



# Slicer Developer Tutorial: Programming in Slicer5 - Part 1

**Sonia Pujol, Ph.D.**

Assistant Professor of Radiology  
Director of 3D Slicer Training & Education  
Brigham and Women's Hospital  
Harvard Medical School

**Steve Pieper, Ph.D.**

3D Slicer Chief Architect  
Isomics Inc.

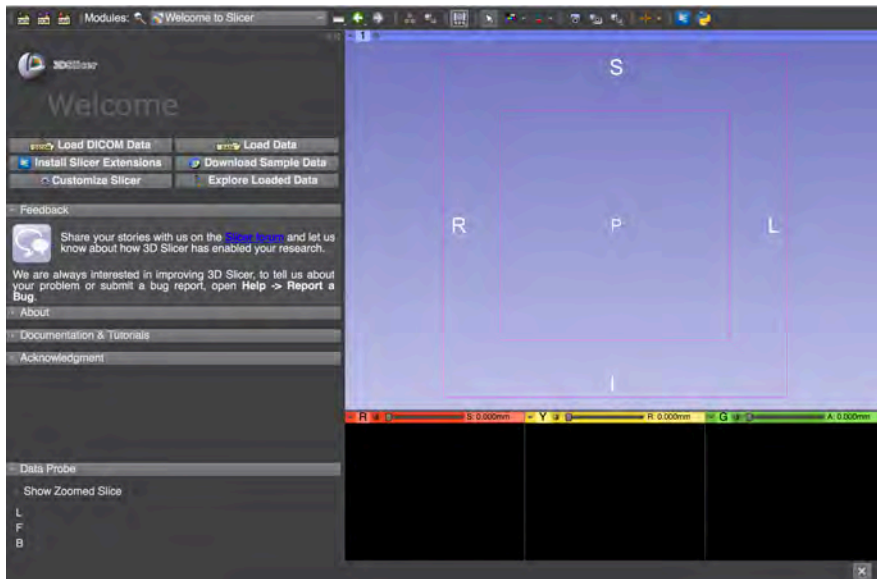
# Tutorial Outline

- Part 1: Slicer Modules Overview
- Part 1: Getting Familiar with the Python environment in 3D Slicer
- Part 2: Getting Familiar with Qt in 3D Slicer

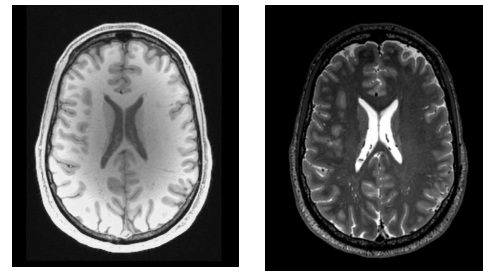
# Disclaimer

- 3D Slicer is a free open source software for medical image computing research distributed under a BDS style license.
- The software is not FDA approved or CE-Marked, and is for research use only.

# Tutorial materials



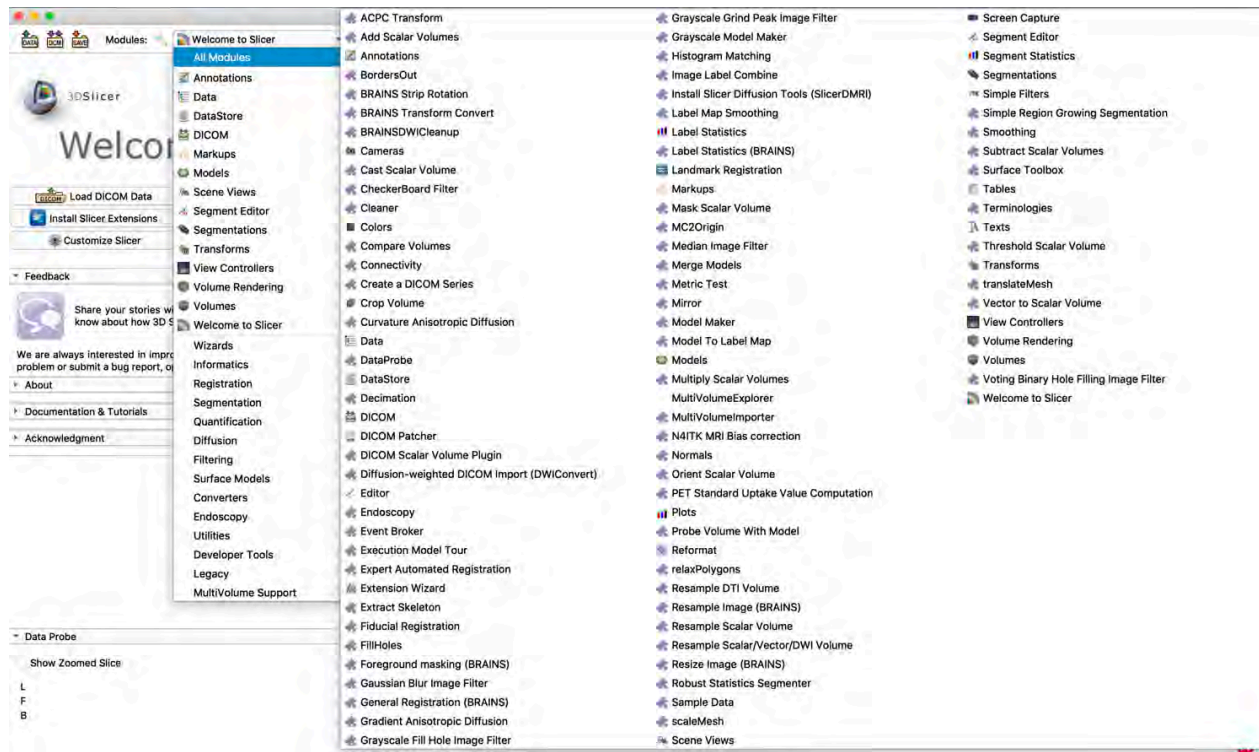
3D Slicer release version 5.0



SlicerDeveloperTutorial.zip

# Part 1

## Slicer Modules Overview



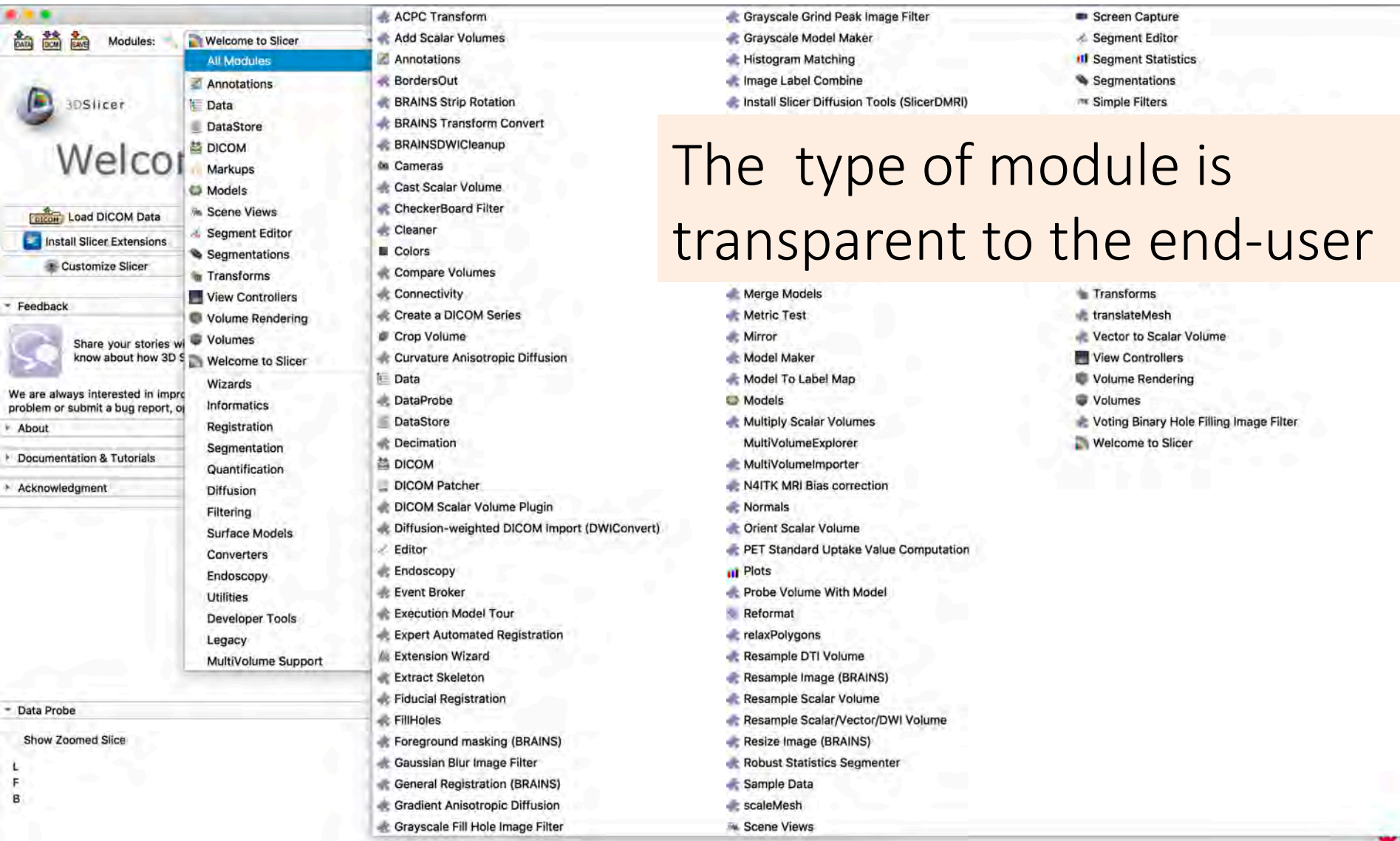
# Slicer Modules

Slicer5 supports three types of modules:

- **Command Line Interface (CLI):** standalone executable with limited input/output arguments
- **Loadable Modules (C++ Plugins):** optimized for heavy computation
- **Scripted Modules (Python):** recommended for fast prototyping and workflow development

Focus of this tutorial

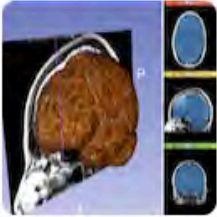


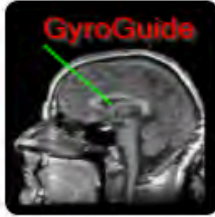




# Slicer5 Modules





# Slicer Extensions

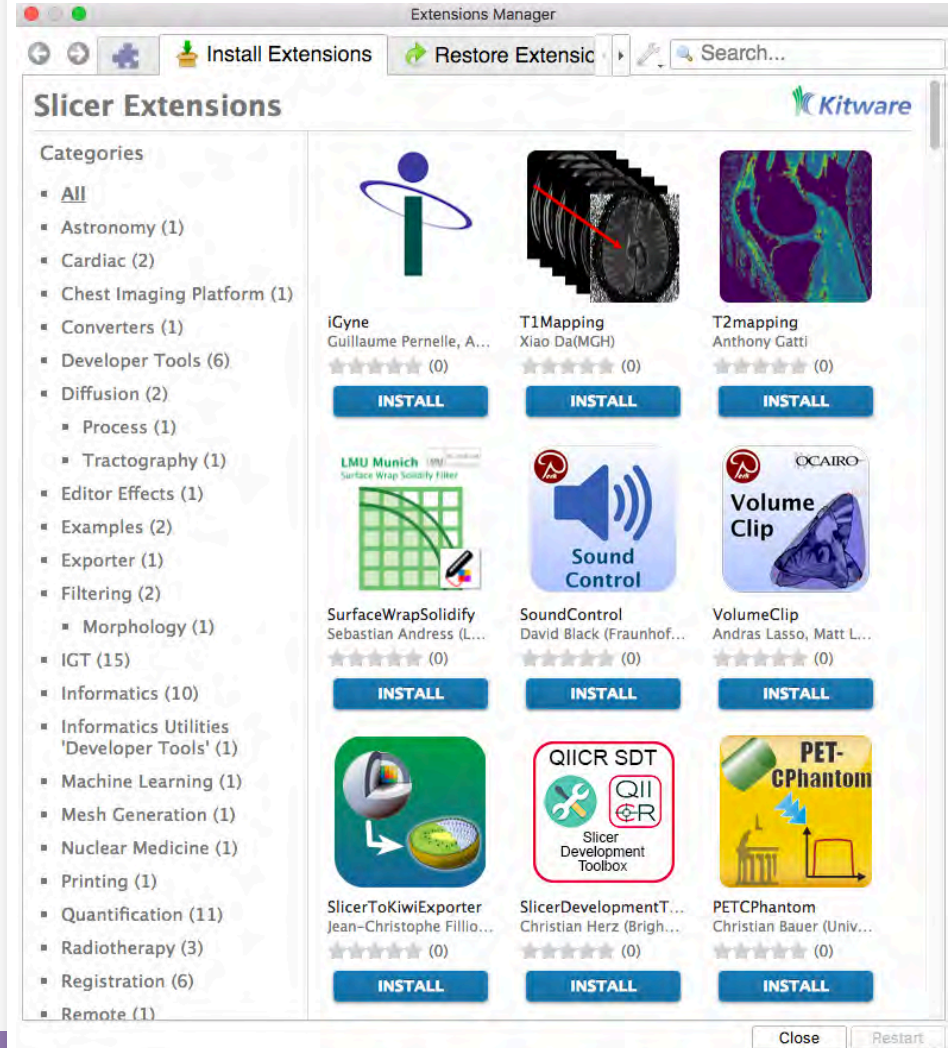
A Slicer Extension is a delivery package bundling together one or more Slicer modules

			
<b>SwissSkullStripper</b> Bill Lorensen (Noware...) ★★★★★ (0)	<b>PETTumorSegmenta...</b> Christian Bauer (Univ...) ★★★★★ (0)	<b>SlicerOpenIGTLink</b> Junichi Tokuda (SPL), ... ★★★★★ (0)	<b>GyroGuide</b> Ruifeng Chen, Luping... ★★★★★ (0)
<b>INSTALL</b>	<b>INSTALL</b>	<b>INSTALL</b>	<b>INSTALL</b>
			
<b>PET-IndiC</b> Ethan Ulrich (Universi...) ★★★★★ (0)	<b>FiducialsToModelDi...</b> Jesse Reynolds (Cante...) ★★★★★ (0)	<b>Slicer-Wasp</b> Thomas Lawson (MR...) ★★★★★ (0)	<b>ImageCompare</b> Paolo Zaffino (Magna ...) ★★★★★ (0)
<b>INSTALL</b>	<b>INSTALL</b>	<b>INSTALL</b>	<b>INSTALL</b>



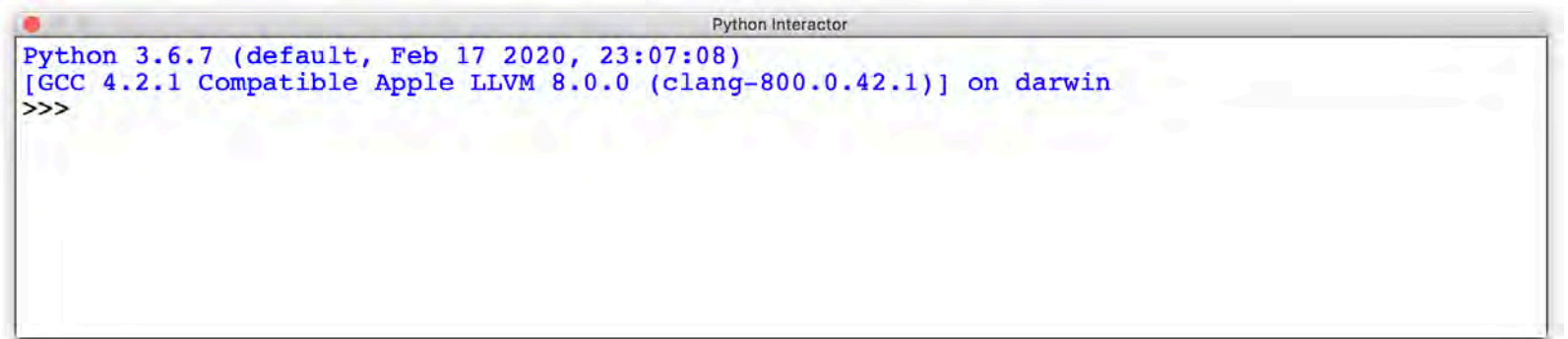
# Slicer Extension Manager

- The Slicer Extension Manager provides an 'App store' platform for the 3D Slicer ecosystem
- The Extension Manager enables an easy creation and installation of Slicer extensions
- Slicer release version 5 includes over 130 extensions



## Part 2

# Getting Familiar with the Python environment in 3D Slicer

A screenshot of a 'Python Interactor' window. The window has a title bar with standard macOS window controls (red, yellow, green buttons). The main content area displays the following text in a monospaced font: 'Python 3.6.7 (default, Feb 17 2020, 23:07:08)' on the first line, '[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.42.1)] on darwin' on the second line, and '>>>' on the third line. The background of the window is light gray with some faint, colorful, abstract patterns.

```
Python 3.6.7 (default, Feb 17 2020, 23:07:08)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.42.1)] on darwin
>>>
```

# Python in Slicer

Slicer5 works with **Python3** and a rich set of standard libraries



NumPy is the fundamental package for scientific computing with Python.



VTK is an open-source library for manipulating and displaying scientific data.



ITK is an open-source library for image analysis.



CTK is an open-source library for biomedical image computing.



PythonQt is a Python binding for Qt.



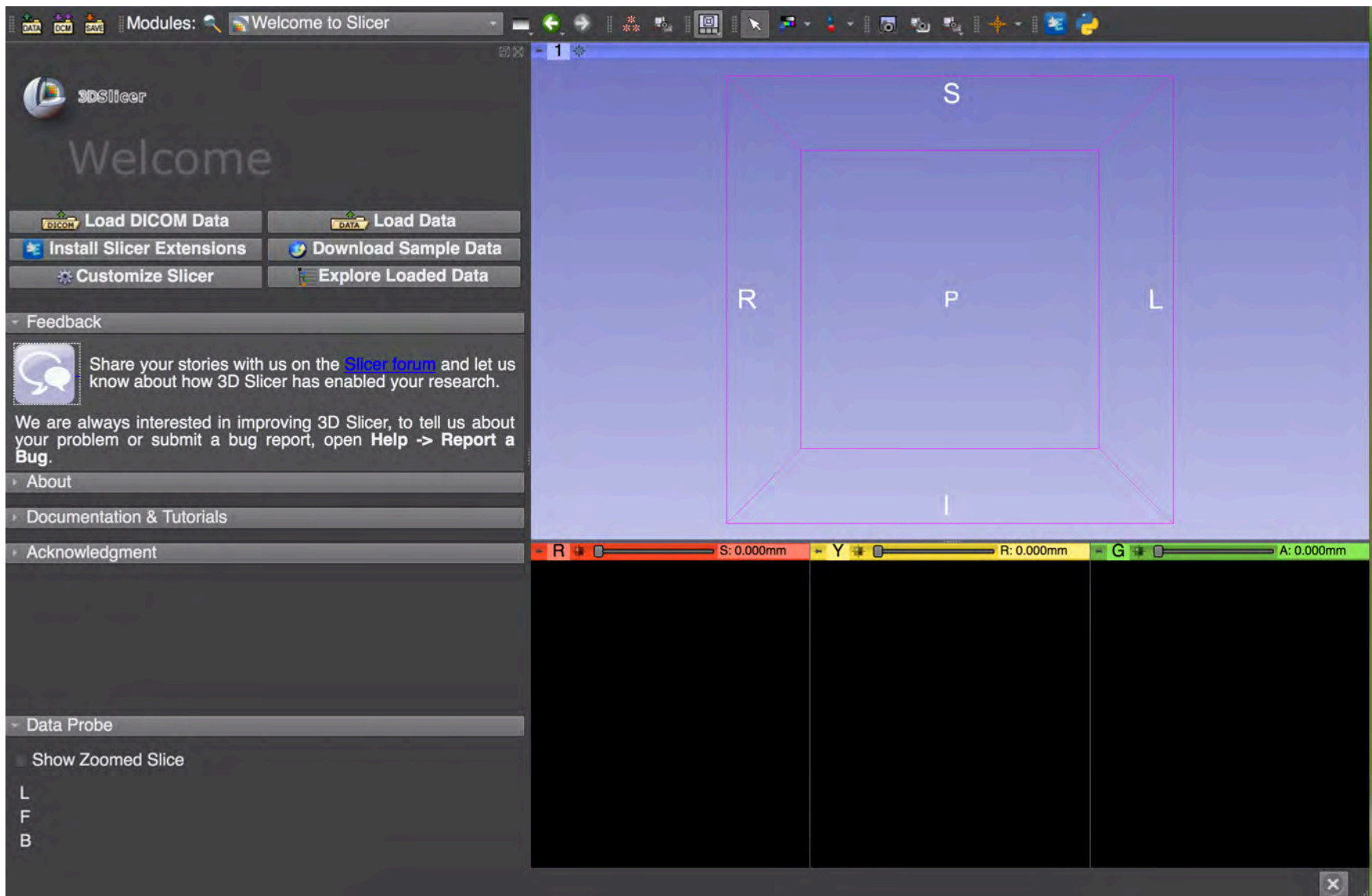
Qt is a cross-platform framework used as a graphical toolkit.

# Python in Slicer



The **Python Package index (PyPi)** gives access to over 200,000 additional Python packages (<http://pipy.org>)

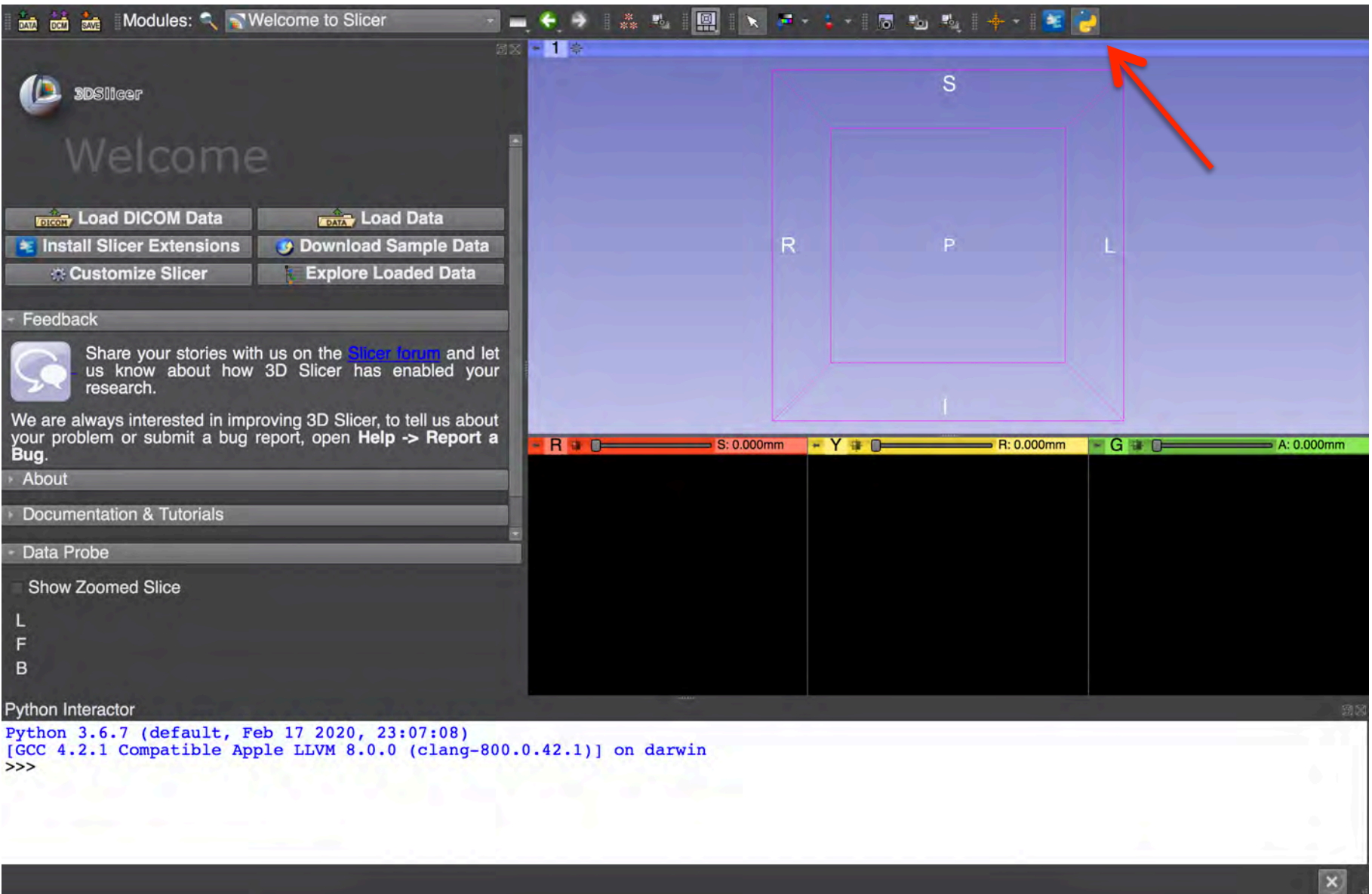
- The **pip install** command in Slicer enables developers to install most common scientific computing tools (e.g. TensorFlow, SciPy, PyTorch, Pandas, etc.)
- Slicer can be used as a **Jupyter notebook** kernel
- PyCharm and other Python development tools can be used with Slicer




Slicer release version 5 integrates  
Python3, VTK5 and ITK5

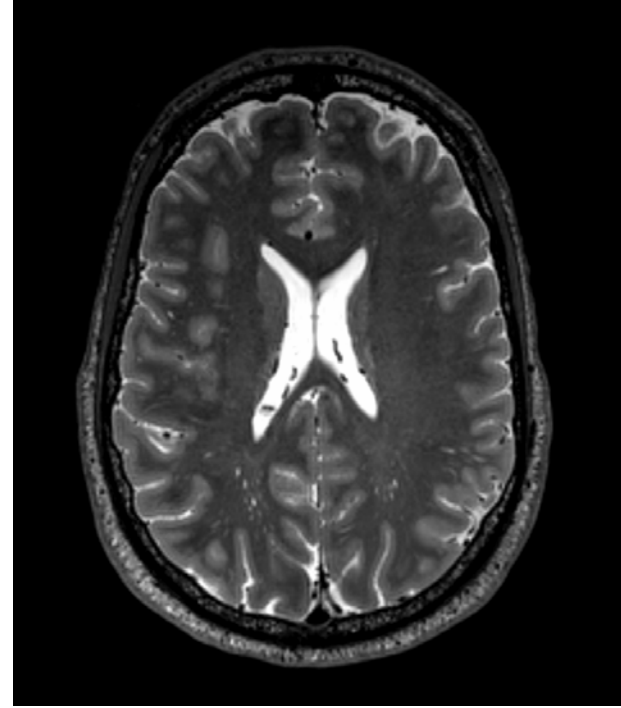
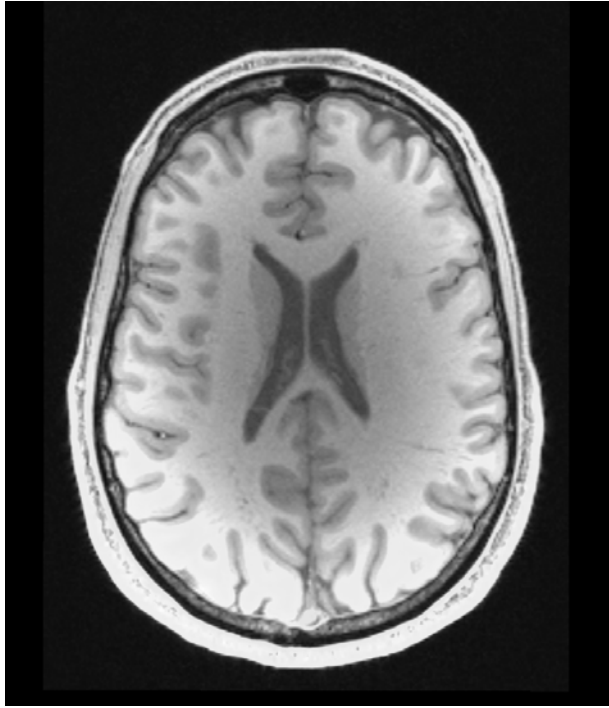
# The Python Console in Slicer

The Python Interactor is a Qt-based console that enables direct access to Slicer MRML Nodes, libraries (NumPy, VTK, ITK, CTK) and Qt.



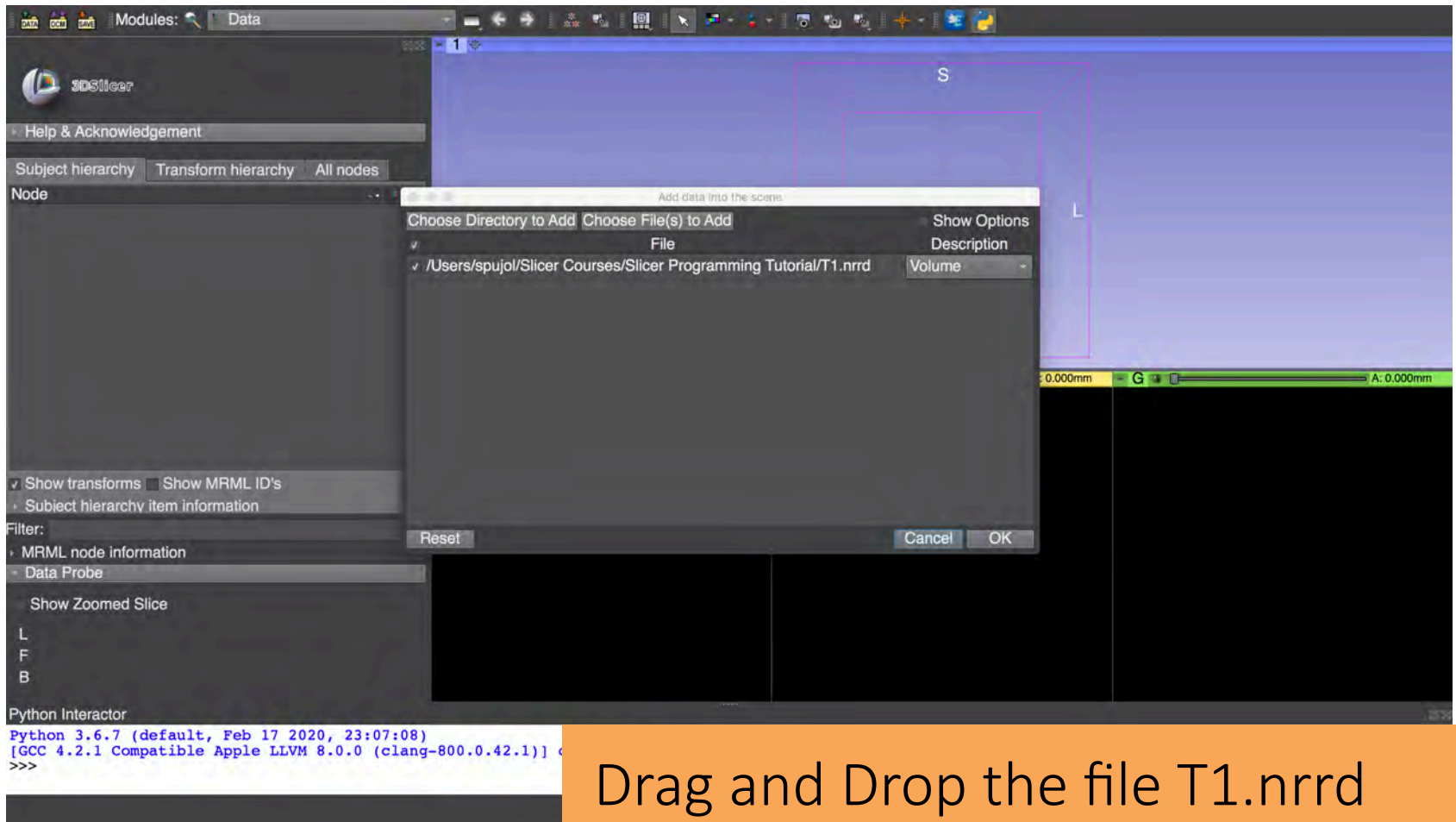
To access the Python Interactor, click on the Python icon  in the top bar menu of Slicer.





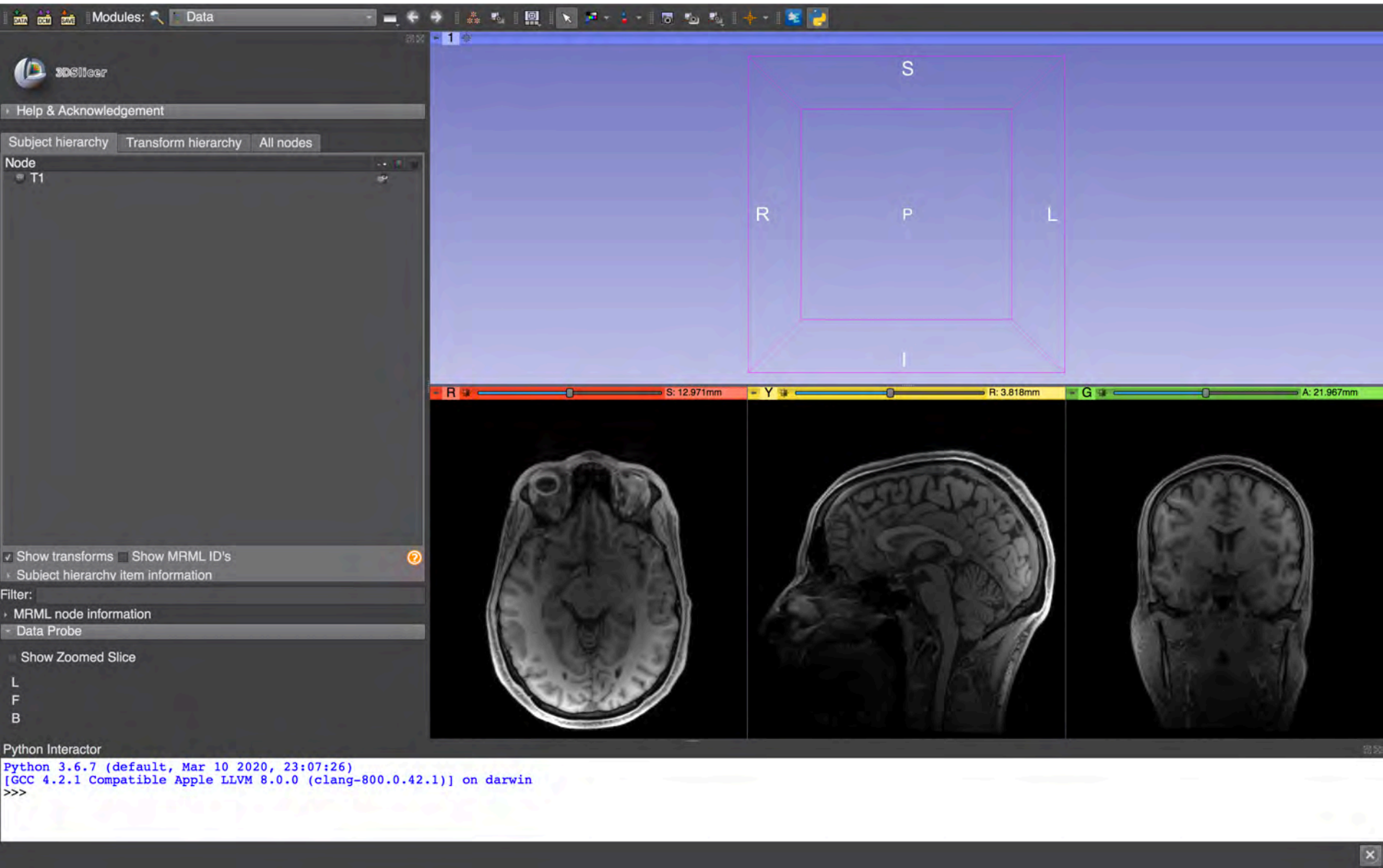
The Slicer Programming tutorial dataset includes a T1-weighted and a T2-weighted MRI scan of a healthy subject

# Tutorial dataset



Drag and Drop the file T1.nrrd  
Click on OK to load the file in Slicer

# Tutorial dataset



# Big Picture

- Slicer is free and open-source software
- There are thousands of sophisticated medical images available on the Internet that you could visualize and analyze with Slicer

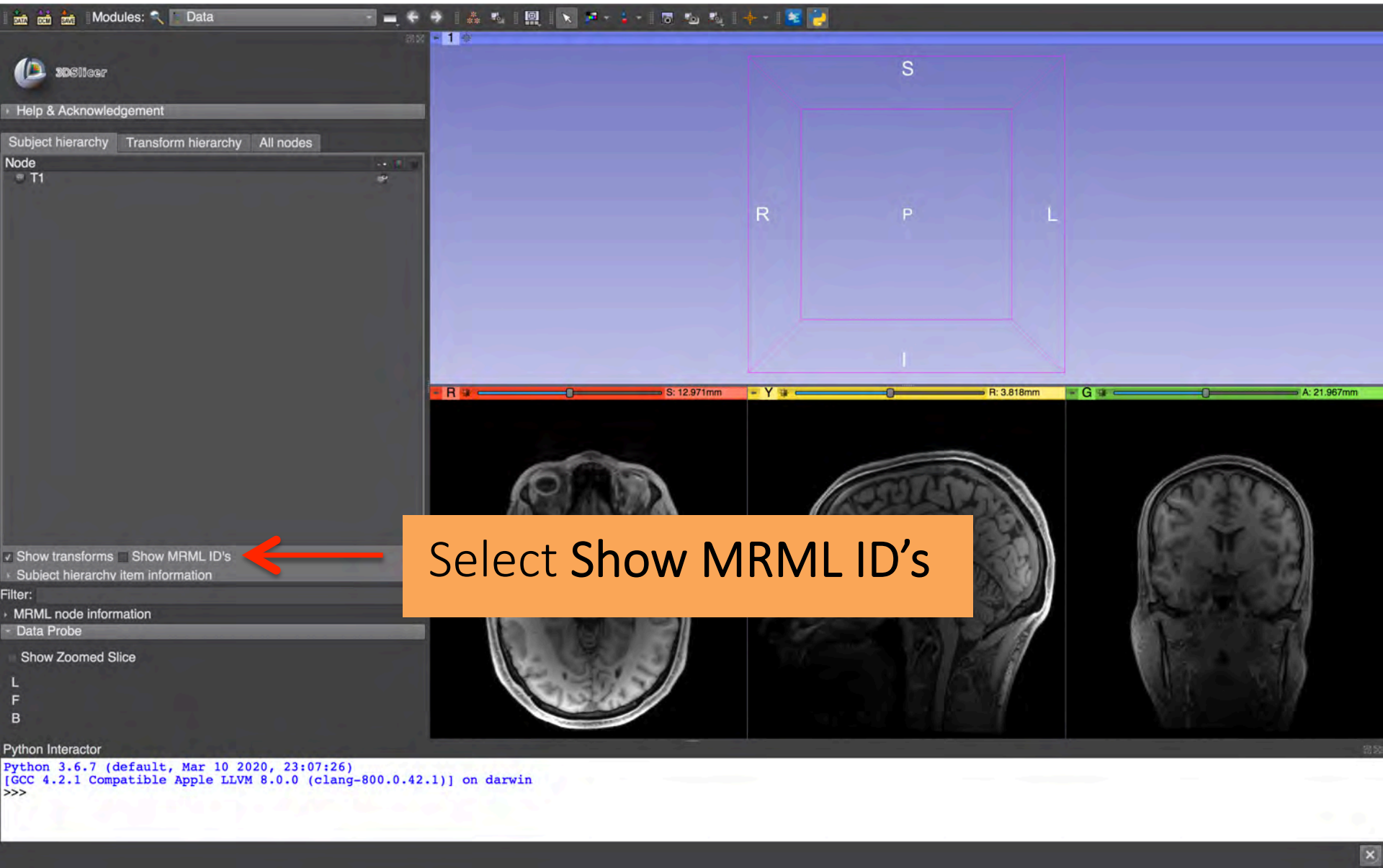
# Slicer Data Model

- The Slicer Data Model is based on the **Slicer Scene Data Structure**
- A Slicer scene is a collection of images, annotations, 3D models, spatial transforms, fiducials and cameras
- The Medical Reality Markup Language (**MRML**) is an XML-based language used to serialize the content of Slicer scene on disk (scene.mrml)
- Each element a scene is called a MRML node

# Slicer MRML Nodes: Basic Types

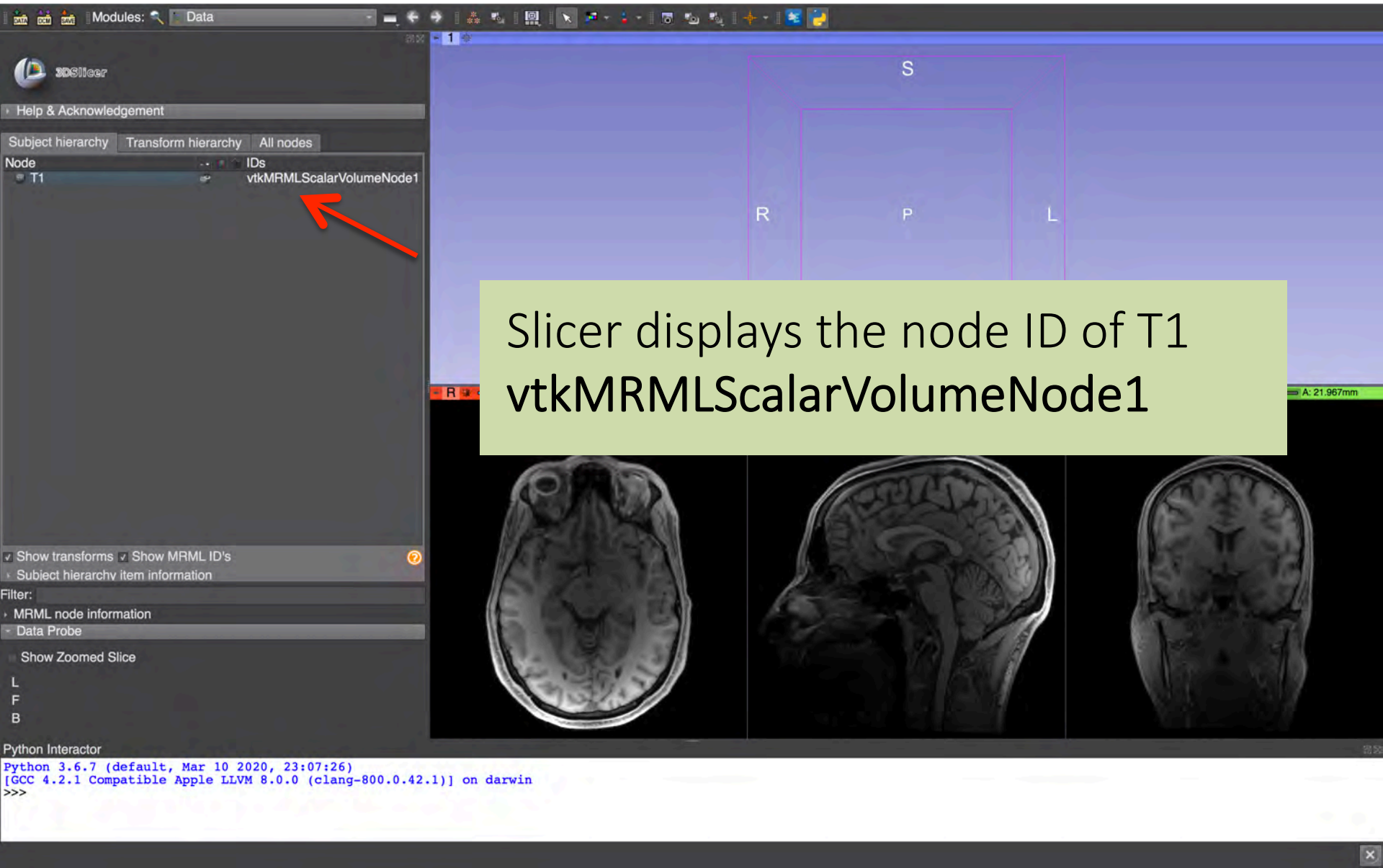
- **Data Node:** Stores the raw data
- **Display Node:** Describes how the data should be visualized
- **Storage Node:** Describes how the data should be stored on disk

# Tutorial dataset

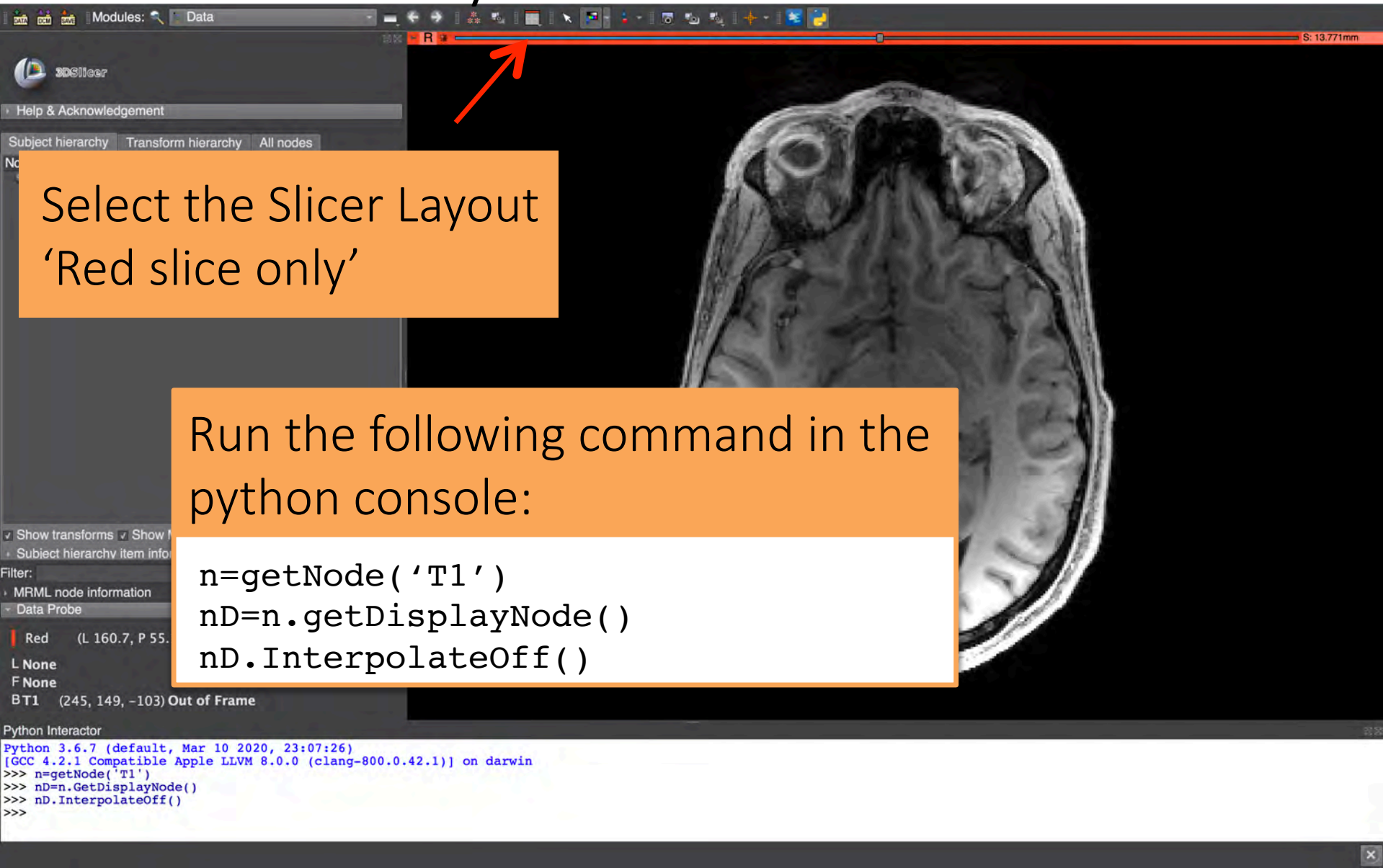




# Slicer Data Model



# Accessing MRML nodes from the Python interactor



Select the Slicer Layout 'Red slice only'

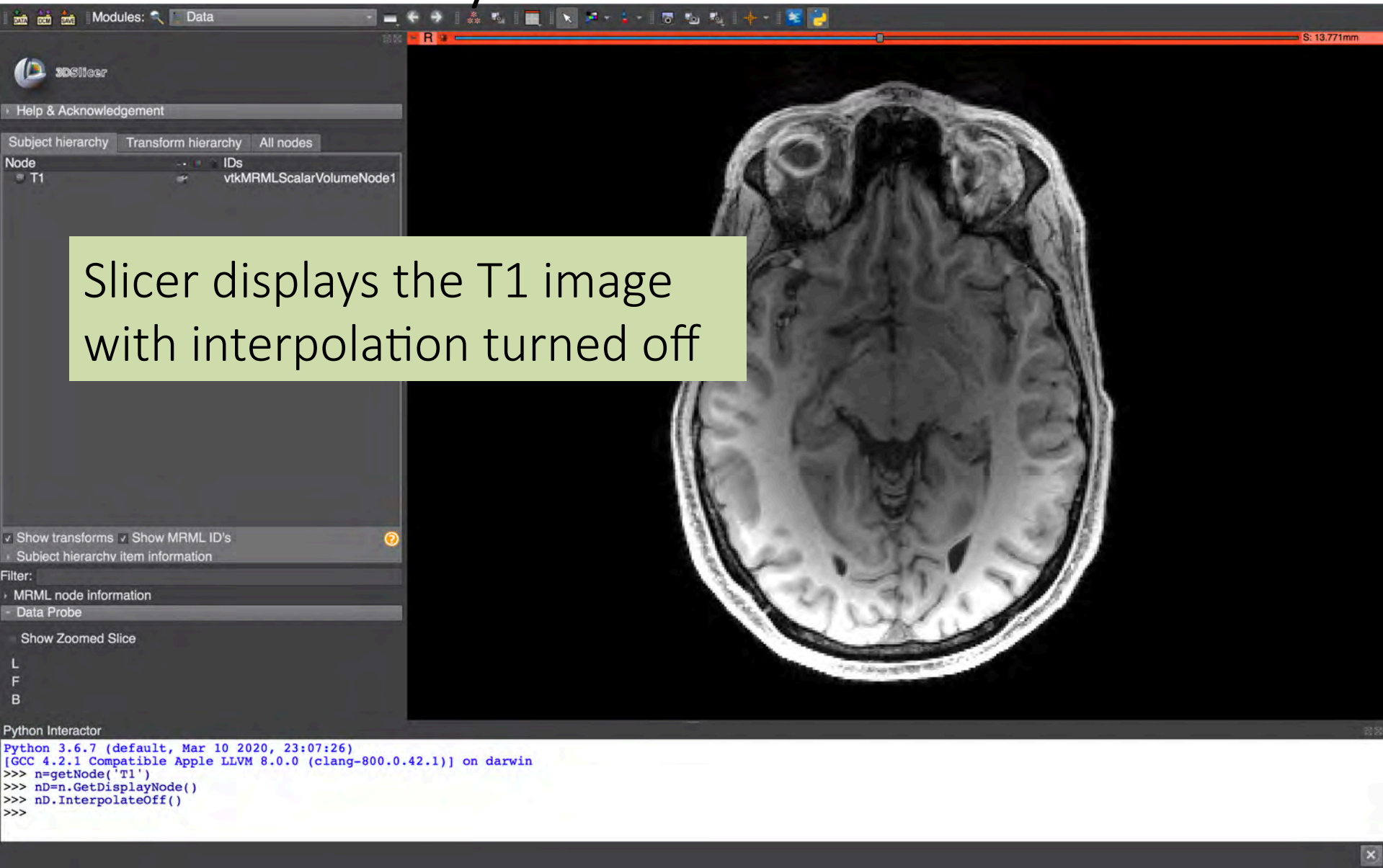
Run the following command in the python console:

```
n=getNode('T1')
nD=n.GetDisplayNode()
nD.InterpolateOff()
```

Python Interactor

```
Python 3.6.7 (default, Mar 10 2020, 23:07:26)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.42.1)] on darwin
>>> n=getNode('T1')
>>> nD=n.GetDisplayNode()
>>> nD.InterpolateOff()
>>>
```

# Accessing MRML nodes from the Python interactor

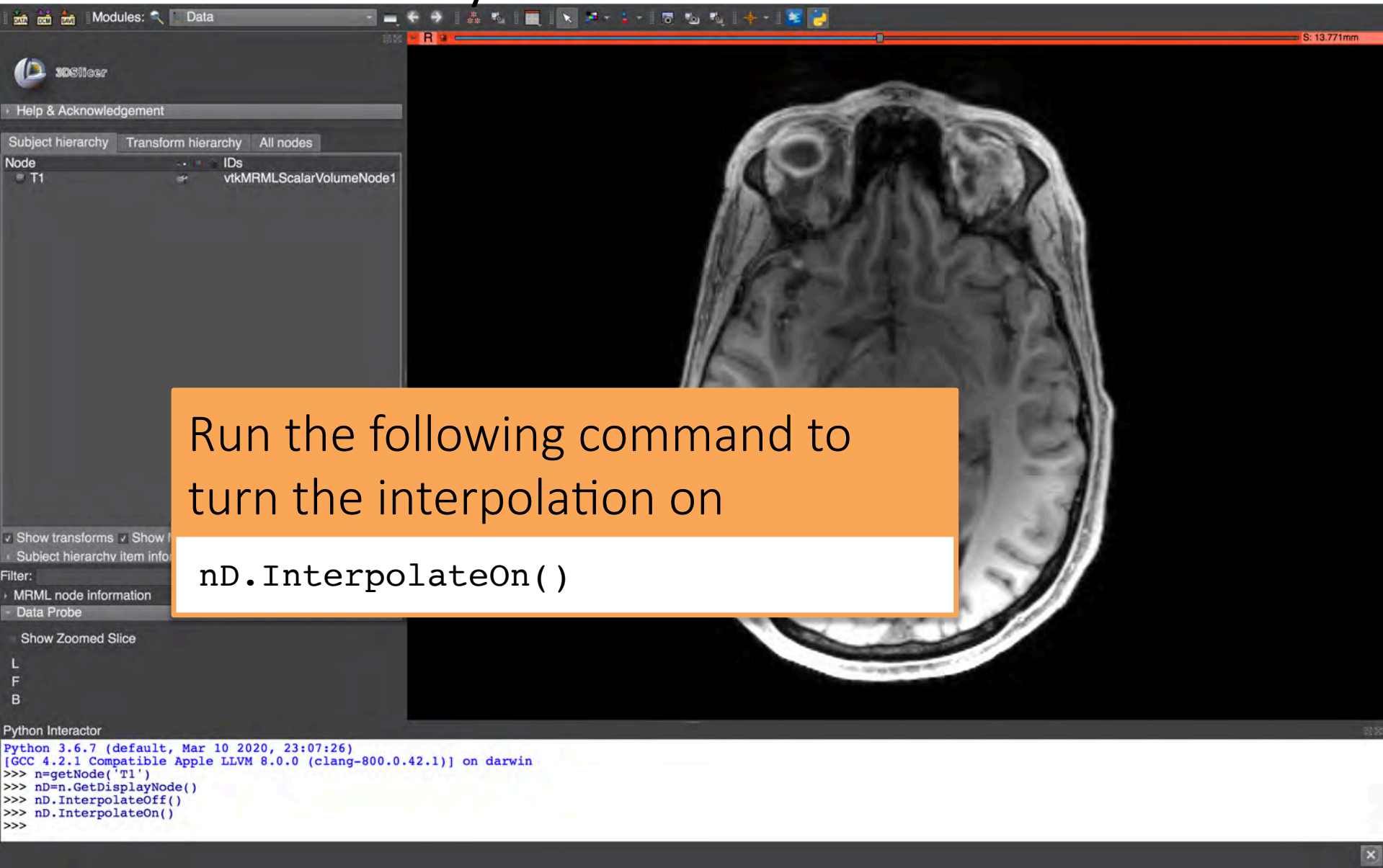


Slicer displays the T1 image with interpolation turned off

The screenshot shows the Slicer application interface. The main window displays a T1 brain MRI slice. The left sidebar contains the 'Subject hierarchy' panel, which shows the 'T1' node selected. Below this, the 'Python Interactor' panel shows the following code:

```
Python 3.6.7 (default, Mar 10 2020, 23:07:26)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.42.1)] on darwin
>>> n=getNode('T1')
>>> nD=n.GetDisplayNode()
>>> nD.InterpolateOff()
>>>
```

# Accessing MRML nodes from the Python interactor

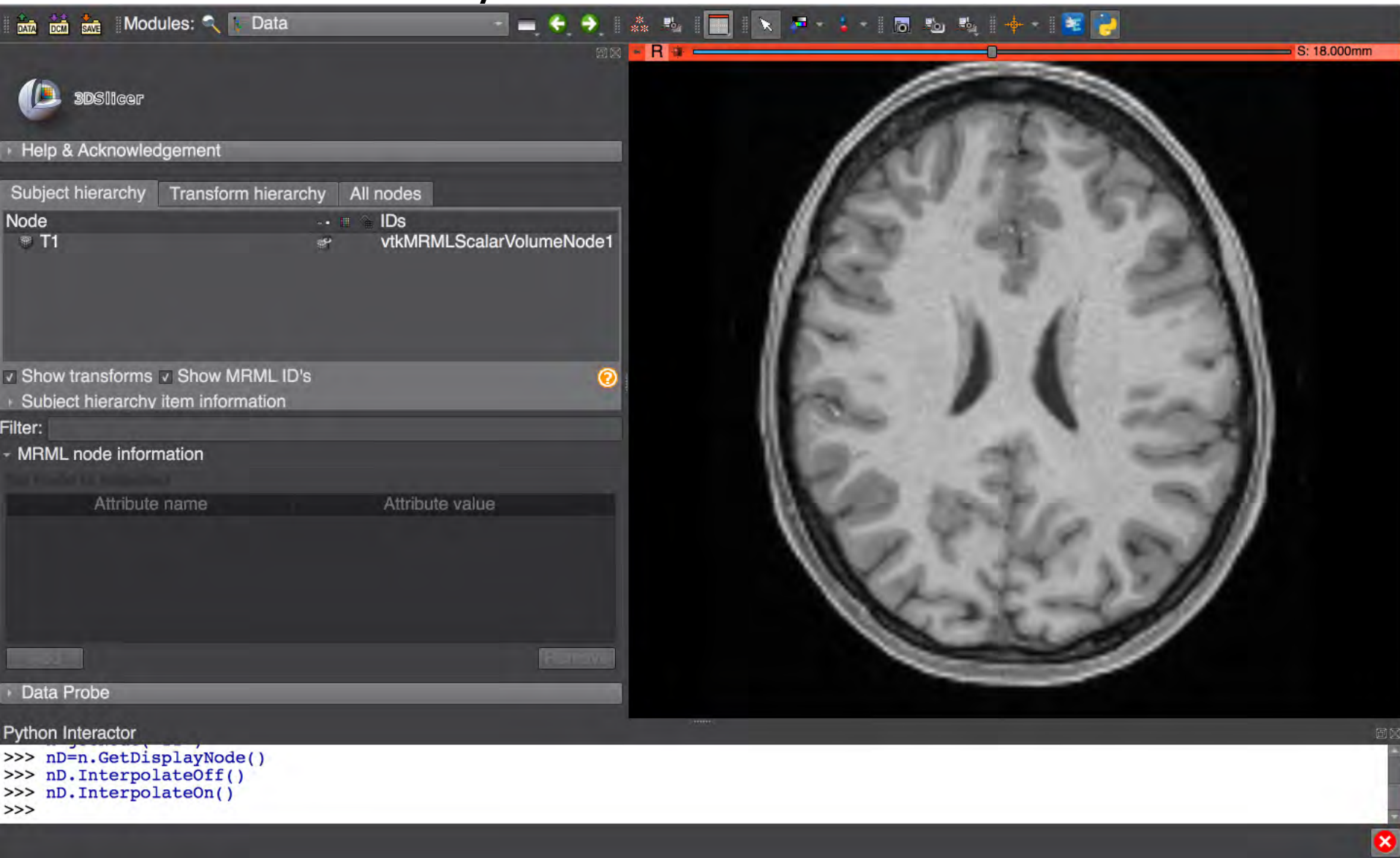


Run the following command to turn the interpolation on

```
nD.InterpolateOn( )
```

```
Python 3.6.7 (default, Mar 10 2020, 23:07:26)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.42.1)] on darwin
>>> n=getNode('T1')
>>> nD=n.GetDisplayNode()
>>> nD.InterpolateOff()
>>> nD.InterpolateOn()
>>>
```

# Accessing MRML nodes from the Python interactor



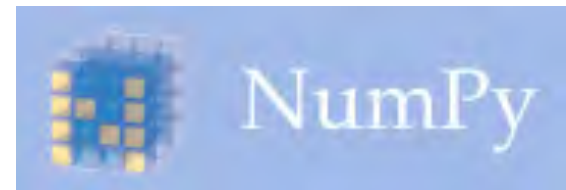
The screenshot displays the 3D Slicer software interface. The main window shows an axial brain MRI slice. The left sidebar contains the 'Data' module, which is currently active. The 'Data' module's 'Subject hierarchy' tab is selected, showing a tree view of the MRML nodes. The tree view shows a 'T1' node, which is a 'vtkMRMLScalarVolumeNode1'. The 'Show transforms' and 'Show MRML ID's' checkboxes are checked. Below the tree view, there is a 'Filter' field and a 'Data Probe' section. The 'Python Interactor' at the bottom shows the following code:

```
>>> nD=n.GetDisplayNode()  
>>> nD.InterpolateOff()  
>>> nD.InterpolateOn()  
>>>
```

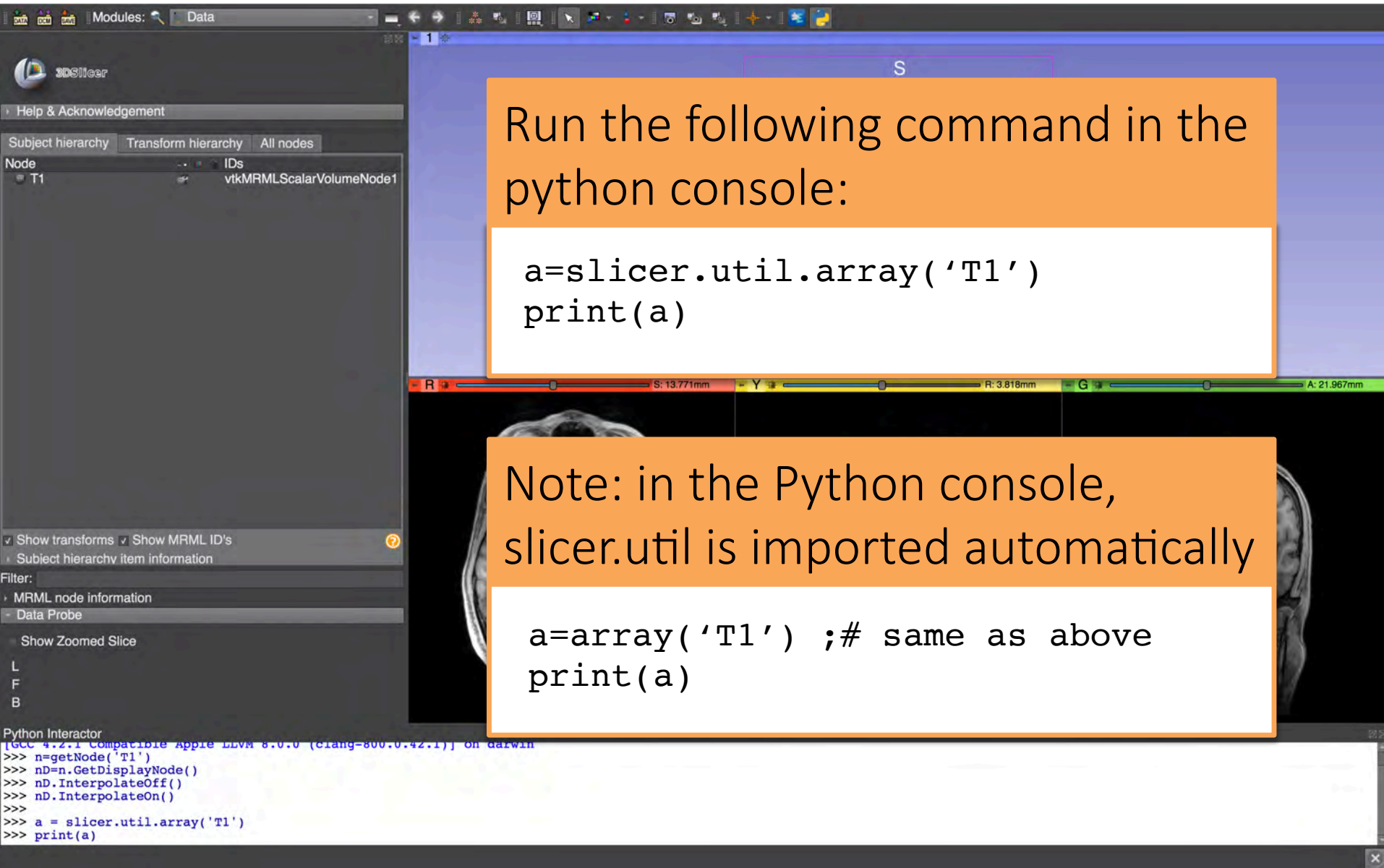


# Accessing voxels in a volume

- The `slicer.util` package gives access to volumes as NumPy multidimensional **arrays**
- Volumes can be modified using standard NumPy methods



# Accessing voxels in a volume



Run the following command in the python console:

```
a=slicer.util.array('T1')  
print(a)
```

Note: in the Python console, `slicer.util` is imported automatically

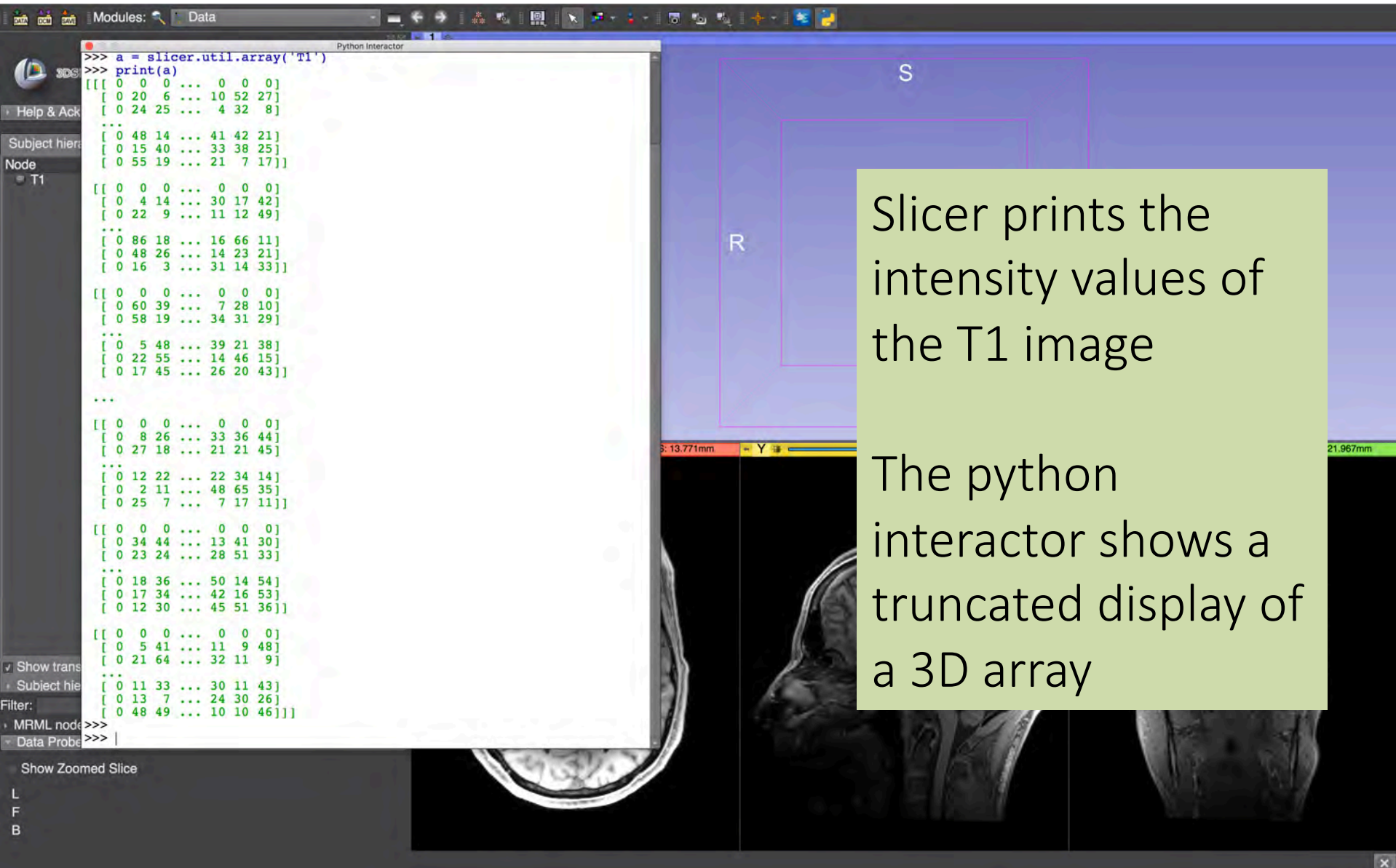
```
a=array('T1') ;# same as above  
print(a)
```

Python Interactor

```
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.42.1)] on darwin  
>>> n=getNode('T1')  
>>> nD=n.GetDisplayNode()  
>>> nD.InterpolateOff()  
>>> nD.InterpolateOn()  
>>>  
>>> a = slicer.util.array('T1')  
>>> print(a)
```



# Accessing voxels in a volume



The screenshot shows the Slicer software interface. On the left, a Python Interactor window displays a truncated 3D array of intensity values for a T1 image. The array is printed in a truncated format, showing only the first few rows and columns of each slice. The main window shows a 3D volume rendering of a brain scan, with a green rectangular box highlighting a region of interest. The box is labeled 'R' and 'S'.

```
>>> a = slicer.util.array('T1')
>>> print(a)
[[[ 0  0  0 ...  0  0  0]
 [ 0 20  6 ... 10 52 27]
 [ 0 24 25 ...  4 32  8]
 ...
 [ 0 48 14 ... 41 42 21]
 [ 0 15 40 ... 33 38 25]
 [ 0 55 19 ... 21  7 17]]]

[[[ 0  0  0 ...  0  0  0]
 [ 0  4 14 ... 30 17 42]
 [ 0 22  9 ... 11 12 49]
 ...
 [ 0 86 18 ... 16 66 11]
 [ 0 48 26 ... 14 23 21]
 [ 0 16  3 ... 31 14 33]]]

[[[ 0  0  0 ...  0  0  0]
 [ 0 60 39 ...  7 28 10]
 [ 0 58 19 ... 34 31 29]
 ...
 [ 0  5 48 ... 39 21 38]
 [ 0 22 55 ... 14 46 15]
 [ 0 17 45 ... 26 20 43]]]

...

[[[ 0  0  0 ...  0  0  0]
 [ 0  8 26 ... 33 36 44]
 [ 0 27 18 ... 21 21 45]
 ...
 [ 0 12 22 ... 22 34 14]
 [ 0  2 11 ... 48 65 35]
 [ 0 25  7 ...  7 17 11]]]

[[[ 0  0  0 ...  0  0  0]
 [ 0 34 44 ... 13 41 30]
 [ 0 23 24 ... 28 51 33]
 ...
 [ 0 18 36 ... 50 14 54]
 [ 0 17 34 ... 42 16 53]
 [ 0 12 30 ... 45 51 36]]]

[[[ 0  0  0 ...  0  0  0]
 [ 0  5 41 ... 11  9 48]
 [ 0 21 64 ... 32 11  9]
 ...
 [ 0 11 33 ... 30 11 43]
 [ 0 13  7 ... 24 30 26]
 [ 0 48 49 ... 10 10 46]]]
```

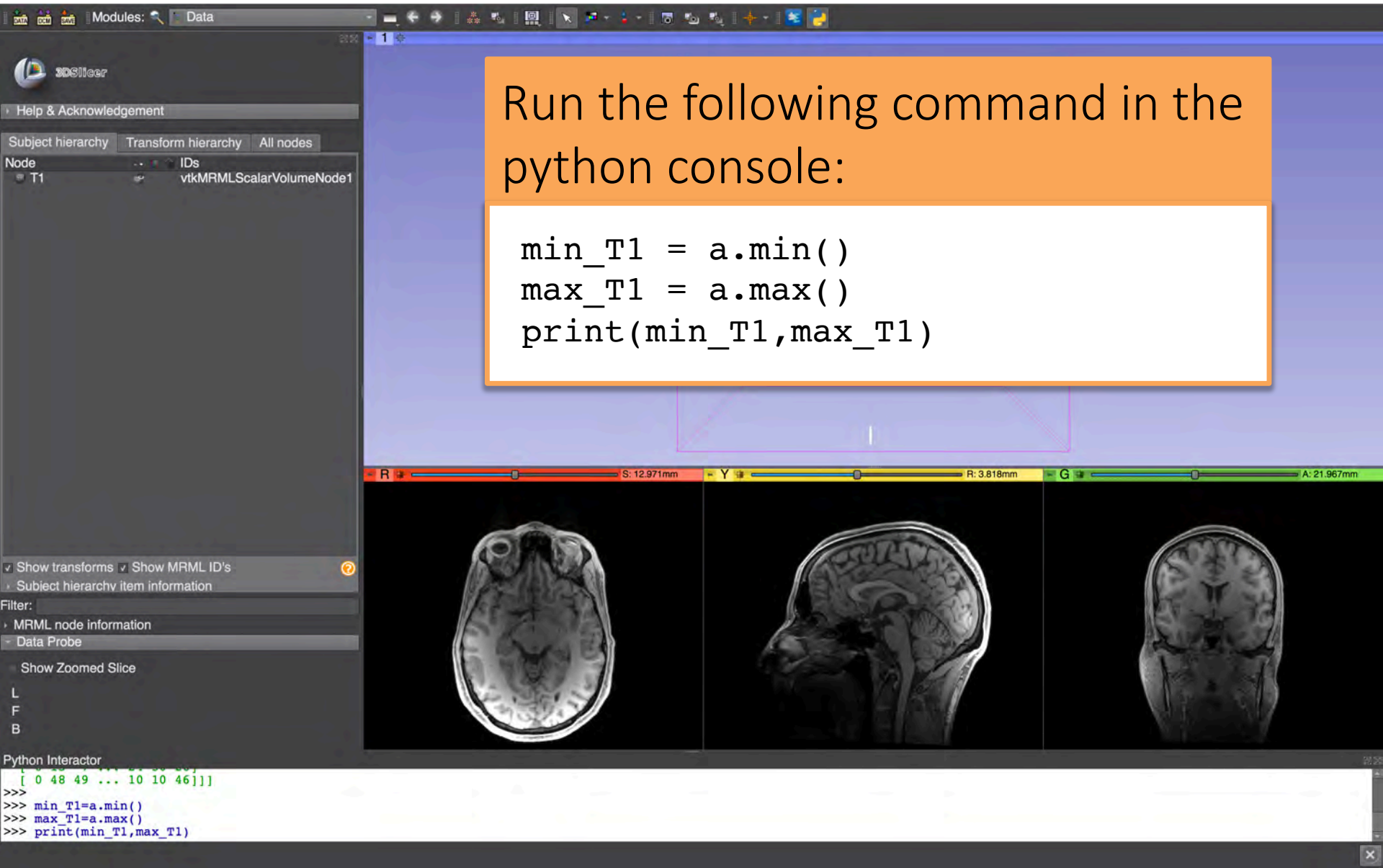
Slicer prints the intensity values of the T1 image

The python interactor shows a truncated display of a 3D array

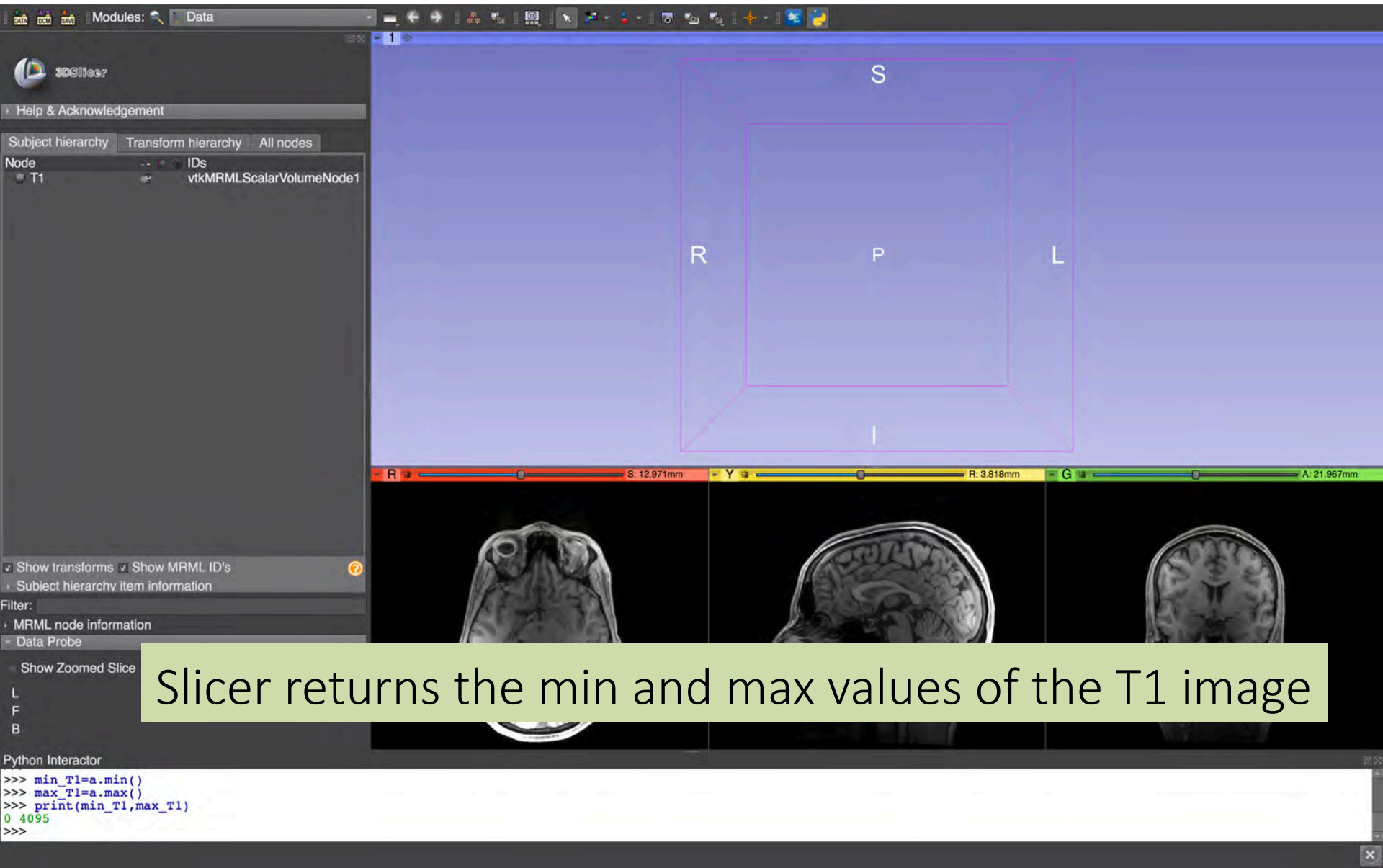
# Accessing voxels in a volume

Run the following command in the python console:

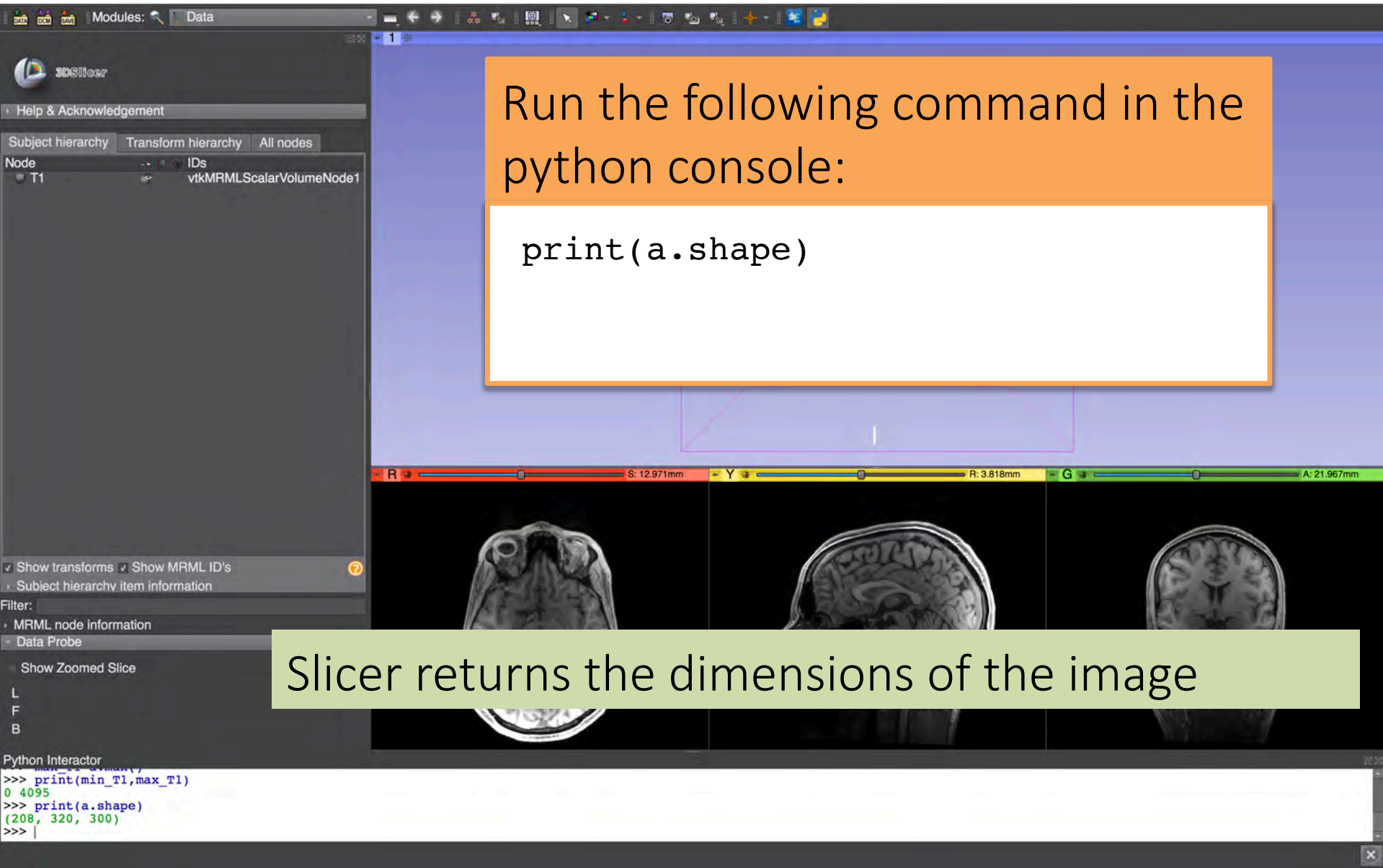
```
min_T1 = a.min()  
max_T1 = a.max()  
print(min_T1,max_T1)
```



# Accessing voxels in a volume



# Modifying voxels in a volume



The screenshot shows the 3D Slicer software interface. On the left, the 'Subject hierarchy' panel shows a tree with 'Node' and 'T1' (vtkMRMLScalarVolumeNode1). Below it, the 'Python Interactor' console shows the following code and output:

```
>>> print(min_T1,max_T1)
0 4095
>>> print(a.shape)
(208, 320, 300)
>>>
```

In the center, an orange text box contains the instruction: 'Run the following command in the python console:'. Below this, a white text box contains the code: `print(a.shape)`.

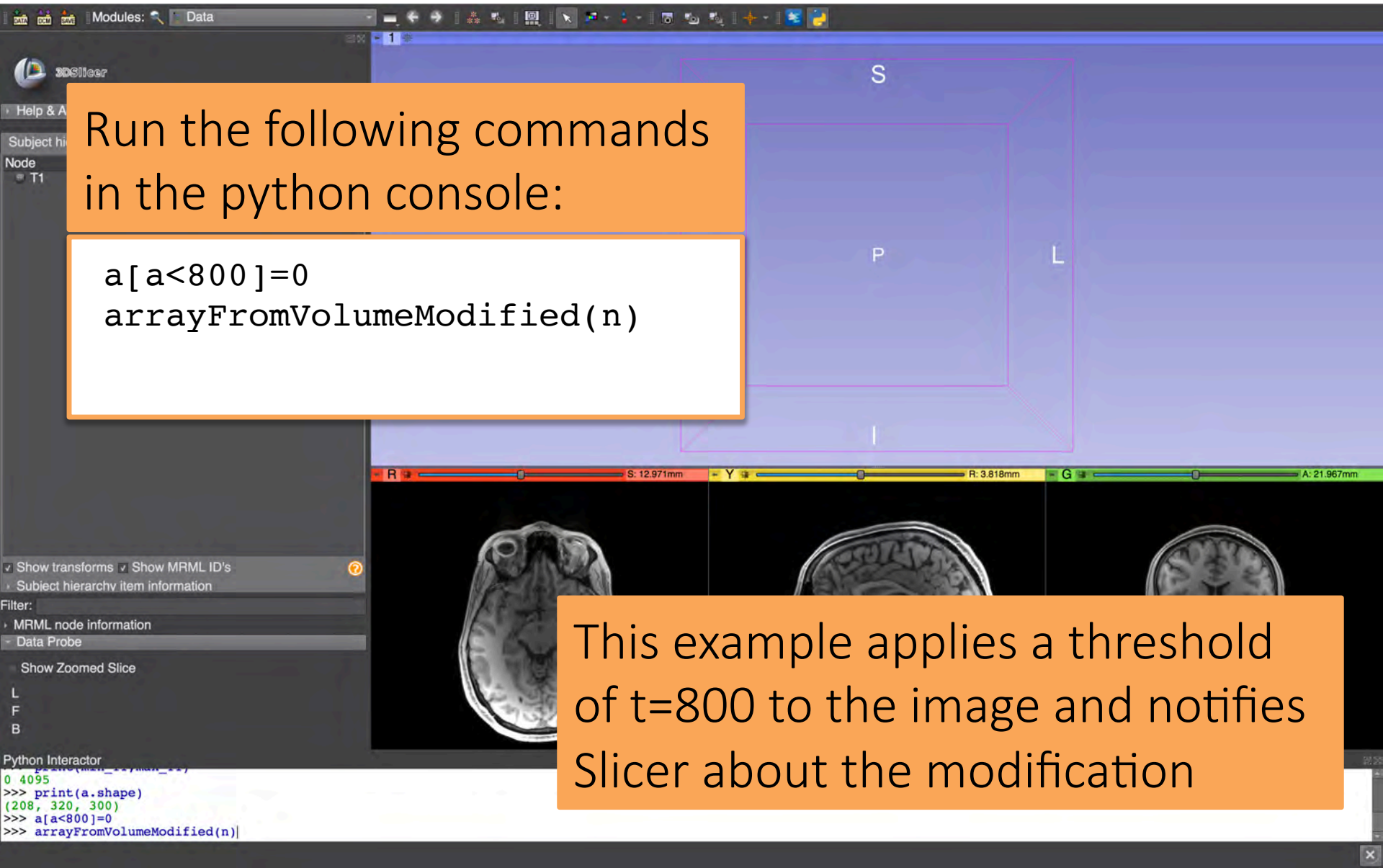
At the bottom, a green text box contains the text: 'Slicer returns the dimensions of the image'. The background shows a 3D view of a brain MRI volume with a purple bounding box and three orthogonal slices (axial, sagittal, and coronal) at the bottom.

# Modifying voxels in a volume

Run the following commands in the python console:

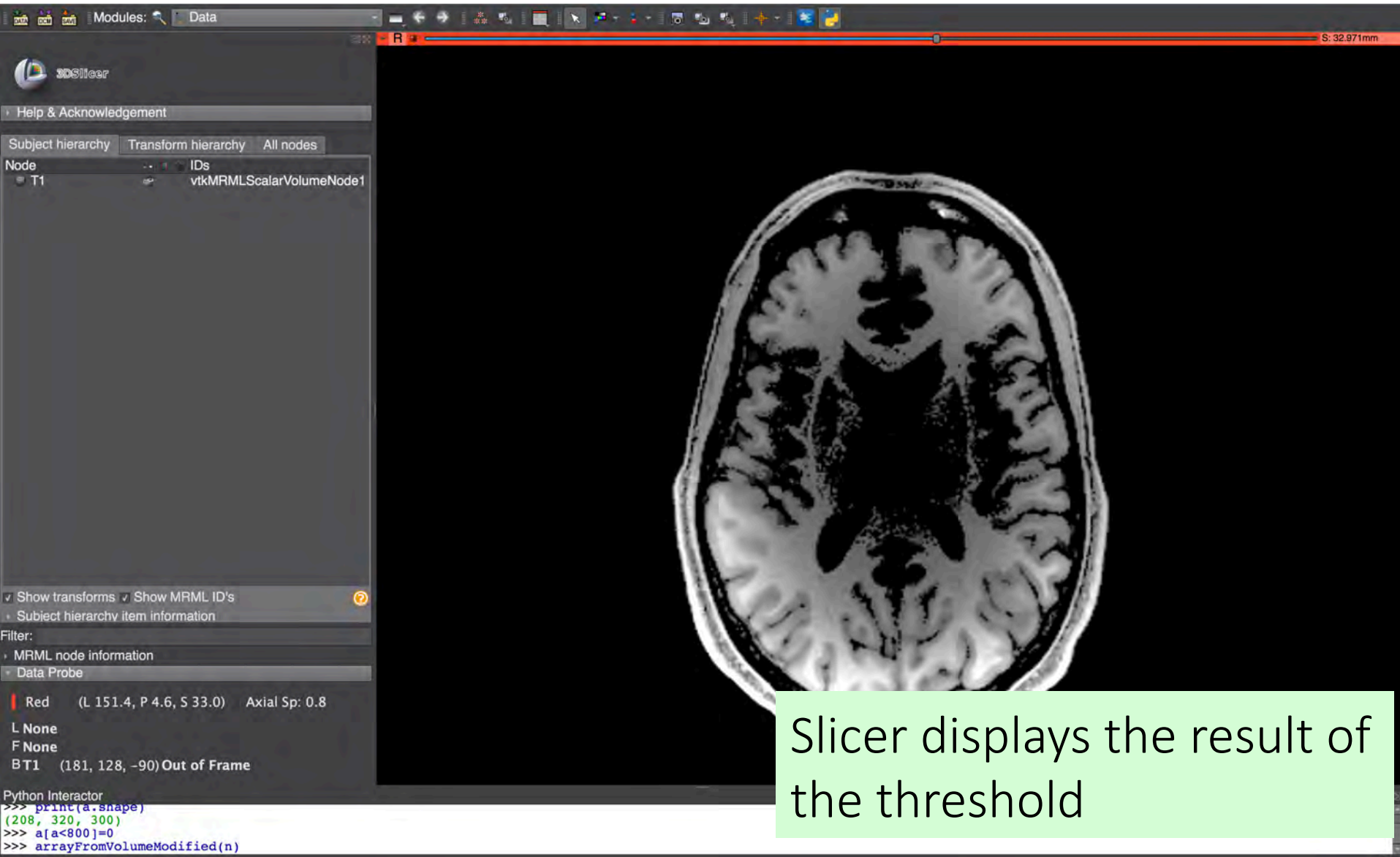
```
a[a<800]=0  
arrayFromVolumeModified(n)
```

This example applies a threshold of  $t=800$  to the image and notifies Slicer about the modification





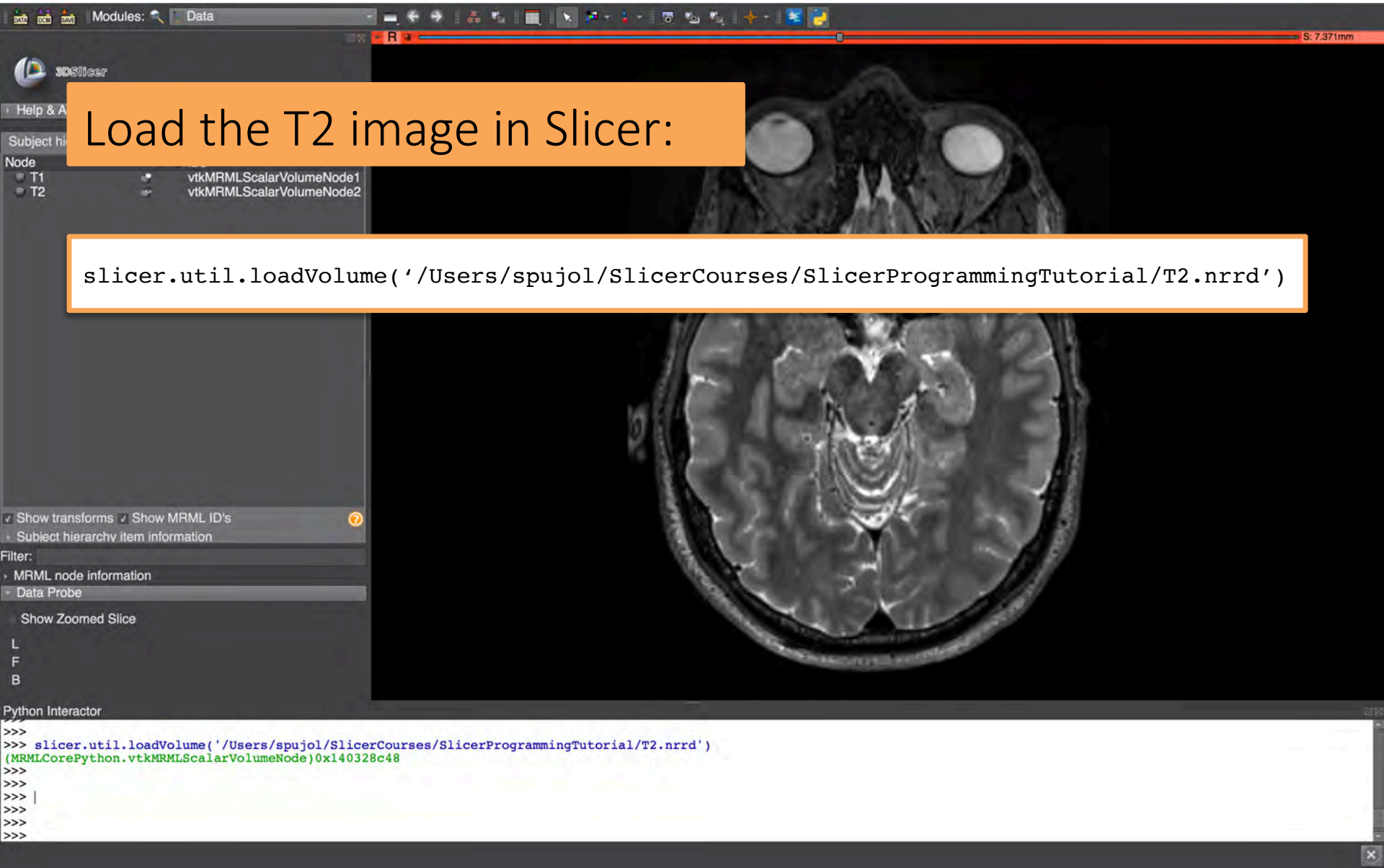
# Modifying voxels in a volume



# Loading the T2 volume

Load the T2 image in Slicer:

```
slicer.util.loadVolume('/Users/spujol/SlicerCourses/SlicerProgrammingTutorial/T2.nrrd')
```

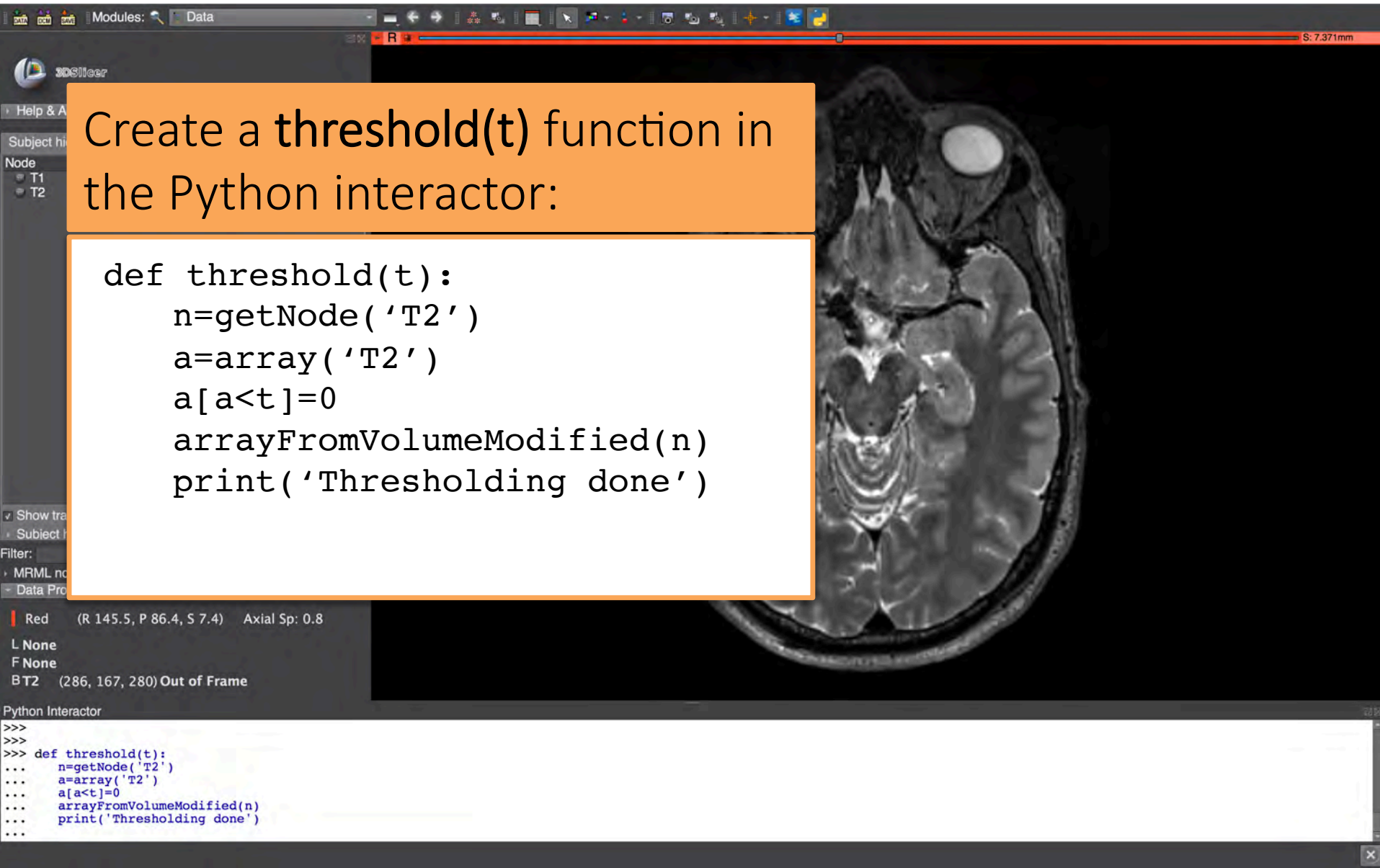




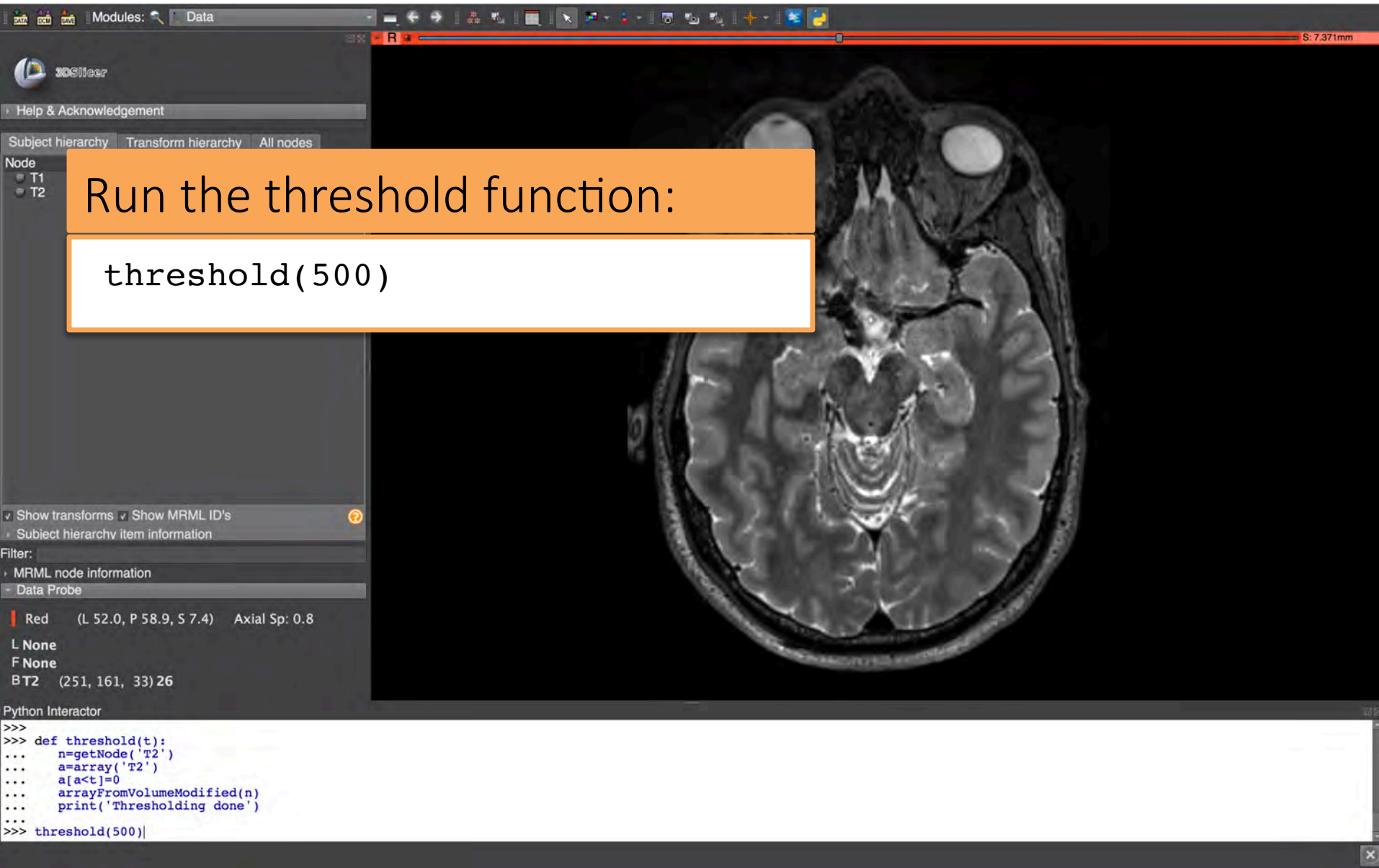
# Python function: threshold

Create a **threshold(t)** function in the Python interactor:

```
def threshold(t):  
    n=getNode('T2')  
    a=array('T2')  
    a[a<t]=0  
    arrayFromVolumeModified(n)  
    print('Thresholding done')
```



# Python function: threshold



Run the threshold function:

```
threshold(500)
```

Python Interactor

```
>>>
>>> def threshold(t):
...     n=getNode('T2')
...     a=array('T2')
...     a[a<t]=0
...     arrayFromVolumeModified(n)
...     print('Thresholding done')
...
>>> threshold(500)|
```

# Python function: threshold

The screenshot displays the 3D Slicer software interface. The main window shows an axial brain MRI slice. The left sidebar contains a 'Subject hierarchy' panel with 'T1' and 'T2' nodes, and a 'Python Interactor' panel at the bottom with a script for thresholding. A green text box on the right contains the instruction: 'Scroll through the slices to display the output of the threshold function'.

Python Interactor

```
... n=getNode('T2')
... a=array('T2')
... a[a<t]=0
... arrayFromVolumeModified(n)
... print('Thresholding done')
...
>>> threshold(500)
Thresholding done
>>>
```

# Big Picture

- Slicer provides easy access to analyze and modify complex data types
- Slicer is compatible with a wide range of Python scientific computing packages
- Slicer is a research environment for performing medical imaging experiments

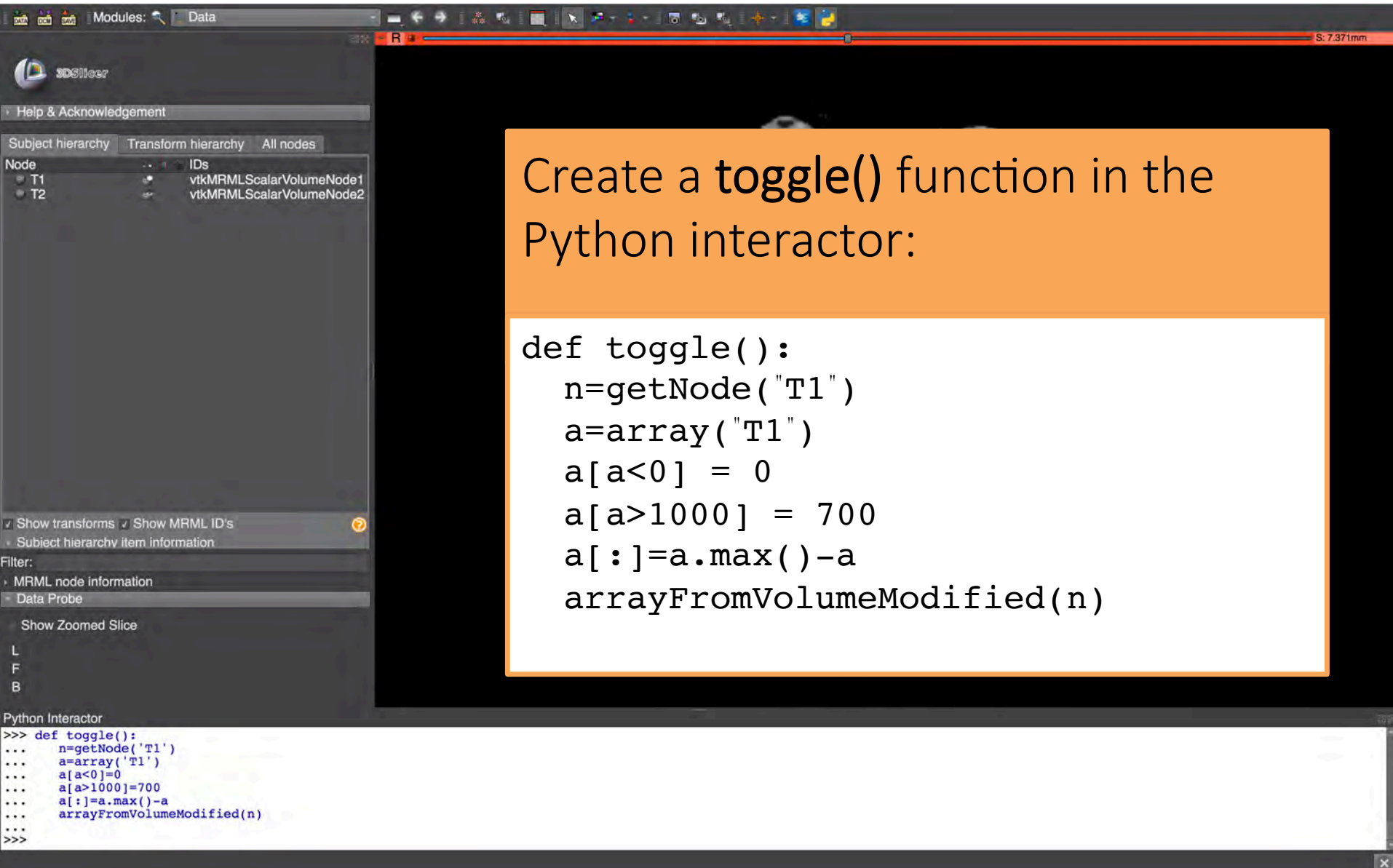
## Part 3

Getting familiar with Qt in Slicer

# Qt & PythonQt

- **Qt** is the main tool in Slicer to create widgets, dialogs, text entries, etc.
- **PythonQt** exposes most Qt functionalities and is accessible through the Python interactor in Slicer
- User interfaces can be created on the fly for rapid prototyping and debugging

# Python function: toggle

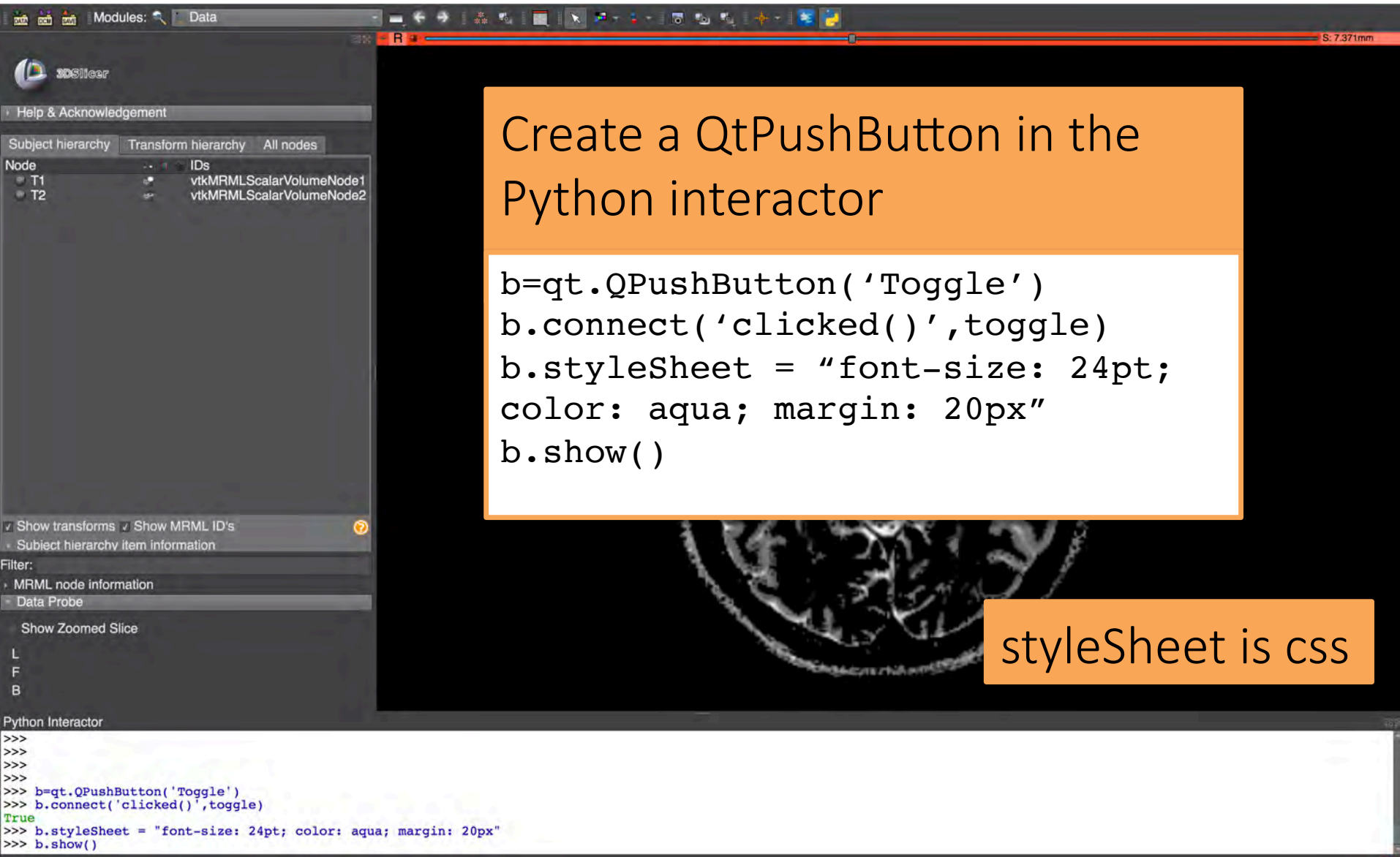


The screenshot shows the 3D Slicer software interface. On the left, the 'Subject hierarchy' panel lists two nodes: 'T1' and 'T2'. The 'T1' node is selected, and its ID is 'vtkMRMLScalarVolumeNode1'. The 'T2' node has the ID 'vtkMRMLScalarVolumeNode2'. Below the hierarchy, there are checkboxes for 'Show transforms', 'Show MRML ID's', and 'Subject hierarchy item information'. The 'Python Interactor' window at the bottom shows the following code:

```
>>> def toggle():
...     n=getNode('T1')
...     a=array('T1')
...     a[a<0]=0
...     a[a>1000]=700
...     a[:]=a.max()-a
...     arrayFromVolumeModified(n)
...
>>>
```

On the right, an orange box contains the text: 'Create a toggle() function in the Python interactor:'.

# Creating a Qt Push Button



The screenshot shows the 3D Slicer software interface. On the left, the 'Subject hierarchy' panel lists nodes T1 and T2, which are identified as 'vtkMRMLScalarVolumeNode1' and 'vtkMRMLScalarVolumeNode2'. Below this, the 'Filter' section shows 'MRML node information' and 'Data Probe'. The 'Python Interactor' at the bottom contains the following code:

```
>>>
>>>
>>>
>>> b=qt.QPushButton('Toggle')
>>> b.connect('clicked()',toggle)
True
>>> b.styleSheet = "font-size: 24pt; color: aqua; margin: 20px"
>>> b.show()
```

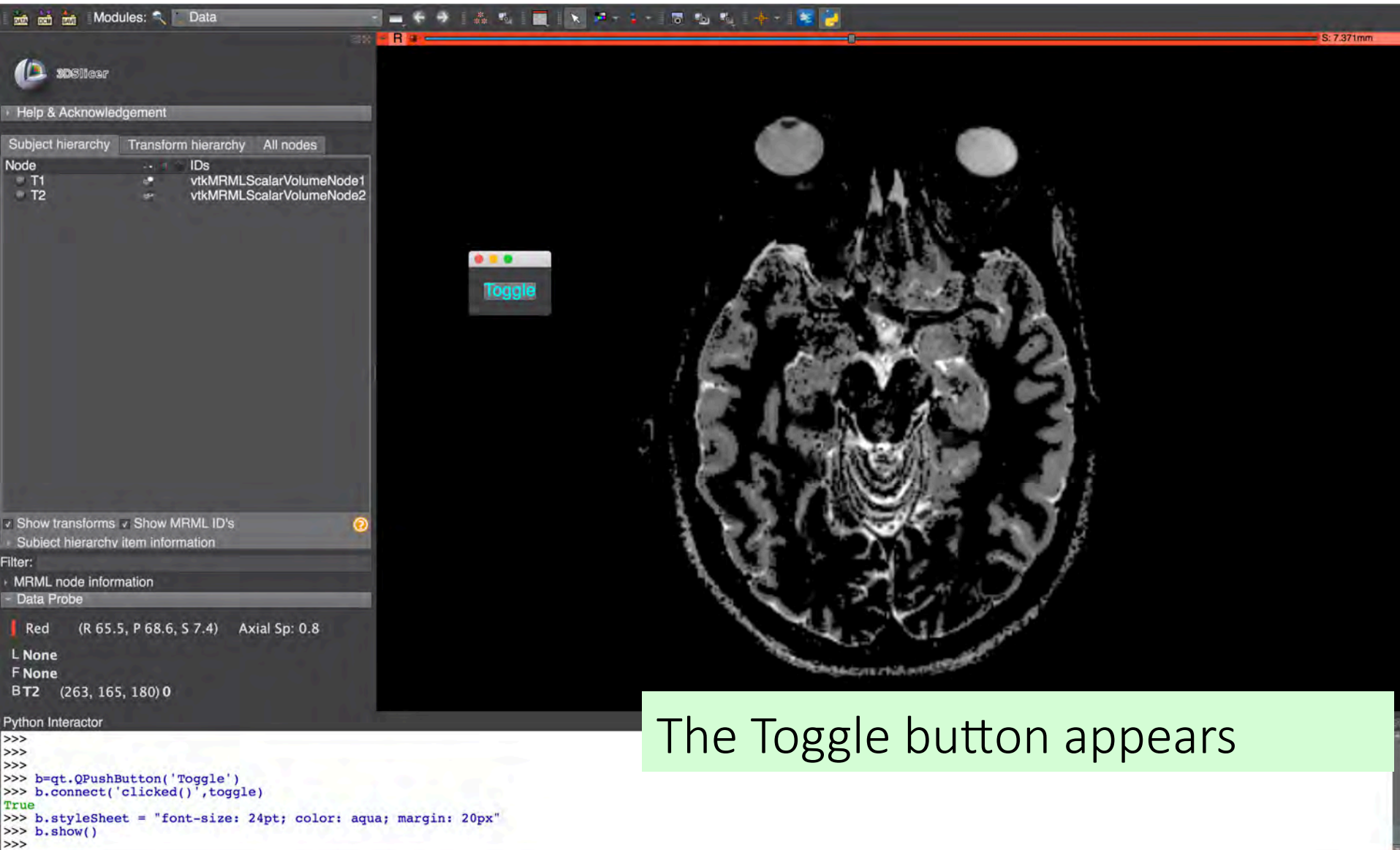
In the center, an orange box contains the text: "Create a QPushButton in the Python interactor".

Below the orange box, a white box contains the same Python code as shown in the Python Interactor.

At the bottom right, another orange box contains the text: "styleSheet is css".

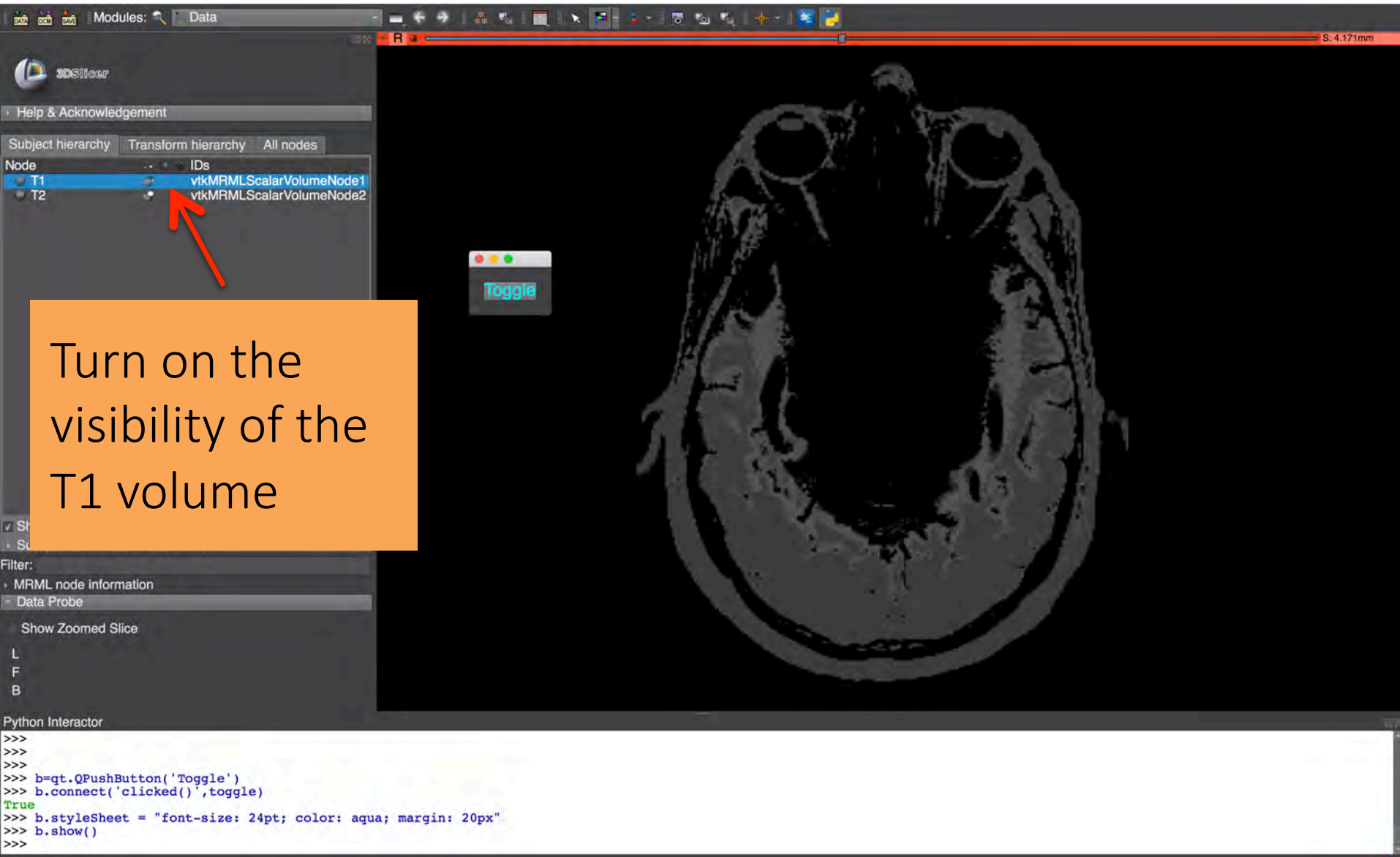


# Creating a Qt Push Button



The Toggle button appears

# Creating a Qt Push Button

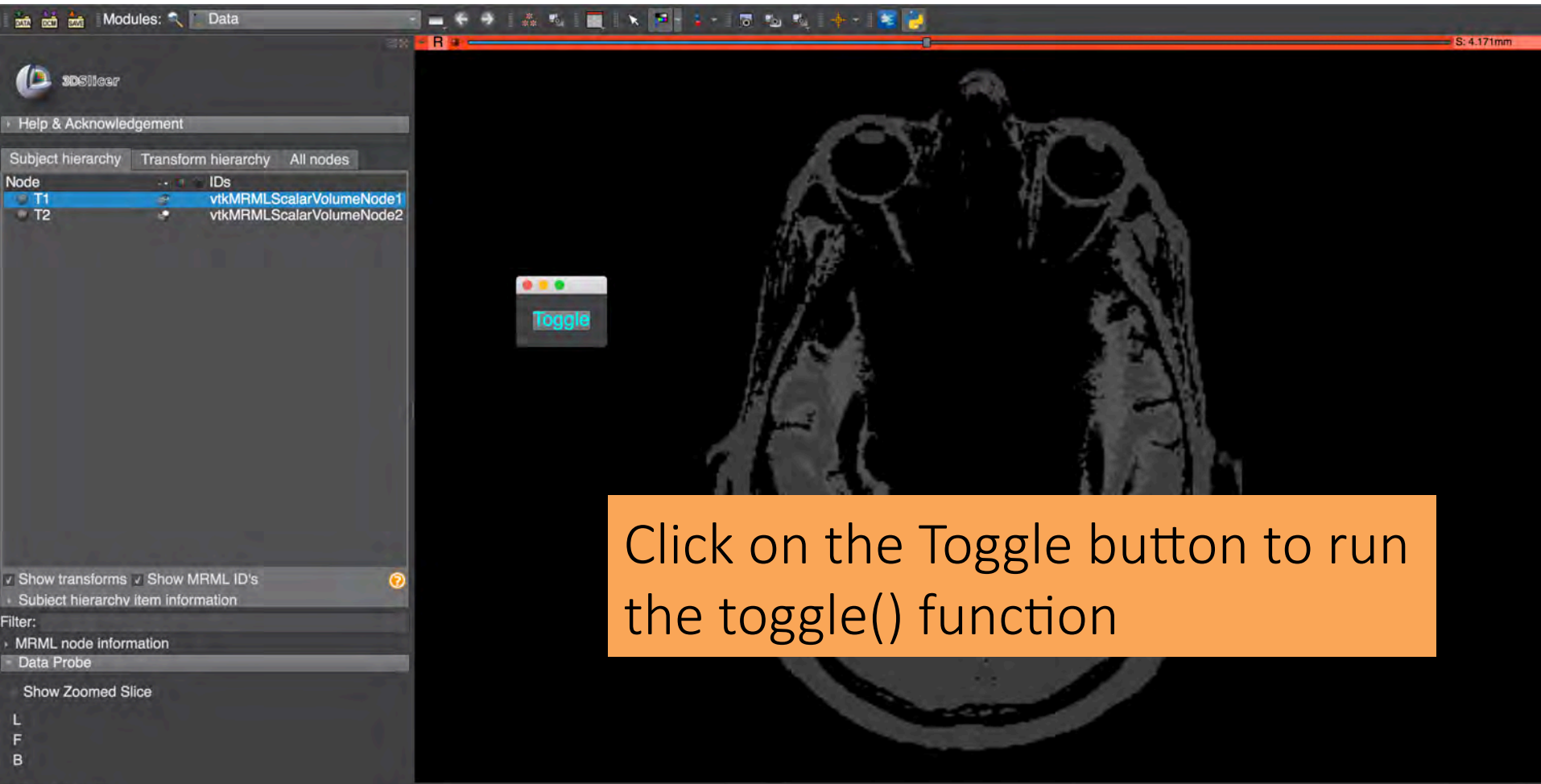


The screenshot displays the 3D Slicer software interface. On the left, the 'Subject hierarchy' panel shows a tree structure with 'T1' and 'T2' nodes. A red arrow points to the 'T1' node, which is highlighted in blue. Below this, the 'Python Interactor' panel shows a code snippet for creating a Qt push button. The main window displays a grayscale axial brain MRI slice. A small window titled 'Toggle' is visible in the center of the main window.

Turn on the visibility of the T1 volume

```
>>>
>>>
>>> b=qt.QPushButton('Toggle')
>>> b.connect('clicked()',toggle)
True
>>> b.setStyleSheet = "font-size: 24pt; color: aqua; margin: 20px"
>>> b.show()
>>>
```

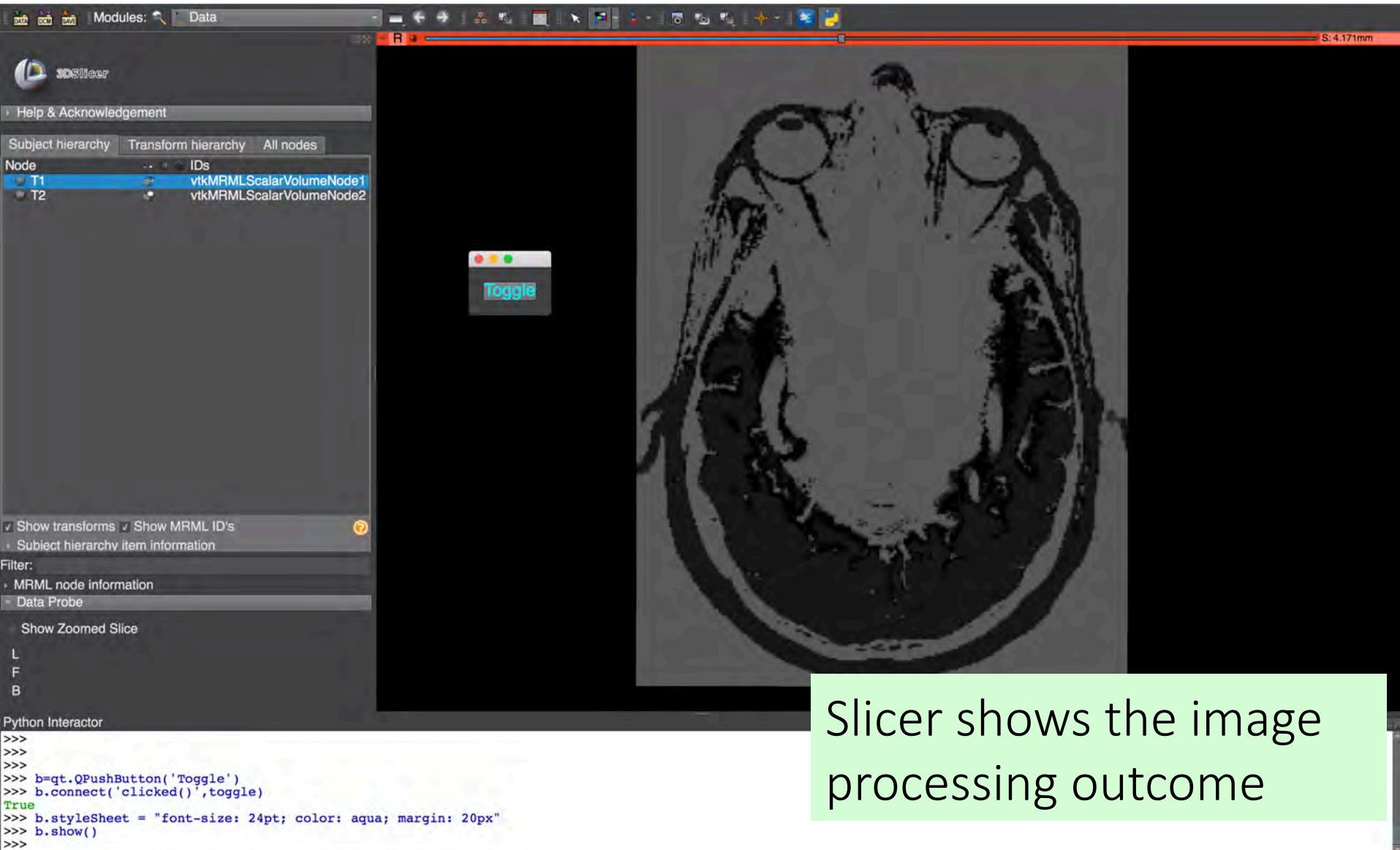
# Creating a Qt Push Button



The screenshot shows the 3D Slicer software interface. On the left, there is a sidebar with a 'Subject hierarchy' panel showing two nodes: 'T1' (vtkMRMLScalarVolumeNode1) and 'T2' (vtkMRMLScalarVolumeNode2). Below this is a 'Python Interactor' panel with a code editor. The main window displays a 3D rendering of a medical scan, likely a brain, with a 'Toggle' button overlaid on it. The button is a small rectangle with the word 'Toggle' in blue text. An orange text box is overlaid on the right side of the image, containing the instruction: 'Click on the Toggle button to run the toggle() function'.

```
>>>
>>>
>>> b=qt.QPushButton('Toggle')
>>> b.connect('clicked()',toggle)
True
>>> b.setStyleSheet = "font-size: 24pt; color: aqua; margin: 20px"
>>> b.show()
>>>
```

# Creating a Qt Push Button



Slicer shows the image processing outcome

# Examples of scripted modules


- The tutorial demonstrates how to create a simple interface in Python
- Slicer integrates many sophisticated scripted modules such as Segment Statistics, Sample Data, Endoscopy module, etc.
- For further reading, please look at the Slicer Script Repository:

<https://www.slicer.org/wiki/Documentation/Nightly/ScriptRepository>

# Conclusion

- Slicer enables you to create complex interfaces that are streamlined for target users
- The software platform provides unlimited customization possibilities
- Slicer gives you access to advanced underlying libraries through a cross-platform package that is easy to deploy to end-users

# Acknowledgments

 Neuroimage Analysis Center  
(NIBIB P41 EB015902)