

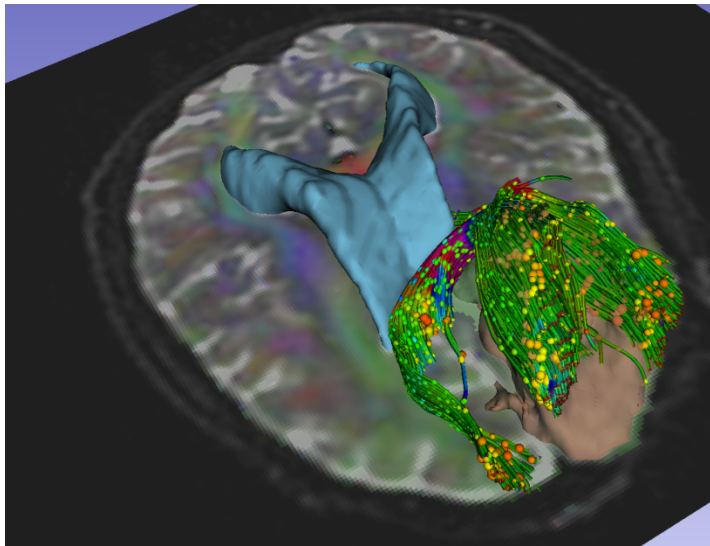
An introduction to programming in Slicer4

Sonia Pujol, Ph.D.

Surgical Planning Laboratory, Harvard Medical School
Director of Training, National Alliance for Medical Image Computing

Madrid-MIT M+Vision Consortium
May 21-22, 2012 Madrid, Spain

3D Slicer version 4 (Slicer4.1)



- An **end-user application** for image analysis
- An **open-source environment** for software development
- A software platform that is both **easy to use** for clinical researchers and **easy to extend** for programmers

Slicer for Translational Research



What does a developer need ?

- Easy to deploy
- Extensible and reconfigurable
- Rich utility libraries
- Stable base

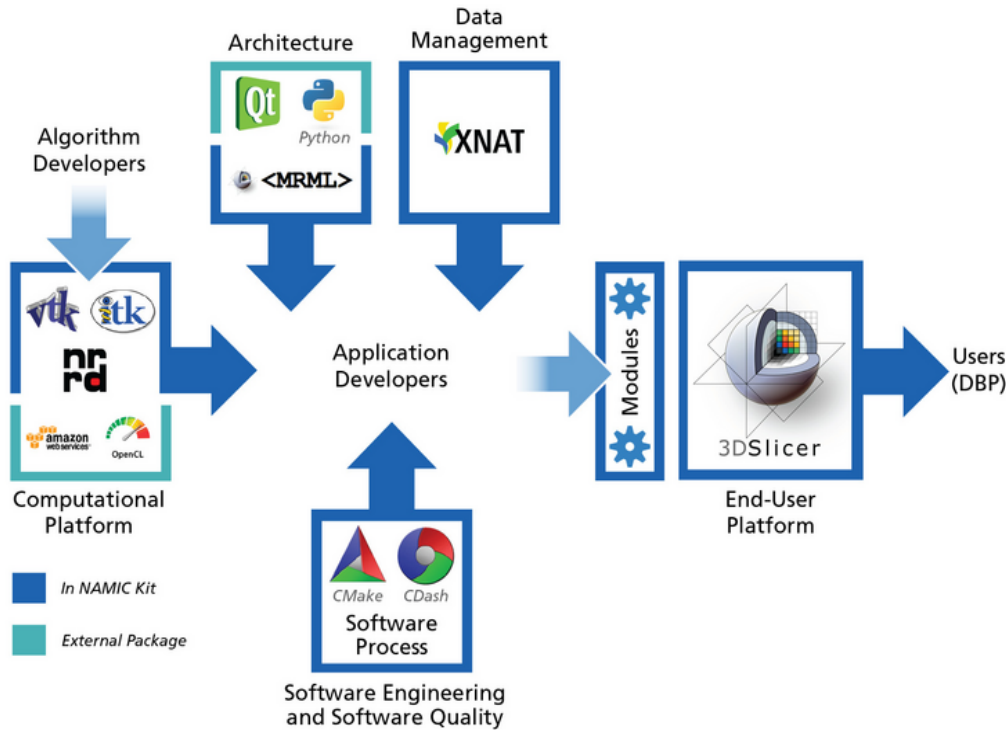
What does a user expect ?

- Easy to install and upgrade
- “Standard” Clinical Behavior
- Advanced Functionality
- Consistent Interface

The NA-MIC Kit



The NA-MIC kit



Application developers create tools within an architectural framework in conjunction with data management facilities and under the control of quality software process

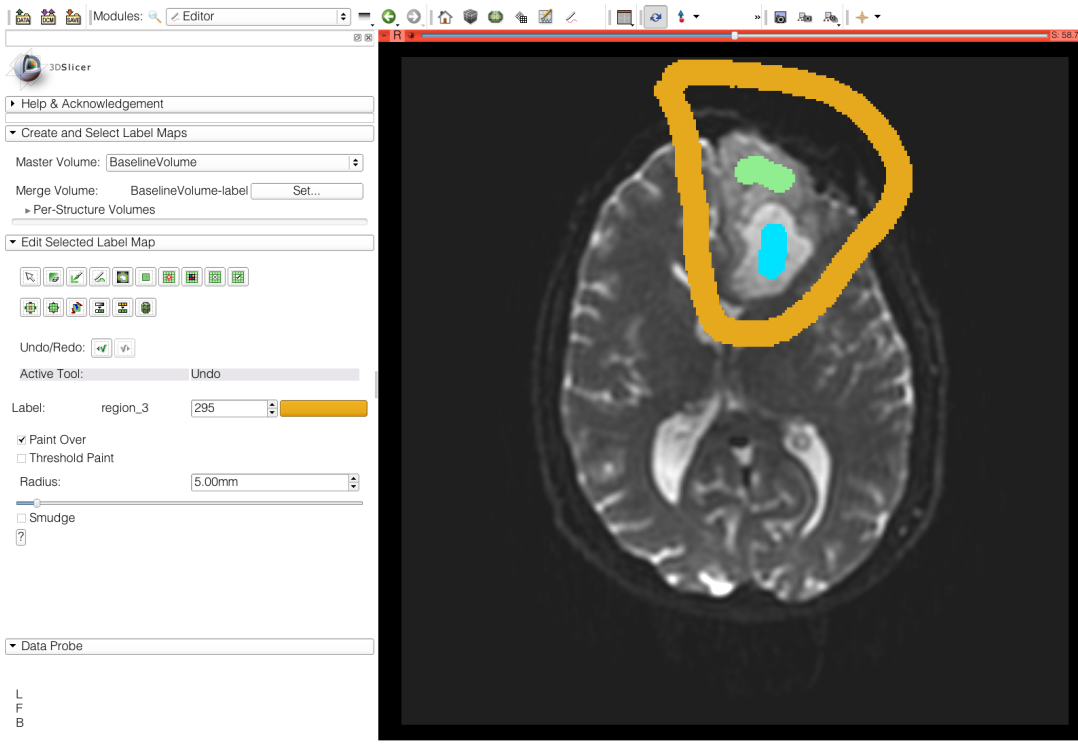
What's inside Slicer ?

- Slicer core: Slicer GUI, I/O, visualization and developer interfaces
- Slicer modules: internal plugins that depend on the slicer core
- Slicer extensions: external plugins installed on demand by the user

Slicer Modules

Slicer supports three types of modules:

- Scripted Modules (Python)

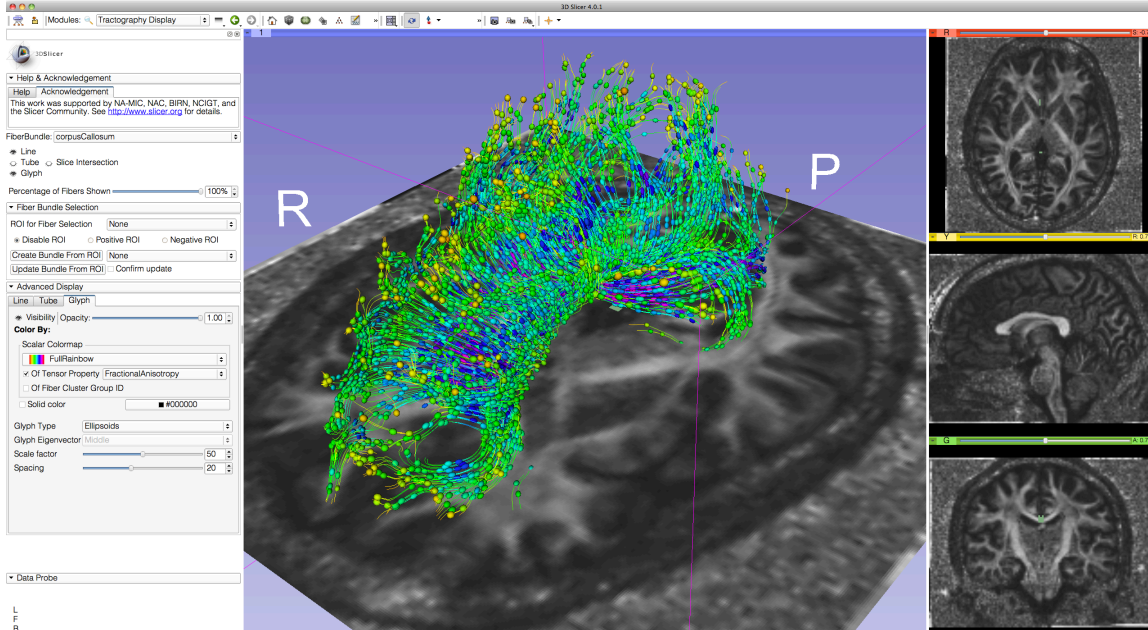


e.g. Editor

Slicer Modules

Slicer supports three types of modules:

- Scripted Modules (Python)
- Command Line Interface (ITK)

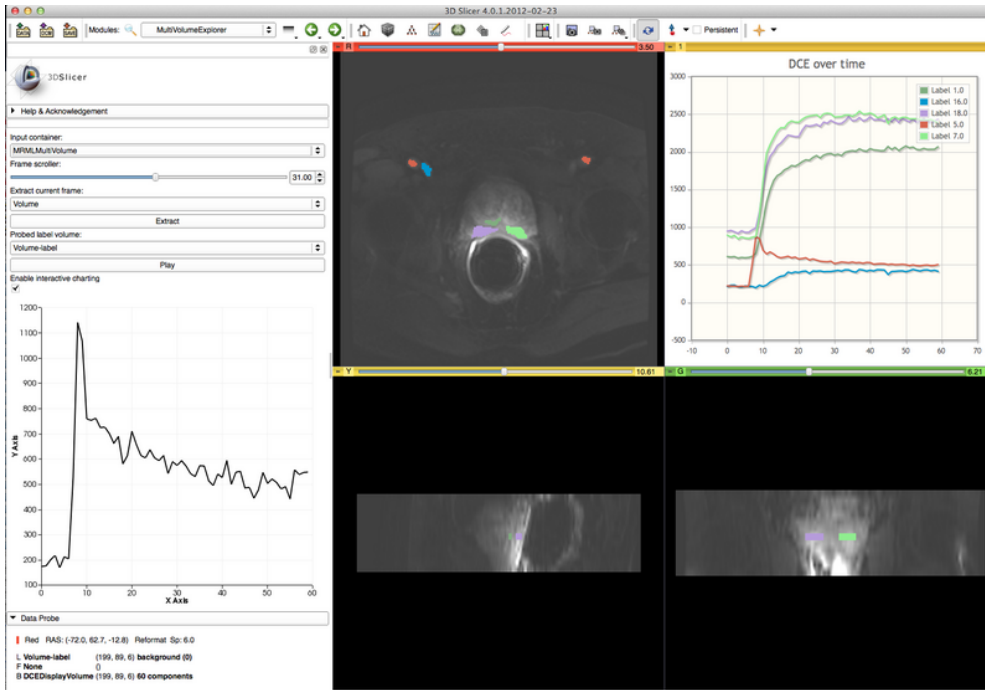


e.g. Tractography
Labelmap Seeding

Slicer Modules

Slicer supports three types of modules:

- Scripted Modules (Python)
- Command Line Interface (ITK)
- Loadable Modules (C++)

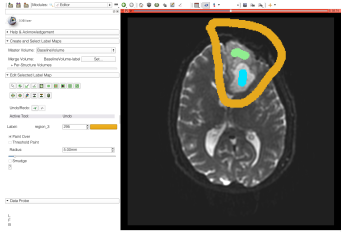


e.g. MultipleVolume Explorer

image courtesy R. Kikinis

Slicer Modules

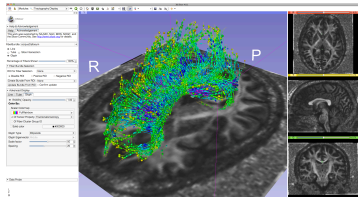
Variety of options for the developers:



Scripted module: TCL or Python scripts

simple, no compilation needed

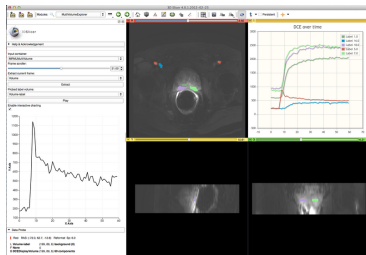
limited access to Slicer internals



Command-line module: .exe file

simple, executable without Slicer

no access to Slicer internals, Slicer compilation needed



Loadable (interactive) module: .dll

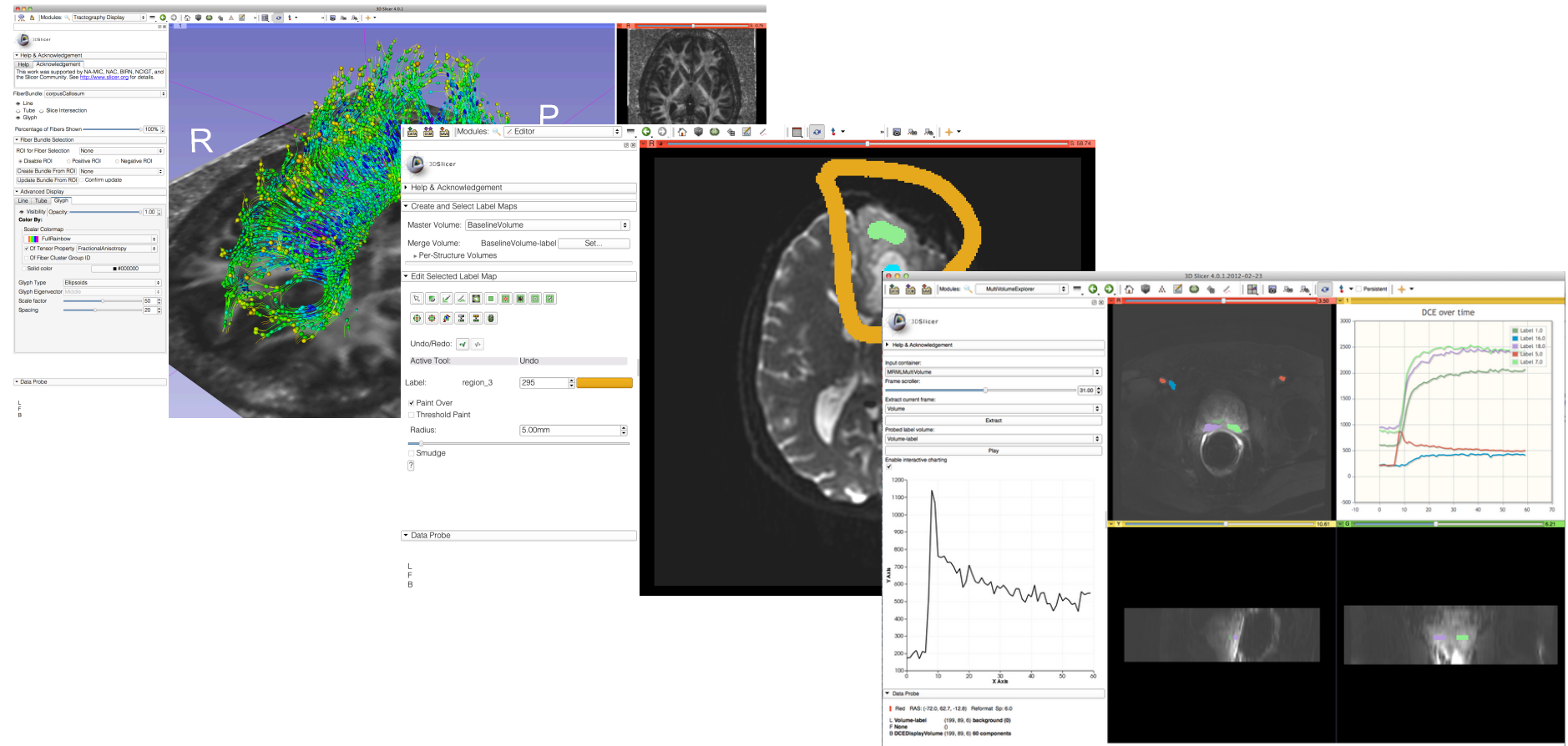
full access to Slicer internals

Slicer compilation needed, requires Slicer core knowledge

Courtesy R. Kikinis

Slicer Modules

Three options for the developers



→ Consistent look and feel for the user

Slicer is Extensible

- The Slicer Extension Catalog offers the possibility to the user to download and install additional Slicer modules

The screenshot shows the Slicer Extensions Manager interface. At the top, there are tabs for "Manage Extensions (1)" and "Install Extensions". The main content area displays the details for the "SkullStripper" extension by Xiaodong Tao (GE). It includes a thumbnail image of a brain scan, a description, screenshots of the software interface, a rating system (5 stars, 1 vote), and a "UNINSTALL" button. The interface also shows a "Restart requested" message and "Restart" and "Close" buttons at the bottom.

SkullStripper
Xiaodong Tao (GE)

Slicer Extensions > Segmentation

Description
Skull stripping for structural MR images of the brain, tested on T1 and T2 contrast.

Screenshots

UNINSTALL

Created April 13, 2012
375.7 KB
9 downloads

★★★★★ 5 (1 votes)

5 star 1
4 star 0
3 star 0
2 star 0
1 star 0

Click to rate ★★★★★

Tutorials
There are no tutorials for this extension.

User Comments

Restart requested

Restart Close

Slicer Extensions Categories

Category 1: Slicer license, open-source, maintained

Category 2: open-source, contact exists

Category 3: work in progress, closed-source...

Programming in Slicer4: The Hello Python tutorial

Sonia Pujol, Ph.D.
Surgical Planning Laboratory,
Harvard Medical School

Steve Pieper, Ph.D.
Isomics Inc.



Paul Cézanne, Moulin sur la Coulevre à Pontoise, 1881, Staatliche Museen zu Berlin, Nationalgalerie

Tutorial Goal

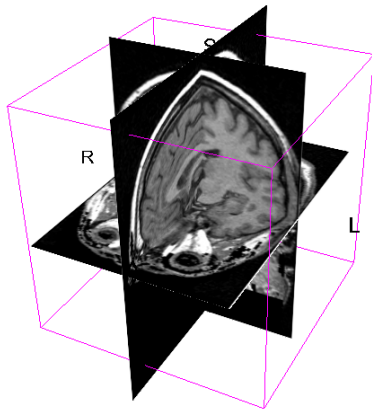
- This tutorial guides you through the steps of programming a HelloPython scripted module for running a Laplacian filtering and sharpening.
- For additional details and pointers, visit the Slicer Documentation page
<http://wiki.slicer.org/slicerWiki/index.php/Documentation/4.0>

Prerequisites

- This course supposes that you have taken the tutorial: “Slicer4 Data Loading and Visualization”- Sonia Pujol Ph.D.
- The tutorial is available on the Slicer4 101 compendium:
<http://www.slicer.org/slicerWiki/index.php/Training/4.0>
- Programming experience is required, and some familiarity with Python is essential.

Course Material

Slicer4.1 version available at www.slicer.org



spgr.nhdr spgr.raw.gz
(124 SPGR images)



Unzip the HelloPython.zip archive

HelloPython.py
HelloLaplace.py
HelloSharpen.py

Slicer4 Highlights: Python

The Python console of Slicer4 gives access to

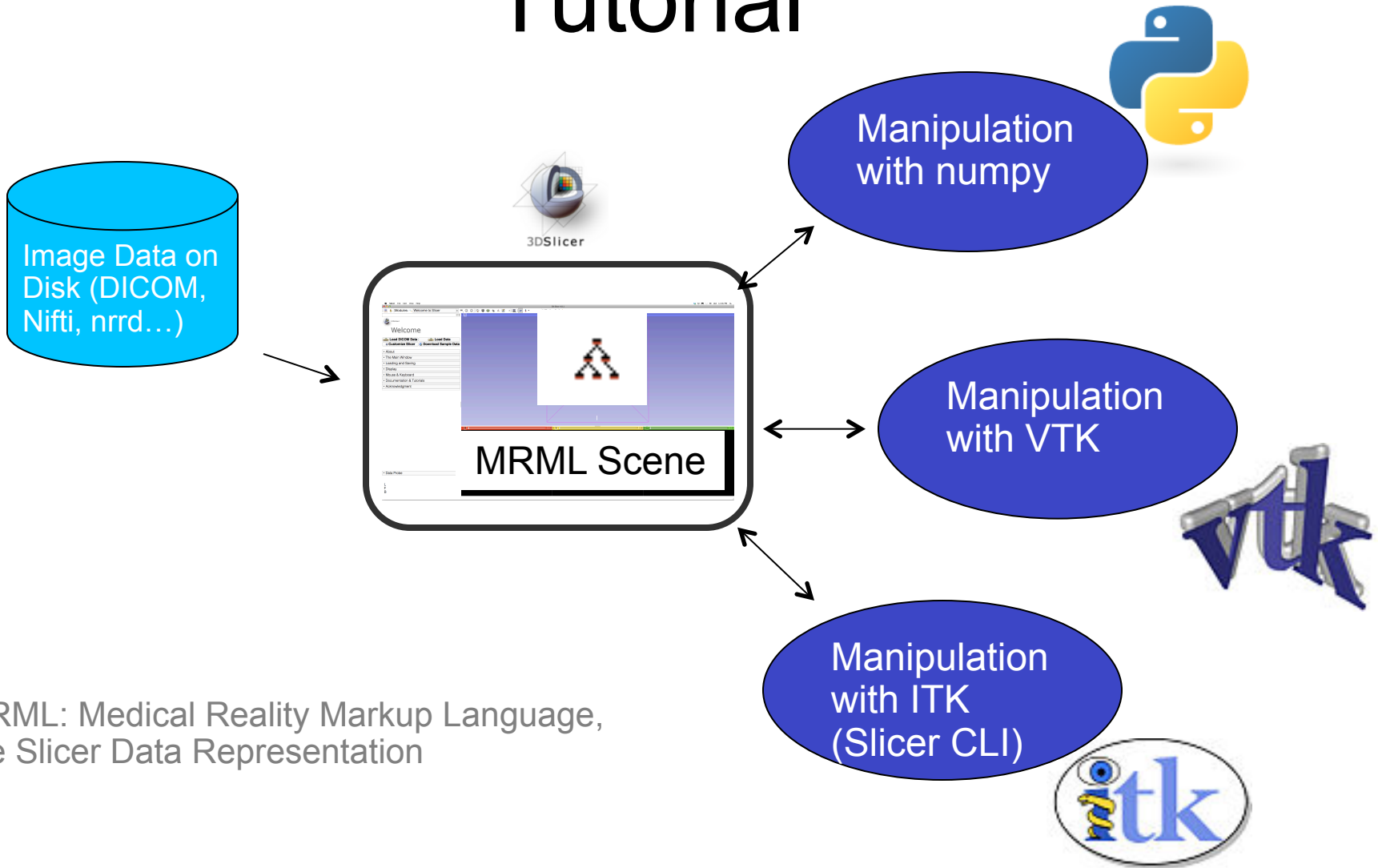
- scene objects (MRML)
- data arrays (volumes, models)
- GUI elements that can be encapsulated in a module
- Processing Libraries: numpy, VTK, ITK,CTK

Slicer4 Scripted Modules

- Python scripted modules allow more **interactive functionalities** (eg 'Flythrough' in Endoscopy module) and **rapid prototyping**
- GUI based on Qt libraries accessed via Python



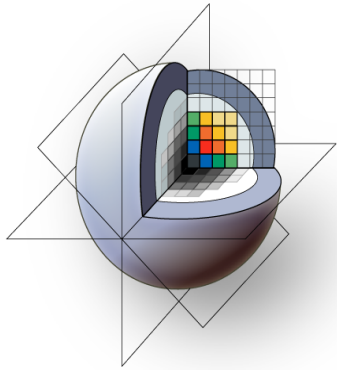
Processing Examples in this Tutorial



MRML: Medical Reality Markup Language, the Slicer Data Representation

Course Overview

- Part A: Exploring Slicer via Python
- Part B: Integration of the HelloPython.py program into Slicer4
- Part C: Implementation of the Laplace operator in the HelloPython module
- Part D: Image Sharpening using the Laplace operator



3DSlicer

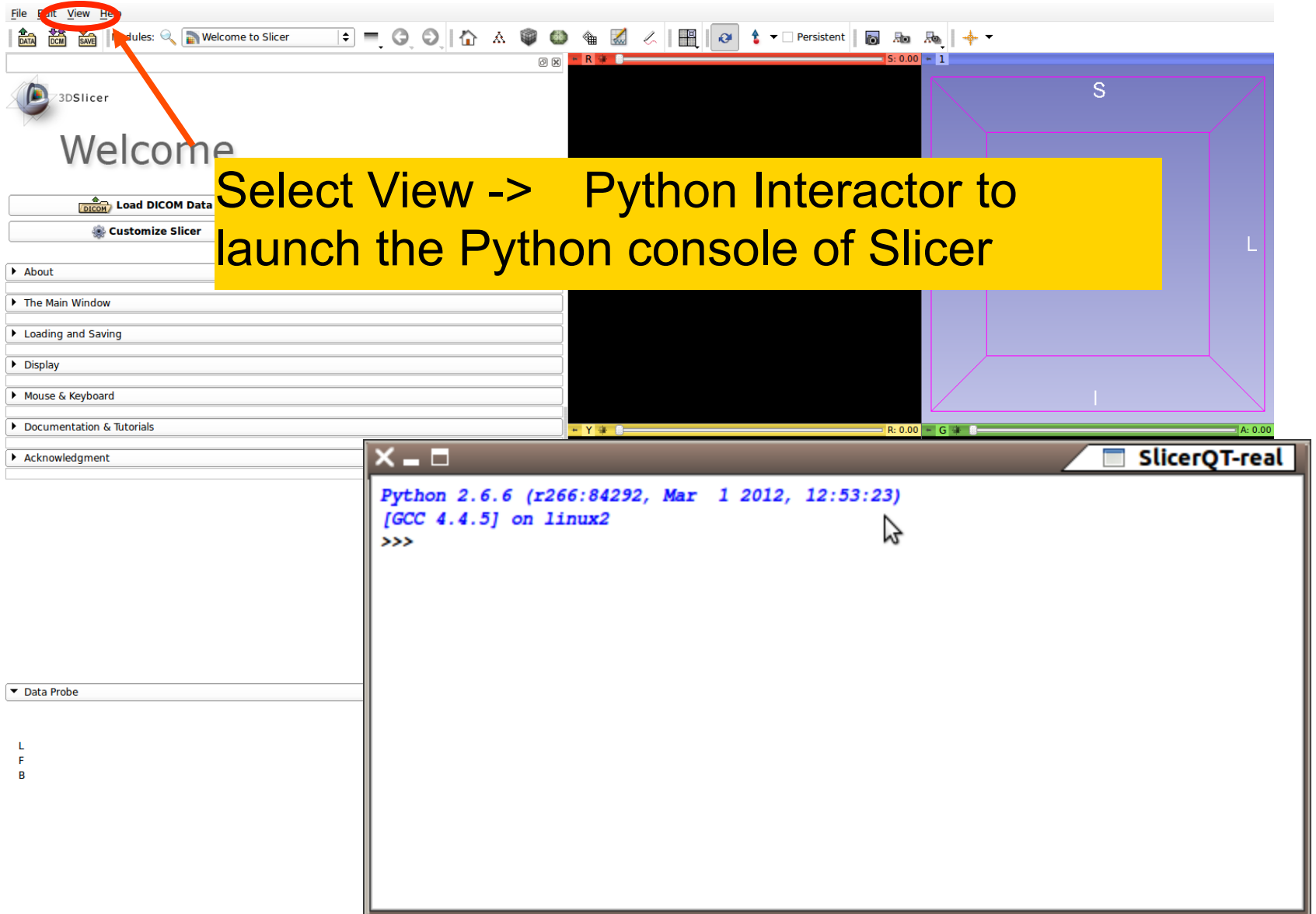


Part A: EXPLORING SLICER VIA PYTHON

Python in Slicer

- Slicer 4 includes python 2.6.6 and a rich set of standard libraries
 - *Included:* **numpy**, **vtk**, **ctk**, **PythonQt**, and most of standard python library
 - *Not included:*
 - scipy (scientific tools for python),
 - matplotlib (python 2D plotting library),
 - ipython (interactive python)
- and some other popular packages that we have found difficult to package for distribution

Python Console in Slicer



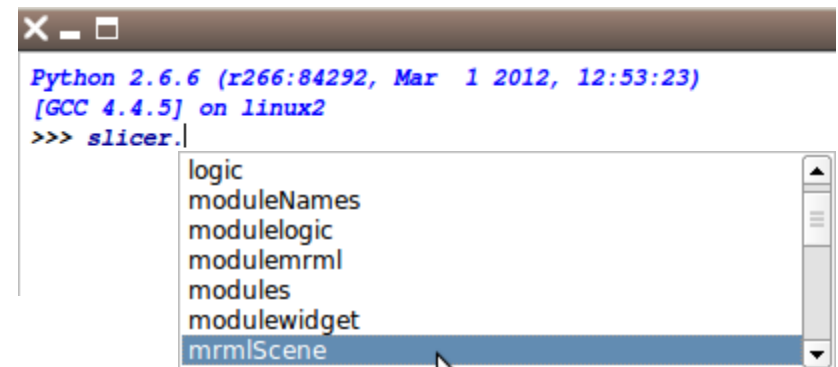
The image shows the 3DSlicer software interface. The 'View' menu in the top-left corner is circled in red, with an arrow pointing to a yellow callout box. The callout box contains the text: "Select View -> Python Interactor to launch the Python console of Slicer". In the bottom-right corner, a separate window titled "SlicerQT-real" is open, displaying a Python console. The console output shows the Python version and environment details, followed by a prompt for user input.

Select View -> Python Interactor to launch the Python console of Slicer

```
Python 2.6.6 (r266:84292, Mar 1 2012, 12:53:23)
[GCC 4.4.5] on linux2
>>>
```


General Python Console Features

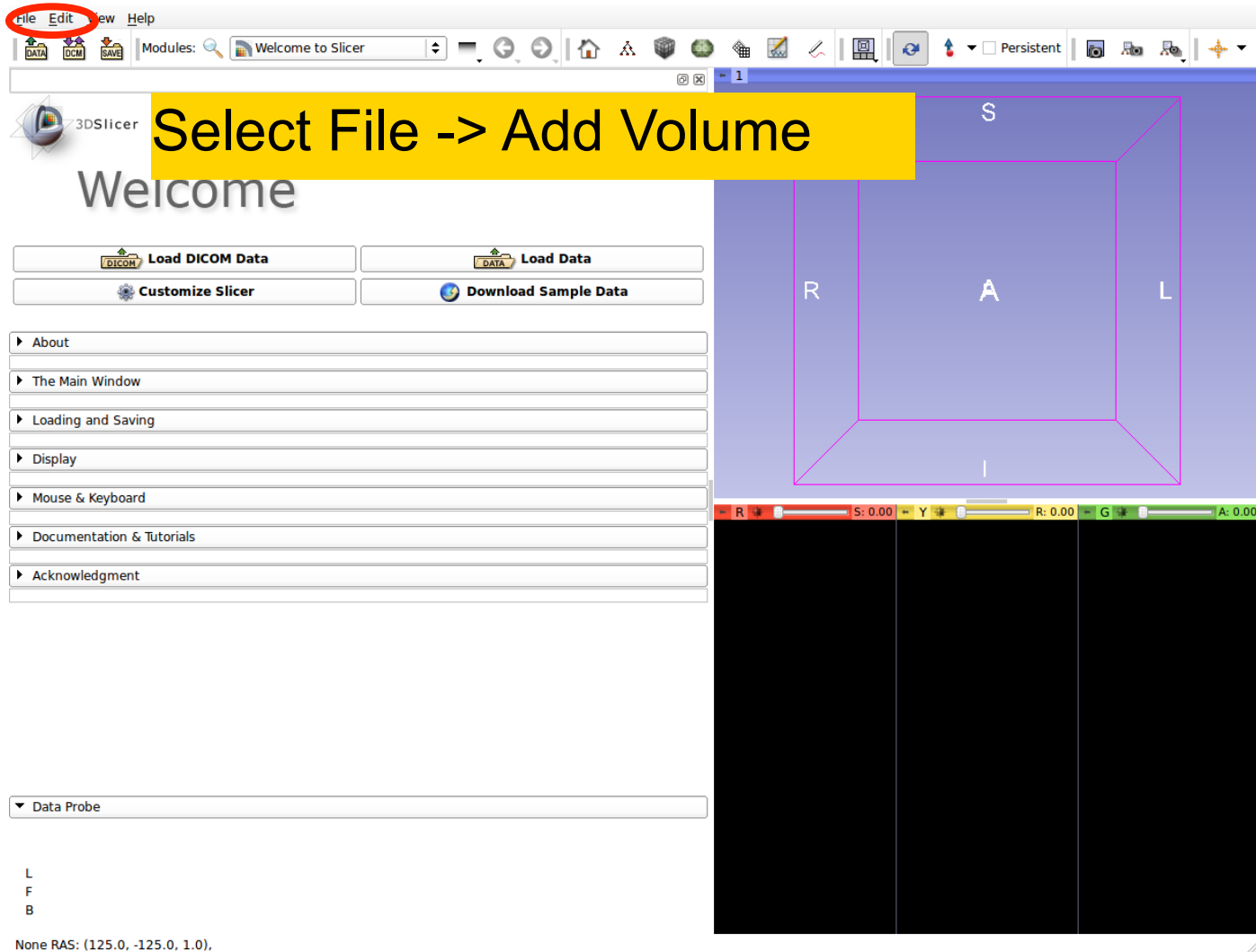
- Command Line Editing:
 - Left/Right Arrow Keys, End
 - Delete (Control-D)
- Input History
 - Up/Down Arrow Keys
- Command Completion
 - Tab Key



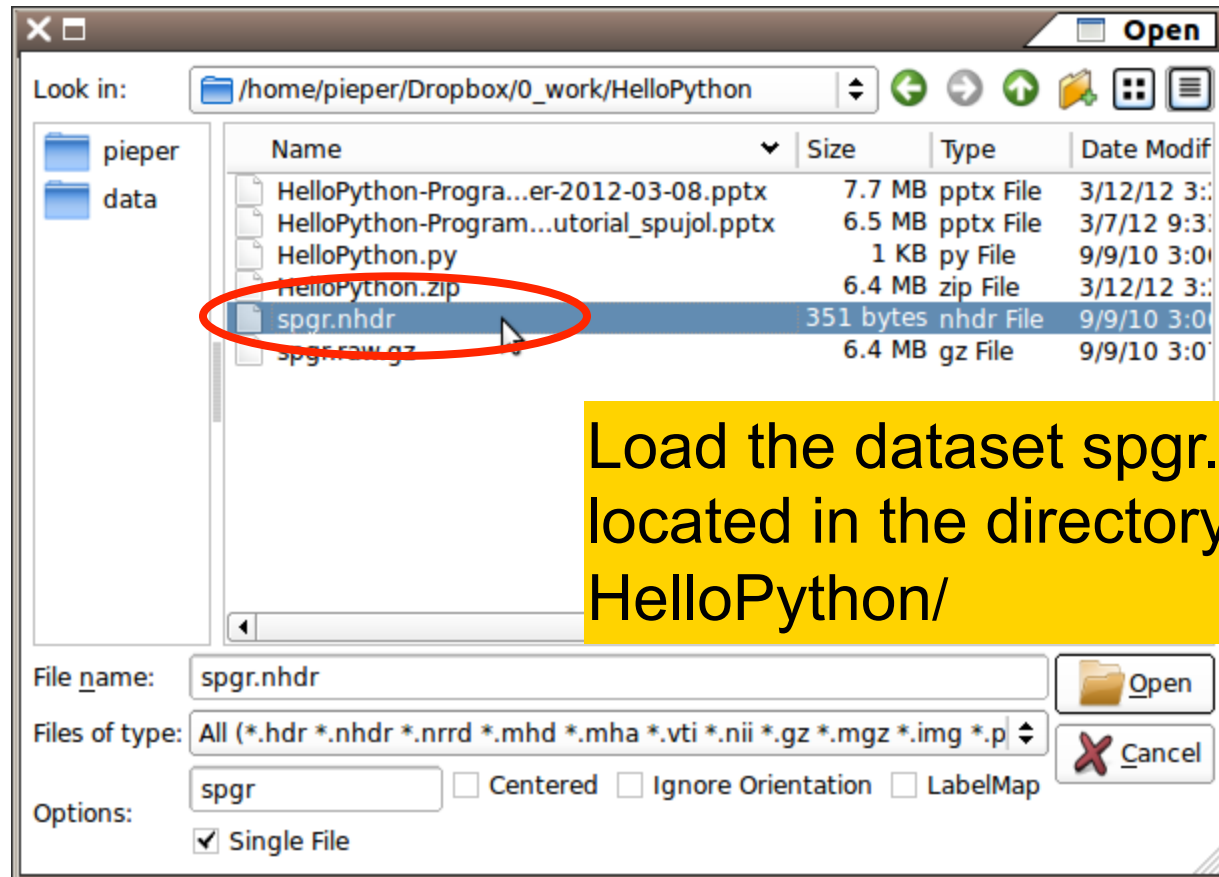
```
Python 2.6.6 (r266:84292, Mar  1 2012, 12:53:23)
[GCC 4.4.5] on linux2
>>> slicer.|
logic
moduleNames
modulelogic
modulermml
modules
modulewidget
mrmlScene
```

The screenshot shows a terminal window with a Python prompt. The user has entered 'slicer.' and a list of completion suggestions is displayed. The suggestions are: logic, moduleNames, modulelogic, modulermml, modules, modulewidget, and mrmlScene. The 'mrmlScene' option is currently selected and highlighted in blue.

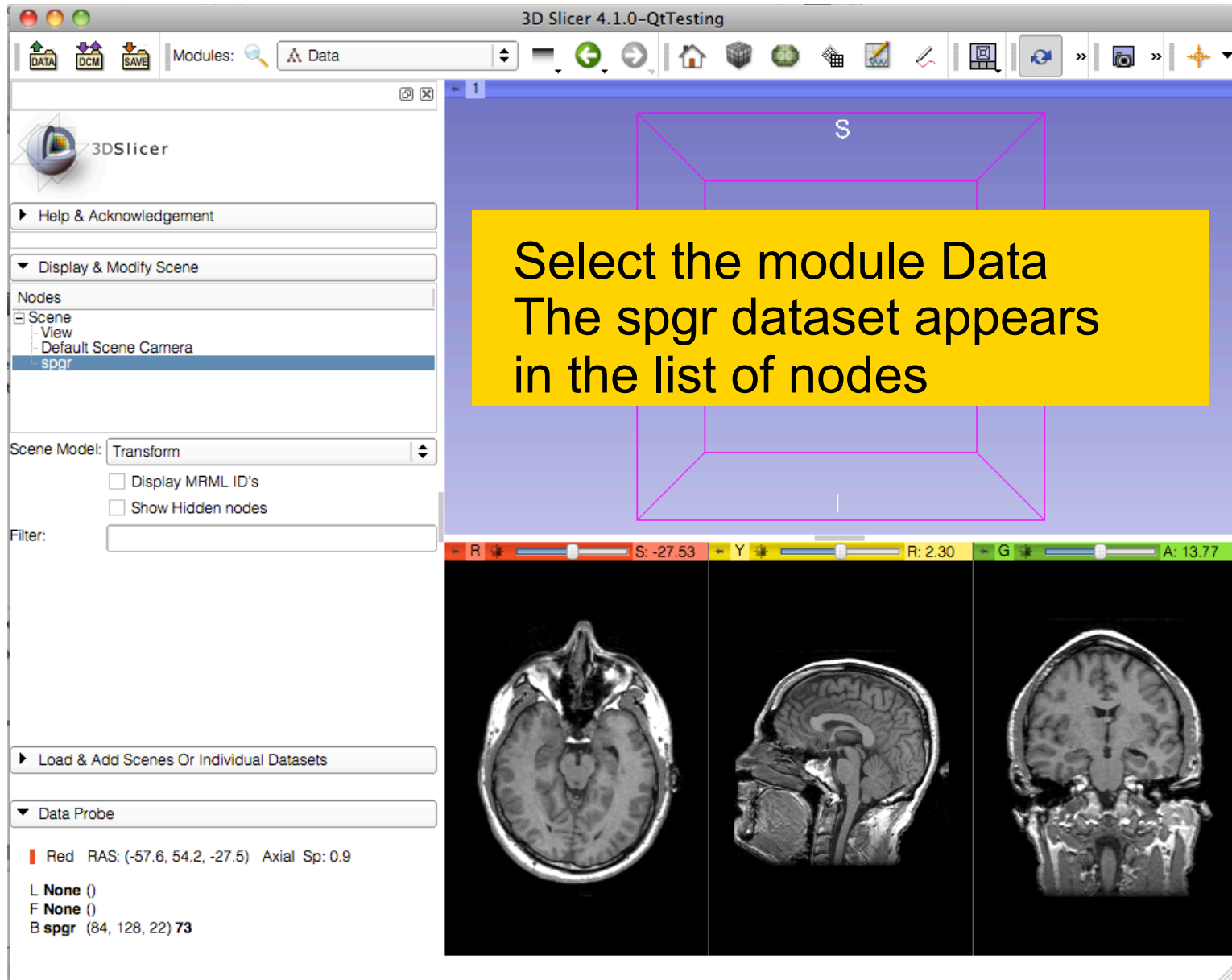
Add Volume Dialog



Add spgr.nhdr



Load the dataset spgr.nhdr
located in the directory
HelloPython/



Access to MRML and Arrays

```
Python 2.6.6 (r266:84292, Mar 15 2012, 03:03:01)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
>>> a = slicer.util.array('spgr')
>>> print(a)
```

Run the following code in the Python console

a = slicer.util.array('spgr')

→ Uses the slicer.util package to return a numpy array of the image

→ The variable 'a' is a numpy ndarray of the volume data we just loaded

print(a)

→ Shows a shortened view of the array

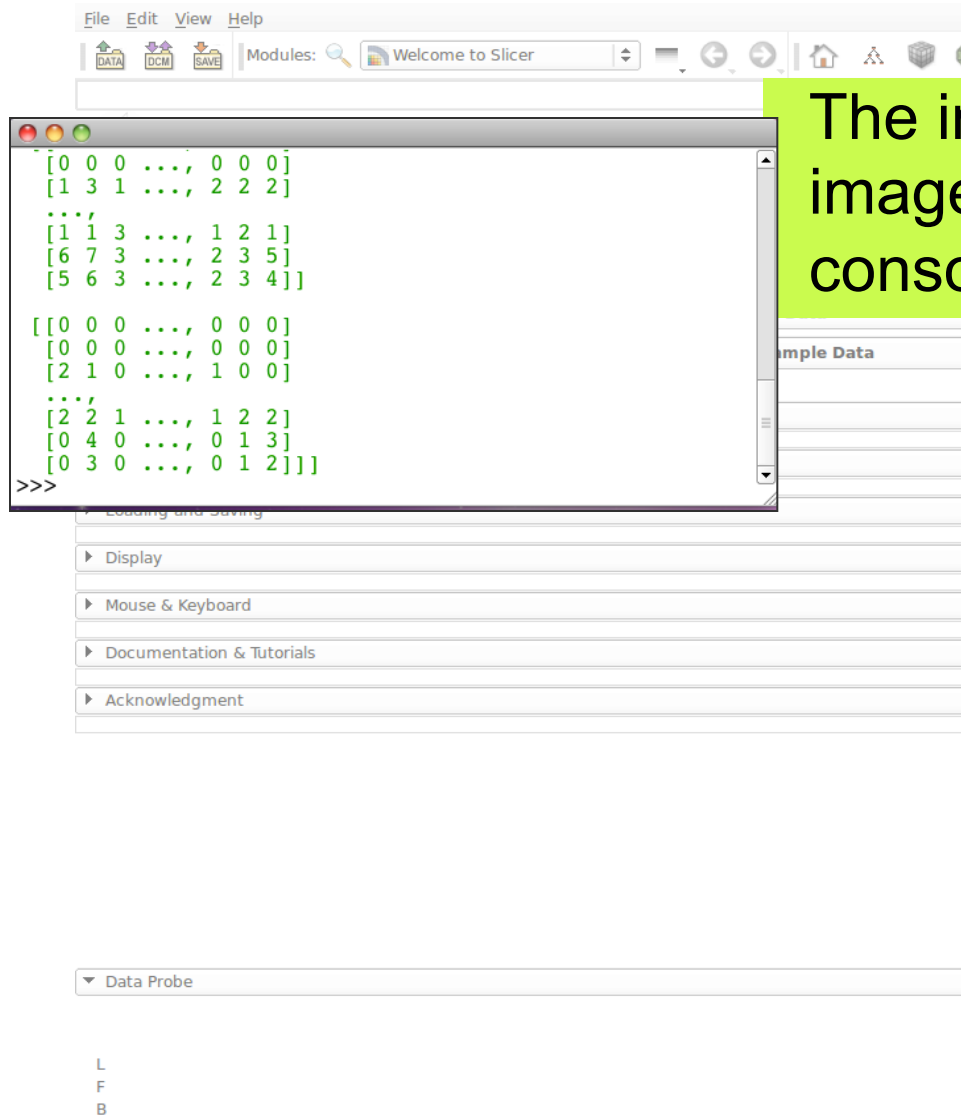
- ▶ The Main Window
- ▶ Loading and Saving
- ▶ Display
- ▶ Mouse & Keyboard
- ▶ Documentation & Tutorials
- ▶ Acknowledgment

▼ Data Probe

L
F
B



Access to MRML and Arrays



The screenshot shows the Slicer software interface. A Python console window is open, displaying the following array data:

```
>>> [0 0 0 ... 0 0 0]
      [1 3 1 ... 2 2 2]
      ...
      [1 1 3 ... 1 2 1]
      [6 7 3 ... 2 3 5]
      [5 6 3 ... 2 3 4]

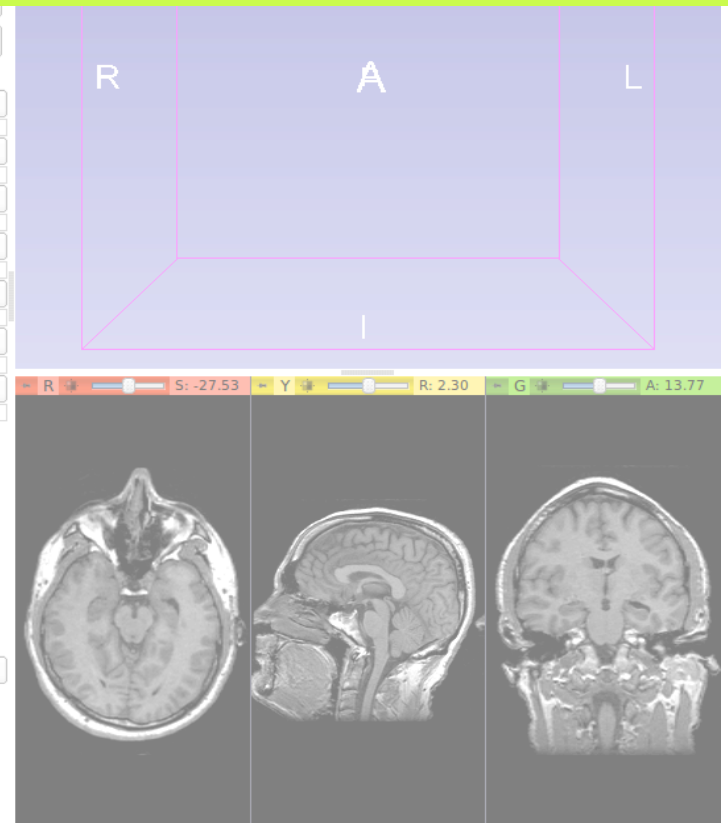
      [[0 0 0 ... 0 0 0]
       [0 0 0 ... 0 0 0]
       [2 1 0 ... 1 0 0]
       ...
       [2 2 1 ... 1 2 2]
       [0 4 0 ... 0 1 3]
       [0 3 0 ... 0 1 2]]]
>>>
```

The sidebar on the right contains the following menu items:

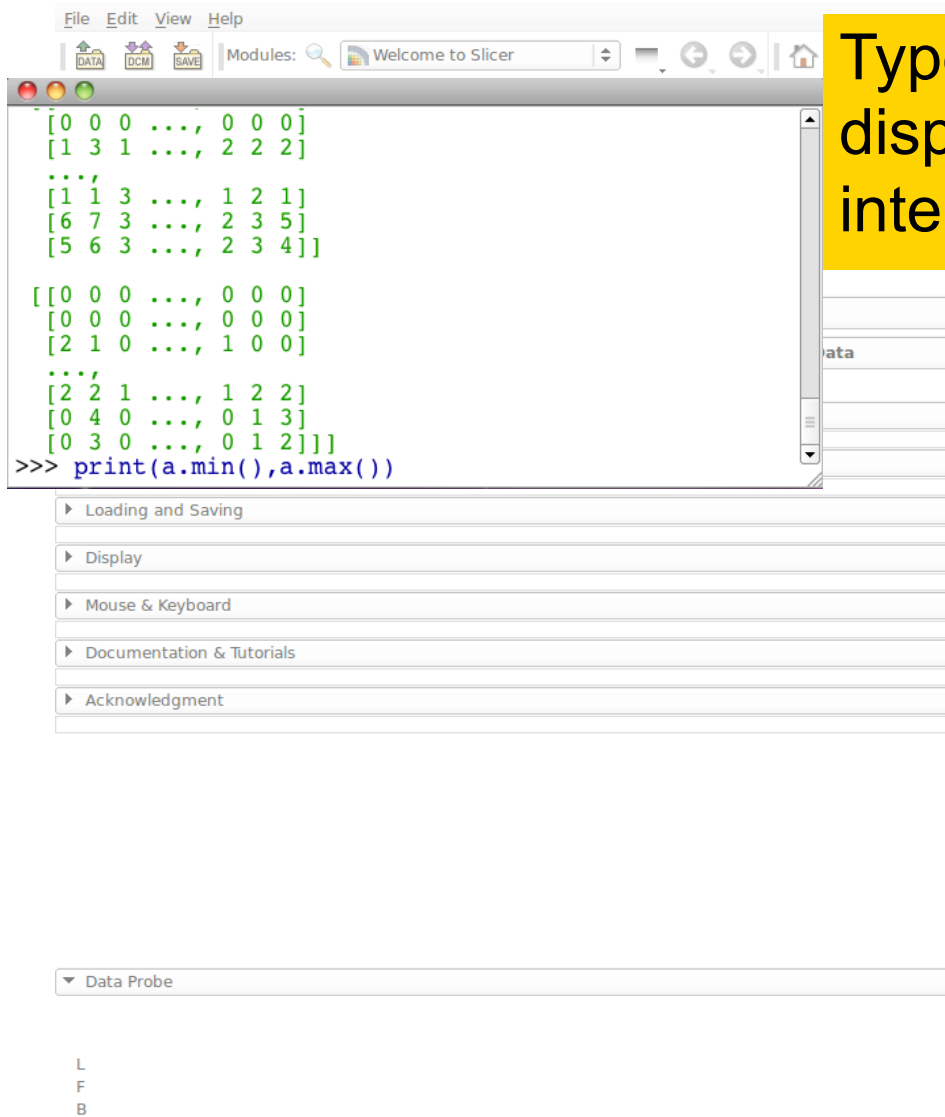
- Display
- Mouse & Keyboard
- Documentation & Tutorials
- Acknowledgment

At the bottom, there is a 'Data Probe' dropdown menu and a list of letters: L, F, B.

The intensity values of the spgr image appear in the Python console



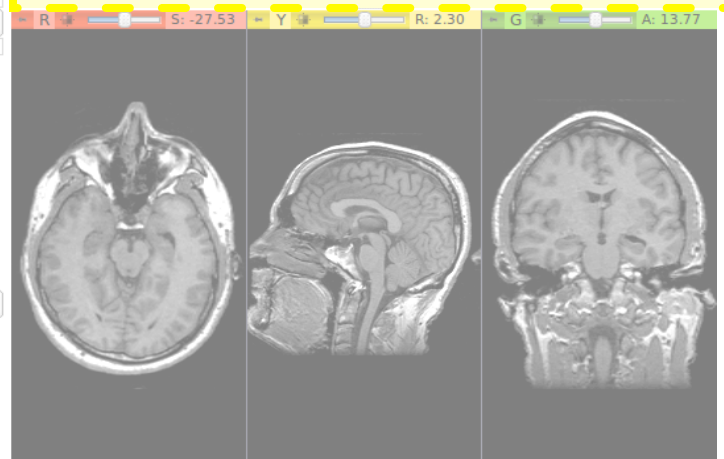
Access to MRML and Arrays



Type the following command to display the min and max intensity value of the spgr image

```
print( a.min(), a.max() )
```

→ Use numpy array methods to explore the data



Access to MRML and Arrays

The screenshot displays the Slicer software interface. On the left, a console window shows the following code and output:

```
...  
[1 1 3 ... 1 2 1]  
[6 7 3 ... 2 3 5]  
[5 6 3 ... 2 3 4]  
  
[[0 0 0 ... 0 0 0]  
[0 0 0 ... 0 0 0]  
[2 1 0 ... 1 0 0]  
  
...  
[2 2 1 ... 1 2 2]  
[0 4 0 ... 0 1 3]  
[0 3 0 ... 0 1 2]]]  
>>> print(a.min(),a.max())  
(0, 355)  
>>>
```

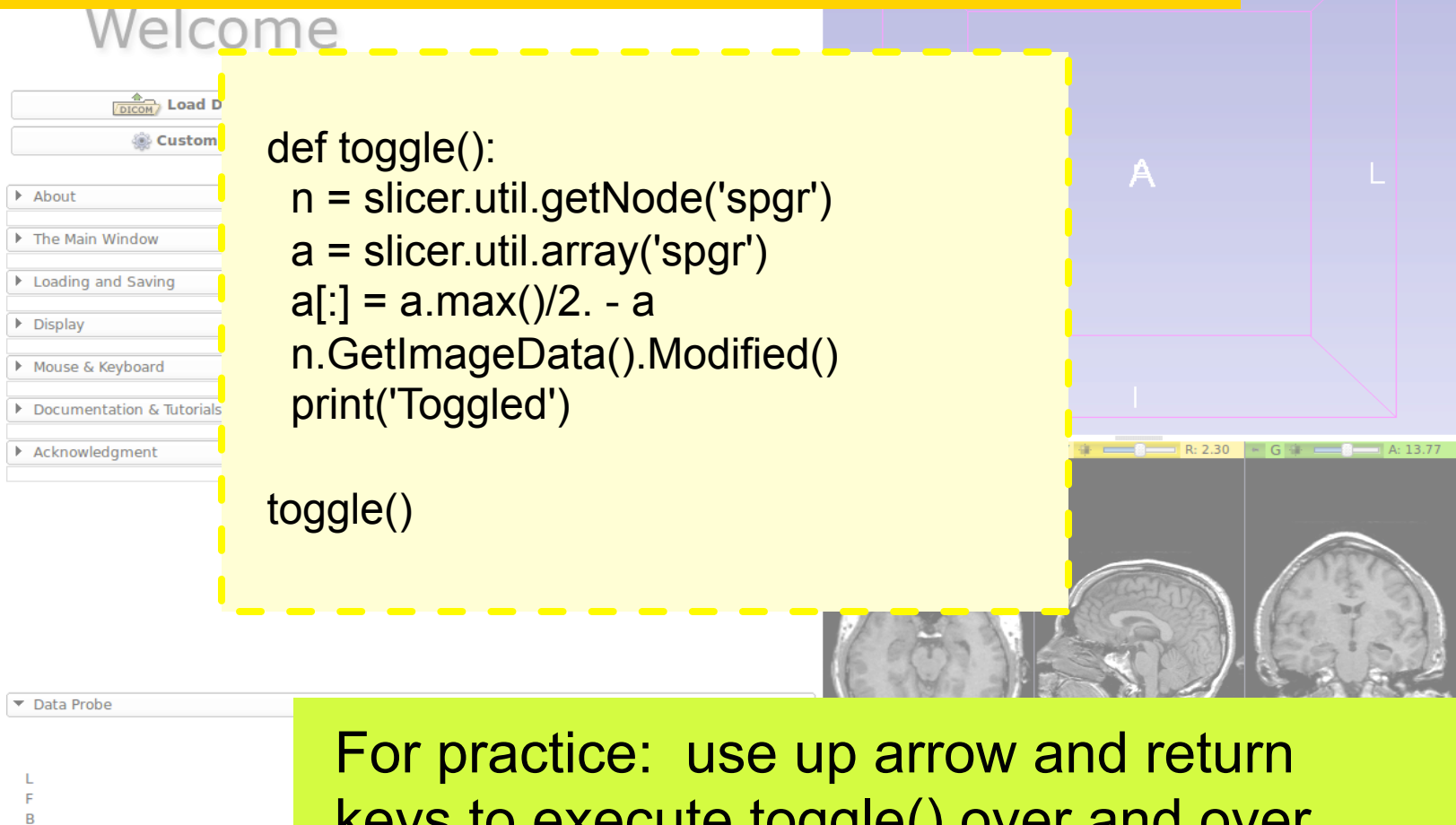
Below the console, a sidebar contains menu items: Display, Mouse & Keyboard, Documentation & Tutorials, and Acknowledgment. At the bottom left, a 'Data Probe' section shows the letters L, F, and B.

On the right side of the interface, a 3D brain model is shown with a purple bounding box. A yellow callout box above the model contains the text: **I min = 0 ; I max = 355**. The model is labeled with 'R' (Right), 'A' (Anterior), and 'I' (Inferior). Below the 3D view, a control bar shows sliders for R (2.30), G, and A (13.77).

At the bottom of the interface, three 2D MRI slices are displayed: an axial view on the left, a sagittal view in the middle, and a coronal view on the right.

Manipulating Arrays

Run the following code in the Python console, **(indent each new line with 2 spaces)**

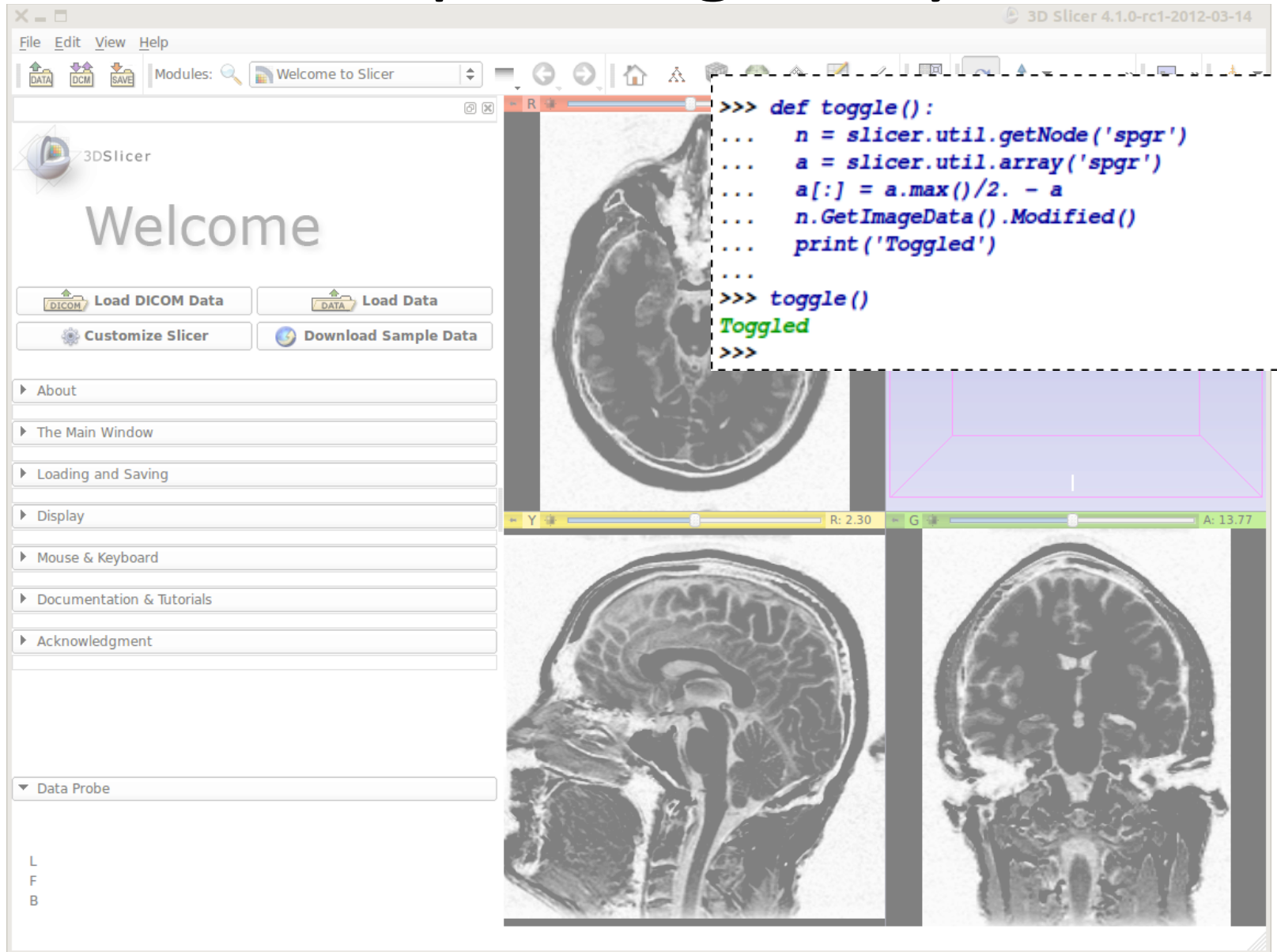
The image shows a screenshot of the Slicer software interface. A yellow dashed box highlights a Python console window. The console contains the following code:

```
def toggle():  
    n = slicer.util.getNode('spgr')  
    a = slicer.util.array('spgr')  
    a[:] = a.max()/2. - a  
    n.GetImageData().Modified()  
    print('Toggled')  
  
toggle()
```

The background shows the Slicer interface with a 3D view of a brain slice and a 2D view of the same slice. The 3D view has axes labeled 'A' and 'L'. The 2D view shows three slices of the brain. The interface also includes a 'Welcome' message, a 'Load Data' button, and a 'Custom' button. A 'Data Probe' window is visible at the bottom left.

For practice: use up arrow and return keys to execute toggle() over and over

Manipulating Arrays



The screenshot displays the 3D Slicer 4.1.0-rc1-2012-03-14 interface. The main window shows a 'Welcome' message and several buttons: 'Load DICOM Data', 'Load Data', 'Customize Slicer', and 'Download Sample Data'. A sidebar on the left contains a table of contents with sections like 'About', 'The Main Window', 'Loading and Saving', 'Display', 'Mouse & Keyboard', 'Documentation & Tutorials', and 'Acknowledgment'. At the bottom left, there is a 'Data Probe' section with 'L', 'F', and 'B' labels.

Overlaid on the right side of the interface is a Python console window with the following code and output:

```
>>> def toggle():  
...     n = slicer.util.getNode('spgr')  
...     a = slicer.util.array('spgr')  
...     a[:] = a.max()/2. - a  
...     n.GetImageData().Modified()  
...     print('Toggled')  
...  
>>> toggle()  
Toggled  
>>>
```

The console output shows 'Toggled' in green text. The background of the console window shows three MRI brain slices: an axial slice at the top, a sagittal slice at the bottom left, and a coronal slice at the bottom right. A purple rectangular box is overlaid on the right side of the console window, partially obscuring the coronal slice. Below the console window, there are two sliders: a yellow one labeled 'Y' with 'R: 2.30' and a green one labeled 'G' with 'A: 13.77'.

The toggle function in more detail

- **def toggle():**
 - Defines a python function
 - Body of function performs element-wise math on entire volume
 - Easy mix of scalar and volume math
- Telling slicer that the image data for node 'n' has been modified causes the slice view windows to refresh

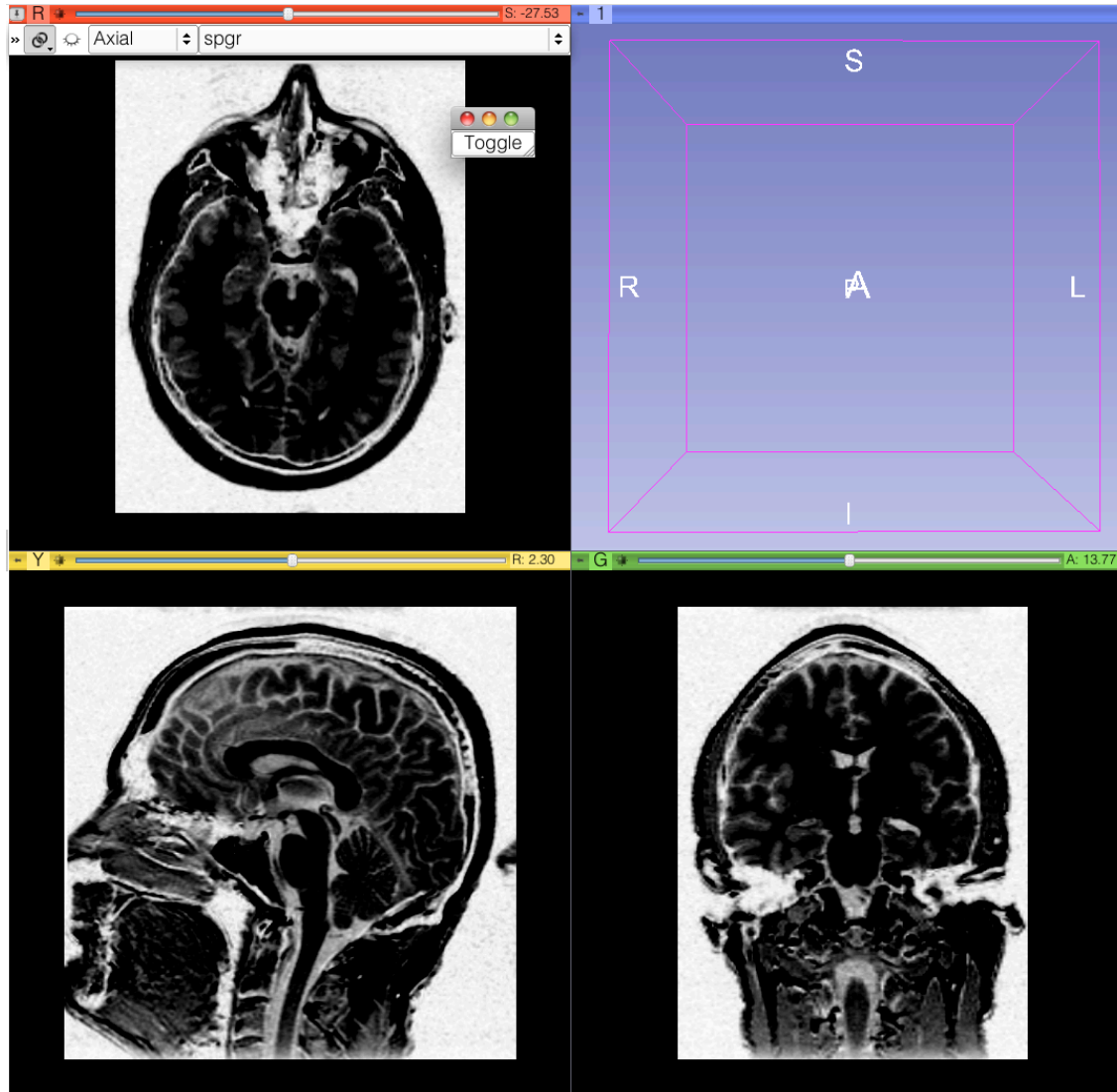
Qt GUI in Python

Run the following code in the Python console

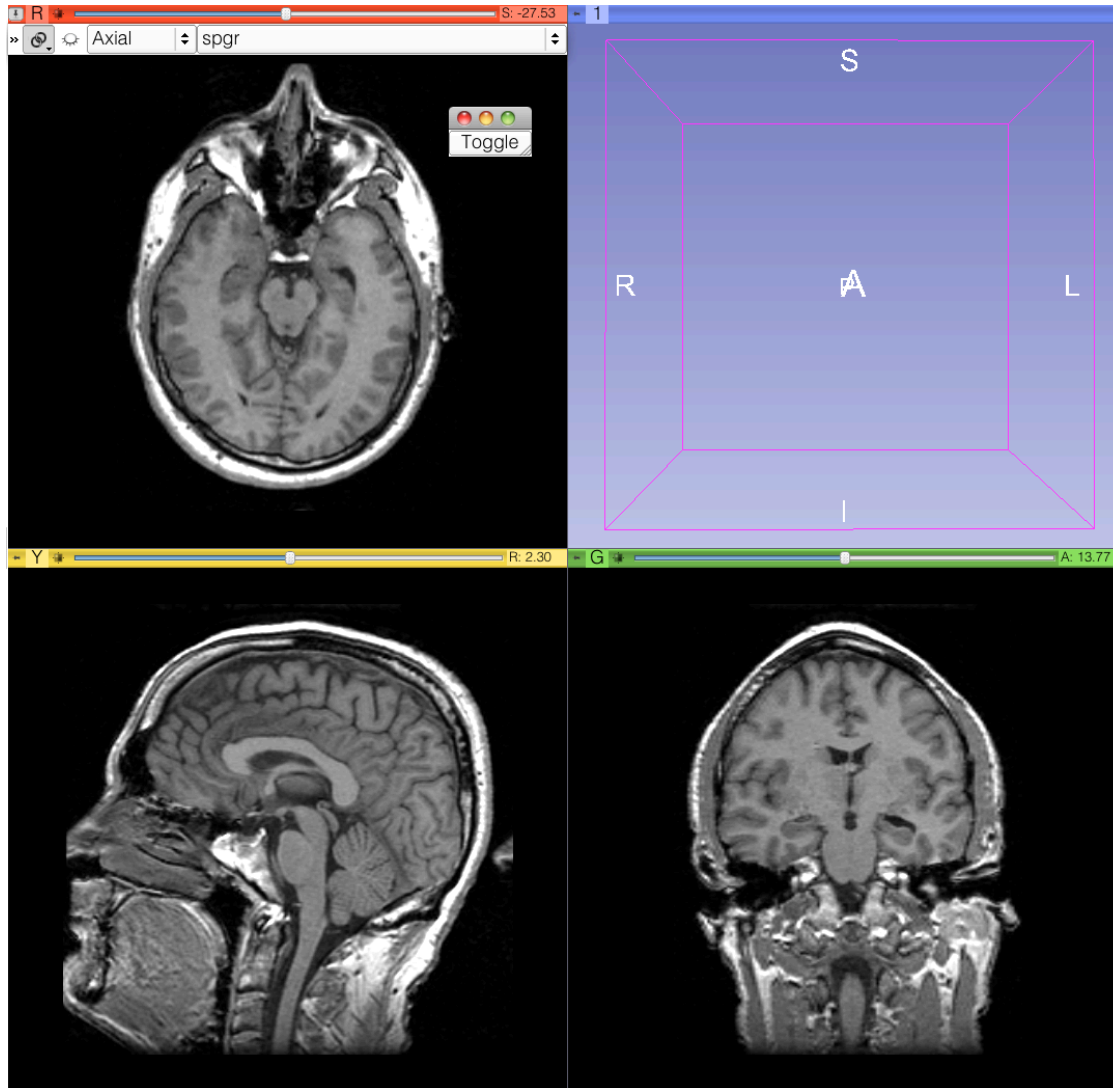
```
b = qt.QPushButton('Toggle')  
b.connect('clicked()', toggle)  
b.show()
```

What do you think will happen when you run this code? What about when you push the button?

Result with button toggling



Result with button toggling



In More Detail

- Slicer uses **PythonQt** to expose the Qt library
- Sophisticated interactive modules can be written entirely with Python code calling C++ code that is wrapped in Python (e.g. Endoscopy, Editor, SampleData, ChangeTracker, and other slicer modules in the Slicer source code)

(*) Qt: <http://qt.nokia.com>

(**) PythonQt: <http://pythonqt.sf.net> /F.Link (MeVis)

```
File Edit Tools Syntax Buffers Window Help
HelloPython.py C:/Bropbox/0_work/helloPython/HelloPython - C:\VIM

from __main__ import vtk, qt, ctk, slicer

# HelloPython
#

class HelloPython:
    def __init__(self, parent):
        parent.title = "Hello Python"
        parent.categories = ["Examples"]
        parent.dependencies = []
        parent.contributors = ["Jean-Christophe Fillard-Robin (Kitware),
                               Steve Pieper (Isomics), Sonia Pujol (Dell)"] # replace with "Firstname Lastname (Org)"
        parent.helpText = """
        Example of scripted loadable extension for the HelloPython tutorial.
        """
        parent.acknowledgementText = """
        This file was originally developed by Jean-Christophe Fillard-Robin, Kitware Inc.,
        Steve Pieper, Isomics, Inc., and Sonia Pujol, Brigham and Women's Hospital and was
        partially funded by NIH grant 5P41MH082228-02 (NAC) and is part of the National Alliance
        For Medical Image Computing (NA-MIC), funded by the National Institutes of Health through the
        NIH Center for Medical Research, Grant U54 EB005149.""" # replace with organization, grant and thanks.
        self.parent = parent

# @HelloPythonWidget
#

class HelloPythonWidget:
    def __init__(self, parent = None):
        if not parent:
            self.parent = slicer.qMRMLWidget()
            self.parent.setLayout(qt.QVBoxLayout())
            self.parent.setMRMLScene(slicer.mrmlScene)
        else:
            self.parent = parent
        self.layout = self.parent.layout()
        if not parent:
            self.setup()
            self.parent.show()

    def setup(self):
        # Instantiate and connect widgets ...

        # Collapsible button
        dummyCollapsibleButton = ctk.ctkCollapsibleButton()
        dummyCollapsibleButton.text = "A collapsible button"
        self.layout.addWidget(dummyCollapsibleButton)

        # Layout within the dummy collapsible button
        dummyFormLayout = qt.QFormLayout(dummyCollapsibleButton)

        # HelloWorld button
        helloWorldButton = qt.QPushButton("Hello world")
        helloWorldButton.setToolTip("Print 'hello world' in standard output.")
        dummyFormLayout.addWidget(helloWorldButton)
        helloWorldButton.connect(clickedSignal, self.onHelloWorldButtonClicked)

        # Add vertical spacer
        self.layout.addStretch()

        # Set local var as instance attribute
        self.helloWorldButton = helloWorldButton

    def onHelloWorldButtonClicked(self):
        print "hello world"
        qt.QMessageBox.information(slicer.util.mainWindow(), "Slicer Python", "Hello World!")

HelloPython.py 22,8 All
```



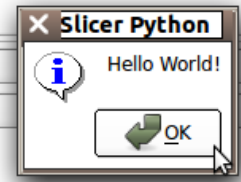
▼ Help & Acknowledgement

Help Acknowledgement

Example of scripted loadable extension for the HelloPython tutorial.

▼ A collapsible button

Hello world



PART B: INTEGRATION OF THE HELLOPYTHON CODE TO SLICER4

HelloPython.py

```
File Edit Tools Syntax Buffers Window Help
HelloPython.py (~/.Dropbox/0_work/HelloPython/HelloPython) - GVIM

from __main__ import vtk, qt, ctk, slicer

# HelloPython
#

class HelloPython:
    def __init__(self, parent):
        parent.title = "Hello Python"
        parent.categories = ["Examples"]
        parent.dependencies = []
        parent.contributors = ["Jean-Christophe Fillion-Robin (Kitware),
                               Steve Pieper (Isomics), Sonia Pujol (BWH)"] # replace with "Firstname Lastname (Org)"
        parent.helpText = """
        Example of scripted loadable extension for the HelloPython tutorial.
        """
        parent.acknowledgementText = """
        This file was originally developed by Jean-Christophe Fillion-Robin, Kitware Inc.,
        Steve Pieper, Isomics, Inc., and Sonia Pujol, Brigham and Women's Hospital and was
        partially funded by NIH grant 3P41RR013218-12S1 (NAC) and is part of the National Alliance
        for Medical Image Computing (NA-MIC), funded by the National Institutes of Health through the
        NIH Roadmap for Medical Research, Grant U54 EB005149.""" # replace with organization, grant and thanks.
        self.parent = parent

#
# qHelloPythonWidget
#

class HelloPythonWidget:
    def __init__(self, parent = None):
        if not parent:
            self.parent = slicer.qMRMLWidget()
            self.parent.setLayout(qt.QVBoxLayout())
            self.parent.setMRMLScene(slicer.mrmlScene)
        else:
            self.parent = parent
            self.layout = self.parent.layout()
            if not parent:

# HelloWorldButton
helloWorldButton = qt.QPushButton("Hello world")
helloWorldButton.setToolTip = "Print 'Hello world' in standard output."
dummyFormLayout.addWidget(helloWorldButton)
helloWorldButton.connect('clicked(bool)', self.onHelloWorldButtonClicked)

# Add vertical spacer
self.layout.addStretch(1)

# Set local var as instance attribute
self.helloWorldButton = helloWorldButton

def onHelloWorldButtonClicked(self):
    print "Hello World !"
    qt.QMessageBox.information(slicer.util.mainWindow(), 'Slicer Python', 'Hello World!')

~
~
HelloPython.py 22,8 All
```

Open the file HelloPython.py
located in the directory HelloPython

HelloPython.py

Module
Description

Module GUI

Processing
Code

```
File Edit Tools Syntax Buffers Window Help
HelloPython.py (~/.Dropbox/0_work/HelloPython/HelloPython) - GVIM

from __main__ import vtk, qt, ctk, slicer

#
# HelloPython
#

class HelloPython:
    def __init__(self, parent):
        parent.title = "Hello Python"
        parent.categories = ["Examples"]
        parent.dependencies = []
        parent.contributors = ["Jean-Christophe Fillion-Robin (Kitware),
                               Steve Pieper (Isomics), Sonia Pujol (BWH)"] # replace with "Firstname Lastname (Org)"
        parent.helpText = """
        Example of scripted loadable extension for the HelloPython tutorial.
        """
        parent.acknowledgementText = """
        This file was originally developed by Jean-Christophe Fillion-Robin, Kitware Inc.,
        Steve Pieper, Isomics, Inc., and Sonia Pujol, Brigham and Women's Hospital and was
        partially funded by NIH grant 3P41RR013218-12S1 (NAC) and is part of the National Alliance
        for Medical Image Computing (NA-MIC), funded by the National Institutes of Health through the
        NIH Roadmap for Medical Research, Grant U54 EB005149.""" # replace with organization, grant and thanks.
        self.parent = parent

#
# qHelloPythonWidget
#

class HelloPythonWidget:
    def __init__(self, parent = None):
        if not parent:
            self.parent = slicer.qMRMLWidget()
            self.parent.setLayout(qt.QVBoxLayout())
            self.parent.setMRMLScene(slicer.mrmlScene)
        else:
            self.parent = parent
            self.layout = self.parent.layout()
        if not parent:
            self.setup()

    def setup(self):
        # Instantiate and connect widgets ...

        # Collapsible button
        dummyCollapsibleButton = ctk.ctkCollapsibleButton()
        dummyCollapsibleButton.text = "A collapsible button"
        self.layout.addWidget(dummyCollapsibleButton)

        # Layout within the dummy collapsible button
        dummyFormLayout = qt.QFormLayout(dummyCollapsibleButton)

        # HelloWorld button
        helloWorldButton = qt.QPushButton("Hello world")
        helloWorldButton.setToolTip = "Print 'Hello world' in standard output."
        dummyFormLayout.addWidget(helloWorldButton)
        helloWorldButton.connect('clicked(bool)', self.onHelloWorldButtonClicked)

        # Add vertical spacer
        self.layout.addStretch(1)

        # Set local var as instance attribute
        self.helloWorldButton = helloWorldButton

    def onHelloWorldButtonClicked(self):
        print "Hello World !"
        qt.QMessageBox.information(slicer.util.mainWindow(), 'Slicer Python', 'Hello World!')

~
HelloPython.py 22,8 All
```

Module Description

```
class HelloPython:
    def __init__(self, parent): ← constructor
        parent.title = "Hello Python"
        parent.categories = ["Examples"]
        parent.dependencies = []
        parent.contributors = ["Jean-Christophe Fillion-Robin (Kitware)",
                               "Steve Pieper (Isomics)",
                               "Sonia Pujol (BWH)"] # replace with "Firstname Lastname (Org)"
        parent.helpText = """
        Example of scripted loadable extension for the HelloPython tutorial.
        """
        parent.acknowledgementText = """
        This file was originally developed by Jean-Christophe Fillion-Robin, Kitware Inc.,
        Steve Pieper, Isomics, Inc., and Sonia Pujol, Brigham and Women's Hospital and was
        partially funded by NIH grant 3P41RR013218-12S1 (NAC) and is part of the National Alliance
        for Medical Image Computing (NA-MIC), funded by the National Institutes of Health through
        the NIH Roadmap for Medical Research, Grant U54 EB005149.""" # replace with organization,
        grant and thanks.
        self.parent = parent
```

This code is
provided in
the template

Module GUI

```
def setup(self):
    # Instantiate and connect widgets ...

    # Collapsible button
    sampleCollapsibleButton = ctk.ctkCollapsibleButton()
    sampleCollapsibleButton.text = "A collapsible button"
    self.layout.addWidget(sampleCollapsibleButton)

    # Layout within the sample collapsible button
    sampleFormLayout = qt.QFormLayout(sampleCollapsibleButton)
```

```
# HelloWorld button
helloWorldButton = qt.QPushButton("Hello world")
helloWorldButton.setToolTip("Print 'Hello world' in standard output.")
sampleFormLayout.addWidget(helloWorldButton)
helloWorldButton.connect('clicked(bool)', self.onHelloWorldButtonClicked)
```

```
# Add vertical spacer
self.layout.addStretch(1)

# Set local var as instance attribute
self.helloWorldButton = helloWorldButton
```

Add this
Text in
section A

Processing Code

```
def onHelloWorldButtonClicked(self):  
    print "Hello World !"
```

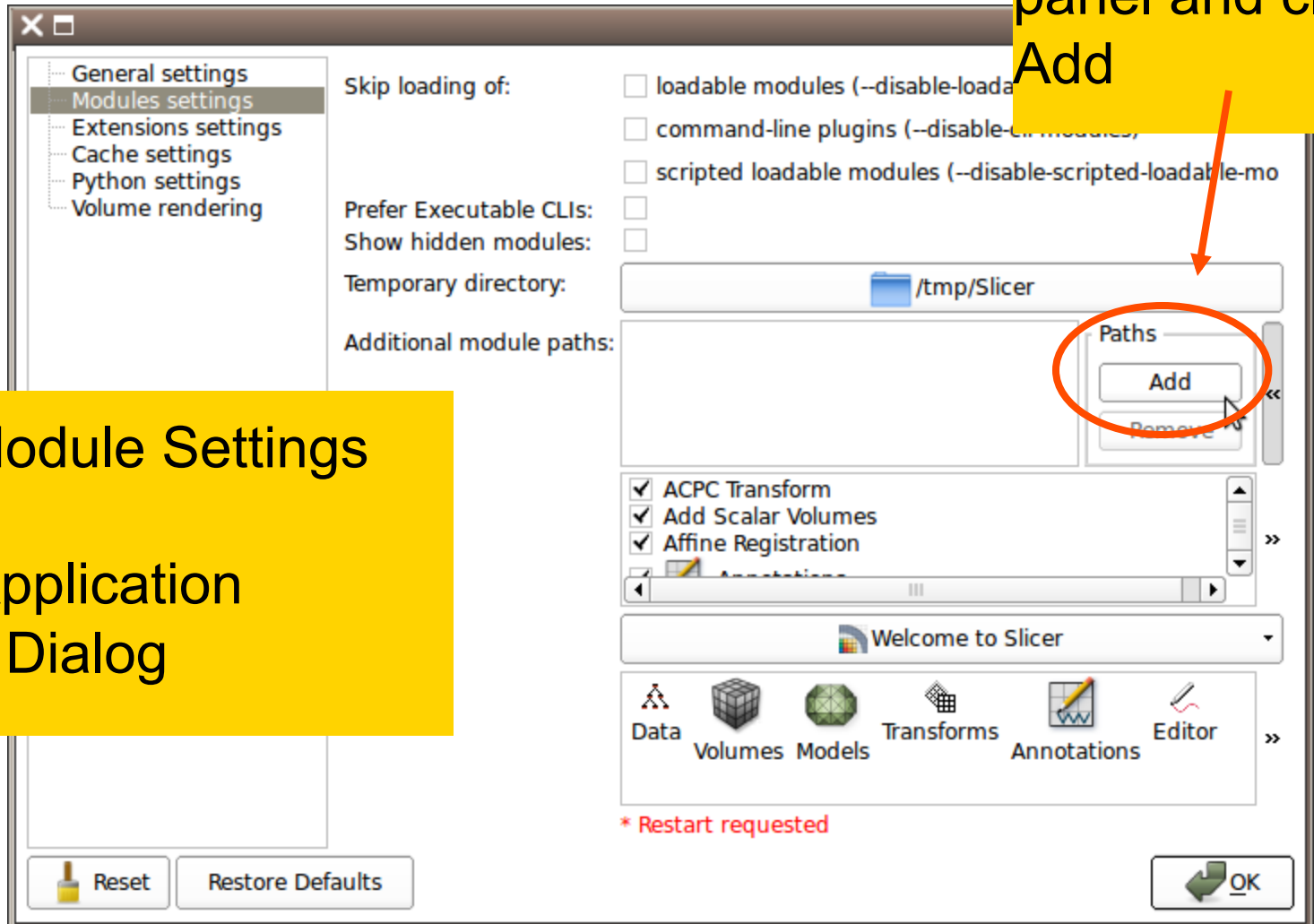
```
qt.QMessageBox.information(  
    slicer.util.mainWindow(),  
    'Slicer Python',  
    'Hello World!')
```

**Add this
Text in
section B**

Integrating HelloPython

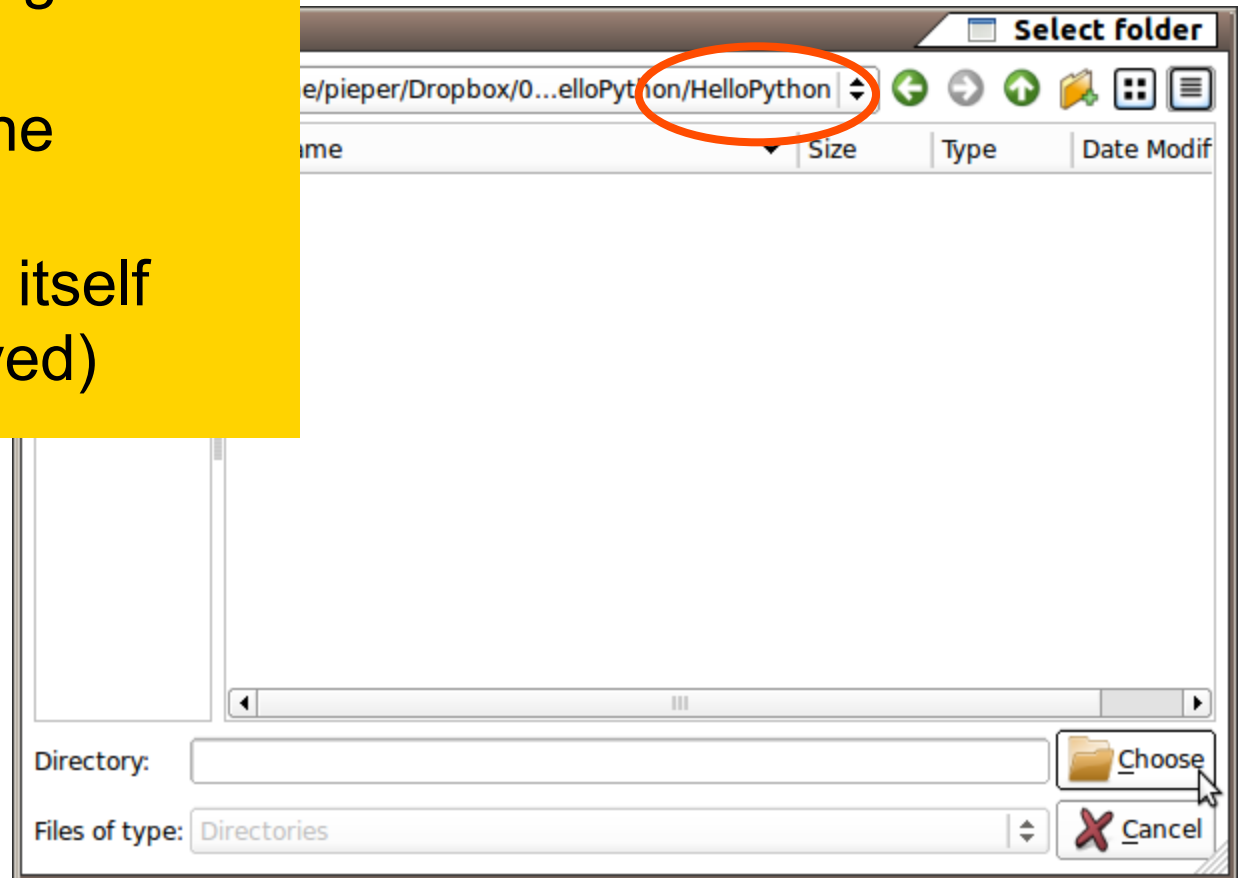
Open the side panel and click Add

Select Module Settings from the Edit -> Application Settings Dialog



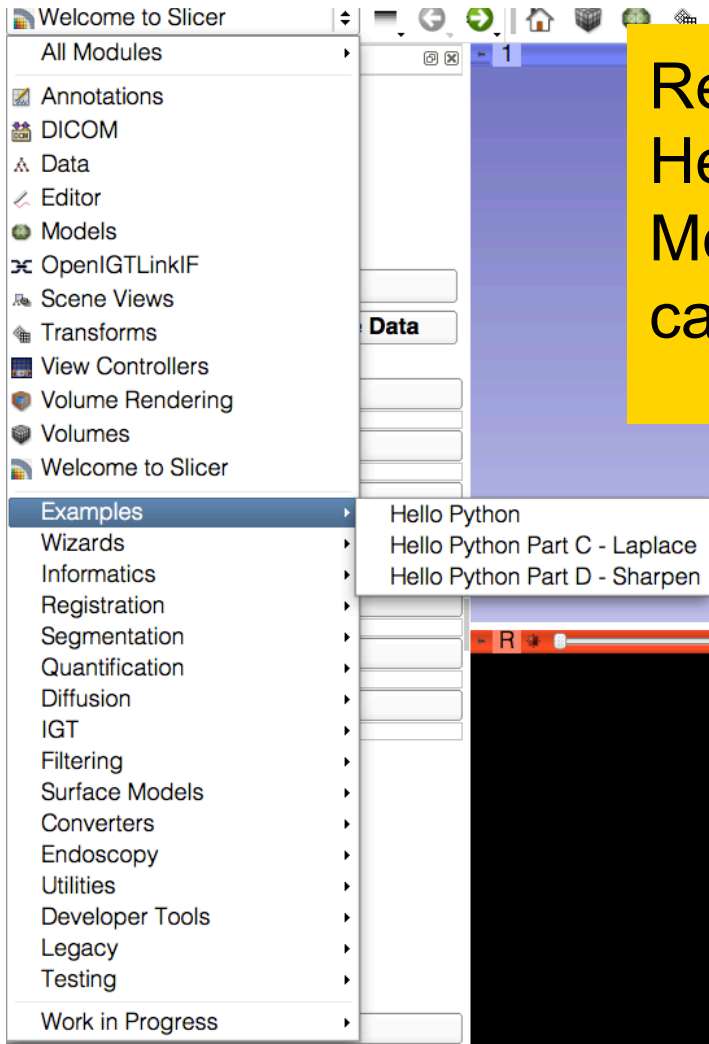
Integrating HelloPython

Add the path to the directory containing HelloPython.py (when selecting the directory, the HelloWorld.py file itself will not be displayed)





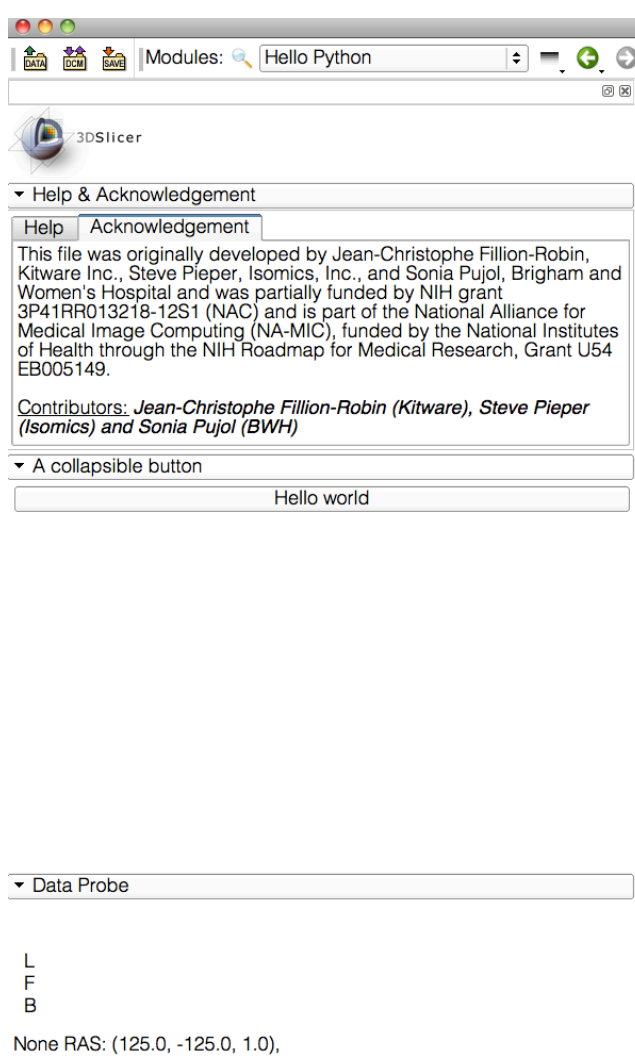
HelloPython in Slicer



Restart Slicer when prompted.
Hello Python is now in the
Modules Menu, under the
category **Examples**

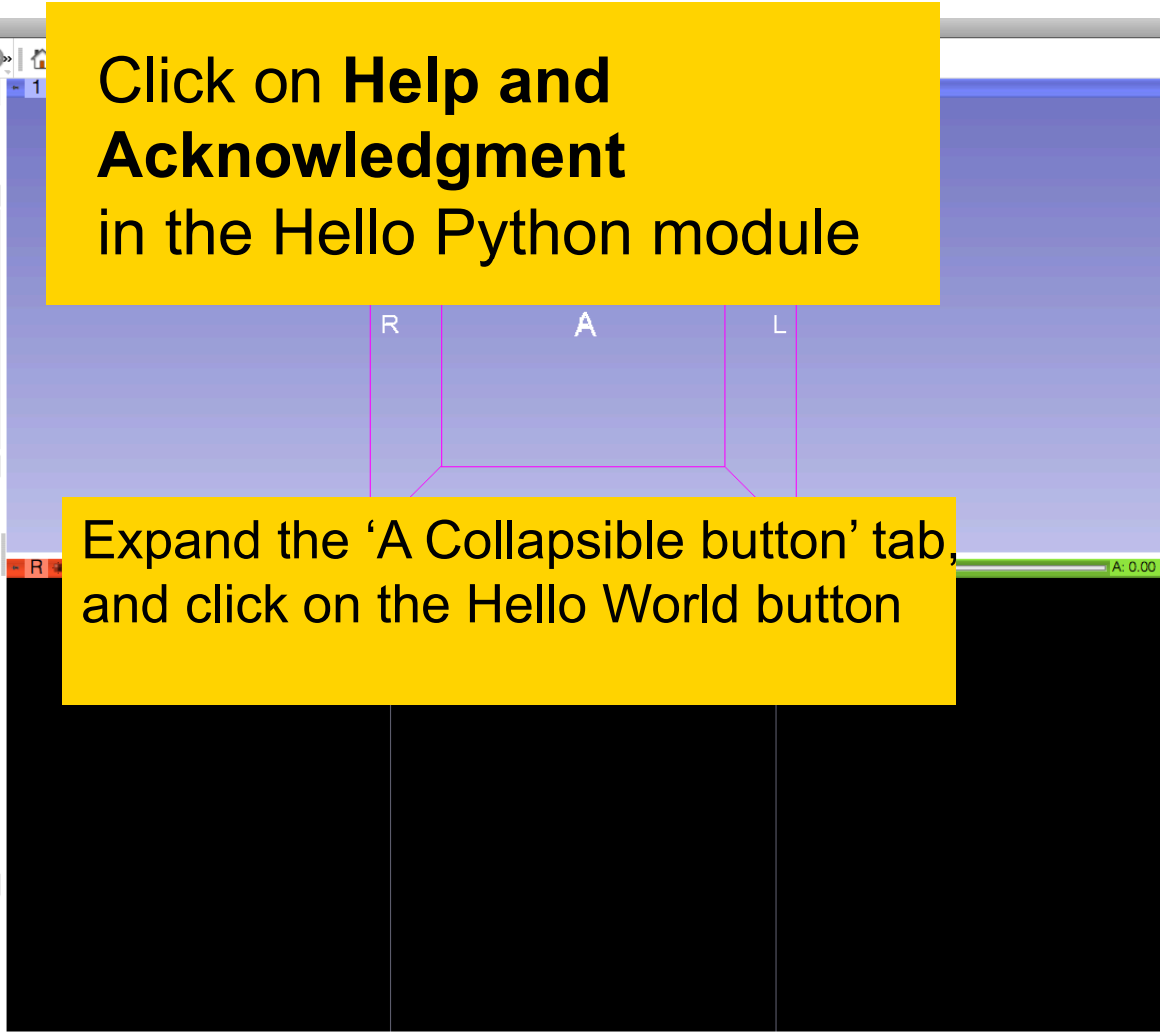


HelloPython in Slicer

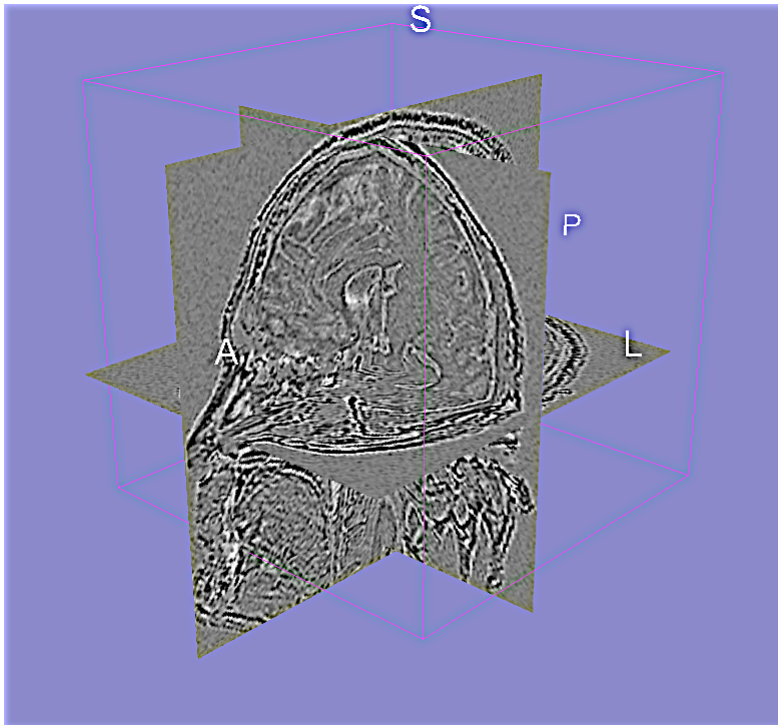


Click on **Help and Acknowledgment** in the Hello Python module

Expand the 'A Collapsible button' tab, and click on the Hello World button



L
F
B
None RAS: (125.0, -125.0, 1.0),



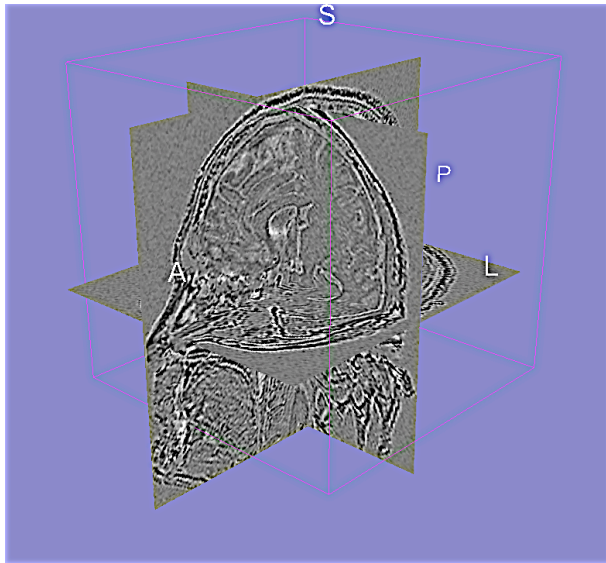
Part C:

Implementing the Laplace* Operator

*named after Pierre-Simon, Marquis de Laplace (1749-1827)

Overview

The goal of this section is to build an image analysis module that implements a Laplacian filter on volume data



- Use qMRML widgets: widgets that automatically track the state of the Slicer MRML scene
- Use VTK filters to manipulate volume data

HelloLaplace.py

```
File Edit Tools Syntax Buffers Window Help
HelloPython.py (~/.Dropbox/0_work/HelloPython/HelloPython) - GVIM

from __main__ import vtk, qt, ctk, slicer

# HelloPython
#

class HelloPython:
    def __init__(self, parent):
        parent.title = "Hello Python"
        parent.categories = ["Examples"]
        parent.dependencies = []
        parent.contributors = ["Jean-Christophe Fillion-Robin (Kitware),
            Steve Pieper (Isomics), Sonia Pujol (BWH)"] # replace with "Firstname Lastname (Org)"
        parent.helpText = """
            Example of scripted loadable extension for the HelloPython tutorial.
            """
        parent.acknowledgementText = """
            This file was originally developed by Jean-Christophe Fillion-Robin, Kitware Inc.,
            Steve Pieper, Isomics, Inc., and Sonia Pujol, Brigham and Women's Hospital and was
            partially funded by NIH grant 3P41RR013218-12S1 (NAC) and is part of the National Alliance
            for Medical Image Computing (NA-MIC), funded by the National Institutes of Health through the
            NIH Roadmap for Medical Research, Grant U54 EB005149. """ # replace with organization, grant and thanks.
        self.parent = parent

#
# qHelloPythonWidget
#

class HelloPythonWidget:
    def __init__(self, parent = None):
        if not parent:
            self.parent = slicer.qMRMLWidget()
            self.parent.setLayout(qt.QVBoxLayout())
            self.parent.setMRMLScene(slicer.mrmlScene)
        else:
            self.parent = parent
        self.layout = self.parent.layout()
        if not parent:

# HelloWorldButton
helloWorldButton = qt.QPushButton("Hello world")
helloWorldButton.setToolTip("Print 'Hello world' in standard output.")
dummyFormLayout.addWidget(helloWorldButton)
helloWorldButton.connect('clicked(bool)', self.onHelloWorldButtonClicked)

# Add vertical spacer
self.layout.addStretch(1)

# Set local var as instance attribute
self.helloWorldButton = helloWorldButton

def onHelloWorldButtonClicked(self):
    print "Hello World !"
    qt.QMessageBox.information(slicer.util.mainWindow(), 'Slicer Python', 'Hello World!')

HelloPython.py 22,8 All
```

Open the file HelloLaplace.py located in the directory HelloPython

Module GUI (Part 1)

```
def setup(self):
    # Collapsible button
    self.laplaceCollapsibleButton = ctk.ctkCollapsibleButton()
    self.laplaceCollapsibleButton.text = "Laplace Operator"
    self.layout.addWidget(self.laplaceCollapsibleButton)

    # Layout within the laplace collapsible button
    self.laplaceFormLayout = qt.QFormLayout(self.laplaceCollapsibleButton)

    # the volume selectors
    self.inputFrame = qt.QFrame(self.laplaceCollapsibleButton)
    self.inputFrame.setLayout(qt.QHBoxLayout())
    self.laplaceFormLayout.addWidget(self.inputFrame)
    self.inputSelector = qt.QLabel("Input Volume: ", self.inputFrame)
    self.inputFrame.layout().addWidget(self.inputSelector)
    self.inputSelector = slicer.qMRMLNodeComboBox(self.inputFrame)
    self.inputSelector.nodeTypeNames = ( "vtkMRMLScalarVolumeNode", "" )
    self.inputSelector.addEnabled = False
    self.inputSelector.removeEnabled = False
    self.inputSelector.setMRMLScene( slicer.mrmlScene )
    self.inputFrame.layout().addWidget(self.inputSelector)
```

This code is
provided in
the template

Module GUI (Part 2)

```
self.outputFrame = qt.QFrame(self.laplaceCollapsibleButton)
self.outputFrame.setLayout(qt.QHBoxLayout())
self.laplaceFormLayout.addWidget(self.outputFrame)
self.outputSelector = qt.QLabel("Output Volume: ", self.outputFrame)
self.outputFrame.layout().addWidget(self.outputSelector)
self.outputSelector = slicer.qMRMLNodeComboBox(self.outputFrame)
self.outputSelector.nodeTypeNames = ( "vtkMRMLScalarVolumeNode", "" )
self.outputSelector.setMRMLScene( slicer.mrmlScene )
self.outputFrame.layout().addWidget(self.outputSelector)

# Apply button
laplaceButton = qt.QPushButton("Apply Laplace")
laplaceButton.setToolTip("Run the Laplace Operator.")
self.laplaceFormLayout.addWidget(laplaceButton)
laplaceButton.connect('clicked(bool)', self.onApply)

# Add vertical spacer
self.layout.addStretch(1)

# Set local var as instance attribute
self.laplaceButton = laplaceButton
```

This code is
provided in
the template

In More Detail

- **CTK** is a Qt Add-On Library with many useful widgets, particularly for visualization and medical imaging see <http://commontk.org>
- **Qt Widgets, Layouts**, and Options are well documented at <http://qt.nokia.com>
- **qMRMLNodeComboBox** is a powerful slicer widget that monitors the scene and allows you to select/create nodes of specified types (*example: here we use Volumes = vtkMRMLScalarVolumeNode*)



Processing Code

```
def onApply(self):
    inputVolume = self.inputSelector.currentNode()
    outputVolume = self.outputSelector.currentNode()
    if not (inputVolume and outputVolume):
        qt.QMessageBox.critical(slicer.util.mainWindow(),
                                'Laplace', 'Input and output volumes are required for Laplacian')
        return
    laplacian = vtk.vtkImageLaplacian()
    laplacian.SetInput(inputVolume.GetImageData())
    laplacian.SetDimensionality(3)
    laplacian.GetOutput().Update()
    ijkToRAS = vtk.vtkMatrix4x4()
    inputVolume.GetIJKToRASMatrix(ijkToRAS)
    outputVolume.SetIJKToRASMatrix(ijkToRAS)
    outputVolume.SetAndObserveImageData(laplacian.GetOutput())
    # make the output volume appear in all the slice views
    selectionNode = slicer.app.applicationLogic().GetSelectionNode()
    selectionNode.SetReferenceActiveVolumeID(outputVolume.GetID())
    slicer.app.applicationLogic().PropagateVolumeSelection(0)
```

Add this
code

In More Detail

- **vtkImageLaplacian** is a `vtkImageAlgorithm` operates on `vtkImageData` (see <http://vtk.org>)
- **vtkMRMLScalarVolumeNode** is a Slicer MRML class that contains `vtkImageData`, plus orientation information `ijkToRAS` matrix (see http://www.slicer.org/slicerWiki/index.php/Coordinate_systems)

In More Detail (Continued)

Global **licer** package gives python access to:

1- GUI (via **licer.app**)

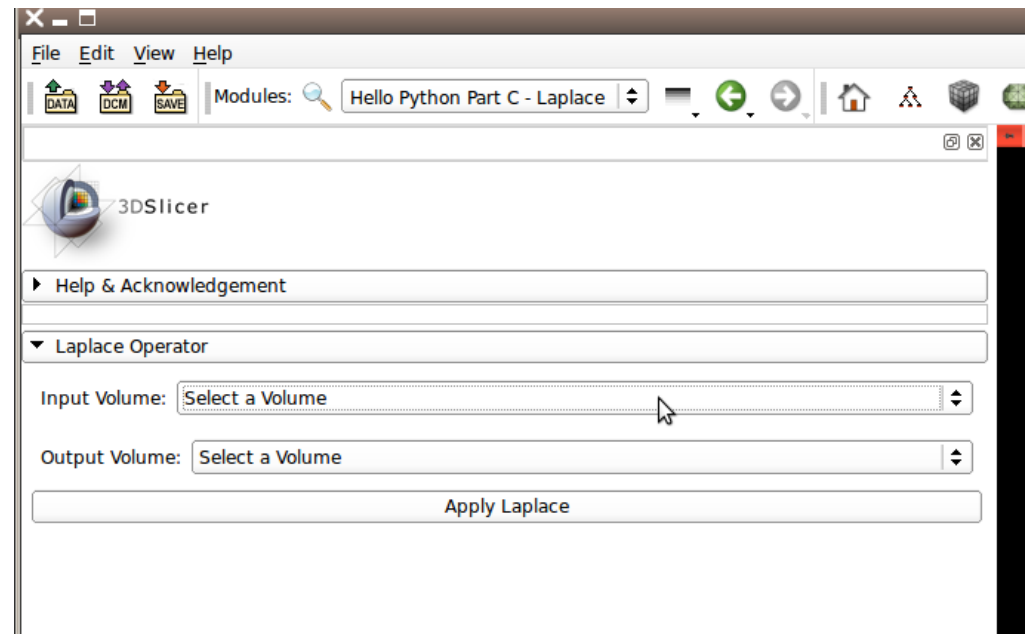
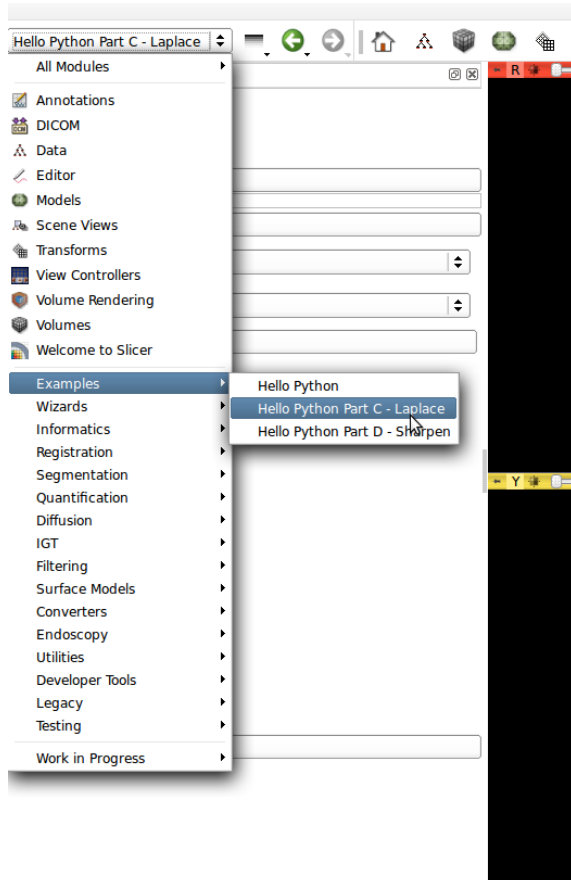
2- modules (via **licer.modules**)

3- data (via **licer.mrmlScene**)

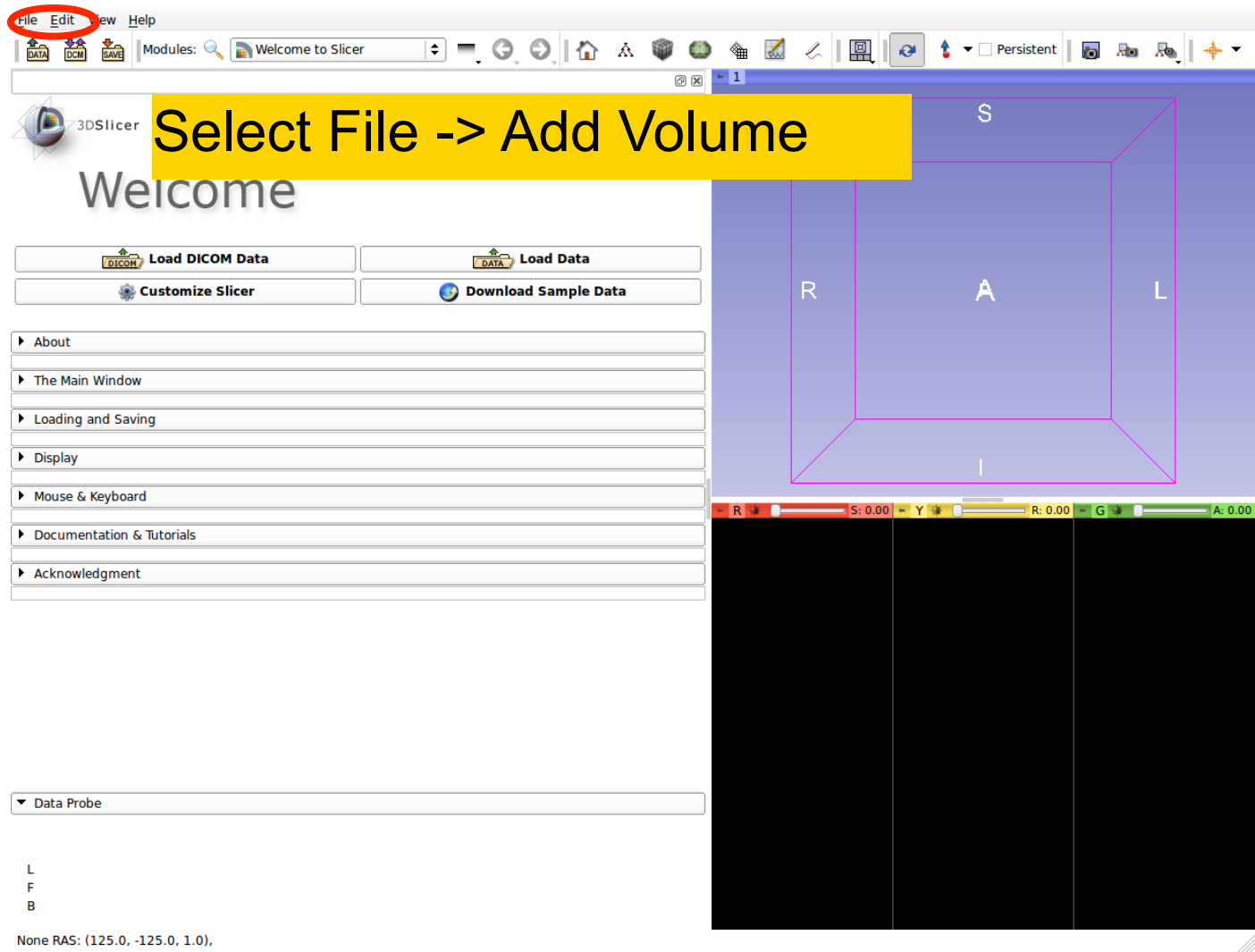
licer.app.applicationLogic() provides helper utilities for manipulating Slicer state

Go To Laplace Module

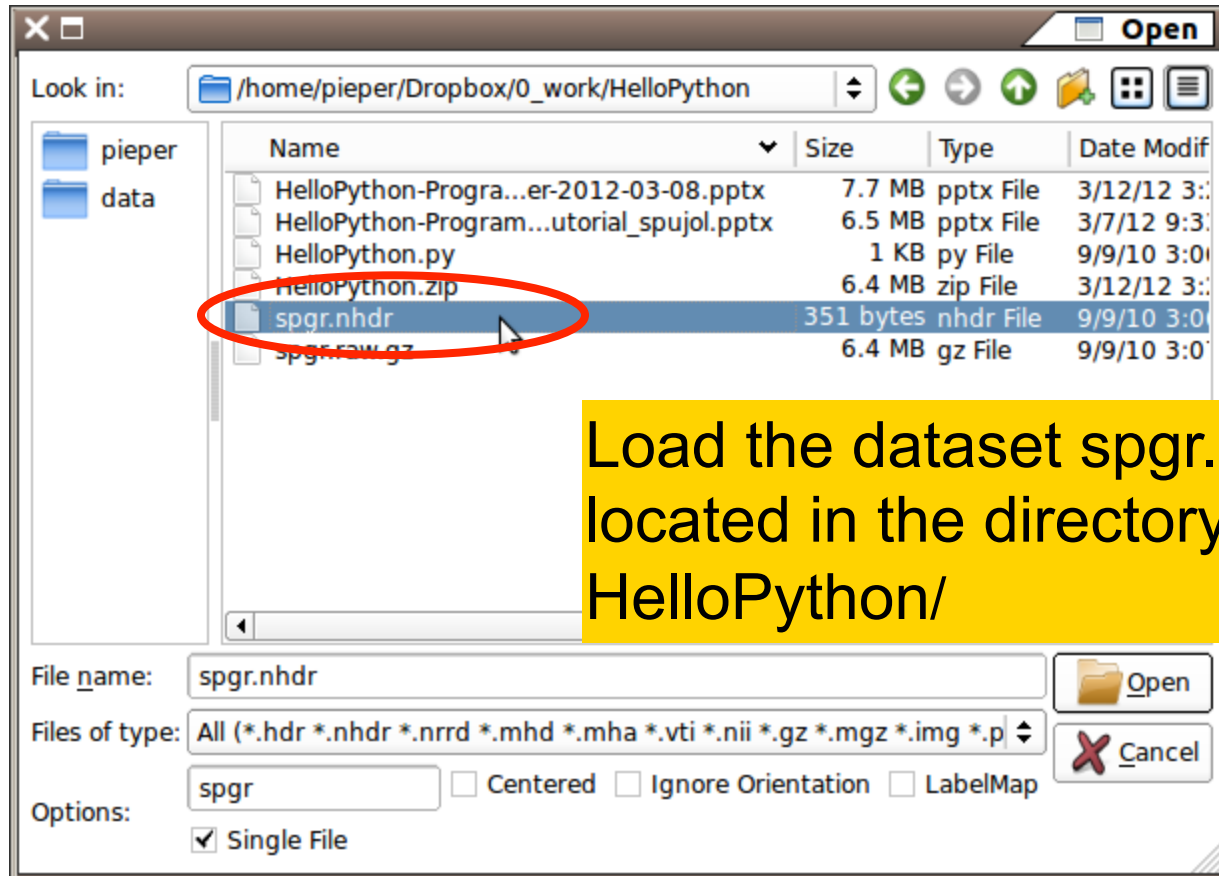
Re-start Slicer and select module. Note that combobox is empty



Add Volume Dialog



Add spgr.nhdr



After Adding Volume

▼ Laplace Operator

Input Volume: spgr

(1) Note that Input Volume combobox autoselected new volume

Output Volume: spgr

Create new Volume
Delete current Volume

(2) Create new volume for output

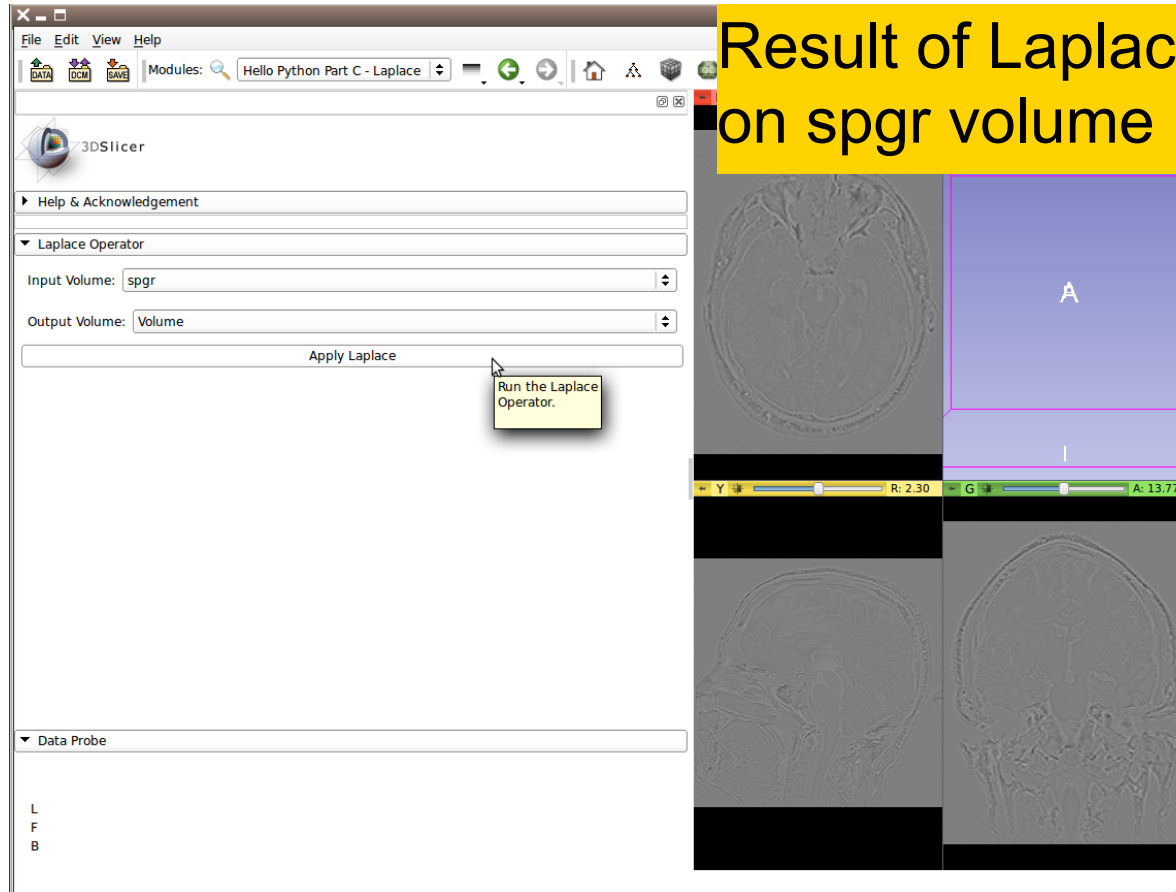
Output Volume: Volume

Apply Laplace

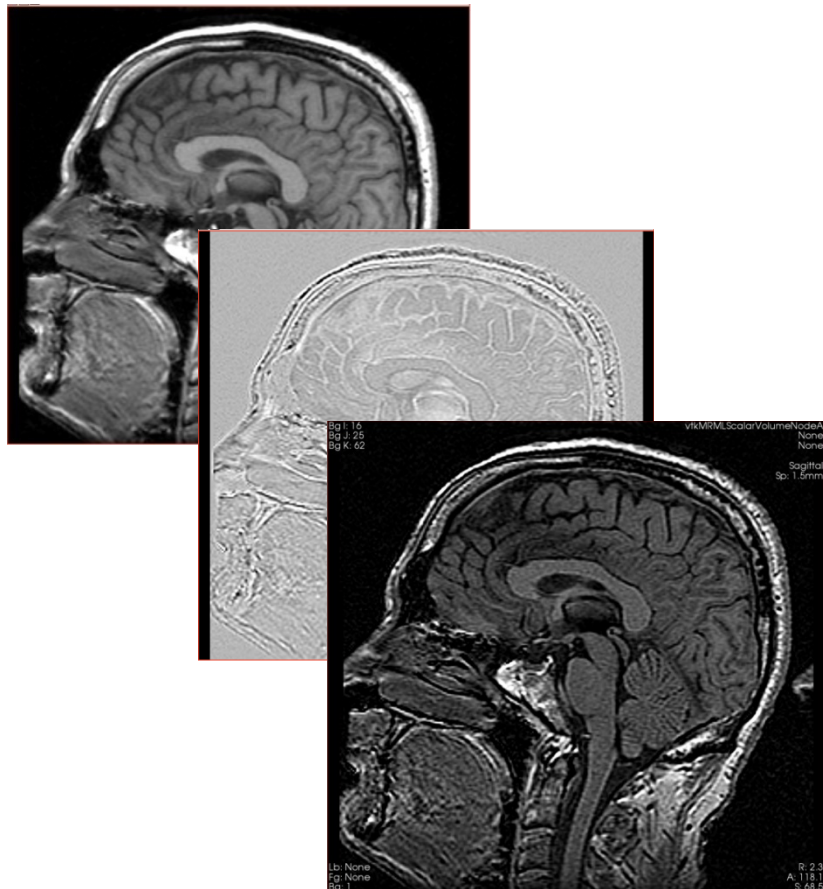
Run the Laplace Operator.

(3) Run the module

Laplace Module

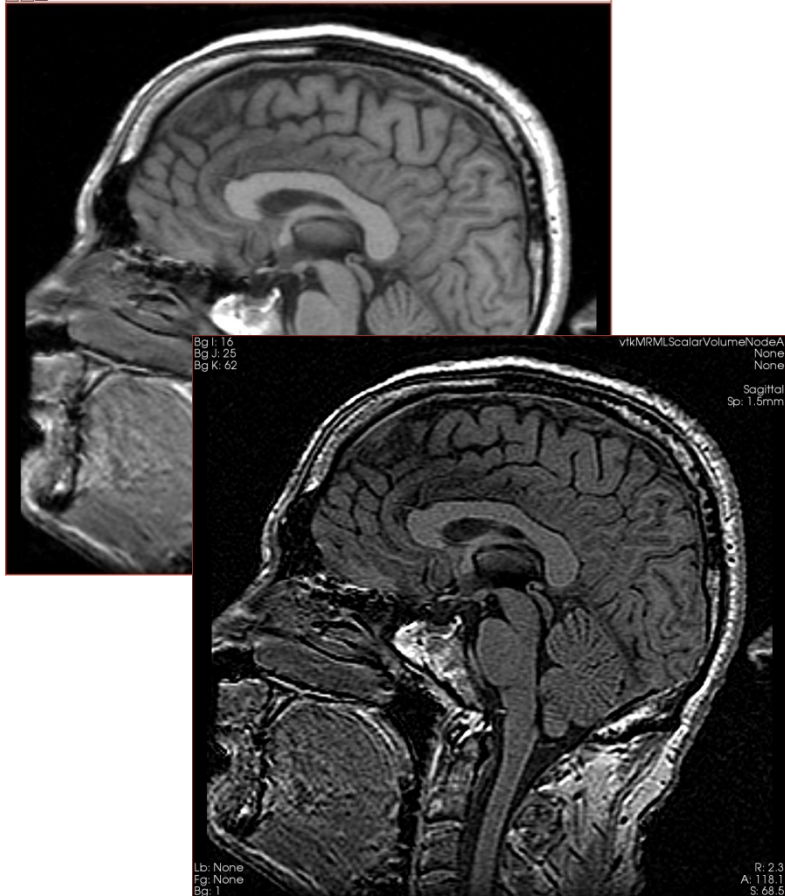


Result of Laplace Operator on spgr volume



Part D: Image Sharpening with the Laplace Operator

Overview



- The goal of this section is to add a processing option for image sharpening. We'll implement this operation using the existing Slicer Command Line Module 'Subtract Scalar Volumes'

HelloSharpen.py

```
File Edit Tools Syntax Buffers Window Help
HelloPython.py (~/.Dropbox/0_work/HelloPython/HelloPython) - GVIM

from __main__ import vtk, qt, ctk, slicer

# HelloPython
#

class HelloPython:
    def __init__(self, parent):
        parent.title = "Hello Python"
        parent.categories = ["Examples"]
        parent.dependencies = []
        parent.contributors = ["Jean-Christophe Fillion-Robin (Kitware),
                               Steve Pieper (Isomics), Sonia Pujol (BWH)"] # replace with "Firstname Lastname (Org)"
        parent.helpText = """
        Example of scripted loadable extension for the HelloPython tutorial.
        """
        parent.acknowledgementText = """
        This file was originally developed by Jean-Christophe Fillion-Robin, Kitware Inc.,
        Steve Pieper, Isomics, Inc., and Sonia Pujol, Brigham and Women's Hospital and was
        partially funded by NIH grant 3P41RR013218-12S1 (NAC) and is part of the National Alliance
        for Medical Image Computing (NA-MIC), funded by the National Institutes of Health through the
        NIH Roadmap for Medical Research, Grant U54 EB005149.""" # replace with organization, grant and thanks.
        self.parent = parent

#
# qHelloPythonWidget
#

class HelloPythonWidget:
    def __init__(self, parent = None):
        if not parent:
            self.parent = slicer.qMRMLWidget()
            self.parent.setLayout(qt.QVBoxLayout())
            self.parent.setMRMLScene(slicer.mrmlScene)
        else:
            self.parent = parent
            self.layout = self.parent.layout()
            if not parent:

# HelloWorldButton
helloWorldButton = qt.QPushButton("Hello world")
helloWorldButton.setToolTip("Print 'Hello world' in standard output.")
dummyFormLayout.addWidget(helloWorldButton)
helloWorldButton.connect('clicked(bool)', self.onHelloWorldButtonClicked)

# Add vertical spacer
self.layout.addStretch(1)

# Set local var as instance attribute
self.helloWorldButton = helloWorldButton

def onHelloWorldButtonClicked(self):
    print "Hello World !"
    qt.QMessageBox.information(slicer.util.mainWindow(), 'Slicer Python', 'Hello World!')

~
~
HelloPython.py 22,8 All
```

Open the file HelloSharpen.py
located in the directory HelloPython

Add to Module GUI

Add this
Text in
section A

```
...
self.outputSelector.setMRMLScene( slicer.mrmlScene )
self.outputFrame.layout().addWidget(self.outputSelector)

self.sharpen = qt.QCheckBox("Sharpen", self.laplaceCollapsibleButton)
self.sharpen.setToolTip = "When checked, subtract laplacian from input volume"
self.sharpen.checked = True
self.laplaceFormLayout.addWidget(self.sharpen)

# Apply button
laplaceButton = qt.QPushButton("Apply")
laplaceButton.setToolTip = "Run the Laplace or Sharpen Operator."
...
```



Add to Processing Code

Add this
Text in
section B

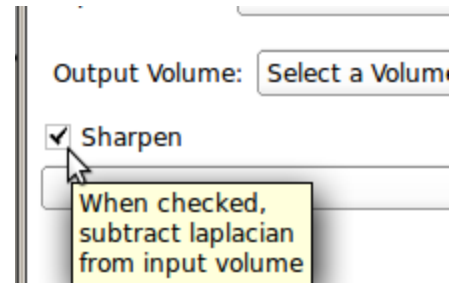
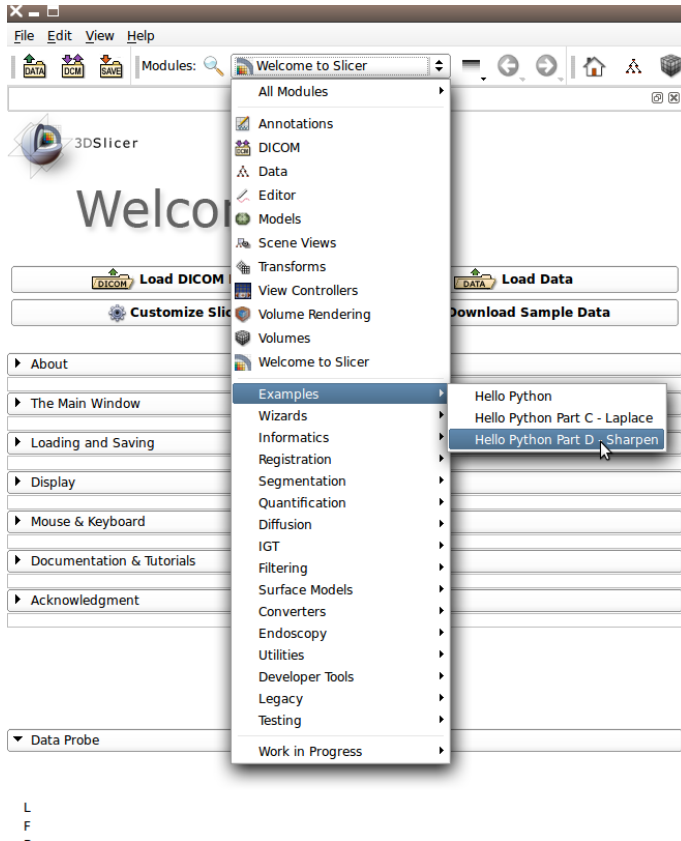
```
...
outputVolume.SetAndObserveImageData(laplacian.GetOutput())
# optionally subtract laplacian from original image
if self.sharpen.checked:
    parameters = {}
    parameters['inputVolume1'] = inputVolume.GetID()
    parameters['inputVolume2'] = outputVolume.GetID()
    parameters['outputVolume'] = outputVolume.GetID()
    slicer.cli.run( slicer.modules.subtractscalarvolumes, None,
parameters, wait_for_completion=True )
# make the output volume appear in all the slice views
selectionNode = slicer.app.applicationLogic().GetSelectionNode()
selectionNode.SetReferenceActiveVolumeID(outputVolume.GetID
())
slicer.app.applicationLogic().PropagateVolumeSelection(0)
```

In More Detail

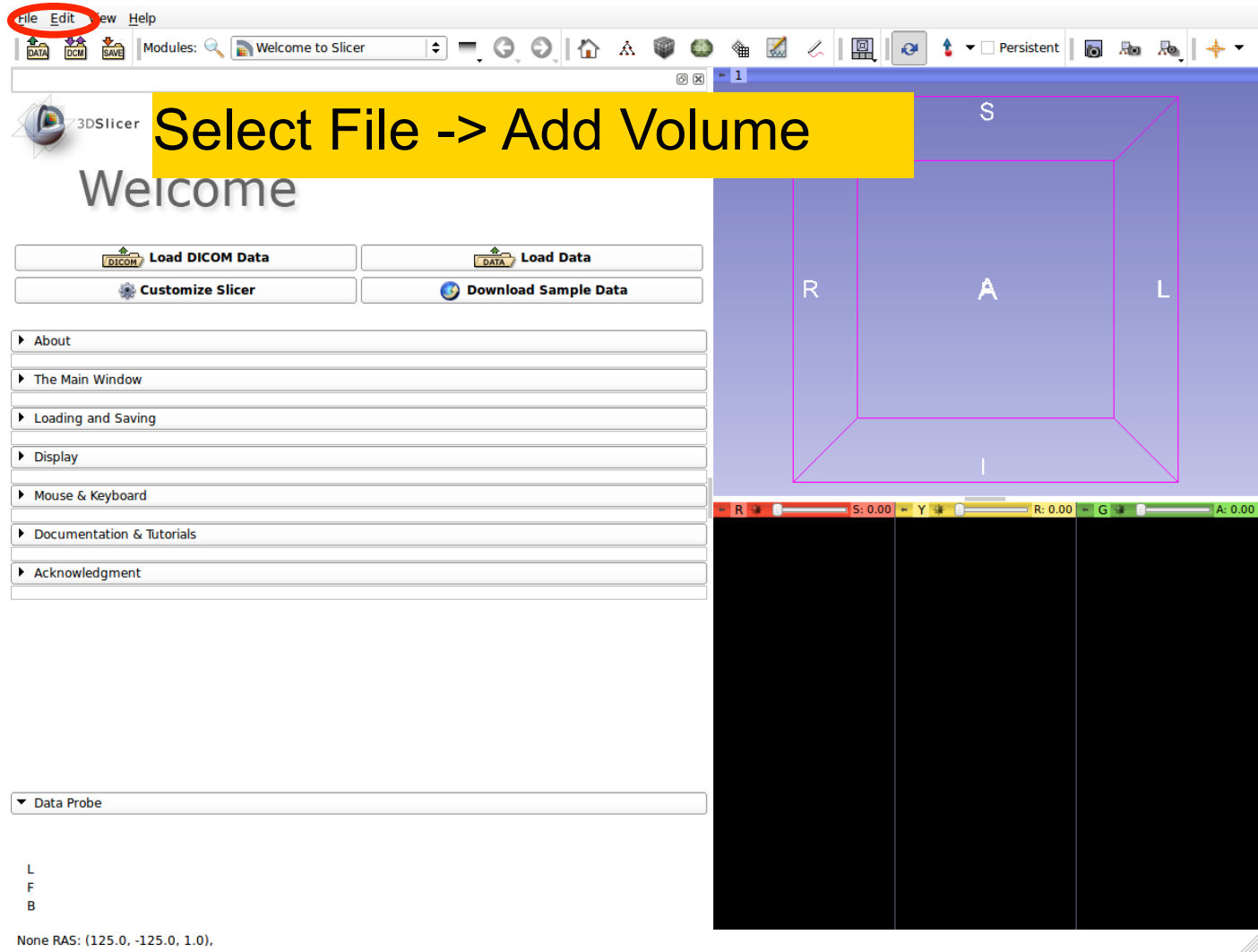
- **slicer.cli** gives access to Command Line Interface (CLI) modules
- CLI modules allow packaging of arbitrary C++ code (often ITK-based) into slicer with automatically generated GUI and python wrapping

Go To Sharpen Module

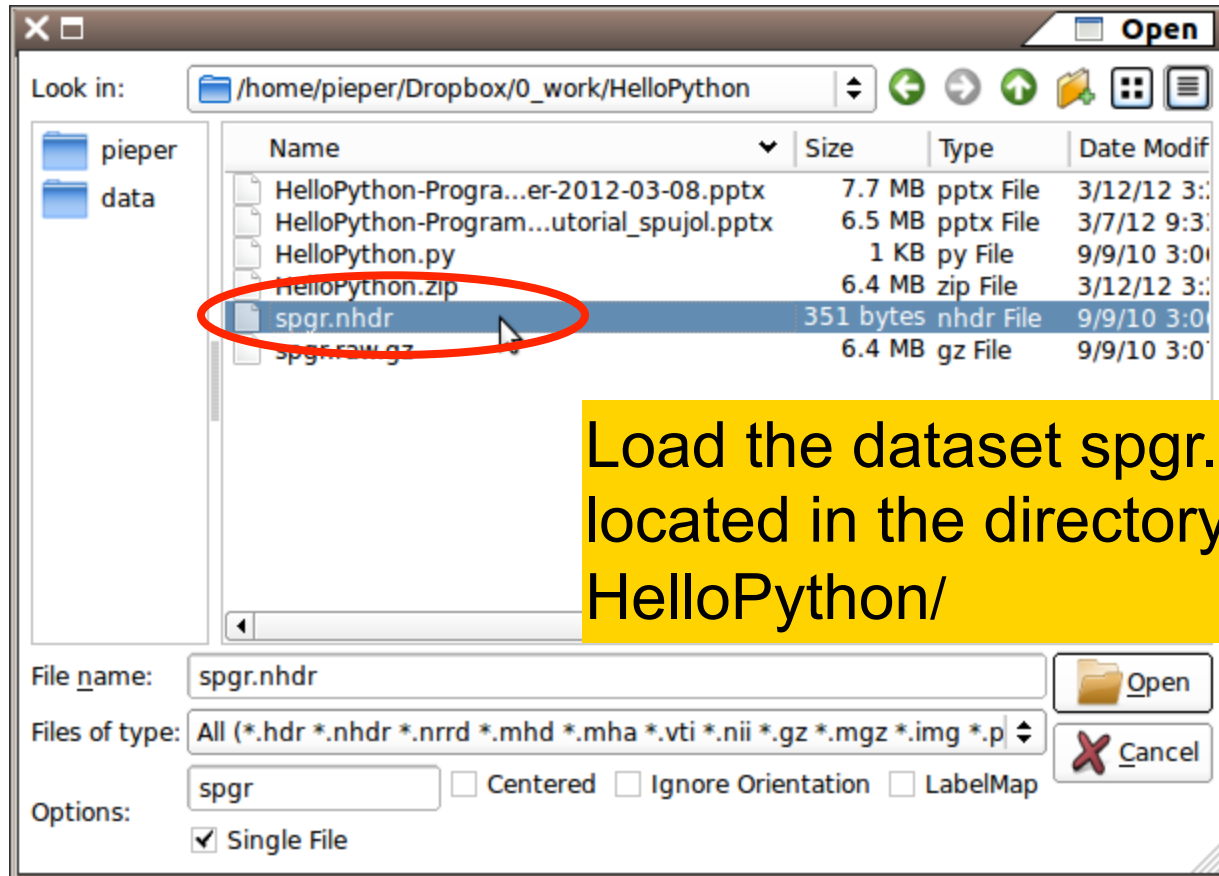
Re-start Slicer and select module. Note the new sharpen check box



Add Volume Dialog



Add spgr.nhdr



After Adding Volume

▼ Laplace Operator

Input Volume: spgr

(1) Note that Input Volume combobox autoselected new volume

Output Volume: spgr

Create new Volume
Delete current Volume

(2) Create new volume for output

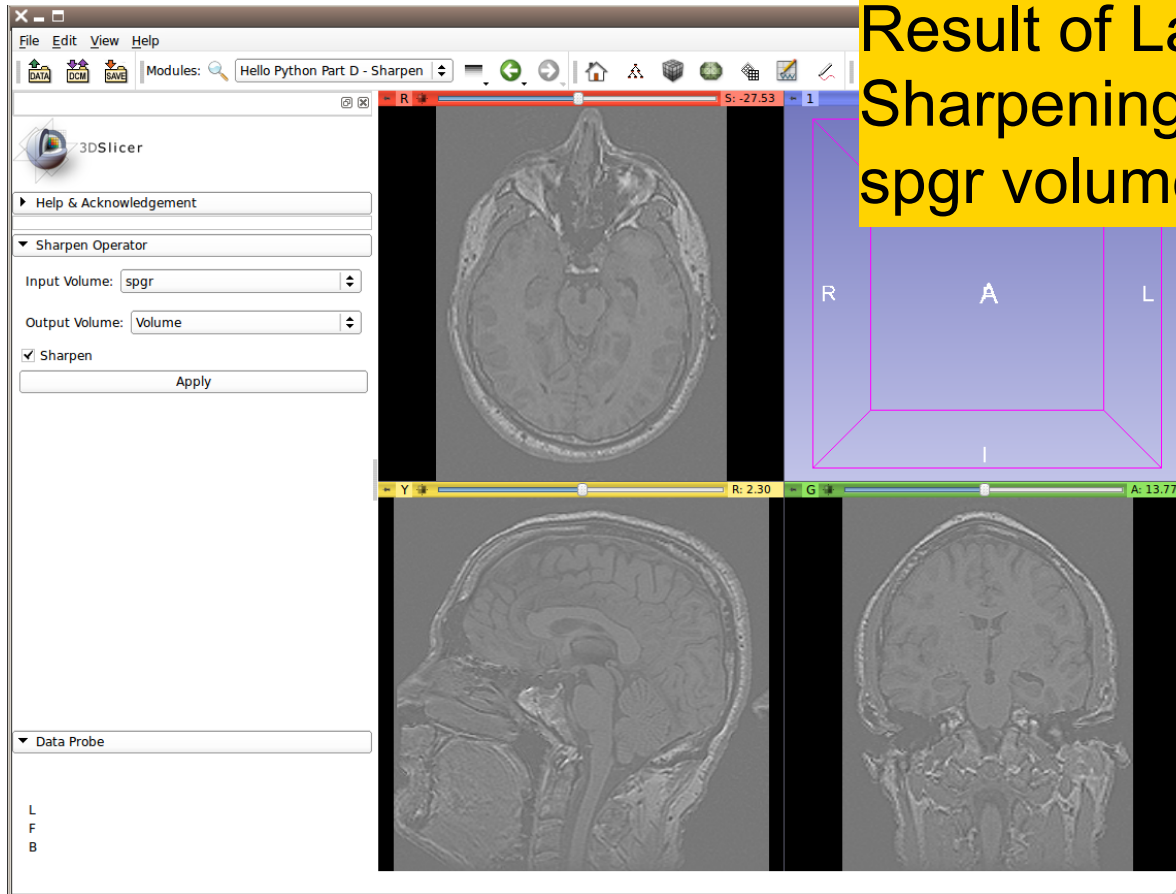
Sharpen

Apply

(3) Run the module in Sharpen mode

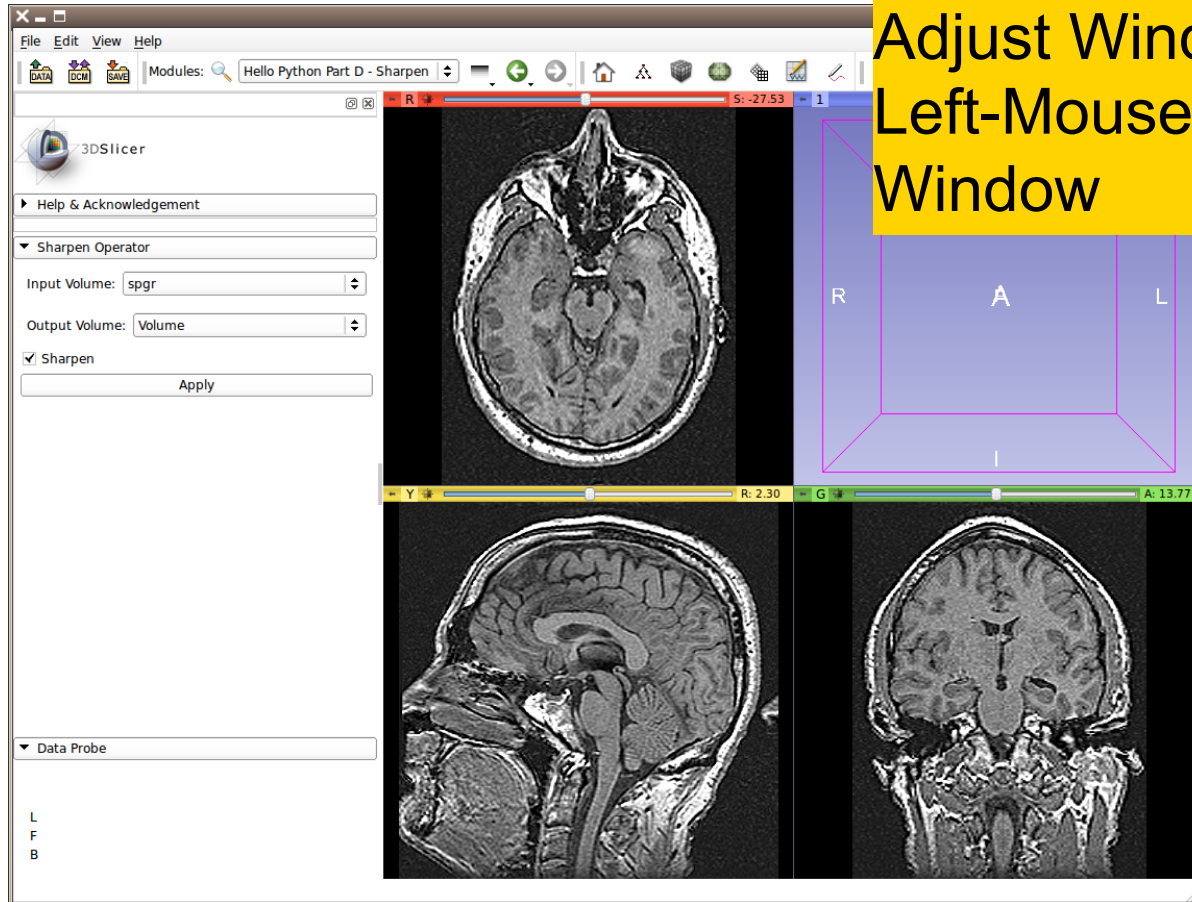
Run the Laplace or Sharpen Operator.

Sharpen Module



Result of Laplacian
Sharpening Operator on
spgr volume

Sharpen Module



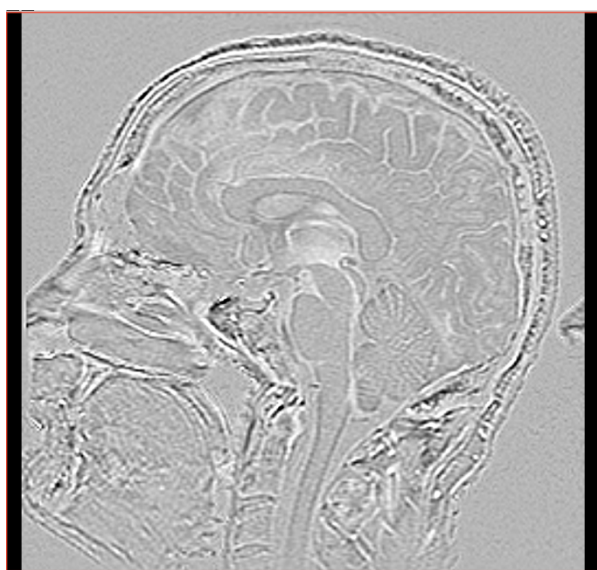
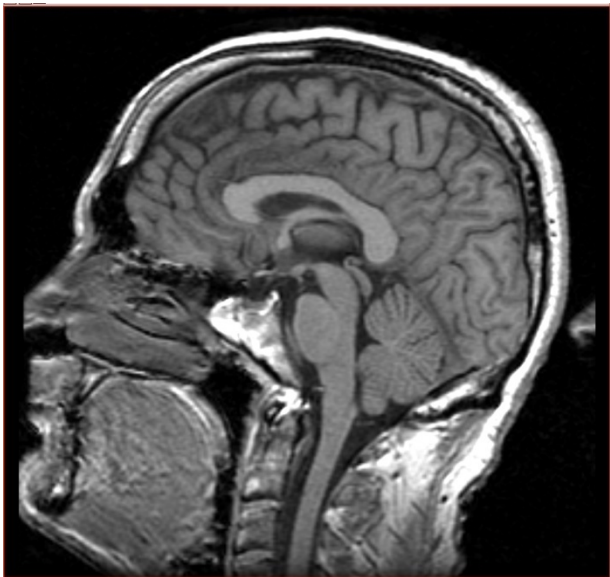
Adjust Window/Level with
Left-Mouse-Drag in Slice
Window

Image Sharpening

original

Laplacian

Laplacian filtered



Going Further

- Explore numpy for numerical array manipulation
- Review Endoscopy Module for interactive data exploration using MRML and VTK
- See the Editor Module for interactive segmentation examples
- Explore SimpleITK for image processing using ITK

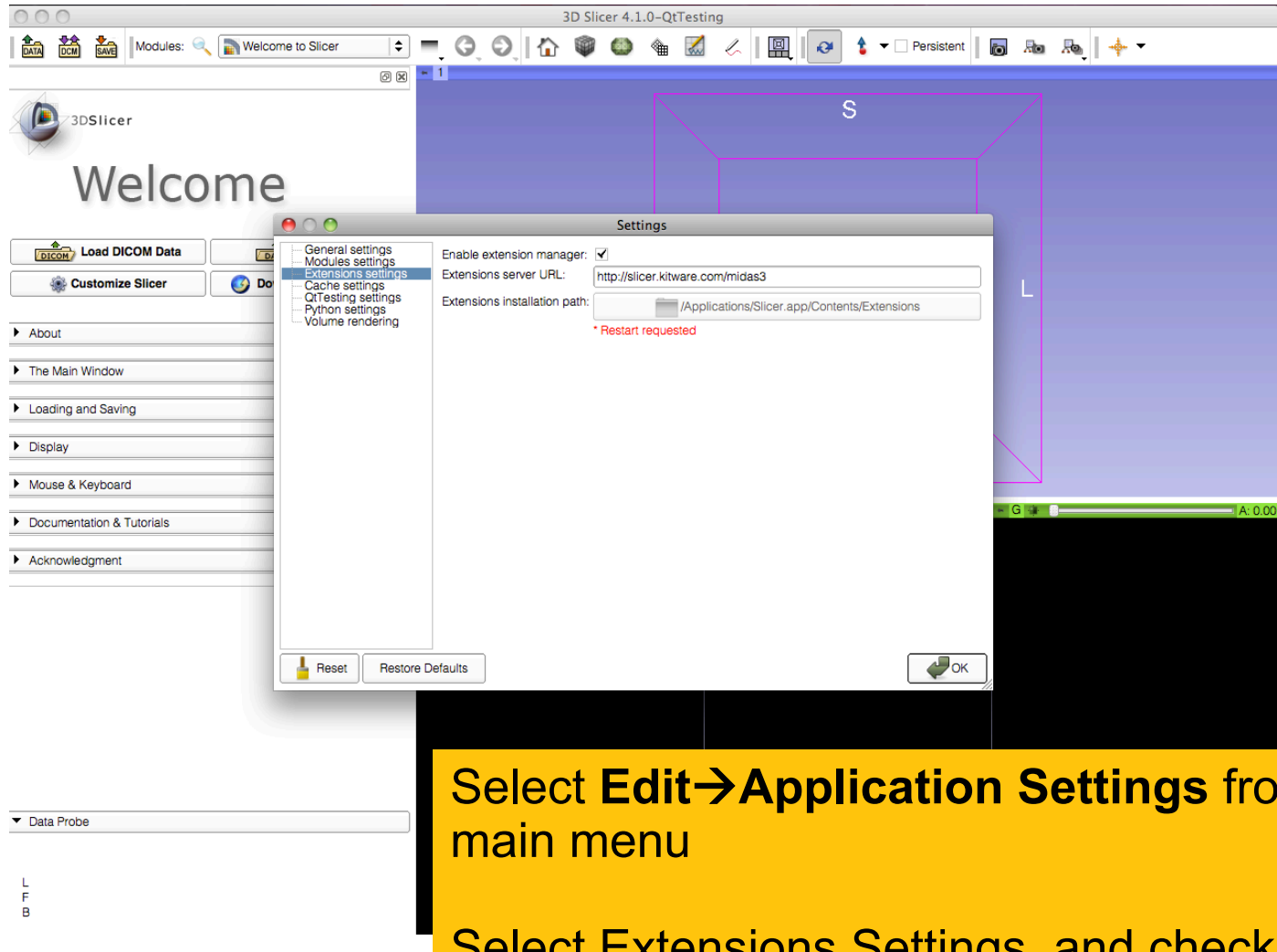


Conclusion

This course demonstrated how to program custom behavior in Slicer with Python



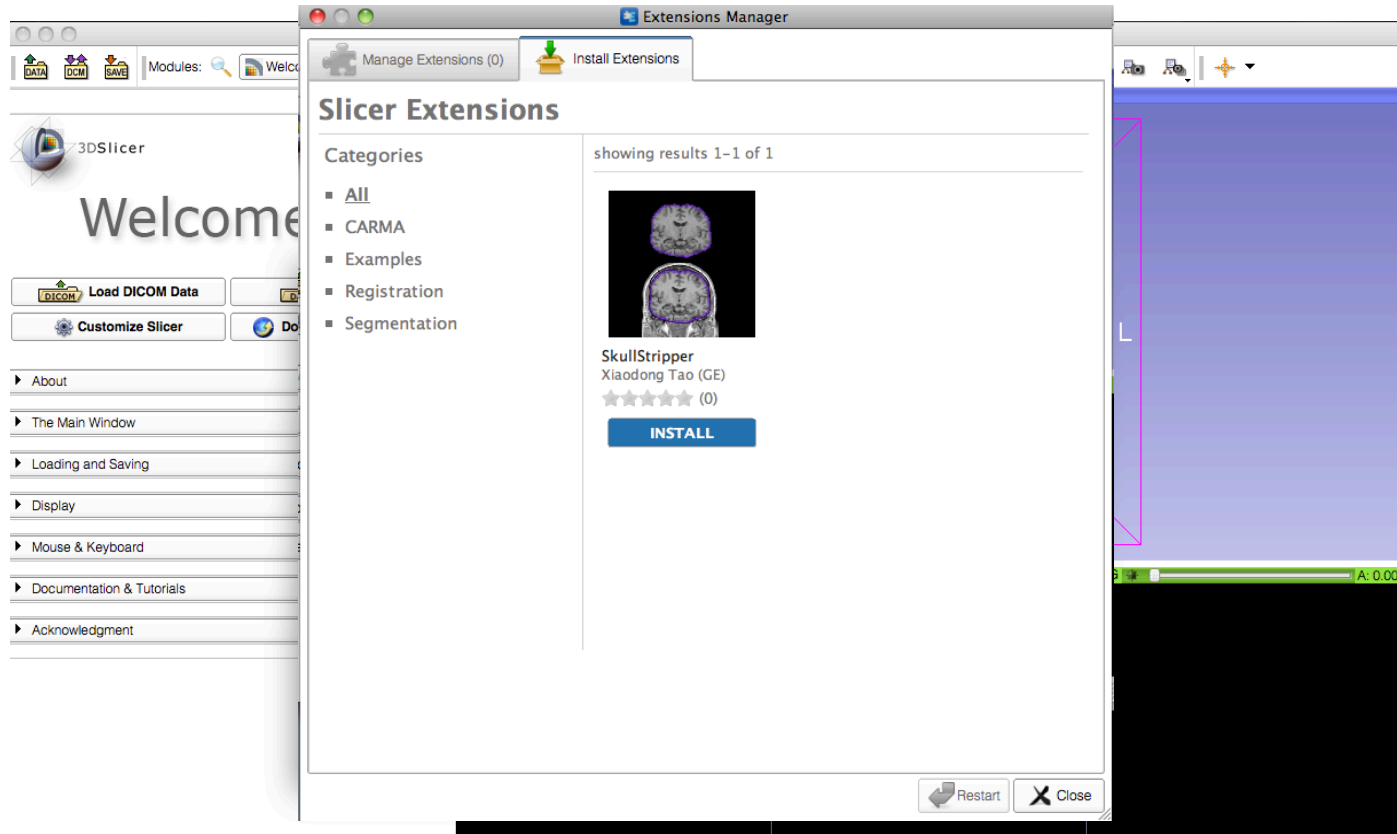
Going Further: Slicer Extensions



Select **Edit**→**Application Settings** from the main menu

Select Extensions Settings, and check the box '**Enable extension manager**'

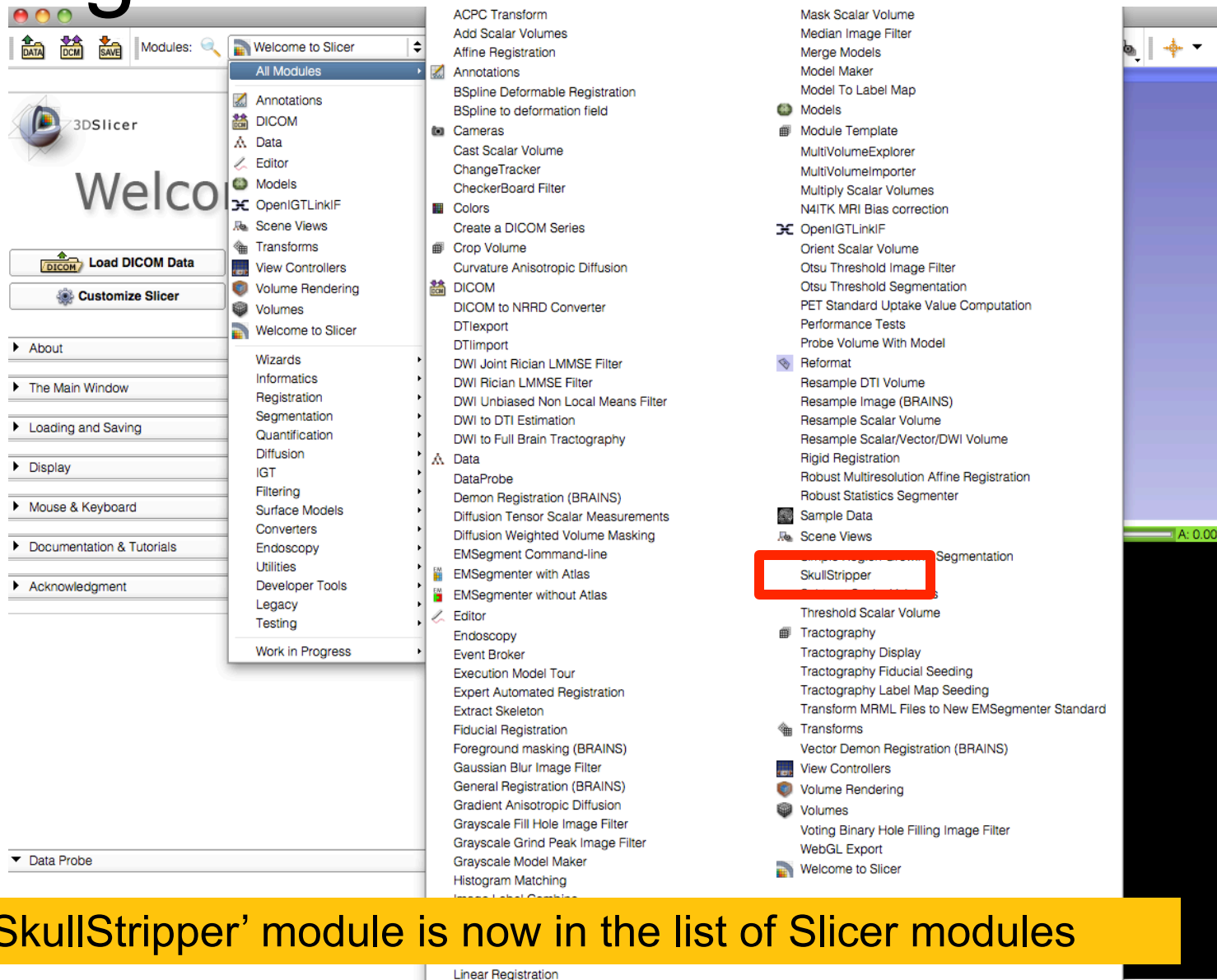
Going Further: Slicer Extensions



Restart Slicer and select **View**→**Extension Manager** from the main menu

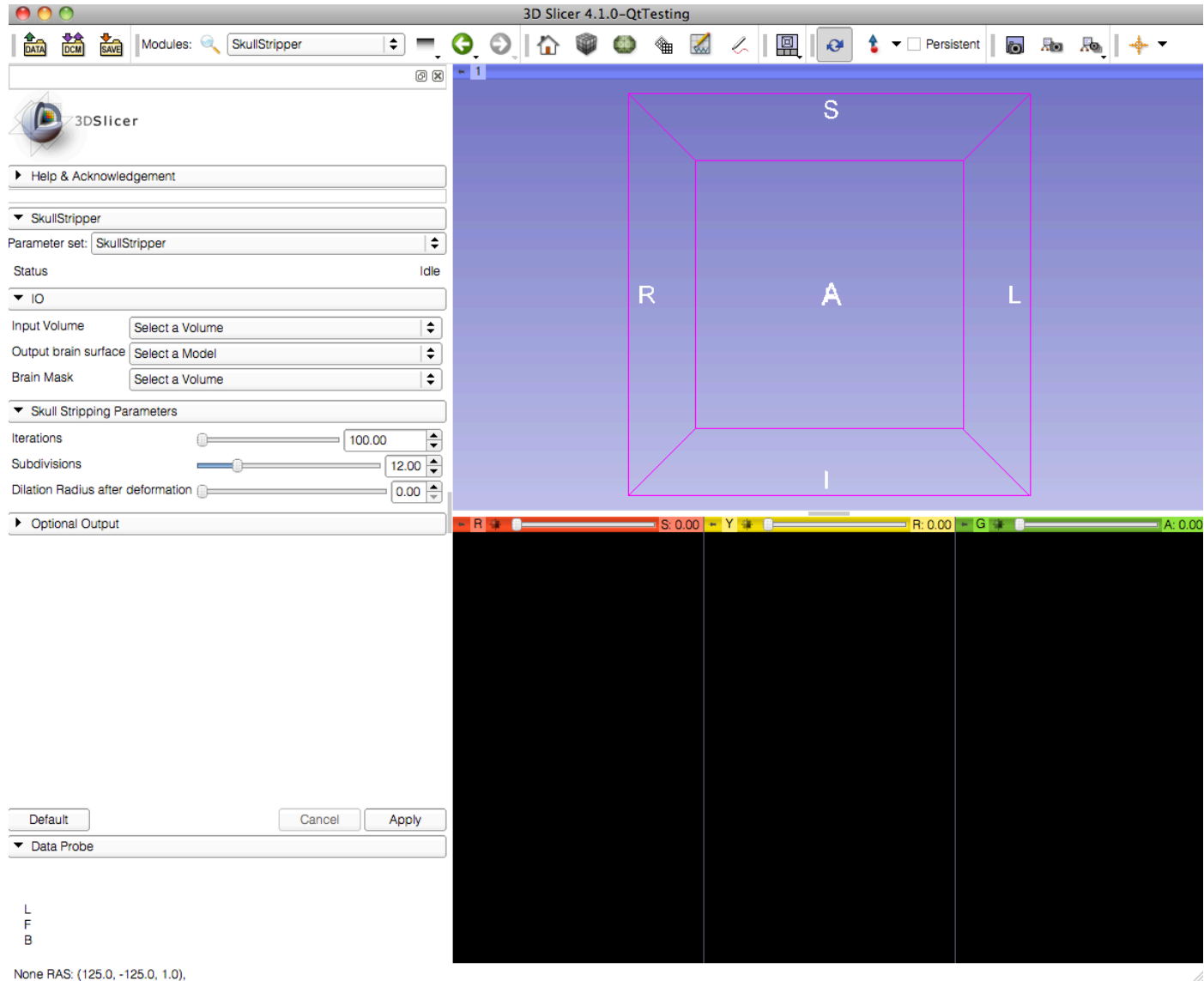
Click on **Install** to install the **SkullStripper** extension, and click on **Restart**

Going Further: Slicer Extensions



The 'SkullStripper' module is now in the list of Slicer modules

Going Further: Slicer Extensions





Acknowledgments



National Alliance for Medical Image Computing

NIH U54EB005149



Neuroimage Analysis Center

NIH P41RR013218

Questions and Comments

spujol@bwh.harvard.edu

