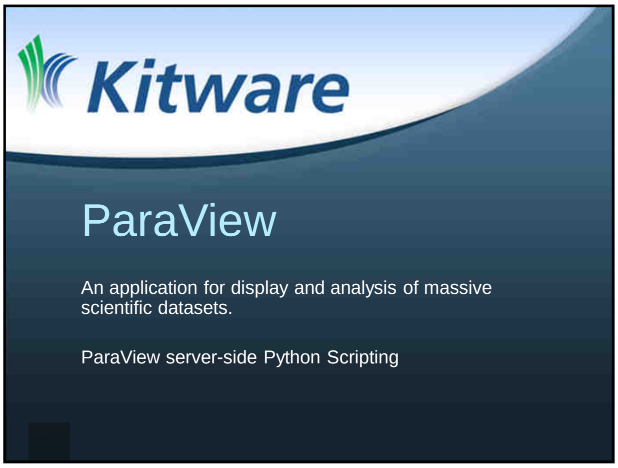
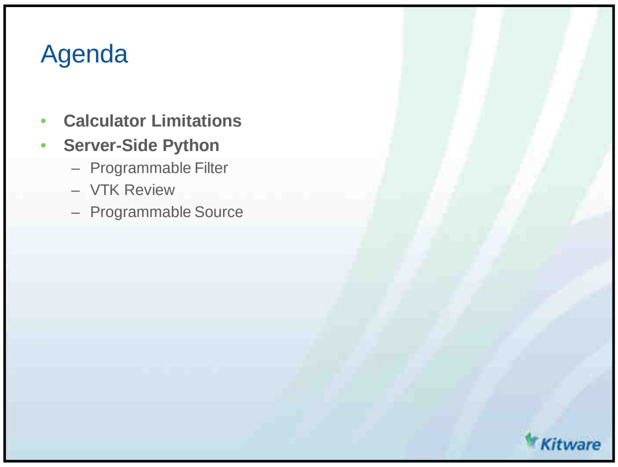


ParaView

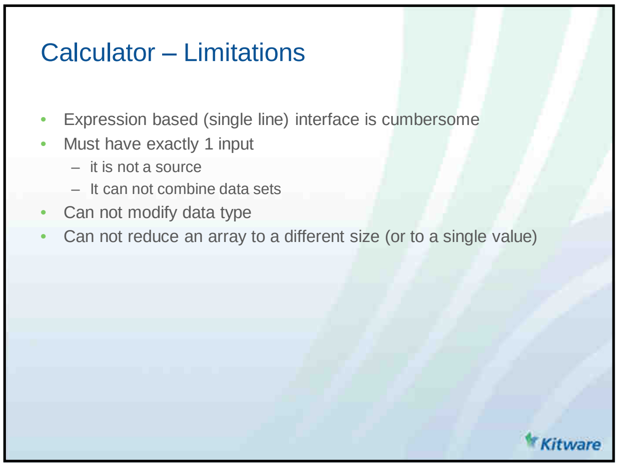
An application for display and analysis of massive scientific datasets.

ParaView server-side Python Scripting



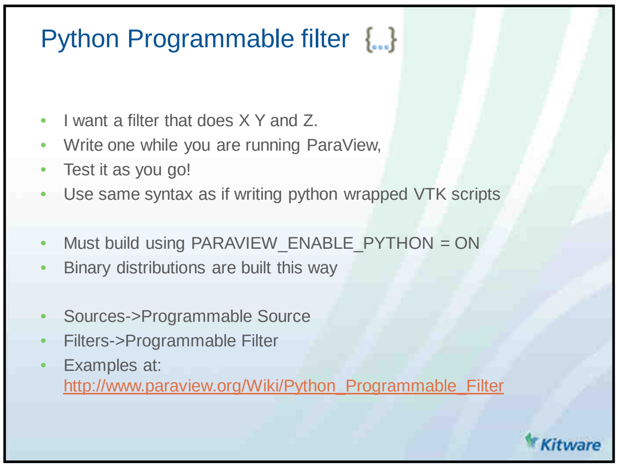
Agenda

- **Calculator Limitations**
- **Server-Side Python**
  - Programmable Filter
  - VTK Review
  - Programmable Source



Calculator – Limitations

- Expression based (single line) interface is cumbersome
- Must have exactly 1 input
  - it is not a source
  - It can not combine data sets
- Can not modify data type
- Can not reduce an array to a different size (or to a single value)



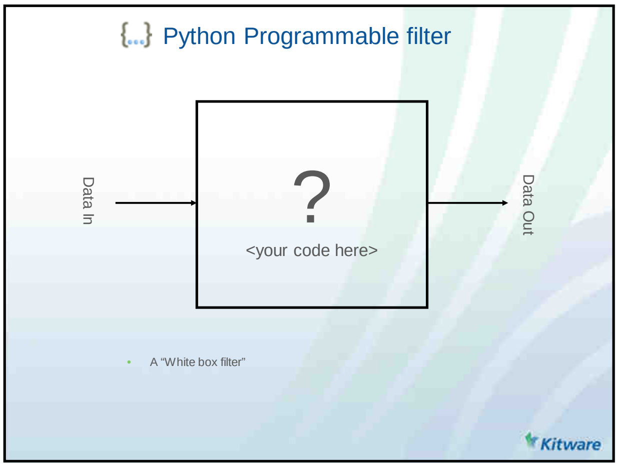
Python Programmable filter {...}

- I want a filter that does X Y and Z.
- Write one while you are running ParaView,
- Test it as you go!
- Use same syntax as if writing python wrapped VTK scripts

- Must build using PARAVIEW_ENABLE_PYTHON = ON
- Binary distributions are built this way

- Sources->Programmable Source
- Filters->Programmable Filter
- Examples at: http://www.paraview.org/Wiki/Python_Programmable_Filter

Kitware



{...} Python Programmable filter

Data In

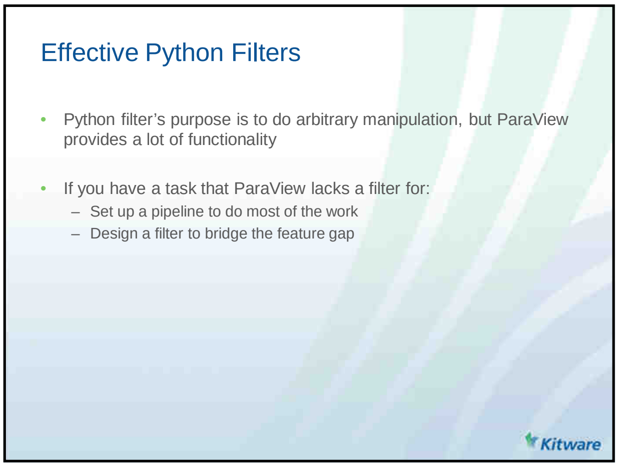
?

<your code here>

Data Out

- A "White box filter"

Kitware

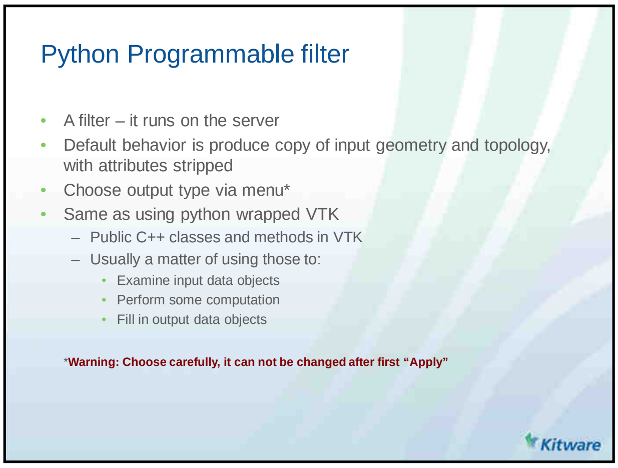


Effective Python Filters

- Python filter's purpose is to do arbitrary manipulation, but ParaView provides a lot of functionality

- If you have a task that ParaView lacks a filter for:
  - Set up a pipeline to do most of the work
  - Design a filter to bridge the feature gap

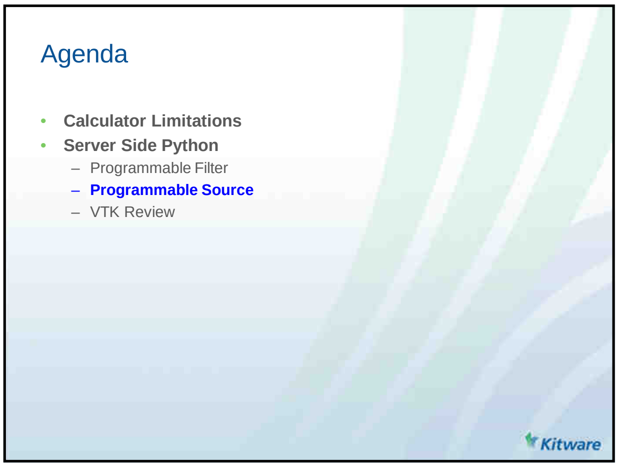
Kitware



Python Programmable filter

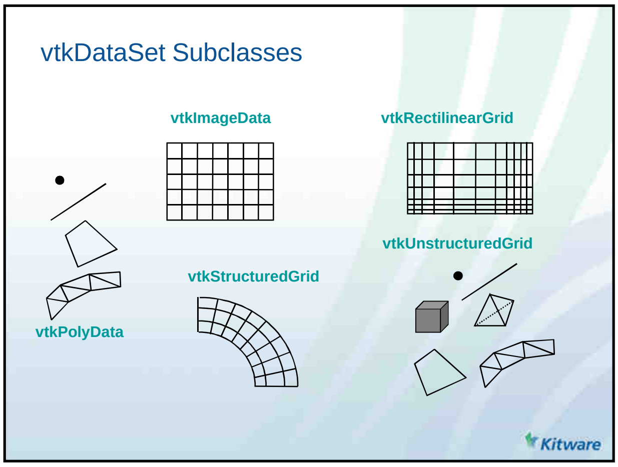
- A filter – it runs on the server
- Default behavior is produce copy of input geometry and topology, with attributes stripped
- Choose output type via menu*
- Same as using python wrapped VTK
  - Public C++ classes and methods in VTK
  - Usually a matter of using those to:
    - Examine input data objects
    - Perform some computation
    - Fill in output data objects

*Warning: Choose carefully, it can not be changed after first "Apply"



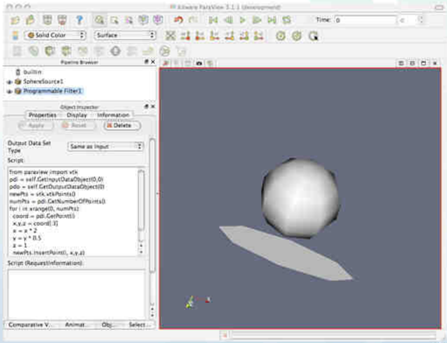
Agenda

- **Calculator Limitations**
- **Server Side Python**
  - Programmable Filter
  - **Programmable Source**
  - VTK Review



vtkDataSet Subclasses

{...} Programmable Filter



Output Data Set Type: Same as Input

Script:

```
from paraview import vtk
pdi = self.GetInputDataObject(0,0)
pdo = self.GetOutputDataObject(0)
newPts = vtk.vtkPoints()
numPts = pdi.GetNumberOfPoints()
for i in xrange(0, numPts):
 coord = pdi.GetPoint(i)
 x,y,z = coord[:3]
 x = x * 2
 y = y * 0.5
 z = 1
 newPts.InsertPoint(i, x,y,z)
```

Script (RequestInformation):

Kitware

VTK Data Structure Review*

- VTK Stores data in vtkAbstractArrays
- A vtkDataSet is container for anything with inherent structure
  - Dataset class consists of arrays for
    - Geometry: points (x-y-z coordinates)
    - Topology: cells (e.g., polygons, lines, voxels) which are defined by connectivity list referring to points ids
      - Implicit vs. Explicit representations
    - Attribute Data: arrays for values
- Access to everything is via integer ID
- vtkAbstractArray *array->GetTuple(ID);

* See VTK User's Guide Chapter 11

Kitware

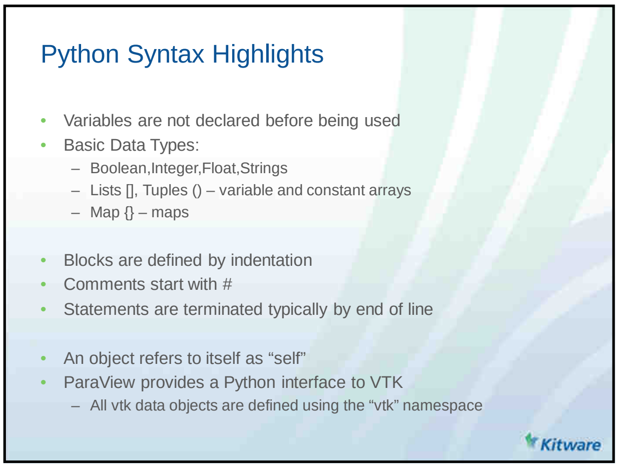
Data Set Attributes

- vtkDataSet can designate arrays as special:
  - Scalars
  - Vectors - 3-vector
  - Tensors - 3x3 symmetric matrix
  - Normals - unit vector
  - Texture Coordinates 1-3D
- For example, many filters operate on "The Active Scalar array" of input unless otherwise directed
- Both Cells and Points have active arrays

```
vtkDoubleArray *cellScalars =
  aDataSet->GetCellData()->GetScalars();
double sixthCellsScalar = cellScalars->GetValue(5);
```

Kitware

For private use only – Sharing not permitted
Copyright © Kitware – www.kitware.com

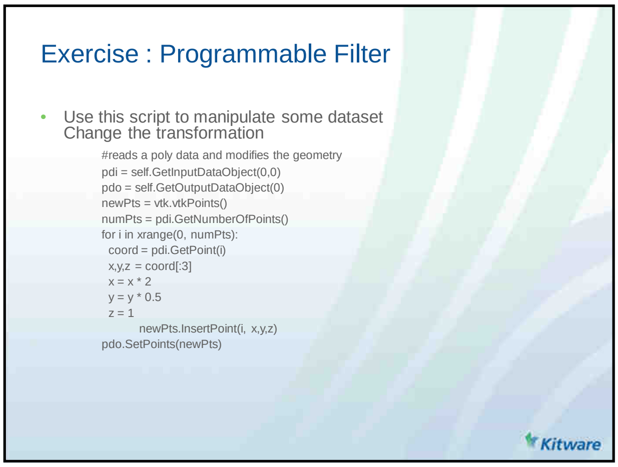


Python Syntax Highlights

- Variables are not declared before being used
- Basic Data Types:
  - Boolean,Integer,Float,Strings
  - Lists [], Tuples () – variable and constant arrays
  - Map {} – maps

- Blocks are defined by indentation
- Comments start with #
- Statements are terminated typically by end of line

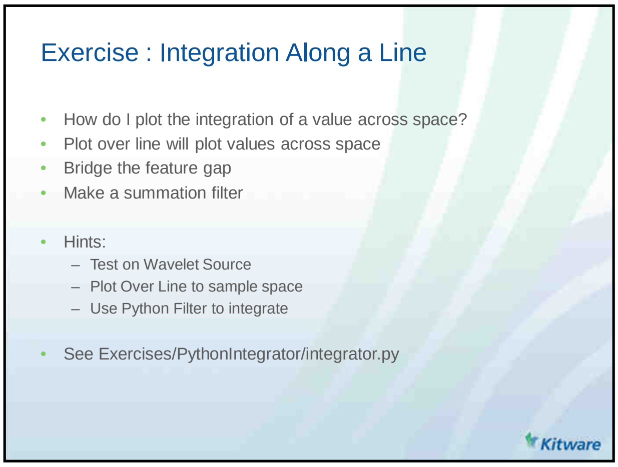
- An object refers to itself as “self”
- ParaView provides a Python interface to VTK
  - All vtk data objects are defined using the “vtk” namespace



Exercise : Programmable Filter

- Use this script to manipulate some dataset Change the transformation

```
#reads a poly data and modifies the geometry
pdi = self.GetInputDataObject(0,0)
pdo = self.GetOutputDataObject(0)
newPts = vtk.vtkPoints()
numPts = pdi.GetNumberOfPoints()
for i in xrange(0, numPts):
  coord = pdi.GetPoint(i)
  x,y,z = coord[:3]
  x = x * 2
  y = y * 0.5
  z = 1
          newPts.InsertPoint(i, x,y,z)
pdo.SetPoints(newPts)
```

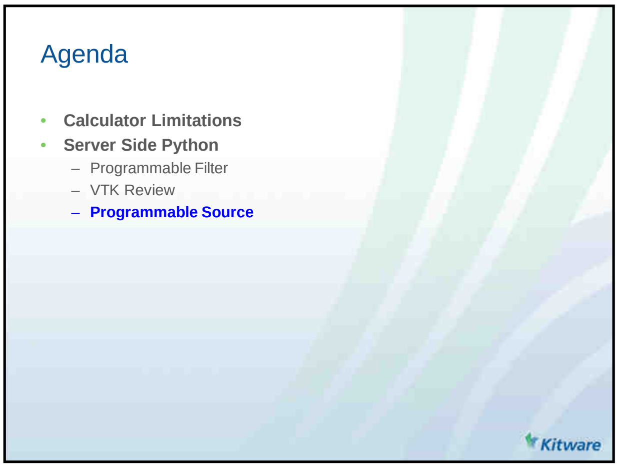


Exercise : Integration Along a Line

- How do I plot the integration of a value across space?
- Plot over line will plot values across space
- Bridge the feature gap
- Make a summation filter

- Hints:
  - Test on Wavelet Source
  - Plot Over Line to sample space
  - Use Python Filter to integrate

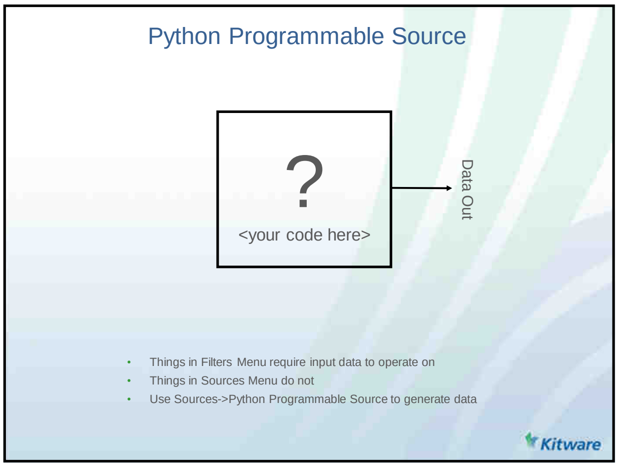
- See Exercises/PythonIntegrator/integrator.py



Agenda

- **Calculator Limitations**
- **Server Side Python**
  - Programmable Filter
  - VTK Review
  - **Programmable Source**

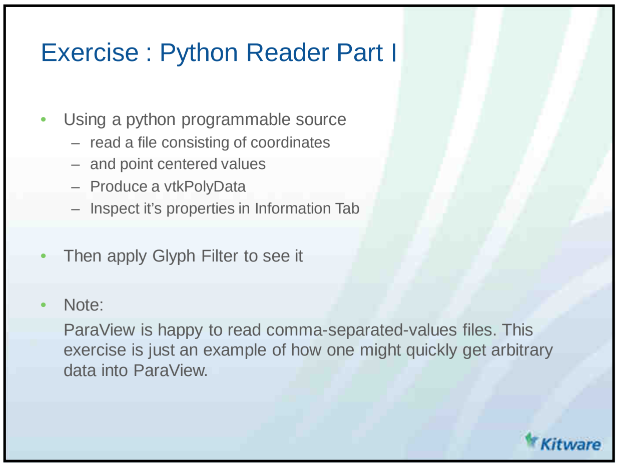
Kitware



Python Programmable Source

- Things in Filters Menu require input data to operate on
- Things in Sources Menu do not
- Use Sources->Python Programmable Source to generate data

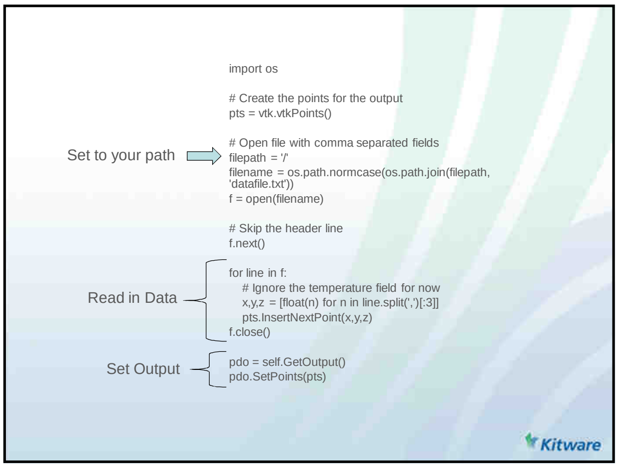
Kitware



Exercise : Python Reader Part I

- Using a python programmable source
  - read a file consisting of coordinates
  - and point centered values
  - Produce a vtkPolyData
  - Inspect it's properties in Information Tab
- Then apply Glyph Filter to see it
- Note:
  ParaView is happy to read comma-separated-values files. This exercise is just an example of how one might quickly get arbitrary data into ParaView.

Kitware

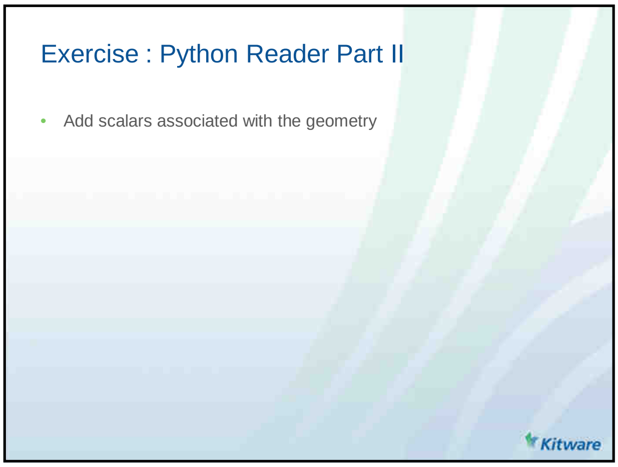


Set to your path

Read in Data

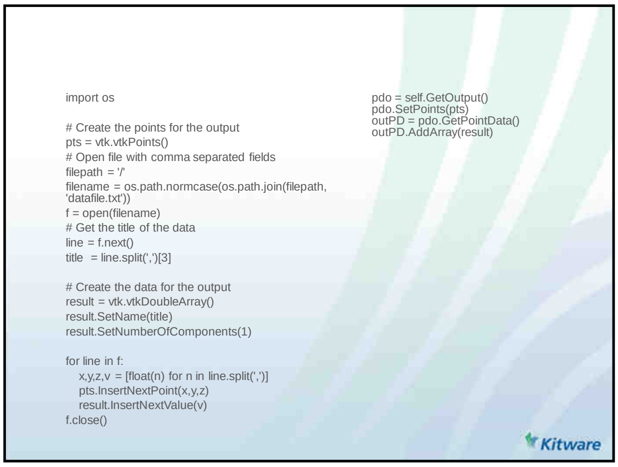
Set Output

```
import os

# Create the points for the output
pts = vtk.vtkPoints()

# Open file with comma separated fields
filepath = '/'
filename = os.path.normcase(os.path.join(filepath,
'datafile.txt'))
f = open(filename)

# Skip the header line
f.next()

for line in f:
    # Ignore the temperature field for now
    x,y,z = [float(n) for n in line.split(',')[:3]]
    pts.InsertNextPoint(x,y,z)
f.close()

pdo = self.GetOutput()
pdo.SetPoints(pts)
```



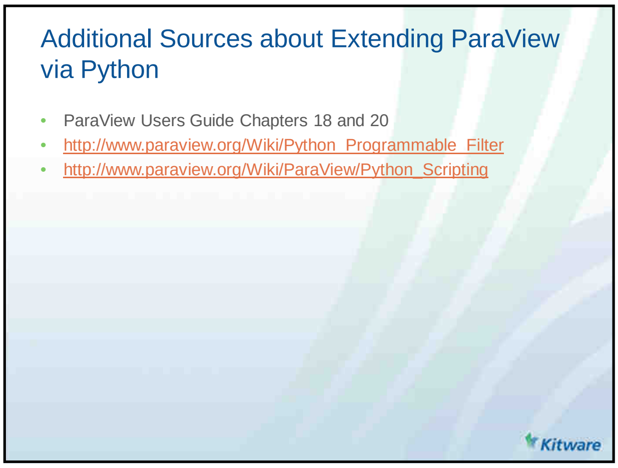
Exercise : Python Reader Part II

- Add scalars associated with the geometry



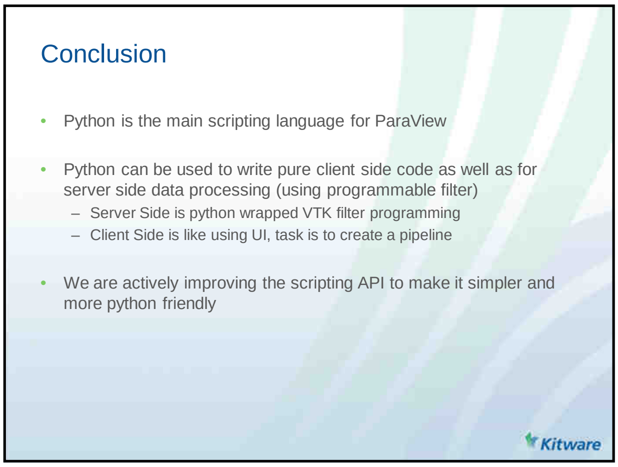
```
import os

# Create the points for the output
pts = vtk.vtkPoints()
# Open file with comma separated fields
filepath = '/'
filename = os.path.normcase(os.path.join(filepath,
'datafile.txt'))
f = open(filename)
# Get the title of the data
line = f.next()
title  = line.split(',')[3]

# Create the data for the output
result = vtk.vtkDoubleArray()
result.SetName(title)
result.SetNumberOfComponents(1)

for line in f:
    x,y,z,v = [float(n) for n in line.split(',')]
    pts.InsertNextPoint(x,y,z)
    result.InsertNextValue(v)
f.close()
```

```
pdo = self.GetOutput()
pdo.SetPoints(pts)
outPD = pdo.GetPointData()
outPD.AddArray(result)
```



Additional Sources about Extending ParaView via Python

- ParaView Users Guide Chapters 18 and 20
- http://www.paraview.org/Wiki/Python_Programmable_Filter
- http://www.paraview.org/Wiki/ParaView/Python_Scripting



Conclusion

- Python is the main scripting language for ParaView
- Python can be used to write pure client side code as well as for server side data processing (using programmable filter)
  - Server Side is python wrapped VTK filter programming
  - Client Side is like using UI, task is to create a pipeline
- We are actively improving the scripting API to make it simpler and more python friendly

THE END