



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.setStyleSheet = "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

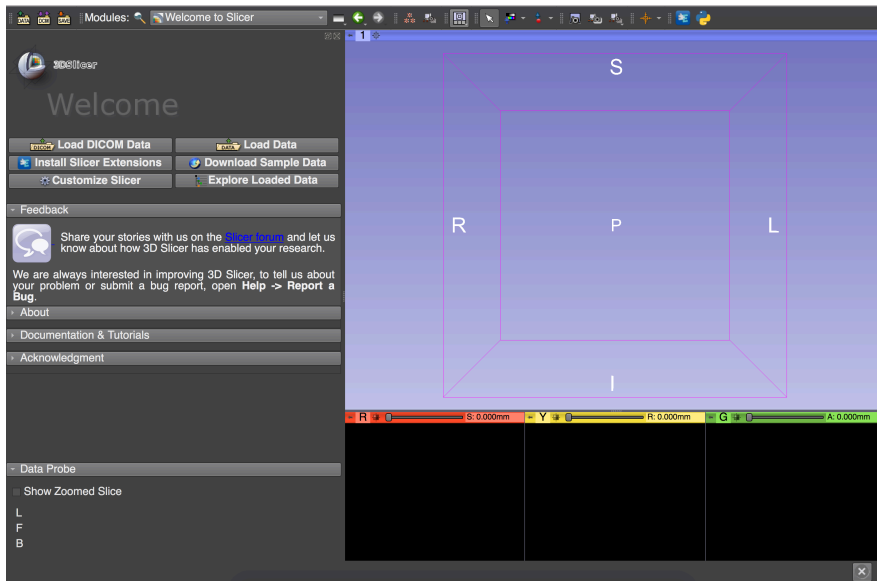


Part 3: Getting Familiar with the Qt widget toolkit in 3D Slicer

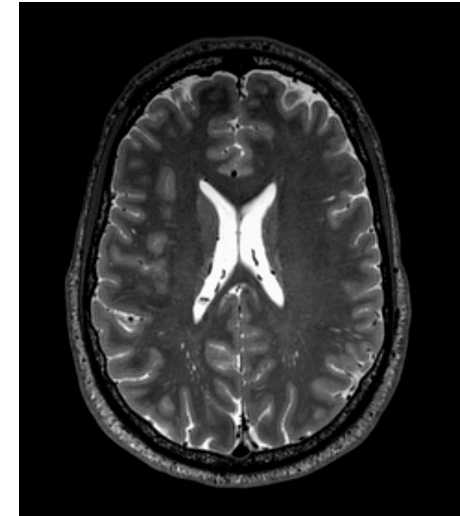
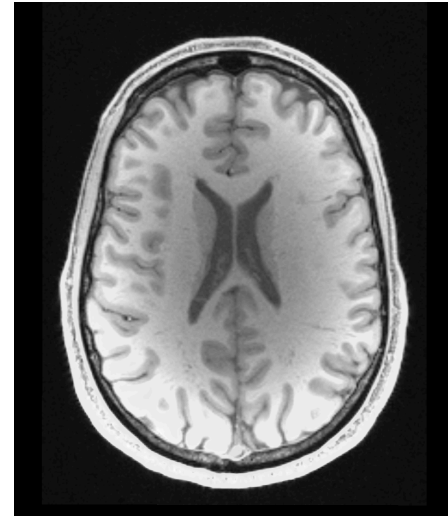
Disclaimer

- 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



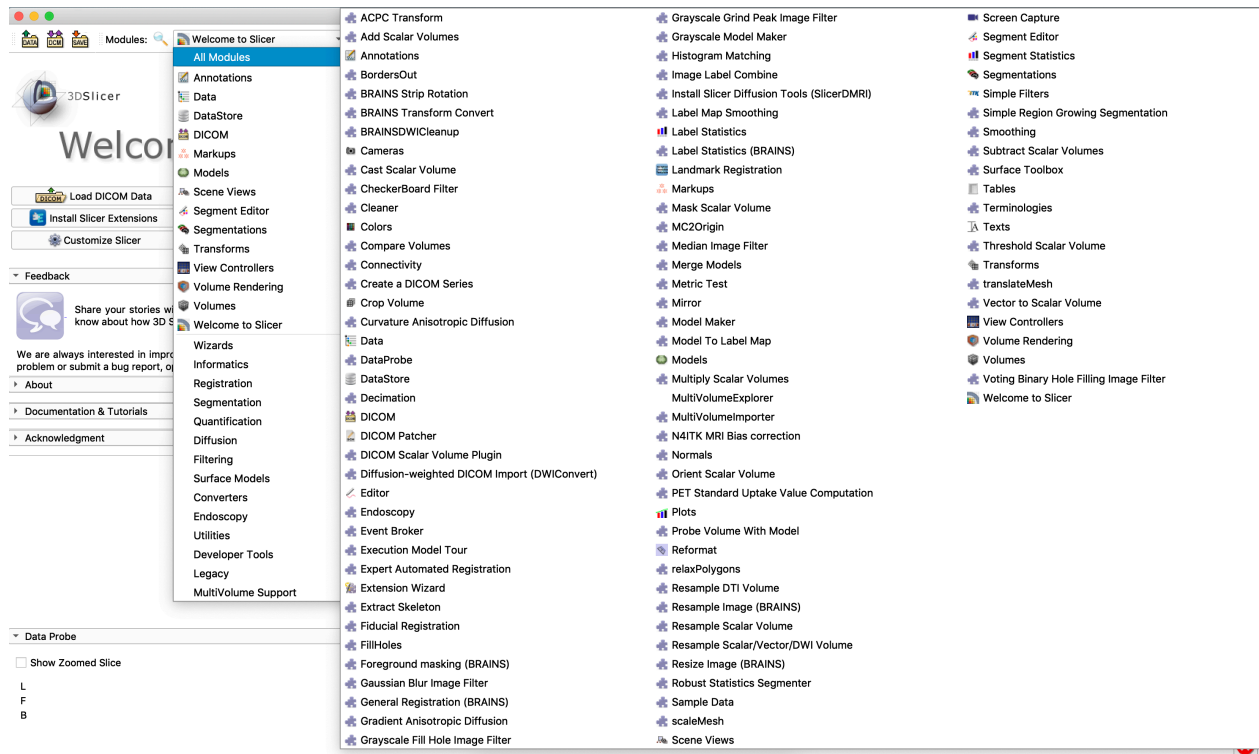
3D Slicer version 4.11



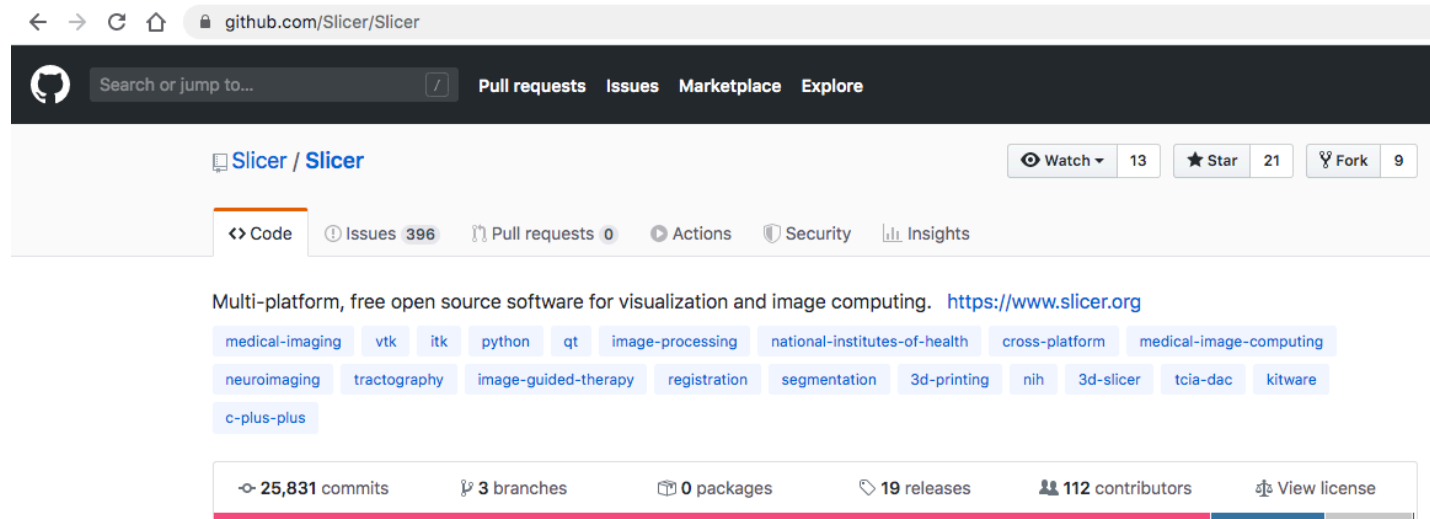
SlicerProgrammingTutorialData.zip

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

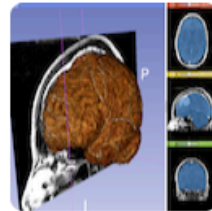
The screenshot displays the Slicer software interface with the 'All Modules' list expanded. The list contains numerous modules, each with a small icon. A large orange text box is overlaid on the right side of the list, containing the text: 'The type of module is transparent to the end-user'.

Module List:

- 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
- Welcome to Slicer
- Wizards
- Informatics
- Registration
- Segmentation
- Quantification
- Diffusion
- Filtering
- Surface Models
- Converters
- Endoscopy
- Utilities
- Developer Tools
- Legacy
- MultiVolume Support

Slicer Extensions

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



SwissSkullStripper
Bill Lorensen (Noware...)
★★★★★ (0)

INSTALL



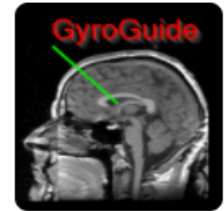
PETTumorSegmenta...
Christian Bauer (Univ...)
★★★★★ (0)

INSTALL



SlicerOpenIGTLink
Junichi Tokuda (SPL, ...)
★★★★★ (0)

INSTALL



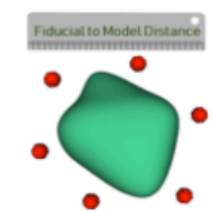
GyroGuide
Ruifeng Chen, Luping...
★★★★★ (0)

INSTALL



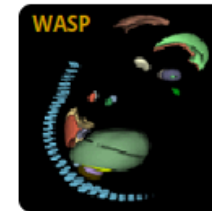
PET-IndiC
Ethan Ulrich (Universi...)
★★★★★ (0)

INSTALL



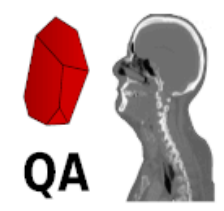
FiducialsToModelDi...
Jesse Reynolds (Cante...)
★★★★★ (0)

INSTALL



Slicer-Wasp
Thomas Lawson (MR...)
★★★★★ (0)

INSTALL

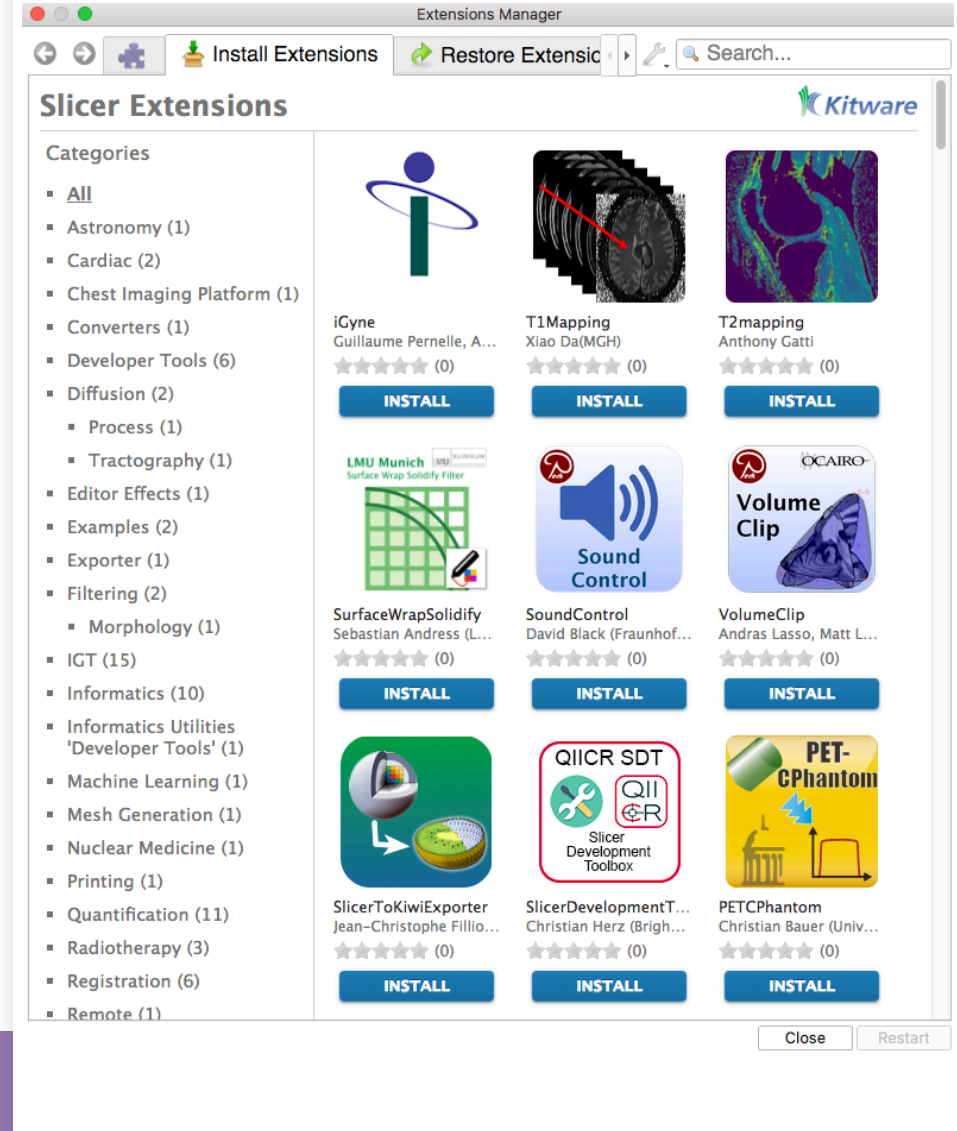


ImageCompare
Paolo Zaffino (Magna ...)
★★★★★ (0)

INSTALL

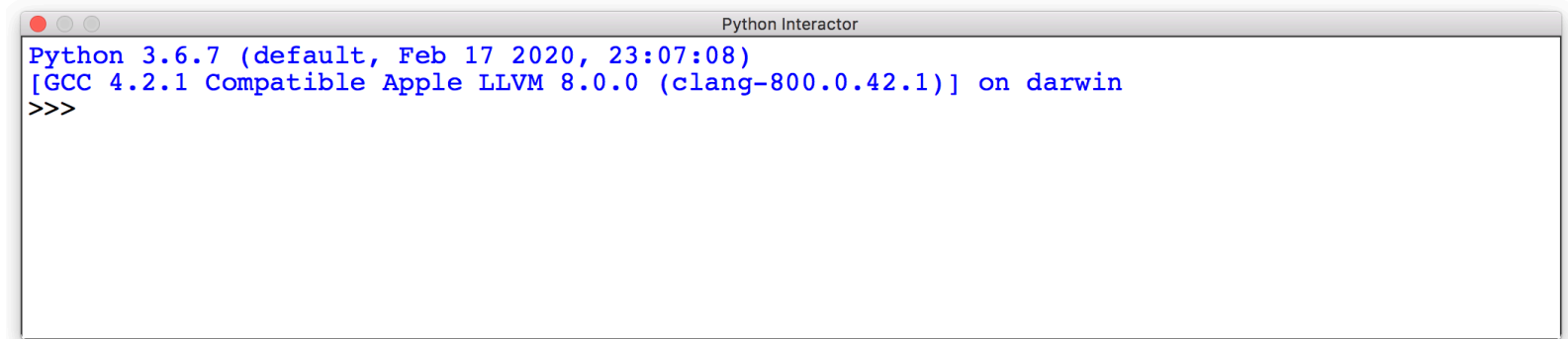
Slicer Extension Manager

- 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 v.4.11 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.



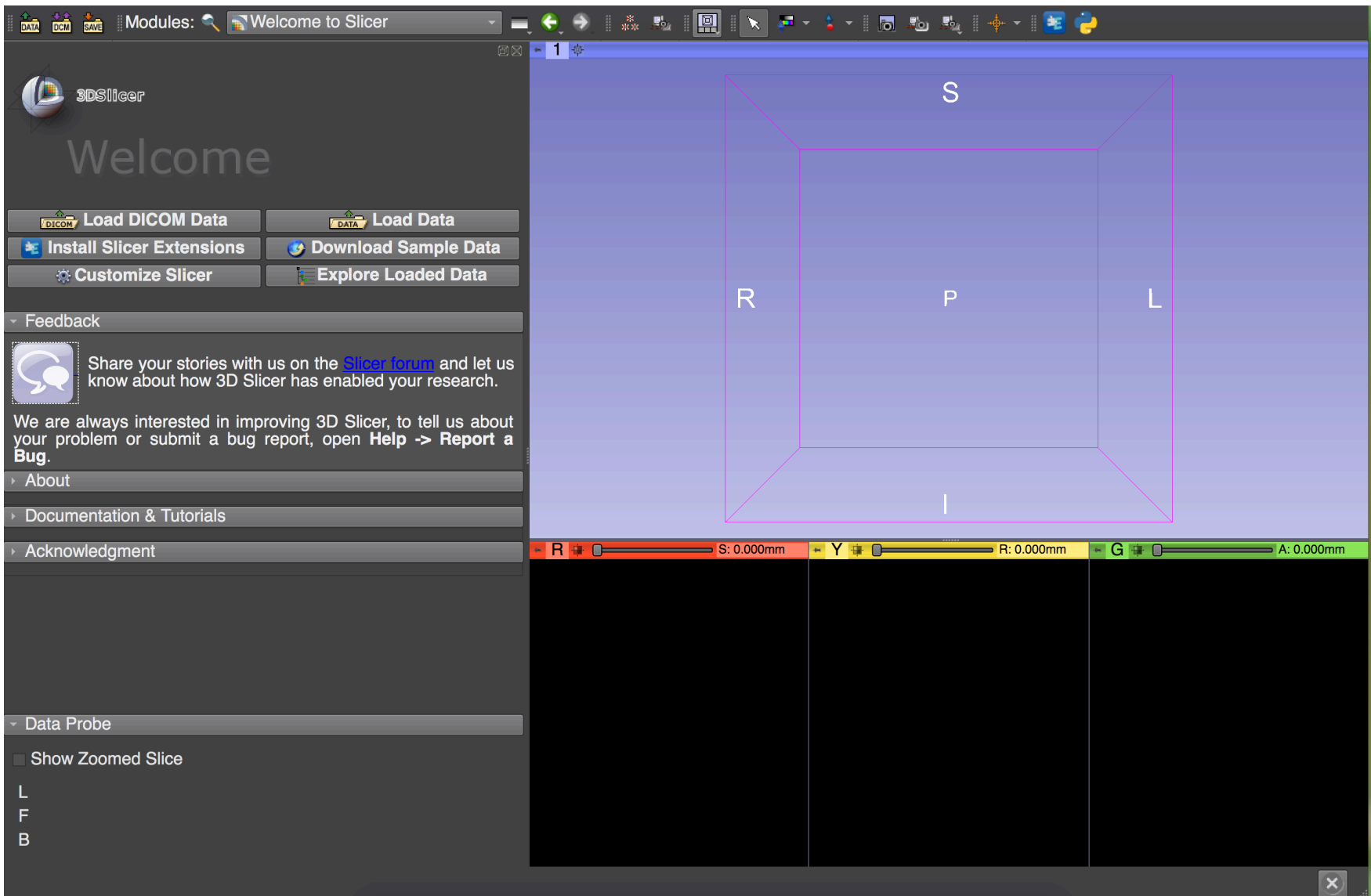
Qt is a cross-platform framework used as a graphical toolkit.

Python in Slicer



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

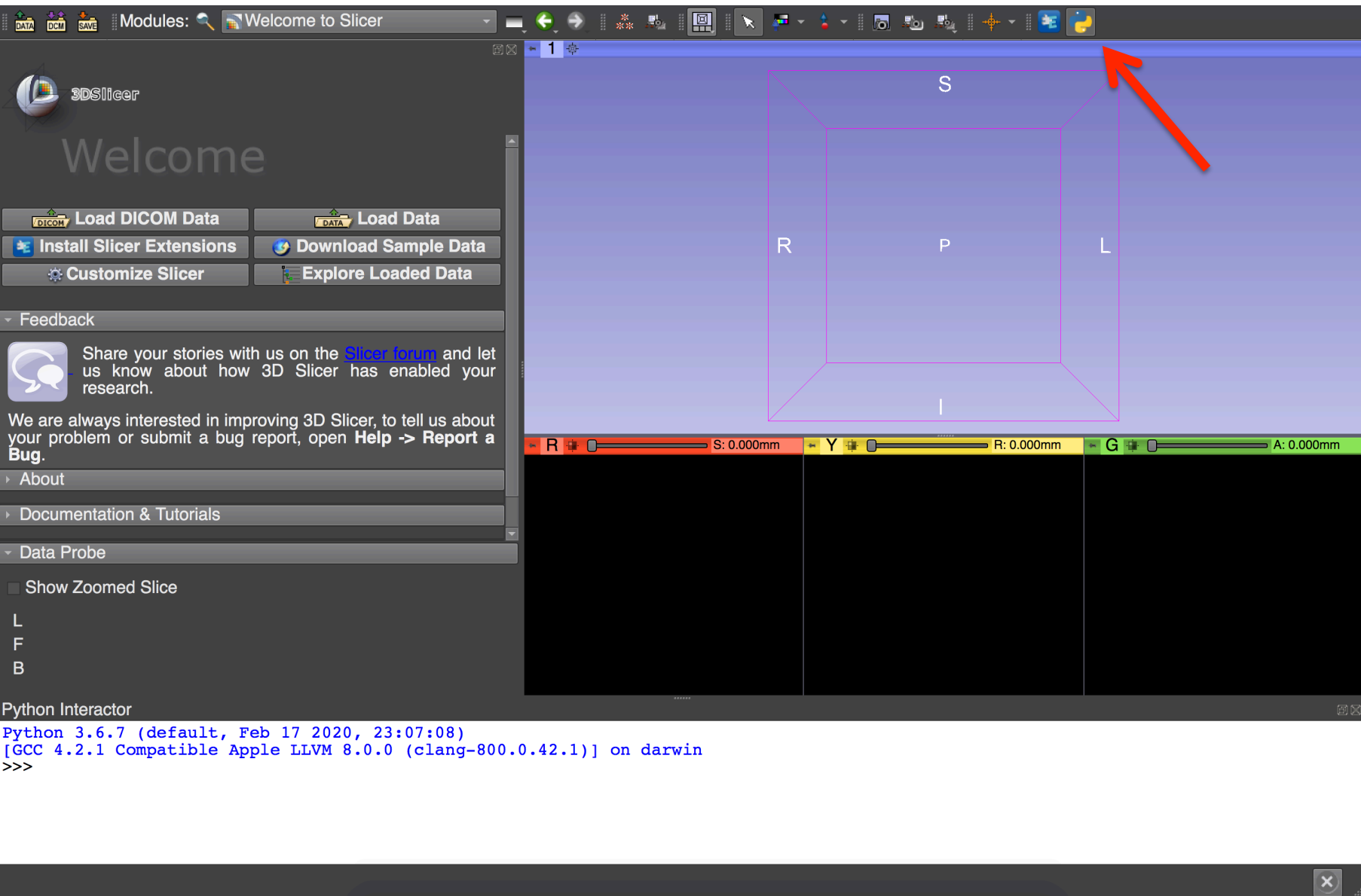
- The **pip install** command in Slicer enables developers to install most common scientific computing tools (e.g. TensorFlow, SciPy, PyTorch, Pandas, etc.)
- Slicer can be used as a **Jupyter notebook** kernel
- PyCharm and other Python development tools can be used with Slicer




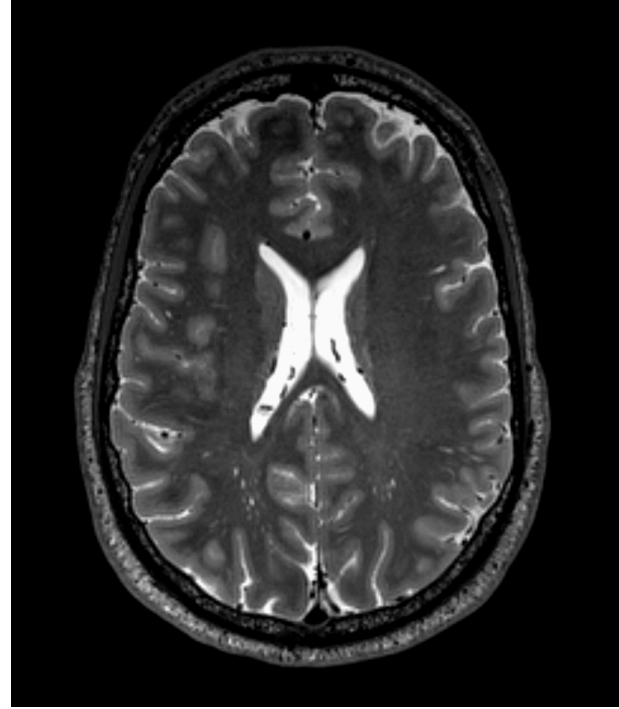
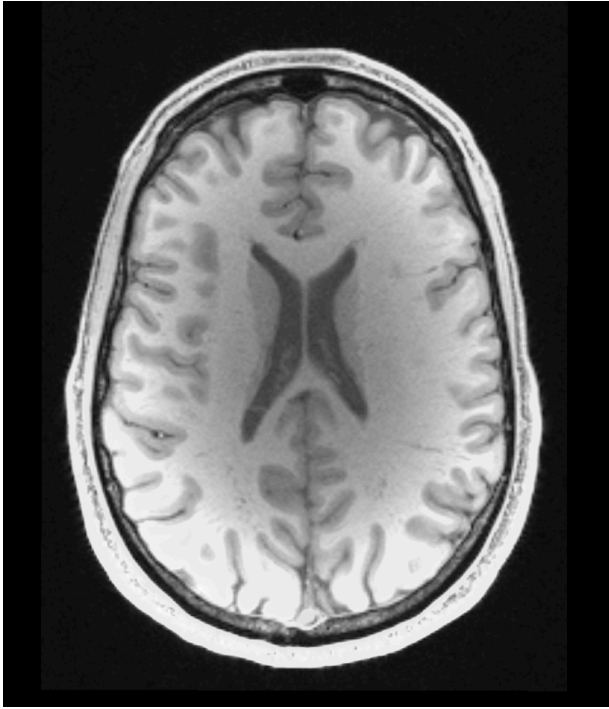
Slicer release version 5 integrates Python3, VTK5 and ITK5

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.

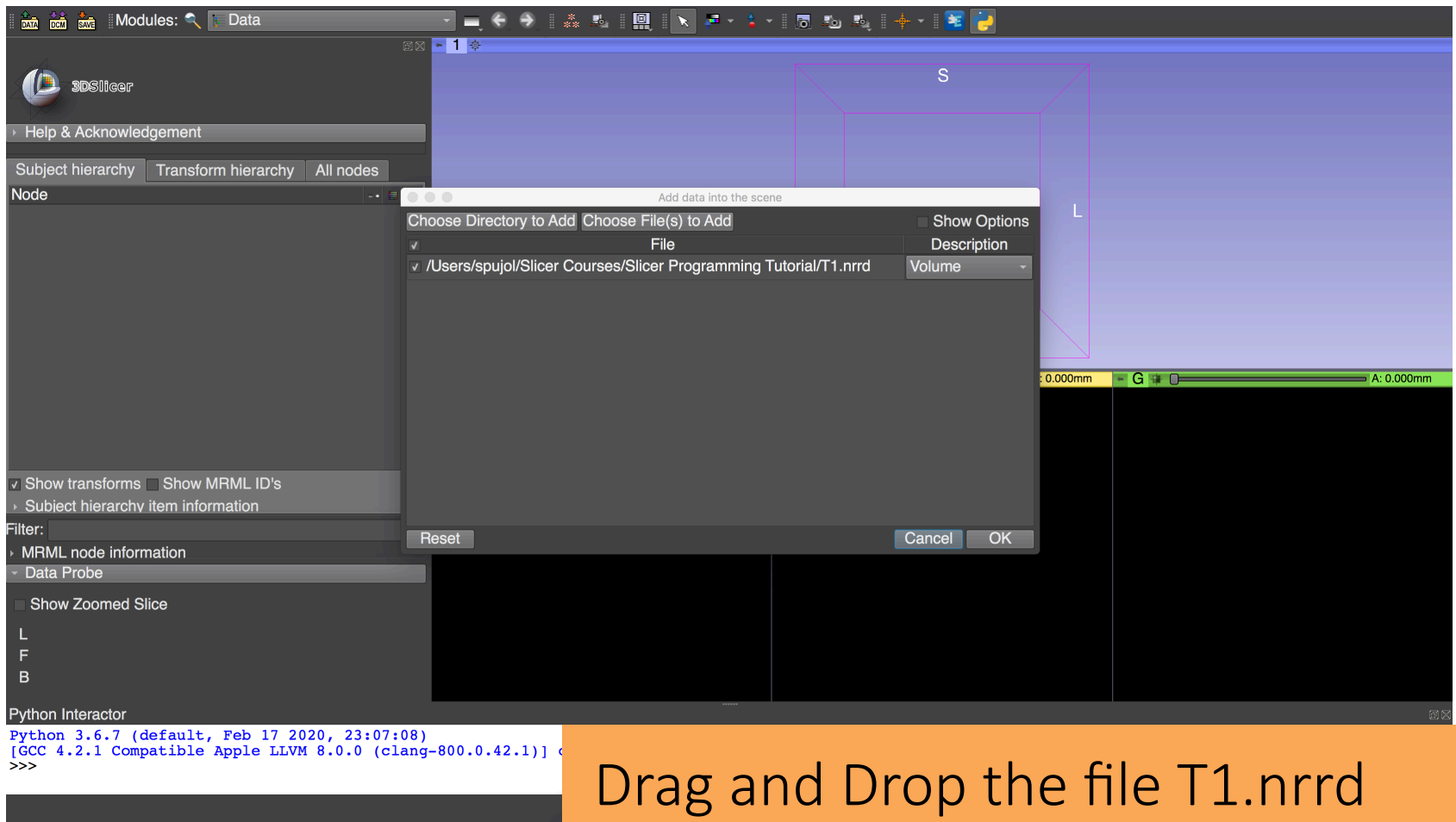


To access the Python Interactor, click on the Python icon  in the top bar menu of Slicer.



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

The screenshot displays the 3DSlicer software interface. At the top, the 'Modules' menu is set to 'Data'. The main workspace shows a 3D coordinate system with axes labeled S (Superior), I (Inferior), R (Right), and L (Left). Below this, three MRI slices are visible: an axial slice (left), a sagittal slice (middle), and a coronal slice (right). The slice positions are indicated by sliders at the bottom of the workspace: S: 12.971mm, R: 3.818mm, and A: 21.967mm.

On the left side, the 'Subject hierarchy' panel shows a tree structure with a single node labeled 'T1'. Below this, there are checkboxes for 'Show transforms' (checked) and 'Show MRML ID's' (unchecked). A 'Filter' section is also present, with 'MRML node information' and 'Data Probe' expanded. At the bottom left, a 'Python Interactor' window shows the following text:

```
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
>>>
```

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



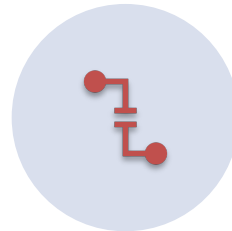
The **Slicer Data Model** is based on the Slicer Scene Data Structure



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**

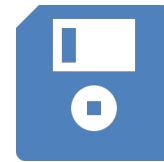
Slicer MRML Nodes: Basic Types



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

The screenshot displays the 3DSlicer software interface. The main window shows a 3D view of a brain scan with axes labeled S (Superior), R (Right), P (Posterior), L (Left), and I (Inferior). Below the 3D view are three 2D slices: an axial slice (left), a sagittal slice (middle), and a coronal slice (right). The interface includes a top toolbar, a left sidebar with a 'Node' list (showing 'T1'), and a bottom Python Interactor window. An orange box with the text 'Select Show MRML ID's' and a red arrow points to the 'Show MRML ID's' checkbox in the sidebar.

3DSlicer

Help & Acknowledgement

Subject hierarchy Transform hierarchy All nodes

Node

- T1

Show transforms Show MRML ID's

Subject hierarchy item information

Filter:

- MRML node information
- Data Probe

Show Zoomed Slice

L
F
B

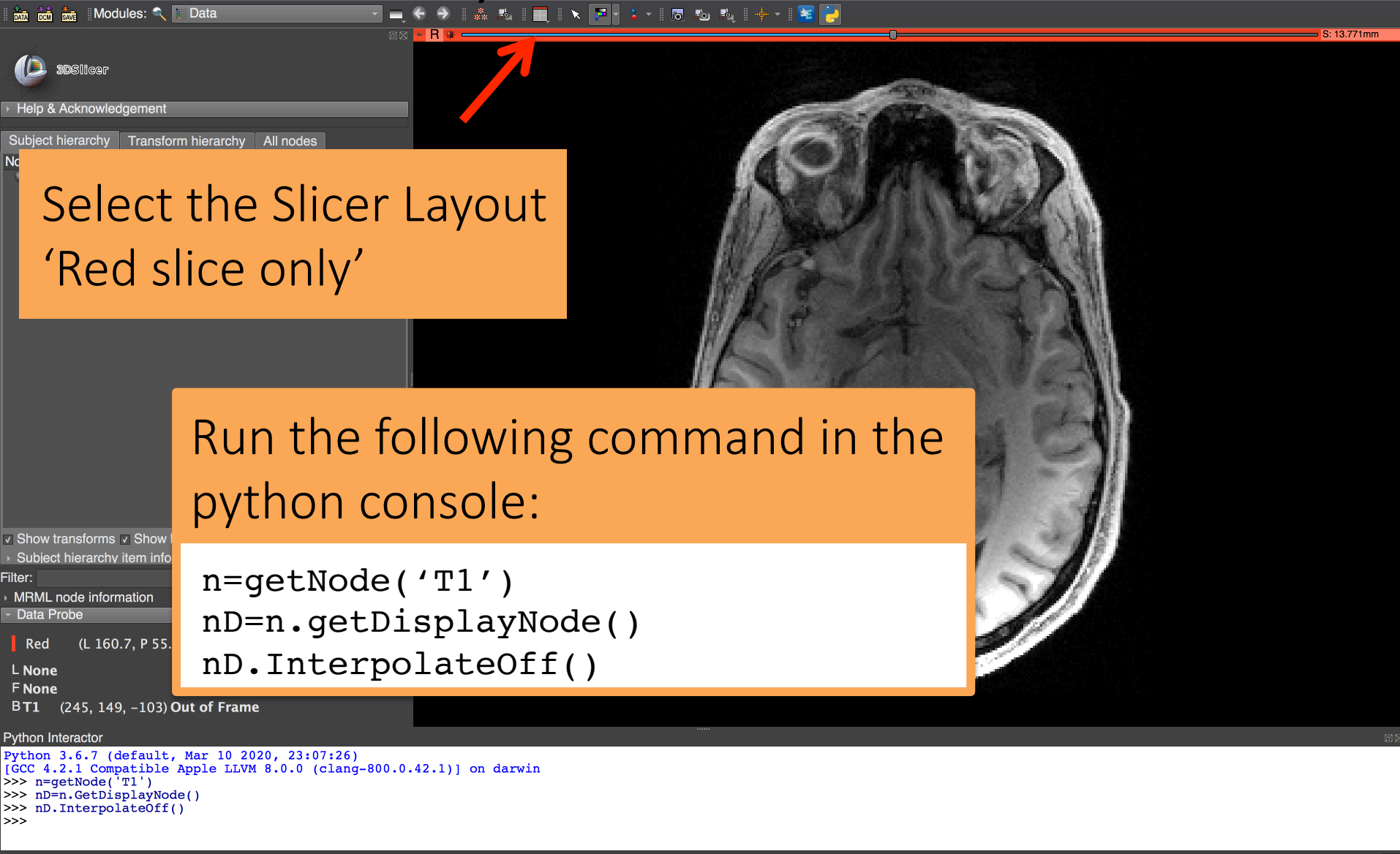
Python Interactor

```
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  
>>>
```


Slicer Data Model

The screenshot displays the Slicer software interface. The main window shows a 3D view of a brain volume with axes labeled S (Superior), R (Right), P (Posterior), and L (Left). A red arrow points to the node ID 'vtkMRMLScalarVolumeNode1' in the 'All nodes' panel. A green text box overlaid on the 3D view contains the text: 'Slicer displays the node ID of T1 vtkMRMLScalarVolumeNode1'. Below the 3D view, three 2D slices of the brain are shown: axial, sagittal, and coronal. The bottom of the interface shows a Python Interactor with the following text: '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 >>>'. The top of the interface shows the 'Modules' menu with 'Data' selected.

Accessing MRML nodes from the Python interactor



The image shows a screenshot of the 3D Slicer software interface. The main window displays a brain MRI slice. A red arrow points to the 'R' icon in the top toolbar, which represents the 'Red slice only' layout. An orange box contains the text: 'Select the Slicer Layout 'Red slice only''. Another orange box contains the text: 'Run the following command in the python console:'. Below this, a white box contains the following Python code:

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

 At the bottom of the interface, the Python Interactor window shows the execution of these commands in a Python 3.6.7 environment.

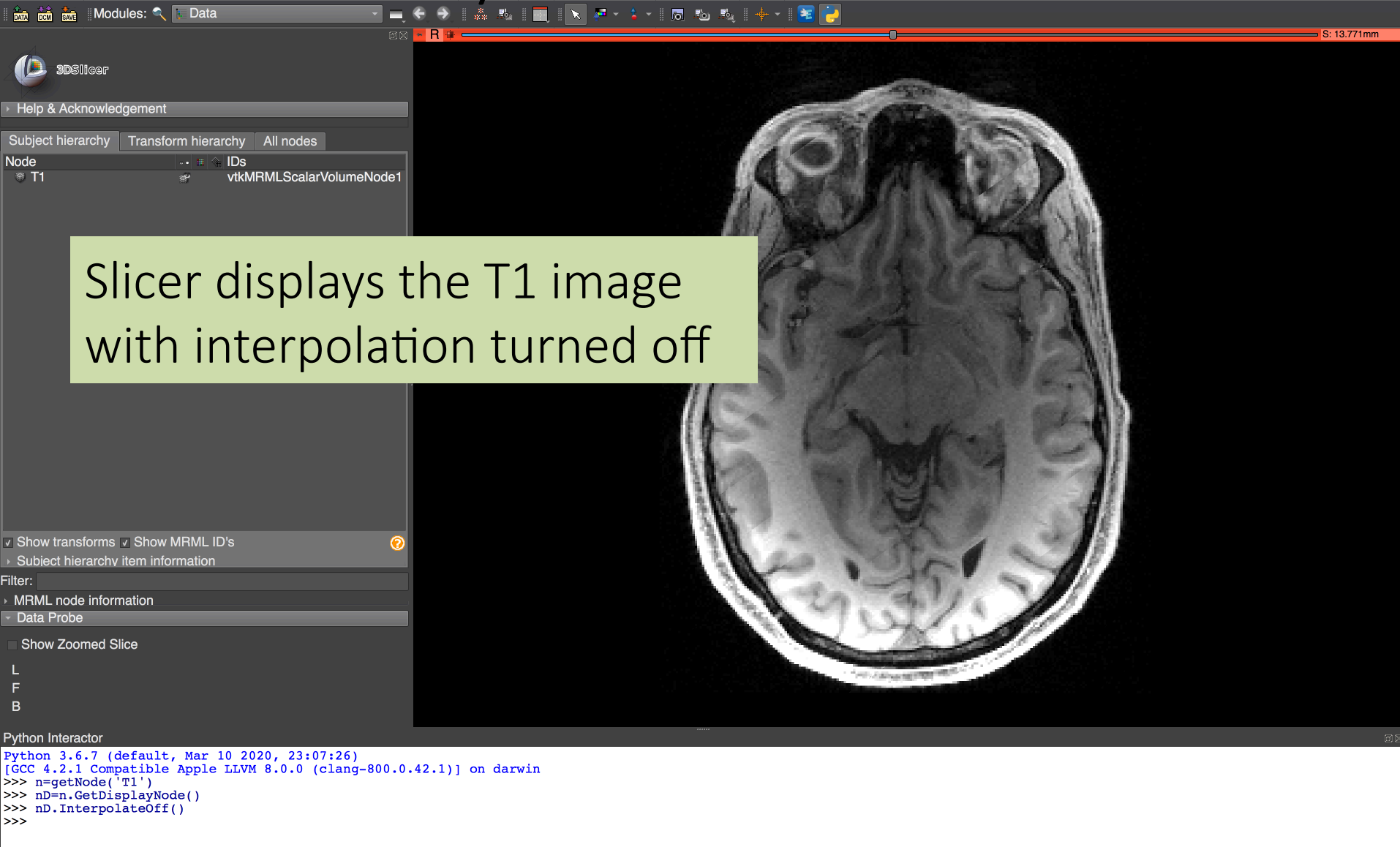
Select the Slicer Layout 'Red slice only'

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()
>>>

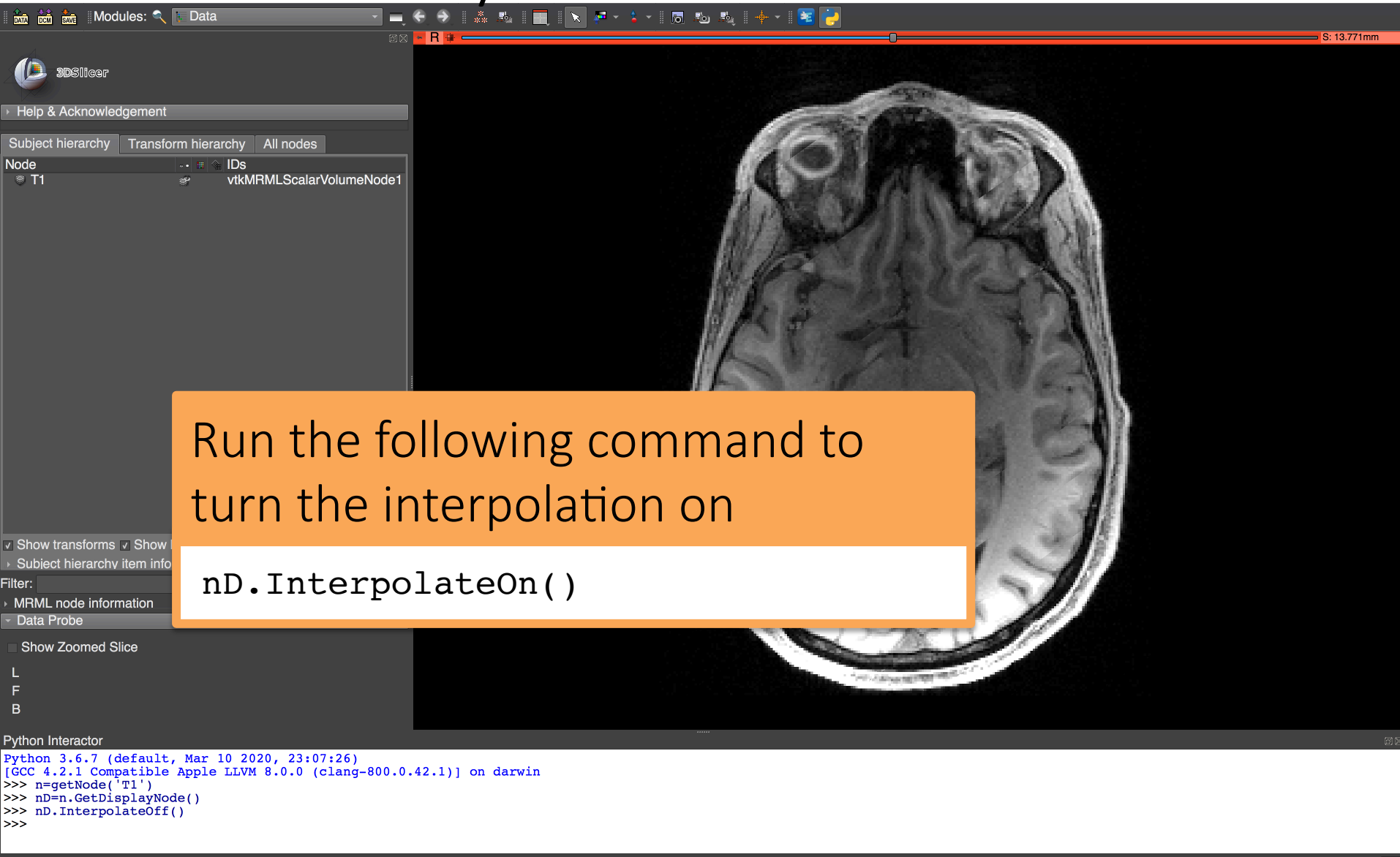
Accessing MRML nodes from the Python interactor



Slicer displays the T1 image with interpolation turned off

```
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

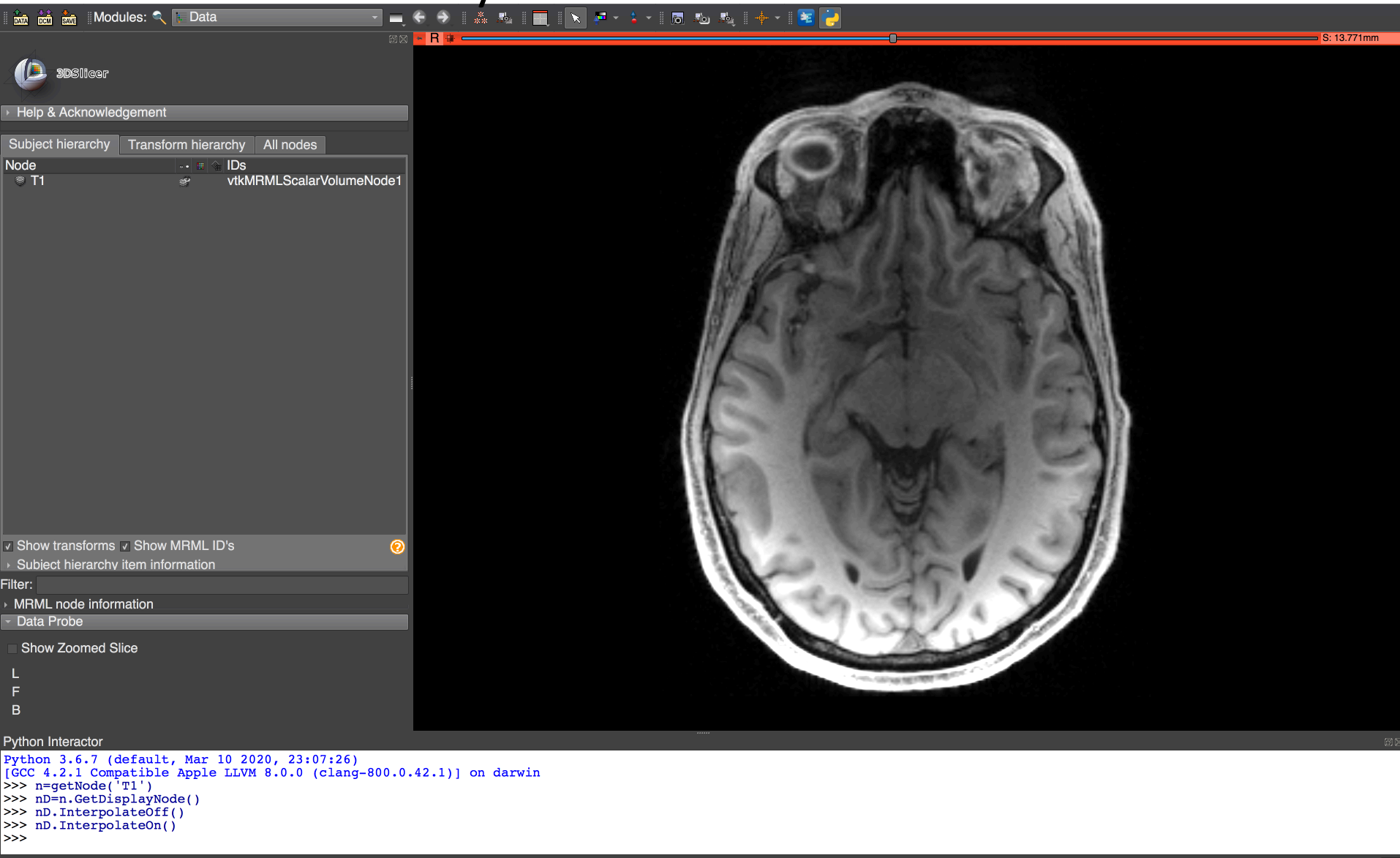


Run the following command to turn the interpolation on

```
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 3DSlicer software interface. The main window shows an axial MRI slice of a brain. On the left, the 'Subject hierarchy' panel is visible, showing a tree structure with 'T1' selected. Below this, the 'Data Probe' section is expanded, showing 'Show Zoomed Slice' checked. At the bottom, the 'Python Interactor' terminal is open, displaying the following code and output:

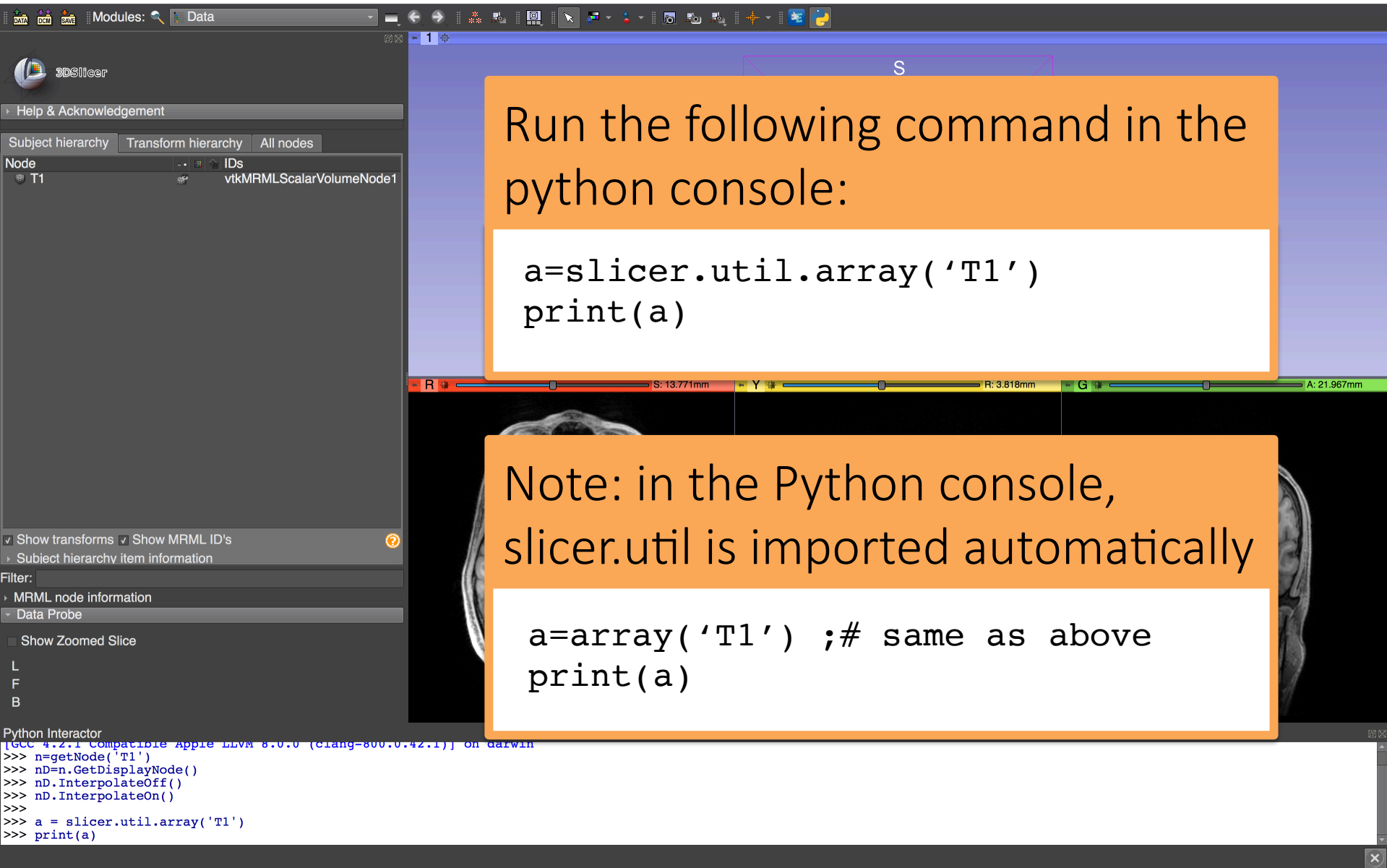
```
Python 3.6.7 (default, Mar 10 2020, 23:07:26)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.42.1)] on darwin
>>> n=getNode('T1')
>>> nd=n.GetDisplayNode()
>>> nd.InterpolateOff()
>>> nd.InterpolateOn()
>>>
```

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



The image shows a screenshot of the Slicer software interface. The main window displays a 3D view of a brain scan with a red box labeled 'S' indicating a slice. The left sidebar shows the 'Subject hierarchy' with a tree view containing 'Node T1' and 'vtkMRMLScalarVolumeNode1'. The bottom of the interface features a 'Python Interactor' window with a terminal-like environment.

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 image shows a screenshot of the Slicer software interface. On the left, a Python Interactor window displays the following code and output:

```
>>> 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]]]

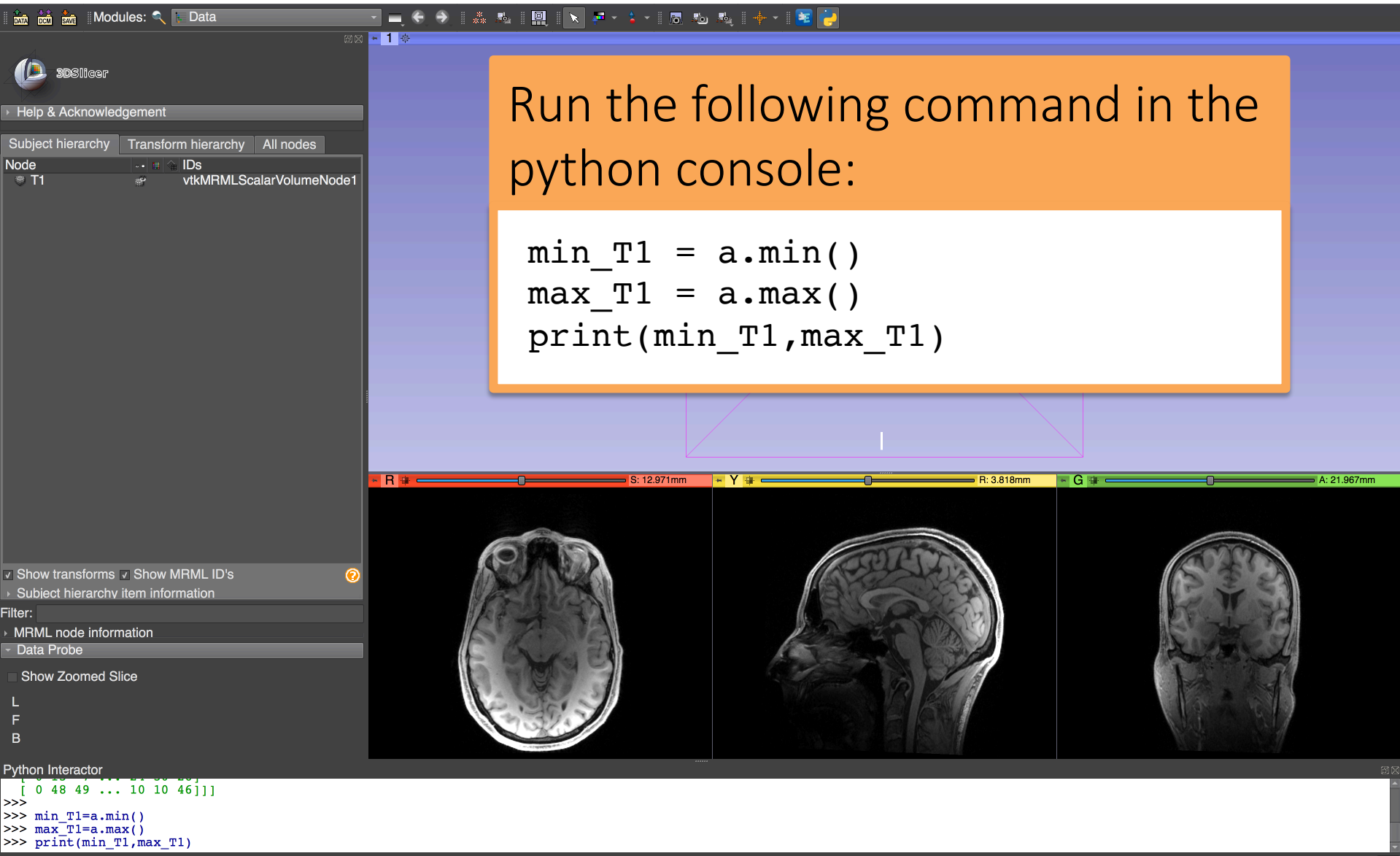
[[[ 0  0  0 ...  0  0  0]
 [ 0  5 41 ... 11  9 48]
 [ 0 21 64 ... 32 11  9]
 ...
 [ 0 11 33 ... 30 11 43]
 [ 0 13  7 ... 24 30 26]
 [ 0 48 49 ... 10 10 46]]]
>>>
```

On the right, the 3D view shows a brain volume with a purple bounding box. The box is labeled 'S' at the top and 'R' on the left side. Below the 3D view, there are two 2D slices of the brain. The bottom left slice is labeled 'L', 'F', and 'B'. The bottom right slice is labeled '21.967mm'. The top right slice is labeled '13.771mm'.

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



The screenshot displays the 3D Slicer software interface. On the left, the 'Subject hierarchy' panel shows a tree structure with 'T1' selected. The main window shows three orthogonal MRI slices: axial, sagittal, and coronal. The bottom panel is the 'Python Interactor', which contains a code editor with the following Python code:

```
[ 0 15 7 11 1 30 20]
[ 0 48 49 ... 10 10 46]]
>>>
>>> min_T1=a.min()
>>> max_T1=a.max()
>>> print(min_T1,max_T1)
```

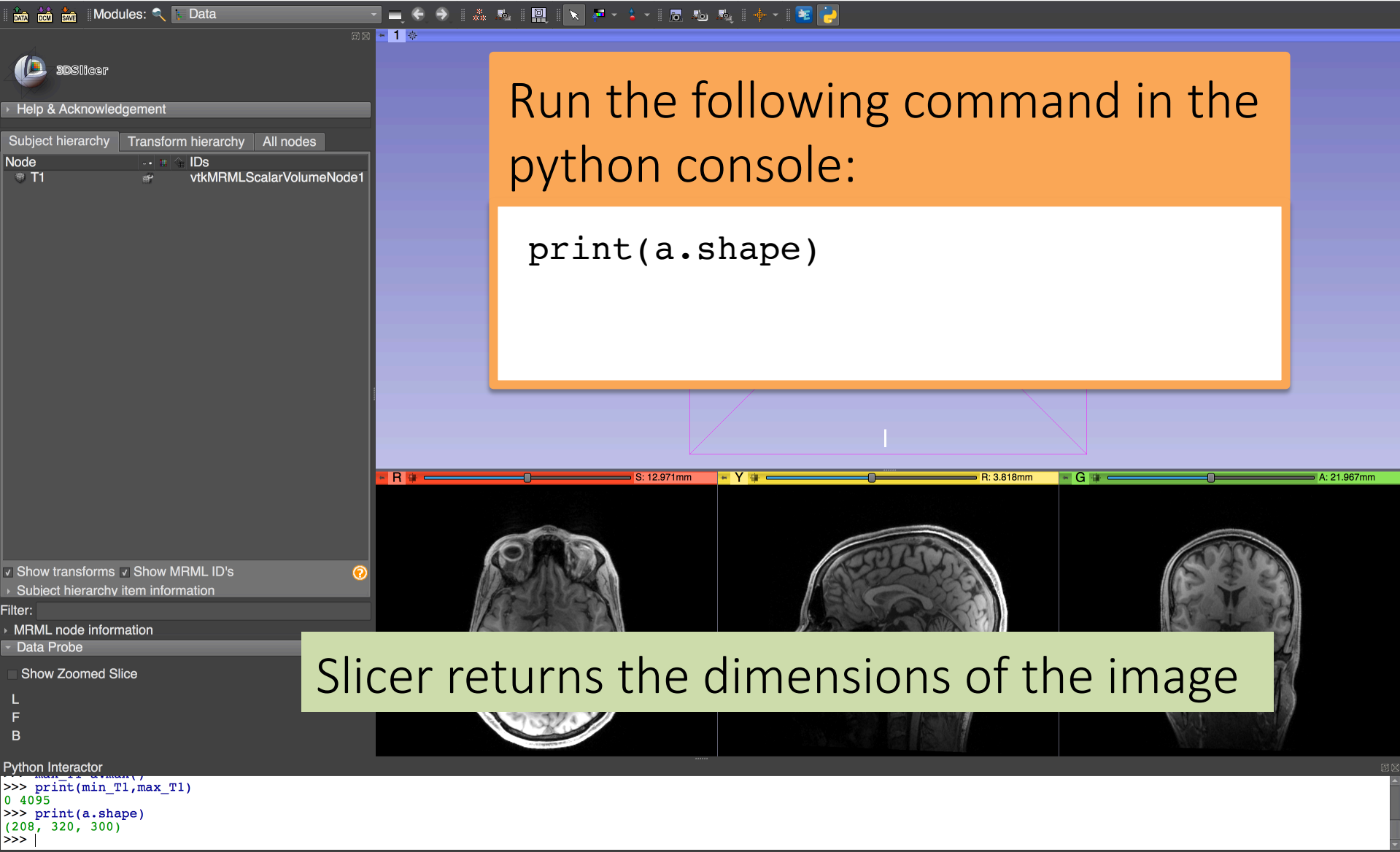
An orange callout box in the center of the main window contains the text: 'Run the following command in the python console:'. Below this box, a white box contains the same Python code as shown in the console. The console output shows the execution of the code, resulting in the values [0 48 49 ... 10 10 46].

Accessing voxels in a volume

The screenshot displays the 3D Slicer software interface. On the left, the 'Subject hierarchy' panel shows a tree structure with 'T1' selected, containing a 'vtkMRMLScalarVolumeNode1'. Below this, the 'Data Probe' section is active, showing 'Show Zoomed Slice' and 'L', 'F', 'B' orientation indicators. The main 3D view shows a blue volume with a purple bounding box labeled with 'S' (Superior), 'I' (Inferior), 'R' (Right), and 'L' (Left). Below the 3D view are three orthogonal views: axial (R), sagittal (Y), and coronal (G). The bottom status bar shows coordinates: S: 12.971mm, Y: 3.818mm, G: 21.967mm. A green text box at the bottom reads: 'Slicer returns the min and max values of the T1 image'. At the bottom left, the Python Interactor shows the following code and output:

```
>>> min_T1=a.min()
>>> max_T1=a.max()
>>> print(min_T1,max_T1)
0 4095
>>>
```

Modifying voxels in a volume



The image shows a screenshot of the 3D Slicer software interface. The main window displays three orthogonal MRI slices of a brain. The top-left panel shows the 'Subject hierarchy' with a tree view containing 'T1' and 'vtkMRMLScalarVolumeNode1'. The bottom-left panel shows the 'Python Interactor' with the following code and output:

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

Overlaid on the interface are two text boxes:

- An orange box in the upper right containing the text: "Run the following command in the python console:"
- A white box with an orange border below it containing the code: `print(a.shape)`

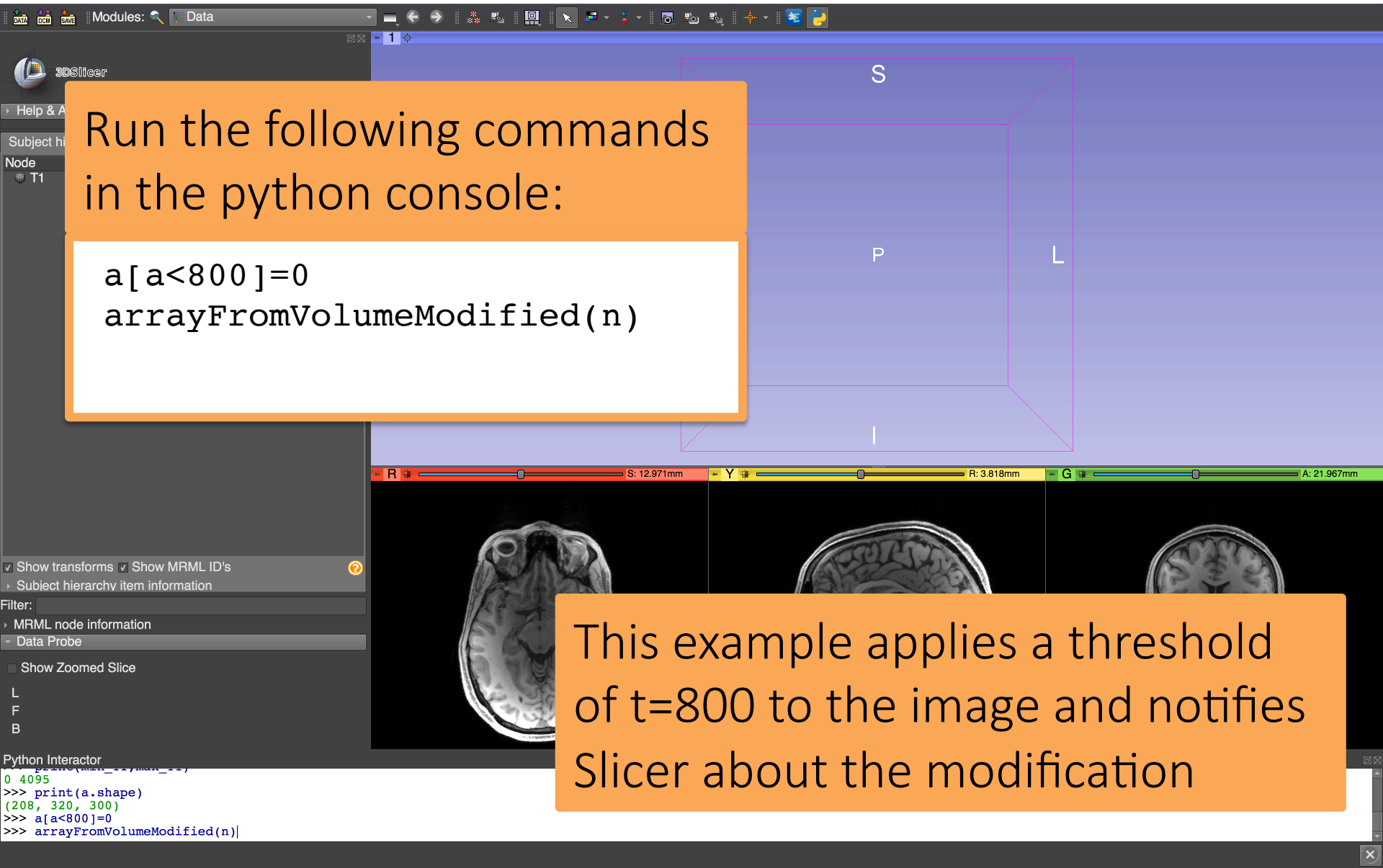
A green box at the bottom center contains the text: "Slicer returns the dimensions of the image".

Modifying voxels in a volume

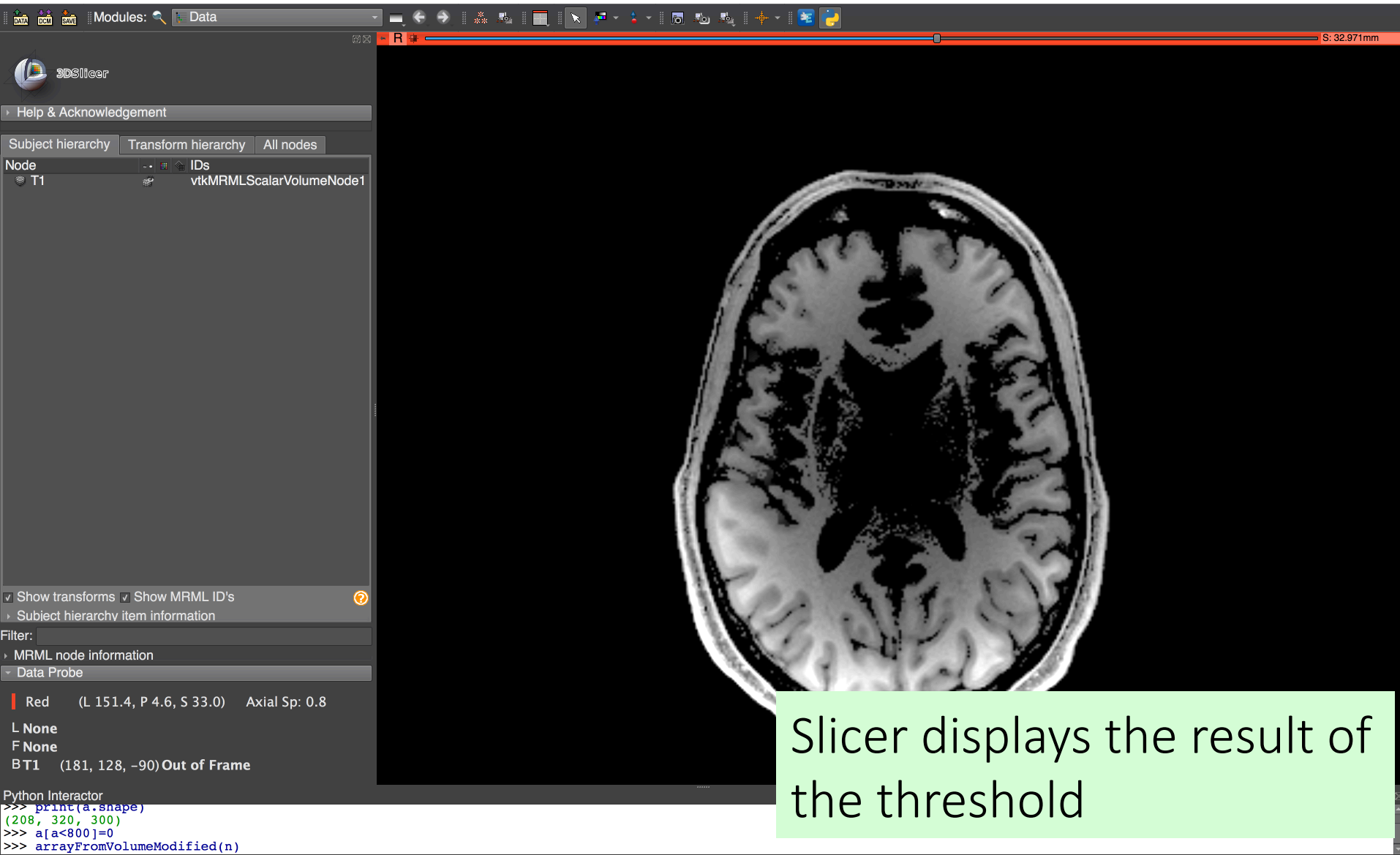
Run the following commands in the python console:

```
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



The screenshot displays the 3D Slicer software interface. The main window shows an axial MRI slice of a brain. The left sidebar contains a 'Subject hierarchy' panel with a tree view showing a 'T1' node and its associated 'vtkMRMLScalarVolumeNode1'. Below this, there are checkboxes for 'Show transforms' and 'Show MRML ID's', and a 'Filter:' section with 'MRML node information' and 'Data Probe' expanded. The 'Data Probe' section shows the following information:

- Red (L 151.4, P 4.6, S 33.0) Axial Sp: 0.8
- L None
- F None
- BT1 (181, 128, -90) Out of Frame

At the bottom left, the Python Interactor shows the following code and output:

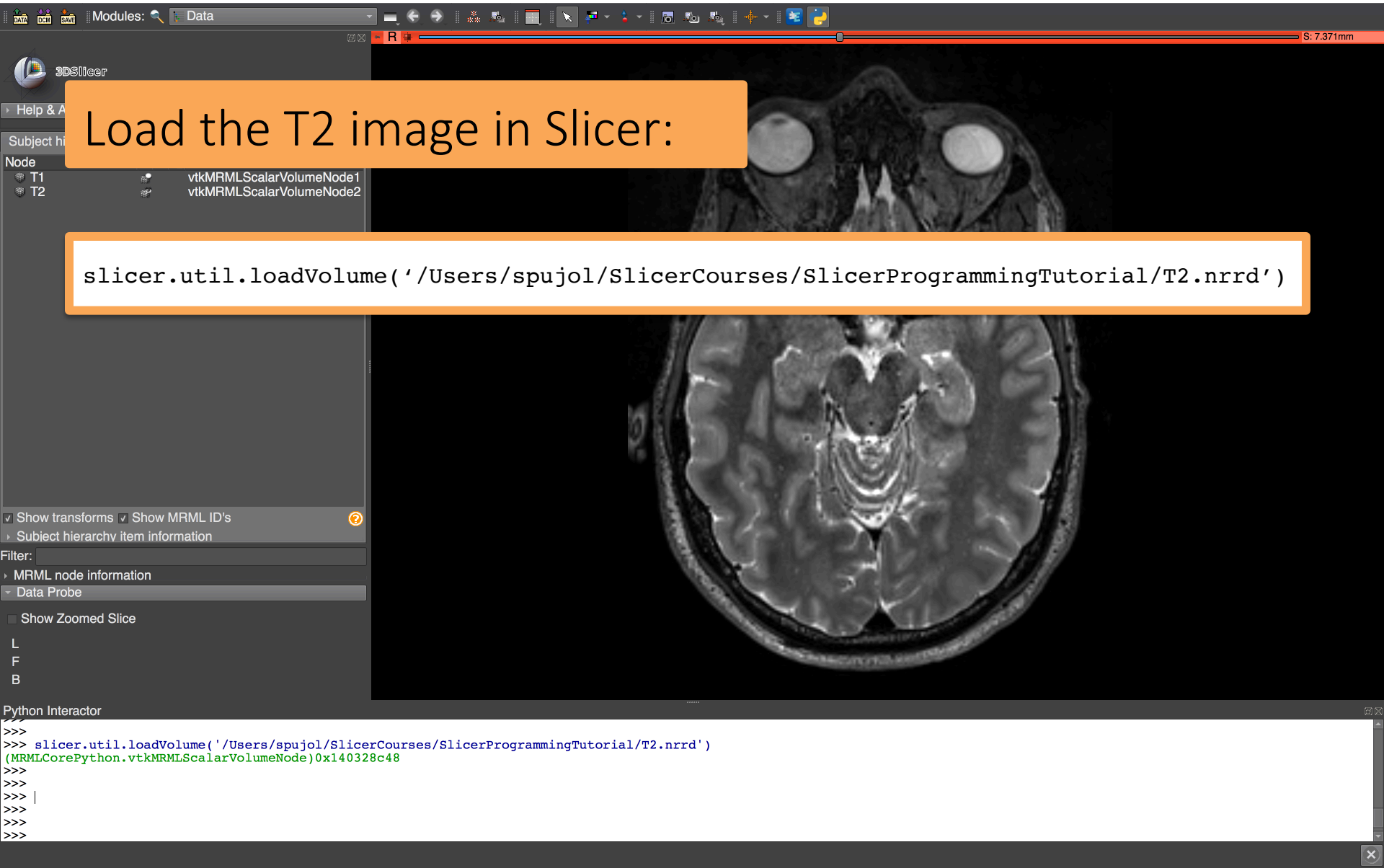
```
>>> print(a.shape)
(208, 320, 300)
>>> a[a<800]=0
>>> arrayFromVolumeModified(n)
```

A green text box in the bottom right corner of the image contains the text: "Slicer displays the result of the threshold".

Loading the T2 volume

Load the T2 image in Slicer:

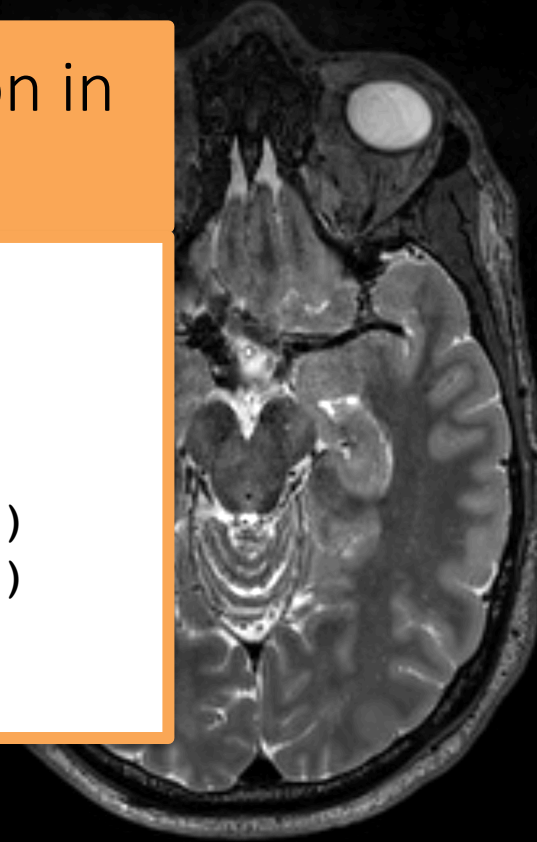
```
slicer.util.loadVolume('/Users/spujol/SlicerCourses/SlicerProgrammingTutorial/T2.nrrd')
```



Python function: threshold

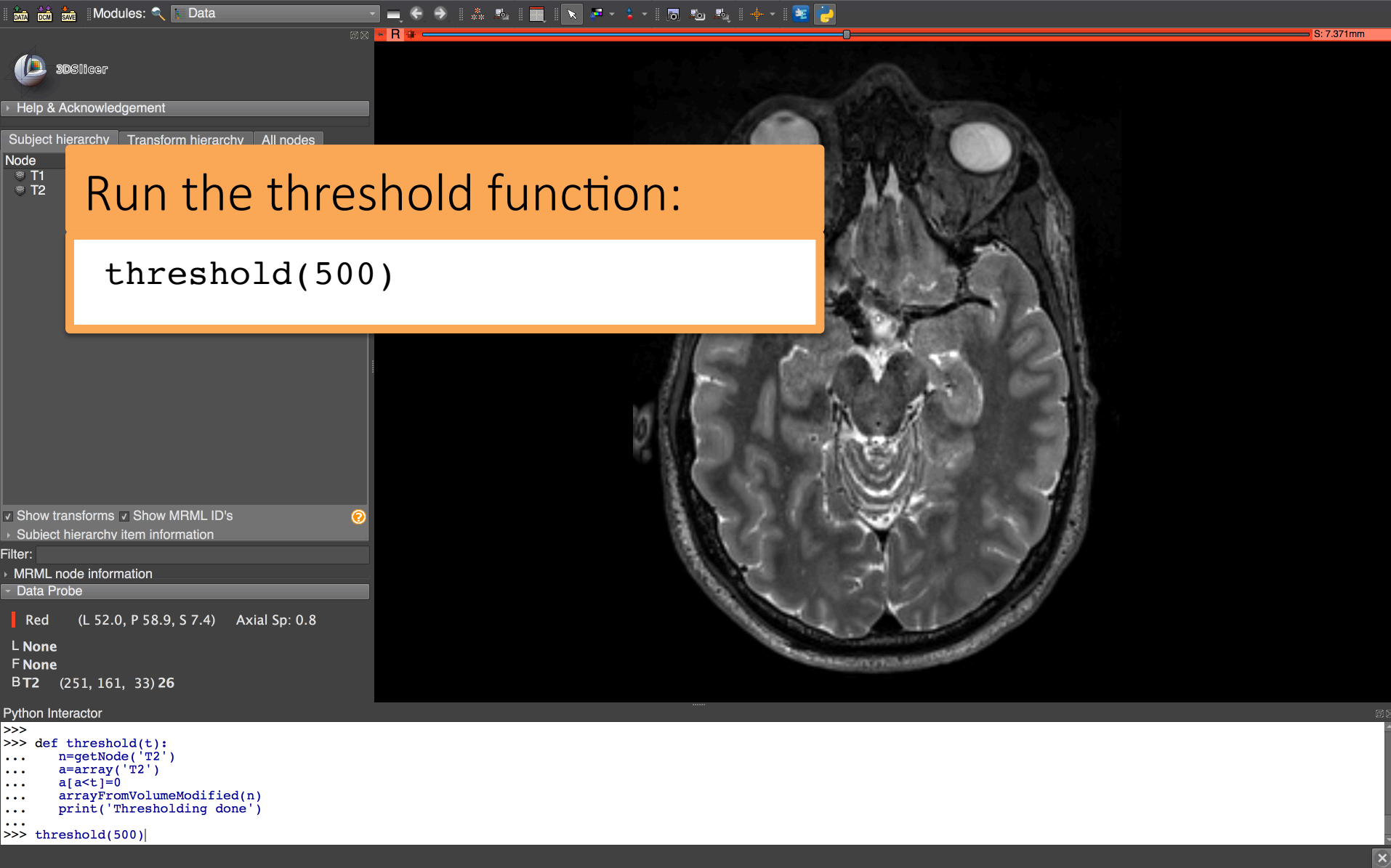
Create a `threshold(t)` function in the Python interactor:

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



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

Python function: threshold



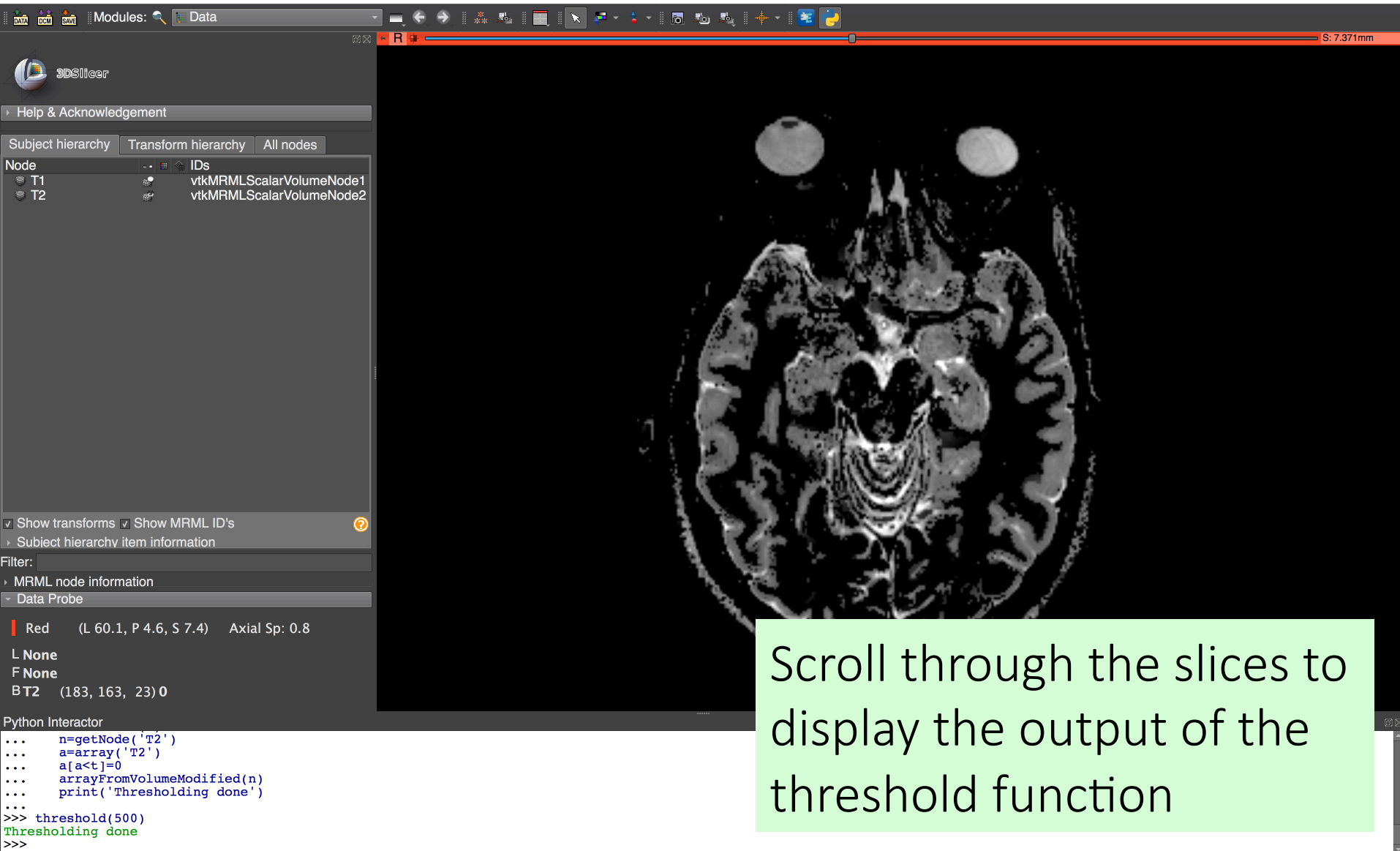
The image shows a screenshot of the 3DSlicer software interface. The main window displays an axial MRI scan of a brain. Overlaid on the left side of the scan is an orange text box containing the instruction "Run the threshold function:" and a white text box containing the code "threshold(500)".

The software interface includes a top menu bar with "Modules" and "Data". The left sidebar shows the "Node" list with "T1" and "T2" selected. Below the node list, there are sections for "Show transforms", "Show MRML ID's", "Subject hierarchy item information", "Filter:", "MRML node information", and "Data Probe". The "Data Probe" section shows "Red (L 52.0, P 58.9, S 7.4) Axial Sp: 0.8".

The bottom of the interface features a "Python Interactor" window with the following code:

```
>>>
>>> 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 MRI slice of a brain, which has been thresholded to highlight specific anatomical features. The left sidebar contains a 'Subject hierarchy' panel with nodes 'T1' and 'T2'. Below it, there are options to 'Show transforms' and 'Show MRML ID's'. The bottom-left corner features a 'Python Interactor' window with the following code and output:

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

A green text box in the bottom right corner of the image contains the text: "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

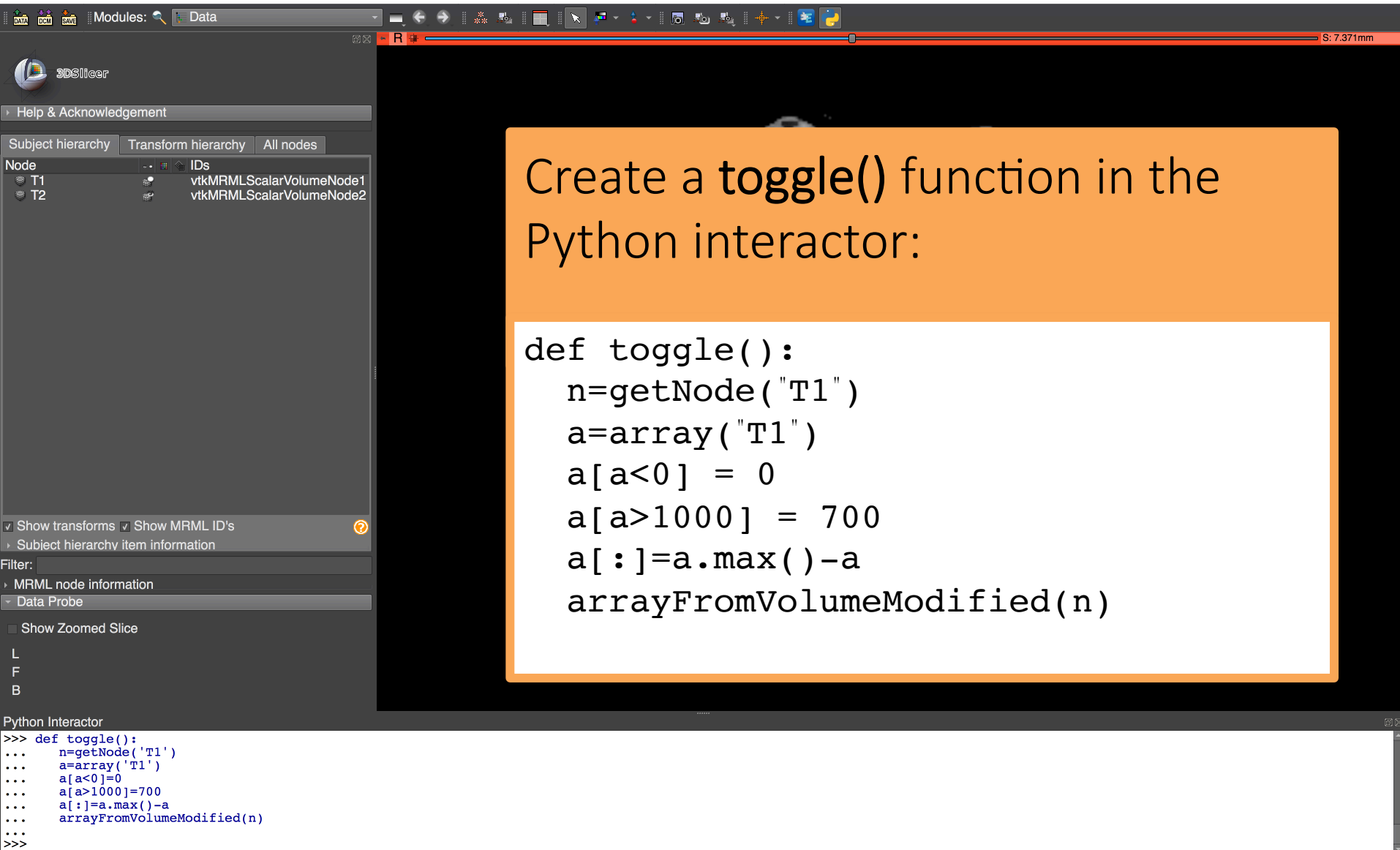
Part 3

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



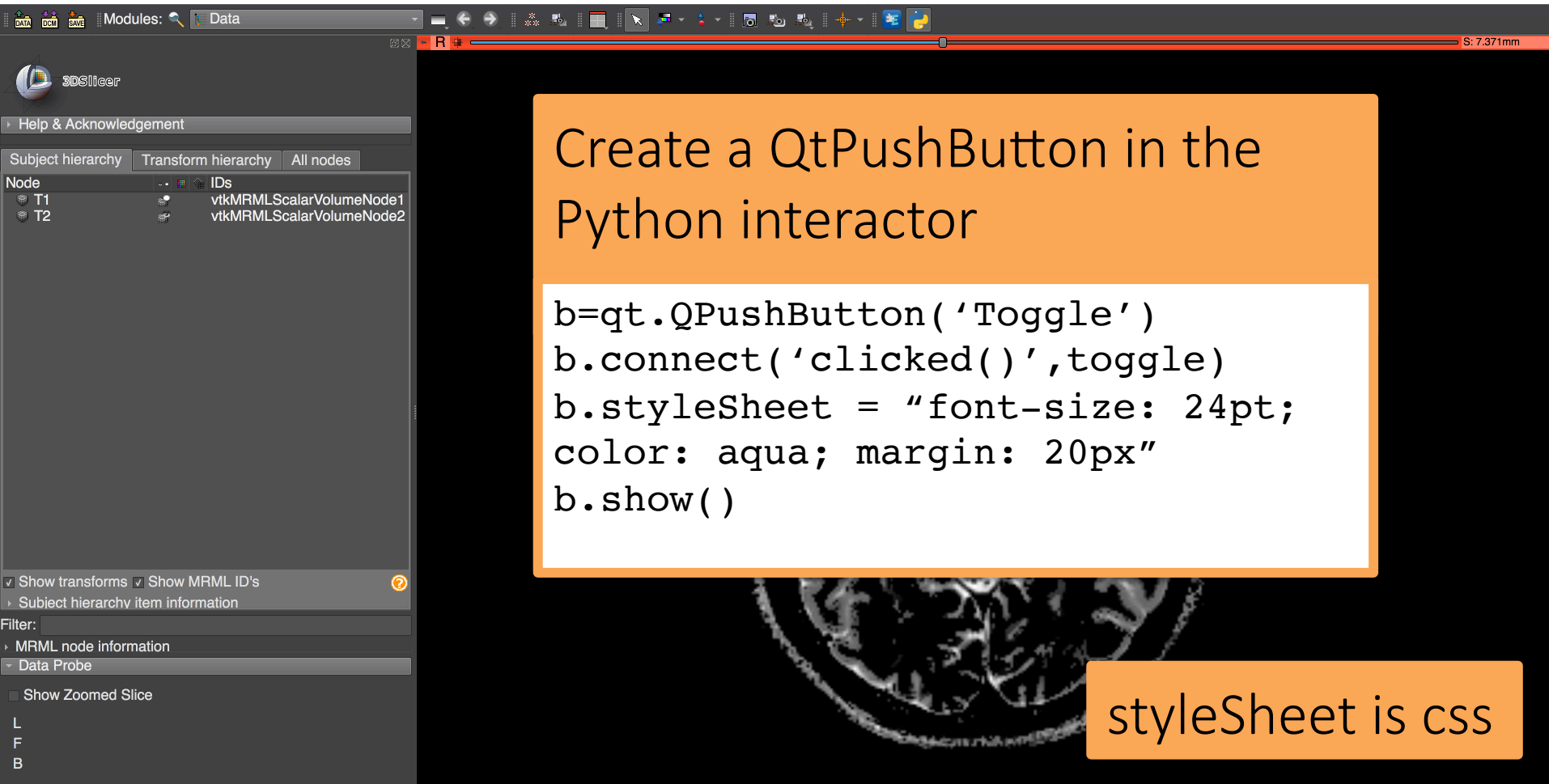
The image shows a screenshot of the 3D Slicer software interface. The top toolbar includes icons for Data, Copy, Save, and Modules. The main window displays a 3D view of a brain slice. On the left, the 'Subject hierarchy' panel shows two nodes: T1 and T2. Below it, the 'Python Interactor' panel shows 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)  
...  
>>>
```

In the center of the image, there is an orange box containing the text: "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)
```

Creating a Qt Push Button



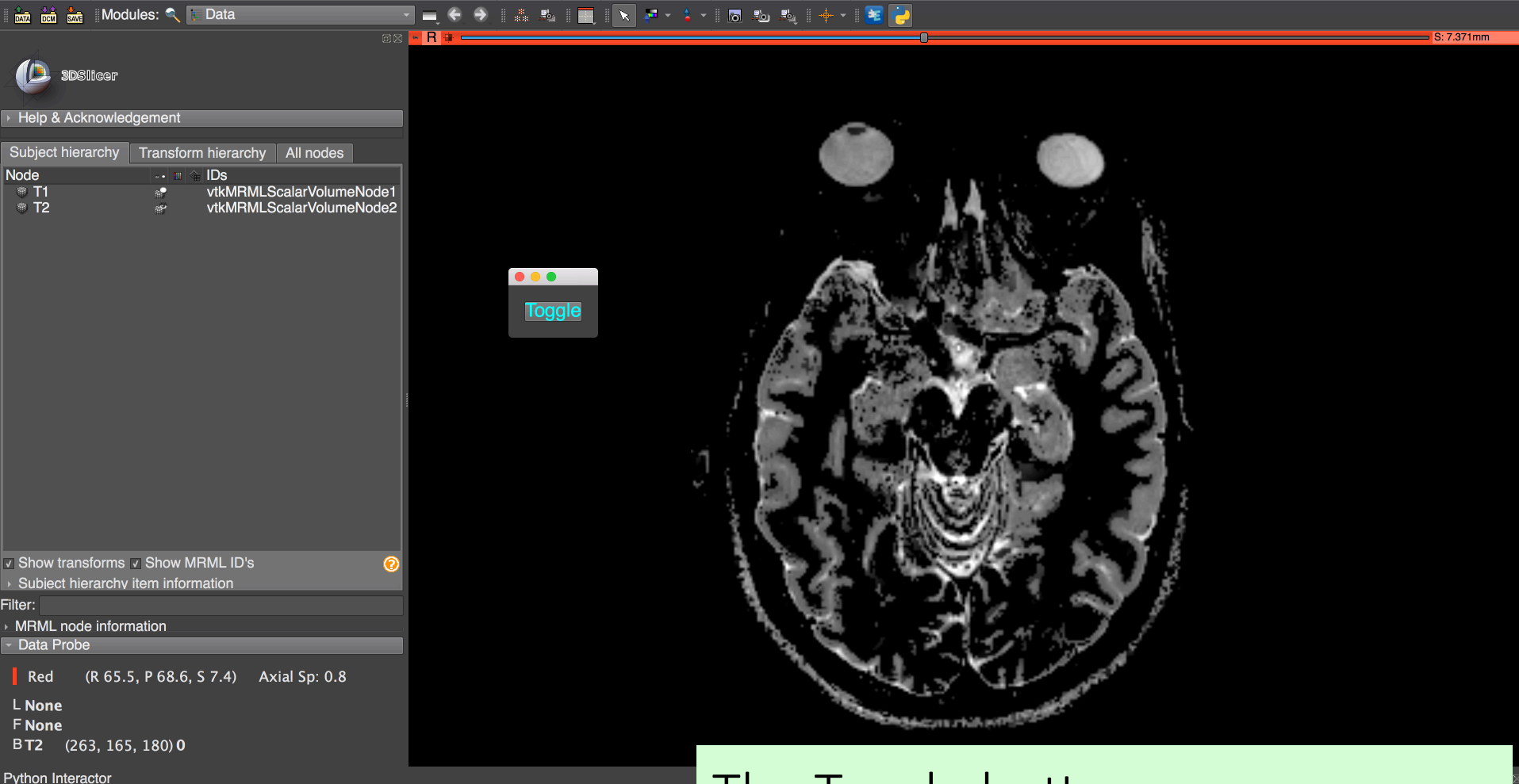
Create a QPushButton in the Python interactor

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

styleSheet is css

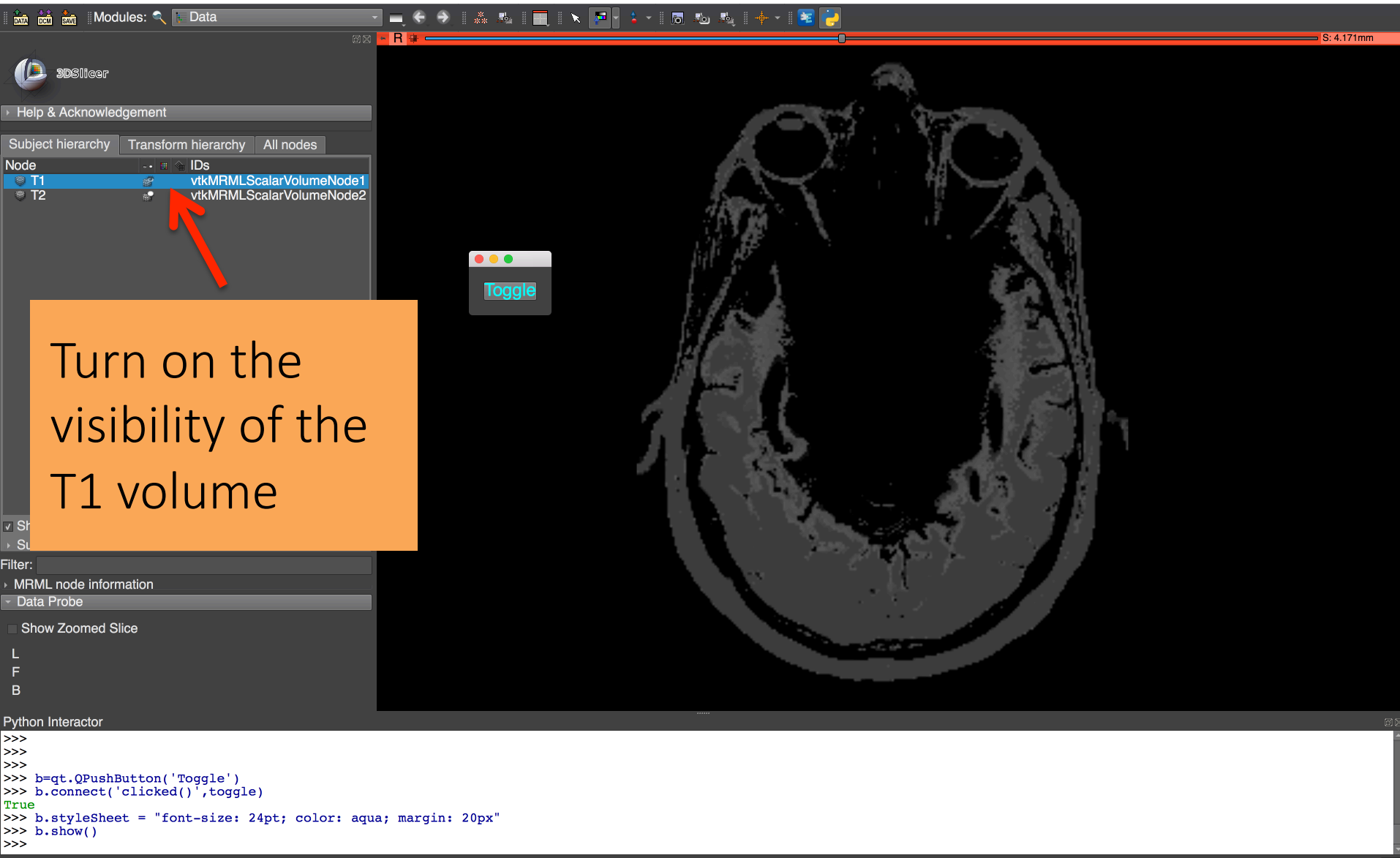
```
Python Interactor  
>>>  
>>>  
>>>  
>>> b=qt.QPushButton('Toggle')  
>>> b.connect('clicked()',toggle)  
True  
>>> b.setStyleSheet = "font-size: 24pt; color: aqua; margin: 20px"  
>>> b.show()
```

Creating a Qt Push Button



The Toggle button appears

Creating a Qt Push Button

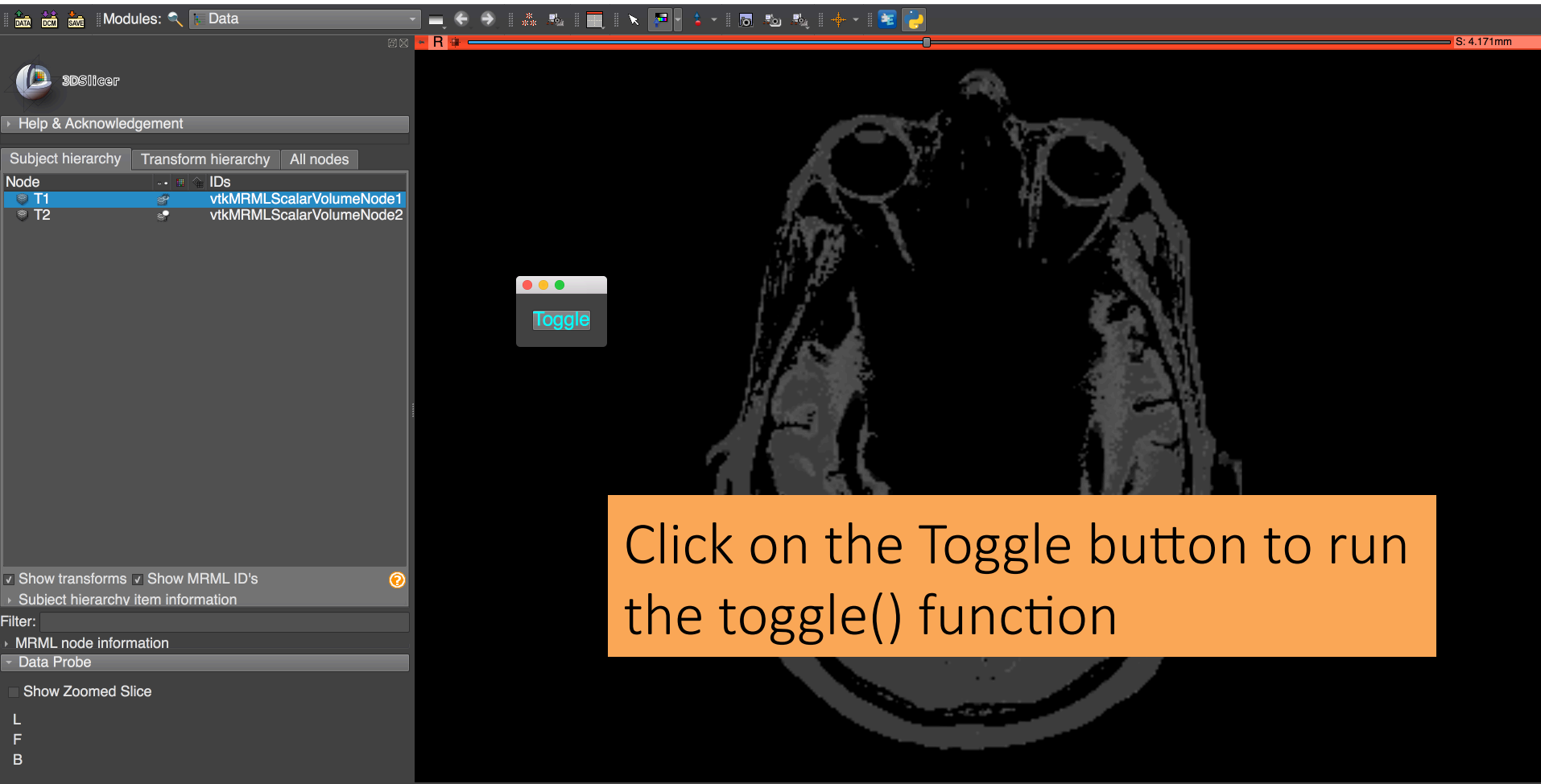


The screenshot displays the 3D Slicer software interface. The main window shows a grayscale axial MRI slice of a human head. On the left, the 'Subject hierarchy' panel is visible, with 'T1' and 'T2' nodes listed. A red arrow points to the 'T1' node, which is highlighted in blue. Below the hierarchy, a 'Toggle' button is shown in a small window. At the bottom, a Python Interactor window contains the following code:

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

Turn on the
visibility of the
T1 volume

Creating a Qt Push Button

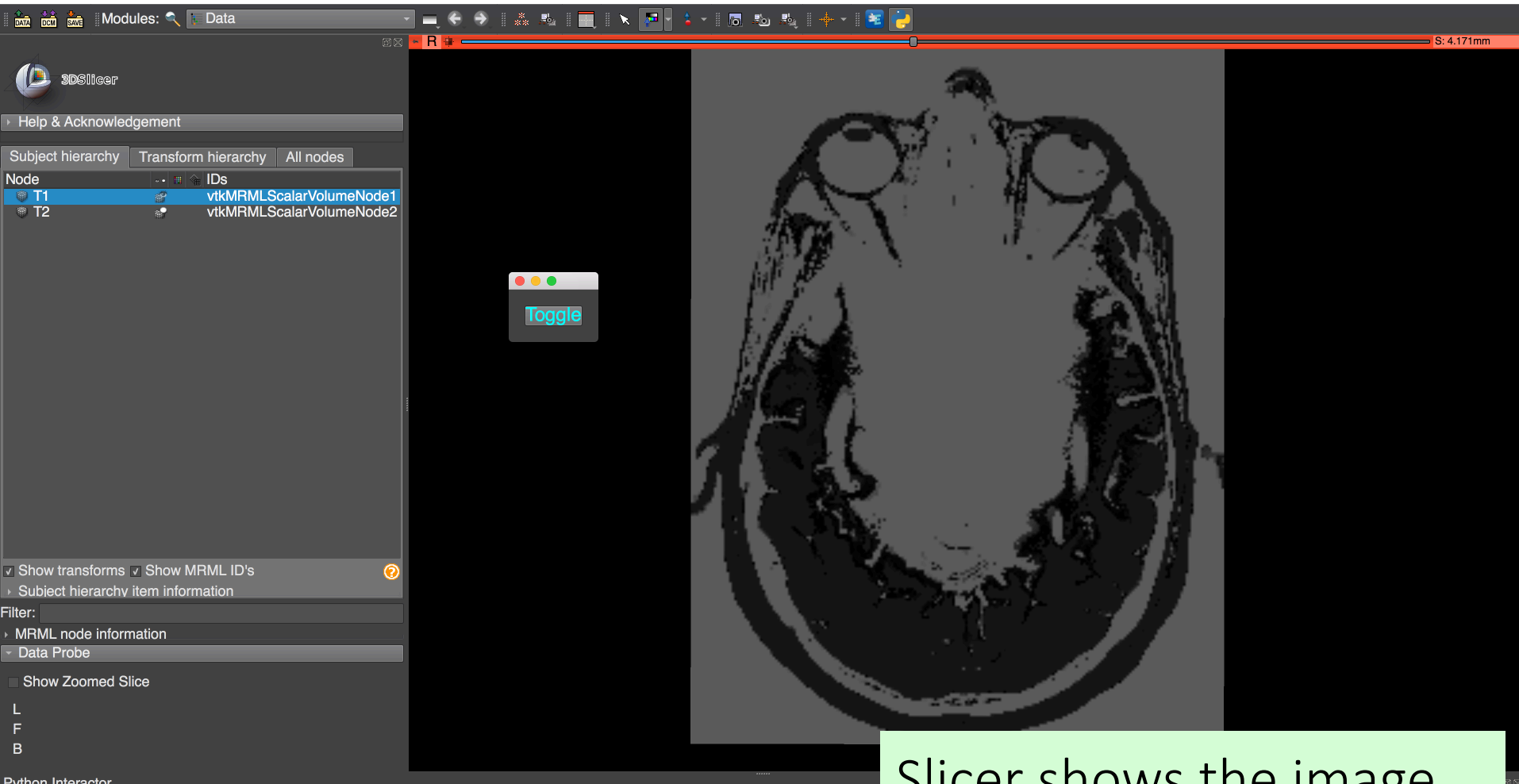


The screenshot shows the 3D Slicer application interface. On the left, there is a sidebar with a 'Subject hierarchy' panel containing nodes 'T1' and 'T2'. Below it is a 'Python Interactor' panel with a code editor. The main window displays a 3D view of a skull. A small Qt window with a 'Toggle' button is overlaid on the 3D view.

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

Click on the Toggle button to run the toggle() function

Creating a Qt Push Button



The screenshot displays the Slicer software interface. The central view shows an axial MRI slice of a brain. On the left, the 'Subject hierarchy' panel is visible, showing nodes T1 and T2. A custom Qt push button labeled 'Toggle' is overlaid on the interface. 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()
>>>
```

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 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)

 Sylvain Bouix, Ph.D.
Psychiatry Neuroimaging Laboratory