



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')
```

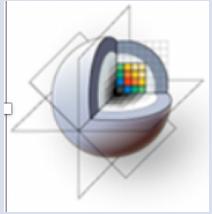


```
b=qt.QPushButton('Toggle')
b.connect('clicked()',toggle)
b.setStyleSheet = "font-size: 24pt; color:
aqua; margin: 20px"
b.show()
```



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

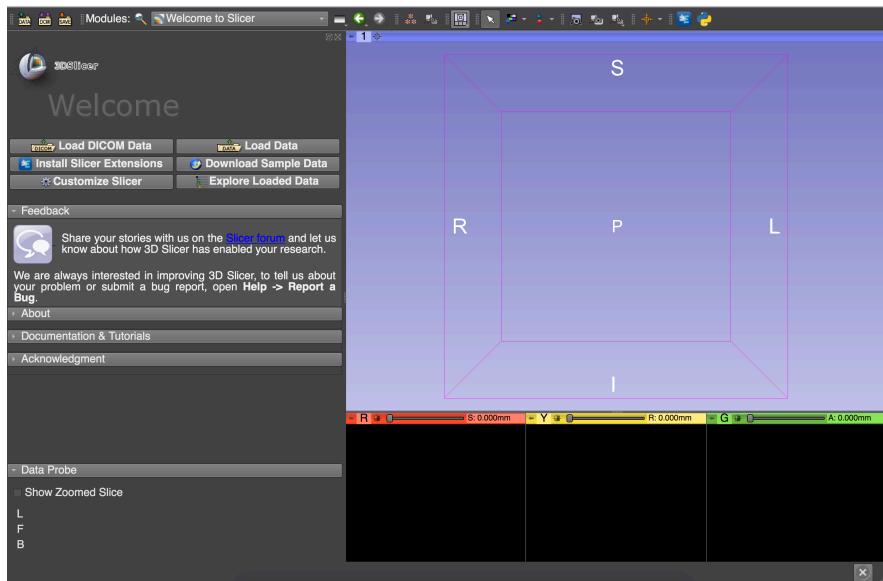


Part 3: Getting Familiar with the Qt widget toolkit in 3D Slicer

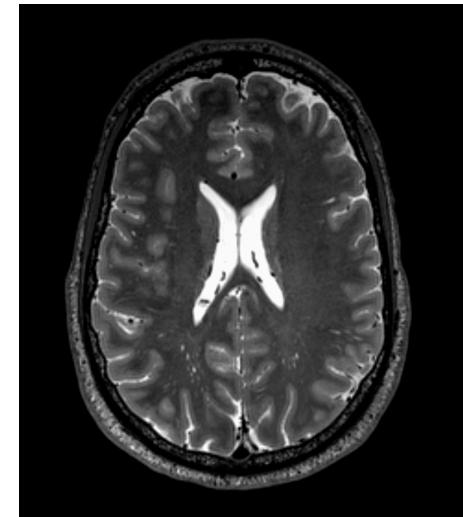
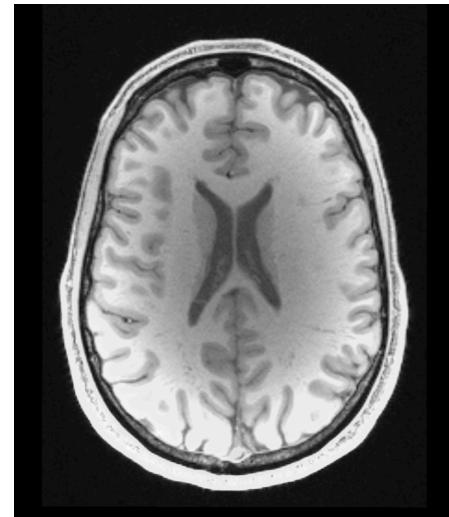
Disclaimer

- 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



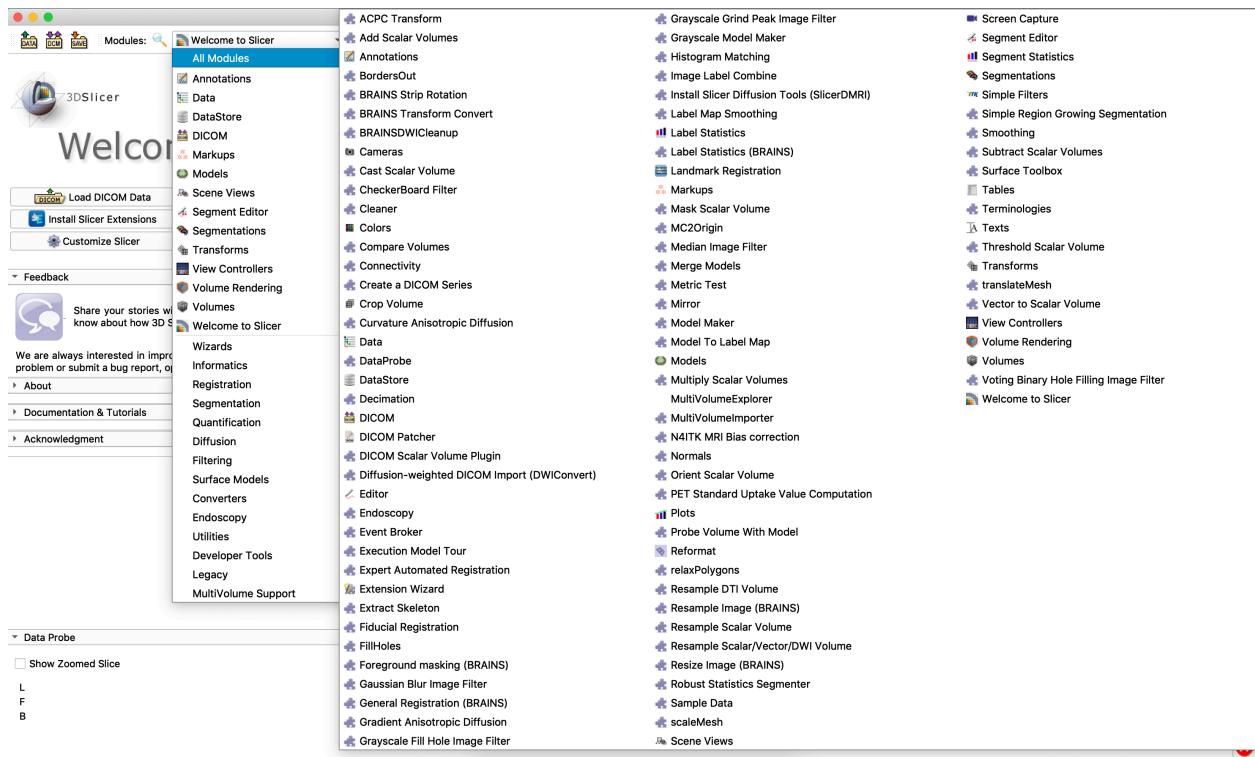
3D Slicer version 4.11



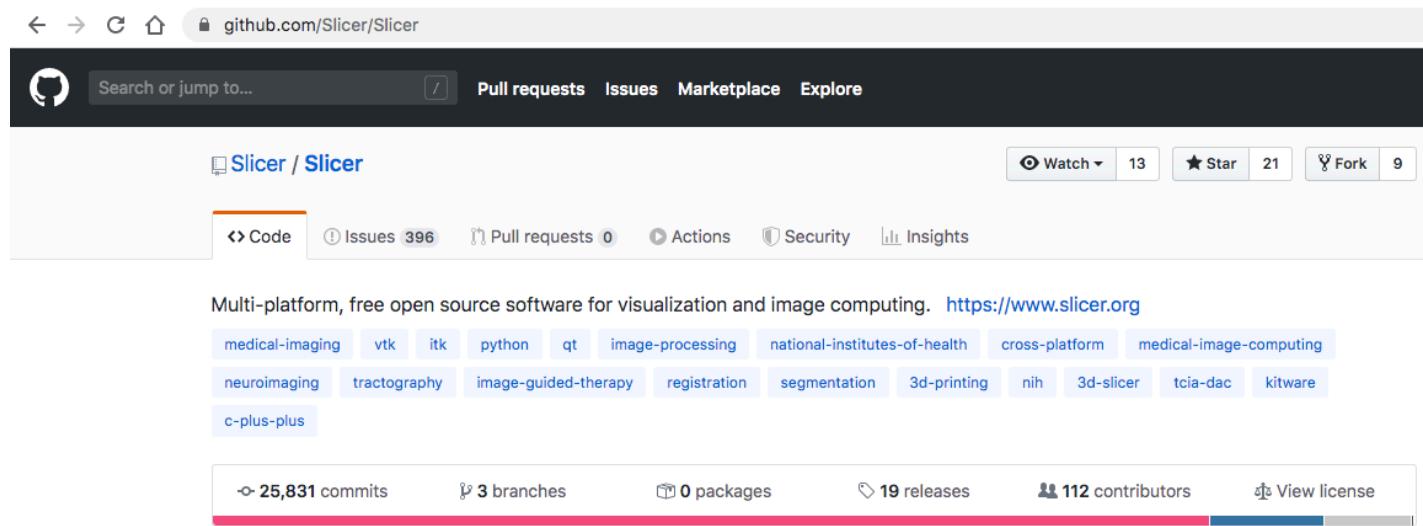
SlicerProgrammingTutorialData.zip

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>

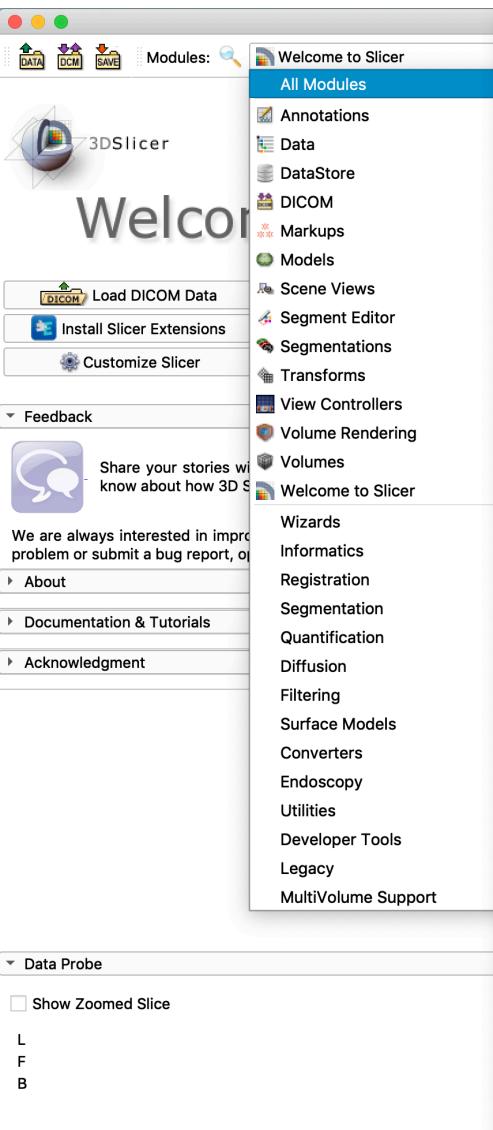
Slicer Modules

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

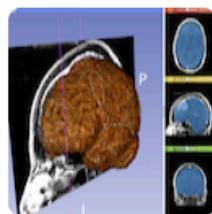
Slicer Modules



- The type of module is transparent to the end-user
- ACPC Transform
 - Add Scalar Volumes
 - Annotations
 - BordersOut
 - BRAINS Strip Rotation
 - BRAINS Transform Convert
 - BRAINS Strip Rotation
 - 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 Segmente
 - Sample Data
 - scaleMesh
 - Scene Views
 - Screen Capture
 - Segment Editor
 - Segment Statistics
 - Segmentations
 - Simple Filters

Slicer Extensions

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



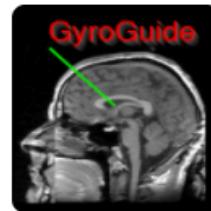
SwissSkullStripper
Bill Lorensen (Noware...)
★★★★★ (0)



PET-TumorSegmenta...
Christian Bauer (Univ...)
★★★★★ (0)



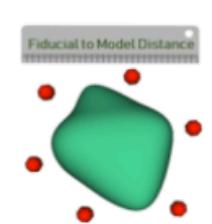
SlicerOpenIGTLINK
Junichi Tokuda (SPL, ...)
★★★★★ (0)



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



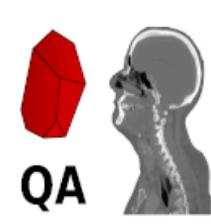
PET-IndiC
Ethan Ulrich (Universi...)
★★★★★ (0)



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



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



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

[INSTALL](#)

[INSTALL](#)

[INSTALL](#)

[INSTALL](#)

[INSTALL](#)

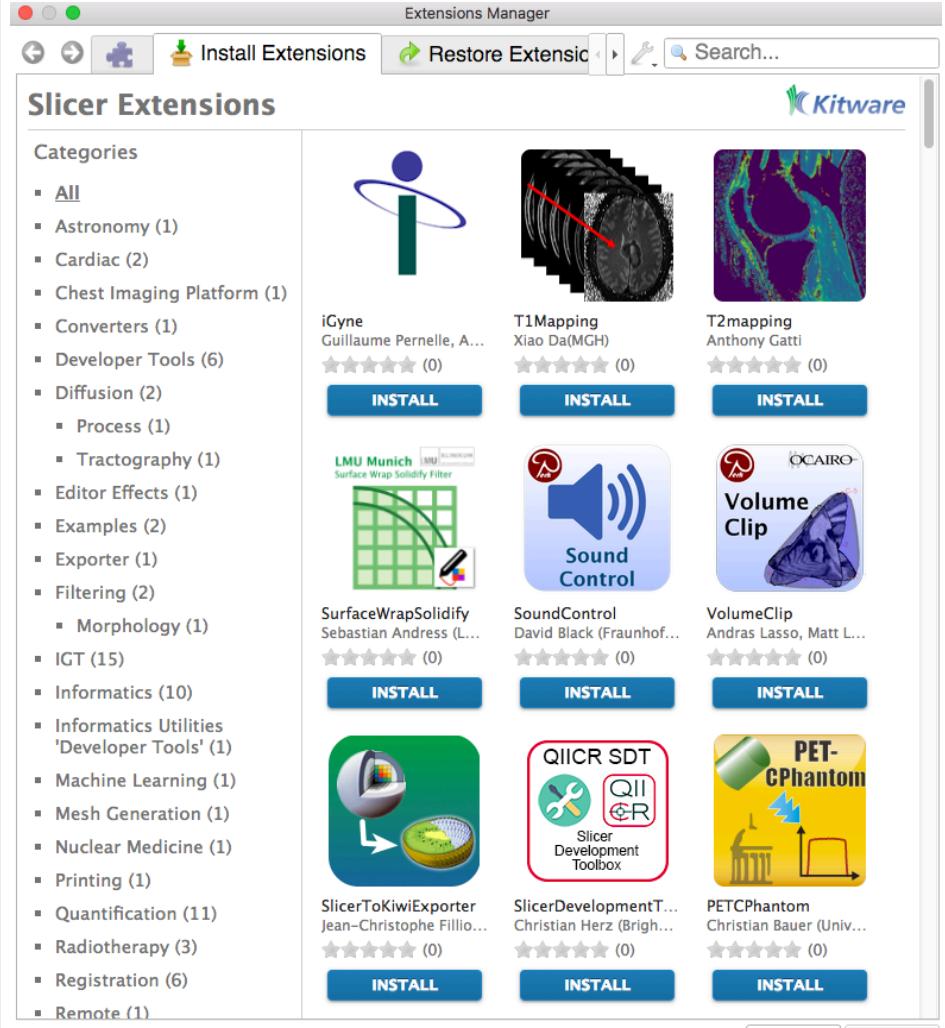
[INSTALL](#)

[INSTALL](#)

[INSTALL](#)

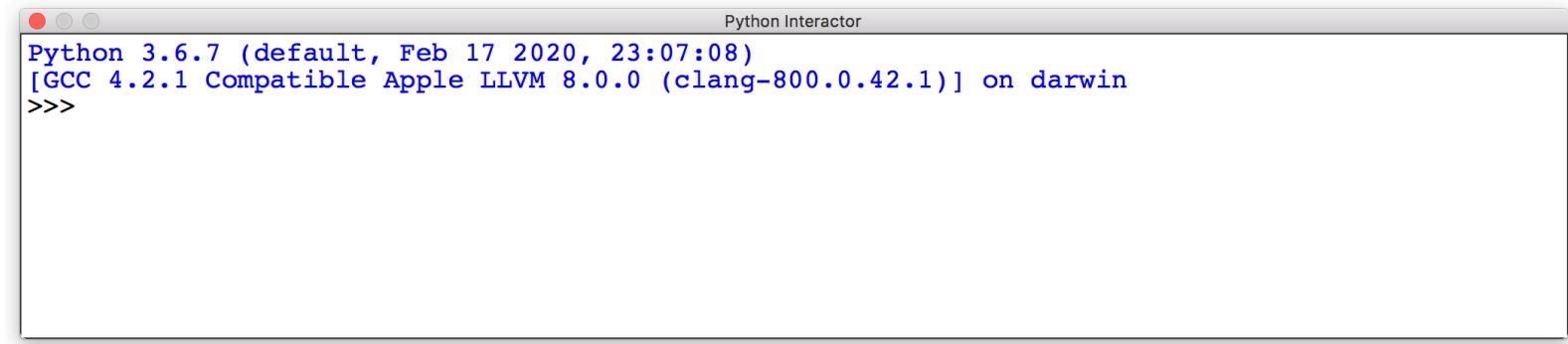
Slicer Extension Manager

- 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



A screenshot of a "Python Interactor" window. The title bar says "Python Interactor". The window contains the following text:

```
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 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.



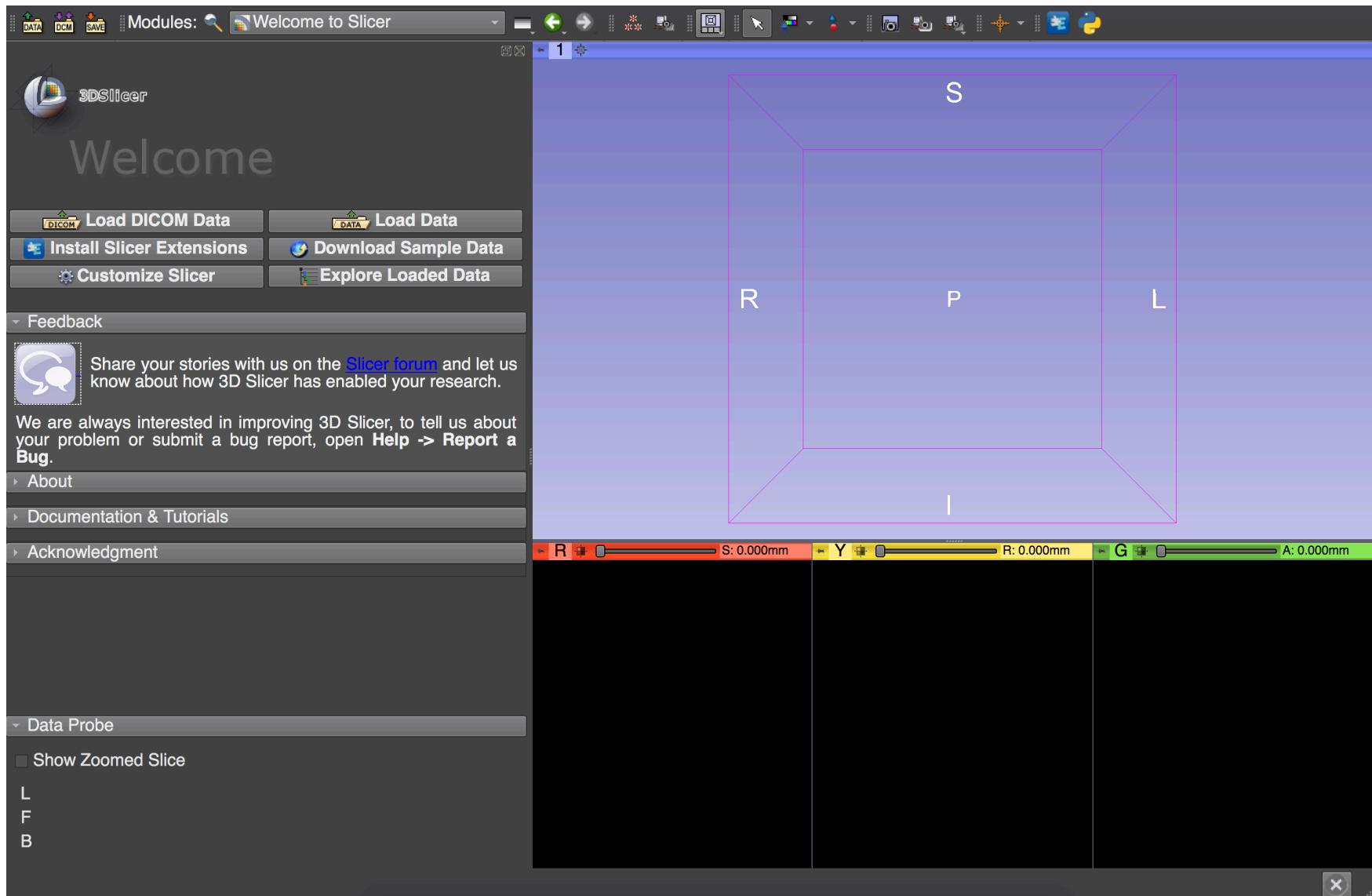
Qt is a cross-platform framework used as a graphical toolkit.

Python in Slicer



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

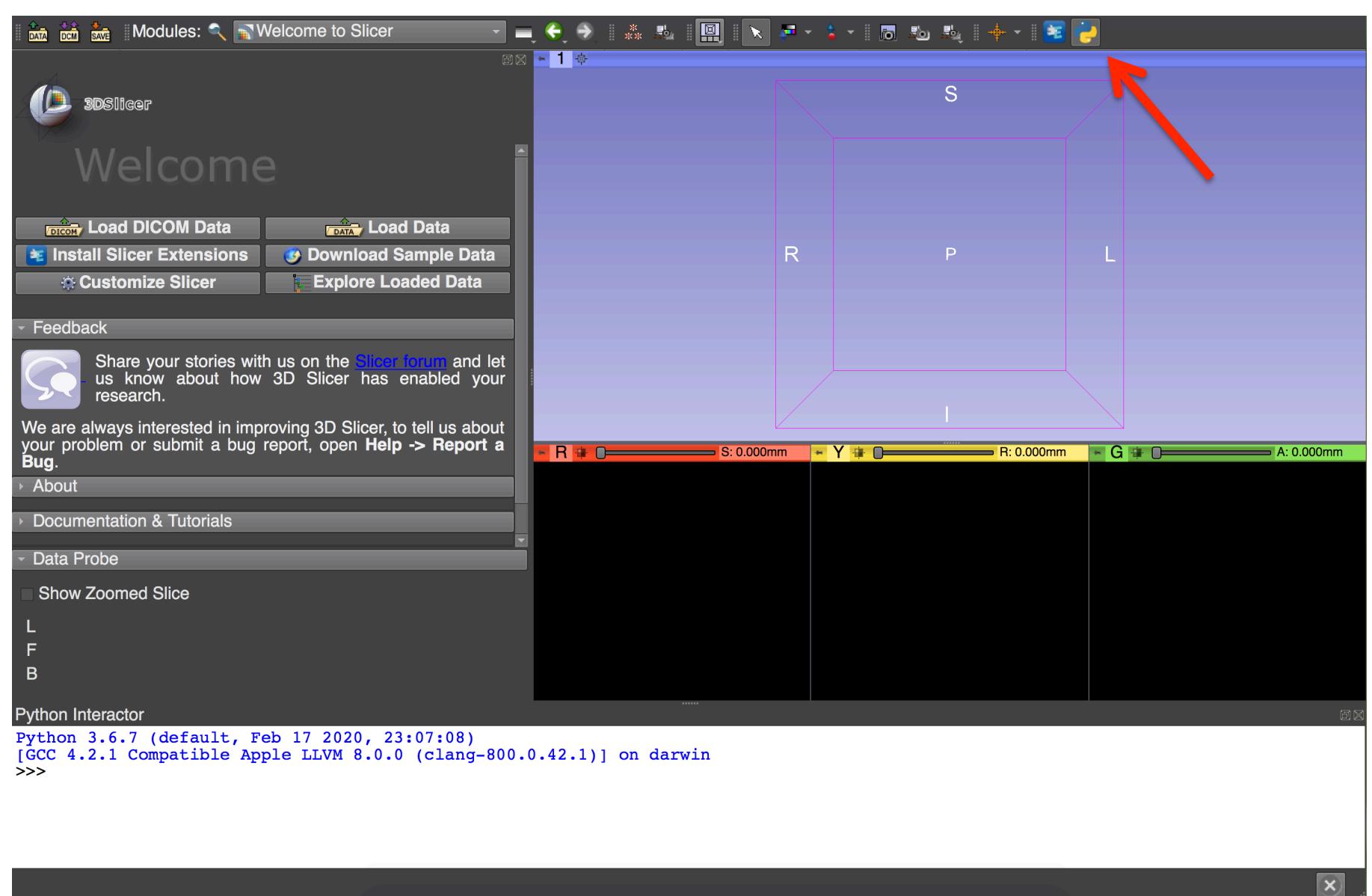
- The **pip install** command in Slicer enables developers to install most common scientific computing tools (e.g. TensorFlow, SciPy, PyTorch, Pandas, etc.)
- Slicer can be used as a **Jupyter notebook** kernel
- PyCharm and other Python development tools can be used with Slicer



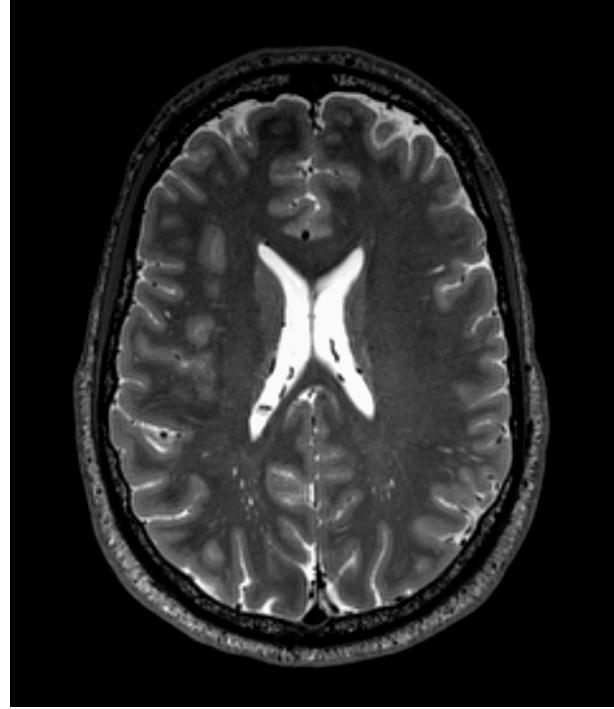
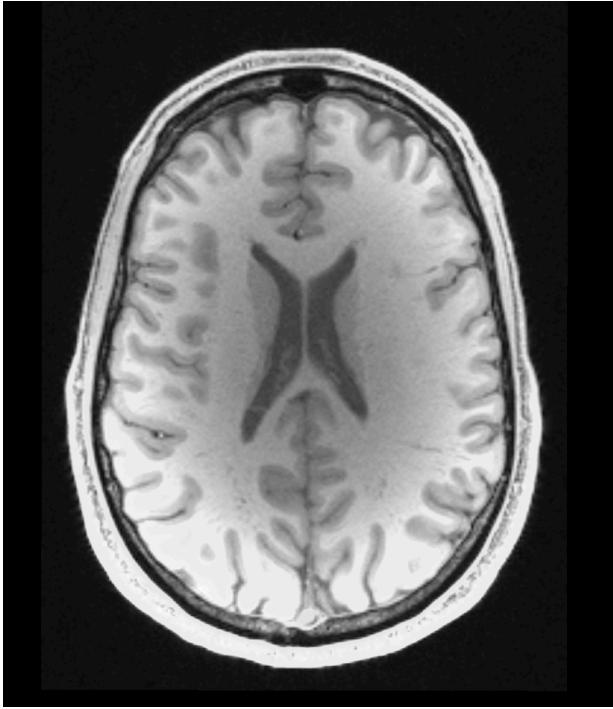
Slicer release version 5 integrates
Python3, VTK5 and ITK5

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.

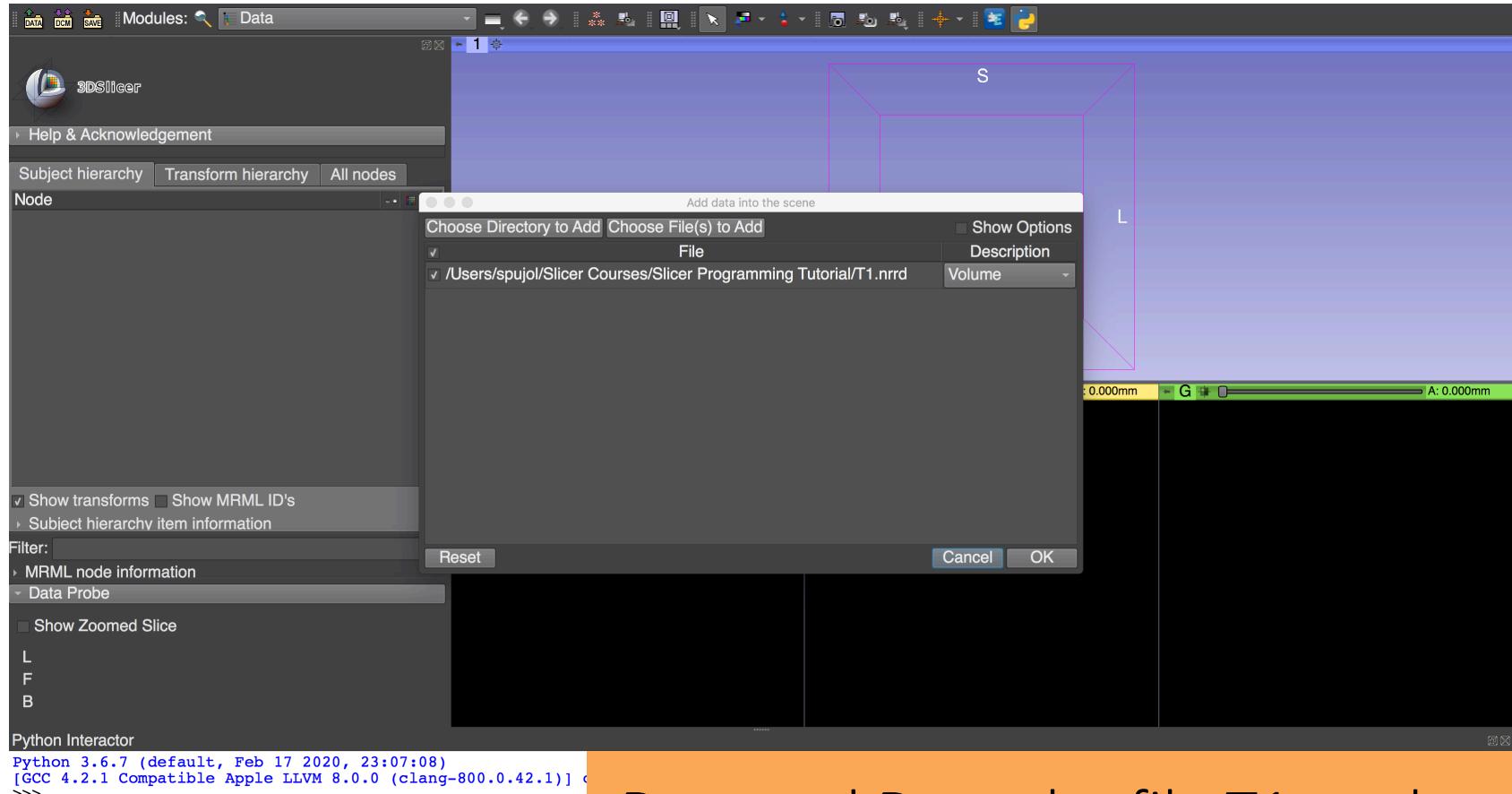


To access the Python Interactor, click on the Python icon  in the top bar menu of Slicer



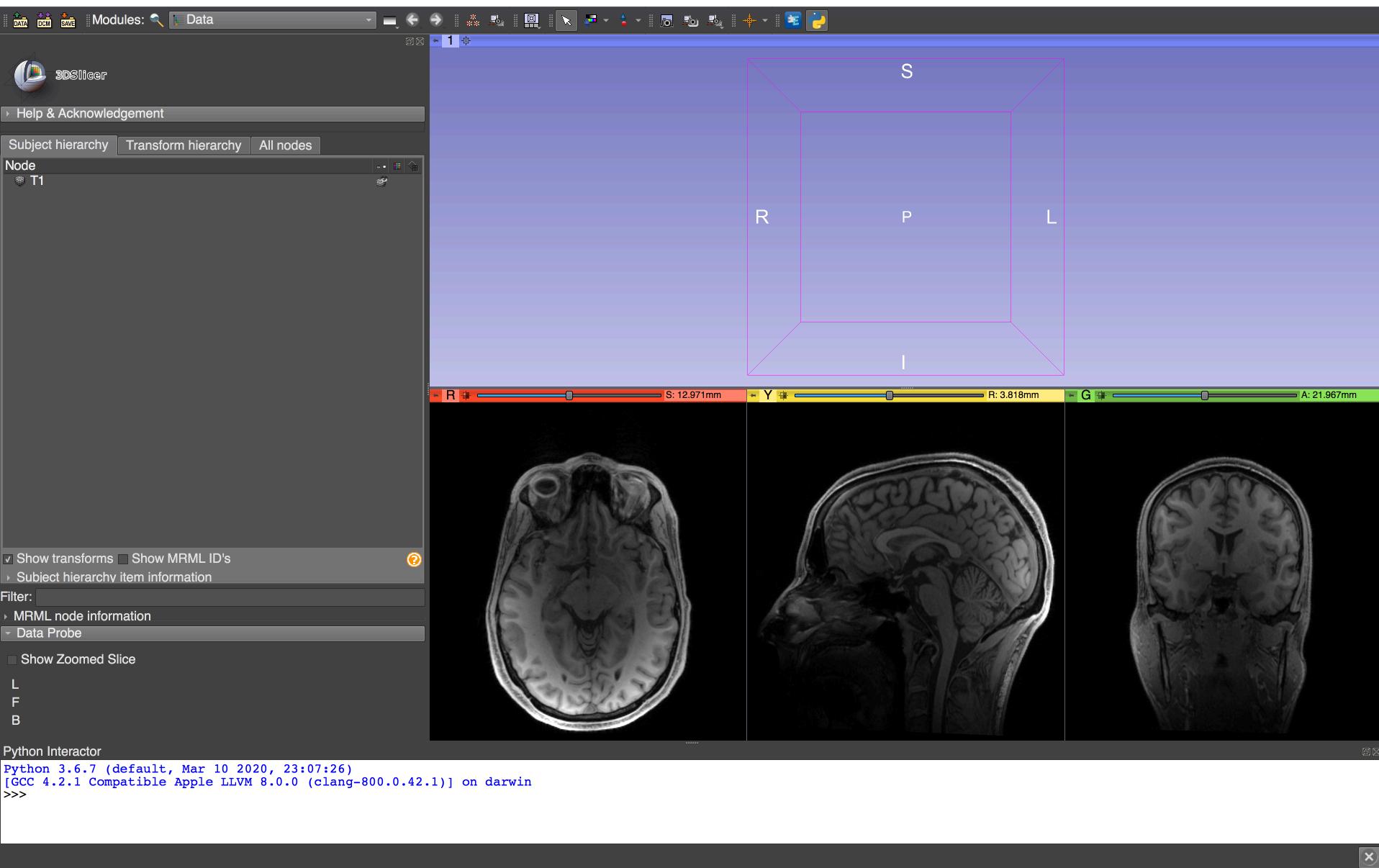
The Slicer Programming tutorial dataset includes a T1-weighted and a T2-weighted MRI scan of a healthy subject

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



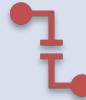
The **Slicer Data Model** is based on the Slicer Scene Data Structure



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)



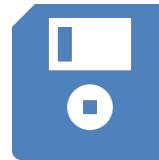
Each element a scene is called a **MRML node**

Slicer MRML Nodes: Basic Types



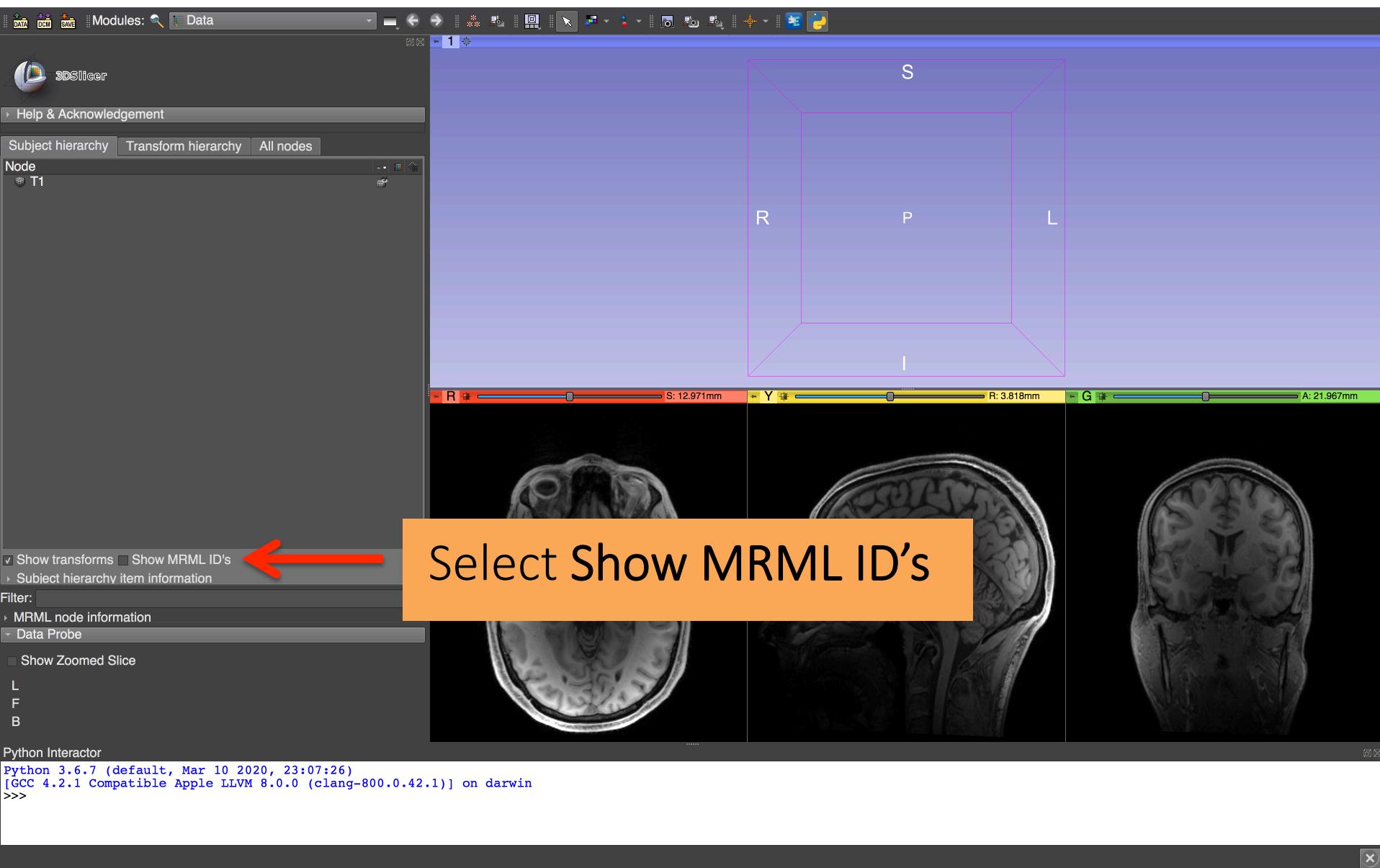
Data Node: Stores the raw data

Display Node: Describes how the data should be visualized

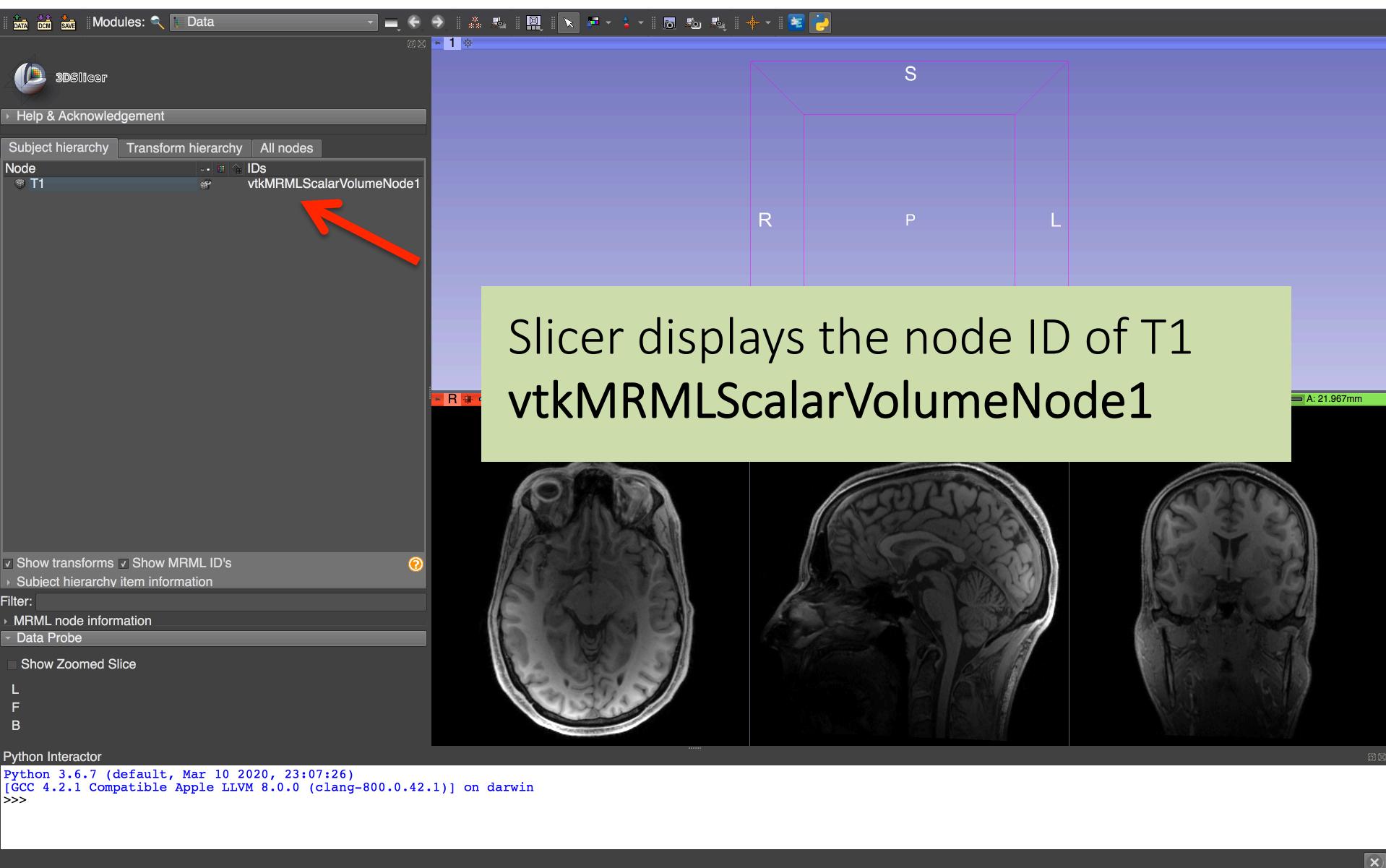


Storage Node: Describes how the data should be stored on disk

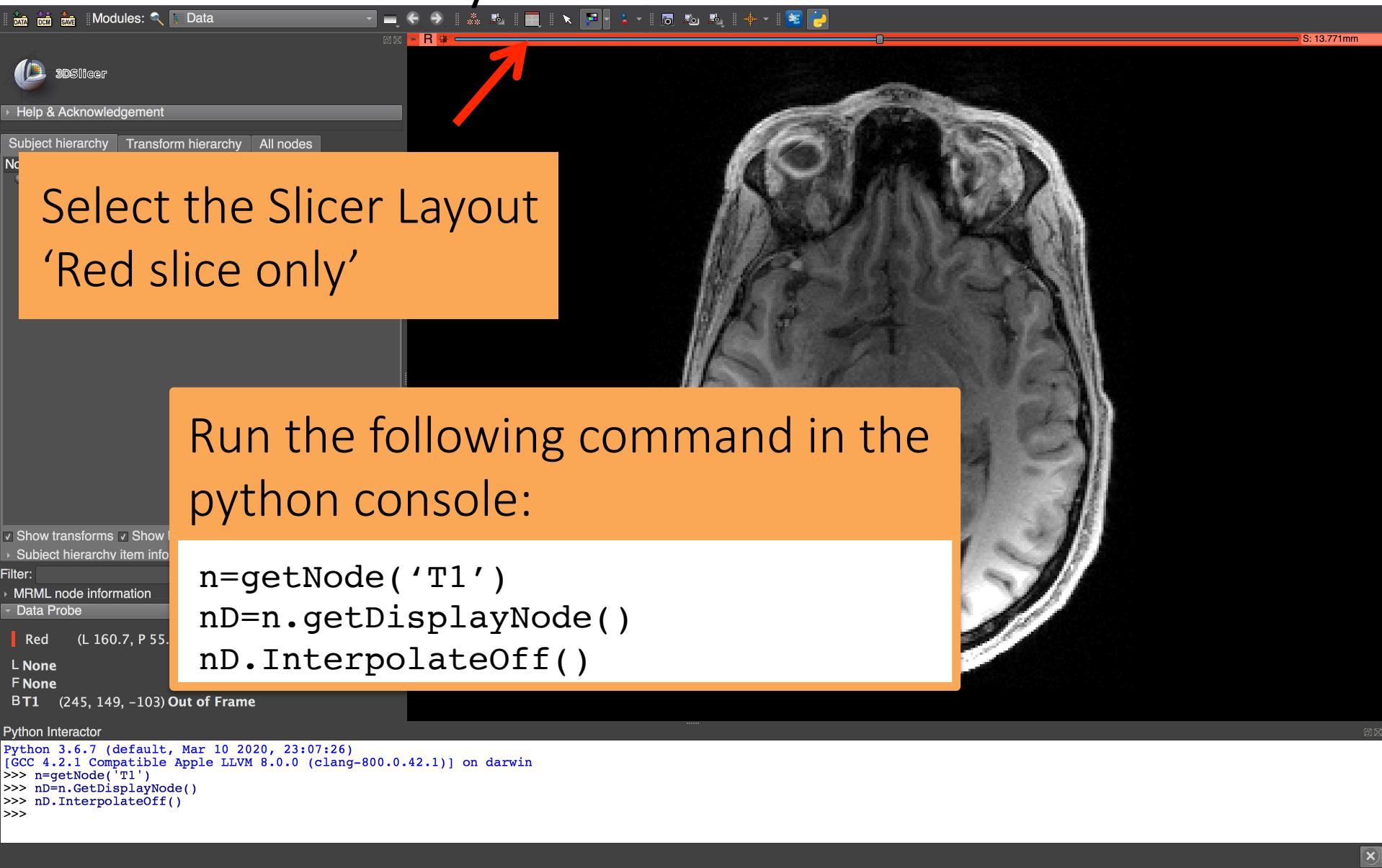
Tutorial dataset



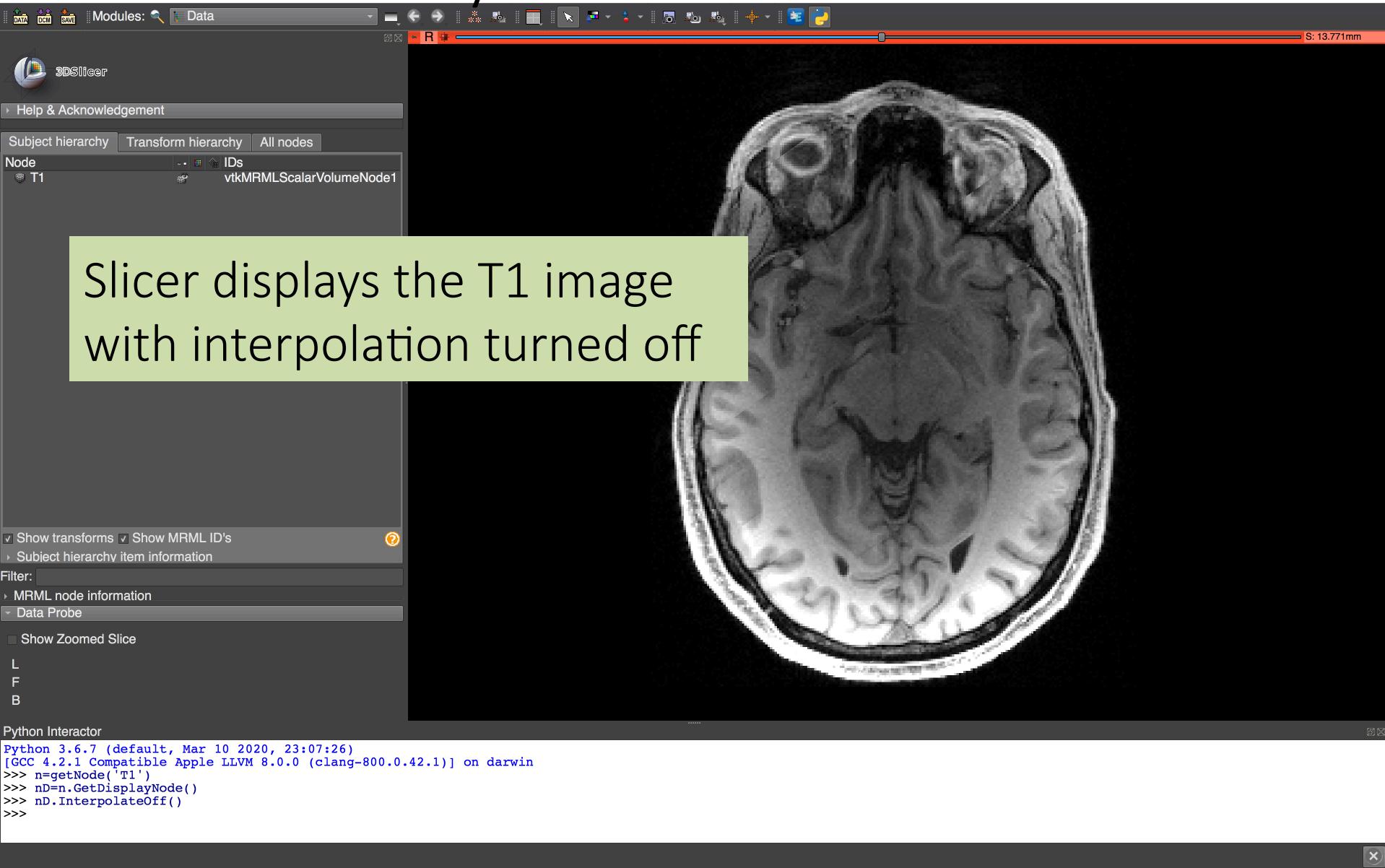
Slicer Data Model



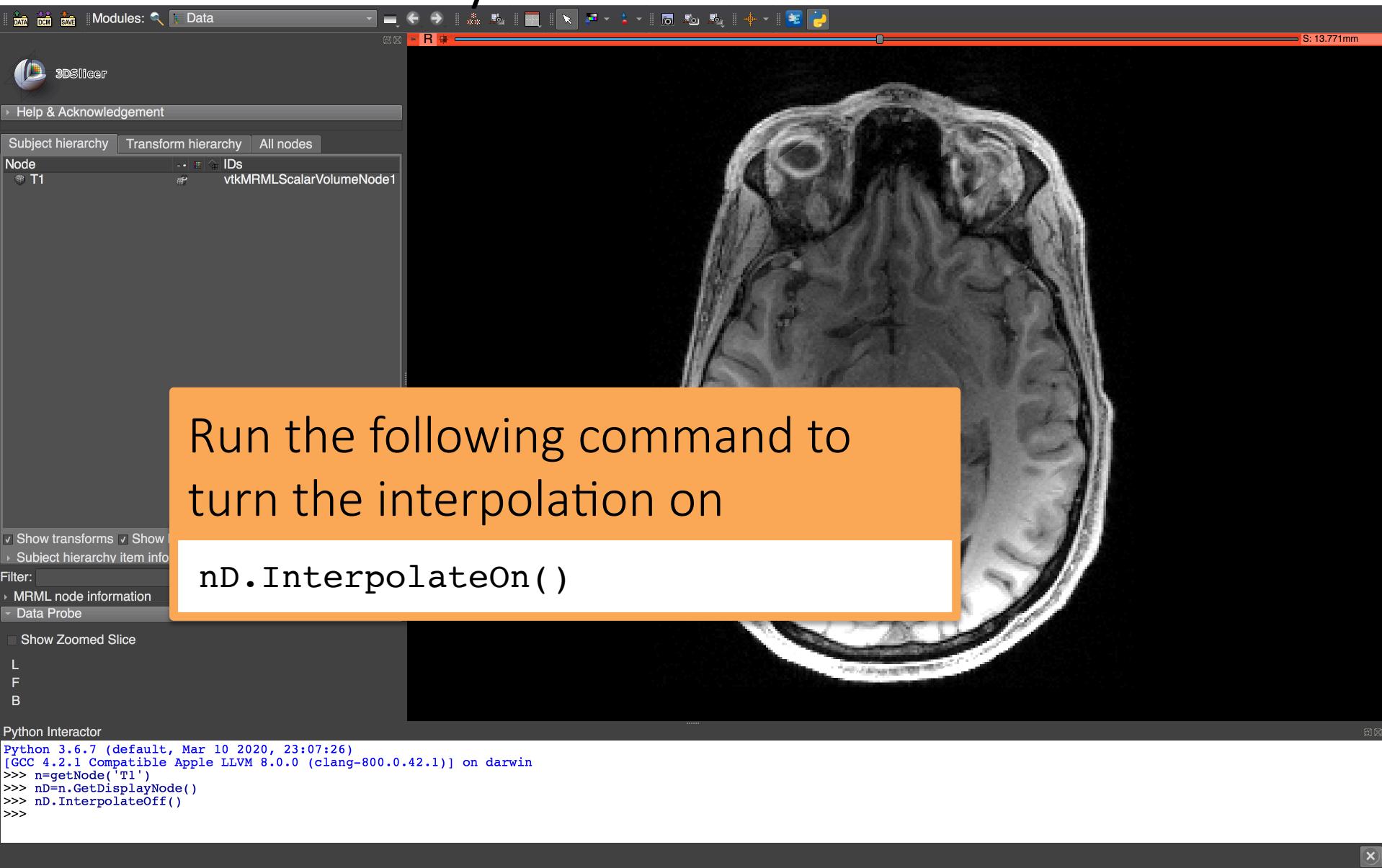
Accessing MRML nodes from the Python interactor



Accessing MRML nodes from the Python interactor



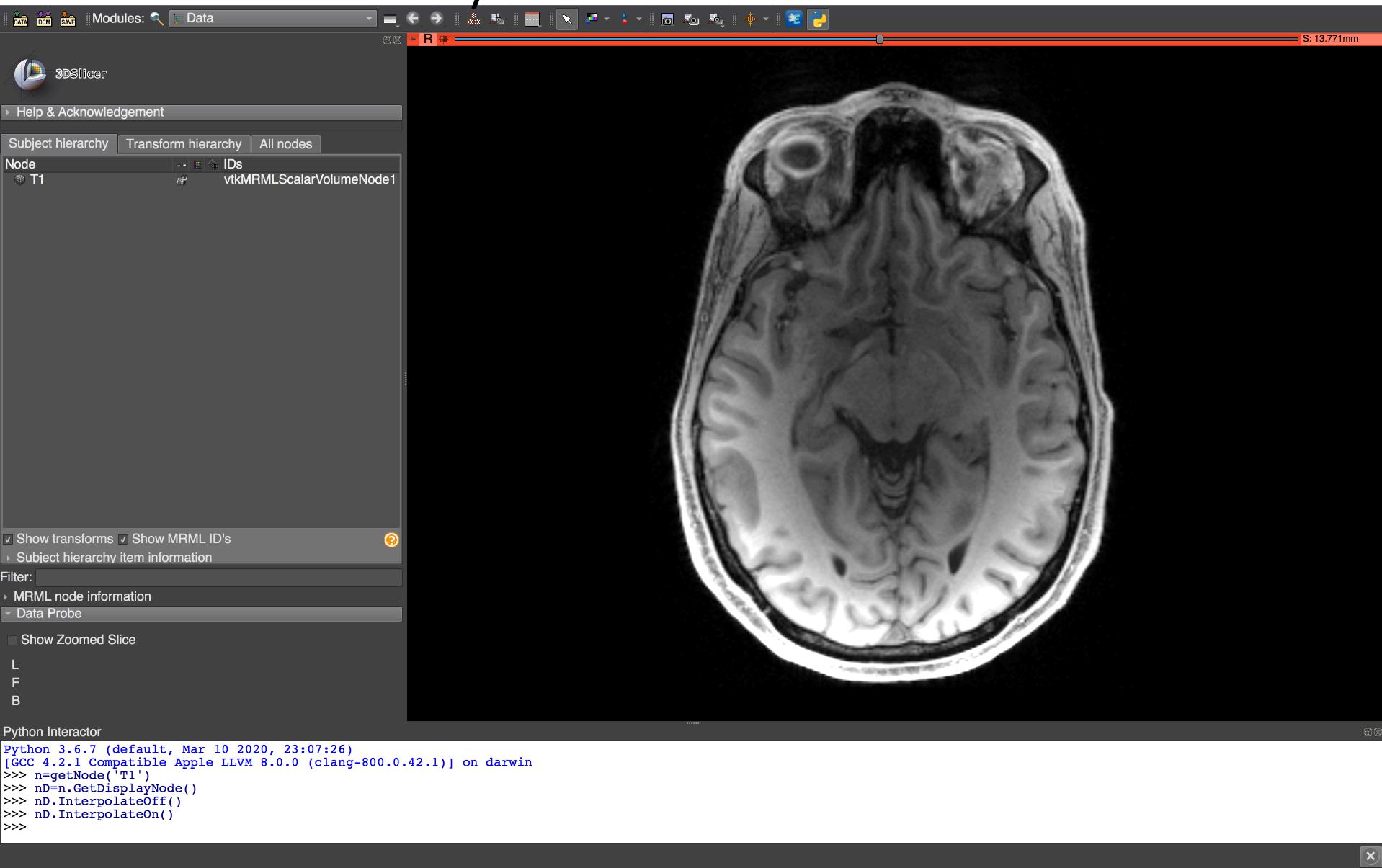
Accessing MRML nodes from the Python interactor



Run the following command to turn the interpolation on

```
nD.InterpolateOn()
```

Accessing MRML nodes from the Python interactor



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 screenshot shows the 3DSlicer application interface. On the left, there's a tree view of the subject hierarchy with a node named 'T1'. At the bottom, a Python Interactor window displays a series of commands related to 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)
```

```
[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 medical image processing application. On the left, the Python Interactor window displays a truncated list of intensity values for the T1 image, starting with:

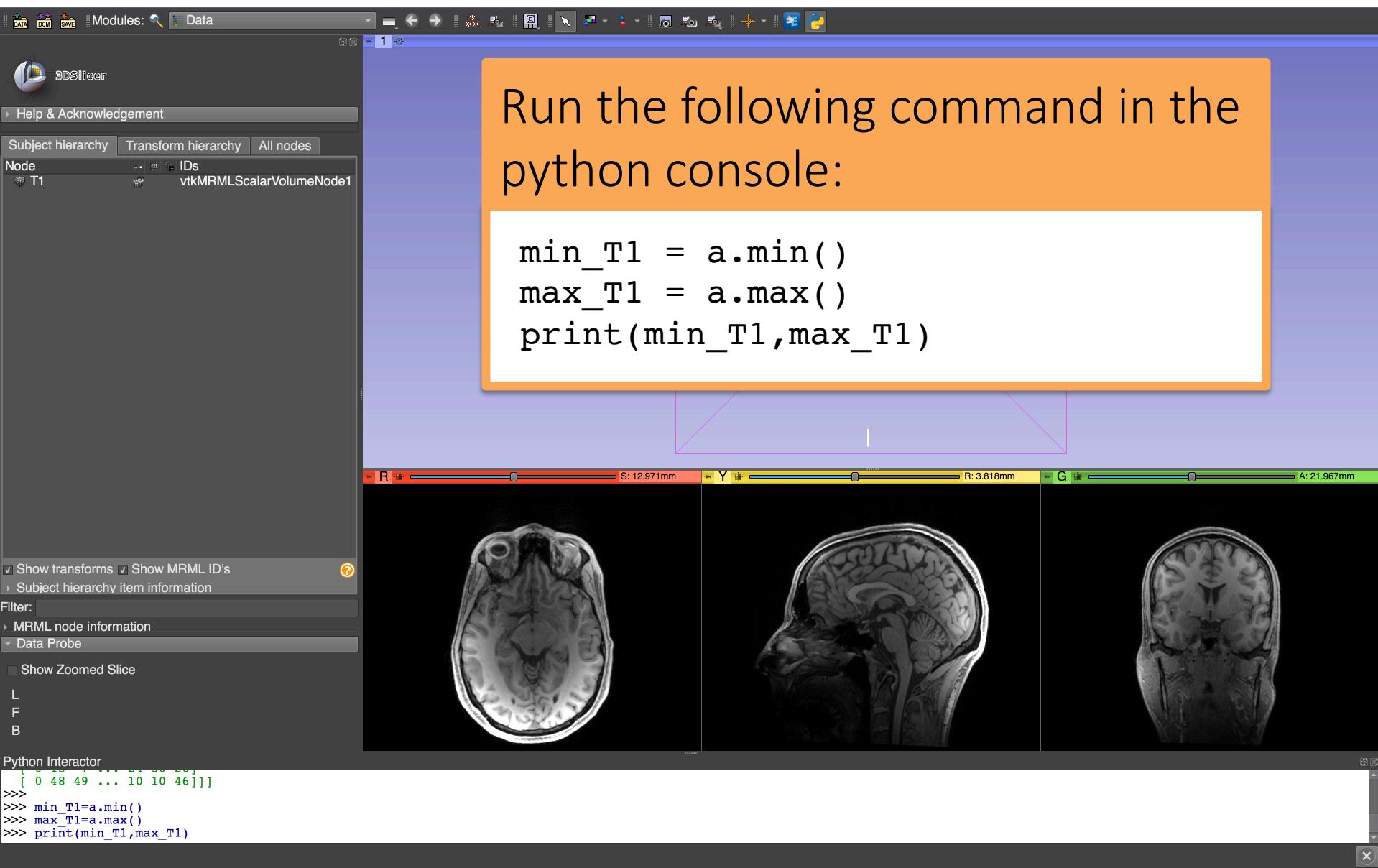
```
>>> a = slicer.util.array('T1')
>>> print(a)
[[[ 0  0  0 ...  0  0  0]
 [ 0 20  6 ... 10 52 27]
 [ 0 24 25 ...  4 32  8]
 ...
 [[ 0 48 14 ... 41 42 21]
 [ 0 15 40 ... 33 38 25]
 [ 0 55 19 ... 21  7 17]]
 ...
 [[ 0  0  0 ...  0  0  0]
 [ 0  4 14 ... 30 17 42]
 [ 0 22  9 ... 11 12 49]
 ...
 [[ 0 86 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, a 3D volume rendering of a brain scan is shown with a 3D coordinate system overlaid. The axes are labeled R (Right), S (Superior), and I (Inferior). The image is displayed in three orthogonal planes (Sagittal, Coronal, Axial) at the bottom.

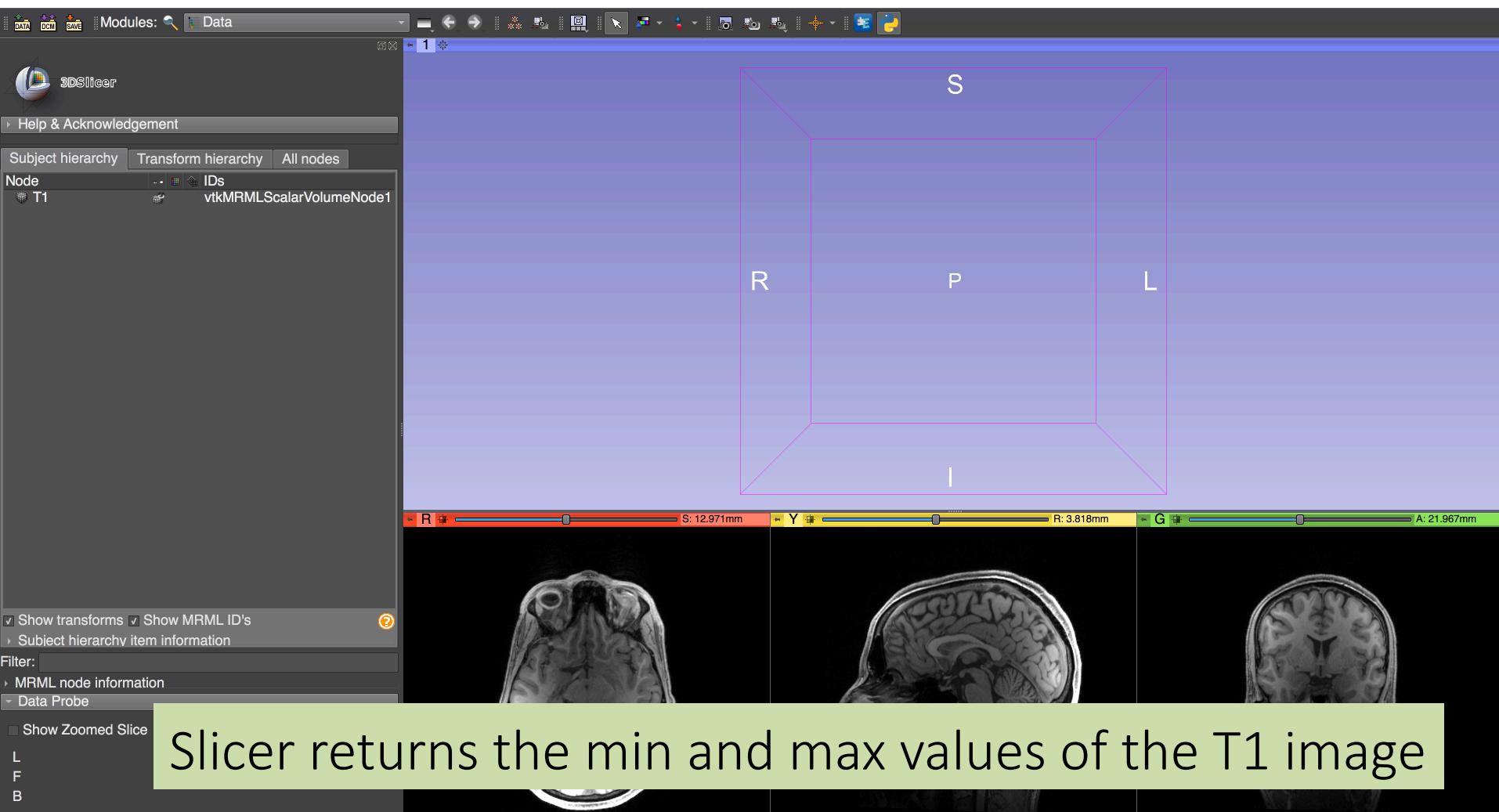
Slicer prints the intensity values of the T1 image

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

Accessing voxels in a volume

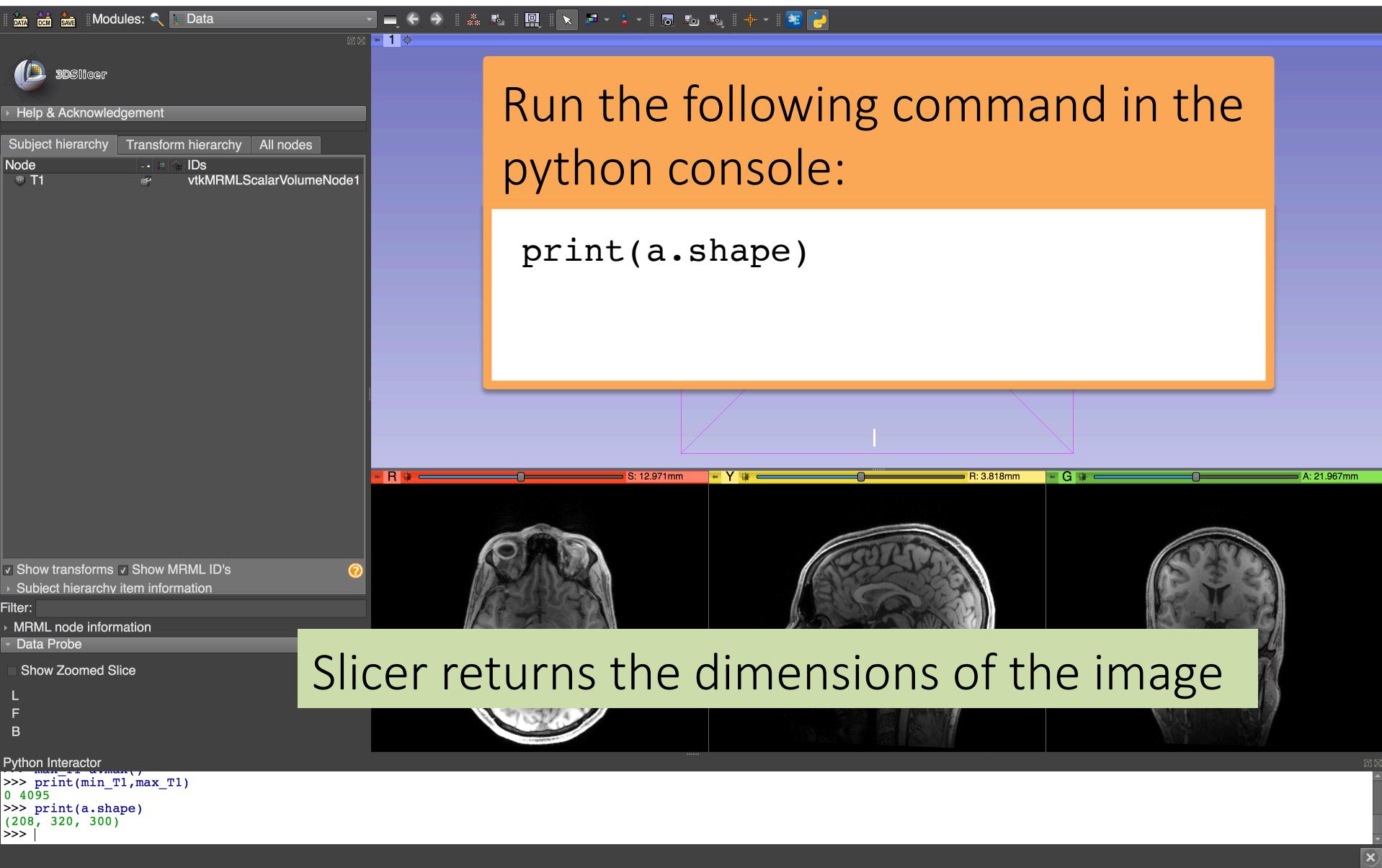


Accessing voxels in a volume

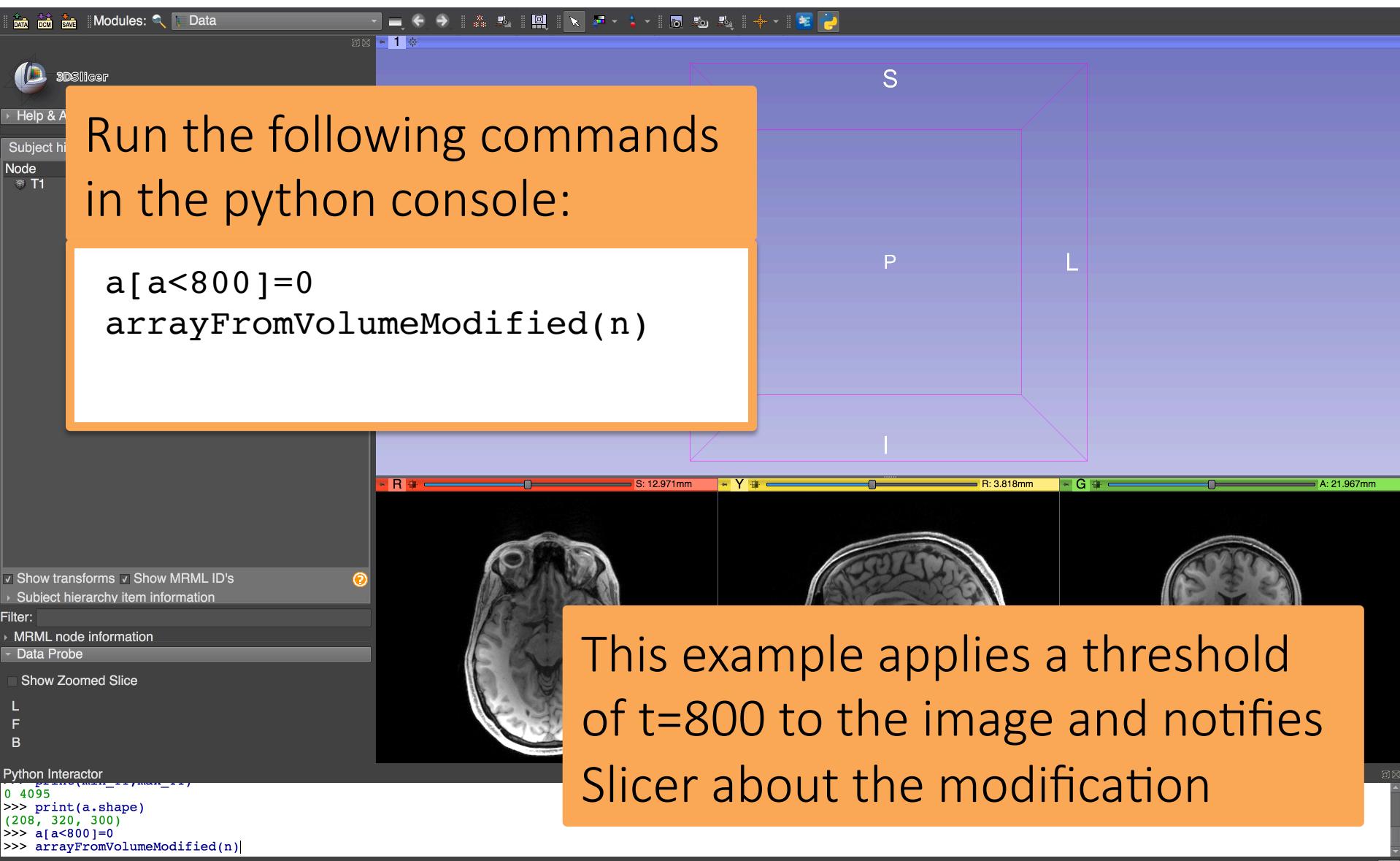


Slicer returns the min and max values of the T1 image

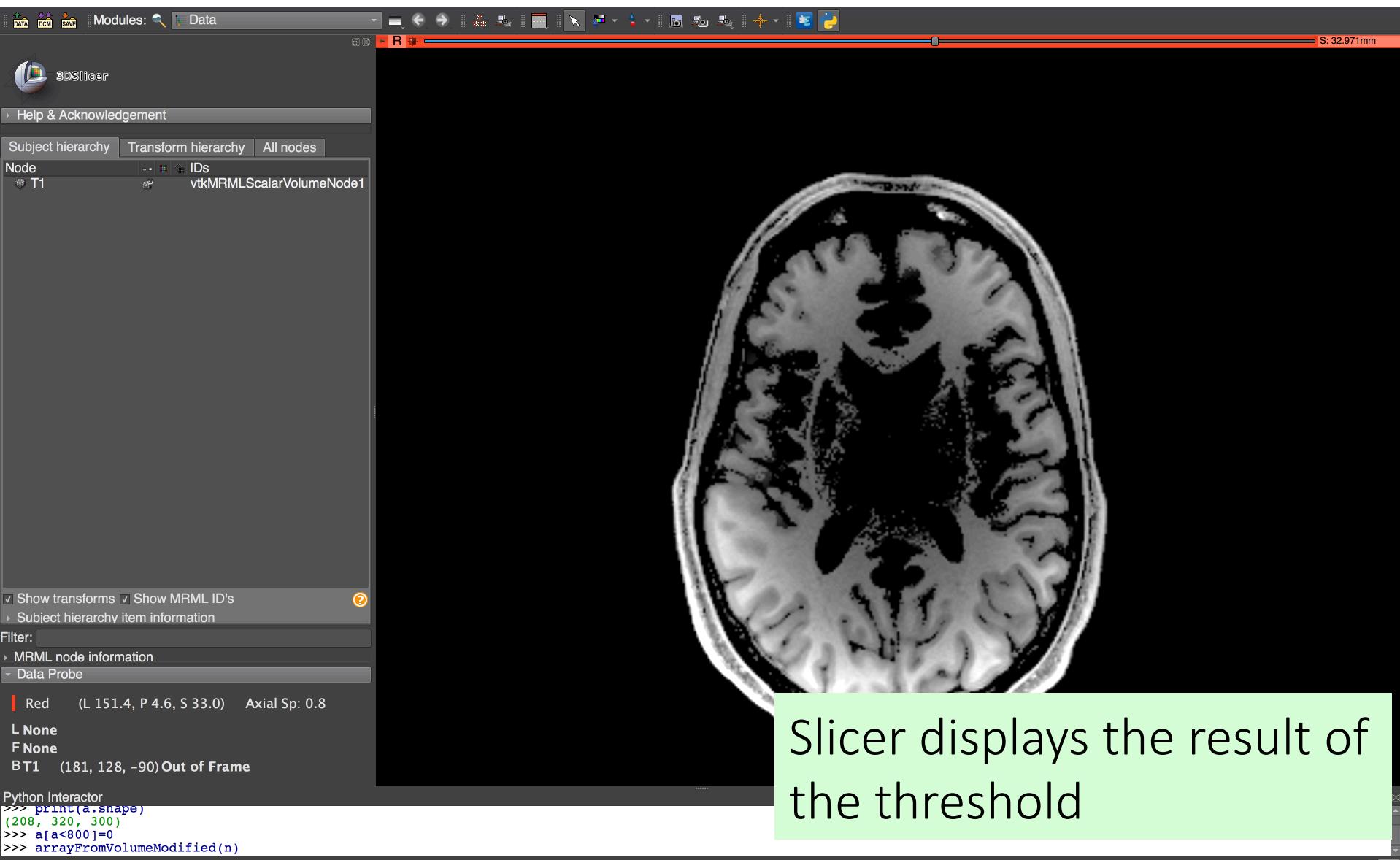
Modifying voxels in a volume



Modifying voxels in a volume

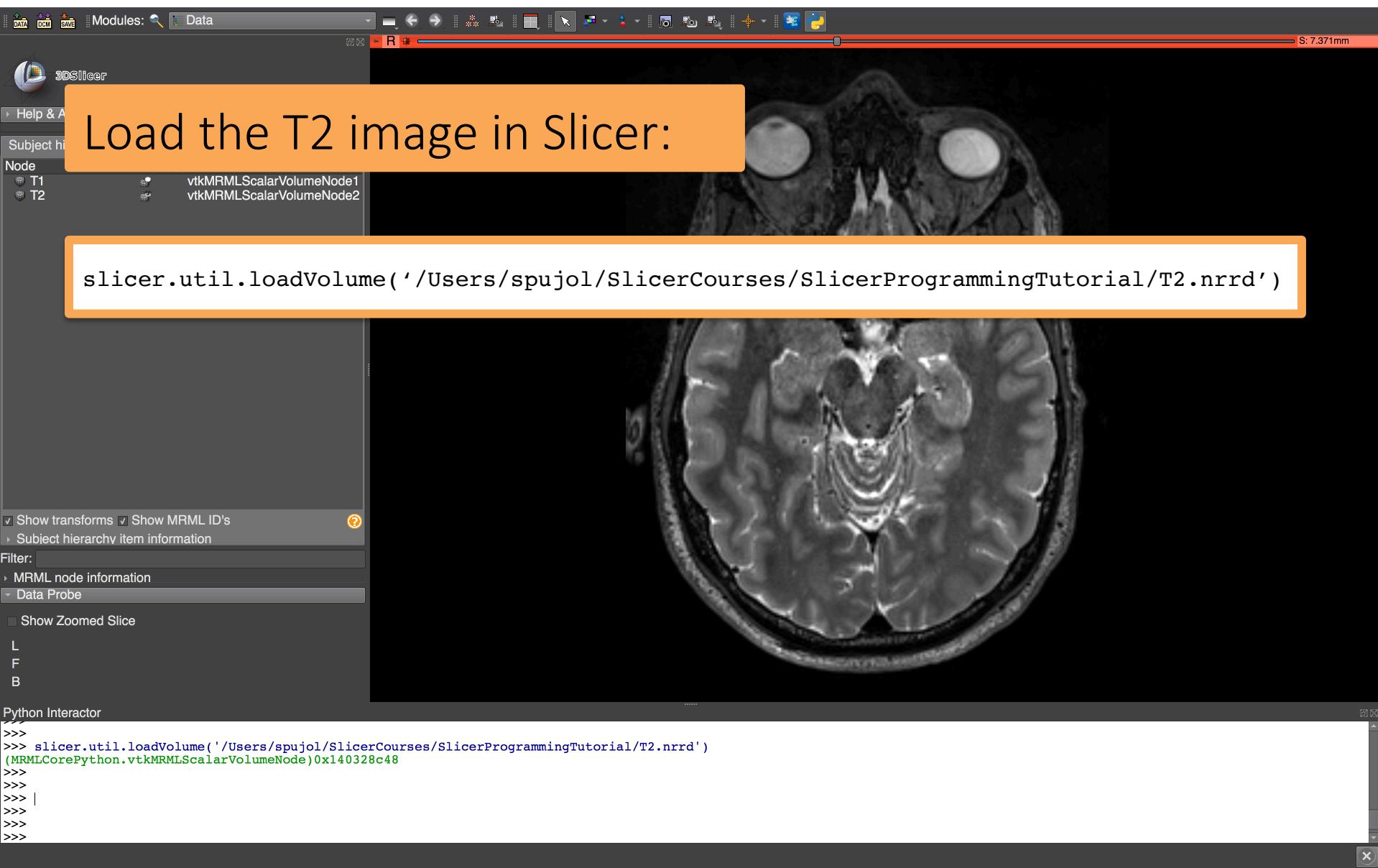


Modifying voxels in a volume



Slicer displays the result of
the threshold

Loading the T2 volume



Python function: threshold

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

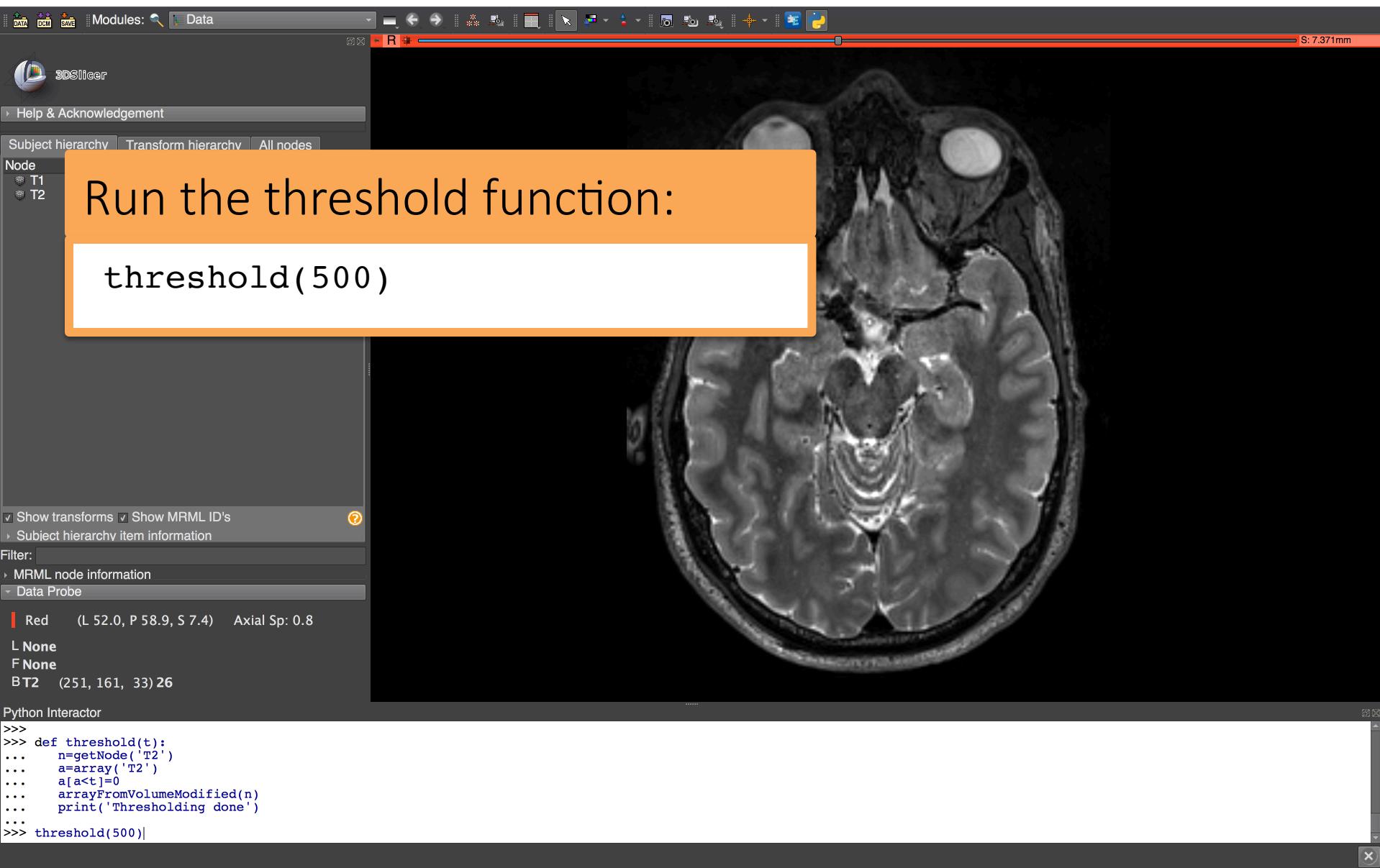
```
def threshold(t):
    n=getNode('T2')
    a=array('T2')
    a[a<t]=0
    arrayFromVolumeModified(n)
    print('Thresholding done')
```

Red (R 145.5, P 86.4, S 7.4) Axial Sp: 0.8
L None
F None
BT2 (286, 167, 280) Out of Frame

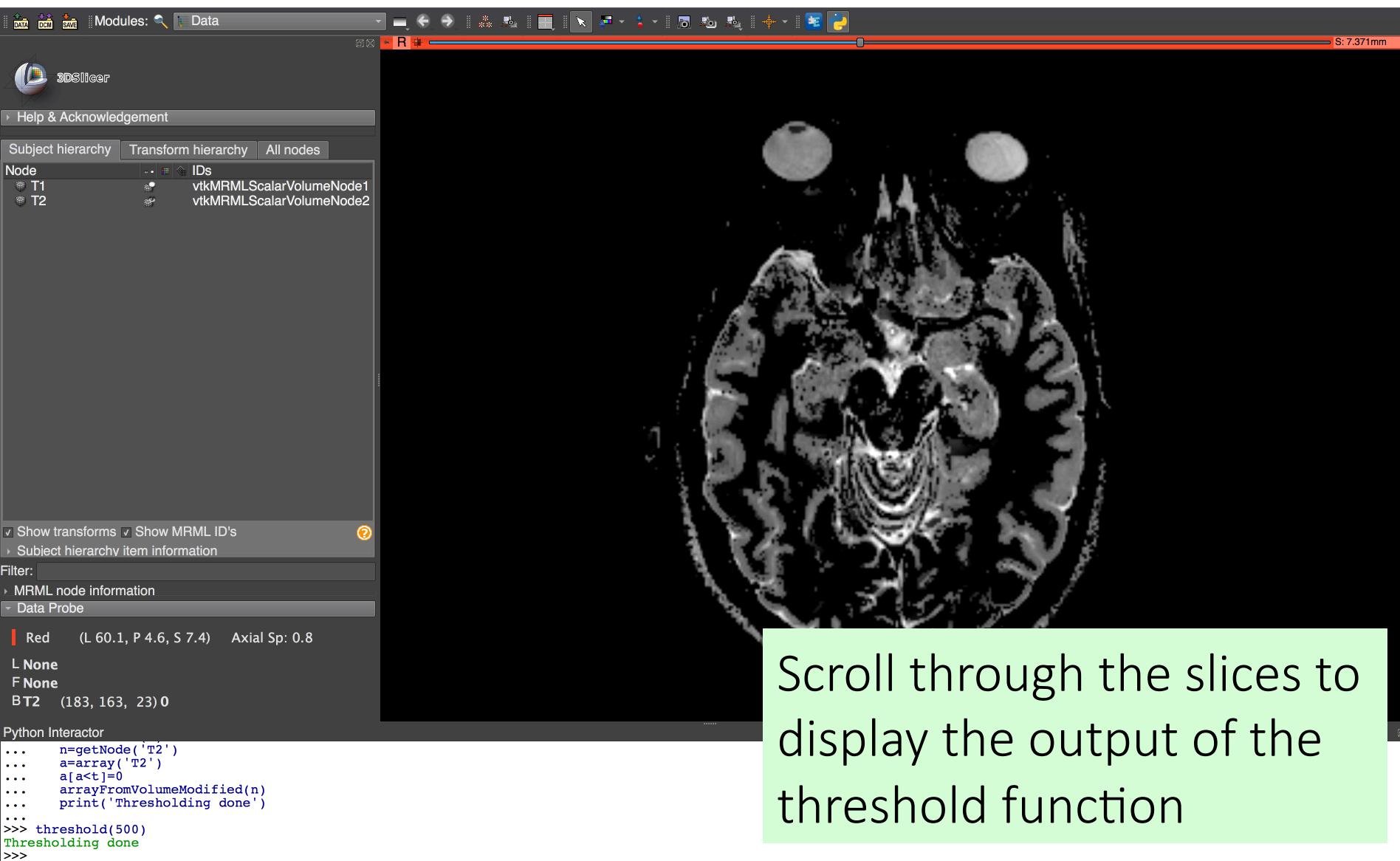
Python Interactor

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

Python function: threshold



Python function: threshold



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 3DSlicer application interface. On the left, the 3D Slicer node browser displays a subject hierarchy with nodes T1 and T2, and their corresponding MRML scalar volume nodes. The Python Interactor window on the right contains a code snippet for creating a toggle() function.

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)
```

```
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

The screenshot shows the 3DSlicer application interface. On the left is the 3D Slicer module panel, which includes a node browser, transform browser, and various settings like 'Show transforms' and 'Show MRML ID's'. In the center is a Python interactor window with a title bar 'R'. The main area of the interactor contains the following text:

Create a QPushButton in the Python interactor

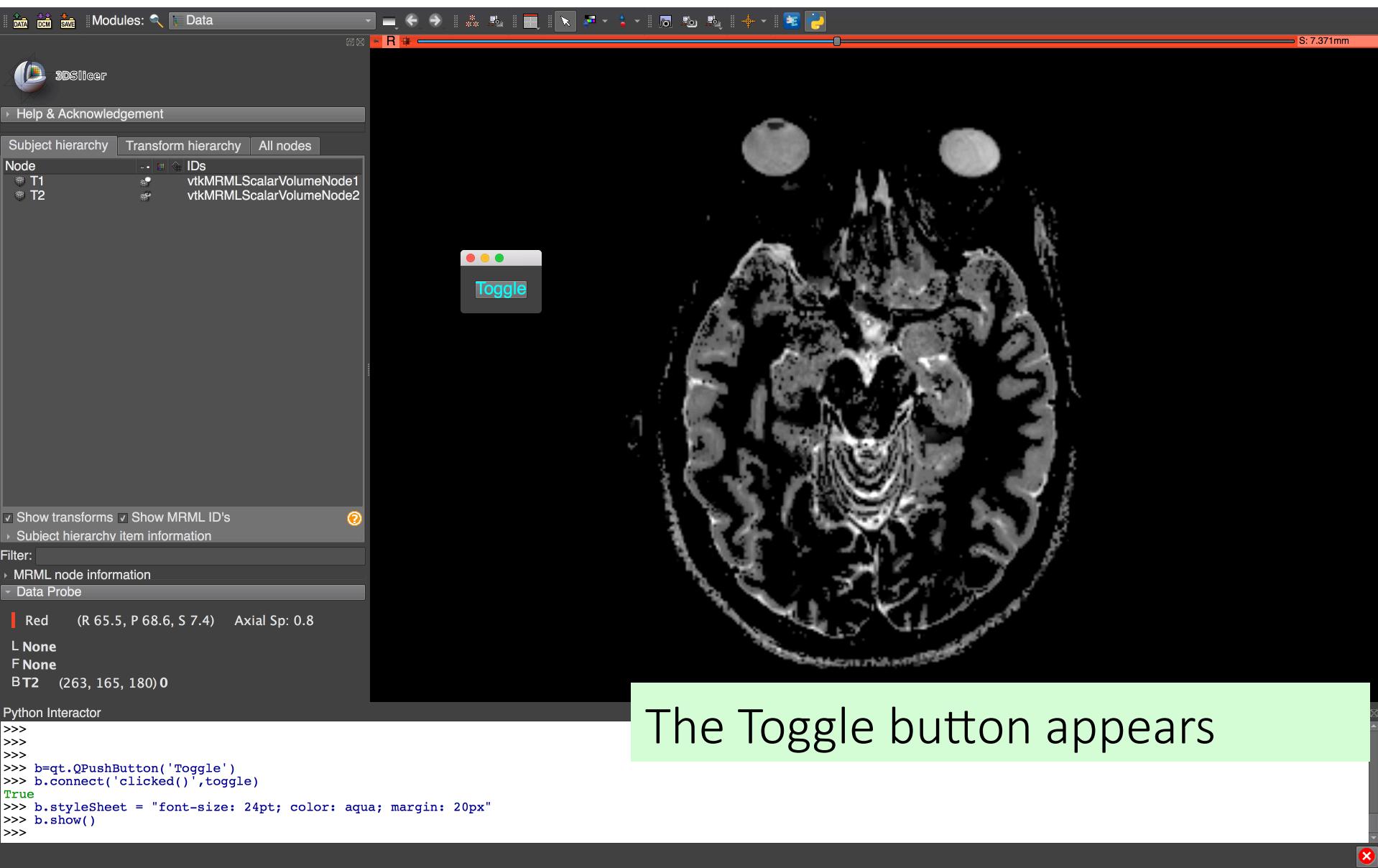
```
b=qt.QPushButton('Toggle')
b.connect('clicked()',toggle)
b.setStyleSheet = "font-size: 24pt;
color: aqua; margin: 20px"
b.show()
```

At the bottom right of the interactor window, there is a callout bubble with the text 'styleSheet is css'.

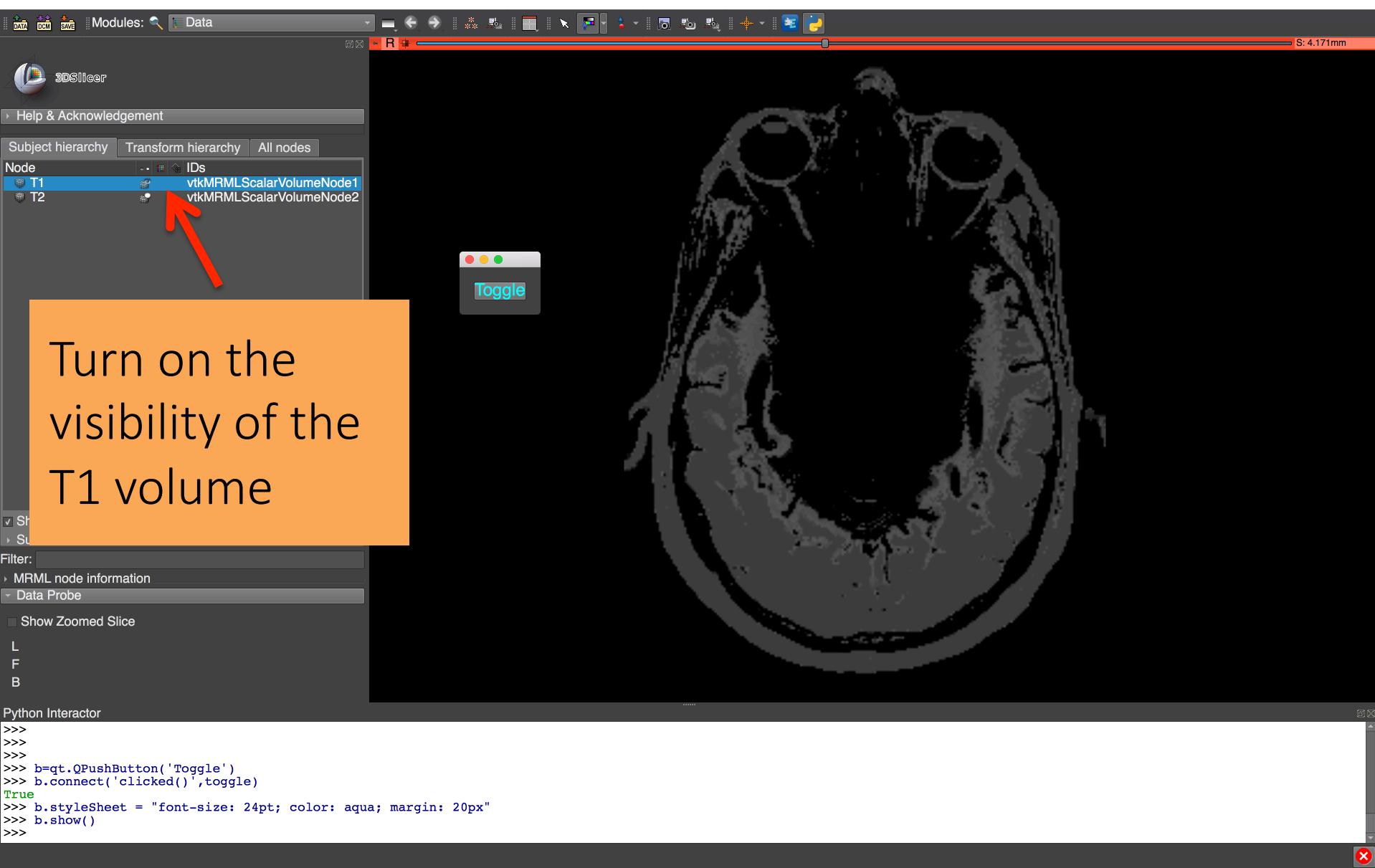
In the bottom left corner of the main 3DSlicer window, there is a Python interactor window showing the same code. The output of the code execution is visible in the interactor window.

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

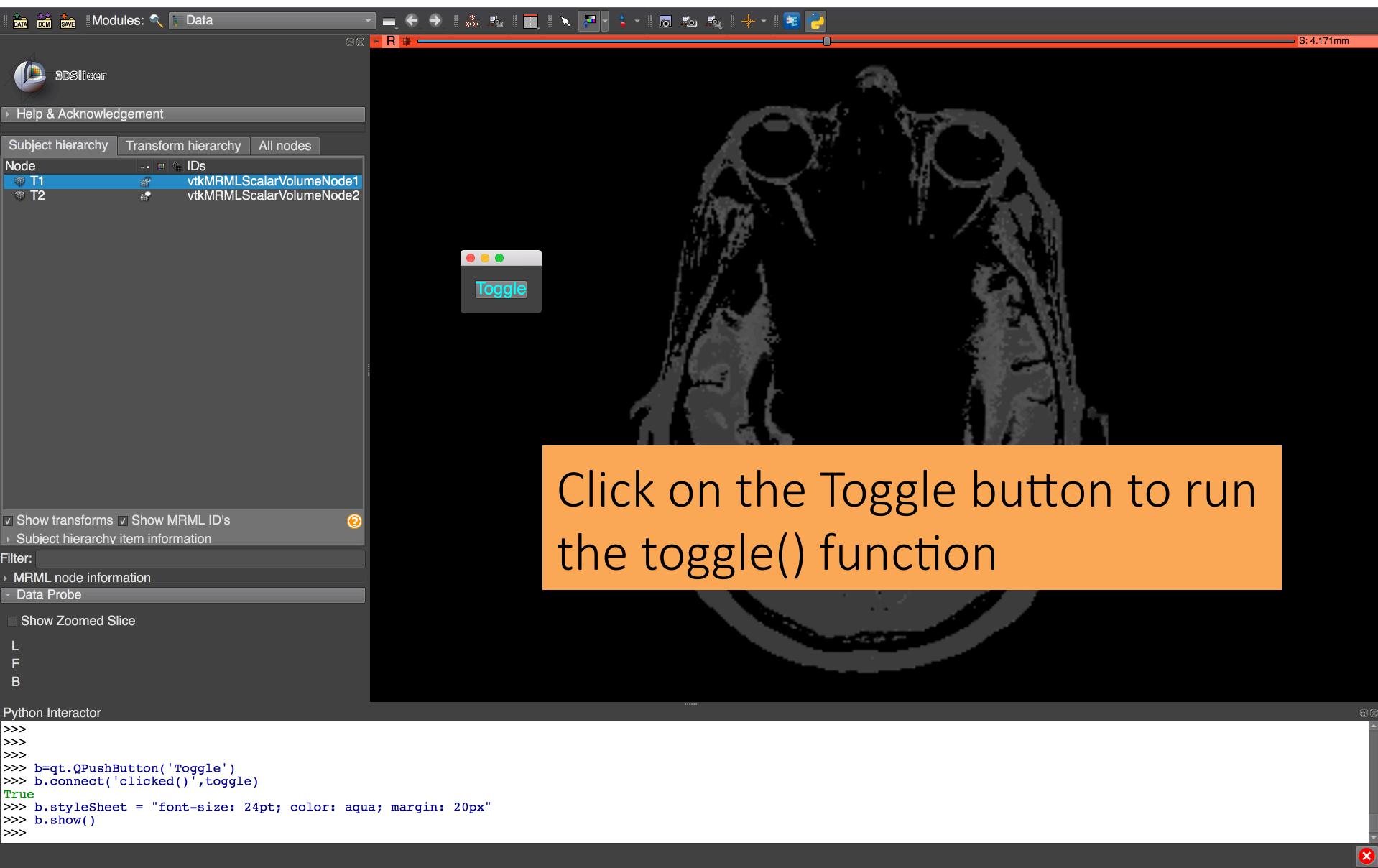
Creating a Qt Push Button



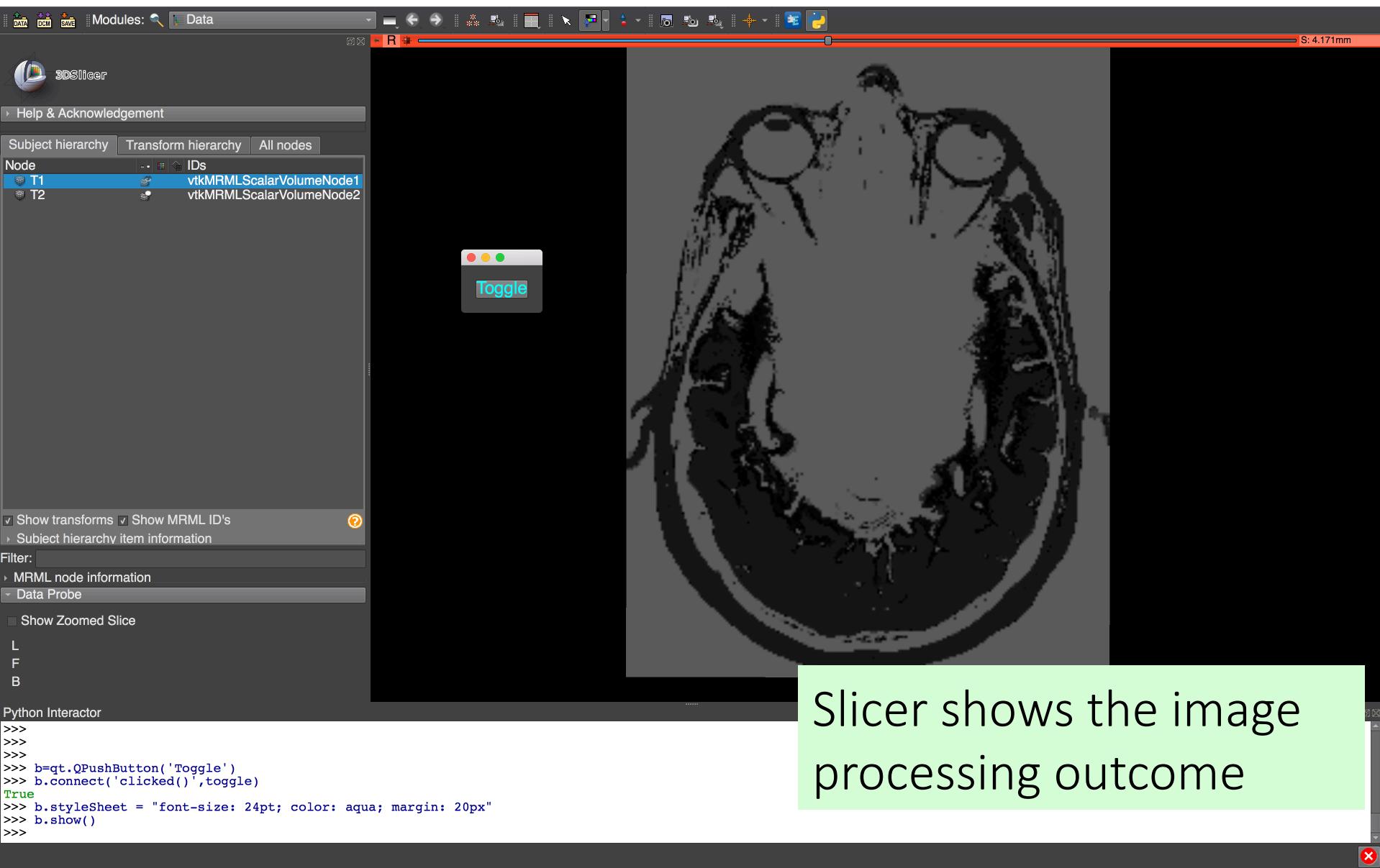
Creating a Qt Push Button



Creating a Qt Push Button



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 module 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](https://www.slicer.org/wiki/Documentation/Nightly/ScriptRepository)

Conclusion

- Slicer enables developees 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