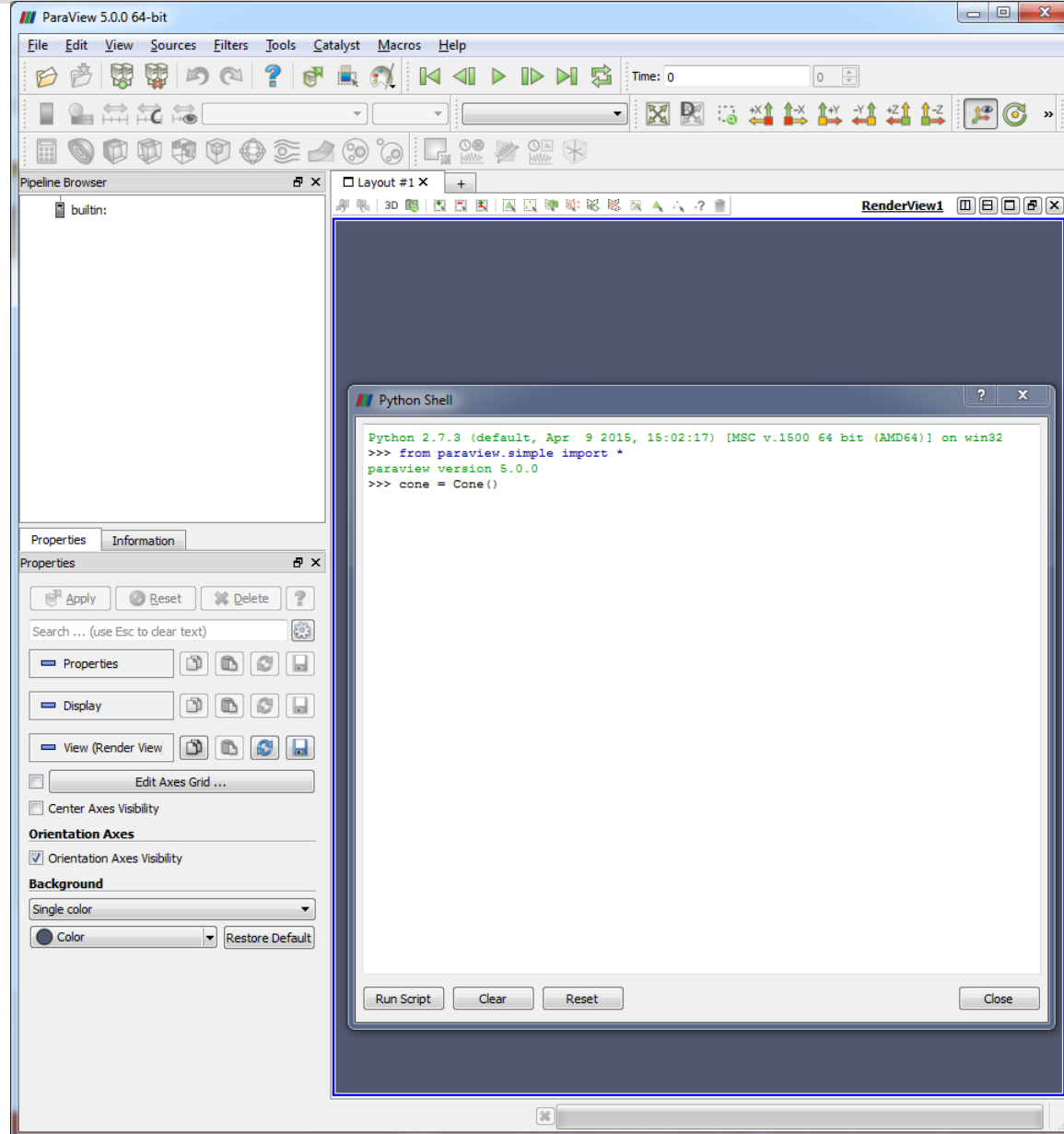
Run Script Clear Reset Close

# PARAVIEW/PYTHON SCRIPTING

## CREATING A PIPELINE

Create a Cone Object:

```
>>> cone = Cone()
```



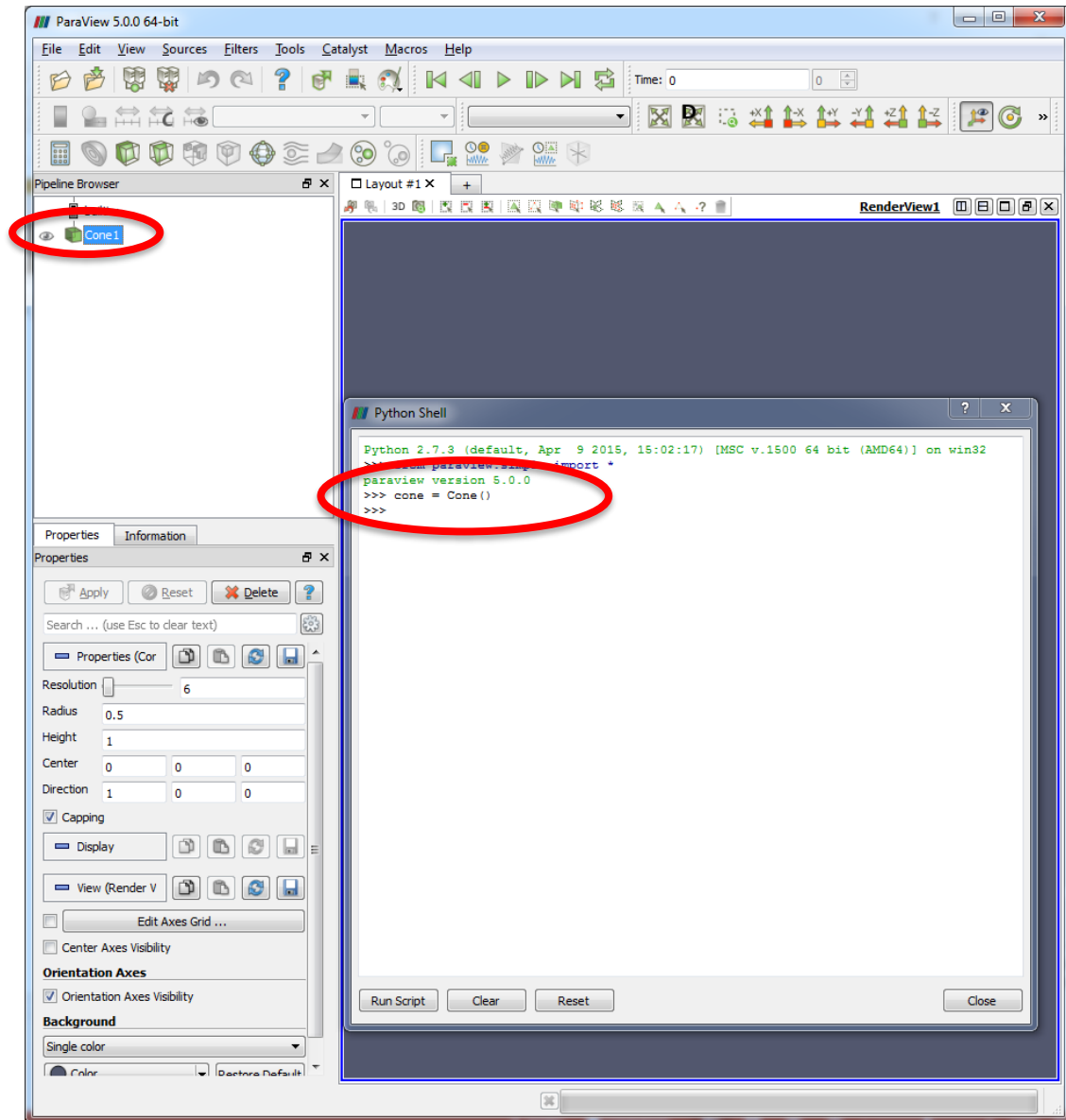


# PARAVIEW/PYTHON SCRIPTING

## CREATING A PIPELINE

Create a Cone Object:

```
>>> cone = Cone()
```



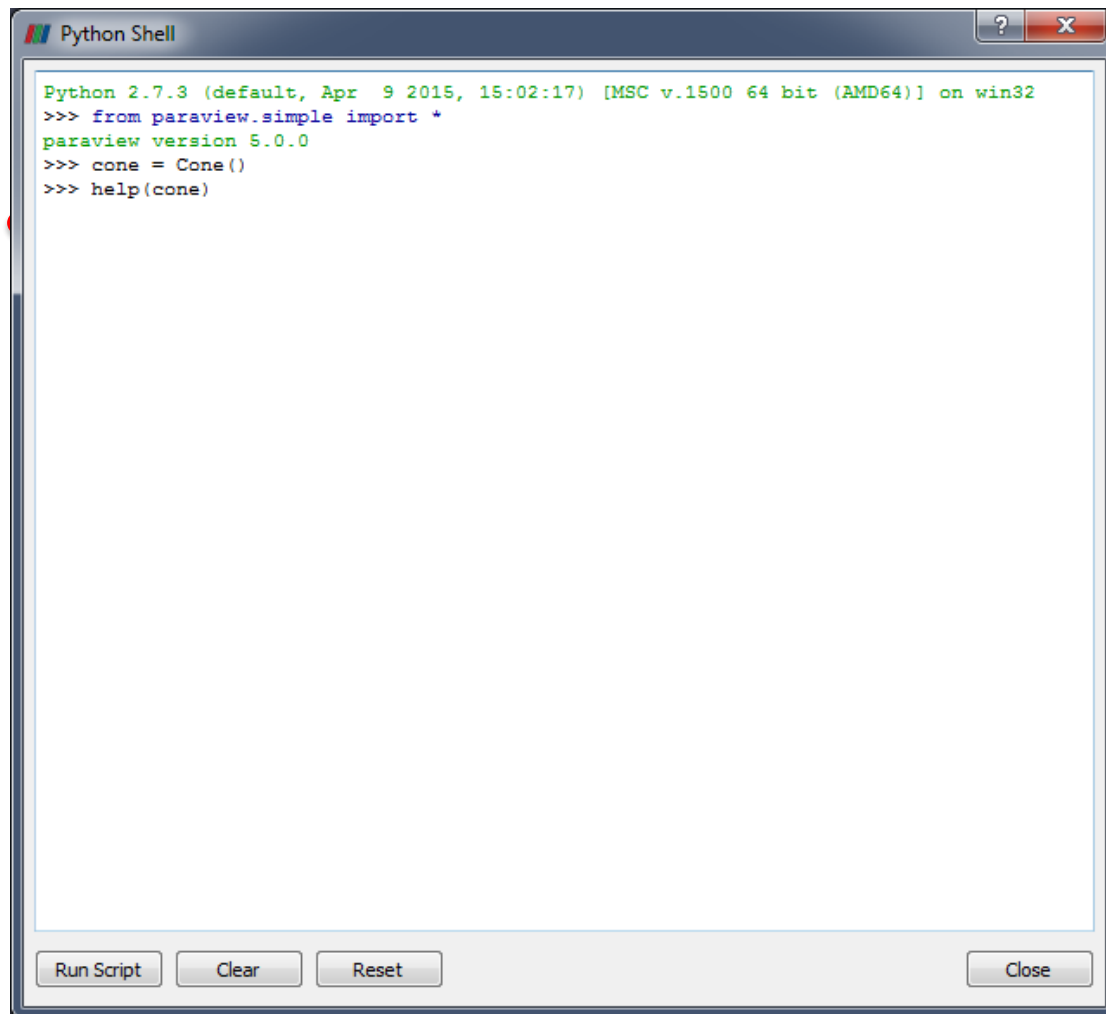
# PARAVIEW/PYTHON SCRIPTING

## CREATING A PIPELINE

### Create a Cone Object:

```
>>> cone = Cone()
```

```
>>> help(cone)
```



```
Python Shell
Python 2.7.3 (default, Apr  9 2015, 15:02:17) [MSC v.1500 64 bit (AMD64)] on win32
>>> from paraview.simple import *
paraview version 5.0.0
>>> cone = Cone()
>>> help(cone)
```

Run Script Clear Reset Close

# PARAVIEW/PYTHON SCRIPTING

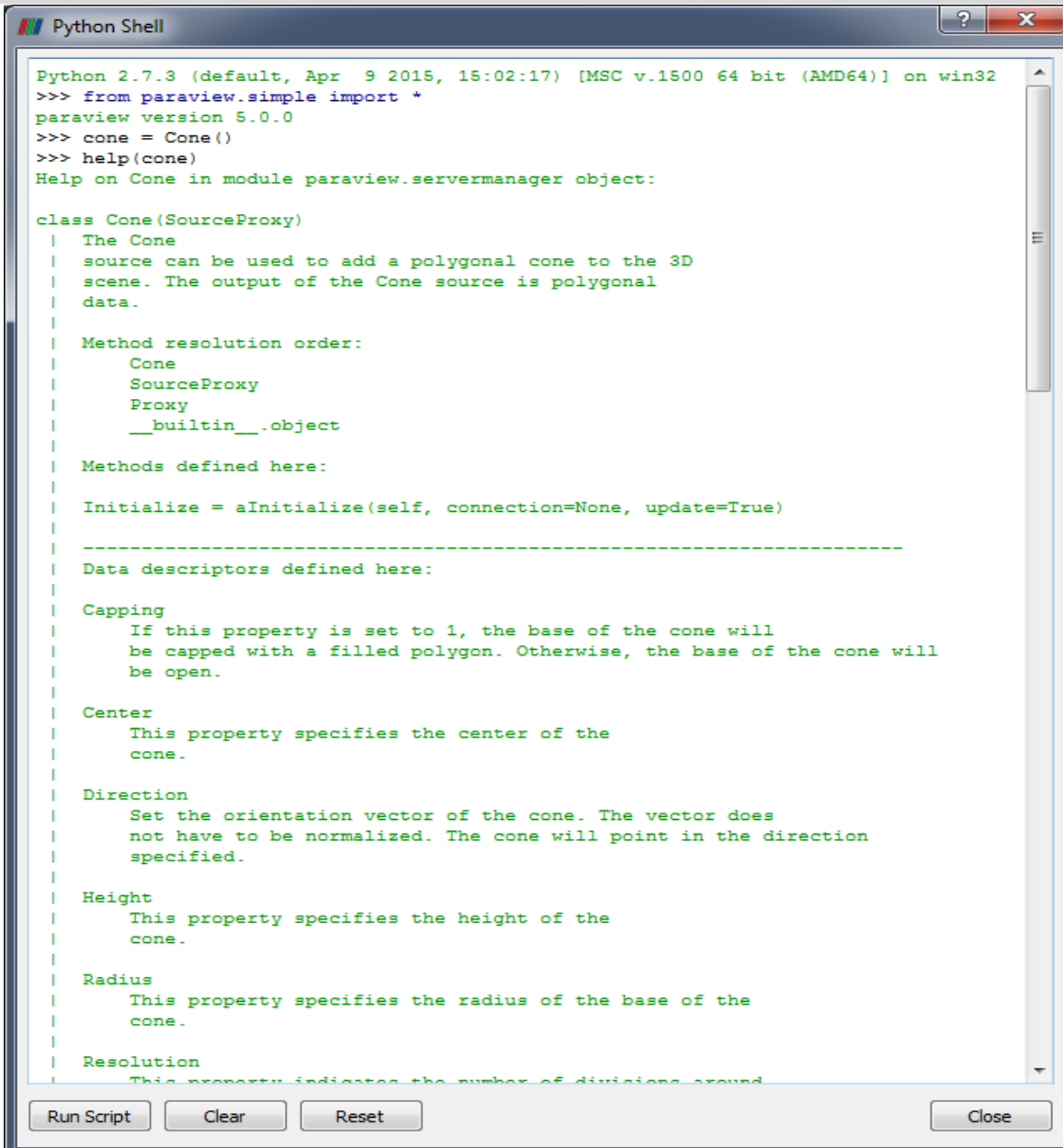
## CREATING A PIPELINE

### Create a Cone Object:

```
>>> cone = Cone()
```

```
>>> help(cone)
```

This gives you the full list of properties.



```
Python Shell
Python 2.7.3 (default, Apr  9 2015, 15:02:17) [MSC v.1500 64 bit (AMD64)] on win32
>>> from paraview.simple import *
paraview version 5.0.0
>>> cone = Cone()
>>> help(cone)
Help on Cone in module paraview.servermanager object:

class Cone(SourceProxy)
 | The Cone
 | source can be used to add a polygonal cone to the 3D
 | scene. The output of the Cone source is polygonal
 | data.
 |
 | Method resolution order:
 |   Cone
 |   SourceProxy
 |   Proxy
 |   __builtin__.object
 |
 | Methods defined here:
 |
 | Initialize = aInitialize(self, connection=None, update=True)
 |
 |-----
 | Data descriptors defined here:
 |
 | Capping
 |   If this property is set to 1, the base of the cone will
 |   be capped with a filled polygon. Otherwise, the base of the cone will
 |   be open.
 |
 | Center
 |   This property specifies the center of the
 |   cone.
 |
 | Direction
 |   Set the orientation vector of the cone. The vector does
 |   not have to be normalized. The cone will point in the direction
 |   specified.
 |
 | Height
 |   This property specifies the height of the
 |   cone.
 |
 | Radius
 |   This property specifies the radius of the base of the
 |   cone.
 |
 | Resolution
 |   This property indicates the number of divisions around
```

# PARAVIEW/PYTHON SCRIPTING

## CREATING A PIPELINE

### Create a Cone Object:

```
>>> cone = Cone()
```

```
>>> help(cone)
```

```
>>> cone.Resolution
```

```
6
```

```
>>>
```

Check what the resolution property is set to:

```
>>> cone.Resolution
```

# PARAVIEW/PYTHON SCRIPTING

## CREATING A PIPELINE

### Create a Cone Object:

```
>>> cone = Cone()
```

```
>>> help(cone)
```

```
>>> cone.Resolution
```

```
>>> cone.Resolution
```

```
6
```

```
>>>
```

You can increase the resolution:

```
>>> cone.Resolution = 32
```

# PARAVIEW/PYTHON SCRIPTING

## CREATING A PIPELINE

### Create a Cone Object:

```
>>> cone = Cone()
>>> help(cone)
>>> cone.Resolution
```

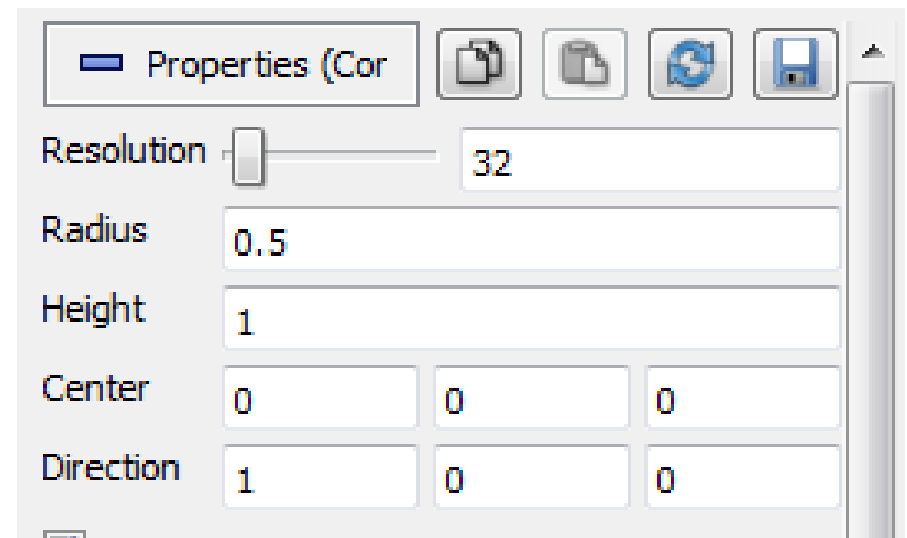
You can increase the resolution:

```
>>> cone.Resolution = 32
```

You could have specified a value for resolution when creating the object

```
>>> cone = Cone(Resolution=32)
```

```
>>> cone.Resolution
6
>>> cone.Resolution = 32
>>>
```



# PARAVIEW/PYTHON SCRIPTING

## CREATING A PIPELINE

### Create a Cone Object:

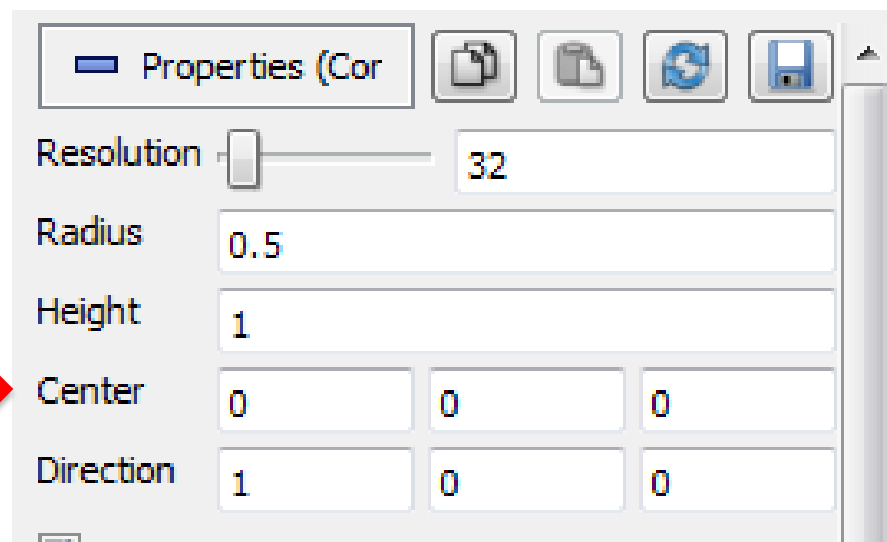
```
>>> cone = Cone()
>>> help(cone)
>>> cone.Resolution
```

You can assign values to any number of properties during construction using keyword arguments:

Type:

```
>>> cone.Center
[0.0, 0.0, 0.0]
```

```
>>> cone.Resolution
6
>>> cone.Resolution = 32
>>> cone.Center
[0.0, 0.0, 0.0]
```



# PARAVIEW/PYTHON SCRIPTING

## CREATING A PIPELINE

### Create a Cone Object:

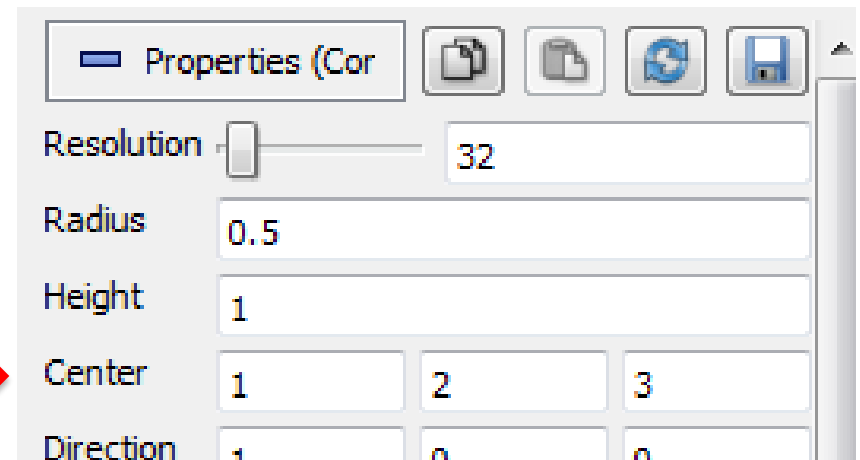
```
>>> cone = Cone()
```

```
>>> help(cone)
```

```
>>> cone.Resolution
```

```
>>> cone.Center
```

```
>>> cone.Center = [1, 2, 3]
```





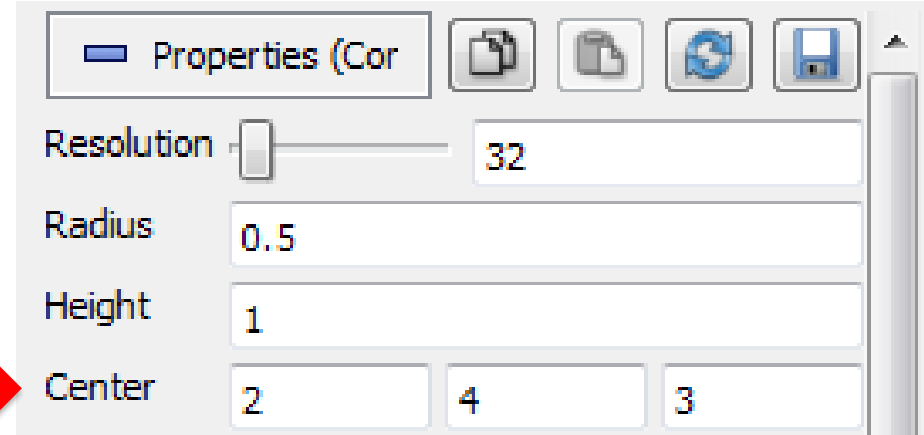
# PARAVIEW/PYTHON SCRIPTING

## CREATING A PIPELINE

### Create a Cone Object:

```
>>> cone = Cone()
>>> help(cone)
>>> cone.Resolution
>>> cone.Center
>>> cone.Center = [1, 2, 3]
>>> cone.Center[0:2] = [2, 4]
>>> cone.Center
[2.0, 4.0, 3.0]
```

Vector properties such as this one support setting and retrieval of individual elements, as well as slices (ranges of elements).



# PARAVIEW/PYTHON SCRIPTING

## CREATING A PIPELINE

### Create a Cone Object:

```
>>> cone = Cone()
>>> help(cone)
>>> cone.Resolution
>>> cone.Center
>>> cone.Center = [1, 2, 3]
>>> cone.Center[0:2] = [2, 4]
>>> cone.Center
[2.0, 4.0, 3.0]
```

Apply a shrink filter to the cone

```
>>> shrinkFilter = Shrink(cone)
```

# PARAVIEW/PYTHON SCRIPTING

## CREATING A PIPELINE

### Create a Cone Object:

```
>>> cone = Cone()
>>> help(cone)
>>> cone.Resolution
>>> cone.Center
>>> cone.Center = [1, 2, 3]
>>> cone.Center[0:2] = [2, 4]
>>> cone.Center
[2.0, 4.0, 3.0]
```

Apply a shrink filter to the cone

```
>>> shrinkFilter = Shrink(cone)
>>> shrinkFilter.Input
```

# PARAVIEW/PYTHON SCRIPTING

## CREATING A PIPELINE

### Create a Cone Object:

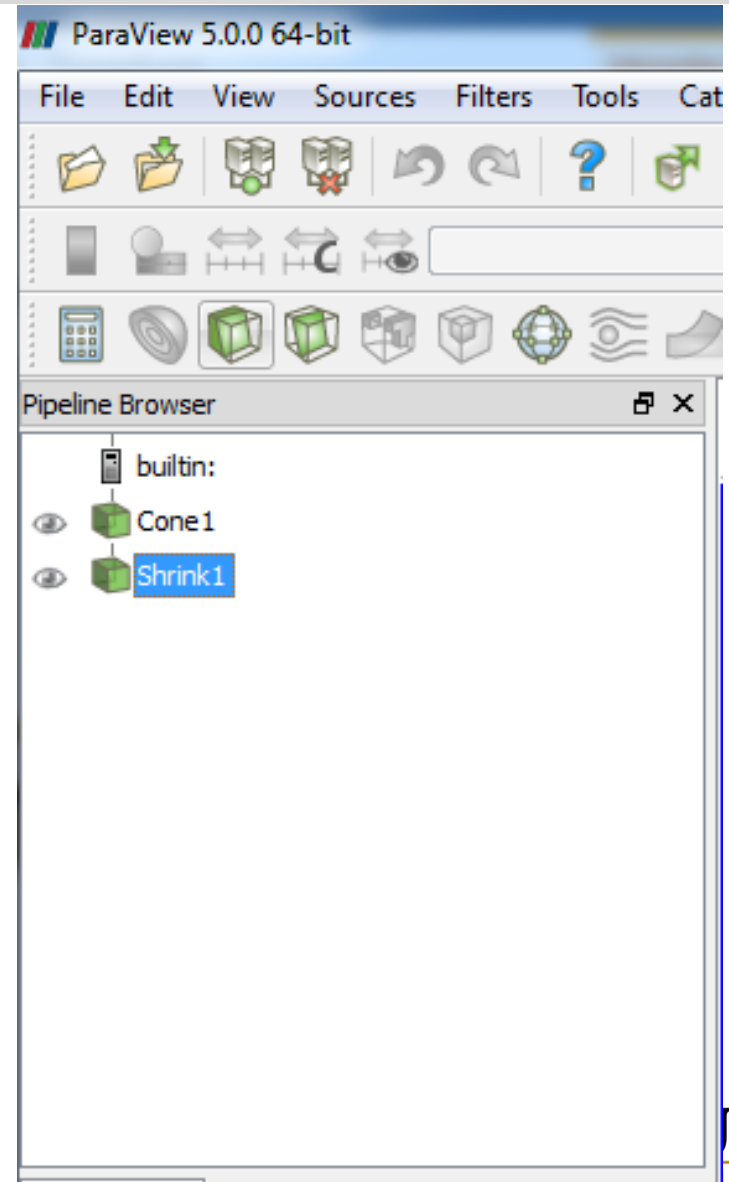
```
>>> cone = Cone()
>>> help(cone)
>>> cone.Resolution
>>> cone.Center
>>> cone.Center = [1, 2, 3]
>>> cone.Center[0:2] = [2, 4]
>>> cone.Center
[2.0, 4.0, 3.0]
>>> shrinkFilter = Shrink(cone)
>>> shrinkFilter.Input
<paraview.servermanager.Cone object at 0x000000000896EEB8>
>>>
```

# PARAVIEW/PYTHON SCRIPTING

## CREATING A PIPELINE

### Create a Cone Object:

```
>>> cone = Cone()
>>> help(cone)
>>> cone.Resolution
>>> cone.Center
>>> cone.Center = [1, 2, 3]
>>> cone.Center[0:2] = [2, 4]
>>> cone.Center
[2.0, 4.0, 3.0]
>>> shrinkFilter = Shrink(cone)
>>> shrinkFilter.Input
<paraview.servermanager.Cone
object at 0x000000000896EEB8>
>>>
```



# PARAVIEW/PYTHON SCRIPTING

## CREATING A PIPELINE

### Create a Cone Object:

```
>>> cone = Cone()
>>> help(cone)
>>> cone.Resolution
>>> cone.Center
>>> cone.Center = [1, 2, 3]
>>> cone.Center[0:2] = [2, 4]
>>> cone.Center
[2.0, 4.0, 3.0]
>>> shrinkFilter = Shrink(cone)
>>> shrinkFilter.Input
<paraview.servermanager.Cone
object at 0x000000000896EEB8>
>>>
```

At this point you can force ParaView to update, which will also cause the execution of the cone source

# PARAVIEW/PYTHON SCRIPTING

## CREATING A PIPELINE

### Create a Cone Object:

```
>>> shrinkFilter.UpdatePipeline()
```

```
>>> shrinkFilter.GetDataInformation().GetNumberOfCells()
```

```
33L
```

```
>>> shrinkFilter.GetDataInformation().GetNumberOfPoints()
```

```
128L
```

```
>>>
```

# PARAVIEW/PYTHON SCRIPTING

## CREATING A PIPELINE

- Create Cone Object
- Set Cone Resolution
- Set Cone Center Properties
- Apply Shrink Filter to the Cone
- Updated Pipeline



# PARAVIEW/PYTHON SCRIPTING

## RENDERING

Two objects are needed to render the output

- **A representation** – takes a data object and renders it in a view
- **A view** – responsible for managing a render context and a collection of representations

# PARAVIEW/PYTHON SCRIPTING

## RENDERING

Type at prompt:

```
>>> Show(shrinkFilter)
```

```
>>> Render()
```

```
>>> Show(shrinkFilter)
```

```
<paraview.servermanager.UnstructuredGridRe  
presentation object at 0x000000000BE85B70>
```

```
>>> Render()
```

```
<paraview.servermanager.RenderView object  
at 0x000000000C26D278>
```

```
>>>
```

# PARAVIEW/PYTHON SCRIPTING

## RENDERING

Type at prompt:

```
>>> Show(shrinkFilter)
```

```
>>> Render()
```

```
>>> Show(shrinkFilter)
```

```
<paraview.servermanager.UnstructuredGridRe  
presentation object at 0x000000000BE85B70>
```

```
>>> Render()
```

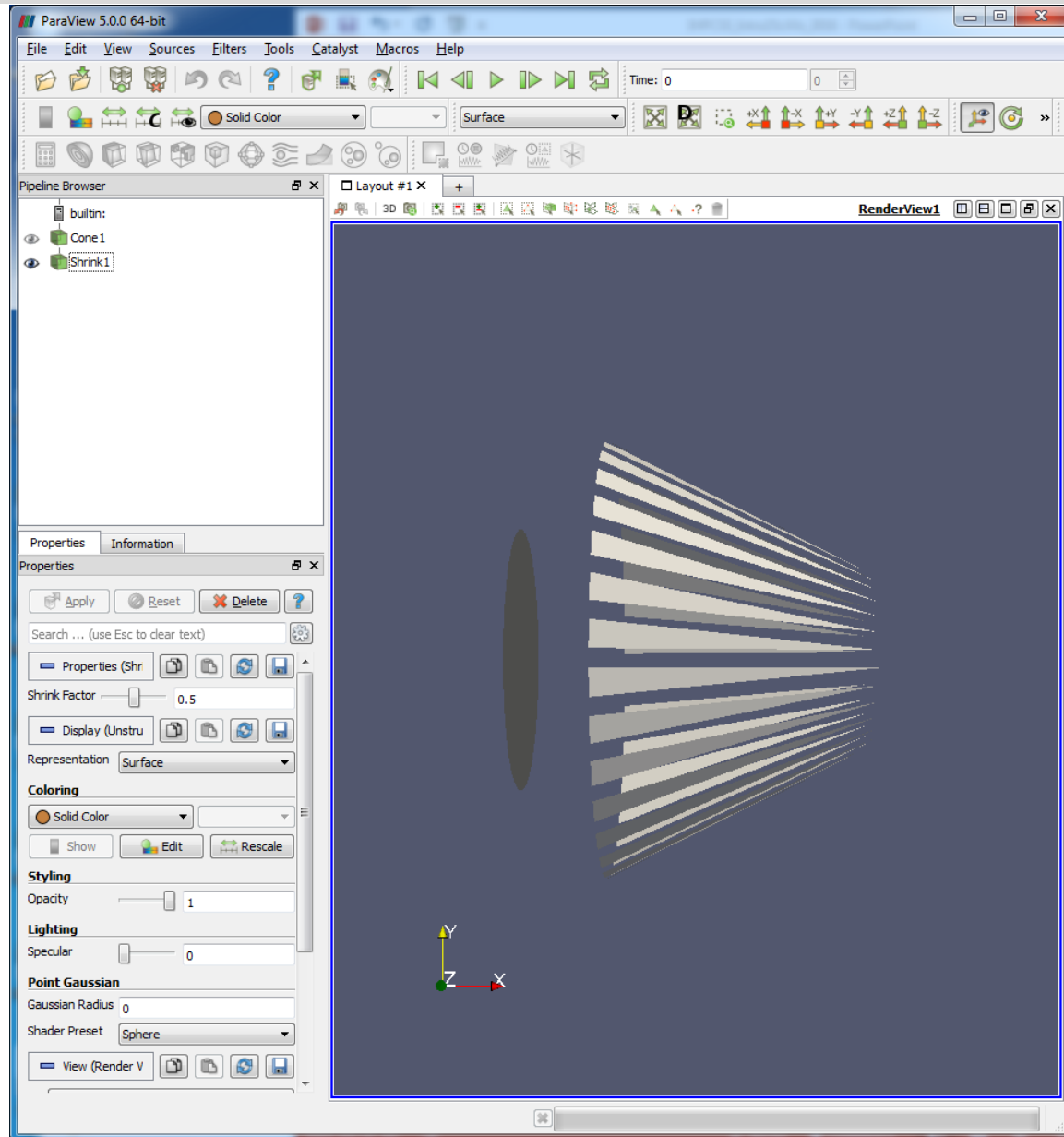
```
<paraview.servermanager.RenderView object  
at 0x000000000C26D278>
```

```
>>>
```

# PARAVIEW/PYTHON SCRIPTING

## RENDERING

- Should see something similar to this



# PARAVIEW/PYTHON SCRIPTING

## CREATING A PIPELINE – WHAT DID WE DO

```
# Create a cone and assign it as the active object  
# Set a property of the active object  
# Apply the shrink filter to the active object  
# Shrink is now active  
# Show shrink  
# Render the active view
```

# PARAVIEW/PYTHON SCRIPTING

## CREATING A PIPELINE

- The value returned by Cone() and Shrink() was assigned to Python variables and used to build the pipeline
- ParaView keeps track of the last pipeline object created by the user. This allows you to accomplish everything that was just done

# PARAVIEW/PYTHON SCRIPTING

## CREATING A PIPELINE

```
>>> from paraview.simple import *
# Create a cone and assign it as the active object
>>> Cone()
<paraview.servermanager.Cone object at 0x2910f0>
# Set a property of the active object
>>> SetProperty(Resolution=32)
# Apply the shrink filter to the active object
# Shrink is now active
>>> Shrink()
<paraview.servermanager.Shrink object at 0xaf64050>
# Show shrink
>>> Show()
<paraview.servermanager.UnstructuredGridRepresentation object at
0xaf57f90>
# Render the active view
>>> Render()
<paraview.servermanager.RenderView object at 0xaf57ff0>
```

# RUN FROM SCRIPT

Type the following code in a text editor

```
Cone()  
SetProperties(Resolution=32)  
Shrink()  
Show()  
Render()
```

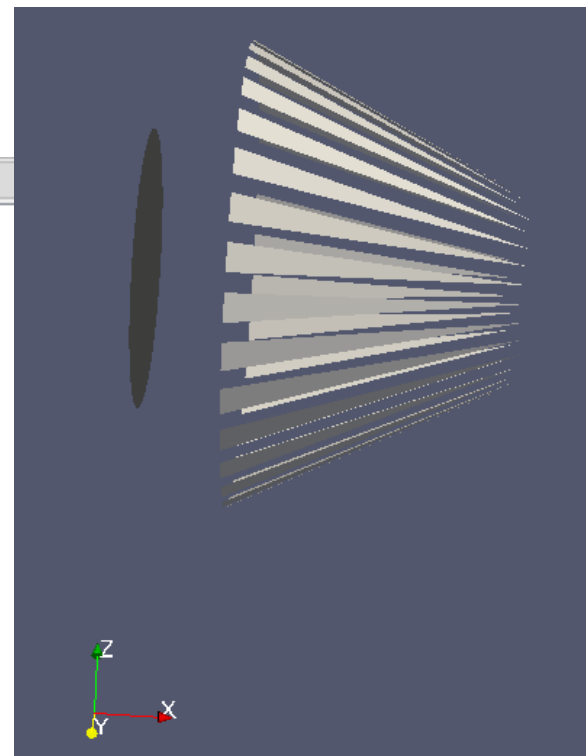
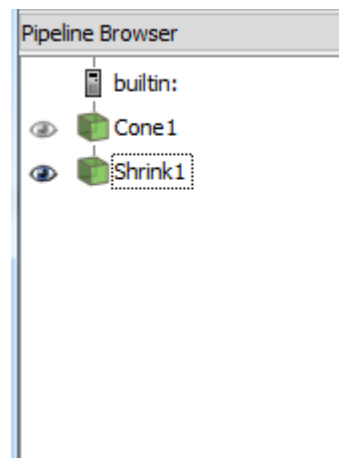
Save file as **testScript.py**

Click RUN SCRIPT from Python Shell  
Locate and select script  
Click OK

**Should see**

**New objects in Pipeline Browser**

**Cone rendering in 3D Viewer**





# ADDITIONAL RESOURCES

<http://www.paraview.org>

ParaView User's Guide: Downloaded with ParaView

ParaView Sample Data

[http://www.paraview.org/Wiki/The\\_ParaView\\_Tutorial](http://www.paraview.org/Wiki/The_ParaView_Tutorial)

ParaView/Python Scripting – KitwarePublic

[http://www.paraview.org/Wiki/ParaView/Python\\_Scripting](http://www.paraview.org/Wiki/ParaView/Python_Scripting)

ParaView Server Manager

<http://www.paraview.org/ParaView/Doc/Nightly/www/py-doc/paraview.servermanager.html>

## Vetria L. Byrd

Assistant Professor

Computer Graphics Technology

[vlbyrd@purdue.edu](mailto:vlbyrd@purdue.edu)

<https://polytechnic.purdue.edu/profile/vbyrd>

@VByrdPhD, @BPViz, @VisREU

## Purdue Polytechnic Institute

[polytechnic.purdue.edu](http://polytechnic.purdue.edu)



**PURDUE**  
POLYTECHNIC

