

---



Curso de

# ***ParaView***






## 04. Visualización Python Views

Michael Heredia Pérez  
[mherediap@unal.edu.co](mailto:mherediap@unal.edu.co)

Universidad Nacional de Colombia  
Sede Manizales

## Python Views

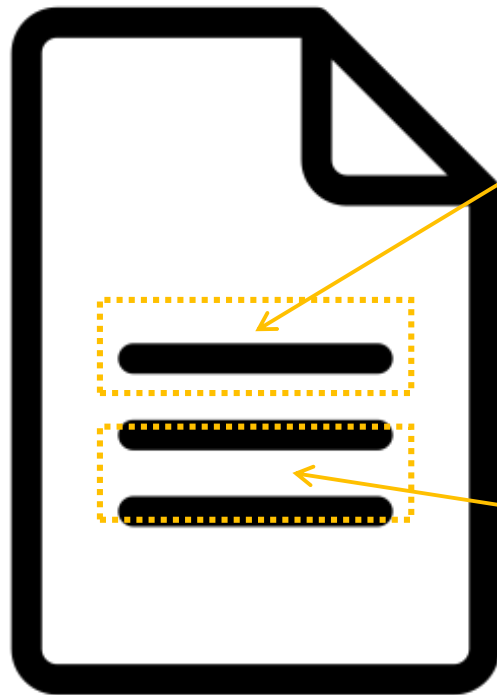
---

-  Provee una manera de presentar gráficos hechos en scripts de Python directamente dentro de paraview.
-  Es un código de Python que genera la imagen en la ventana de visualización (*viewport*).
-  Se puede graficar cualquier conjunto de datos que pueda ser cargado a paraview.
-  El código se lee en un espacio independiente, no tiene en cuenta las variables que ya estén presentes en paraview u otros archivos, tomando las nuevas variables como globales.
-  El entorno se reinicia cada que el código es leído, por lo tanto la información no se guarda entre corridas.

## Sobre el código

---

Un código para generar *Python Views* debe contener dos funciones:



### Función de lectura

Esta función lee los datos que se van a graficar

### Función de ploteo

Esta función grafica los datos cargados mediante interacciones Python – VTK - ParaView

## Función de lectura

---

- ❖ Todas las *Python Views* ocurren en el cliente, inclusive trabajando el modo *client-server*.
- ❖ Esta función solamente corre en procesos de *data-server*.
- ❖ Tipos de arreglos leíbles:
  - Point data
  - Cell data
  - Field data
  - Table row data
- ❖ Provee acceso al data objeto seleccionado en el servidor y así poder el usuario indagar cualquier aspecto de los datos usando los métodos que VTK y *paraview* tienen con las partes del *Python-wrapping*.

Patrón de diseño utilizado cuando se trabaja con funciones relativamente complejas y que se relacionan:

<https://wiki.python.org/moin/FunctionWrappers>

## Función de ploteo

---

- ❖ Se dan los comandos de ploteo o renderizado.
- ❖ Tiene acceso a todos los objetos reunidos en los nodos del servidor, pero solamente accede a los datos que se hayan cargado con la primera función.
- ❖ La lógica de la función es:
  1. Establecer objetos de una librería para graficar.
  2. Convertir información de VTK a un formato que se pueda entender por la librería.
  3. Plotear la información.
  4. Convertir el *plot* a una imagen válida para la ventana de visualización, un objeto `avtkImageData`.

## Seleccionar arreglos de datos para plotear

---

- ❖ Todos los renderizados de las Python Views ocurren el cliente, así que se le deben pedir al servidor.
- ❖ Ya que los datos que residen en el servidor de ParaView son demasiado grandes, resulta impráctico llamarlo completamente al cliente, así que se tienen mecanismo para seleccionar específicamente los arreglos de datos que se transferirán al cliente.
- ❖ El código contiene una función `setup_data(view)`.

## Seleccionar arreglos de datos para plotear

```
def setup_data(view):
```

El argumento `view` es el objeto VTK para la *Python View*

```
# Se itera sobre los objetos visibles.
```

```
for i in xrange(view.GetNumberOfVisibleDataObjects()):
```

```
# Se accede al objeto:
```

```
dataObjeto = view.GetVisibleDataObjectForSetup(i)
```

```
# Este data objeto tiene el mismo tipo de dato y estructura que el  
# data objeto presente en el servidor, se puede interactuar con el a  
# través del Python wrapping.
```

```
print('Tamaño de memoria: {0} kilobytes.'.format(dataObjeto.GetActualMemorySize()))
```

```
# Se hace una limpieza de los arreglos que hayan sido llamados  
# previamente a esta función.
```

```
view.DisableAllAttributeArrays()
```

```
# Por defecto no se van a pasar arreglos al cliente, esto debe ser  
# explícito.
```

```
view.SetAttributeArrayStatus(i, vtkDataObject.POINT, 'Density', 1)
```

```
view.SetAttributeArrayStatus(i, vtkDataObject.POINT, 'Momentum', 1)
```

```
# Se puede agregar otros arreglos de atributos de manera similar.
```

```
view.SetAttributeArrayStatus(i, vtkDataObject.FIELD, 'fieldData', 1)
```

view cuenta con varios métodos para acceder a la información

Información  
en el  
cuaderno

## Plotear los datos

- Una vez se tiene en el cliente los arreglos de datos, se ejecuta una función `render(view, width, height)` que está en el script.
- Se genera un objeto `vtkImageData`, el cual se visualiza en el *viewport*.

```
def render(view, width, height):
```

El argumento `view`: `vtkPythonObject`

```
# Se define la figura.
figura = python_view.matplotlib_figure(width, height)

# Se decorala figura.
ax = figura.add_subplot(1,1,1)
ax.minorticks_on()
ax.set_title('Título')
ax.set_xlabel('Etiqueta eje x')
ax.set_ylabel('Etiqueta eje y')

# Se procesa el primer objeto visible del pipeline browser.
dataObjeto = view.GetVisibleDataObjectForRendering(0) # 0 : el 1ro.

arreglo = dataObjeto.GetPointData().GetArray('X')

# Se convierte el arreglo VTK a uno de numpy para graficarse.
arreglo_np = vtk_to_numpy(arreglo)

ax.hist(arreglo_np, bins = 10)

return python_view.figure_to_image(figura)
# Convierte la figura de matplotlib en un objeto vtkImageData.
```

IMPORTANTE y necesario  
que la función retorne la  
imagen transformada