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.

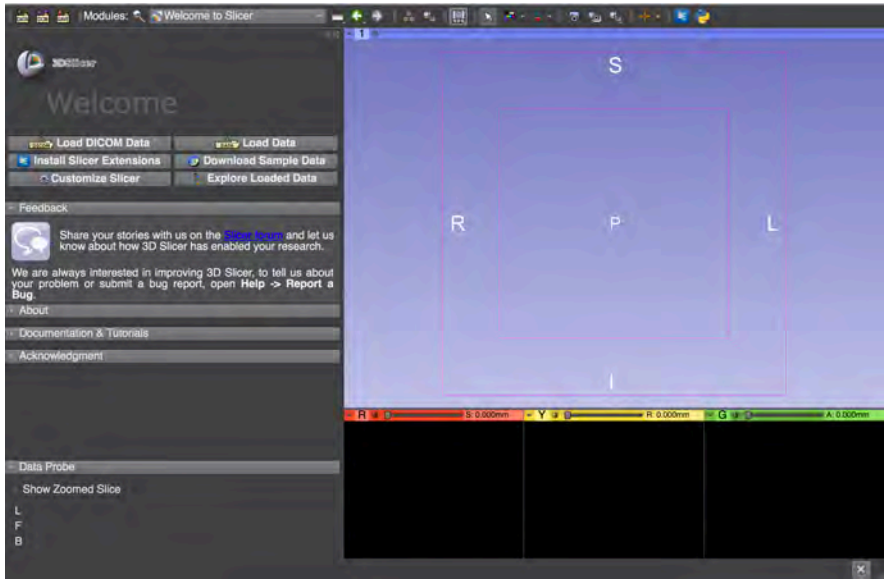
Tutorial Outline

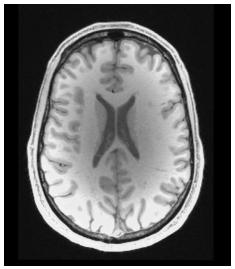
- Part 1: Slicer Modules Overview
- Part 1: Getting Familiar with the Python environment in 3D Slicer
- Part 2: Getting Familiar with Qt in 3D Slicer

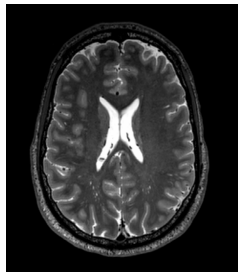
Disclaimer

- 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





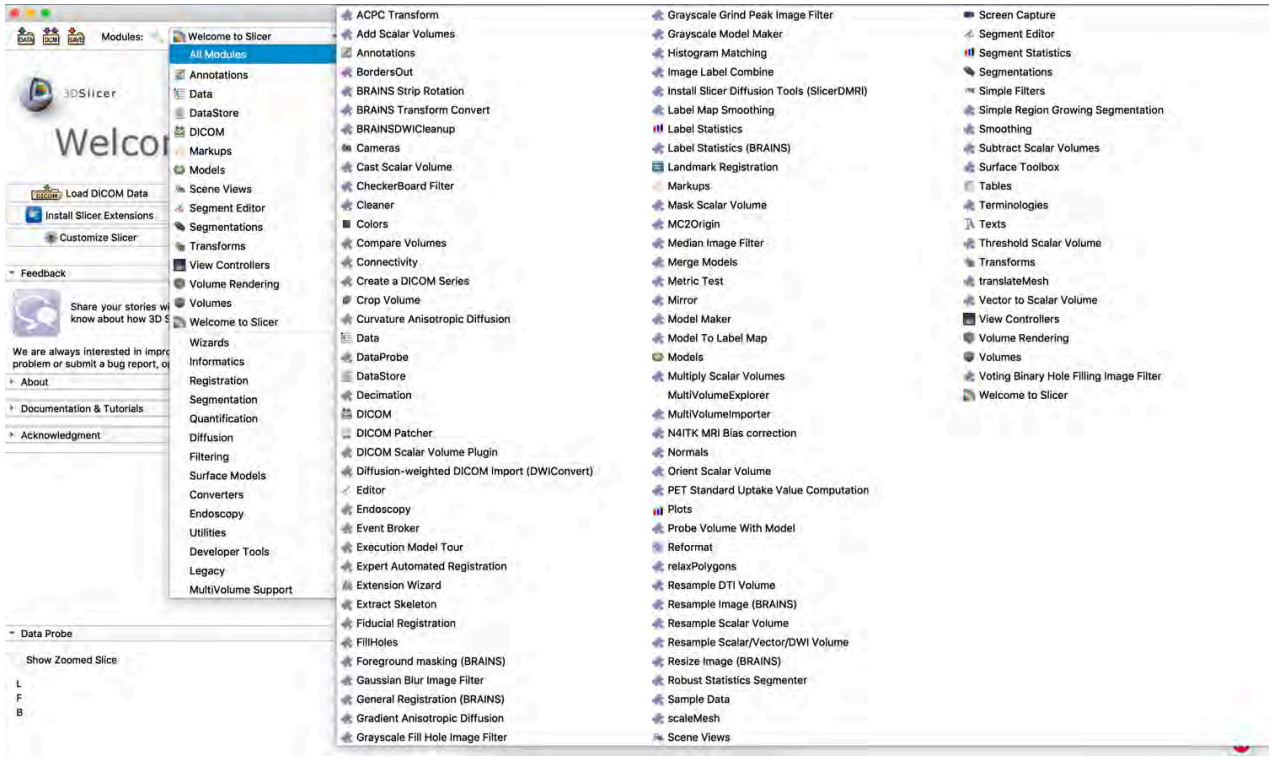


3D Slicer release version 5.0

SlicerDeveloperTutorial.zip

Part 1

Slicer Modules Overview



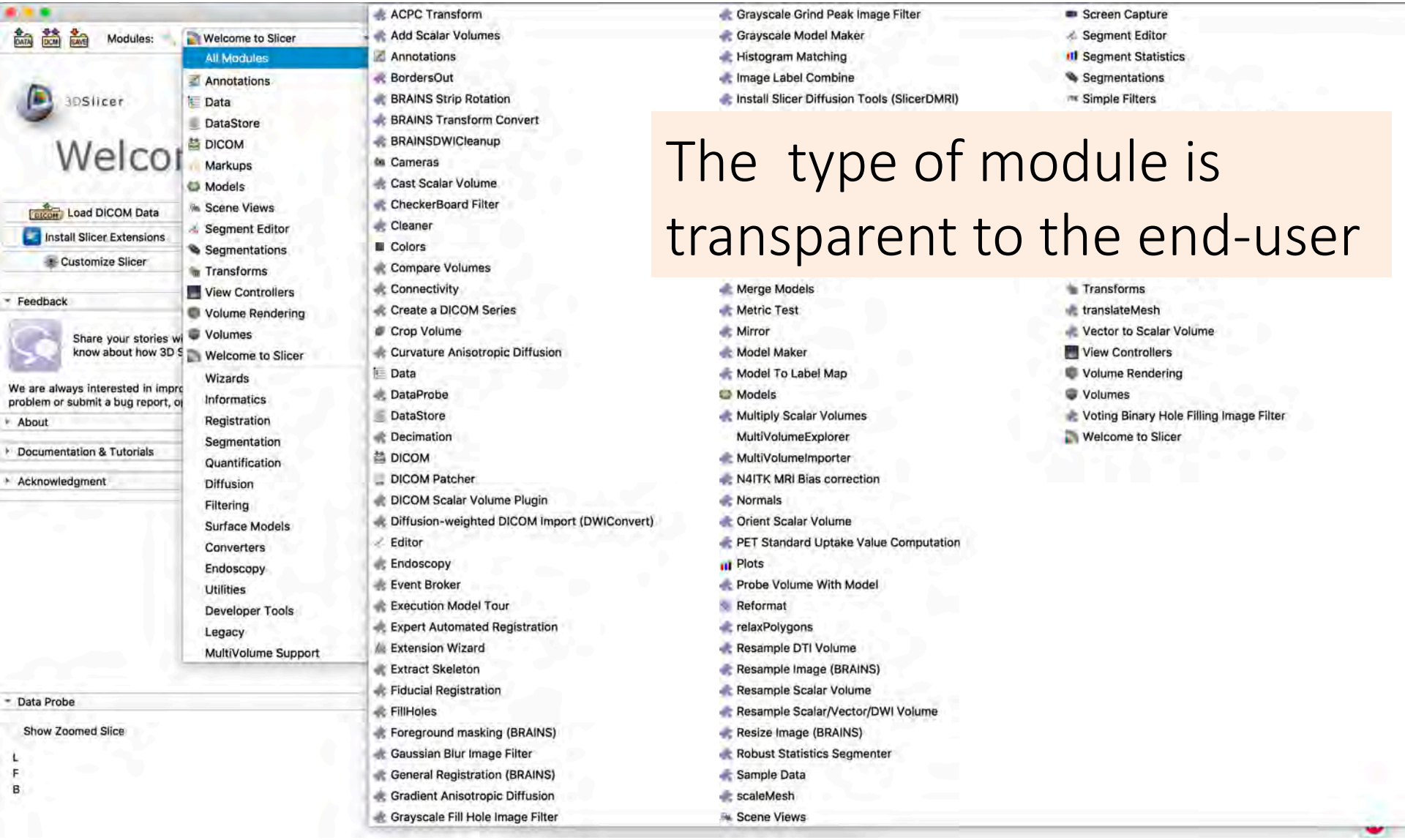
Slicer Modules

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

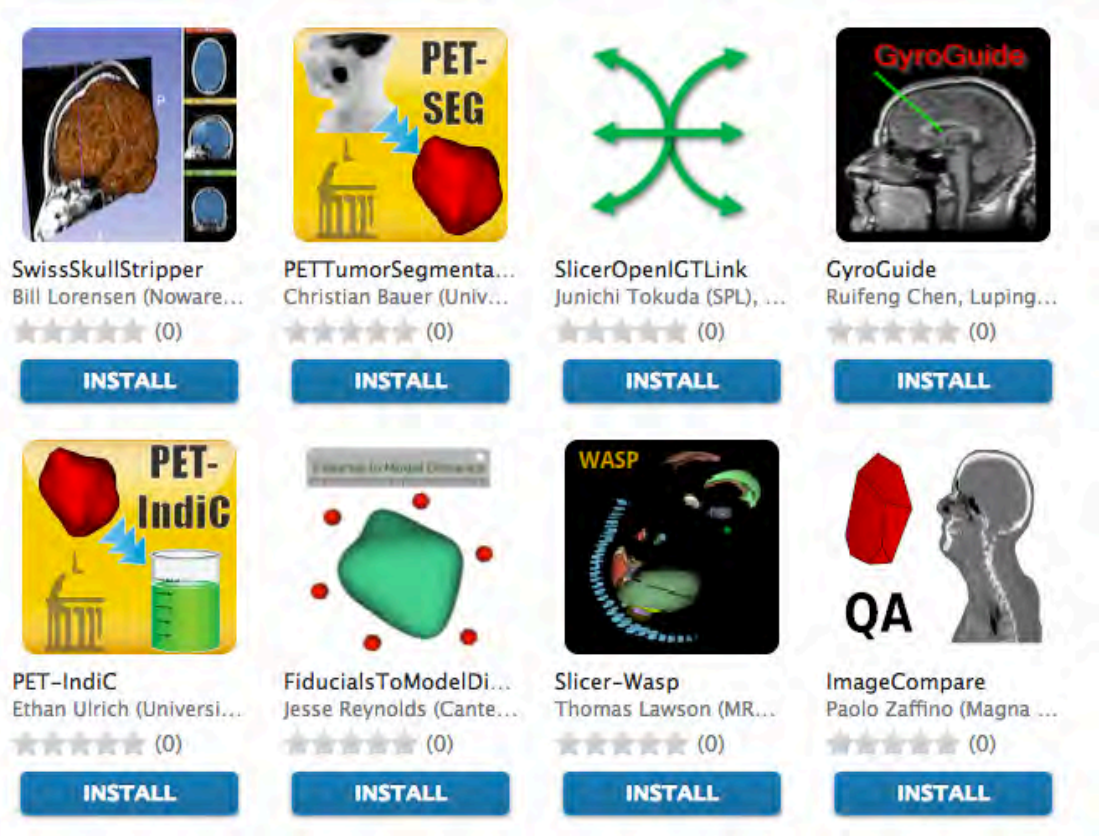
Slicer5 Modules



The type of module is transparent to the end-user

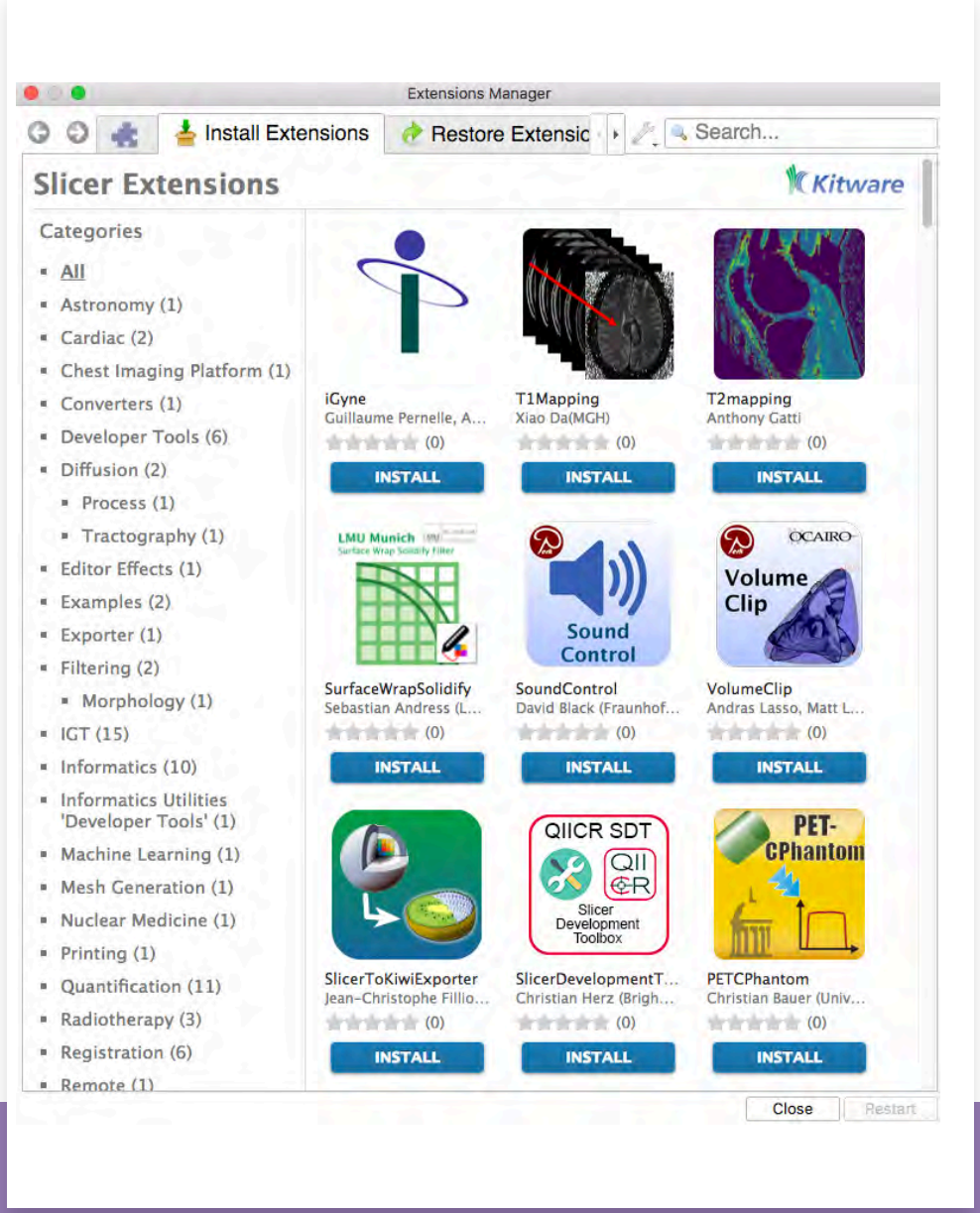
Slicer Extensions

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



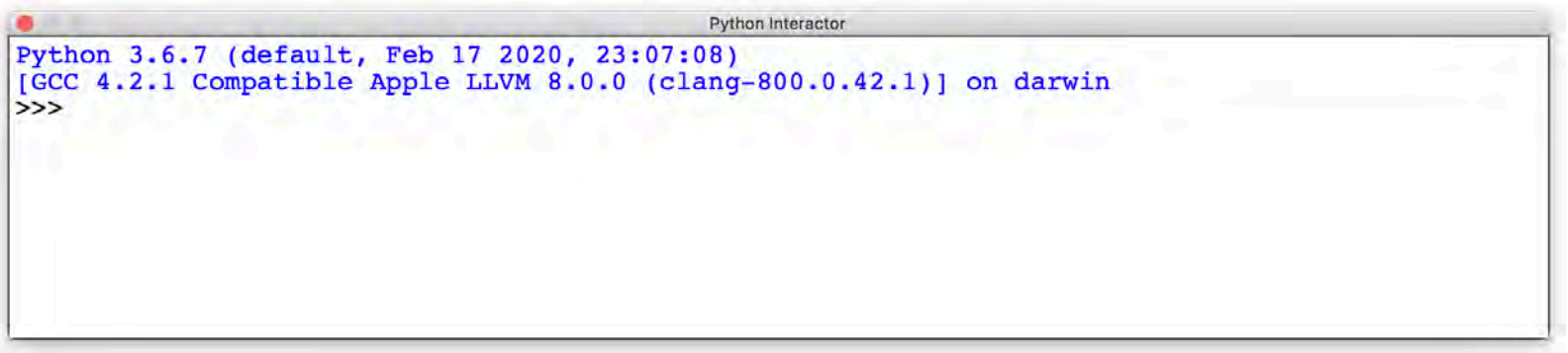
Slicer Extension Manager

- 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 in Slicer

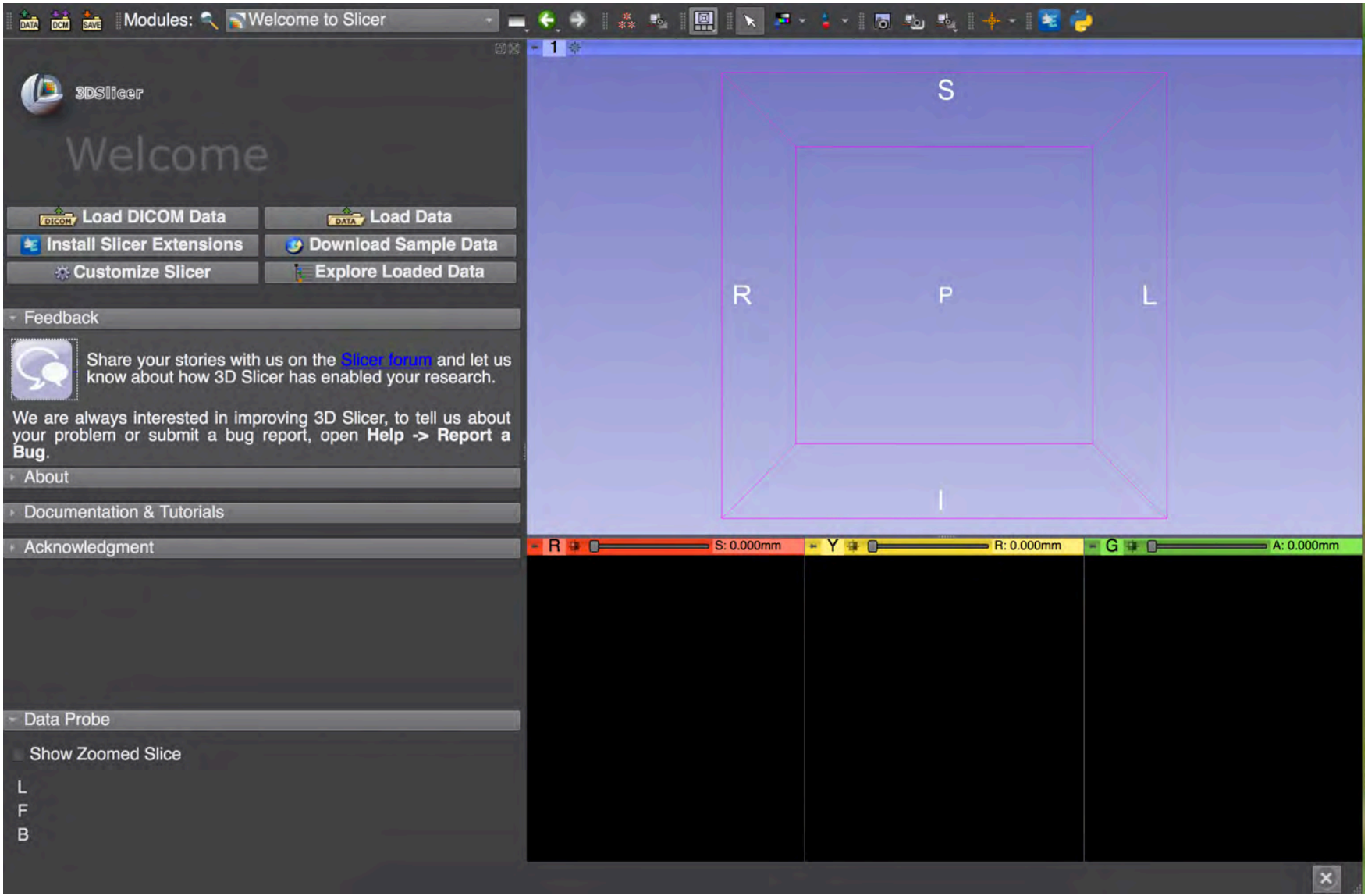
Slicer5 works with **Python3** and a rich set of standard libraries

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

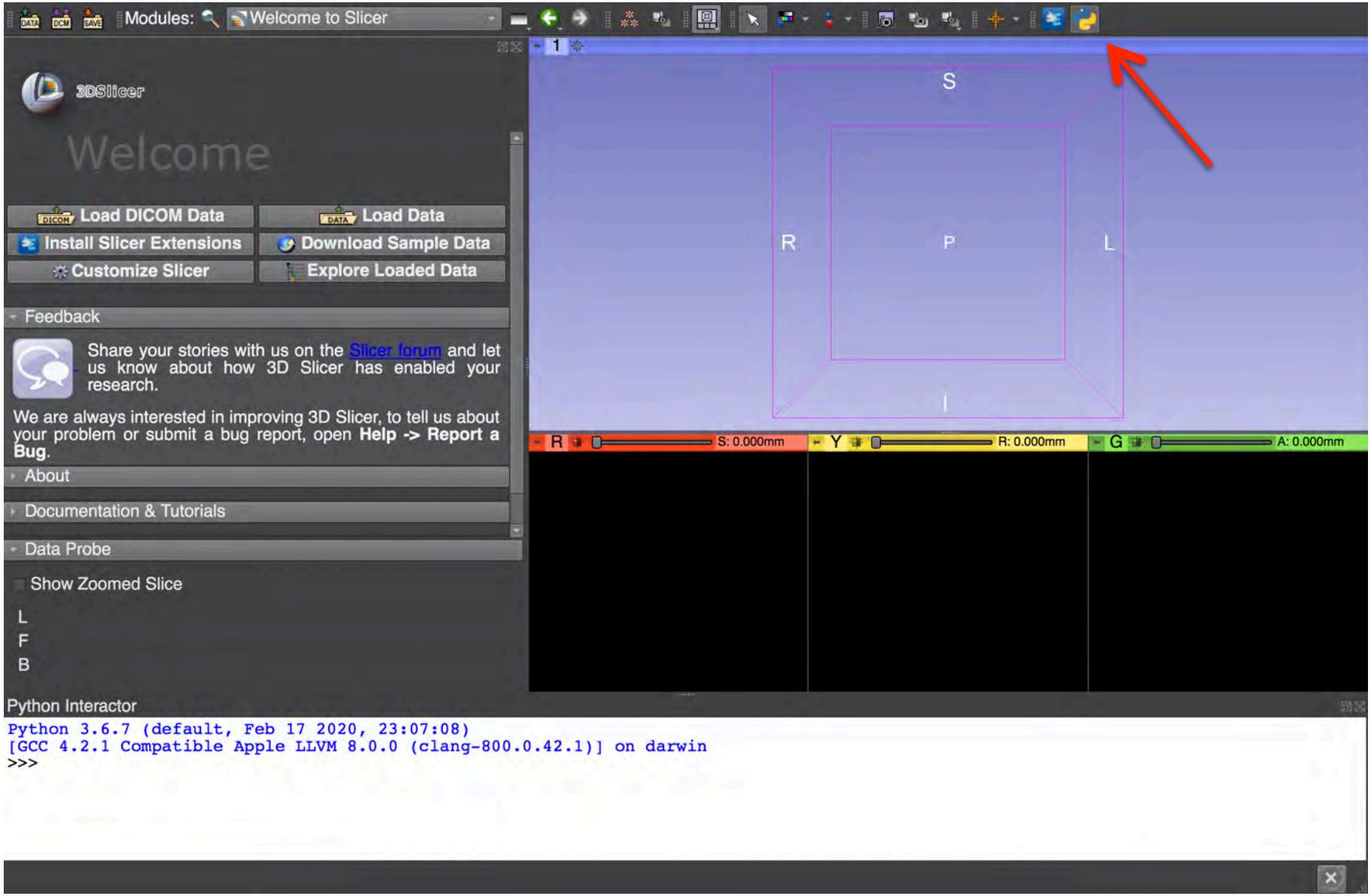
- The **pip install** command in Slicer enables developers to install most common scientific computing tools (e.g. TensorFlow, SciPy, PyTorch, Pandas, etc.)
- Slicer can be used as a **Jupyter notebook** kernel
- PyCharm and other Python development tools can be used with Slicer



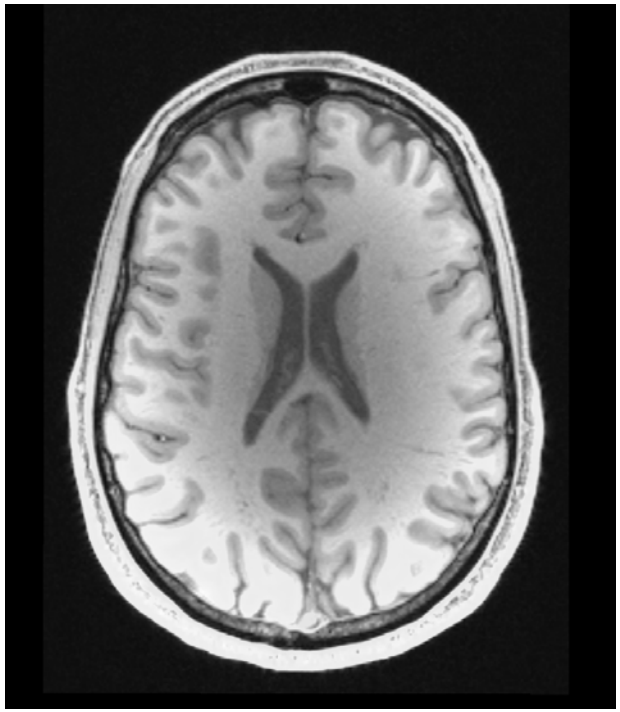
Slicer release version 5 integrates Python3, VTK5 and ITK5

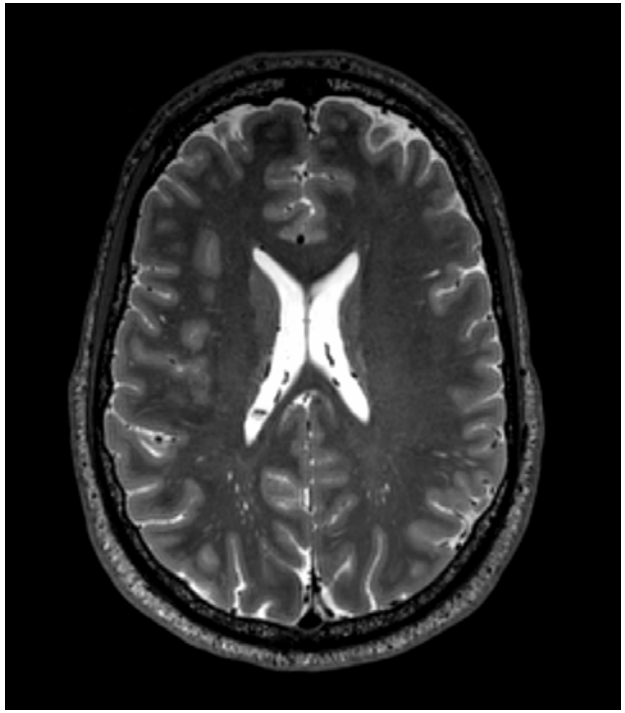
The Python Console in Slicer

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



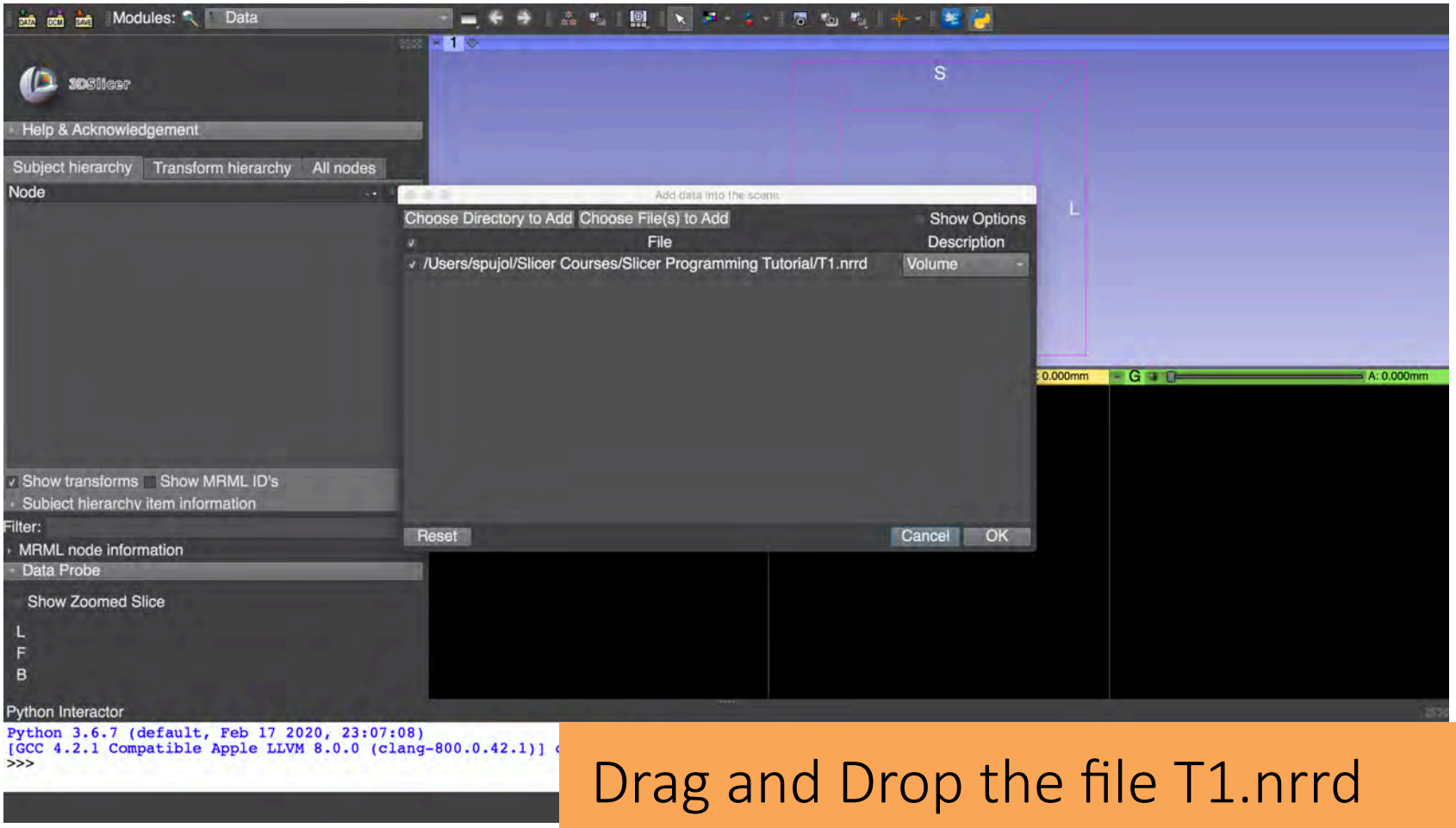
To access the Python Interactor, click on the Python icon in the top bar menu of Slicer





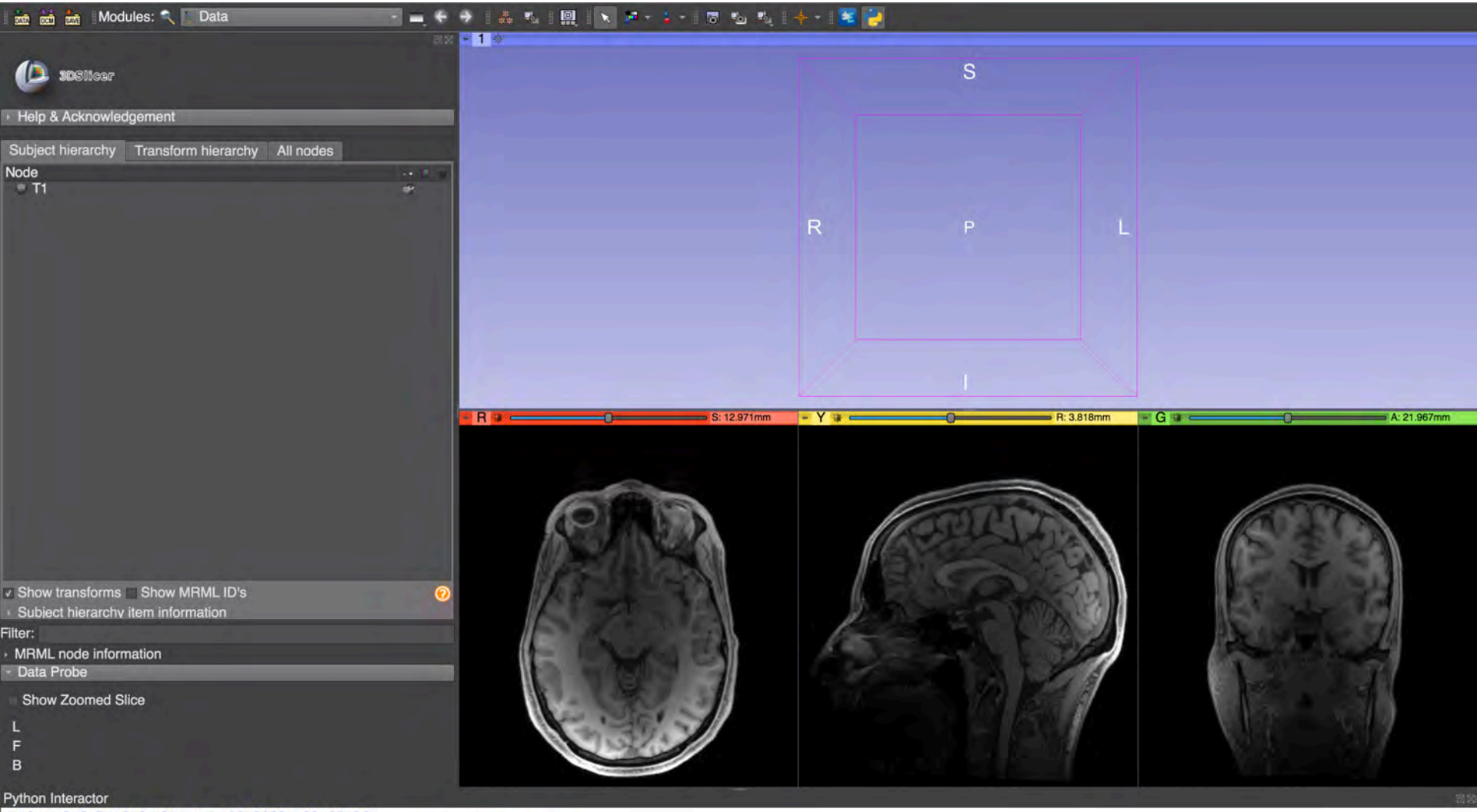
The Slicer Programming tutorial dataset includes a T1-weighted and a T2-weighted MRI scan of a healthy subject

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 are available on the Internet that you could visualize and analyze with Slicer

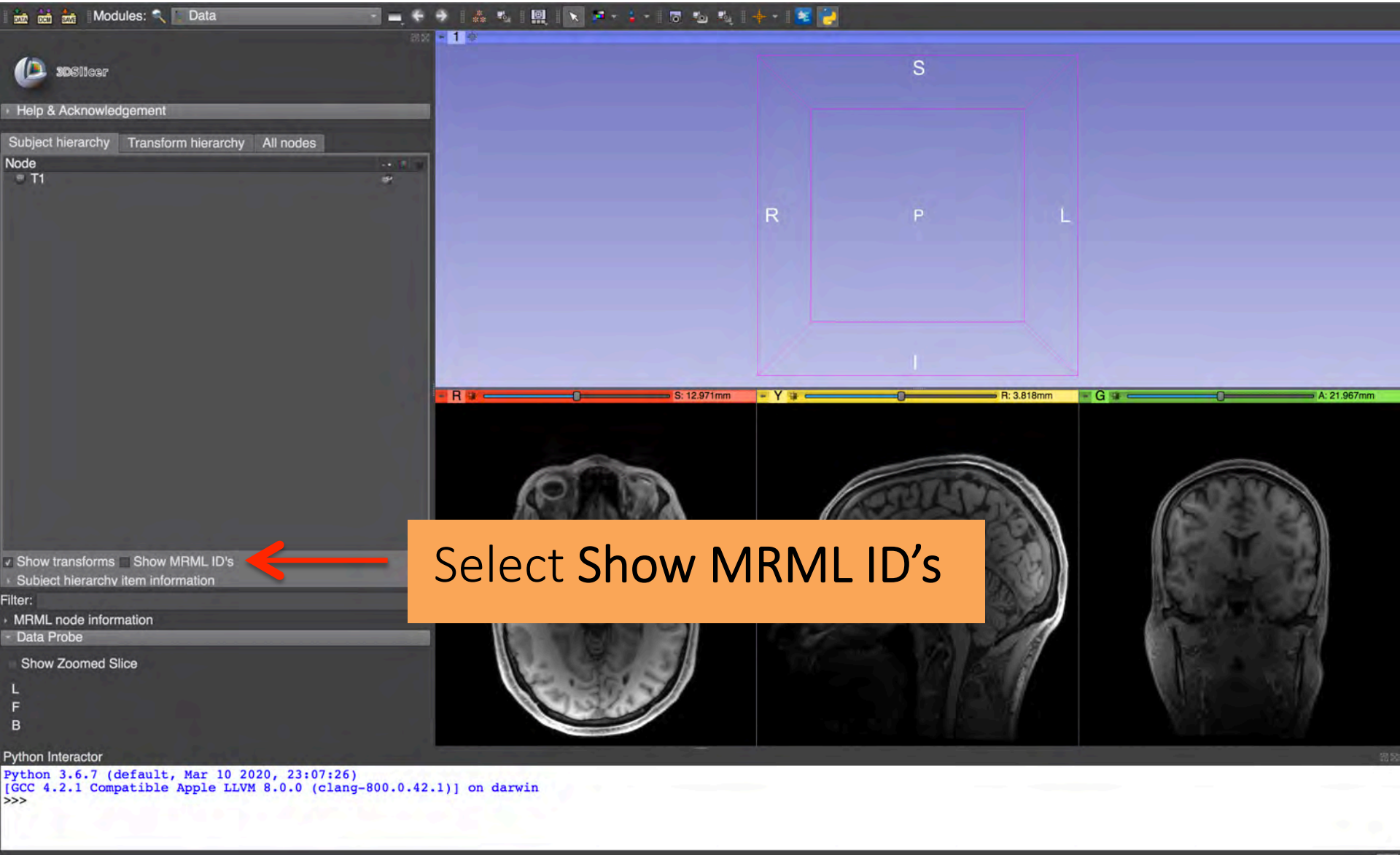
Slicer Data Model

- 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

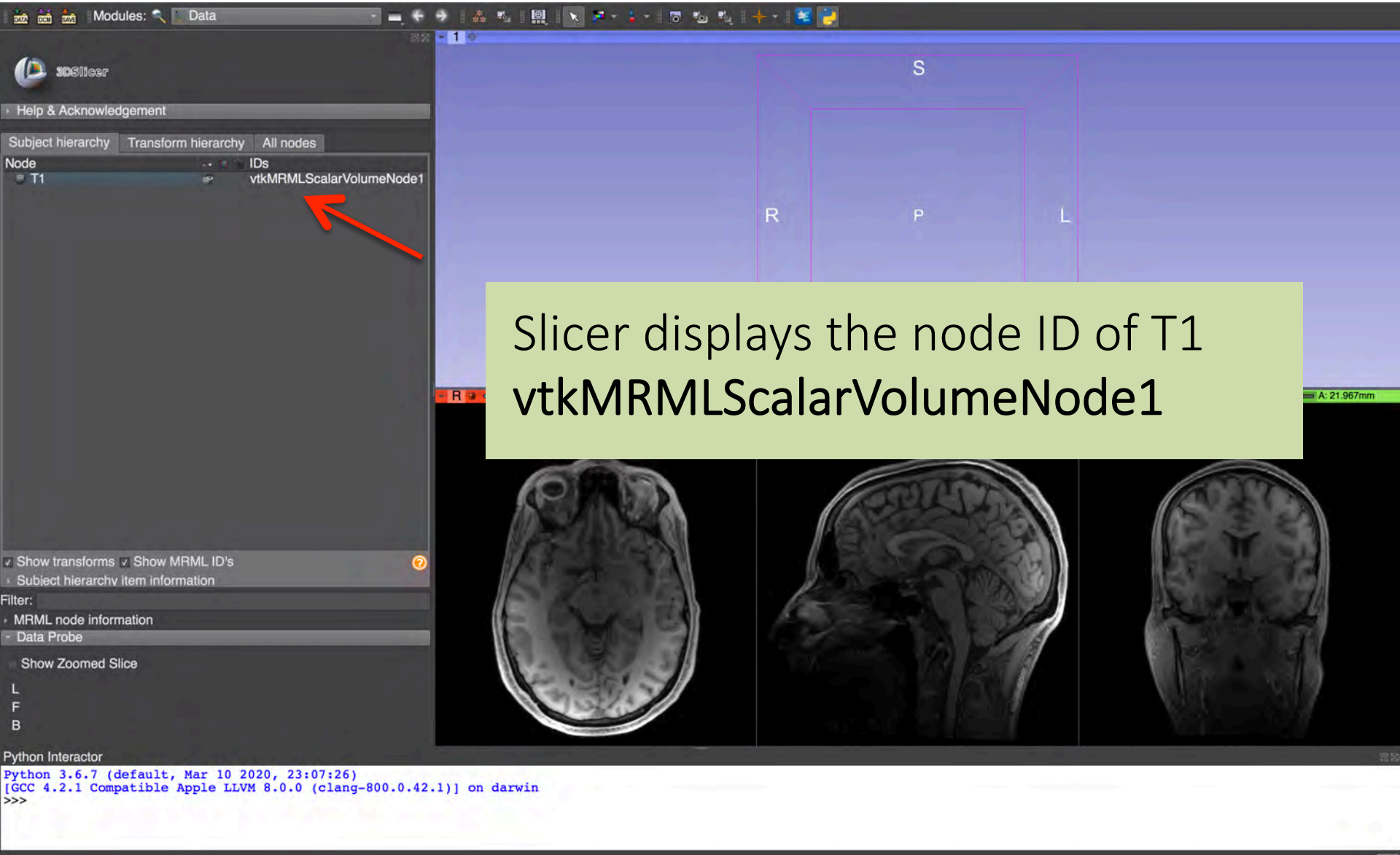
Slicer MRML Nodes: Basic Types

- **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

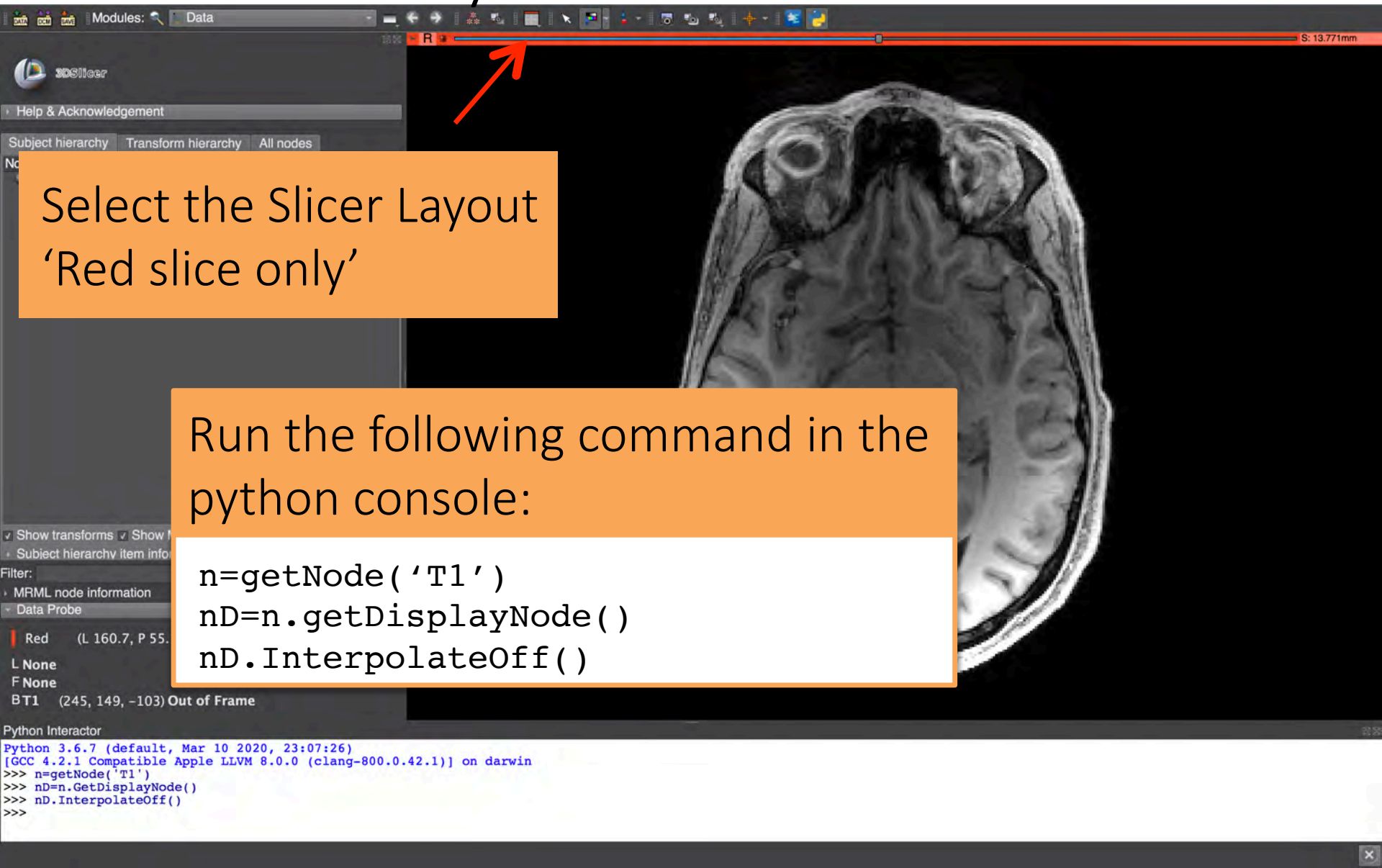


Slicer Data Model



Slicer displays the node ID of T1
vtkMRMLScalarVolumeNode1

Accessing MRML nodes from the Python interactor

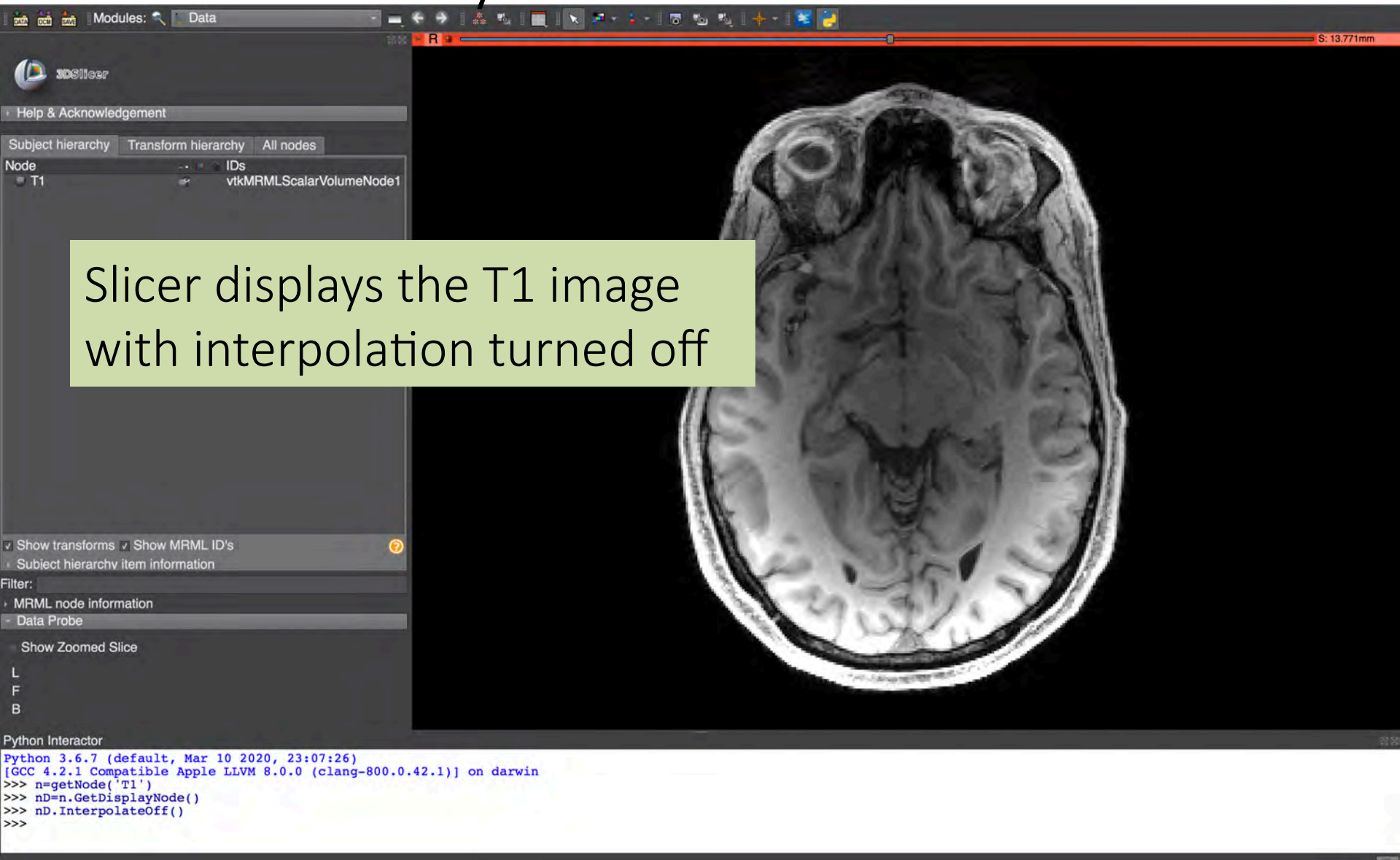


Select the Slicer Layout 'Red slice only'

Run the following command in the python console:

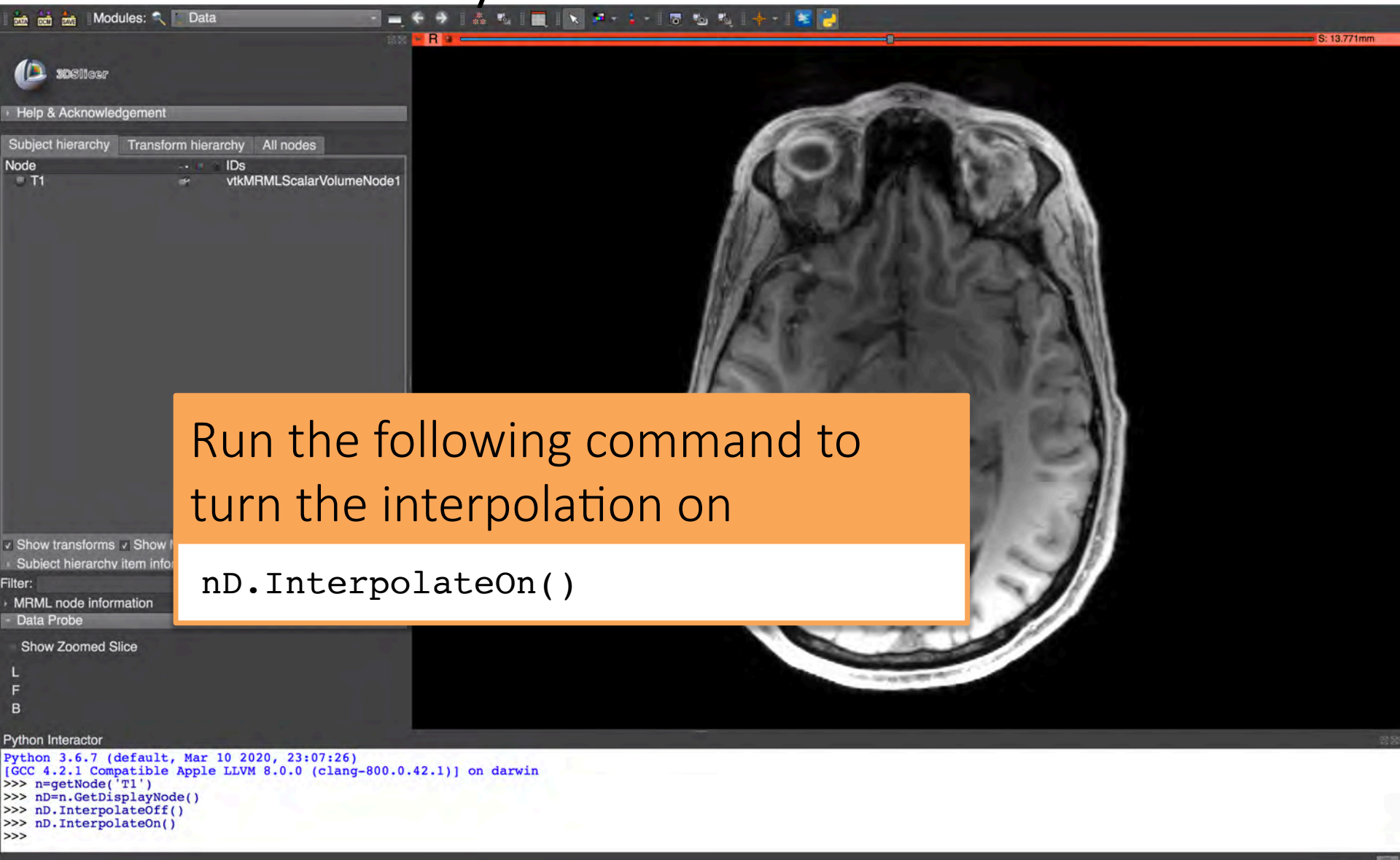
```
n=getNode('T1')
nD=n.getDisplayNode()
nD.InterpolateOff()
```

Accessing MRML nodes from the Python interactor



Slicer displays the T1 image with interpolation turned off

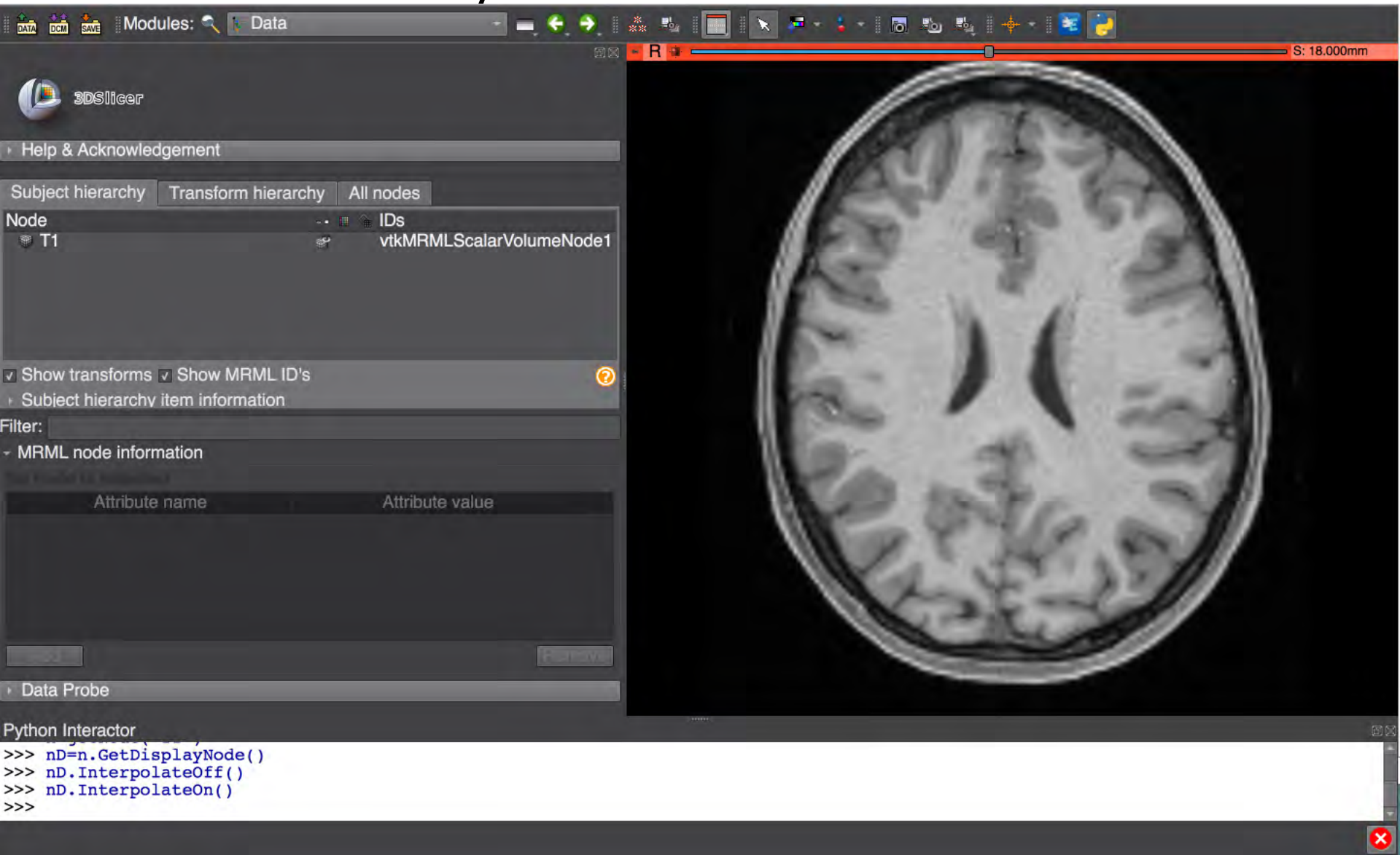
Accessing MRML nodes from the Python interactor



Run the following command to turn the interpolation on

```
nD.InterpolateOn()
```

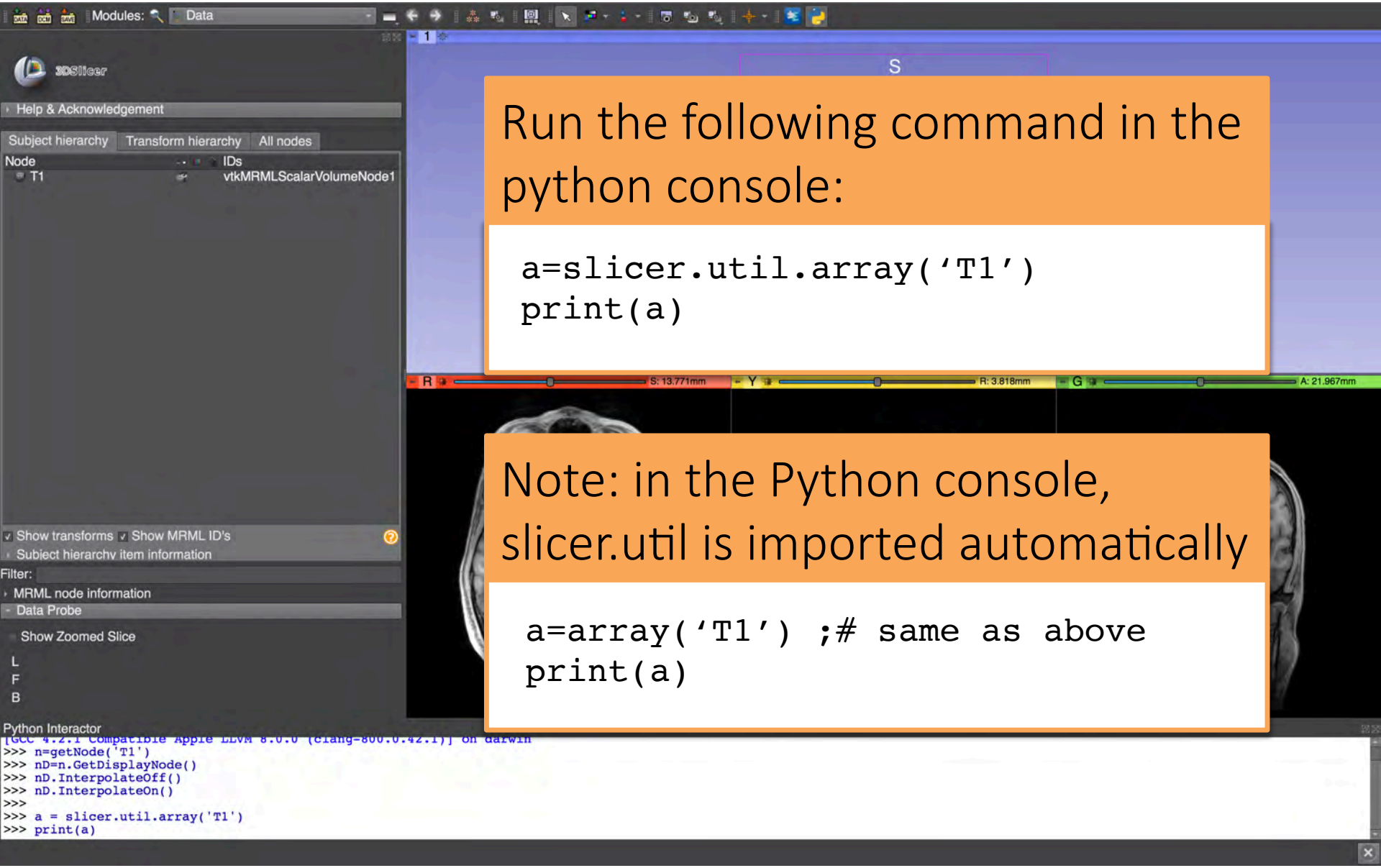
Accessing MRML nodes from the Python interactor



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



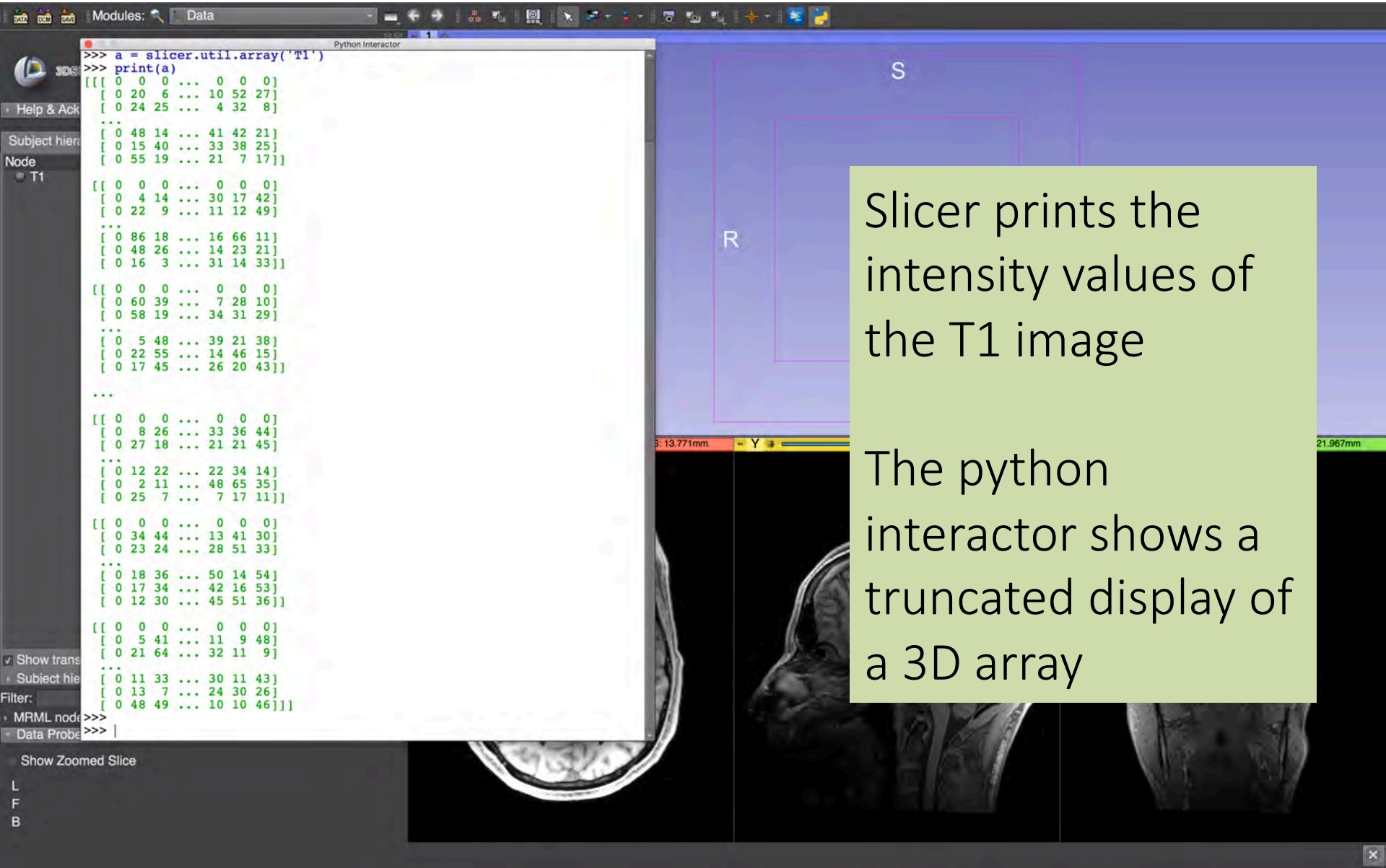
Run the following command in the python console:

```
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)
```

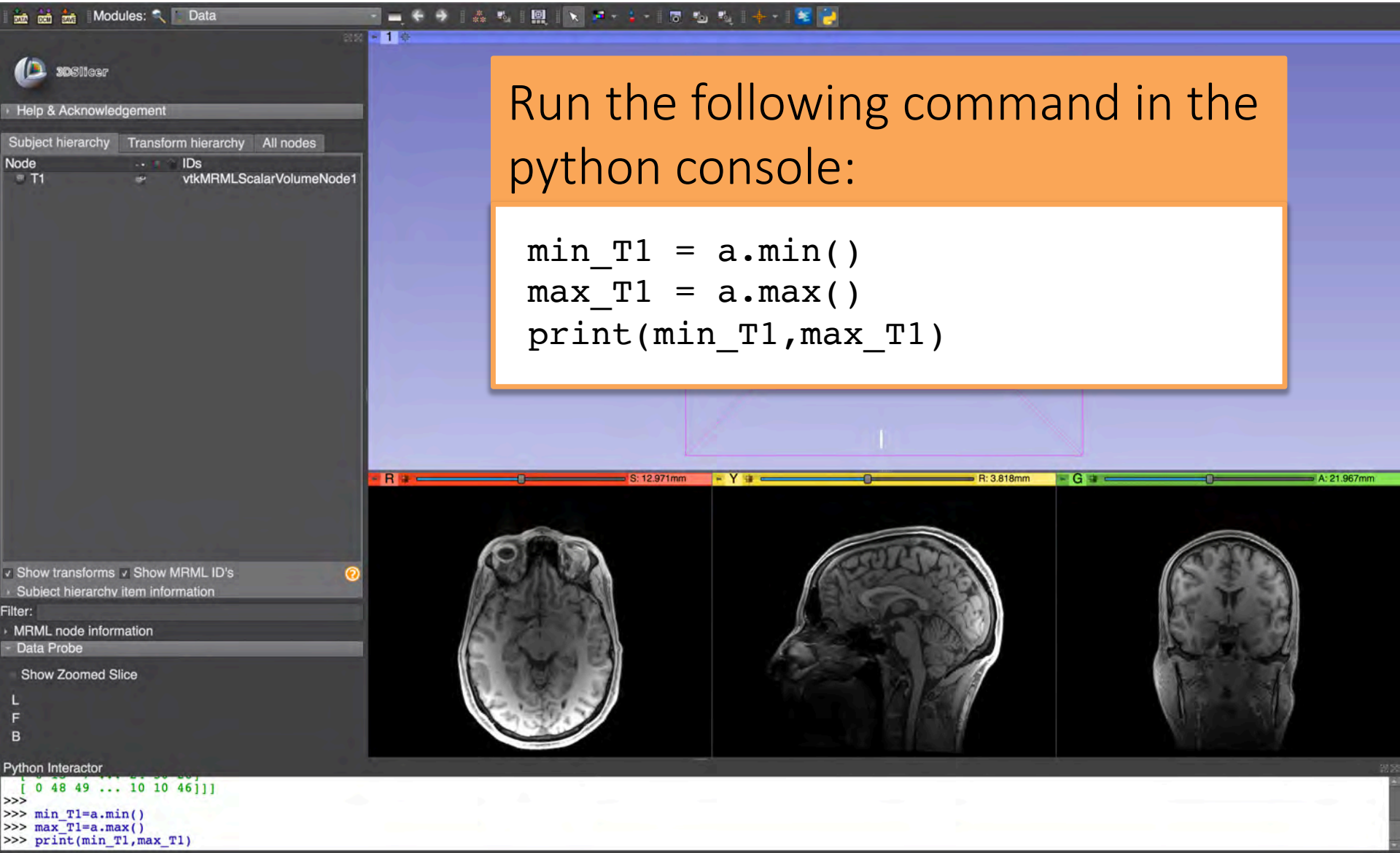
Accessing voxels in a volume



Slicer prints the intensity values of the T1 image

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

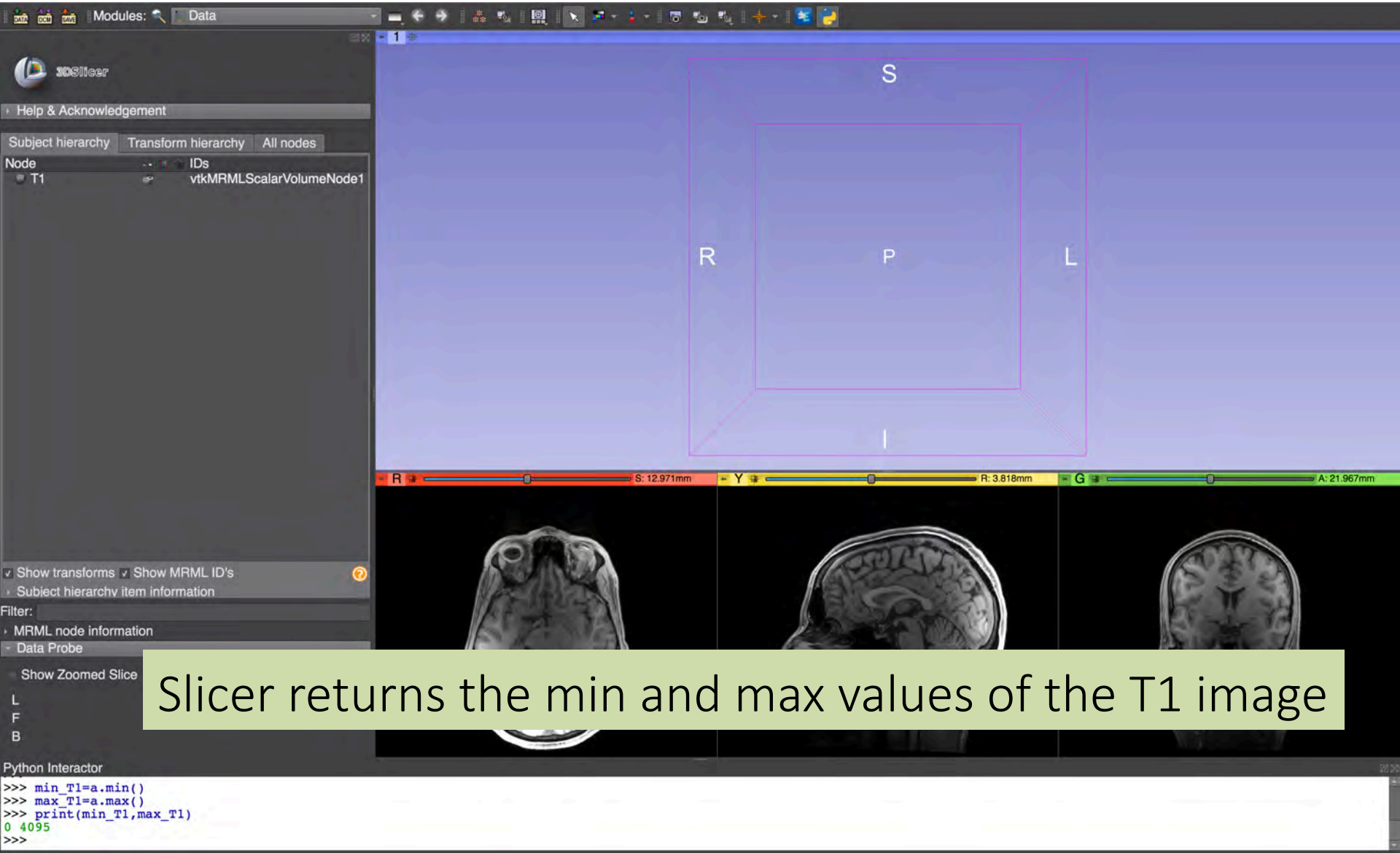
Accessing voxels in a volume



Run the following command in the python console:

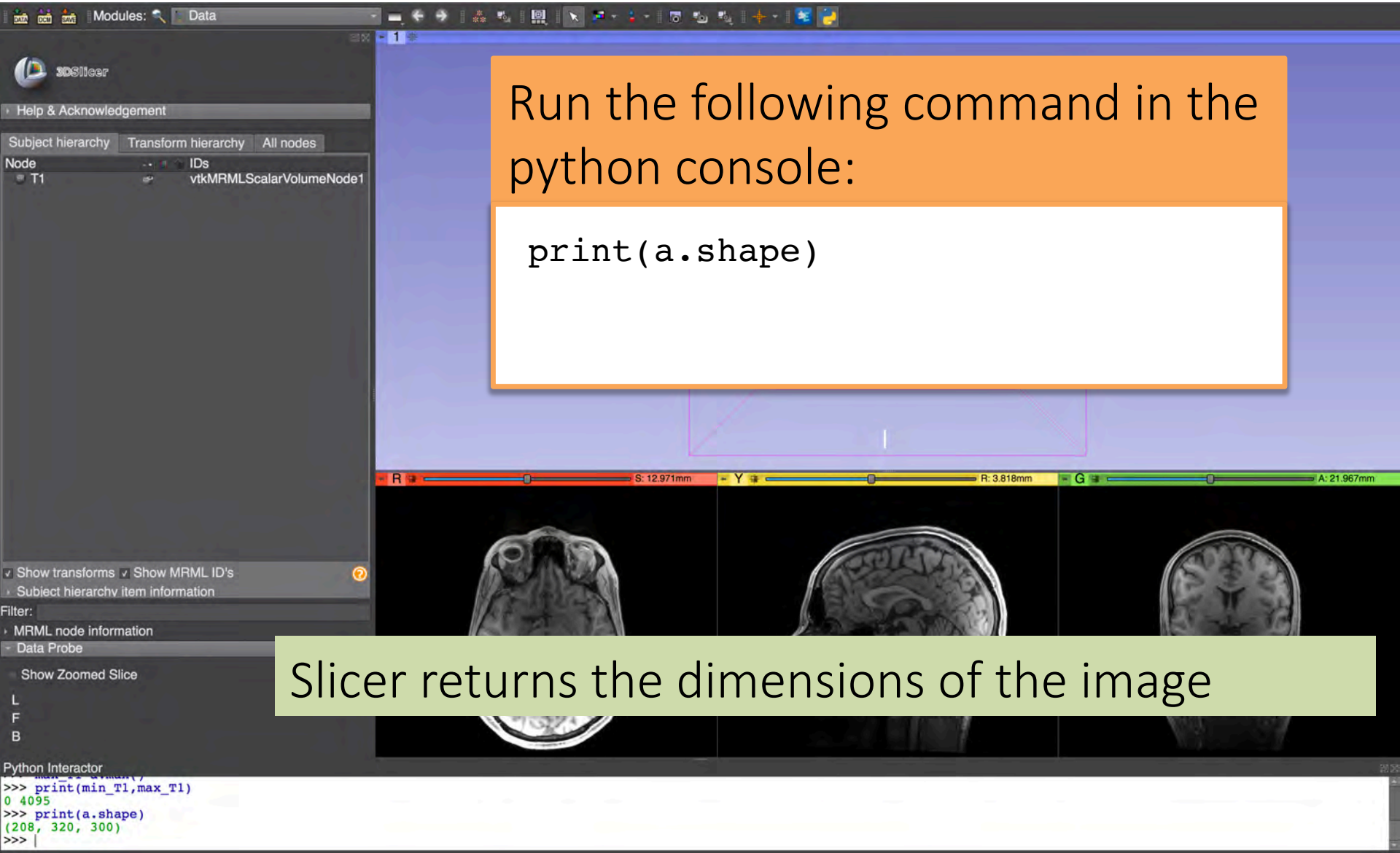
```
min_T1 = a.min()
max_T1 = a.max()
print(min_T1,max_T1)
```

Accessing voxels in a volume



Slicer returns the min and max values of the T1 image

Modifying voxels in a volume

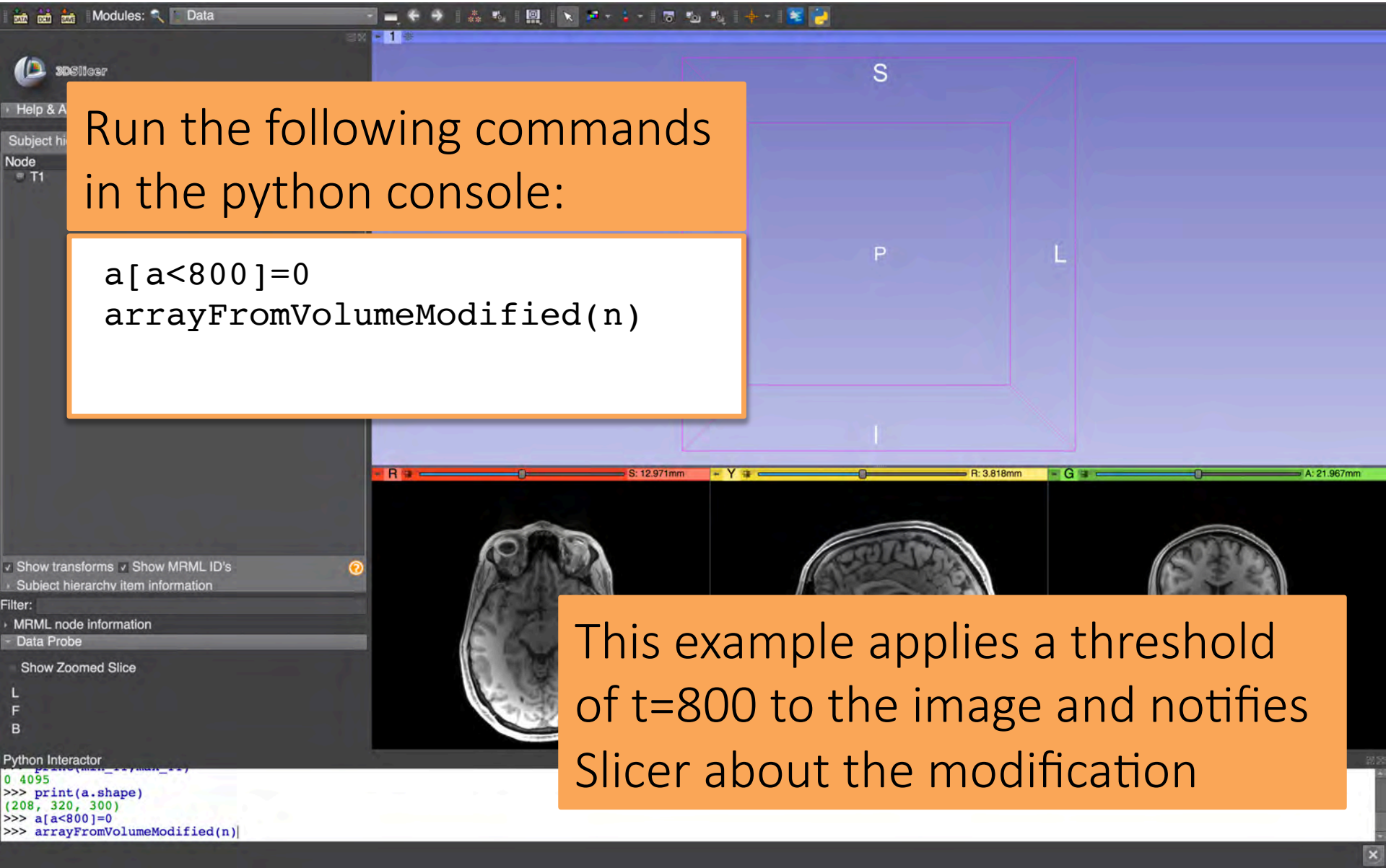


Run the following command in the python console:

```
print(a.shape)
```

Slicer returns the dimensions of the image

Modifying voxels in a volume

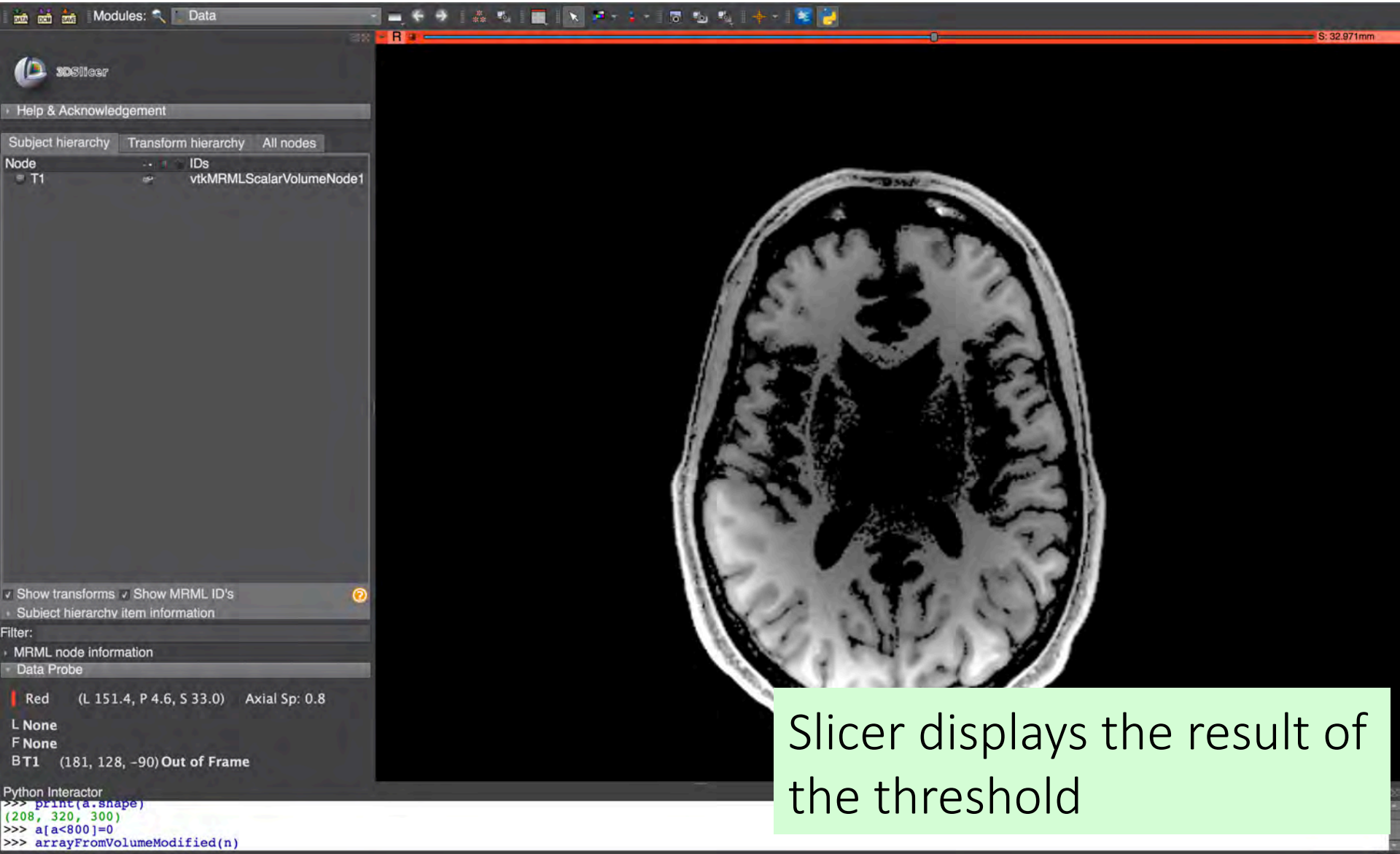


Run the following commands in the python console:

```
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

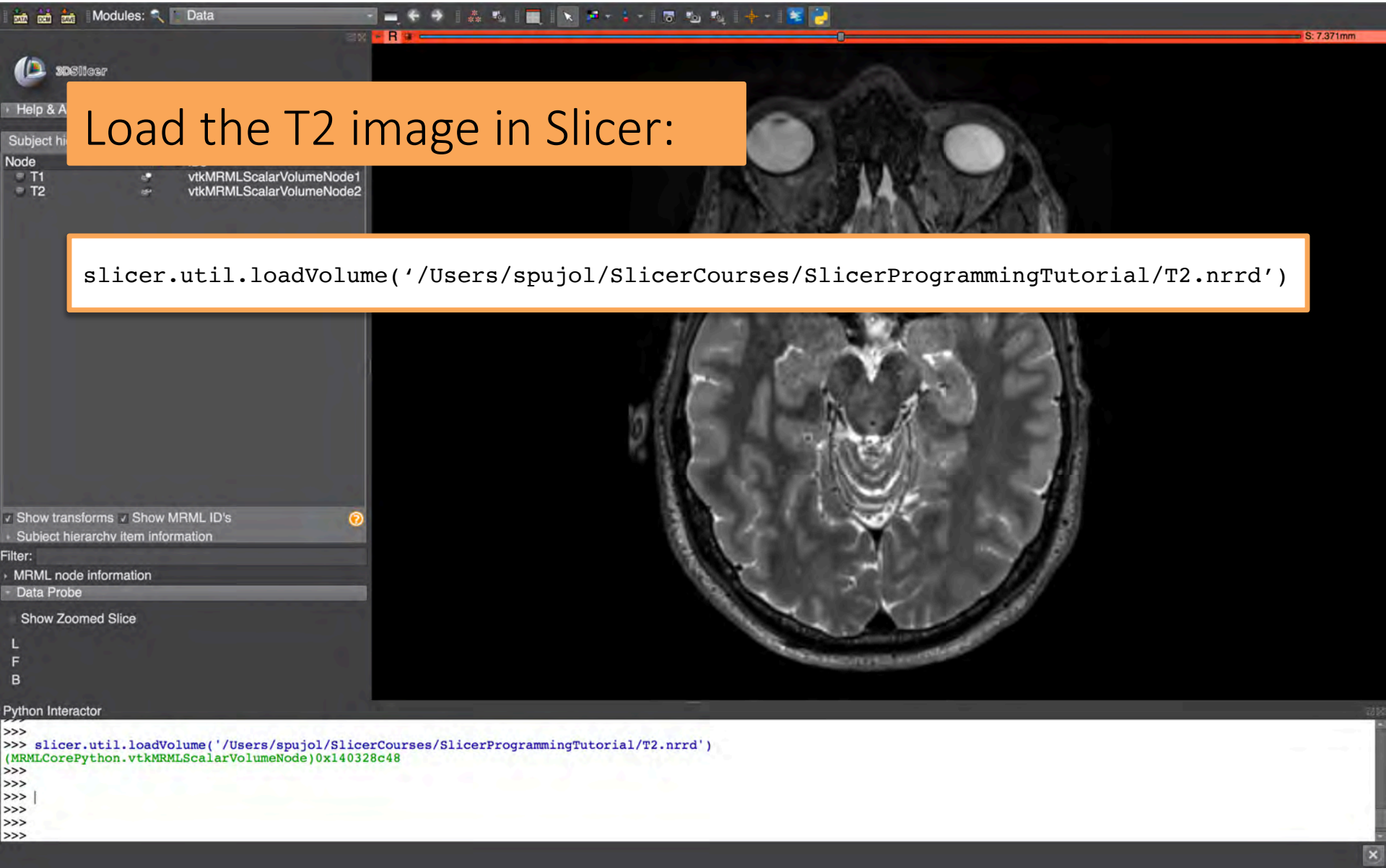


Slicer displays the result of the threshold

Loading the T2 volume

Load the T2 image in Slicer:

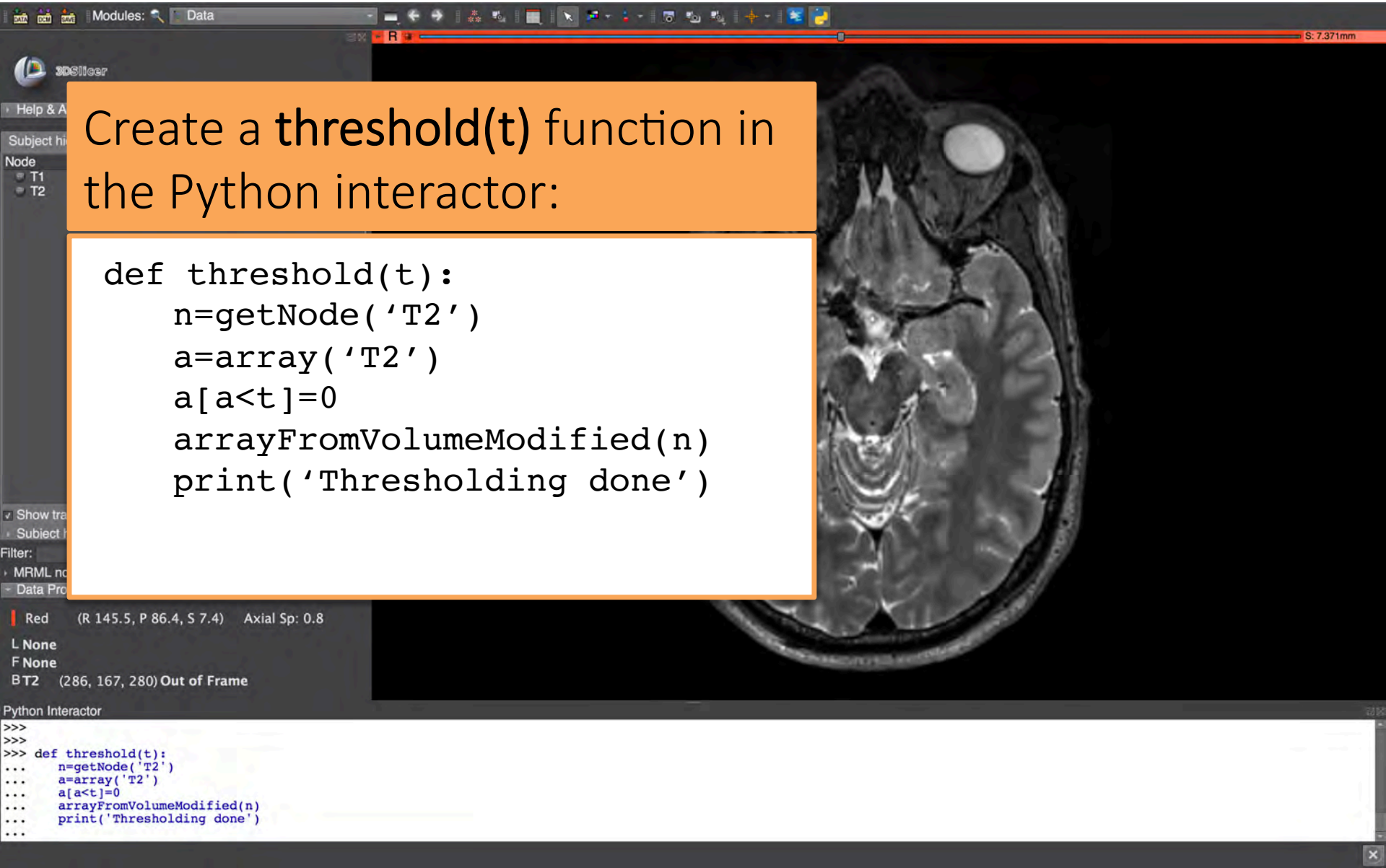
```
slicer.util.loadVolume('/Users/spujol/SlicerCourses/SlicerProgrammingTutorial/T2.nrrd')
```



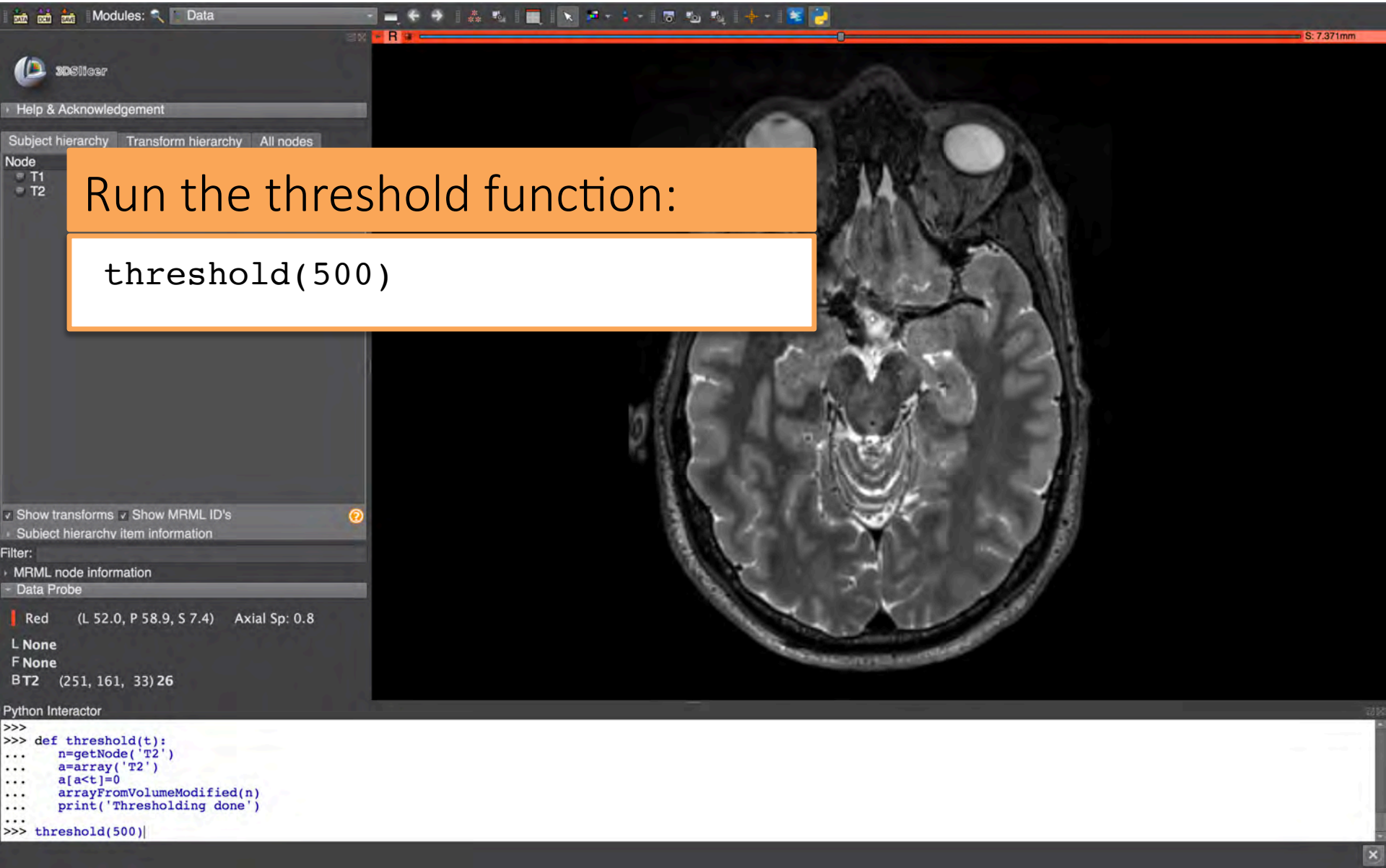
Python function: threshold

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')
```



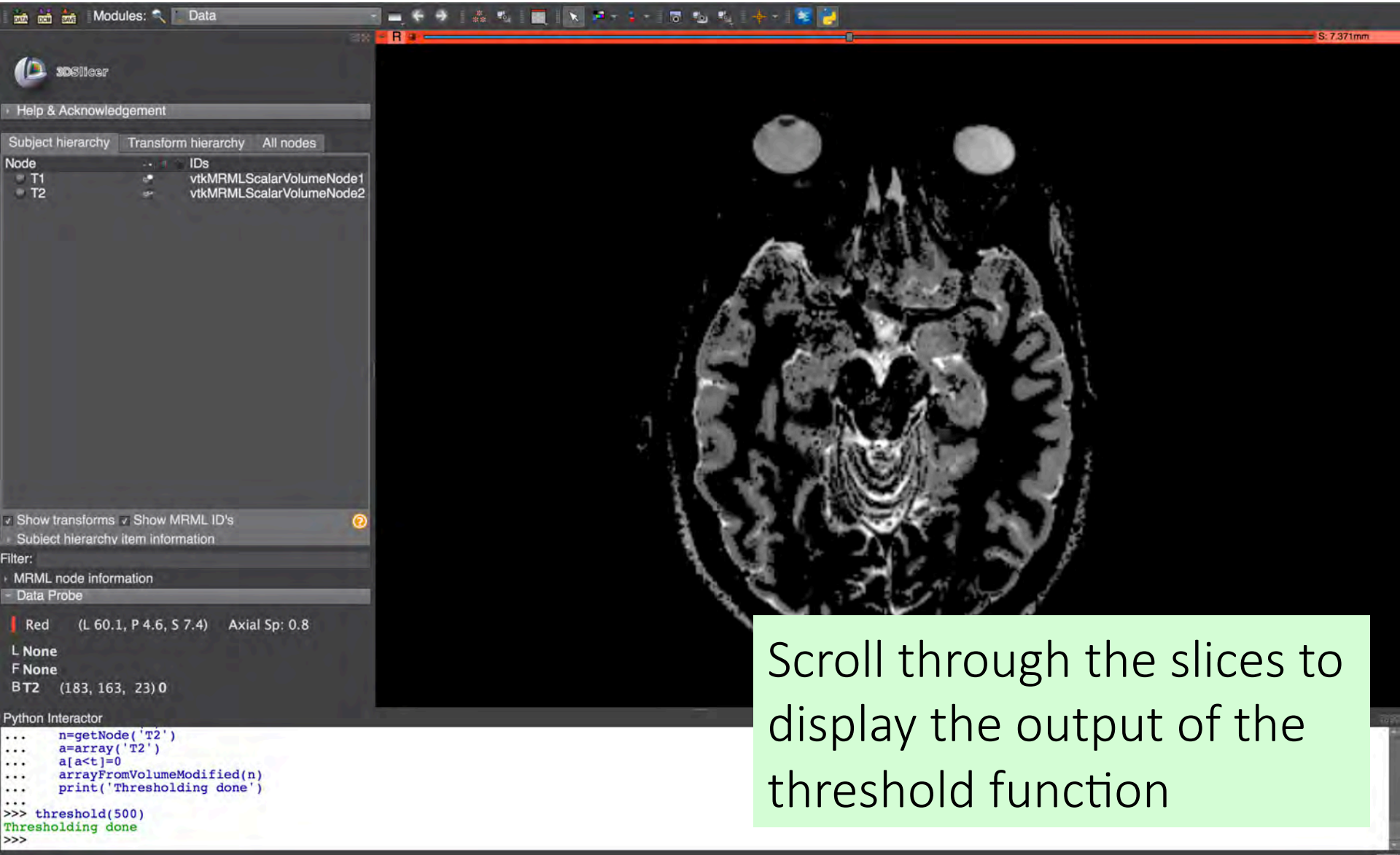
Python function: threshold



Run the threshold function:

```
threshold(500)
```

Python function: threshold



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

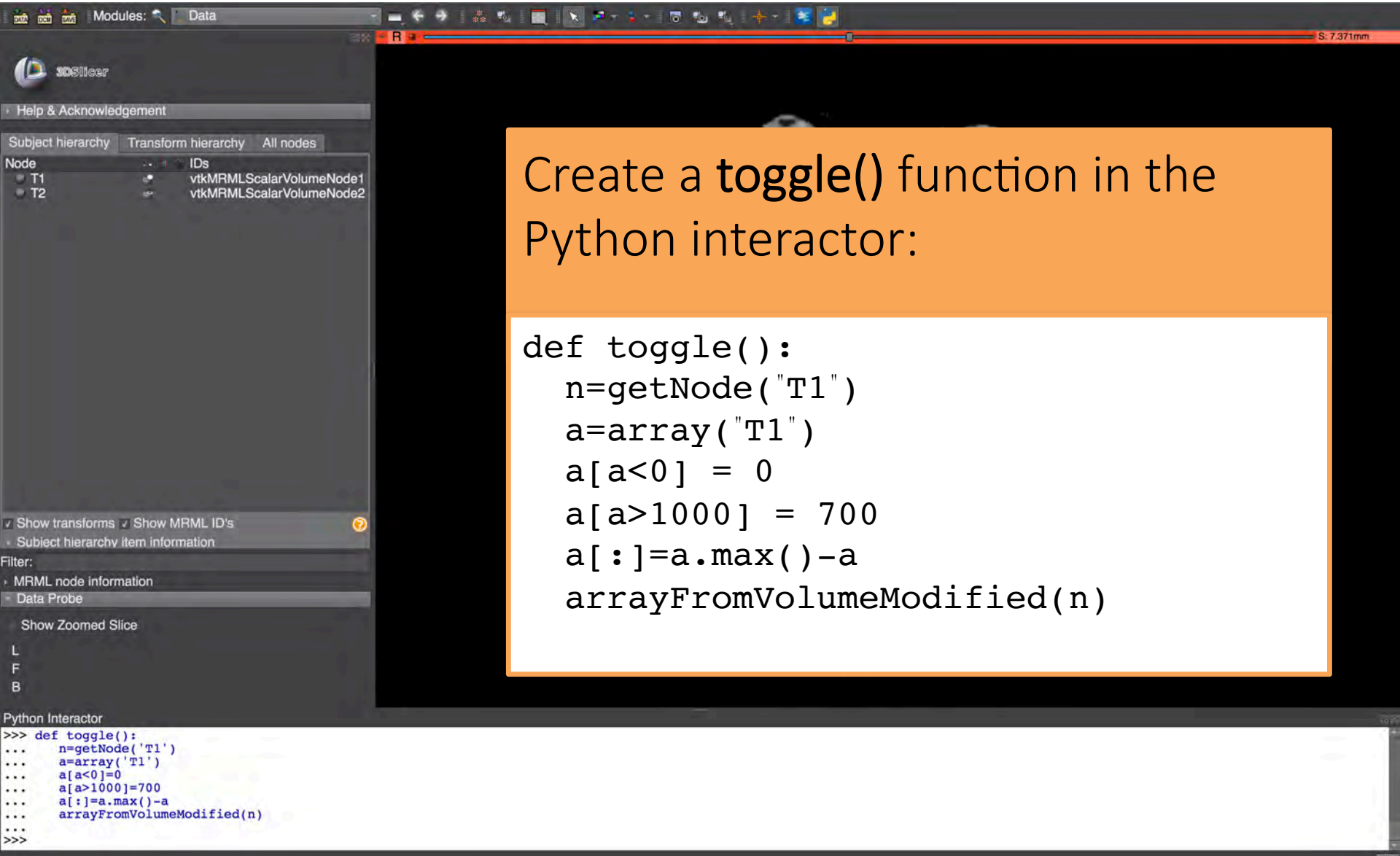
Qt & PythonQt

- **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

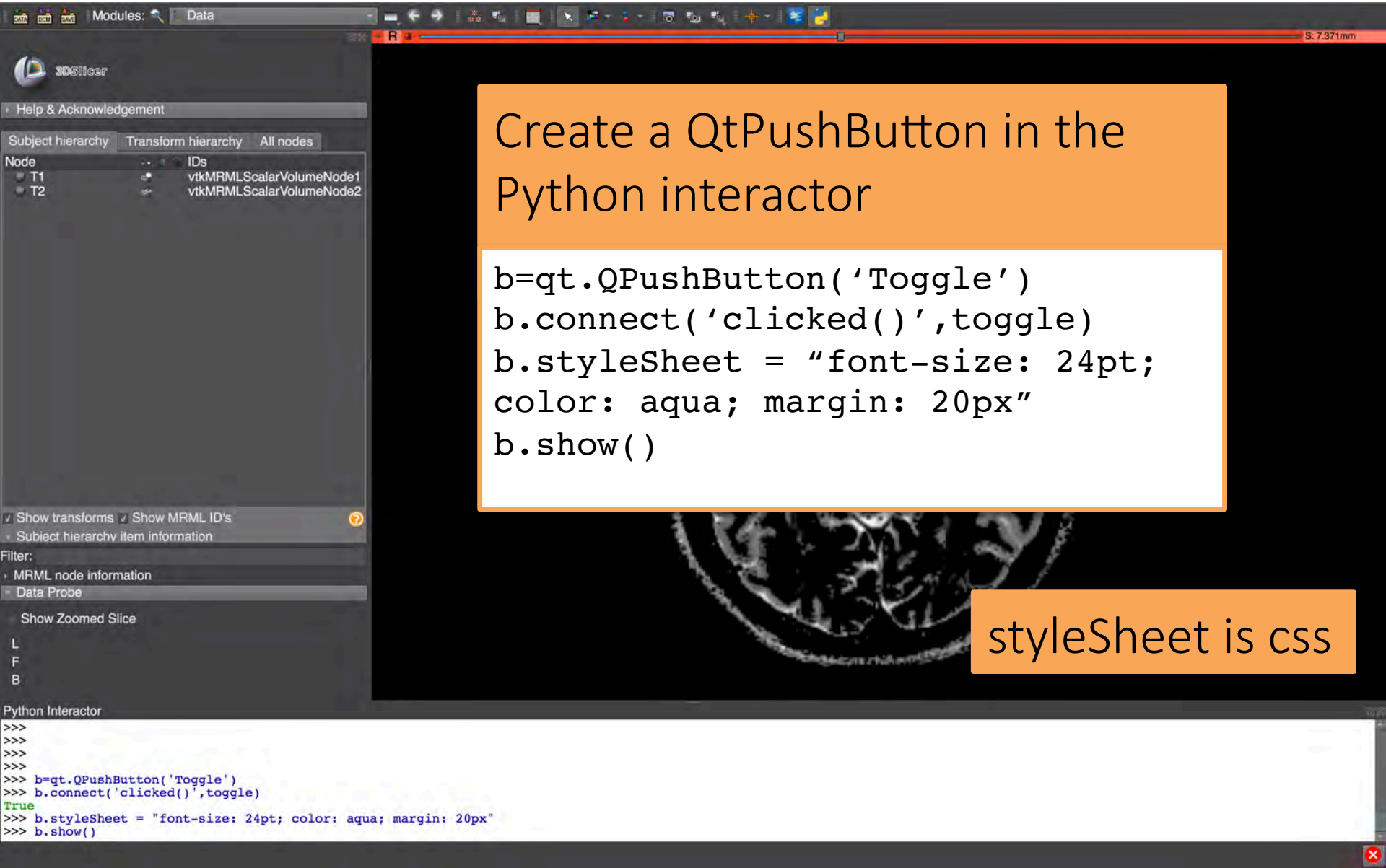
Python function: toggle



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

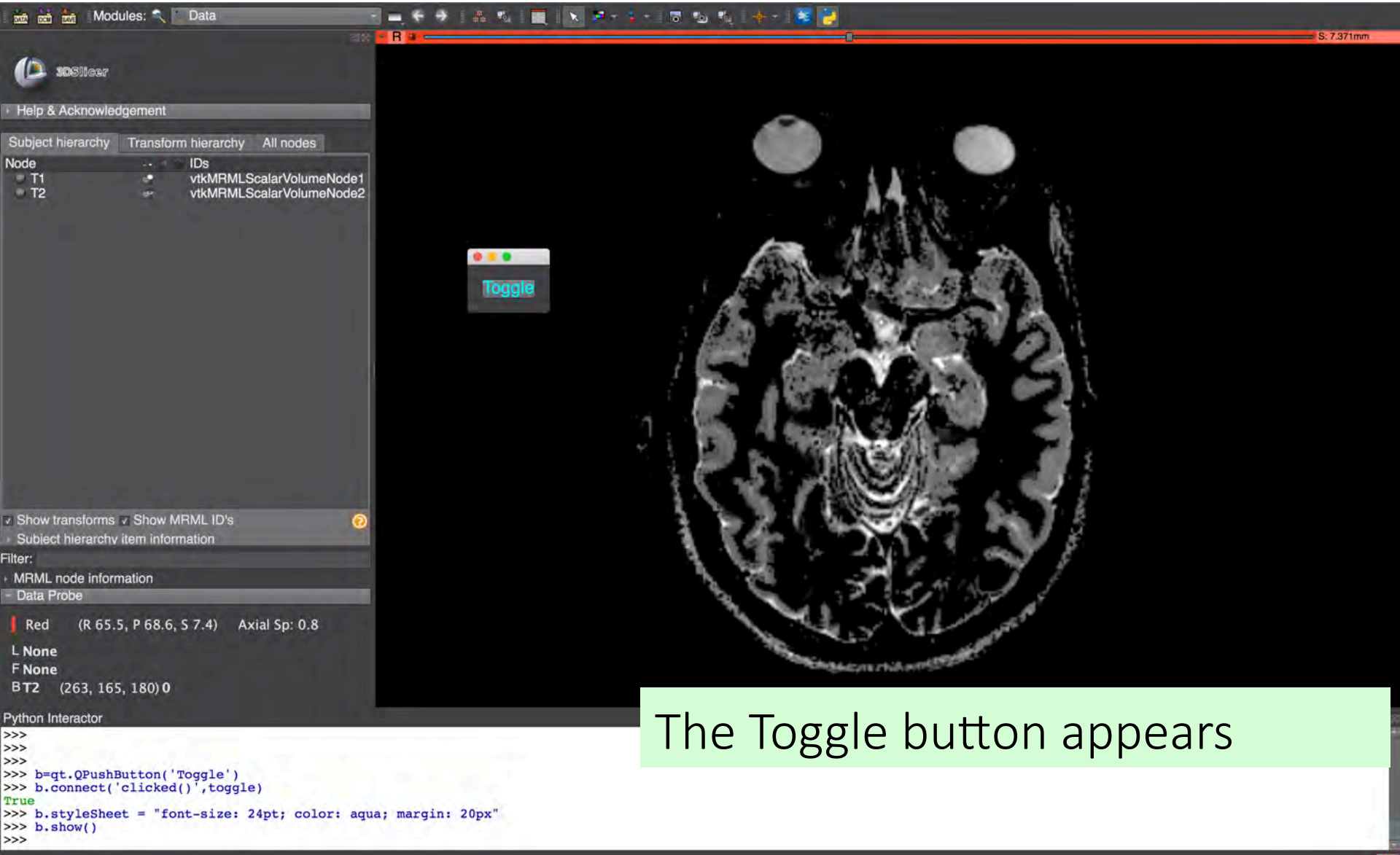


Create a QtPushButton in the Python interactor

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

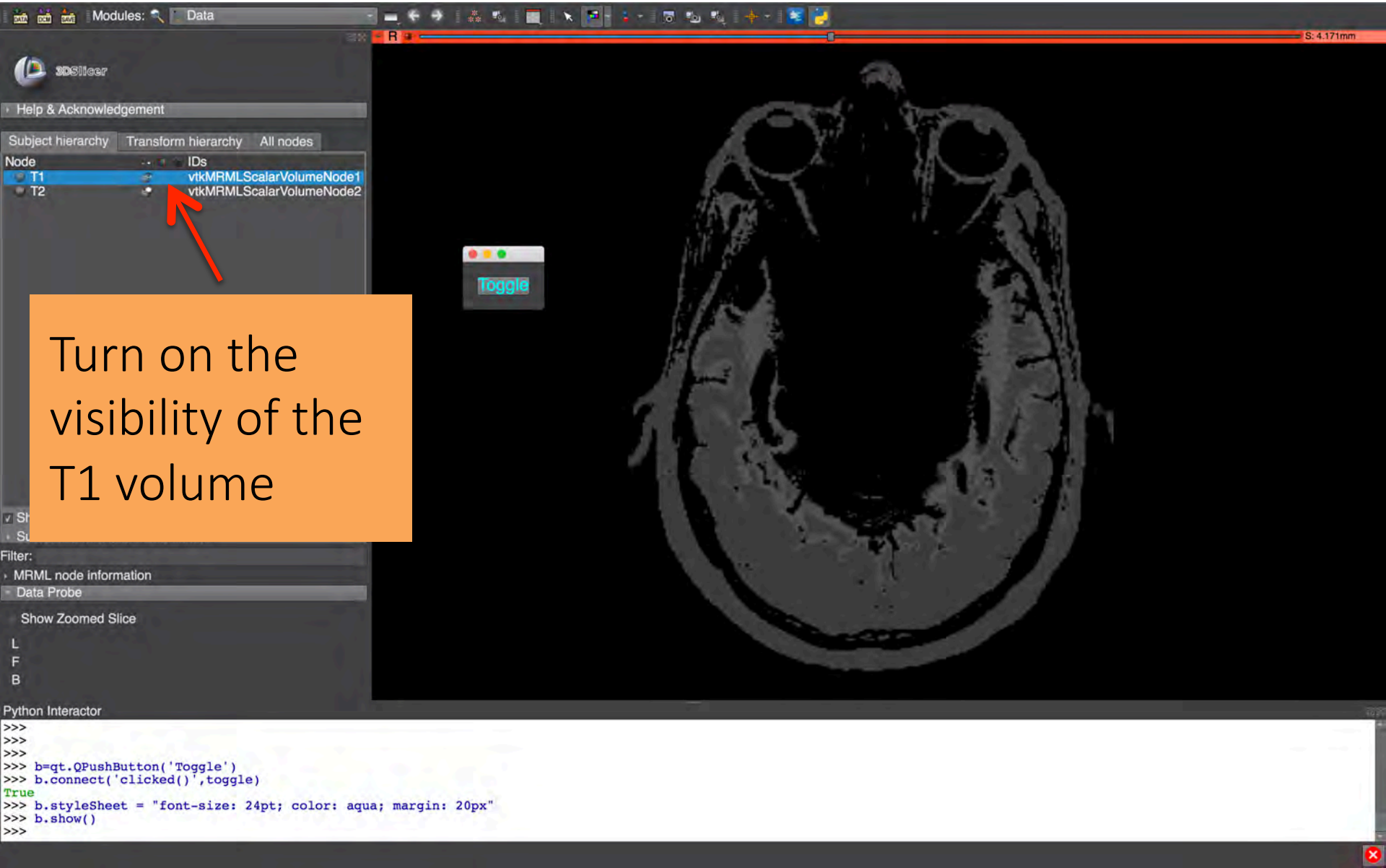
styleSheet is css

Creating a Qt Push Button



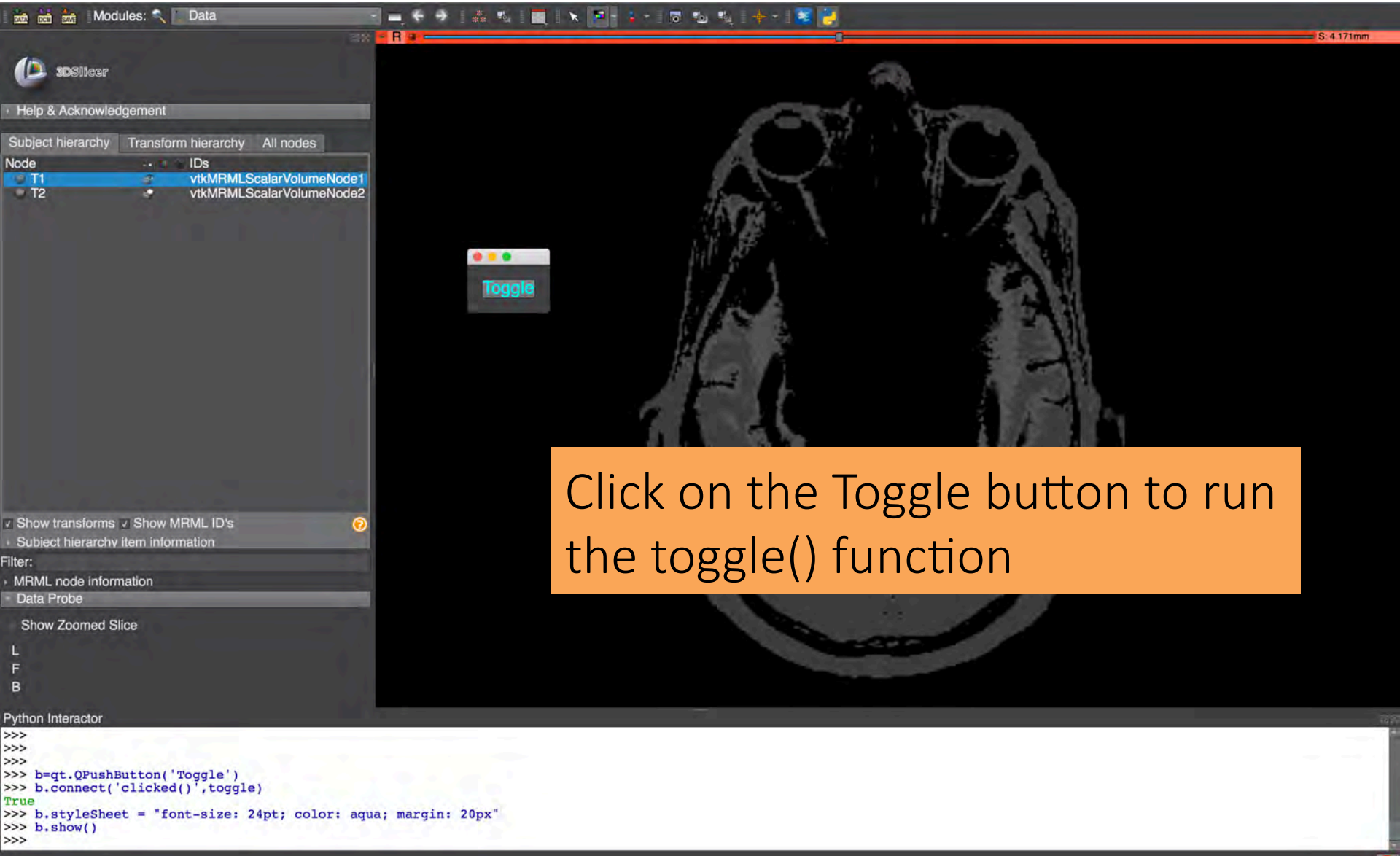
The Toggle button appears

Creating a Qt Push Button



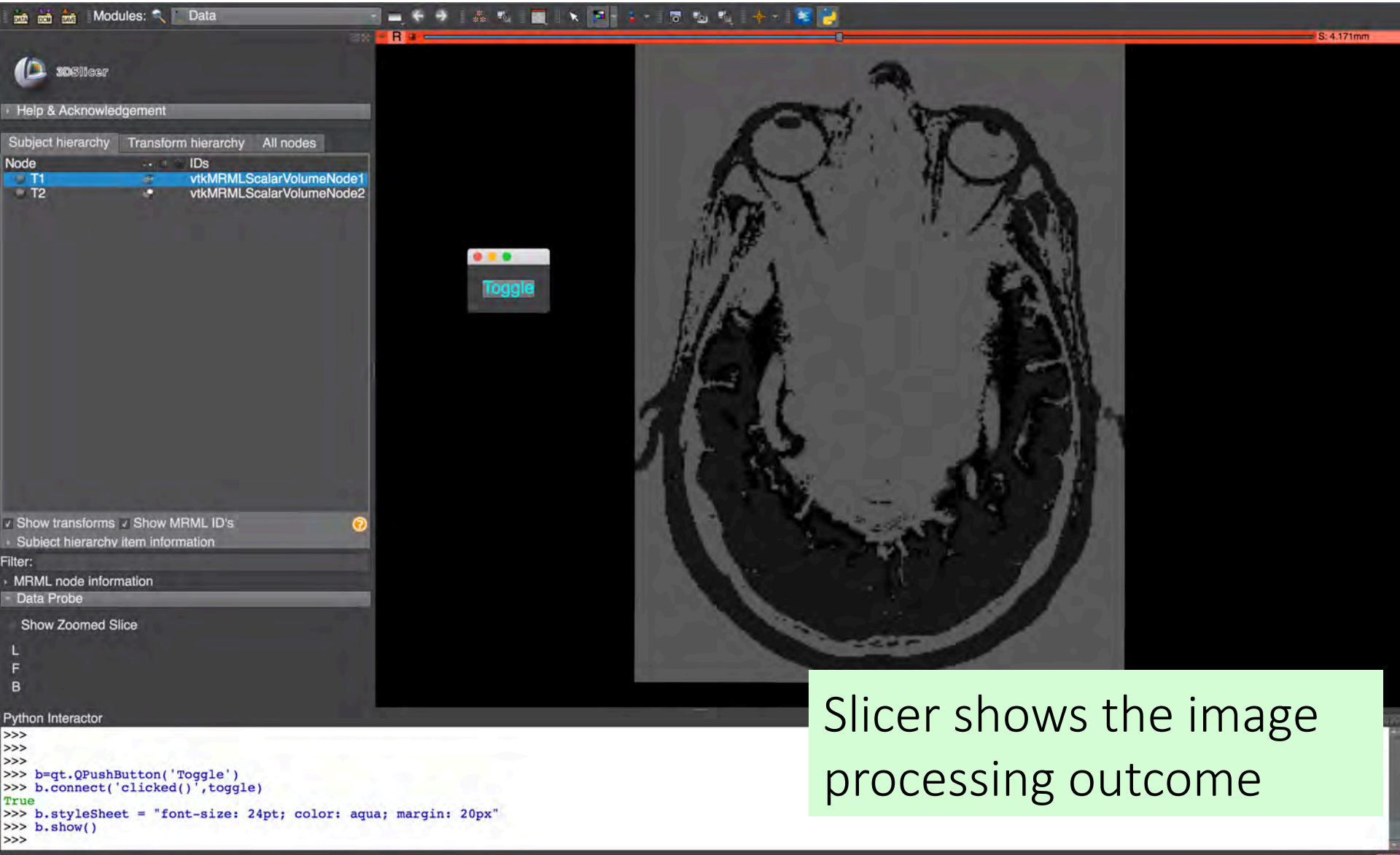
Turn on the visibility of the T1 volume

Creating a Qt Push Button



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

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 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

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)