



Slicer Developer Tutorial: Programming in Slicer

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.

Goal of the tutorial



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



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

This tutorial is an introduction to the Python interactor and the Qt widget toolkit in 3D Slicer release version 5

Tutorial Outline



Part 1: 3D Slicer Modules Overview



Part 2: Getting Familiar with the Python environment in 3D Slicer

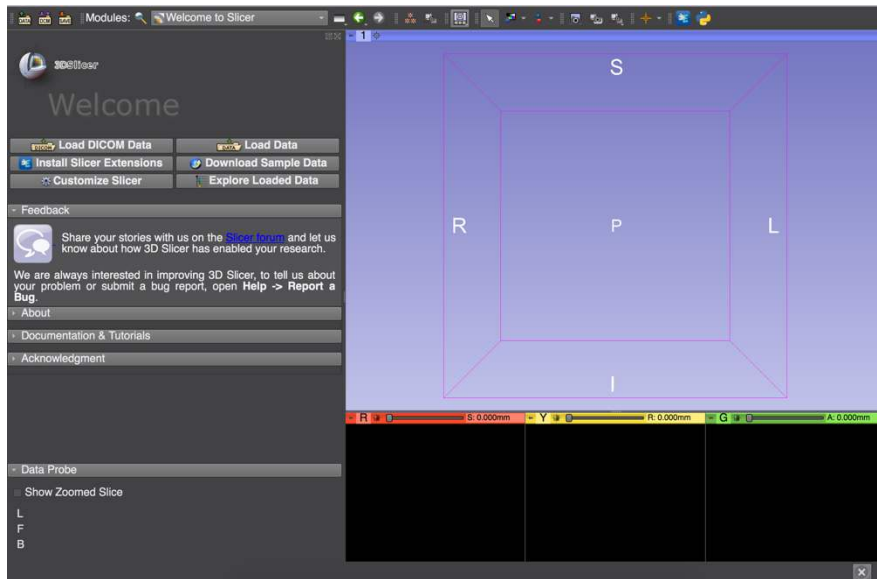


Part 3: Getting Familiar with the Qt widget toolkit in 3D Slicer

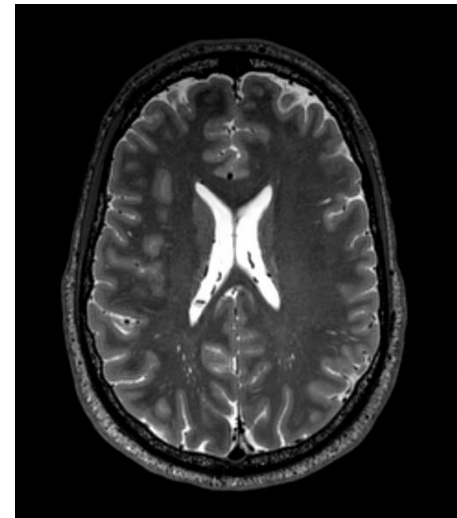
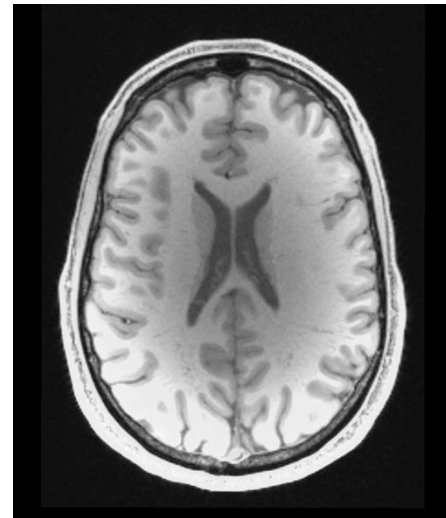
Disclaimer

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

Tutorial materials



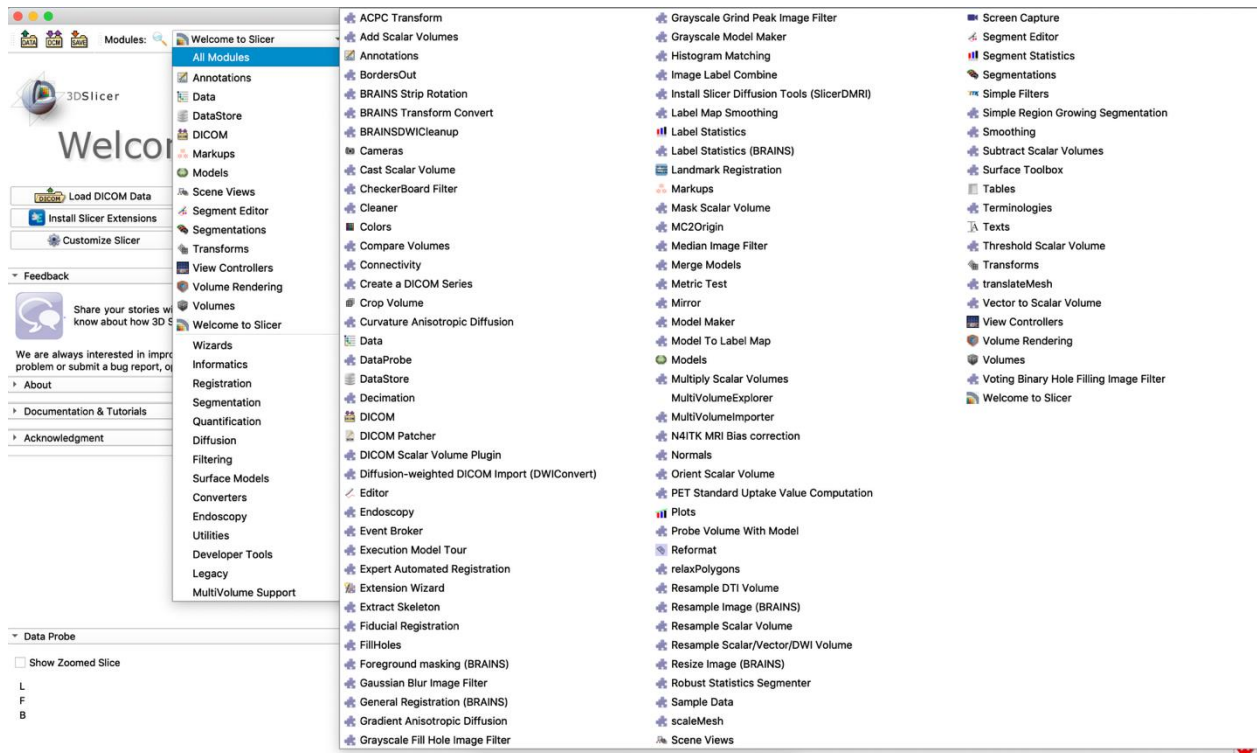
3D Slicer version 5.10



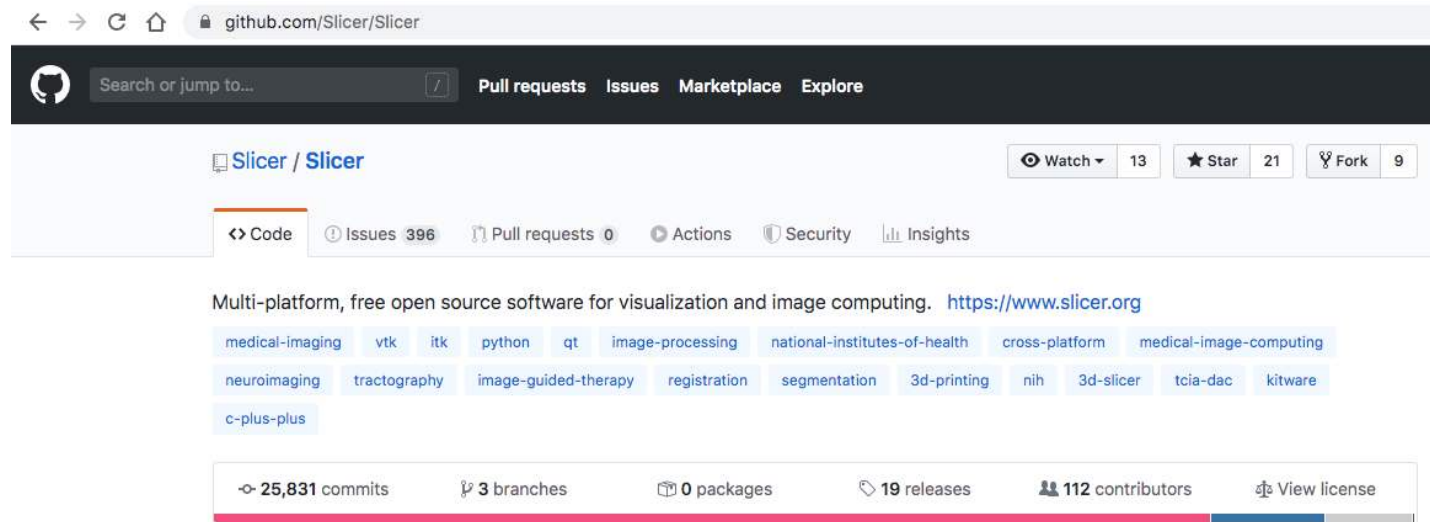
SlicerProgrammingTutorialData.zip

Part 1

Slicer Modules Overview



3D Slicer



- 3D Slicer is an open-source platform for the analysis and visualization of medical imaging data
- 3D Slicer is compiled and tested every day on Windows, Mac, and Linux platforms
- The source code is freely available on GitHub at <http://github.com/Slicer/Slicer>

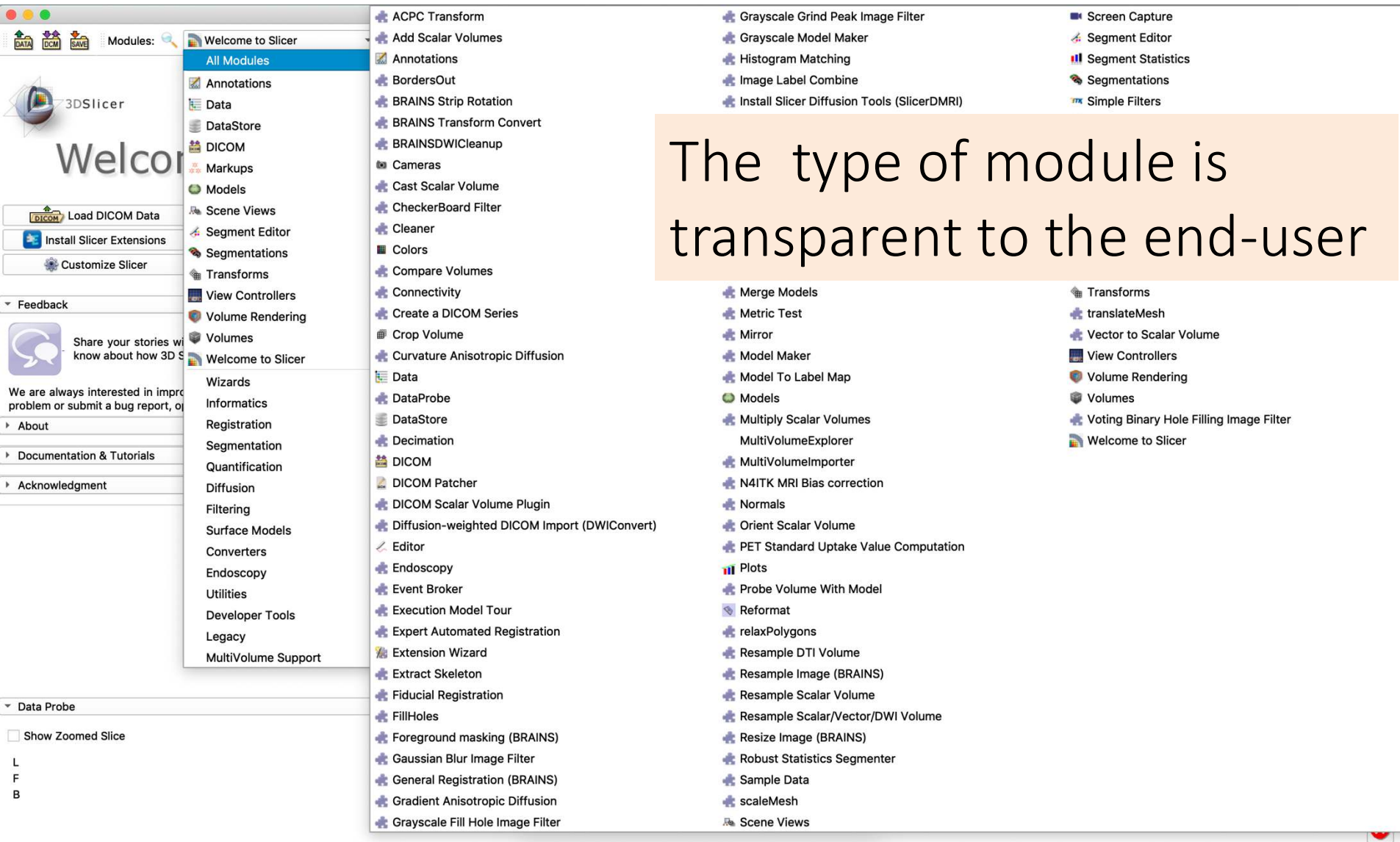
Slicer Modules

3D Slicer 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

Slicer Modules

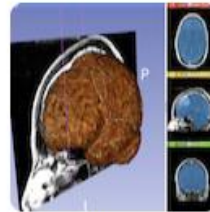


The screenshot displays the 3D Slicer application interface. On the left, the 'Modules' menu is open, showing a list of categories and sub-modules. The 'All Modules' option is selected. The main window shows a list of modules, including 'ACPC Transform', 'Add Scalar Volumes', 'Annotations', 'BordersOut', 'BRAINS Strip Rotation', 'BRAINS Transform Convert', 'BRAINSDWICleanup', 'Cameras', 'Cast Scalar Volume', 'CheckerBoard Filter', 'Cleaner', 'Colors', 'Compare Volumes', 'Connectivity', 'Create a DICOM Series', 'Crop Volume', 'Curvature Anisotropic Diffusion', 'Data', 'DataProbe', 'DataStore', 'Decimation', 'DICOM', 'DICOM Patcher', 'DICOM Scalar Volume Plugin', 'Diffusion-weighted DICOM Import (DWIConvert)', 'Editor', 'Endoscopy', 'Event Broker', 'Execution Model Tour', 'Expert Automated Registration', 'Extension Wizard', 'Extract Skeleton', 'Fiducial Registration', 'FillHoles', 'Foreground masking (BRAINS)', 'Gaussian Blur Image Filter', 'General Registration (BRAINS)', 'Gradient Anisotropic Diffusion', 'Grayscale Fill Hole Image Filter', 'Grayscale Grind Peak Image Filter', 'Grayscale Model Maker', 'Histogram Matching', 'Image Label Combine', 'Install Slicer Diffusion Tools (SlicerDMRI)', 'Merge Models', 'Metric Test', 'Mirror', 'Model Maker', 'Model To Label Map', 'Models', 'Multiply Scalar Volumes', 'MultiVolumeExplorer', 'MultiVolumeImporter', 'N4ITK MRI Bias correction', 'Normals', 'Orient Scalar Volume', 'PET Standard Uptake Value Computation', 'Plots', 'Probe Volume With Model', 'Reformat', 'relaxPolygons', 'Resample DTI Volume', 'Resample Image (BRAINS)', 'Resample Scalar Volume', 'Resample Scalar/Vector/DWI Volume', 'Resize Image (BRAINS)', 'Robust Statistics Segmenter', 'Sample Data', 'scaleMesh', 'Scene Views', 'Screen Capture', 'Segment Editor', 'Segment Statistics', 'Segmentations', 'Simple Filters', 'Transforms', 'translateMesh', 'Vector to Scalar Volume', 'View Controllers', 'Volume Rendering', 'Volumes', 'Voting Binary Hole Filling Image Filter', and 'Welcome to Slicer'.

The type of module is transparent to the end-user

Slicer Extensions

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



SwissSkullStripper
Bill Lorensen (Noware...

★★★★★ (0)

INSTALL



PETTumorSegmenta...
Christian Bauer (Univ...

★★★★★ (0)

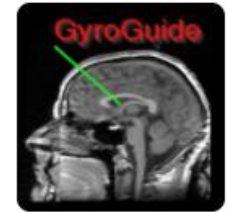
INSTALL



SlicerOpenIGTLink
Junichi Tokuda (SPL, ...

★★★★★ (0)

INSTALL



GyroGuide
Ruifeng Chen, Luping...

★★★★★ (0)

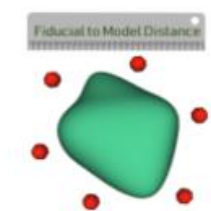
INSTALL



PET-IndiC
Ethan Ulrich (Universi...

★★★★★ (0)

INSTALL



FiducialsToModelDi...
Jesse Reynolds (Cante...

★★★★★ (0)

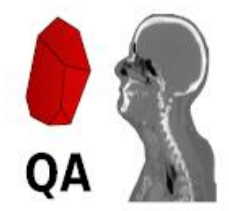
INSTALL



Slicer-Wasp
Thomas Lawson (MR...

★★★★★ (0)

INSTALL



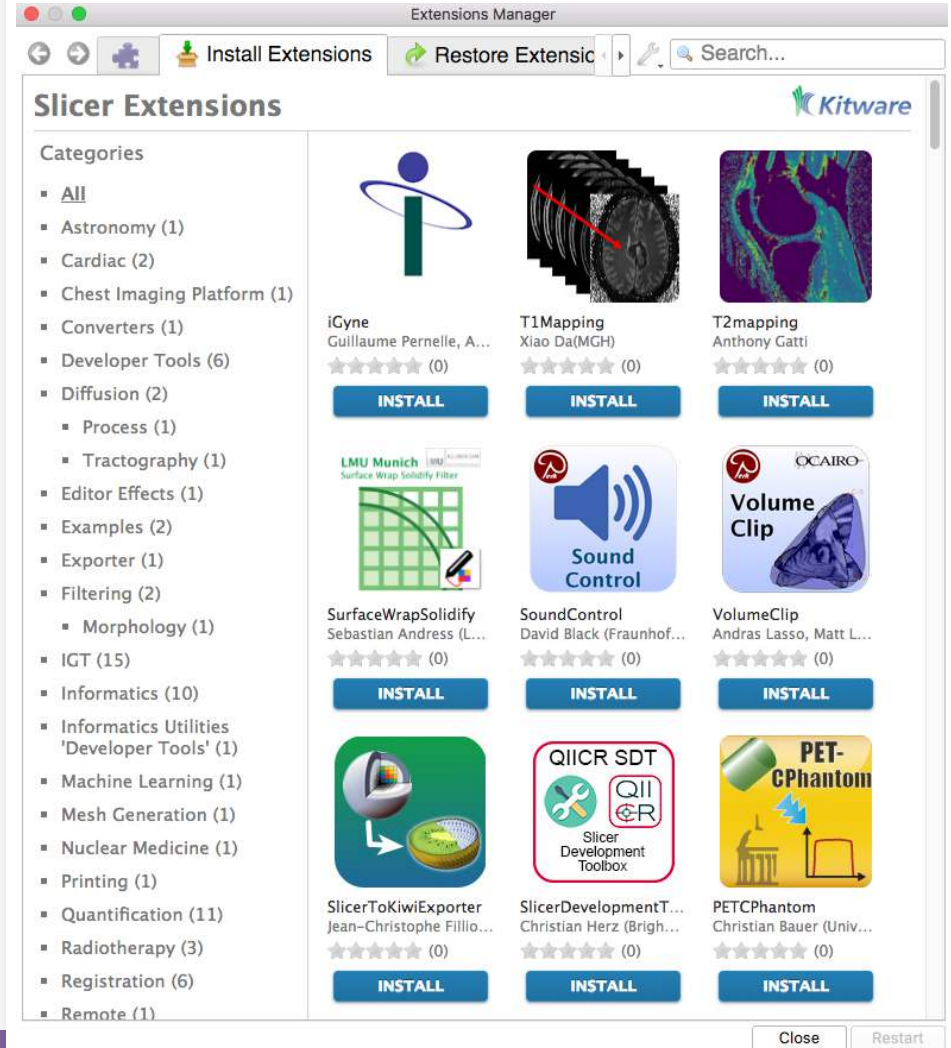
ImageCompare
Paolo Zaffino (Magna ...

★★★★★ (0)

INSTALL

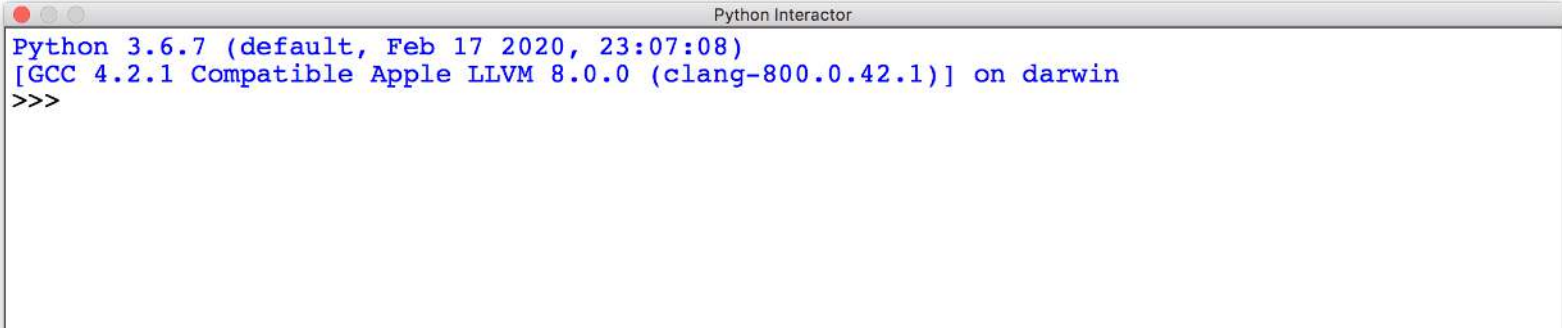
Slicer Extension Manager

- 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



```
Python Interactor
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

Slicer v5.10 works with **Python3** and a rich set of standard libraries

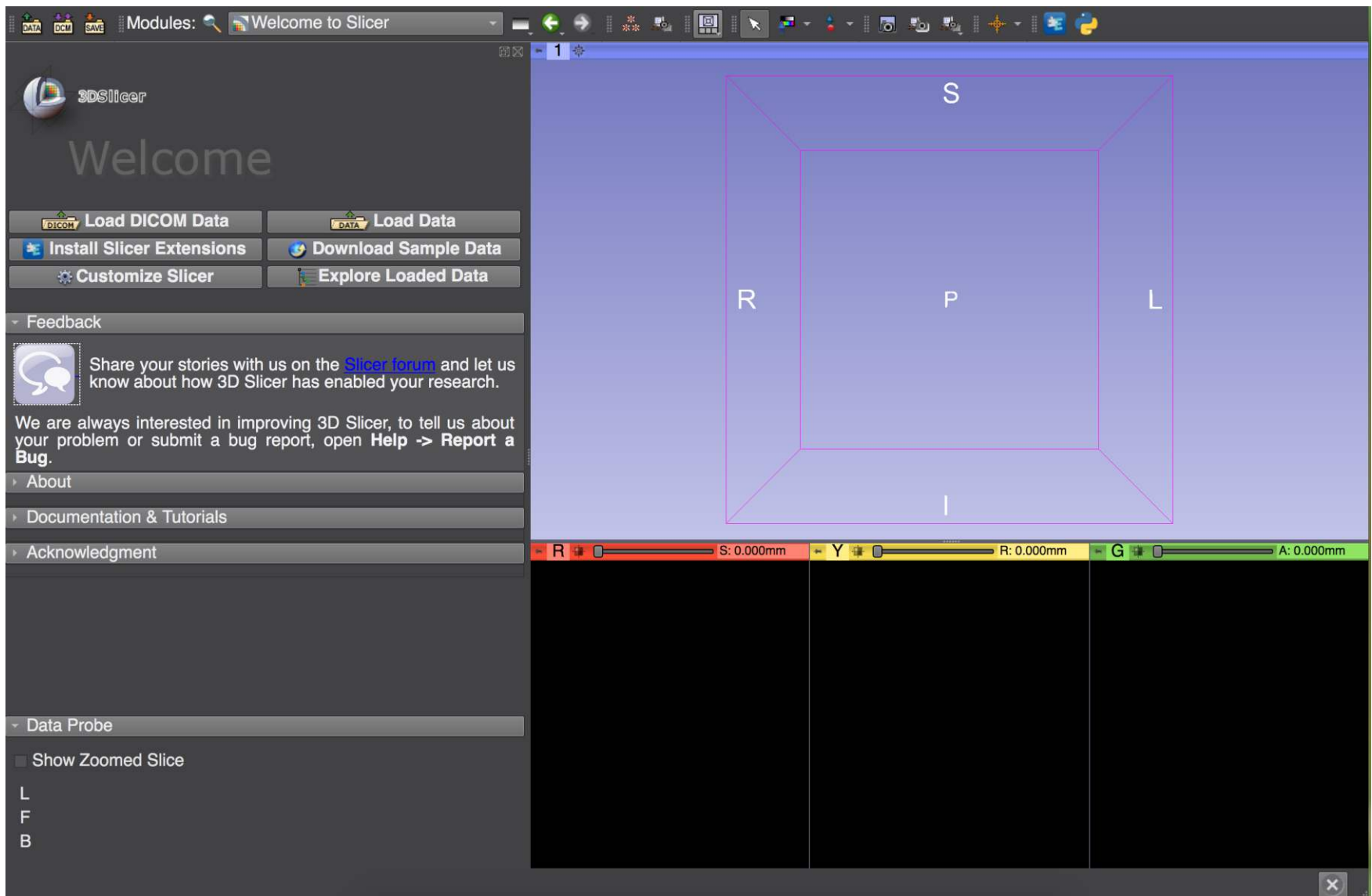
 NumPy	NumPy is the fundamental package for scientific computing with Python.
	VTK is an open-source library for manipulating and displaying scientific data.
	ITK is an open-source library for image analysis.
	CTK is an open-source library for biomedical image computing.
	PythonQT is a Python binding 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>)

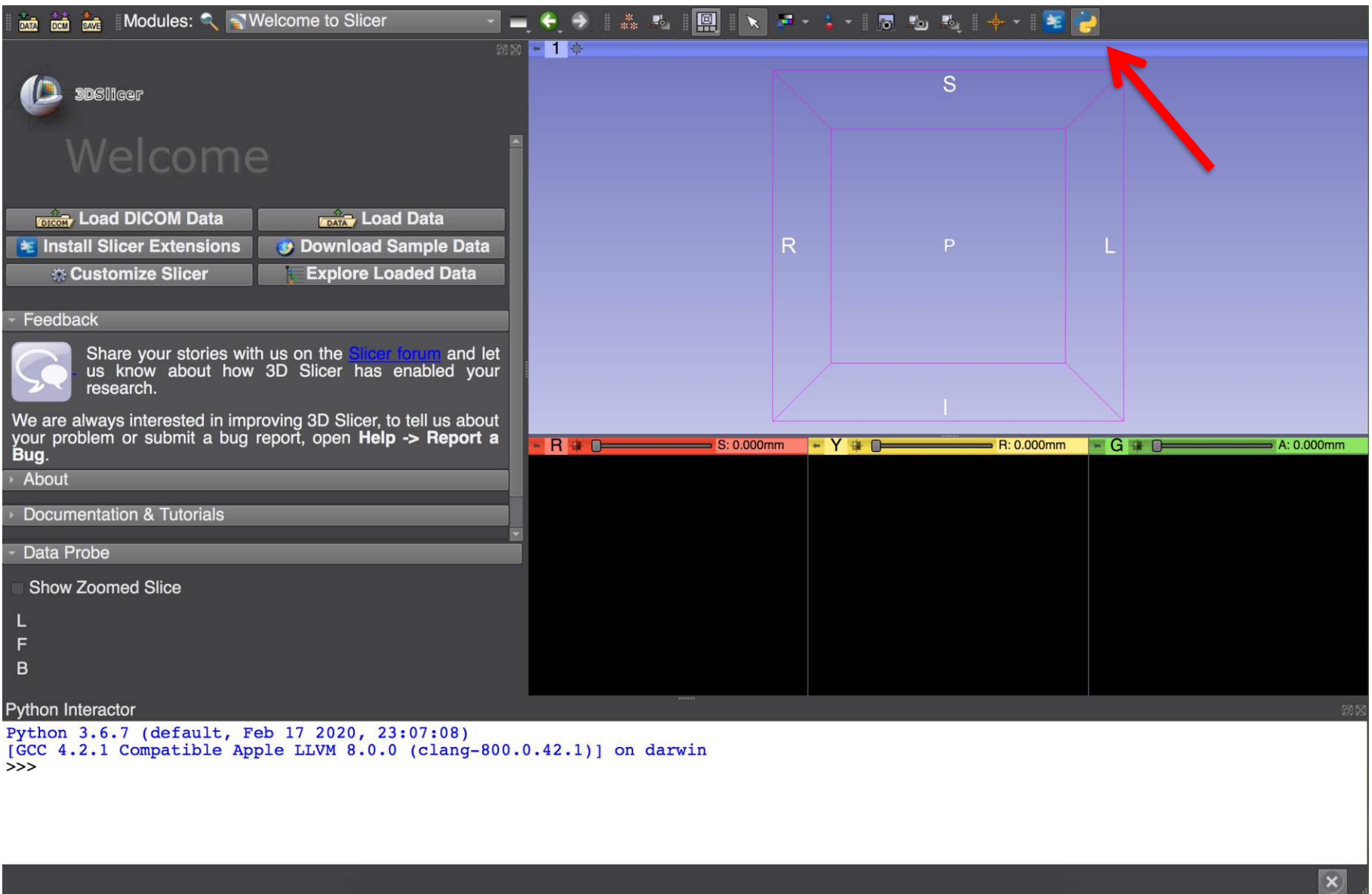
- The **pip install** command in Slicer enables developers to install most common scientific computing tools (e.g. TensorFlow, SciPy, PyTorch, Pandas, etc.)
- Slicer can be used as a **Jupyter notebook** kernel
- PyCharm and other Python development tools can be used with Slicer



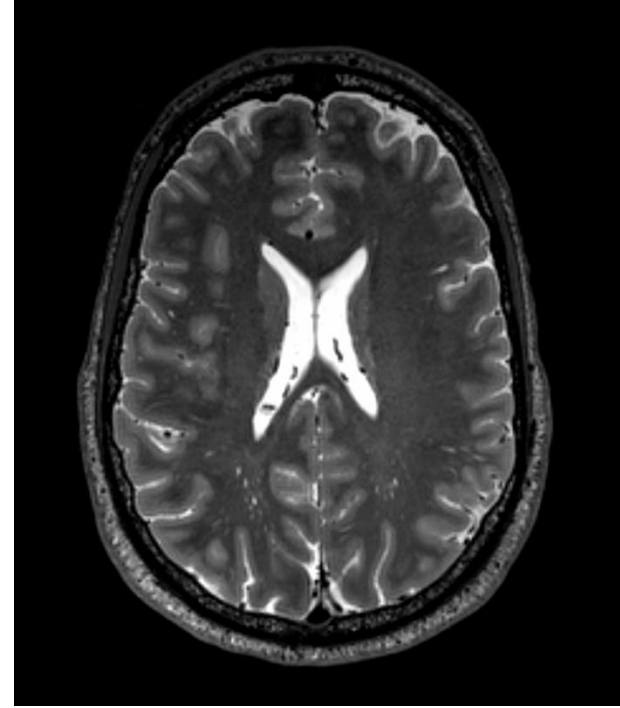
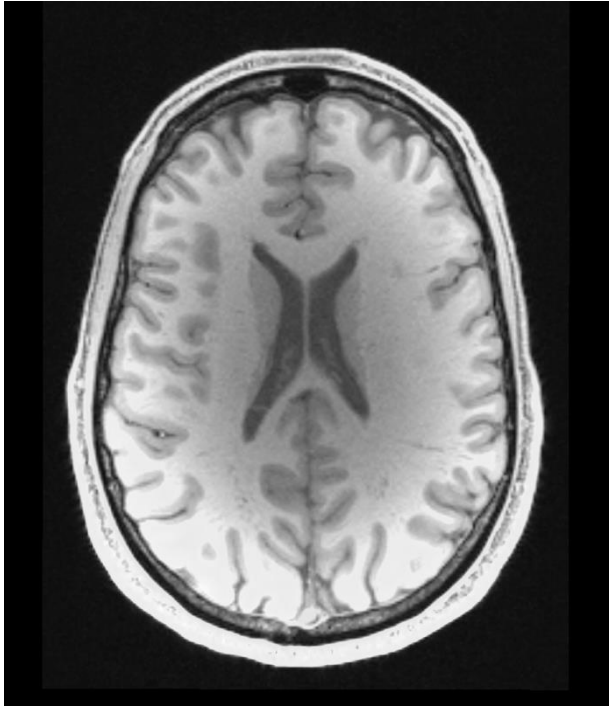
Slicer release version 5 integrates
Python3, VTK5 and ITK5

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.

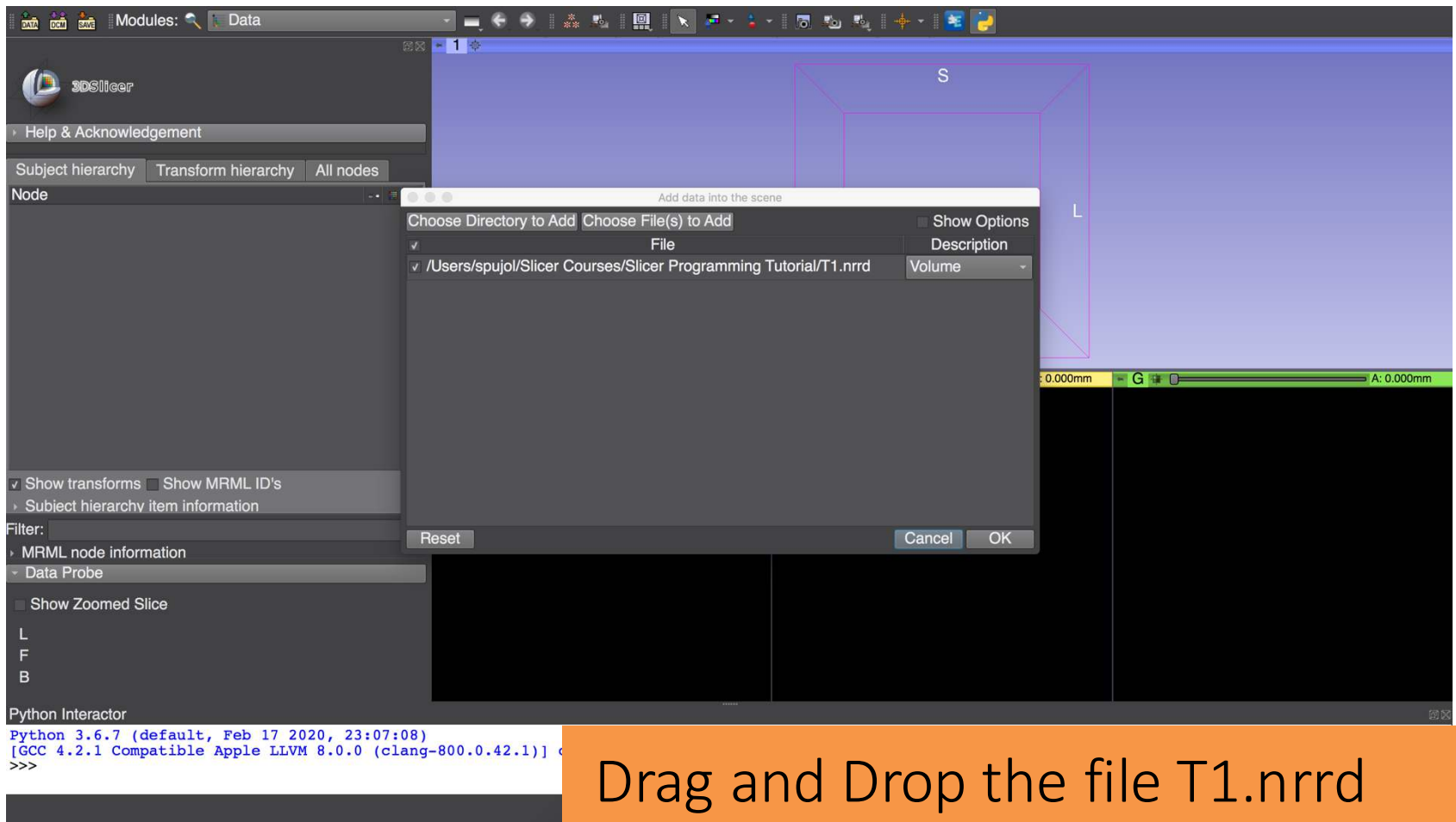


To access the Python Interactor, click on the Python icon  in the top bar menu of Slicer



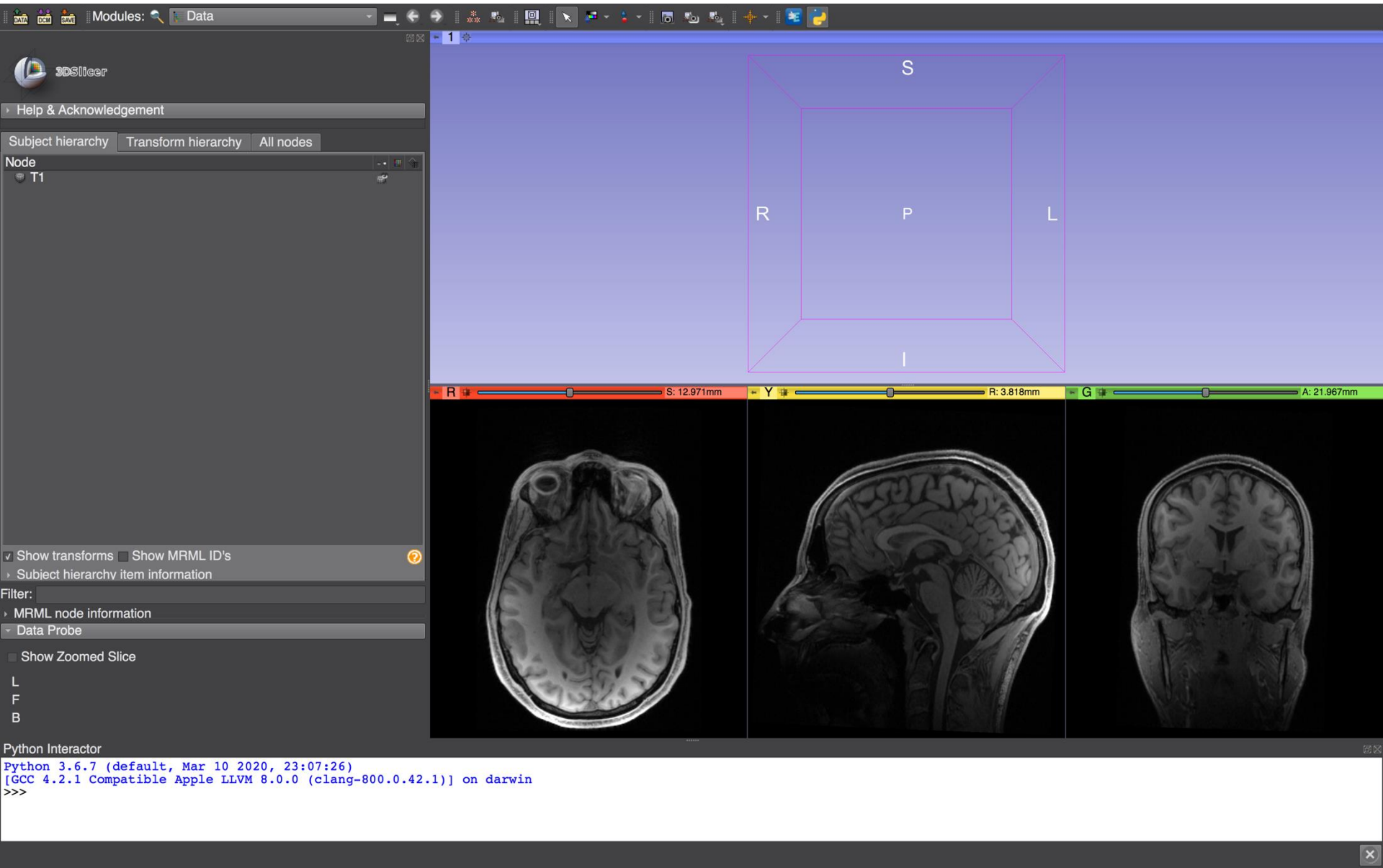
The Slicer Programming tutorial dataset includes a T1-weighted and a T2-weighted MRI scan of a healthy subject

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 3D Slicer

Slicer Data Model



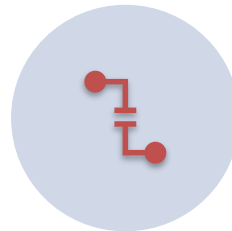
The **Slicer Data Model** is based on the Slicer Scene Data Structure



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

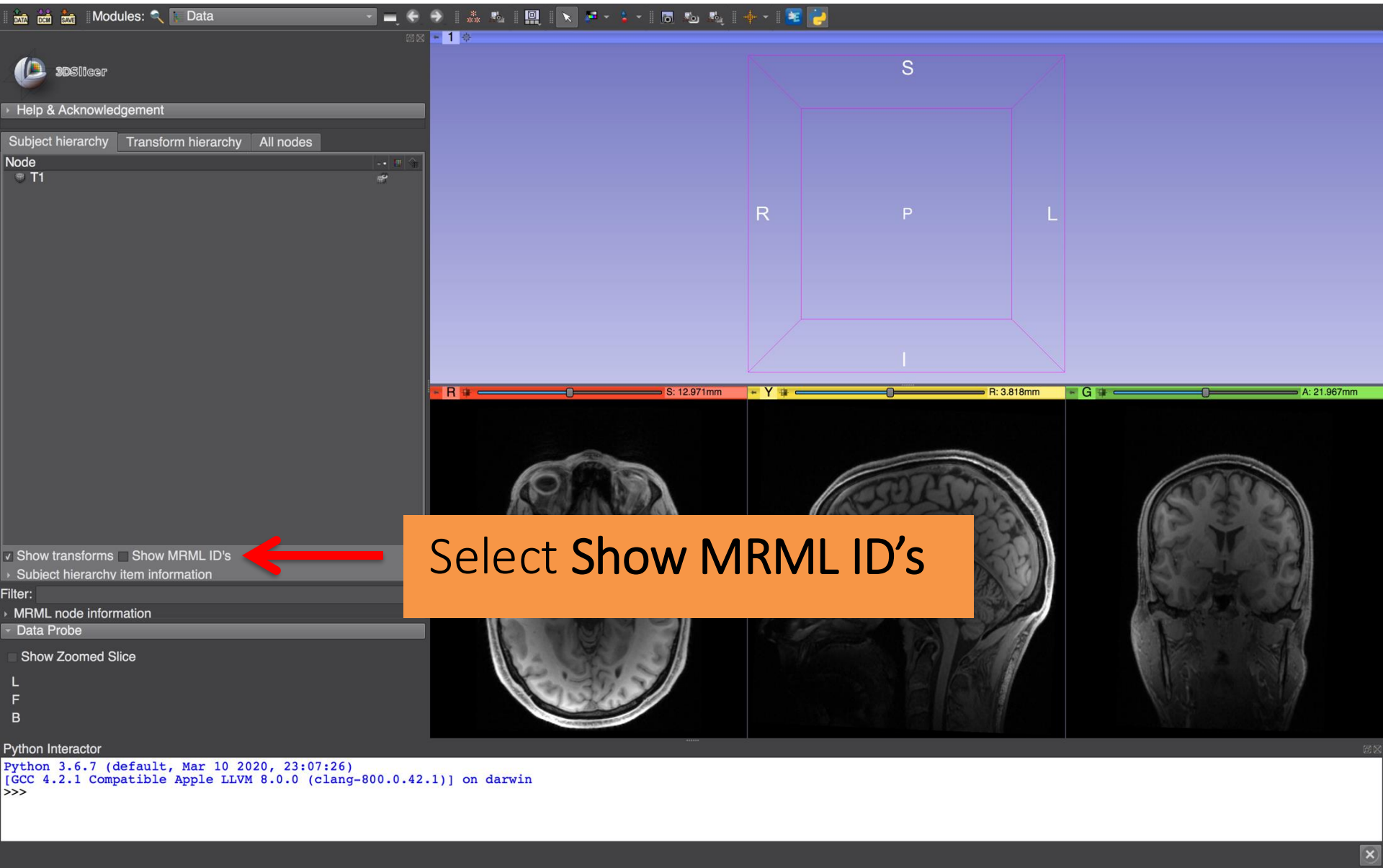


Display Node: Describes how the data should be visualized

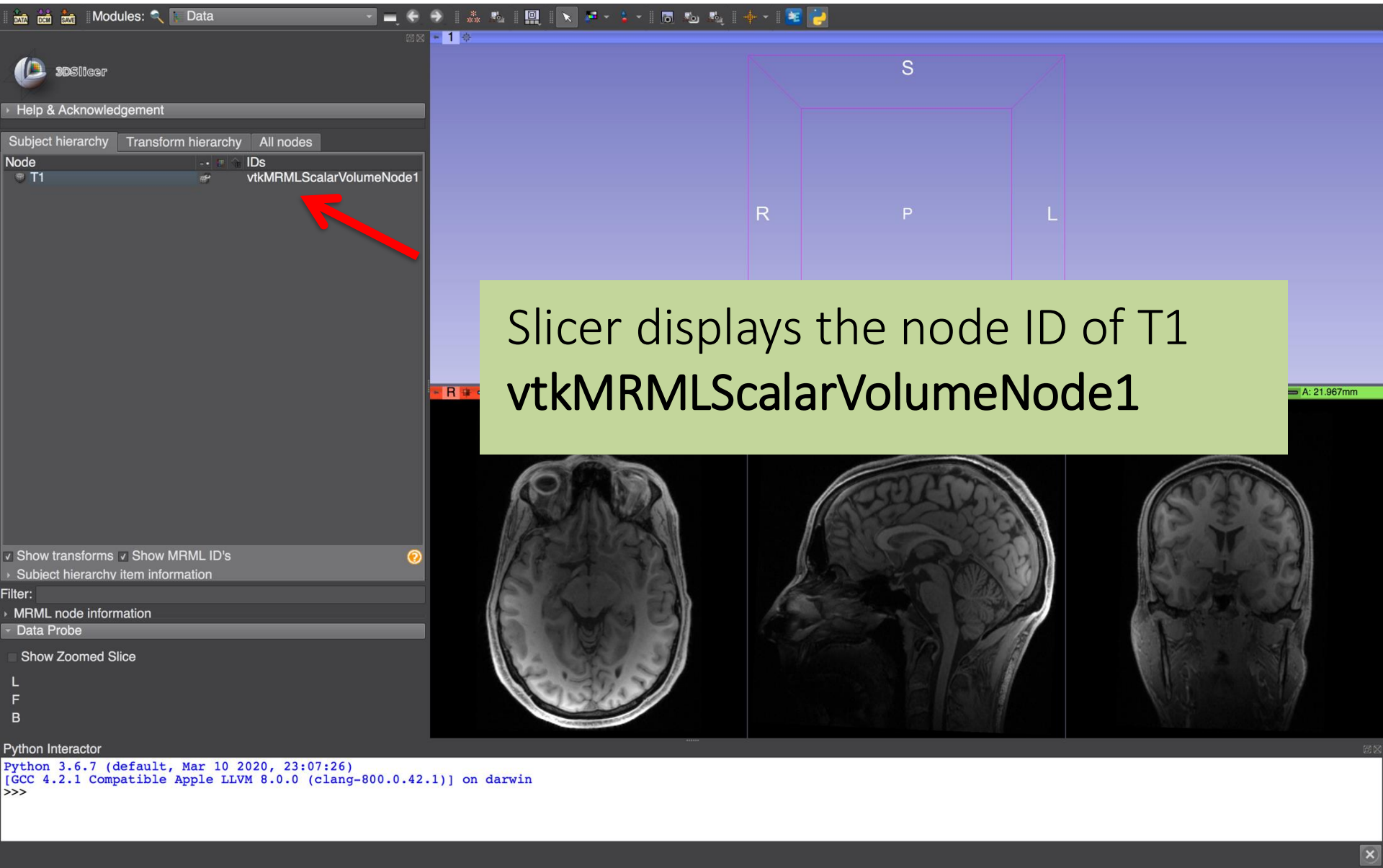


Storage Node: Describes how the data should be stored on disk

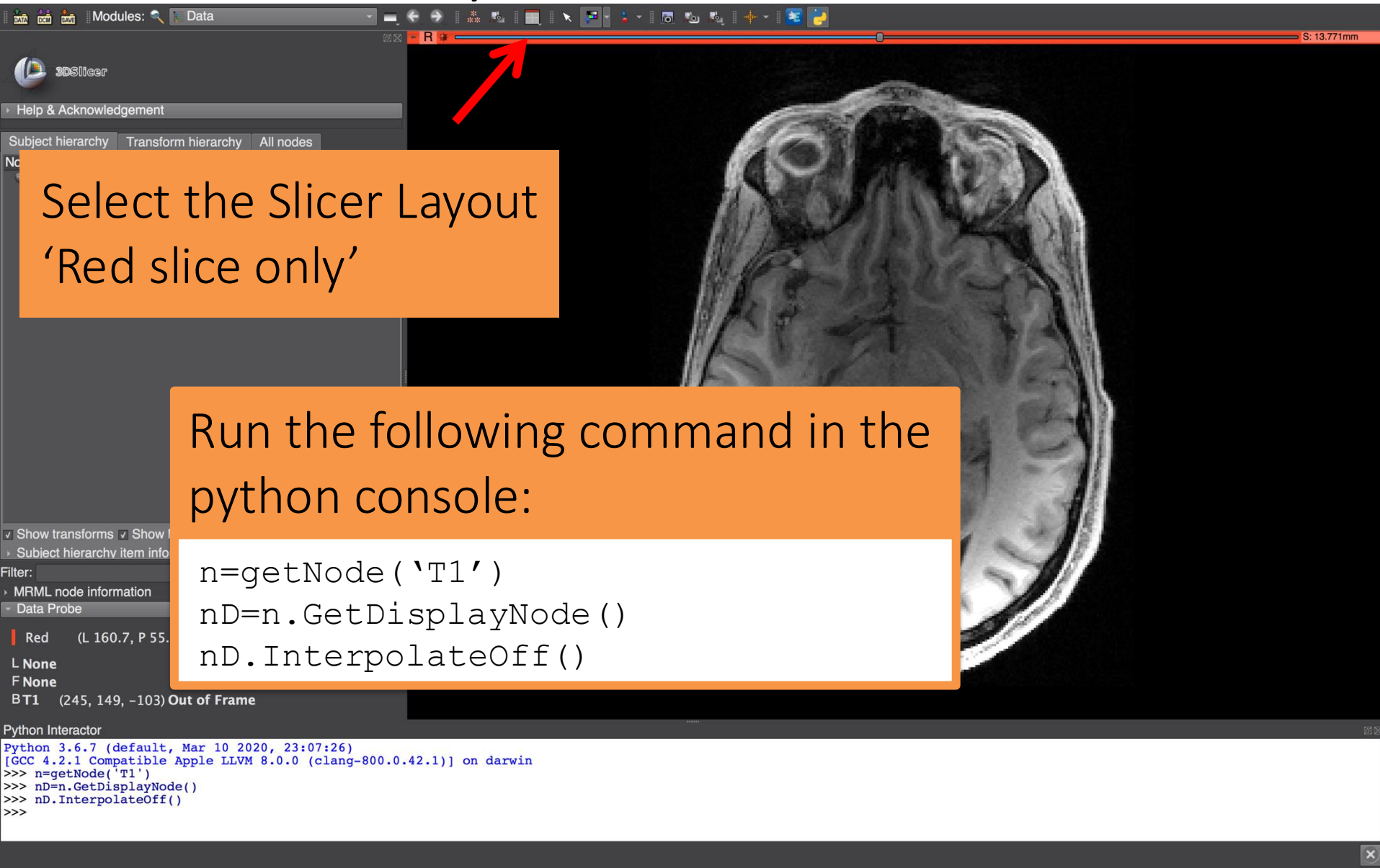
Tutorial dataset



Slicer Data Model



Accessing MRML nodes from the Python interactor



Select the Slicer Layout 'Red slice only'

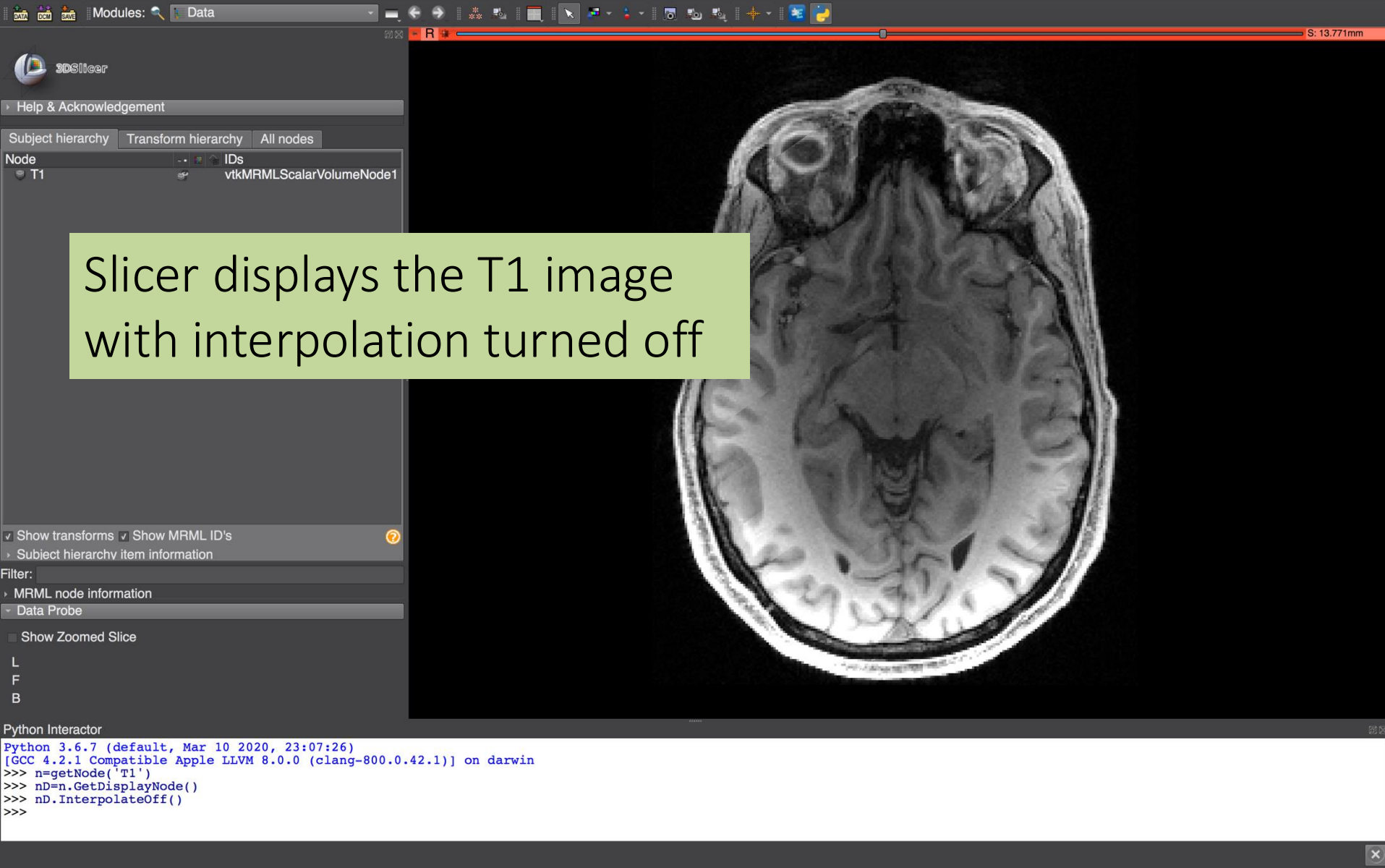
Run the following command in the python console:

```
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

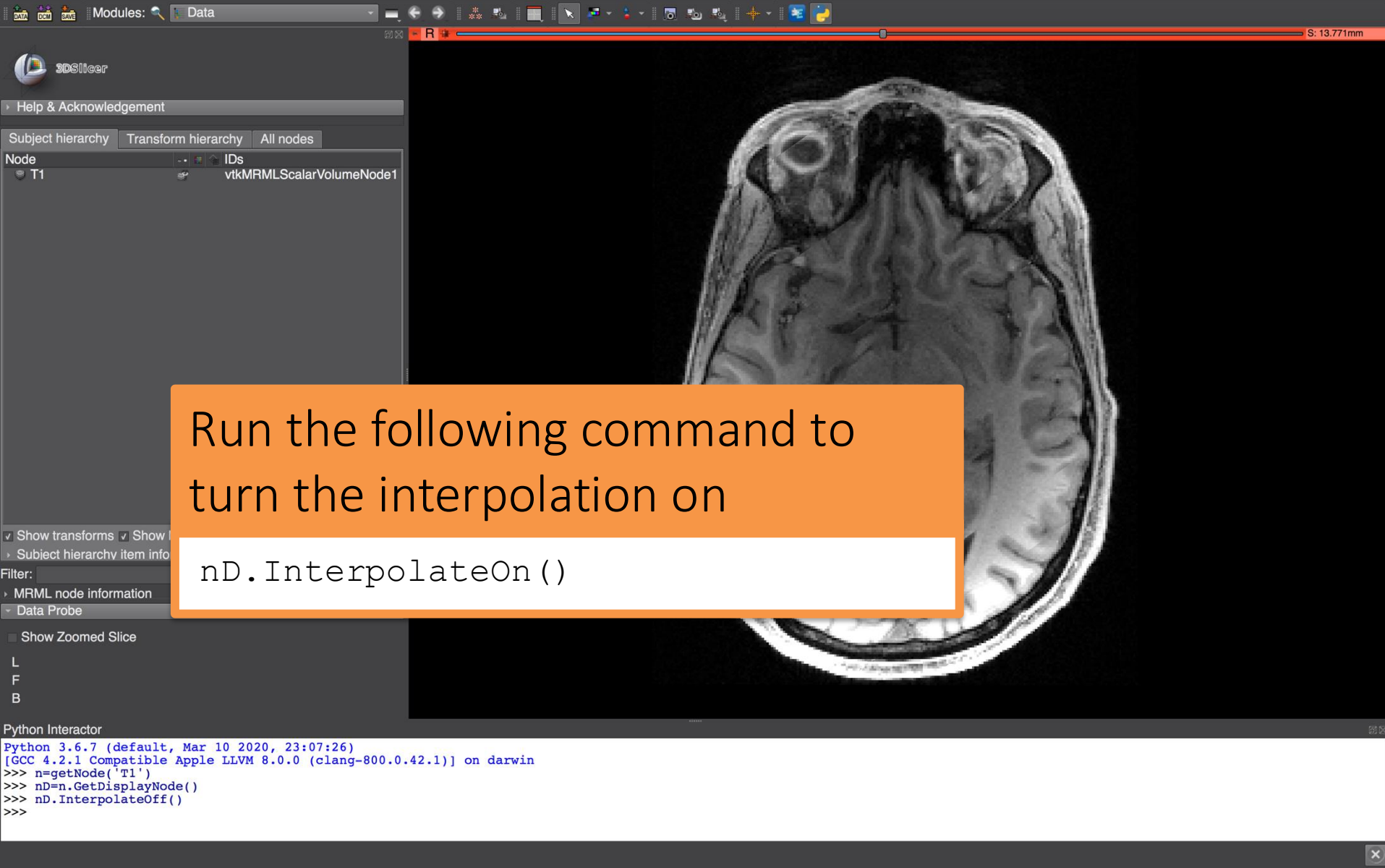


Slicer displays the T1 image with interpolation turned off

The screenshot shows the 3D Slicer software interface. The main window displays a T1 brain MRI scan. The left sidebar contains the 'Subject hierarchy' panel, which shows the 'Node' list with 'T1' selected. Below this, the 'Data Probe' panel is visible, showing 'Show Zoomed Slice' and 'L', 'F', 'B' buttons. The bottom of the interface features a 'Python Interactor' window with 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

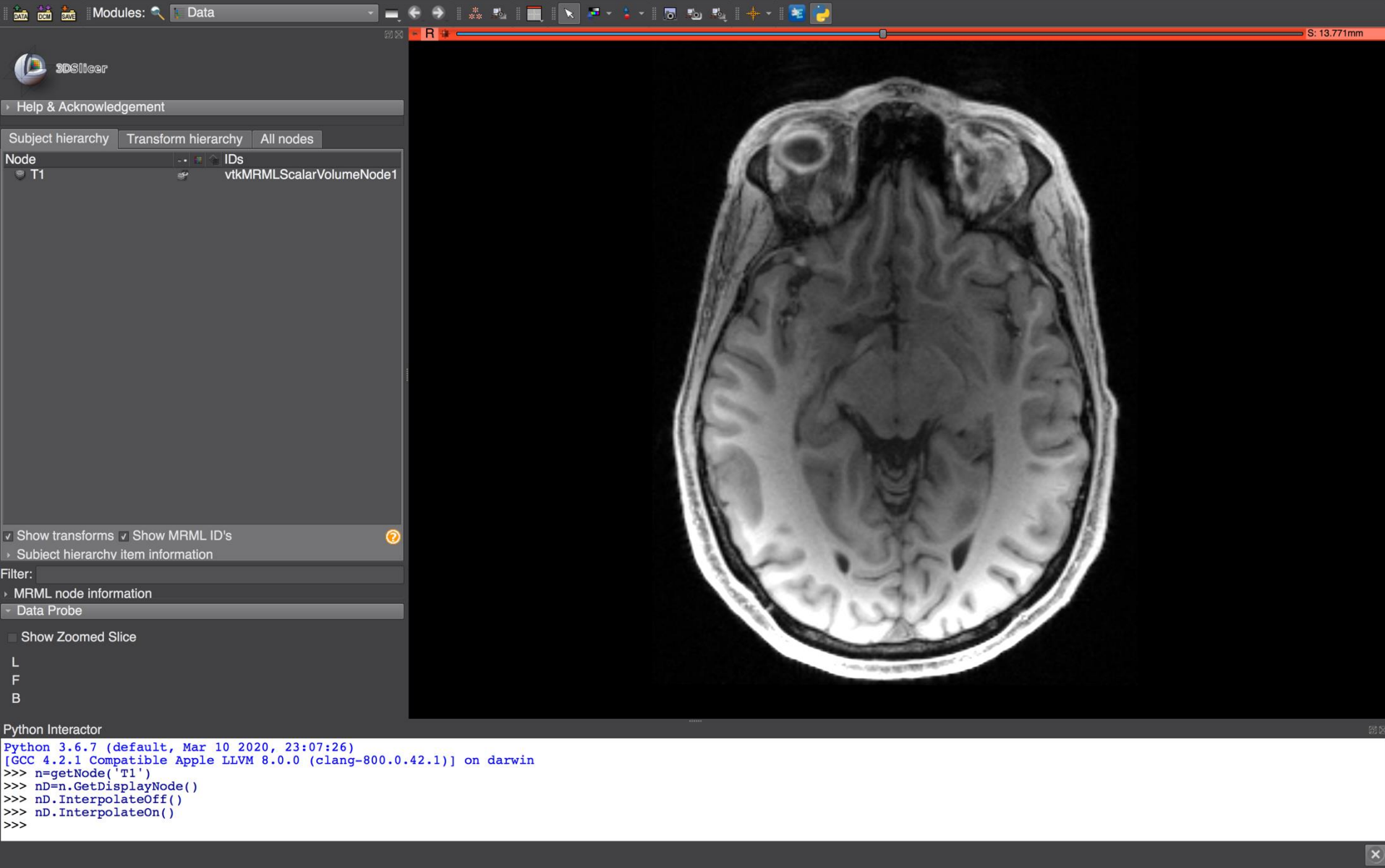


Run the following command to turn the interpolation on

```
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()
>>>
```

Accessing MRML nodes from the Python interactor

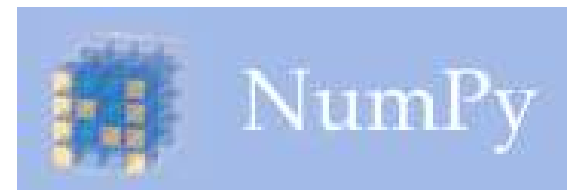


The screenshot displays the 3D Slicer software interface. The main window shows an axial brain MRI slice. The left sidebar contains the 'Subject hierarchy' panel, which lists the 'T1' node under the 'Node' tab. Below this, the 'MRML node information' panel is visible, showing the 'Data Probe' section. The bottom panel is the 'Python Interactor', which contains the following code and output:

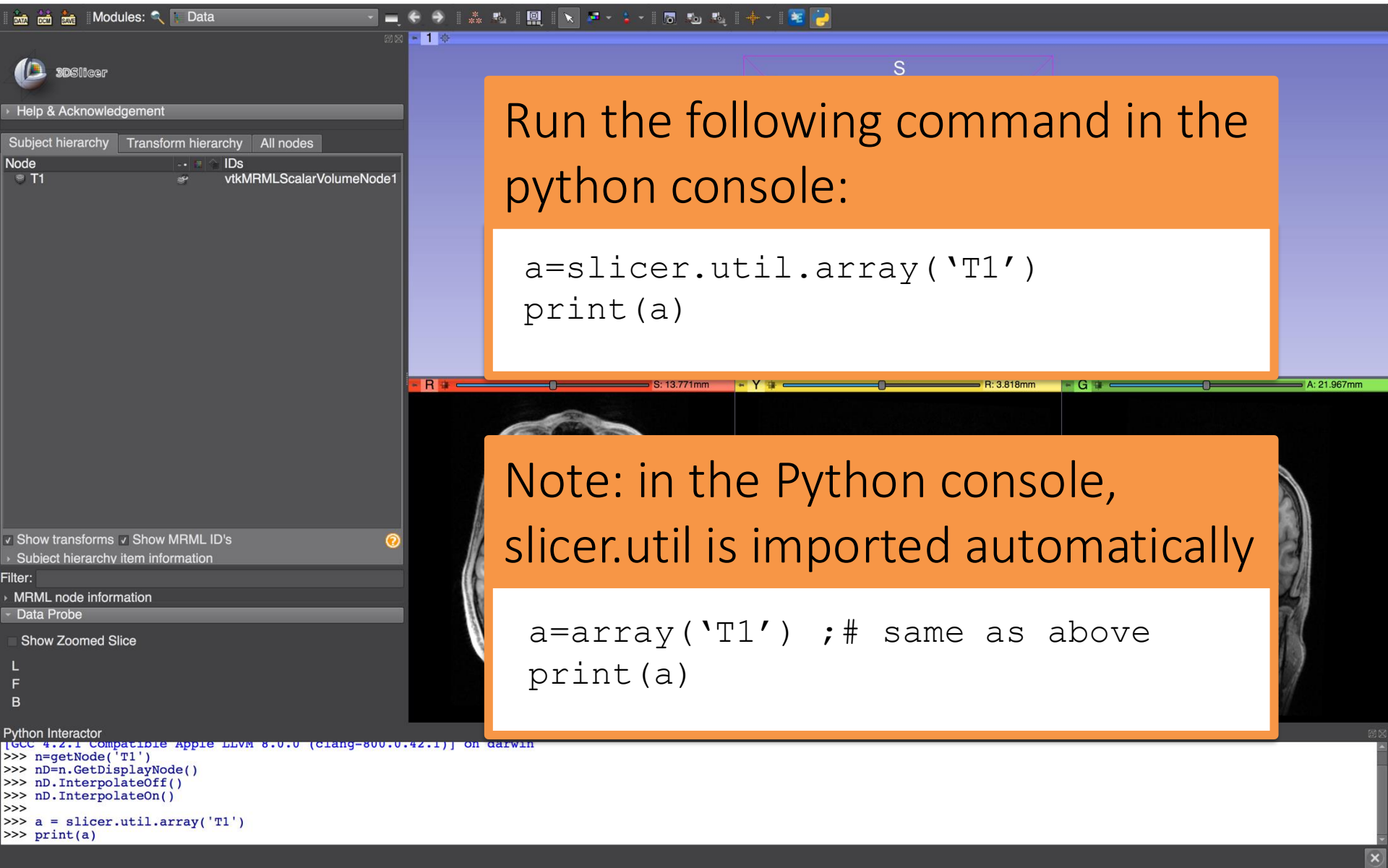
```
Python 3.6.7 (default, Mar 10 2020, 23:07:26)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.42.1)] on darwin
>>> n=getNode('T1')
>>> nD=n.GetDisplayNode()
>>> nD.InterpolateOff()
>>> nD.InterpolateOn()
>>>
```

Accessing 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 Python Interactor window. The Python console displays the output of a script that prints a 3D array 'a' using a truncated display. The array is a 3x3x3 volume with values ranging from 0 to 49. The output is truncated to show only the first few elements of each row and column. The 3D view shows a brain model with a red bounding box labeled 'R' and a green bounding box labeled '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]]

...

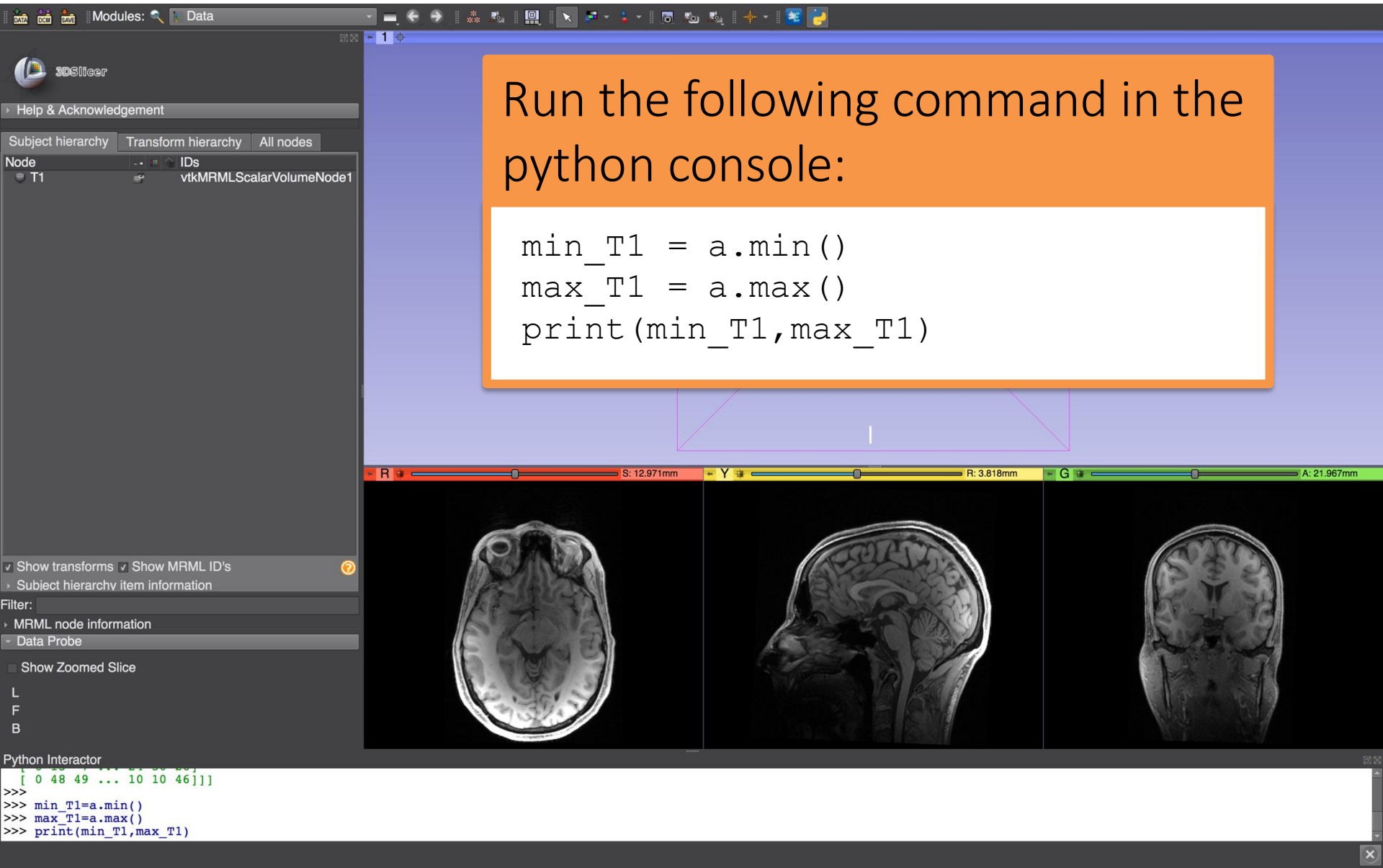
[[ 0  0  0 ...  0  0  0]
 [ 0  8 26 ... 33 36 44]
 [ 0 27 18 ... 21 21 45]
 ...
 [ 0 12 22 ... 22 34 14]
 [ 0  2 11 ... 48 65 35]
 [ 0 25  7 ...  7 17 11]]

[[ 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]]]
>>>

```

Accessing voxels in a volume

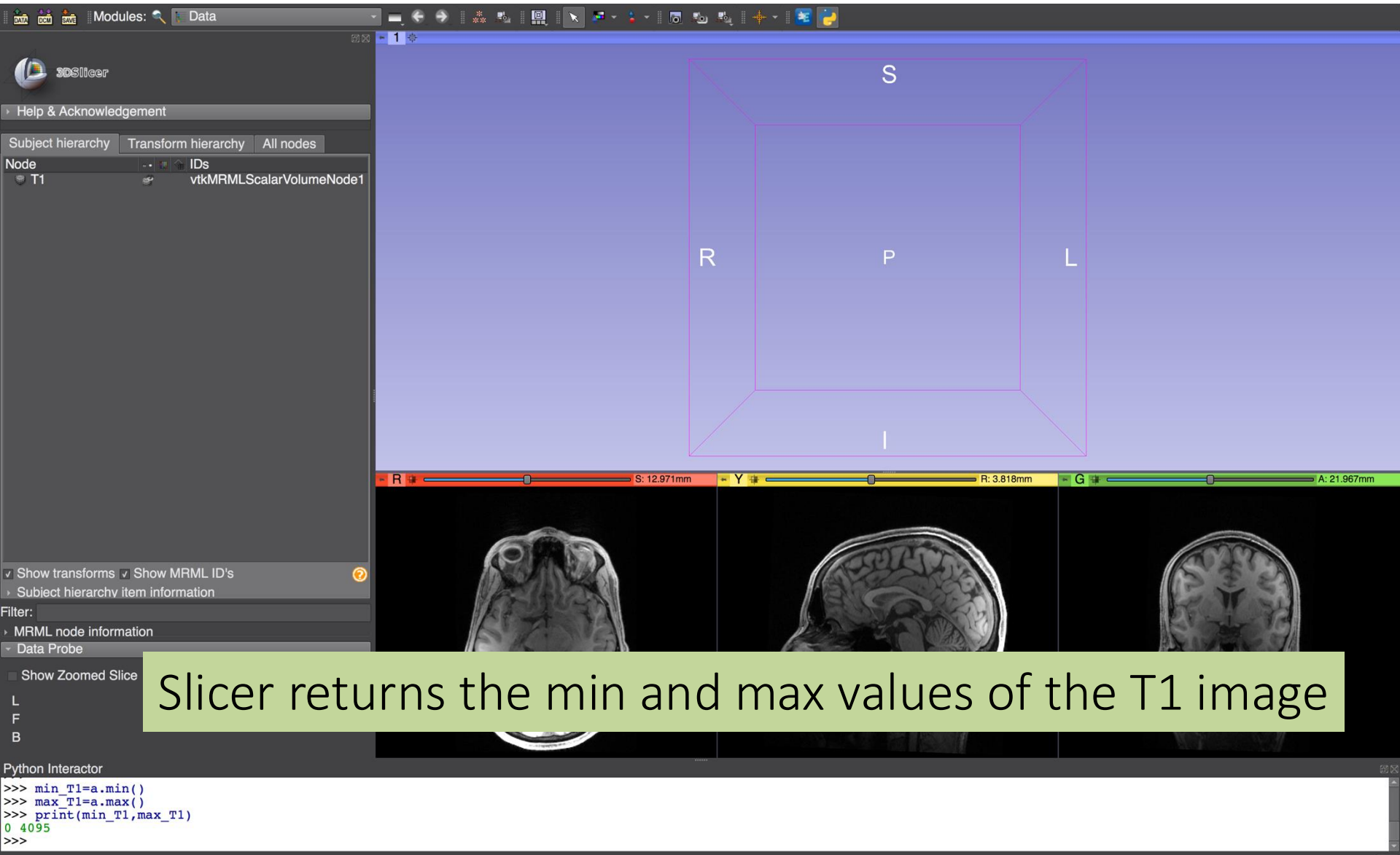


Run the following command in the python console:

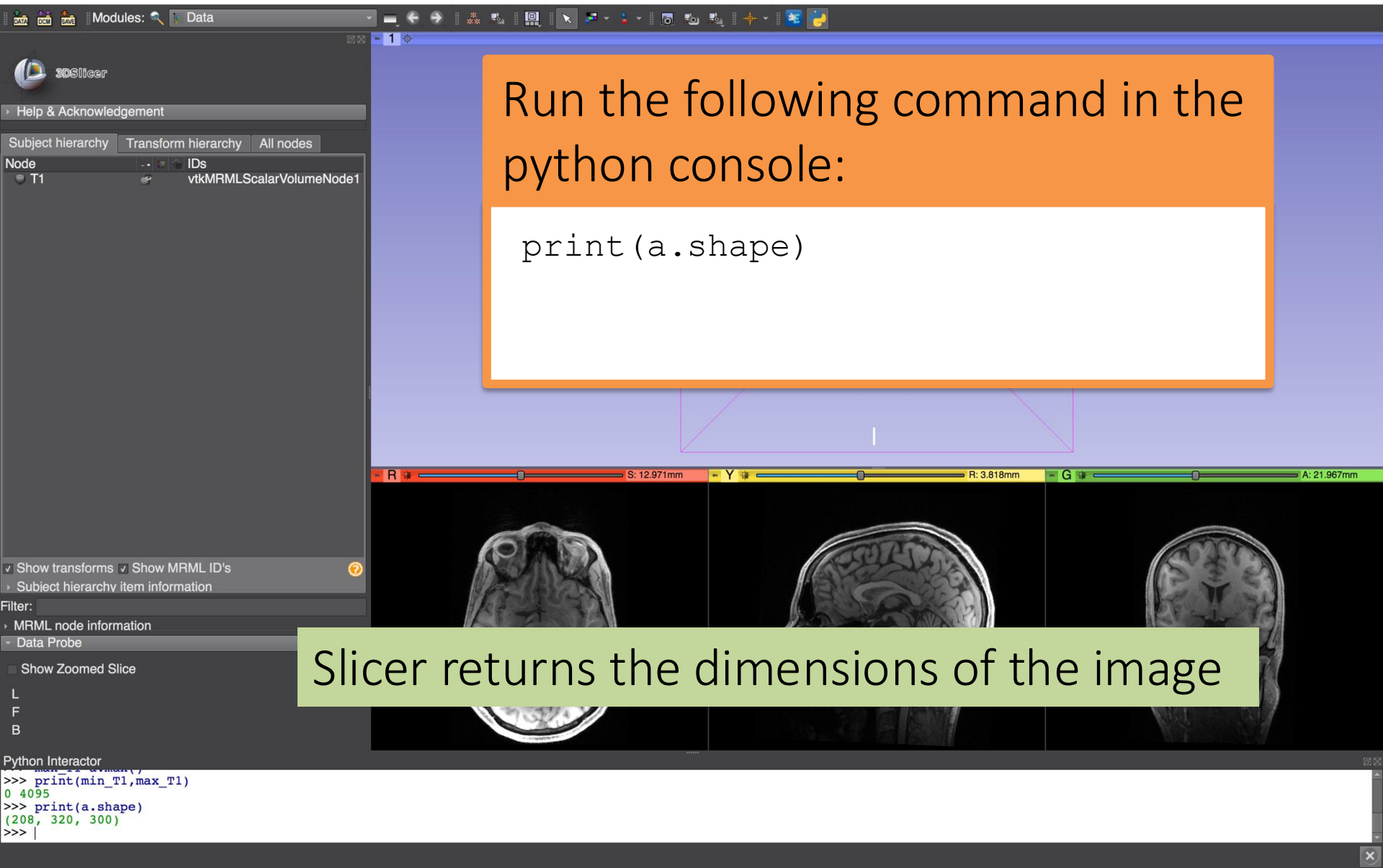
```
min_T1 = a.min()
max_T1 = a.max()
print(min_T1,max_T1)
```

The screenshot shows the 3D Slicer software interface. The left sidebar contains the 'Subject hierarchy' panel with a tree view showing 'Node' and 'T1' (vtkMRMLScalarVolumeNode1). Below this is the 'Python Interactor' panel with a console window showing the execution of the provided Python code. The main window displays three orthogonal views of a brain MRI volume: axial, sagittal, and coronal. The top of the main window features a toolbar with various icons for navigation and manipulation. The bottom of the main window shows the 'Data Probe' panel with a 'Show Zoomed Slice' checkbox and a 'Filter' dropdown menu.

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:' followed by a white box with the code `print(a.shape)`.

At the bottom, a green text box contains the text: 'Slicer returns the dimensions of the image'.

The main 3D view shows three orthogonal slices of a brain MRI volume. Above the slices are sliders for the 'R' (Right) axis (S: 12.971mm), 'Y' (Anterior-Posterior) axis (R: 3.818mm), and 'G' (Superior-Inferior) axis (A: 21.967mm).

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

Run the following commands in the python console:

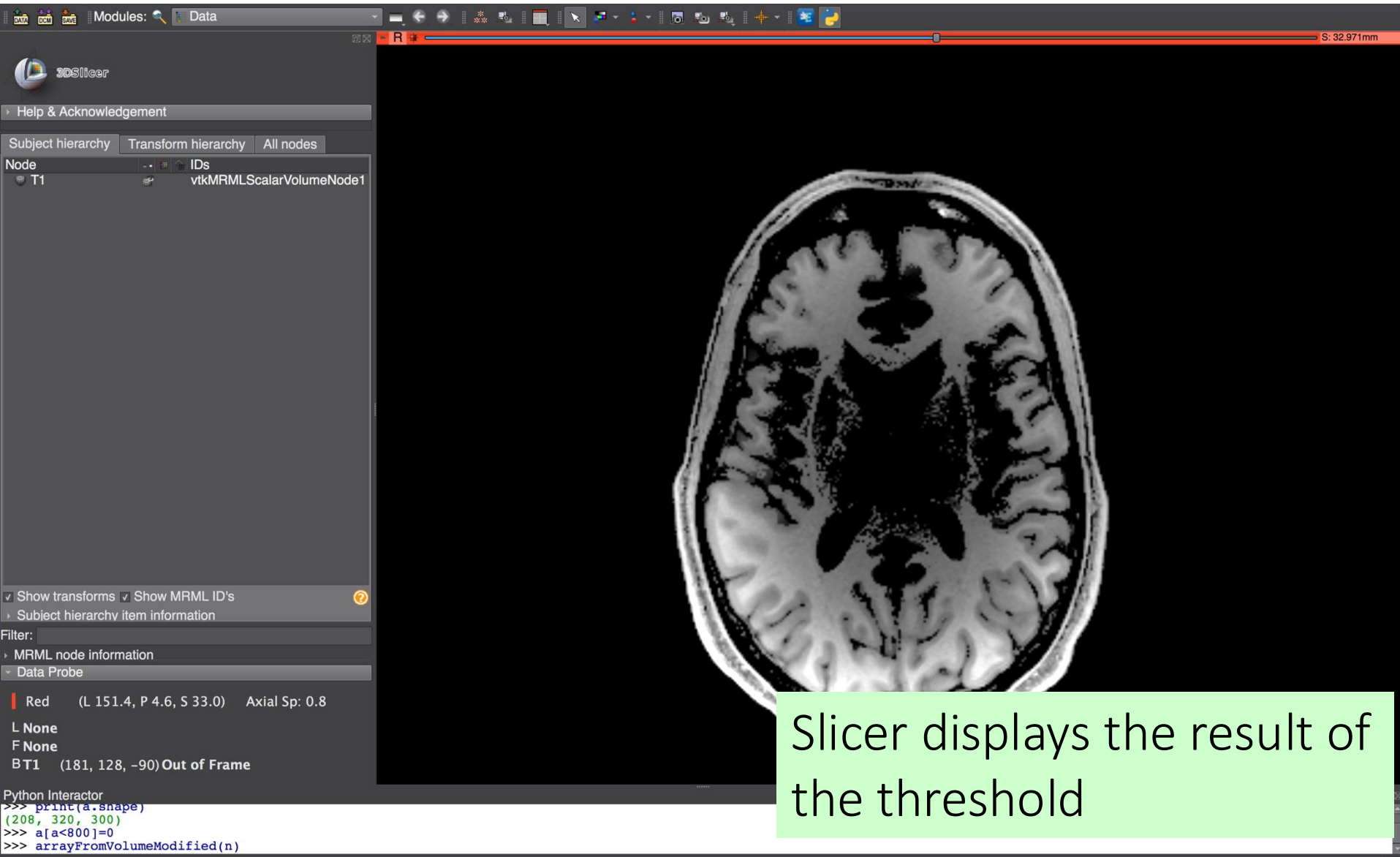
```
a[a<800]=0  
arrayFromVolumeModified(n)
```

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

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

```
Python Interactor
0 4095
>>> print(a.shape)
(208, 320, 300)
>>> a[a<800]=0
>>> arrayFromVolumeModified(n)
```

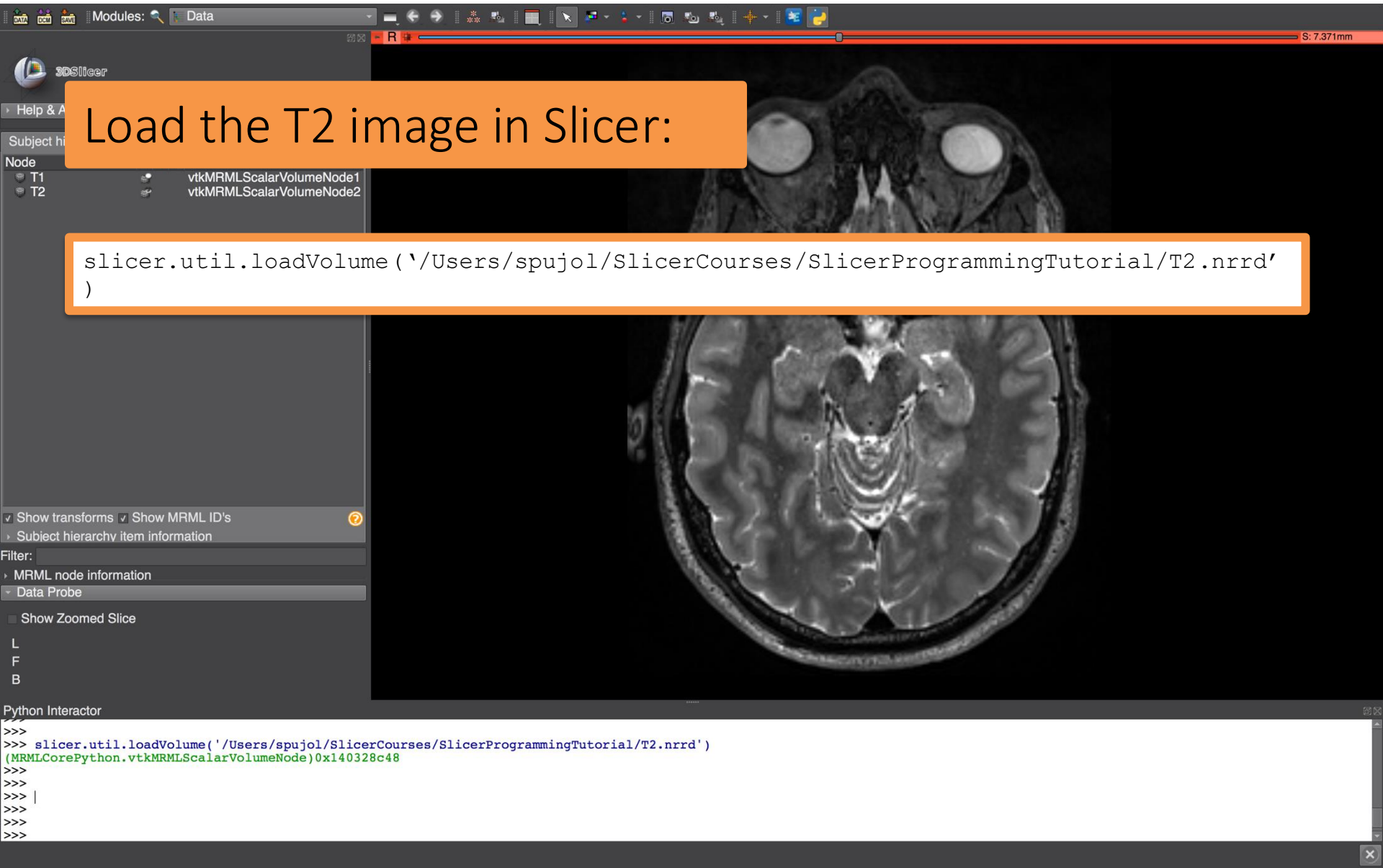
Modifying voxels in a volume



Loading the T2 volume

Load the T2 image in Slicer:

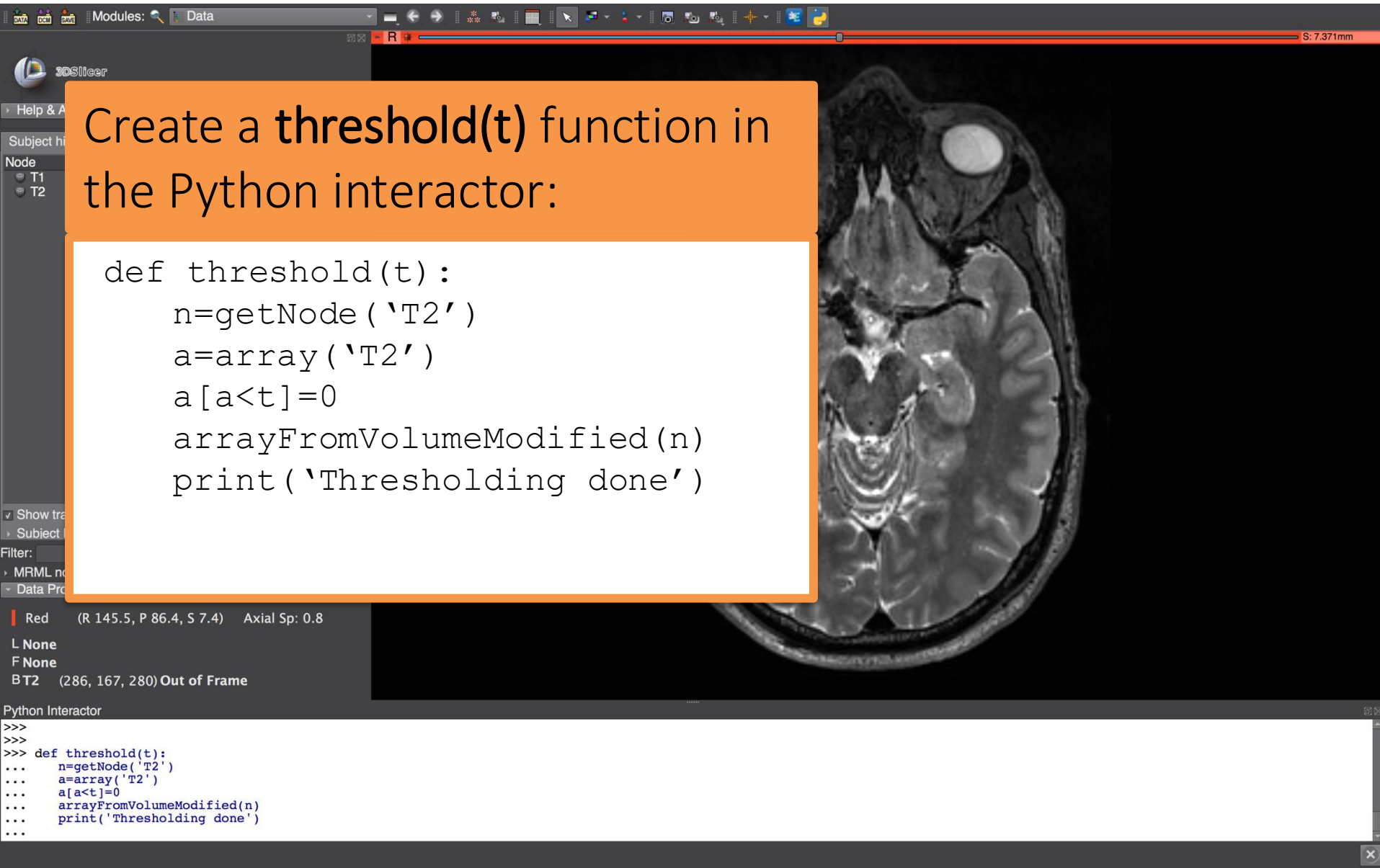
```
slicer.util.loadVolume('/Users/spujol/SlicerCourses/SlicerProgrammingTutorial/T2.nrrd')
```



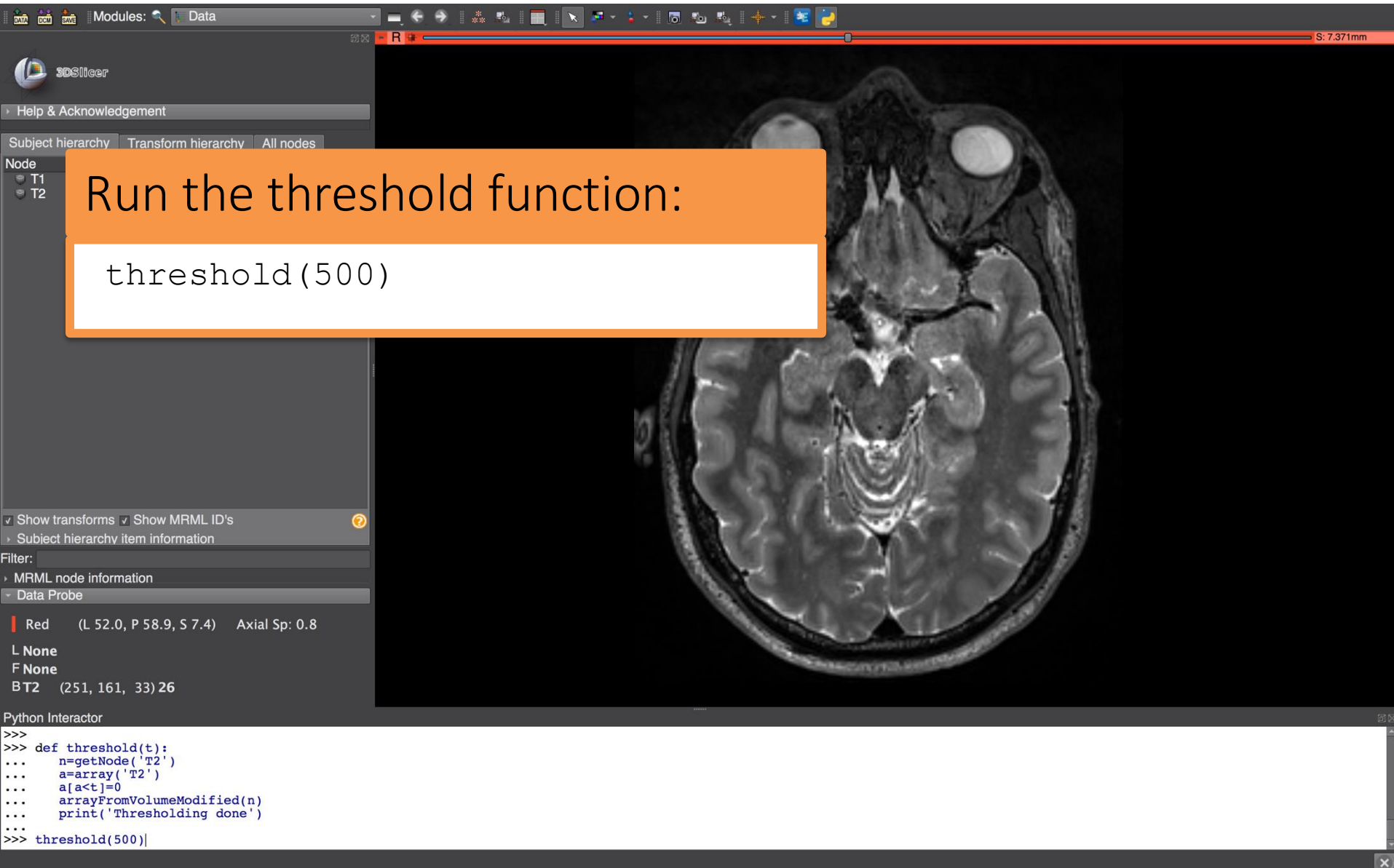
Python function: threshold

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)
```

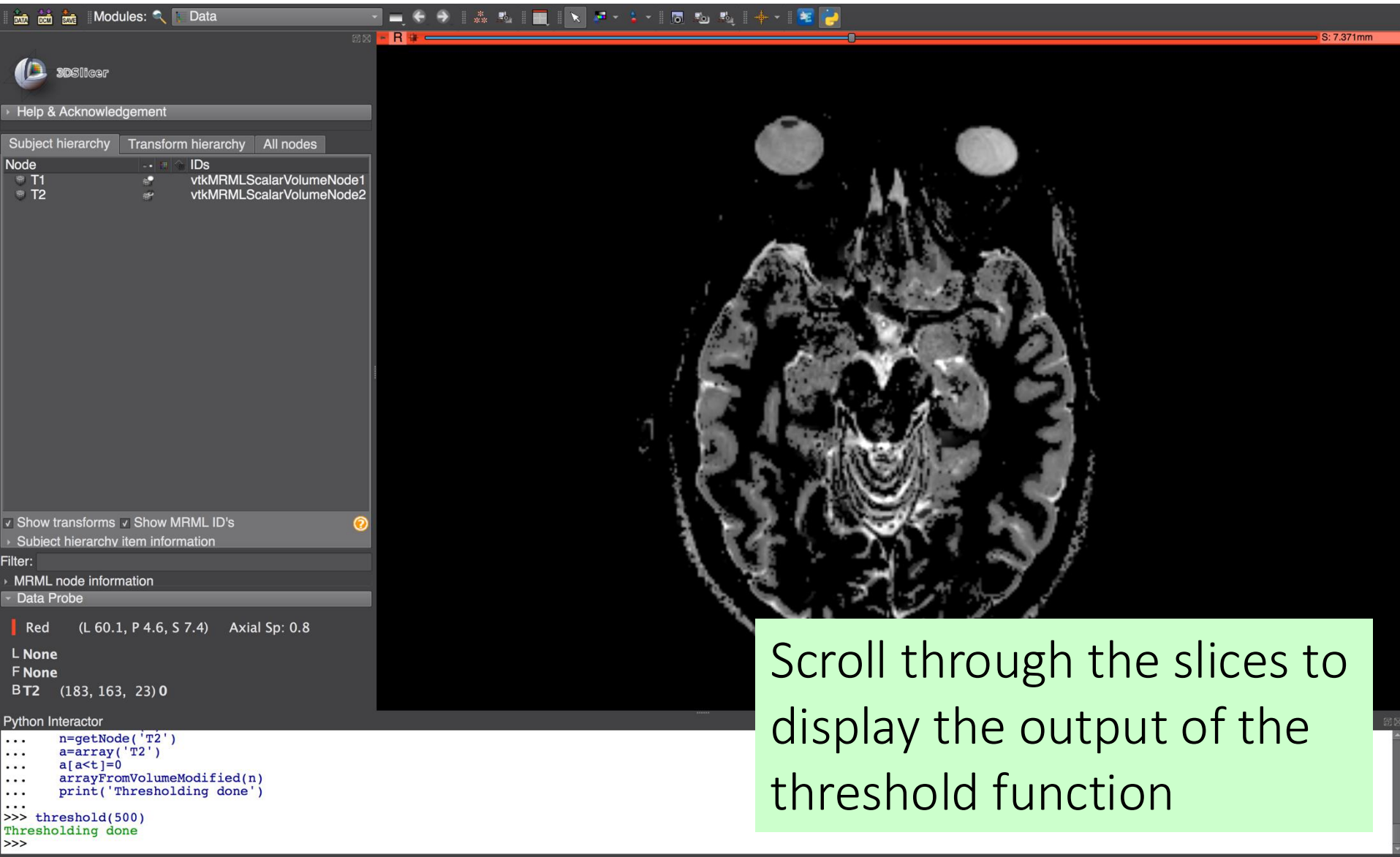
3D Slicer interface details:

- Modules: Data
- Subject hierarchy: T1, T2
- Transform hierarchy: All nodes
- Node: T1, T2
- Filter: MRML node information, Data Probe
- Red (L 52.0, P 58.9, S 7.4) Axial Sp: 0.8
- L None
- F None
- BT2 (251, 161, 33) 26

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 the 'Subject hierarchy' panel with nodes 'T1' and 'T2'. The 'T2' node is selected, and its ID is 'vtkMRMLScalarVolumeNode2'. The 'Data Probe' panel shows the current slice parameters: 'Red (L 60.1, P 4.6, S 7.4) Axial Sp: 0.8'. The 'Python Interactor' panel at the bottom shows the following code:

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

Scroll through the slices to display the output of the threshold function

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

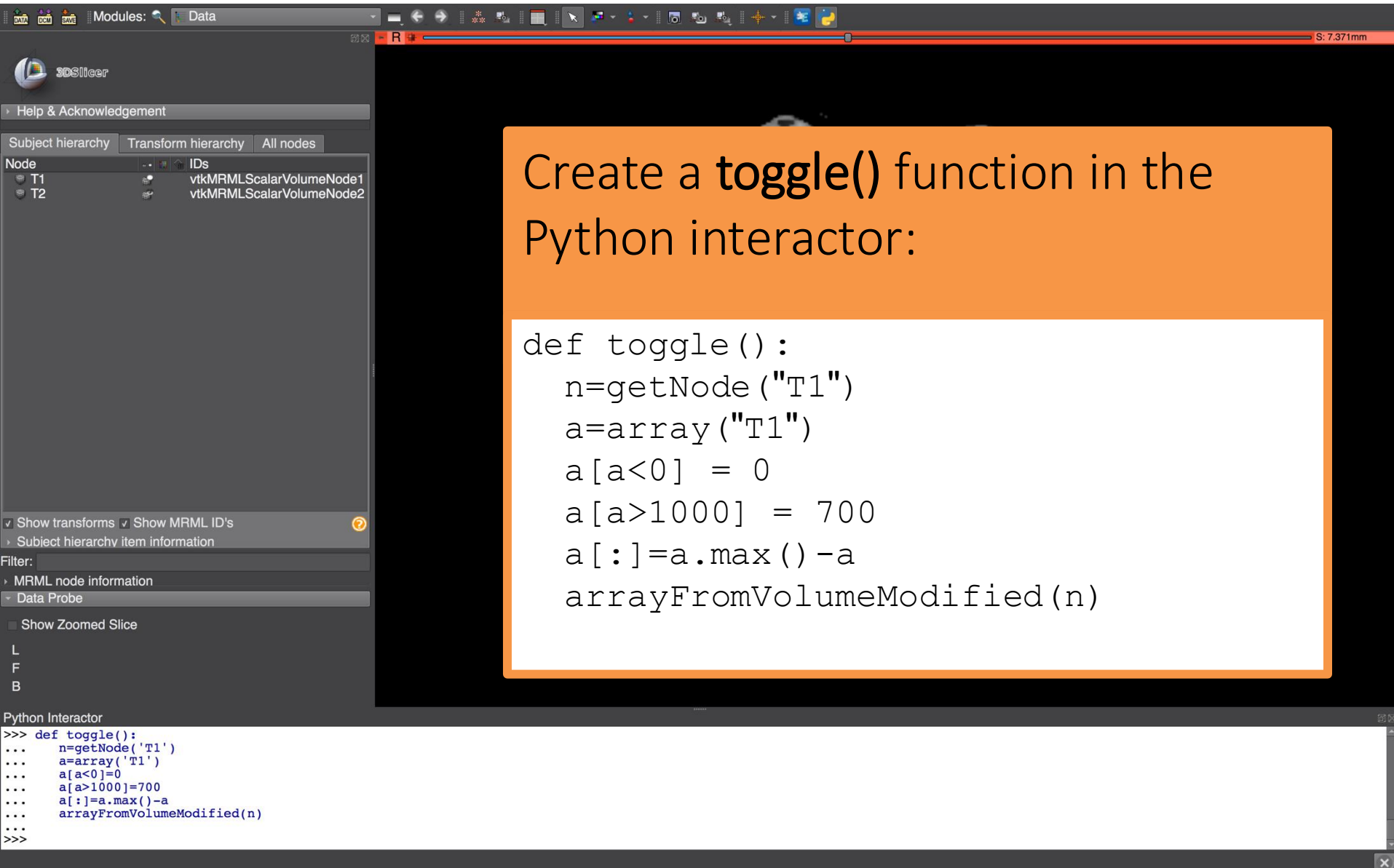
Part 3

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



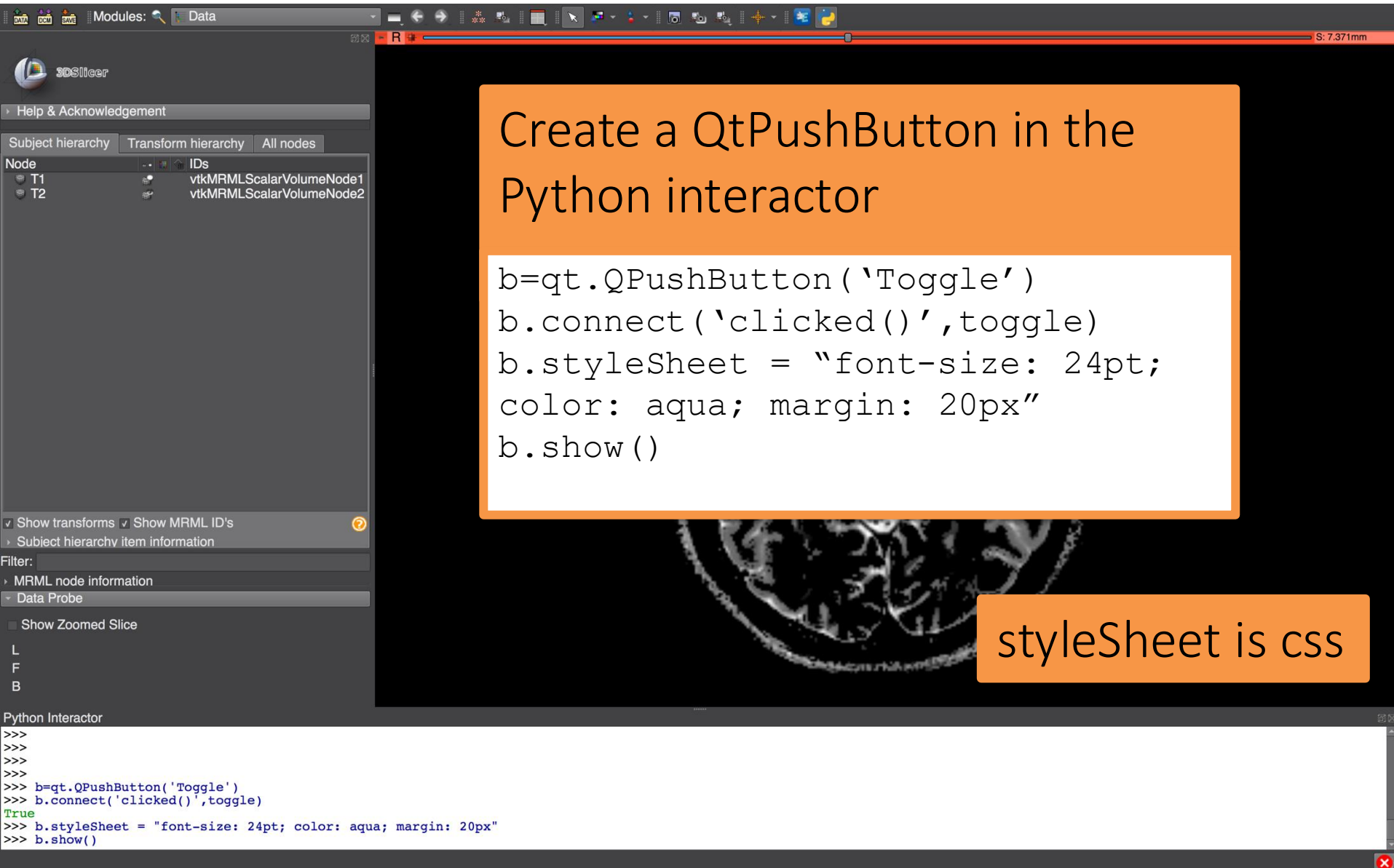
The screenshot shows the 3D Slicer software interface. On the left, the 'Subject hierarchy' panel lists nodes T1 and T2, both of type 'vtkMRMLScalarVolumeNode'. The 'Python Interactor' at the bottom displays a Python script defining a 'toggle()' function. On the right, an orange box contains the same function definition. The main 3D view area is currently empty.

Create a **toggle()** function in the Python interactor:

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

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

Creating a Qt Push Button



The screenshot shows the 3D Slicer software interface. On the left, the 'Subject hierarchy' panel lists two nodes: T1 (vtkMRMLScalarVolumeNode1) and T2 (vtkMRMLScalarVolumeNode2). Below this, the 'Data Probe' section is visible. The main 3D view area shows a grayscale axial brain slice. Overlaid on this view is an orange rectangular box containing text. At the bottom of the interface, the 'Python Interactor' console shows the execution of Python code to create and style a Qt push button.

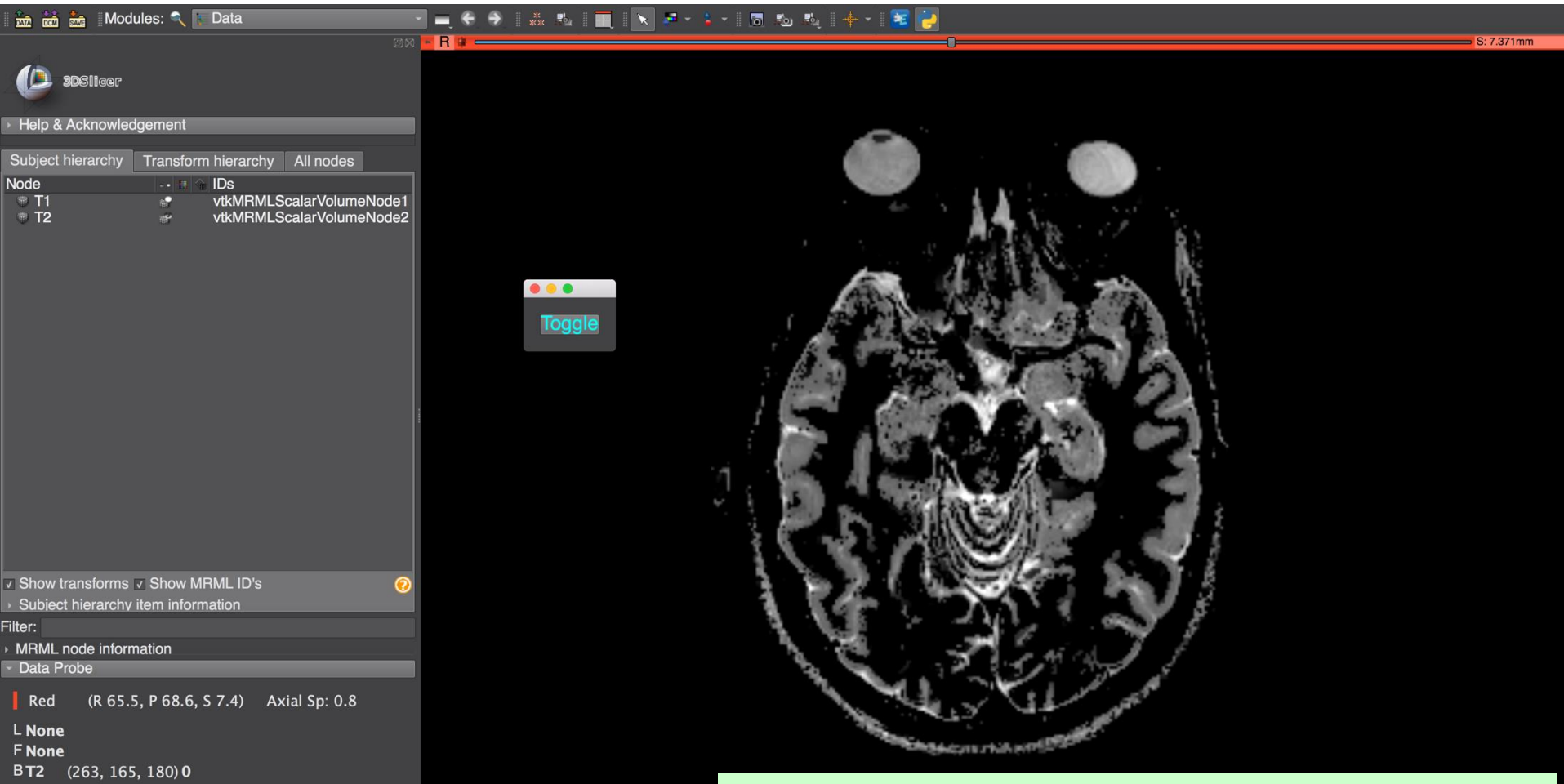
Create a QPushButton in the Python interactor

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

styleSheet is css

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

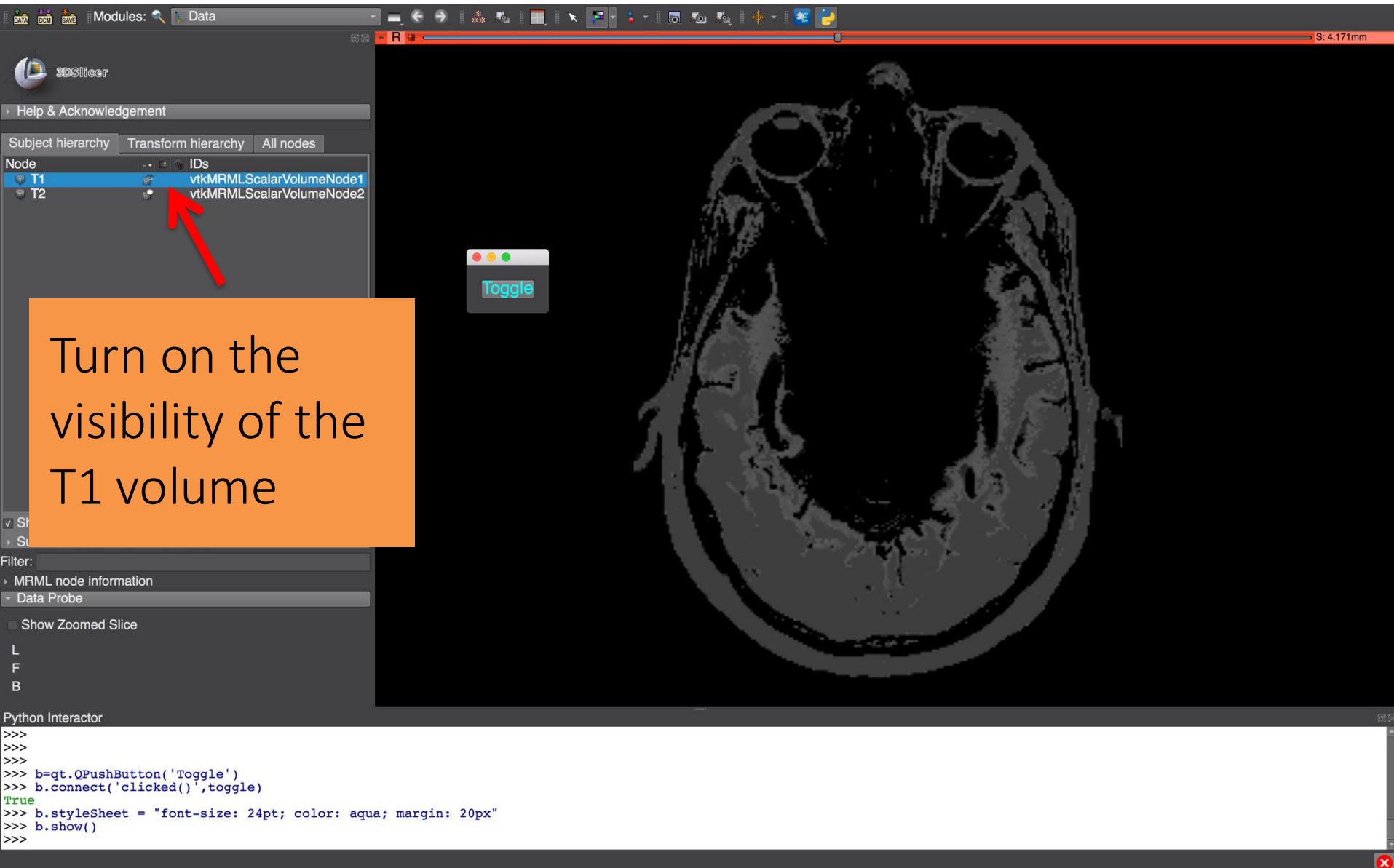
Creating a Qt Push Button



The Toggle button appears

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

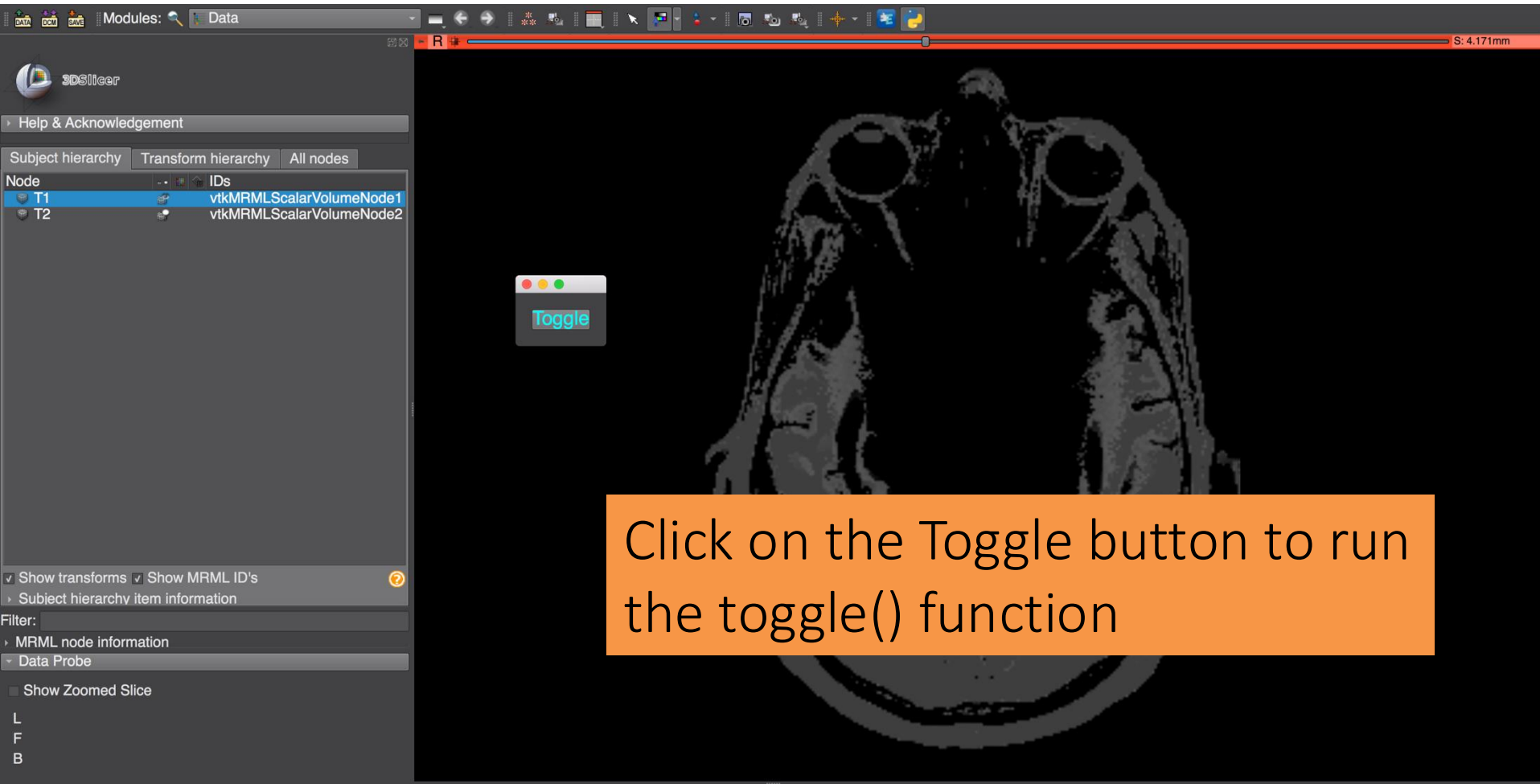
Creating a Qt Push Button



The screenshot displays the 3D Slicer software interface. On the left, the 'Subject hierarchy' panel shows a tree of nodes. A red arrow points to the 'T1' node, which is highlighted in blue. The 'T1' node is a 'vtkMRMLScalarVolumeNode1'. Below it is the 'T2' node, which is a 'vtkMRMLScalarVolumeNode2'. The main 3D view shows a grayscale axial brain MRI slice. A small Qt push button labeled 'Toggle' is visible in the 3D view. The button has a light blue background and a black border. The Python Interactor at the bottom shows the following code:

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

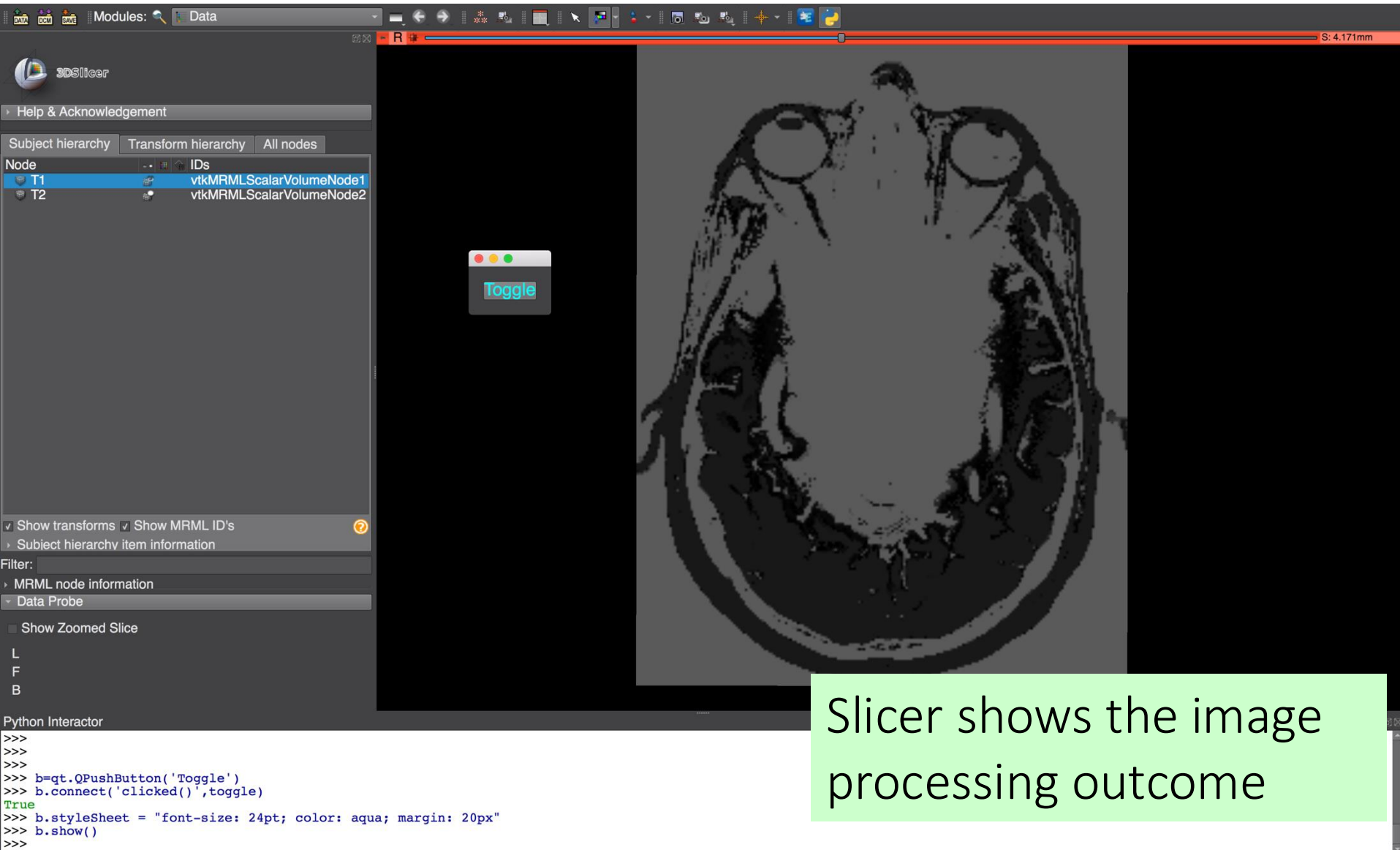
Creating a Qt Push Button



The screenshot shows the 3D Slicer software interface. On the left, the 'Subject hierarchy' panel lists two nodes: 'T1' (vtkMRMLScalarVolumeNode1) and 'T2' (vtkMRMLScalarVolumeNode2). Below this, the 'Data Probe' section is visible. The main 3D view displays a grayscale medical scan of a head. A custom Qt button, labeled 'Toggle', is overlaid on the 3D view. The button has a light blue background and a dark blue border. An orange text box is overlaid on the 3D view, 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



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://slicer.readthedocs.io/en/latest/developer_guide/script_repository.html

Conclusion

- Slicer enables developers to create complex interfaces that are streamlined for target users
- The software platform provides unlimited customization possibilities
- Slicer gives 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)