



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

# Programming in Slicer4

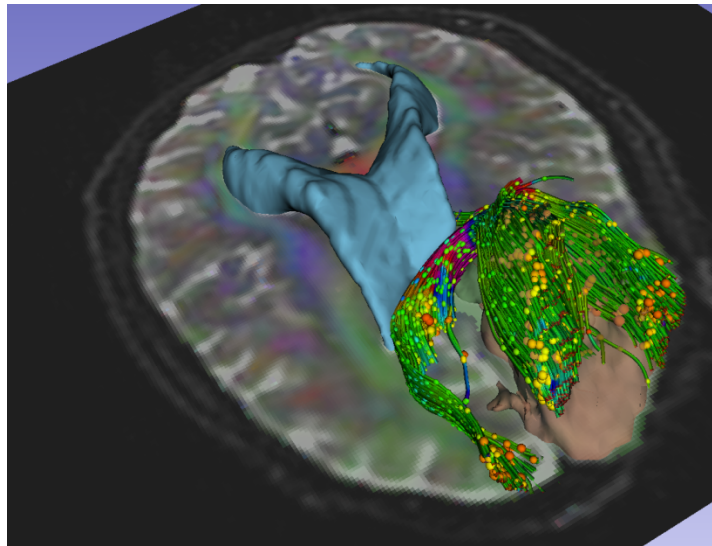
Sonia Pujol, Ph.D.  
Surgical Planning Laboratory

Steve Pieper, Ph.D.  
Isomics Inc.

# The NA-MIC Kit



# 3D Slicer version 4 (Slicer4)



- 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

# 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, custom code

# Slicer4 Scripted Module

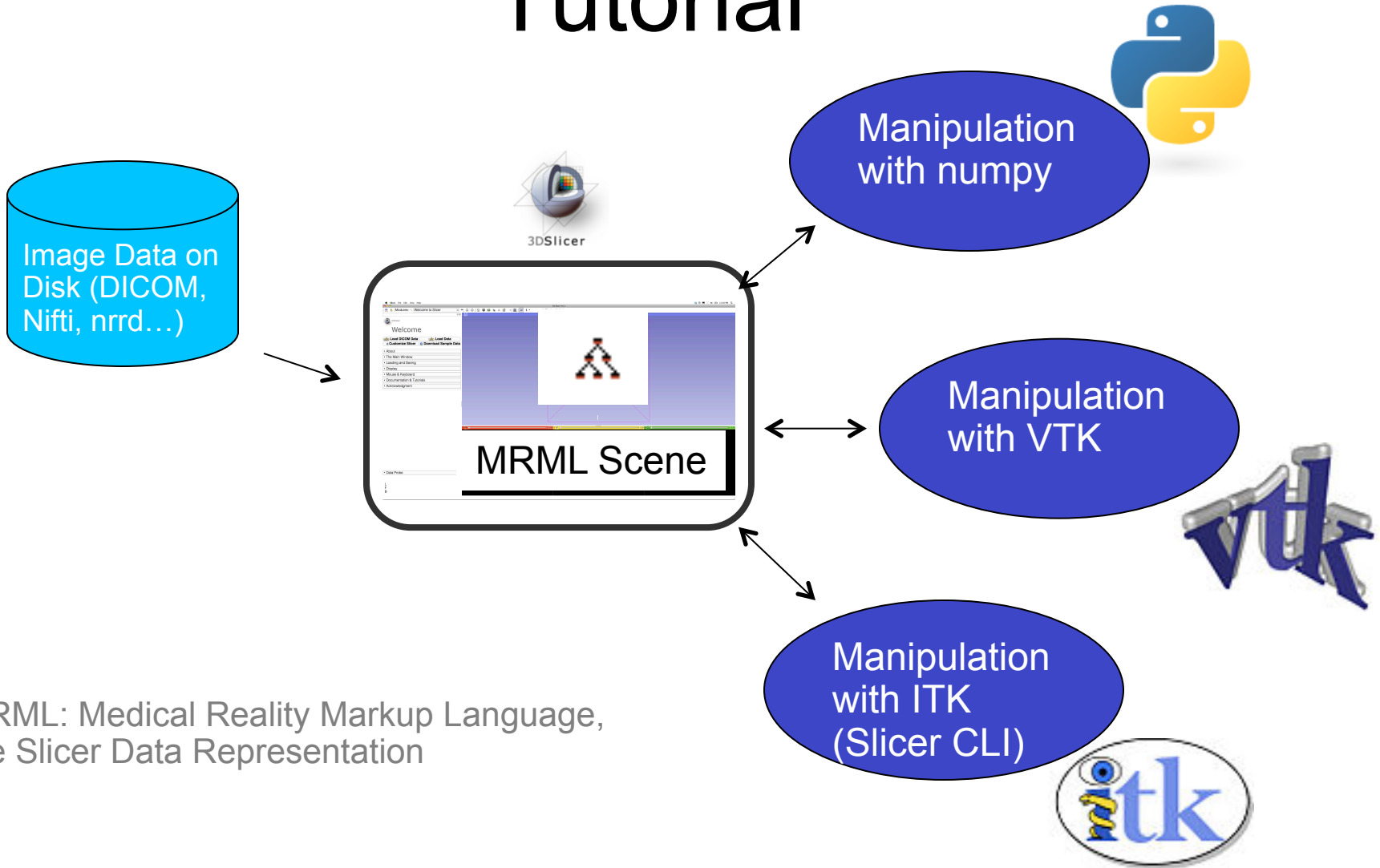
- Python scripted modules allow more interactive functionalities (eg 'Flythrough' in Endoscopy module)
- GUI based on Qt libraries accessed via Python



# 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>

# Processing Examples in this Tutorial



MRML: Medical Reality Markup Language, the Slicer Data Representation

# Prerequisites

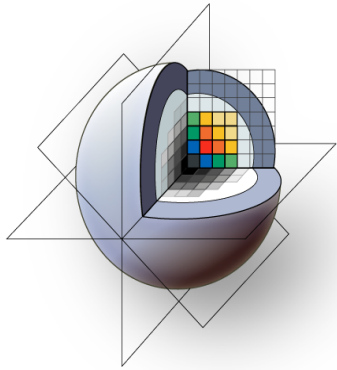
- This course supposes that you have taken the tutorial: “Slicer4 Data Loading and Visualization”- Sonia Pujol Ph.D.
- The tutorial is available on the Slicer4 101 compendium:  
<http://www.slicer.org/slicerWiki/index.php/Training/4.0>
- Programming experience is required, and some familiarity with Python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





# Course 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

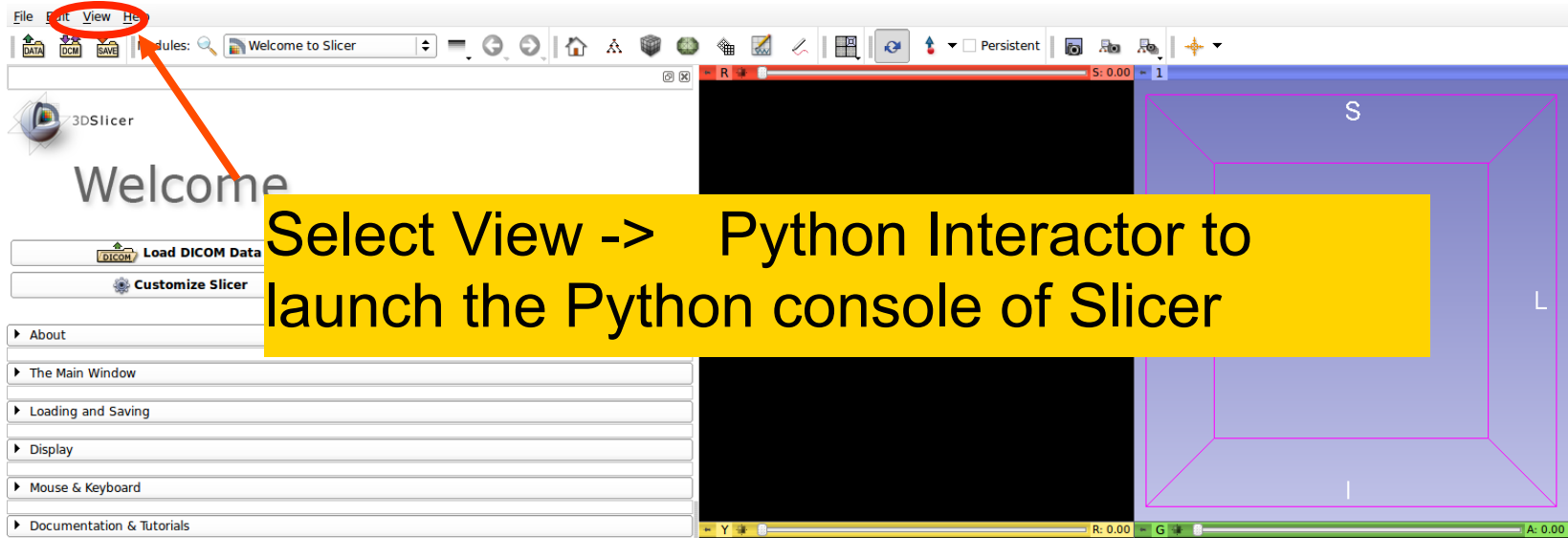


# Part A: EXPLORING SLICER VIA PYTHON

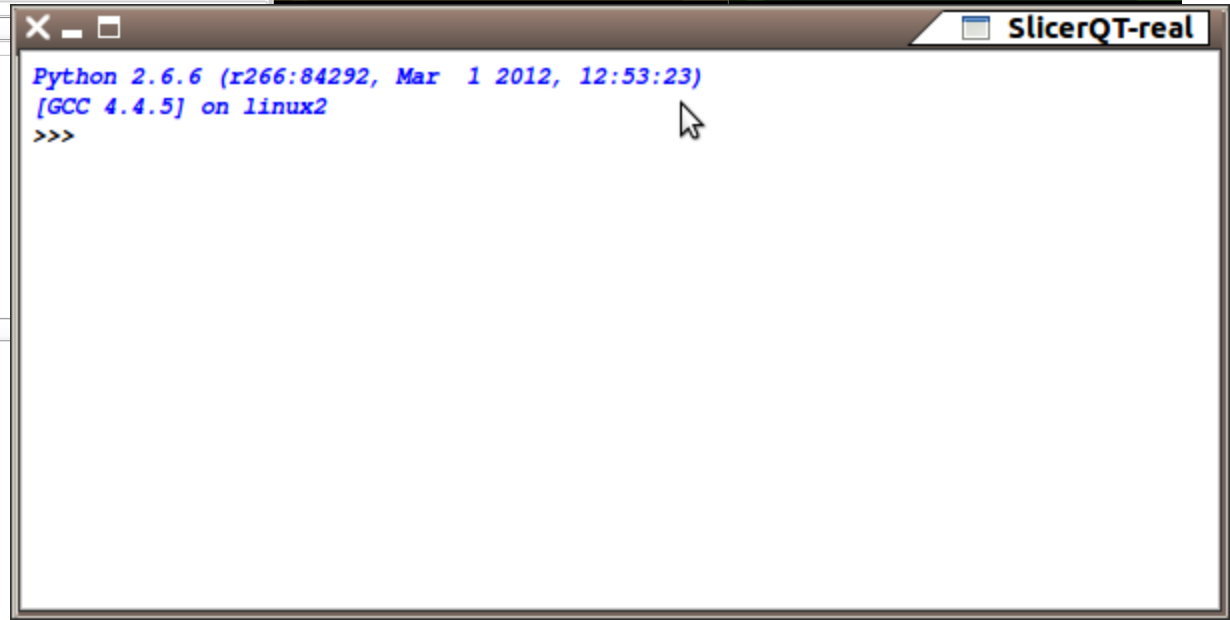
# Python in Slicer

- Slicer 4 includes python 2.6.6 and a rich standard library
  - *Included:* numpy, vtk, ctk, PythonQt, most of standard python library
  - *Not Included:* scipy, matplotlib, ipython... and some other popular packages that we have found difficult to package for distribution

# Python Console in Slicer

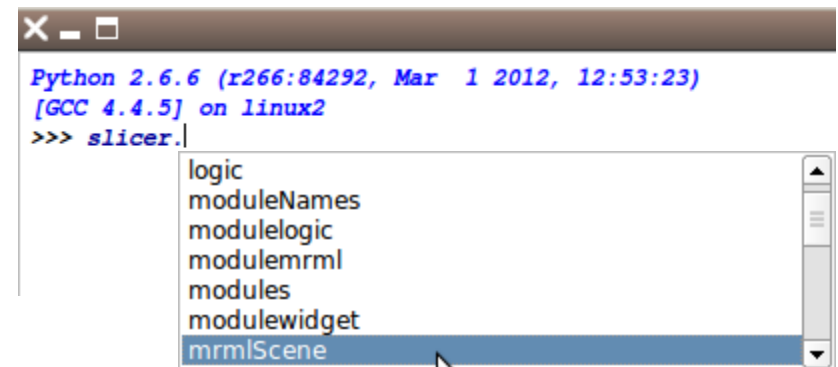


Select View -> Python Interactor to launch the Python console of Slicer



# General Python Console Features

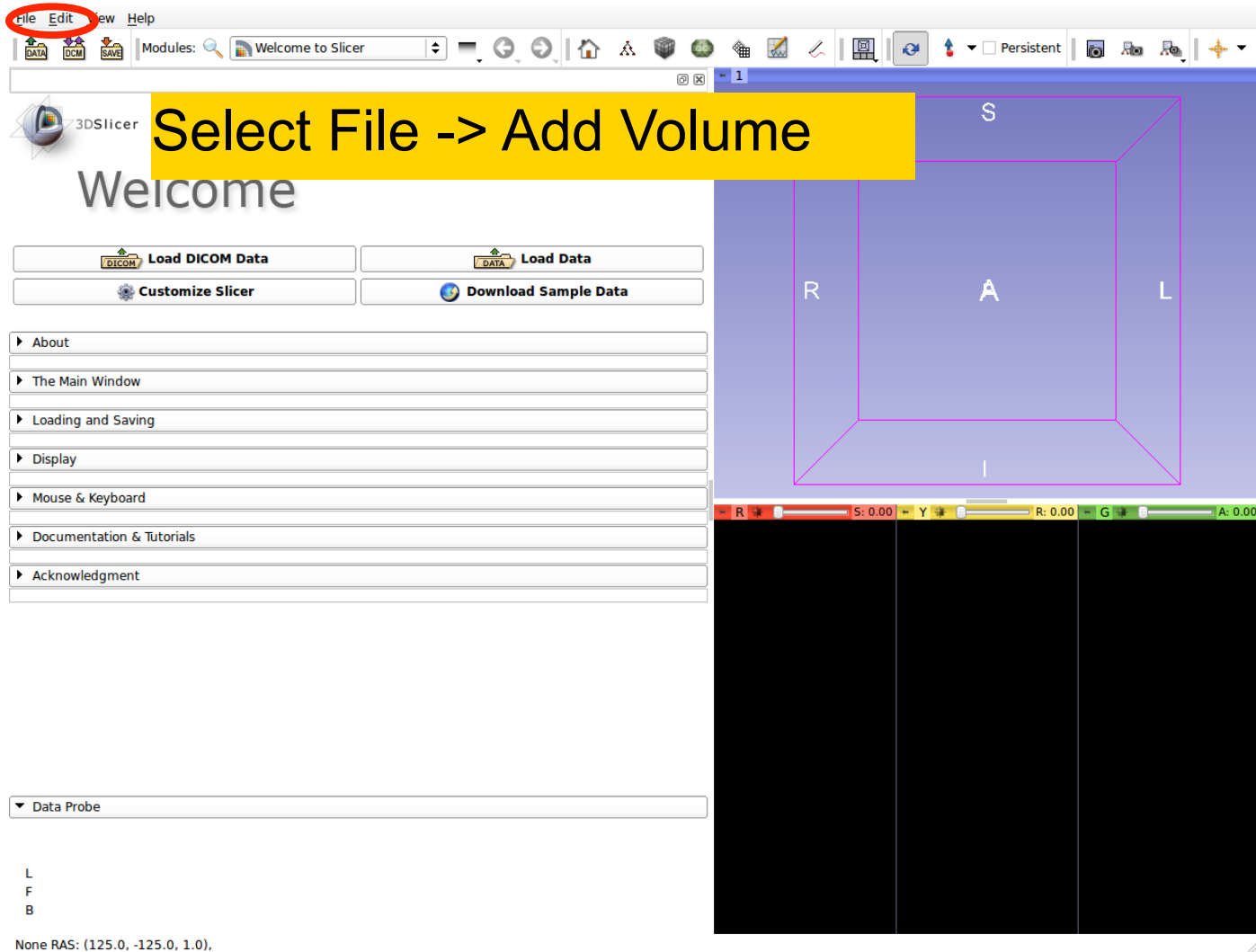
- Command Line Editing:
  - Left/Right Arrow Keys, Home, End
  - Delete (Control-D)
- Input History
  - Up/Down Arrow Keys
- Command Completion
  - Tab Key



```
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
```

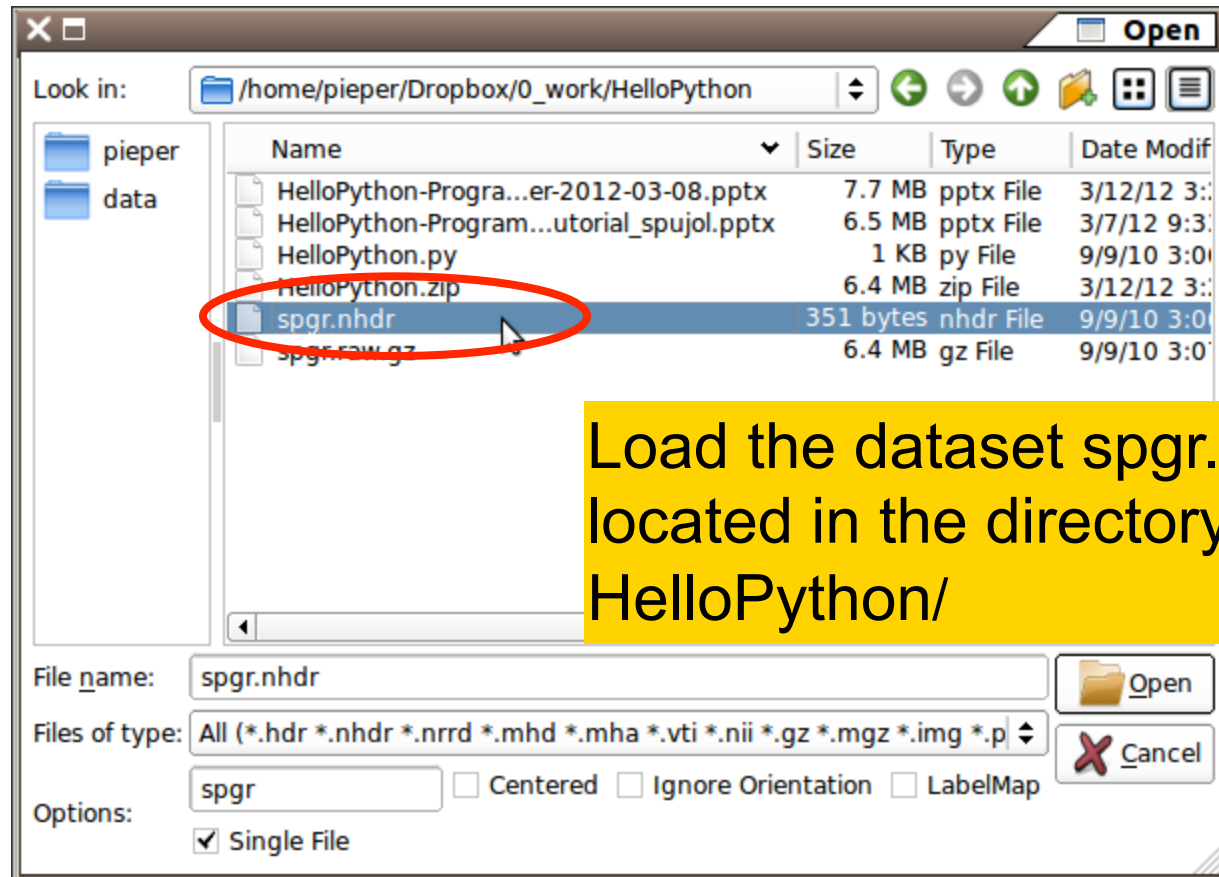
The screenshot shows a terminal window titled "Python 2.6.6 (r266:84292, Mar 1 2012, 12:53:23) [GCC 4.4.5] on linux2". The prompt is ">>> slicer.|". A dropdown menu is visible, listing the following completion options: logic, moduleNames, modulelogic, modulermml, modules, modulewidget, and mrmlScene. The option "mrmlScene" is currently selected and highlighted in blue.

# Add Volume Dialog



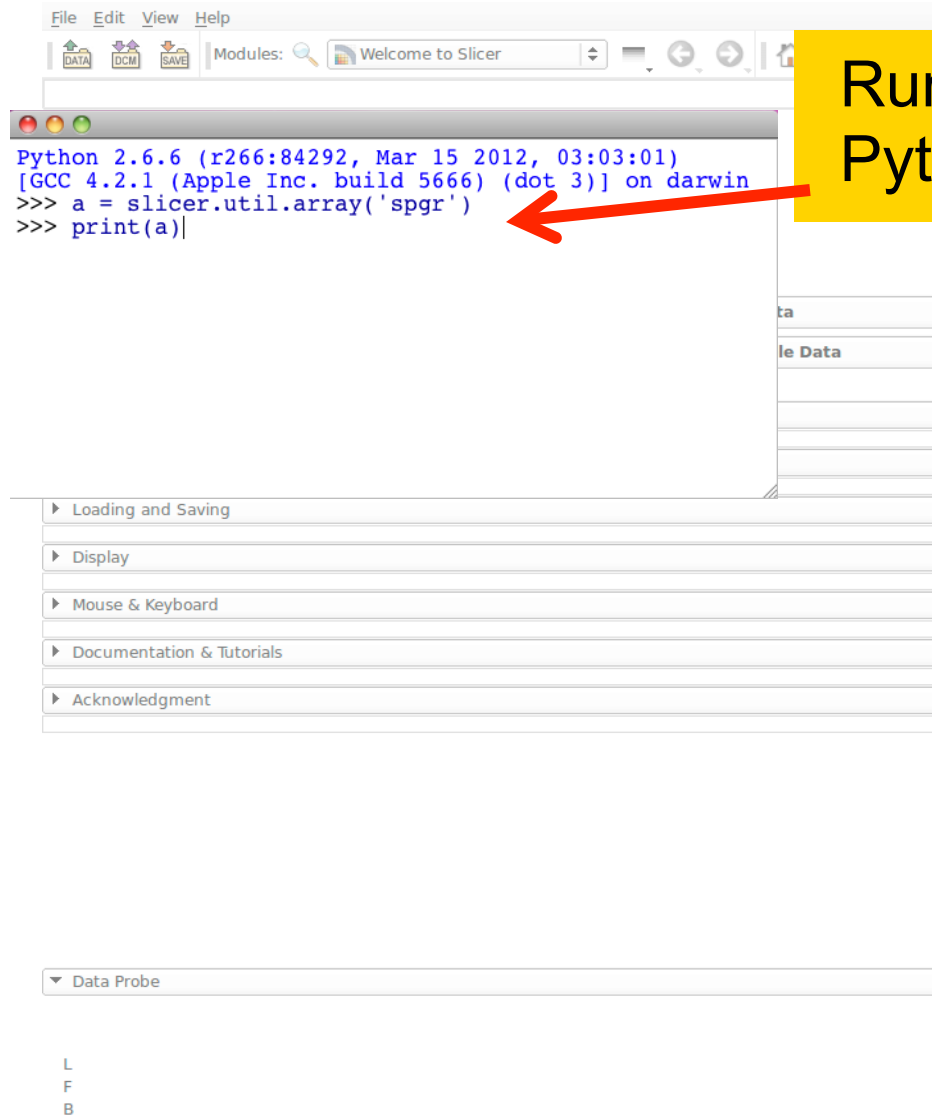
None RAS: (125.0, -125.0, 1.0),

# Add spgr.nhdr





# Access to MRML and Arrays



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



# Access to MRML and Arrays

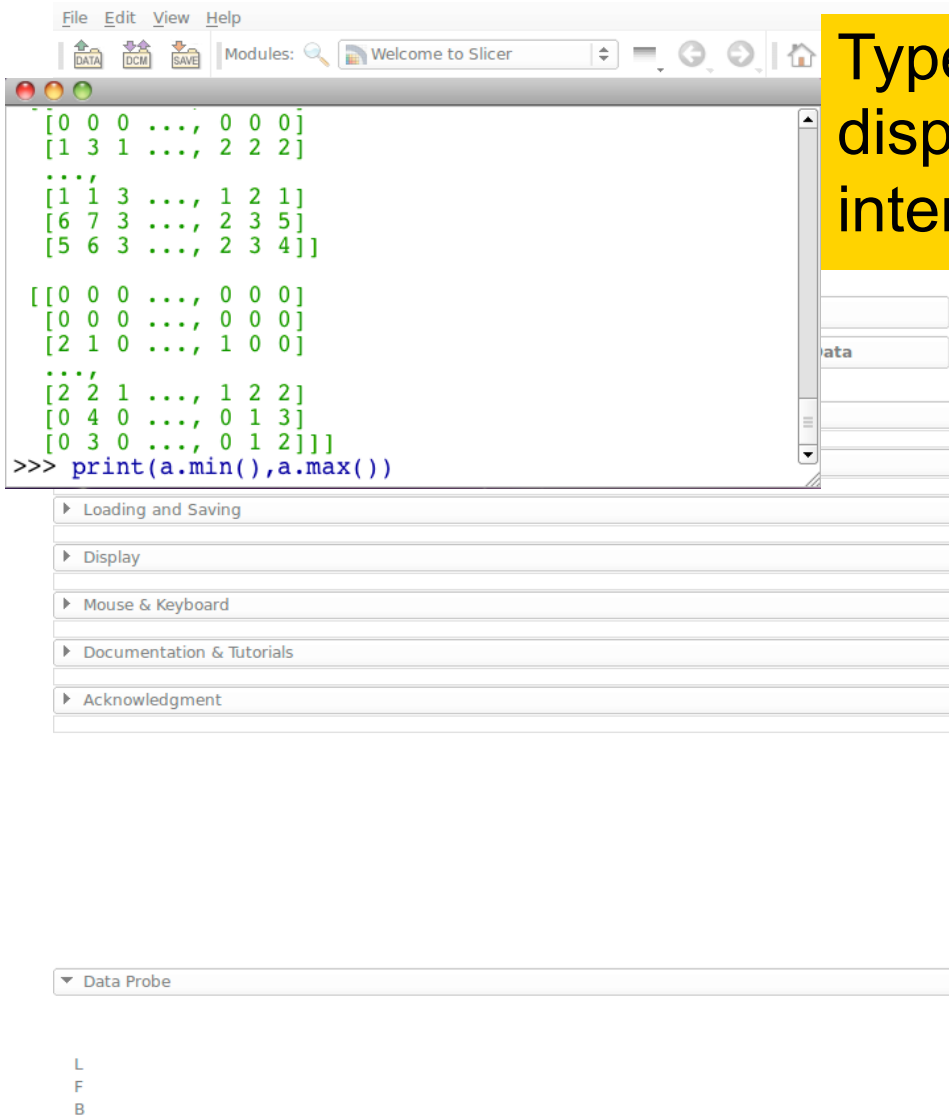
The screenshot displays the Slicer software interface. At the top, the menu bar includes File, Edit, View, and Help. Below it, the 'Modules' section shows 'Welcome to Slicer'. The main window is divided into several panels:

- Python Console:** A window on the left shows a list of MRML data points in a structured format, such as `[0 0 0 ... 0 0 0]` and `[1 3 1 ... 2 2 2]`. The console ends with `>>>`.
- 3D View:** A central 3D view of a brain slice, outlined in purple, with axes labeled R (Right), A (Anterior), and L (Left).
- 2D View:** A bottom panel showing three 2D slices of the brain in axial, sagittal, and coronal views. Above these slices are sliders for color calibration: S: -27.53, Y: 2.30, G: 13.77.
- Left Panel:** A sidebar with expandable sections: Loading and Saving, Display, Mouse & Keyboard, Documentation & Tutorials, and Acknowledgment.
- Data Probe:** A dropdown menu at the bottom left labeled 'Data Probe'.
- Bottom Left:** A small vertical stack of letters: L, F, B.

A green callout box on the right side of the image contains the text: "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



The screenshot shows the Slicer application interface. At the top, there is a menu bar with 'File', 'Edit', 'View', and 'Help'. Below the menu bar, there are icons for 'DATA', 'DCM', and 'SAVE', and a 'Modules' dropdown menu set to 'Welcome to Slicer'. The main window displays a Python console with the following output:

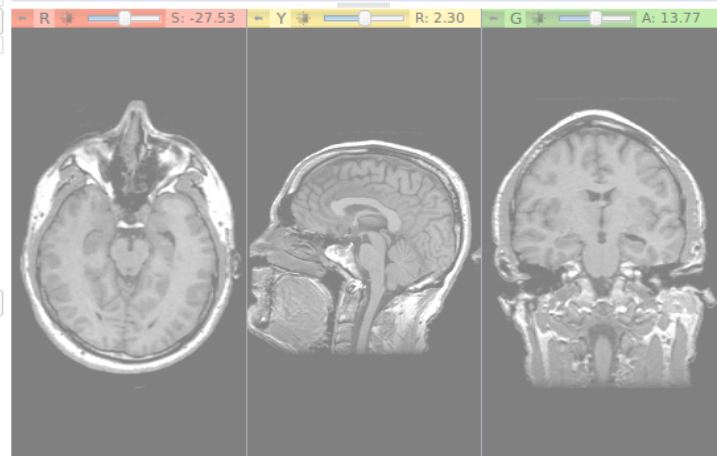
```
[0 0 0 ... 0 0 0]
[1 3 1 ... 2 2 2]
...
[1 1 3 ... 1 2 1]
[6 7 3 ... 2 3 5]
[5 6 3 ... 2 3 4]

[[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]]]
>>> print(a.min(),a.max())
```

Below the console, there is a sidebar with several expandable sections: 'Loading and Saving', 'Display', 'Mouse & Keyboard', 'Documentation & Tutorials', and 'Acknowledgment'. At the bottom, there is a 'Data Probe' section with a dropdown menu and a list of items: 'L', 'F', and 'B'.

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

→Use numpy array methods to analyze the data



# Access to MRML and Arrays

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

```
>>> print(a.min(), a.max())  
(0, 355)  
>>>
```

The array `a` is defined as:

```
[[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]]]
```

On the right, a 3D volume rendering of an MRI slice is shown. A green bounding box is drawn around the volume, with the axes labeled: **R** (Right), **A** (Anterior), and **L** (Left). A vertical line is drawn along the **I** (Inferior) axis. Below the 3D view, a control bar shows the slice position: **S: -27.53**, **R: 2.30**, **G: ...**, and **A: 13.77**. At the bottom, three orthogonal slice views are displayed: axial, sagittal, and coronal.

A green callout box in the upper right of the 3D view contains the text: **I min = 0 ; I max = 355**

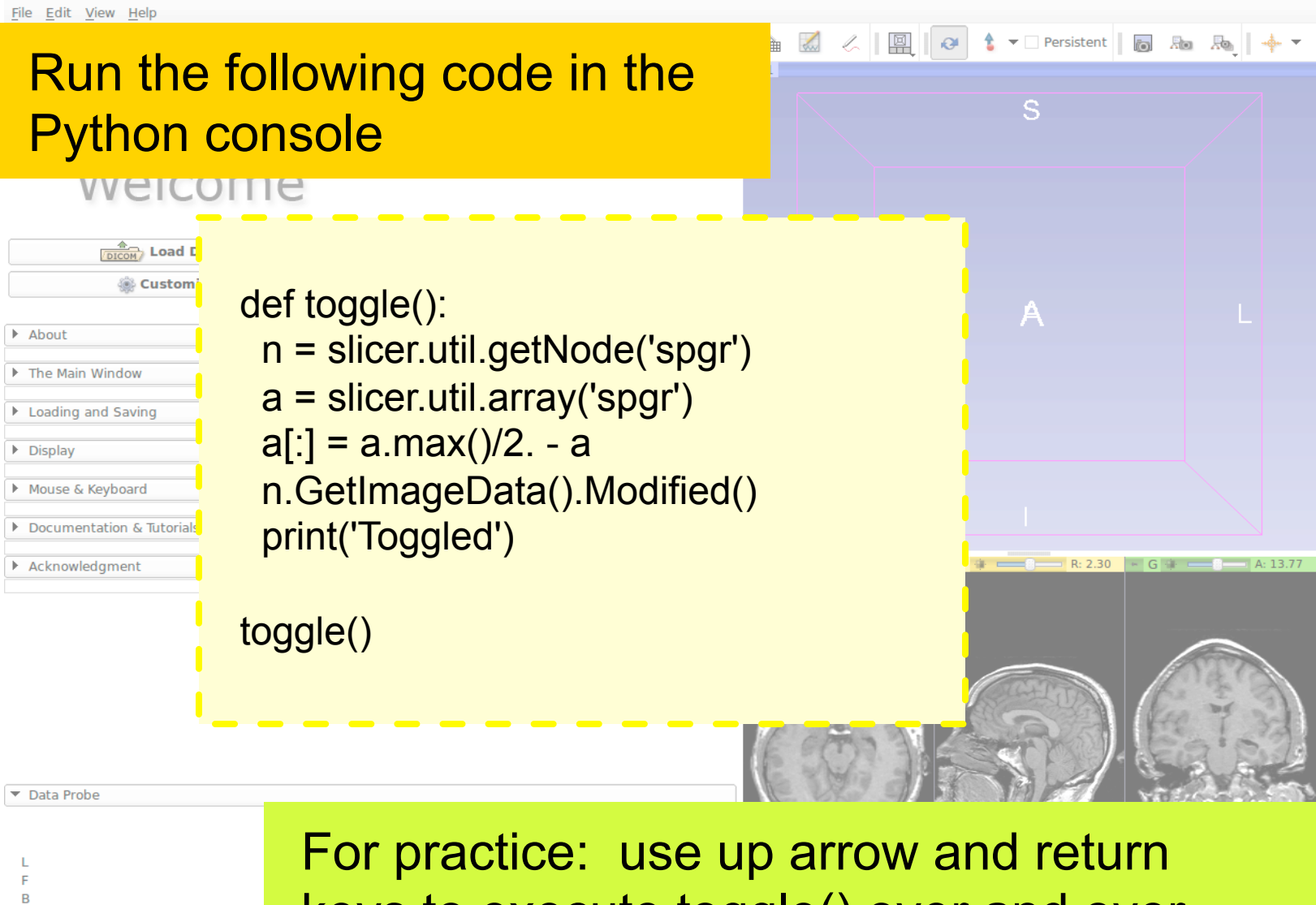
# Manipulating Arrays

Run the following code in the Python console

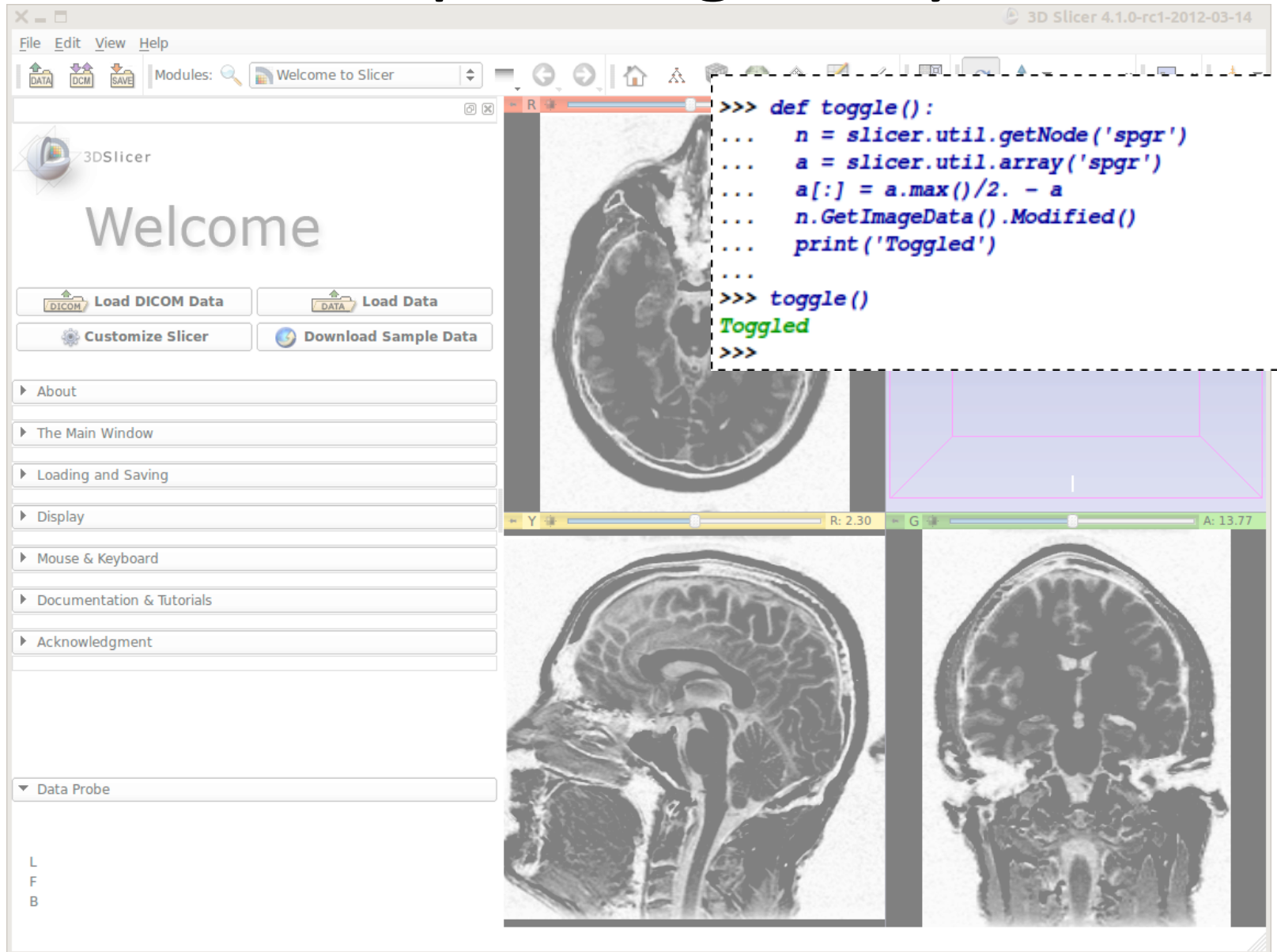
```
def toggle():  
    n = slicer.util.getNode('spgr')  
    a = slicer.util.array('spgr')  
    a[:] = a.max()/2. - a  
    n.GetImageData().Modified()  
    print('Toggled')
```

```
toggle()
```

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



# Manipulating Arrays



The screenshot displays the 3D Slicer software interface. On the left is a sidebar with a 'Welcome' message and several buttons: 'Load DICOM Data', 'Load Data', 'Customize Slicer', and 'Download Sample Data'. Below these are expandable sections for 'About', 'The Main Window', 'Loading and Saving', 'Display', 'Mouse & Keyboard', 'Documentation & Tutorials', and 'Acknowledgment'. At the bottom left is a 'Data Probe' section with 'L', 'F', and 'B' options.

The main 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 Python console is overlaid on the top right, containing 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()  
Toggled  
>>>
```

Below the console, a purple rectangular area is visible, and a status bar at the bottom shows 'R: 2.30', 'G', and '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

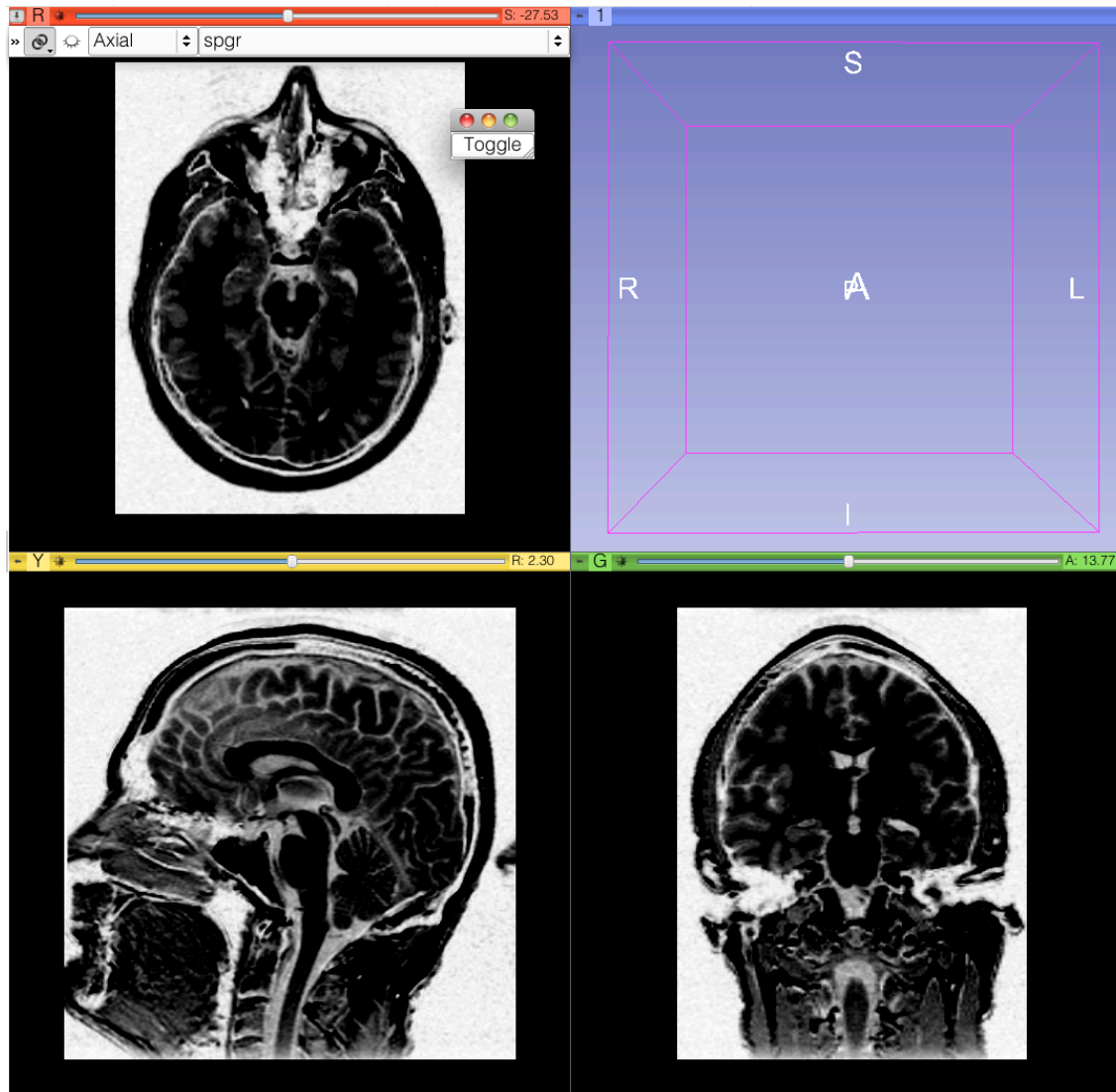
Run the following code in the Python console

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

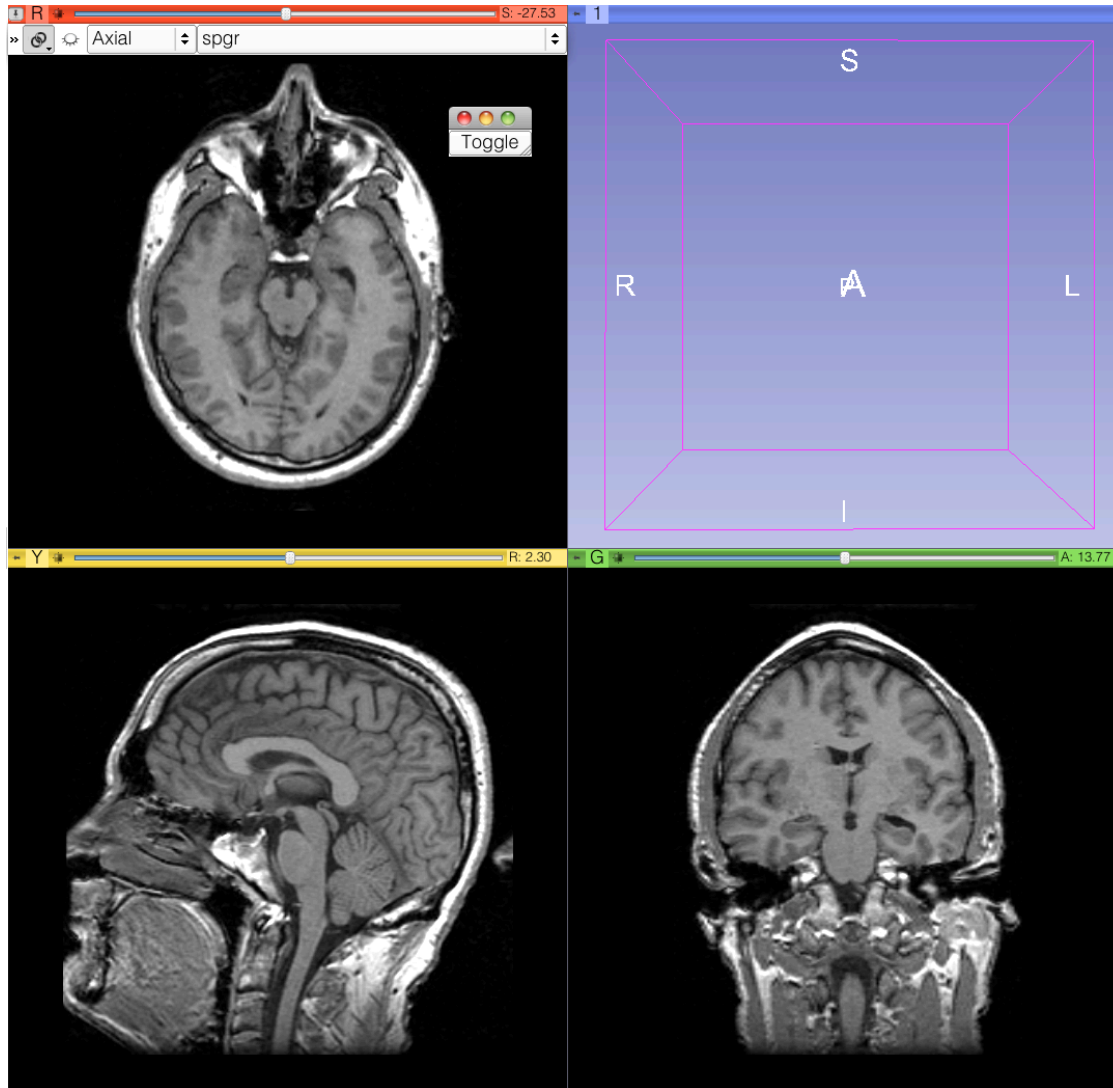
What do you think will happen when you run this code? What about when you push the button?



# 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
- See 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 (DSI)"] # 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 Women's Hospital and was
        partially funded by NIH grant 5P41MH082228-02SI (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!")

```



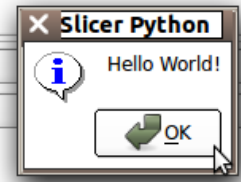
▼ Help & Acknowledgement

Help Acknowledgement

Example of scripted loadable extension for the HelloPython tutorial.

▼ A collapsible button

Hello world



# PART B: INTEGRATION OF THE HELLOPYHTON TUTORIAL 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):
        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 dummy collapsible button
    sampleFormLayout = qt.QFormLayout(sampleCollapsibleButton)
```

```
# HelloWorld button
helloWorldButton = qt.QPushButton("Hello world")
helloWorldButton.setToolTip("Print 'Hello world' in standard ouput.")
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 !"
```

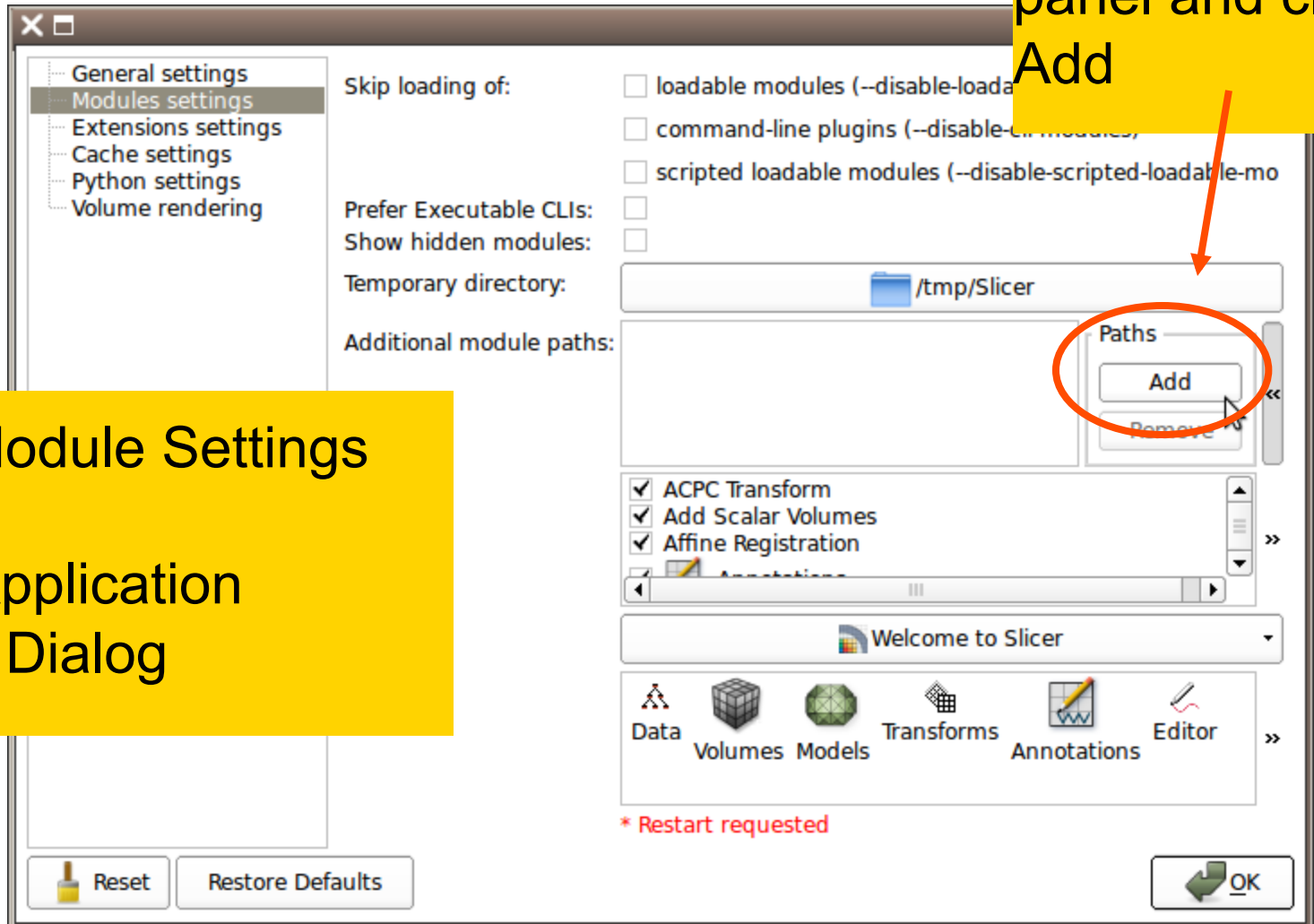
Add this  
Text in  
section B

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

# Integrating HelloPython

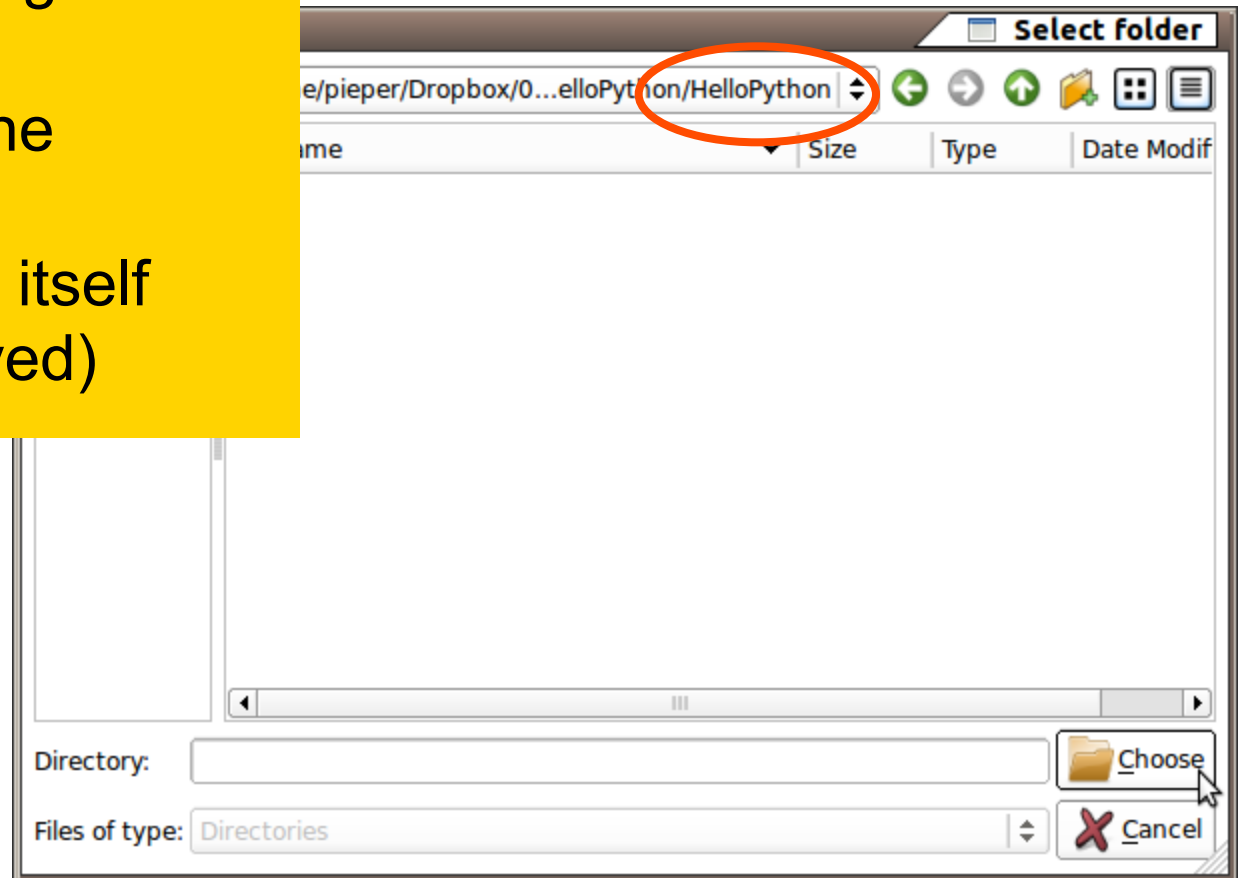
Open the side panel and click Add

Select Module Settings from the Edit -> Application Settings Dialog



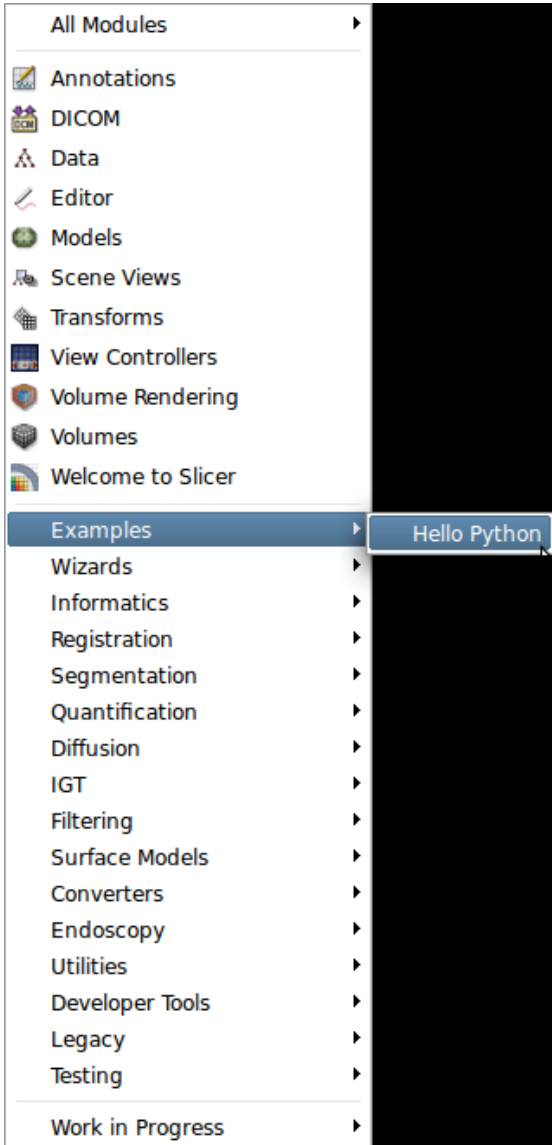
# Integrating HelloPython

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

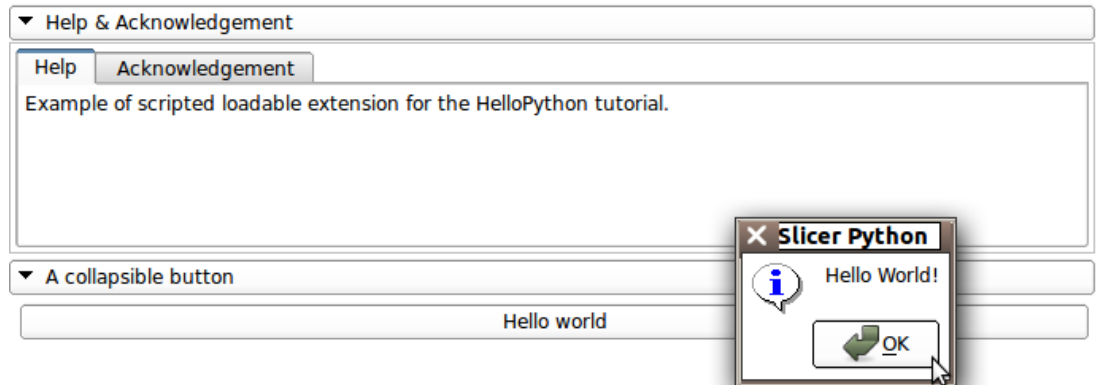


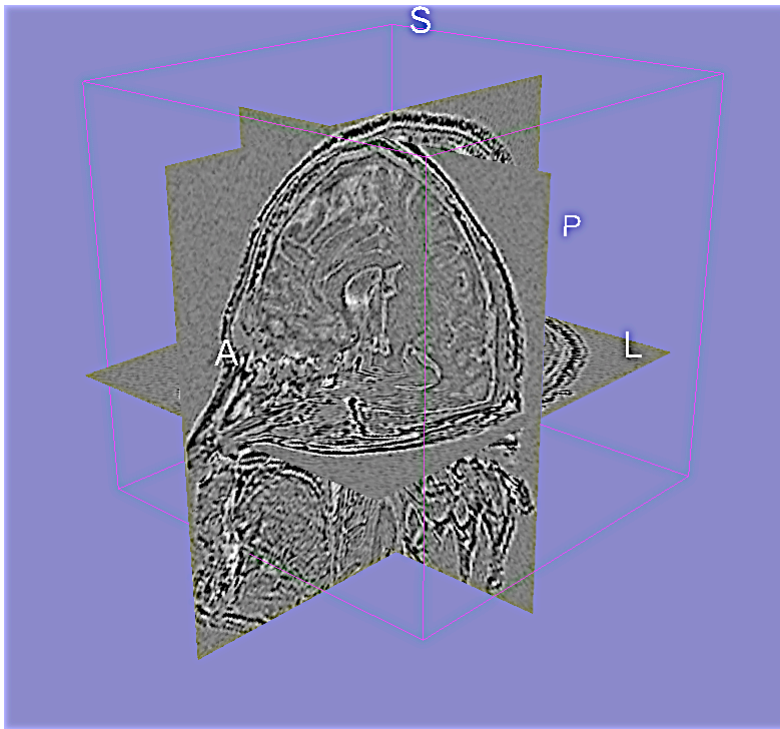


# HelloPython in Slicer



Restart Slicer when prompted.  
Hello Python is now in the  
Modules Menu and can be  
used!





## Part C:

# Implementing the Laplace\* Operator

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

---

# Goals of this part

- 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
  - Manipulate the slice views to display result

# 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

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

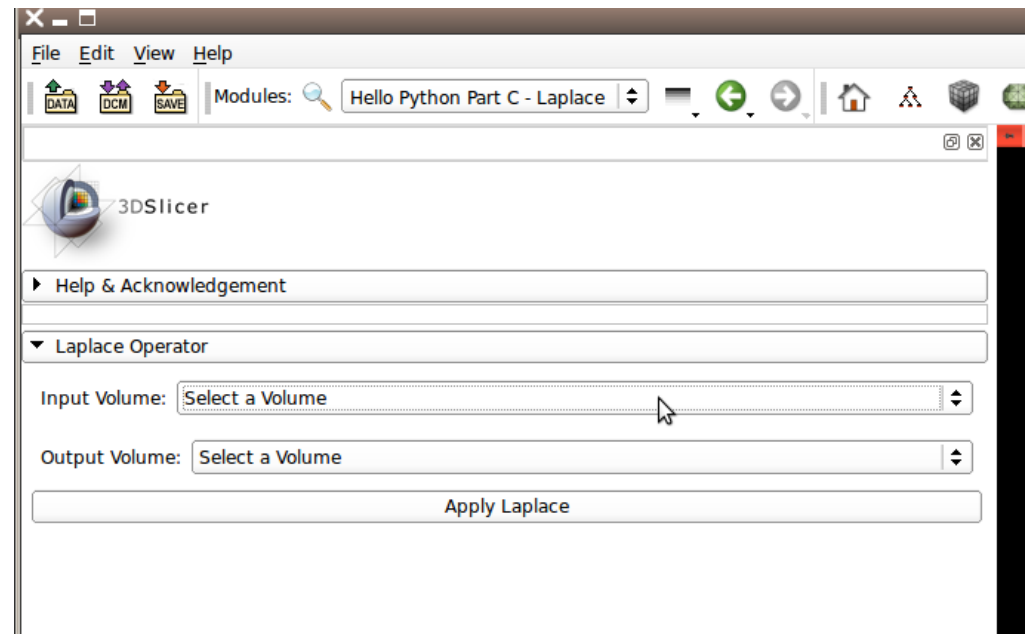
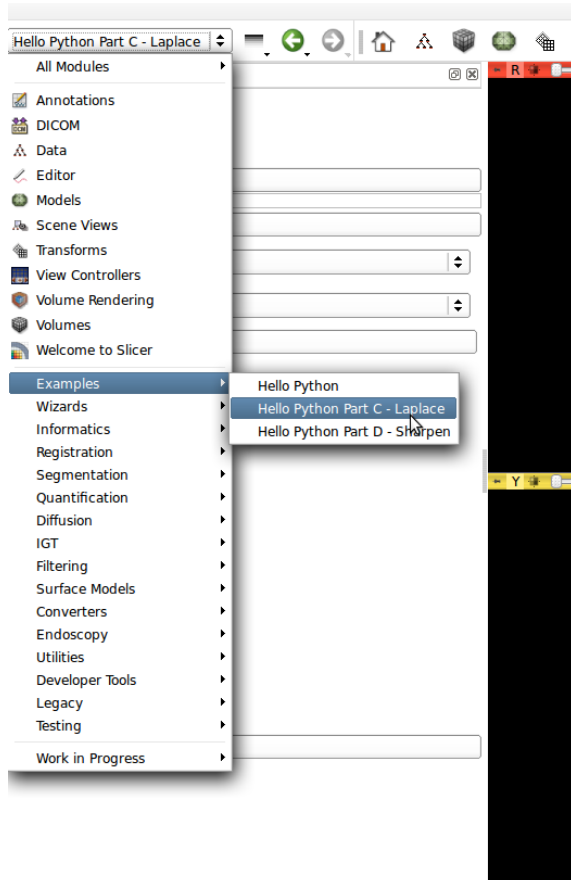
- **vtkImageLaplacian** is a `vtkImageAlgorithm` operates on `vtkImageData` (see <http://vtk.org>)
- **vtkMRMLScalarVolumeNode** is a Slicer MRML class that contains `vtkImageData`, plus orientation information `ijkToRAS` matrix (see [http://www.slicer.org/slicerWiki/index.php/Coordinate\\_systems](http://www.slicer.org/slicerWiki/index.php/Coordinate_systems))
- **qMRMLNodeComboBox** is a Slicer widget that gives direct access to volume nodes in the scene

# In More Detail (Continued)

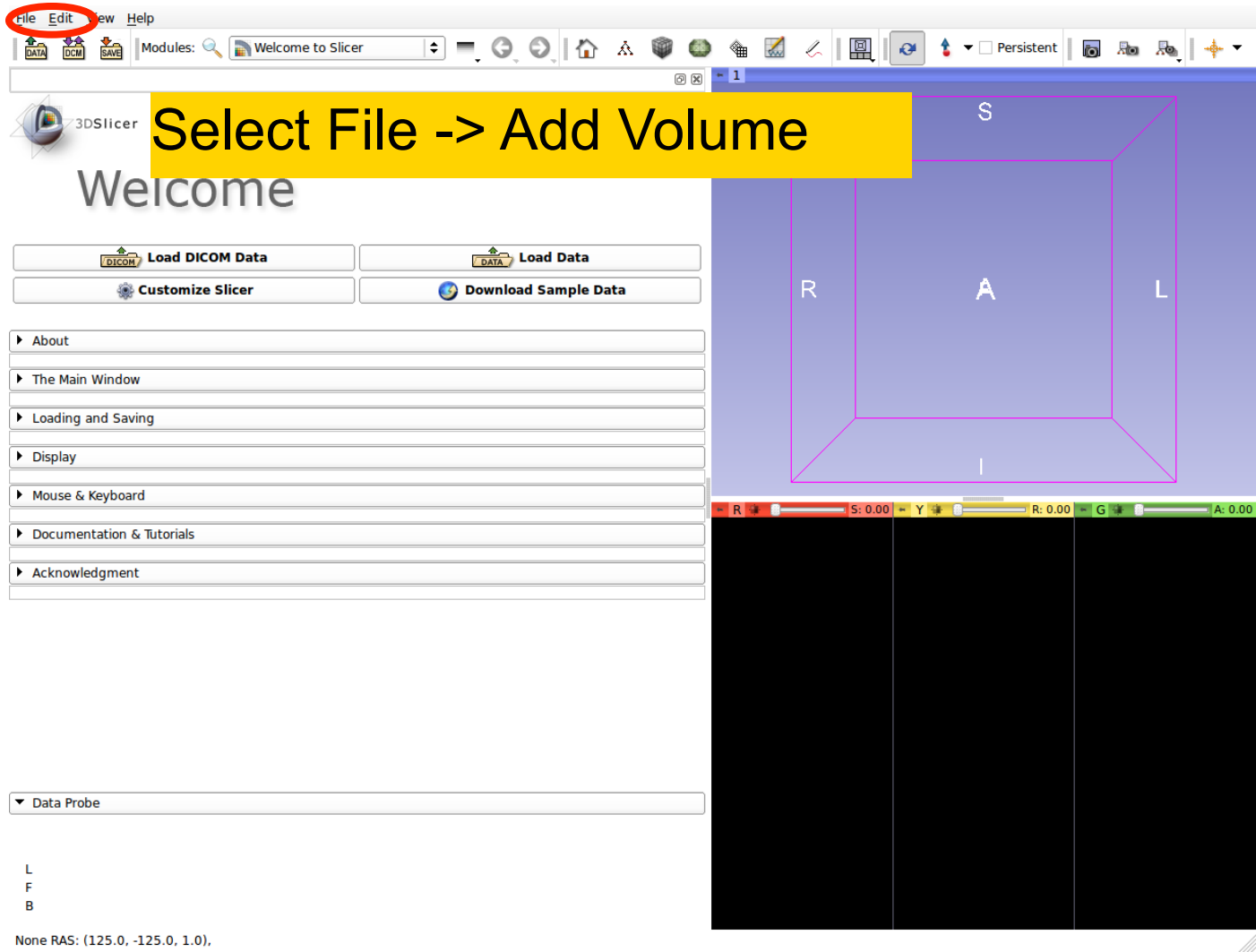
- Global  **slicer**  module gives python access to GUI (via  **slicer.app** ), modules (via  **slicer.modules** ) and data (via  **slicer.mrmlScene** ).
- **slicer.app.applicationLogic()**  provides helper utilities for manipulating Slicer state

# Go To Laplace Module

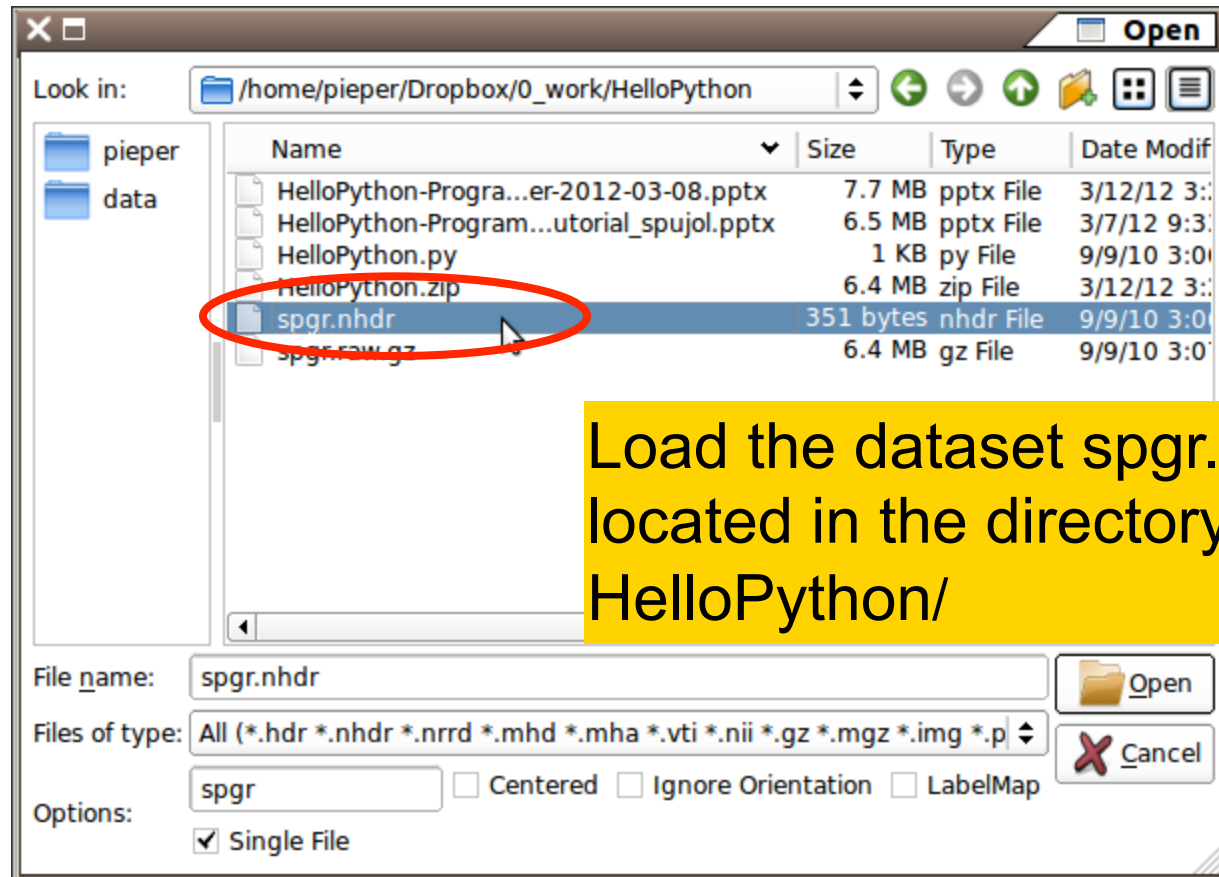
Re-start Slicer and select module. Note that combobox is empty



# 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

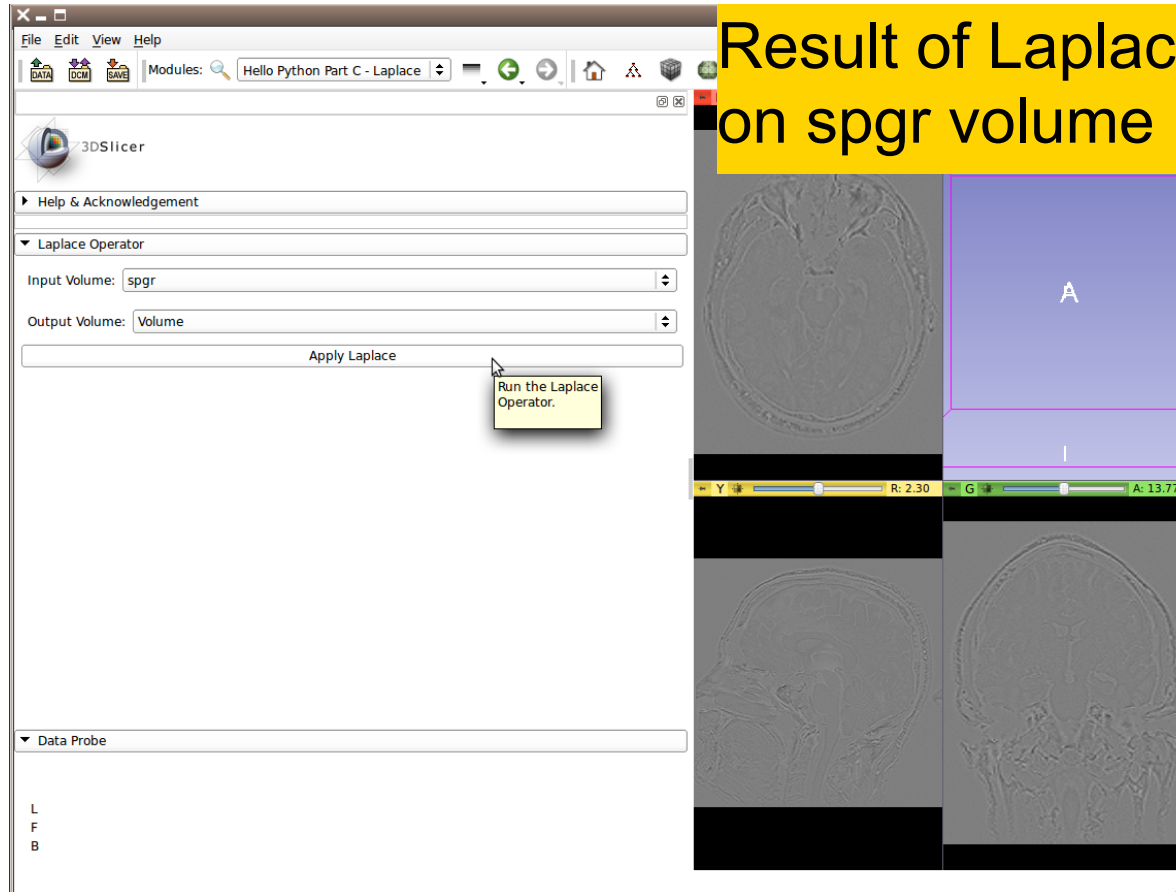
Output Volume: Volume

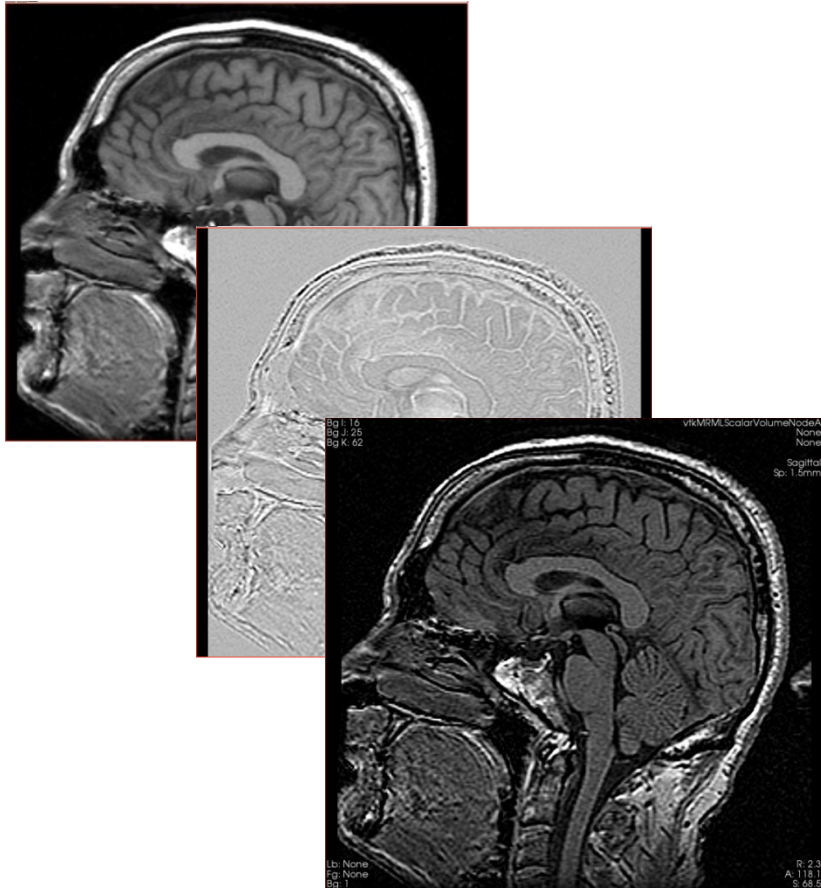
Apply Laplace

Run the Laplace Operator.

(3) Run the module

# Laplace Module





## Part D:

# Image Sharpening with the Laplace Operator

# Goals of this part

- Add a processing option for image sharpening
  - Implement this operation using an existing Slicer Command Line Module

# 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)
```

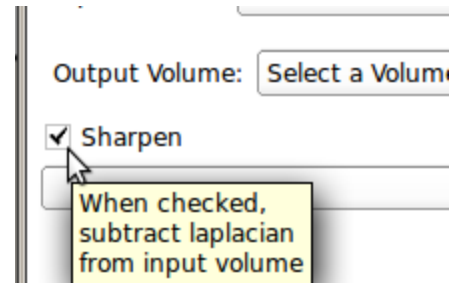
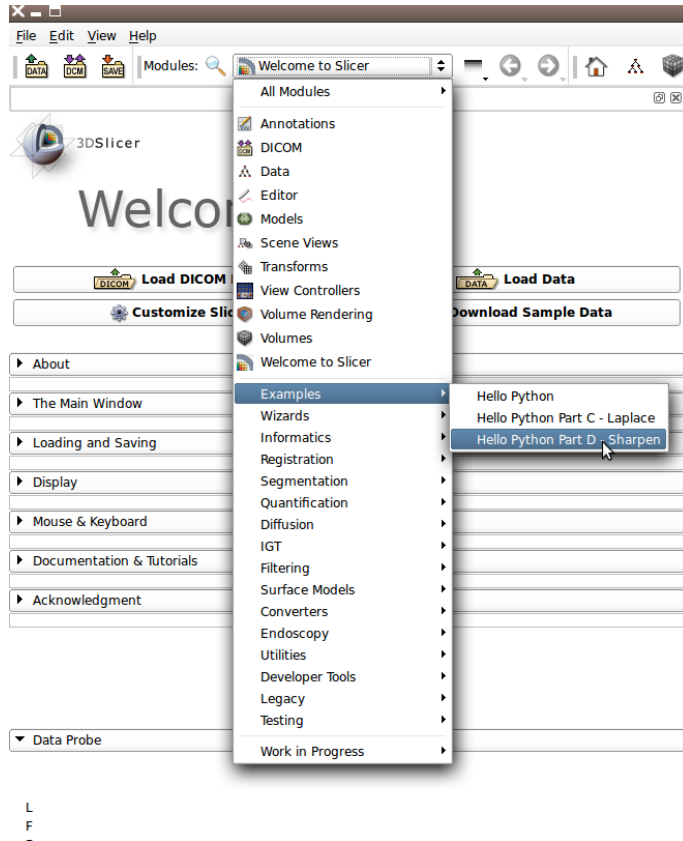
# In More Detail

- **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

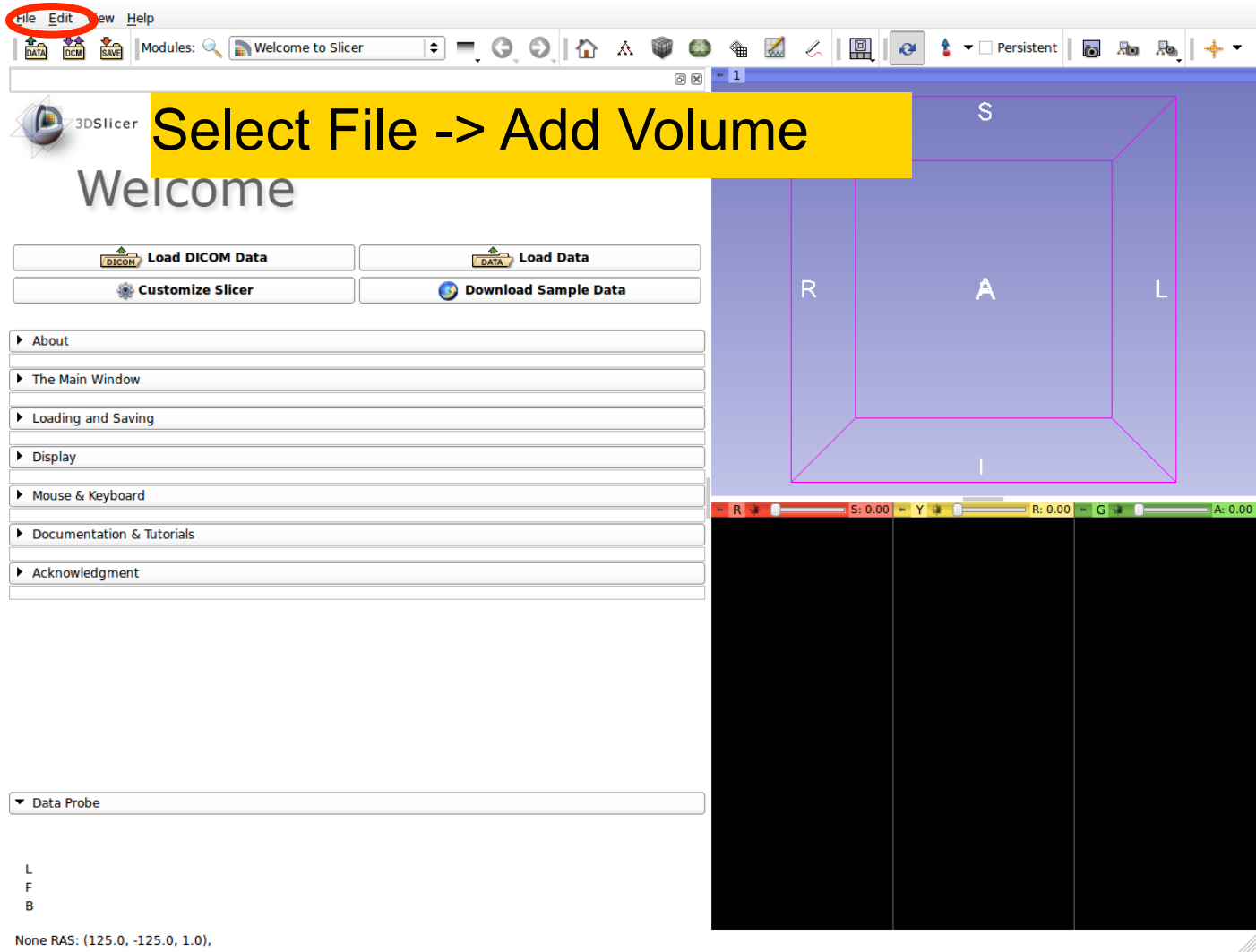


# Go To Sharpen Module

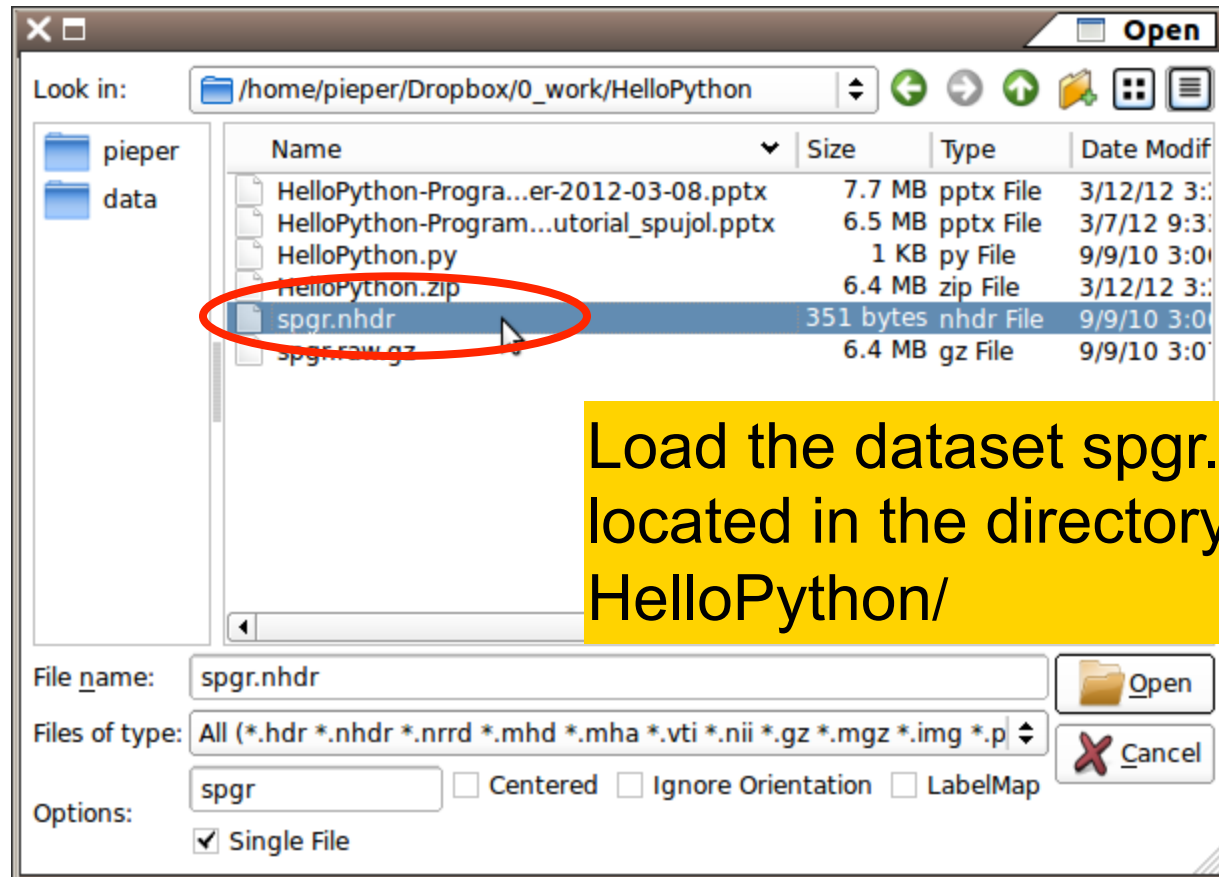
Re-start Slicer and select module. Note the new sharpen check box



# 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

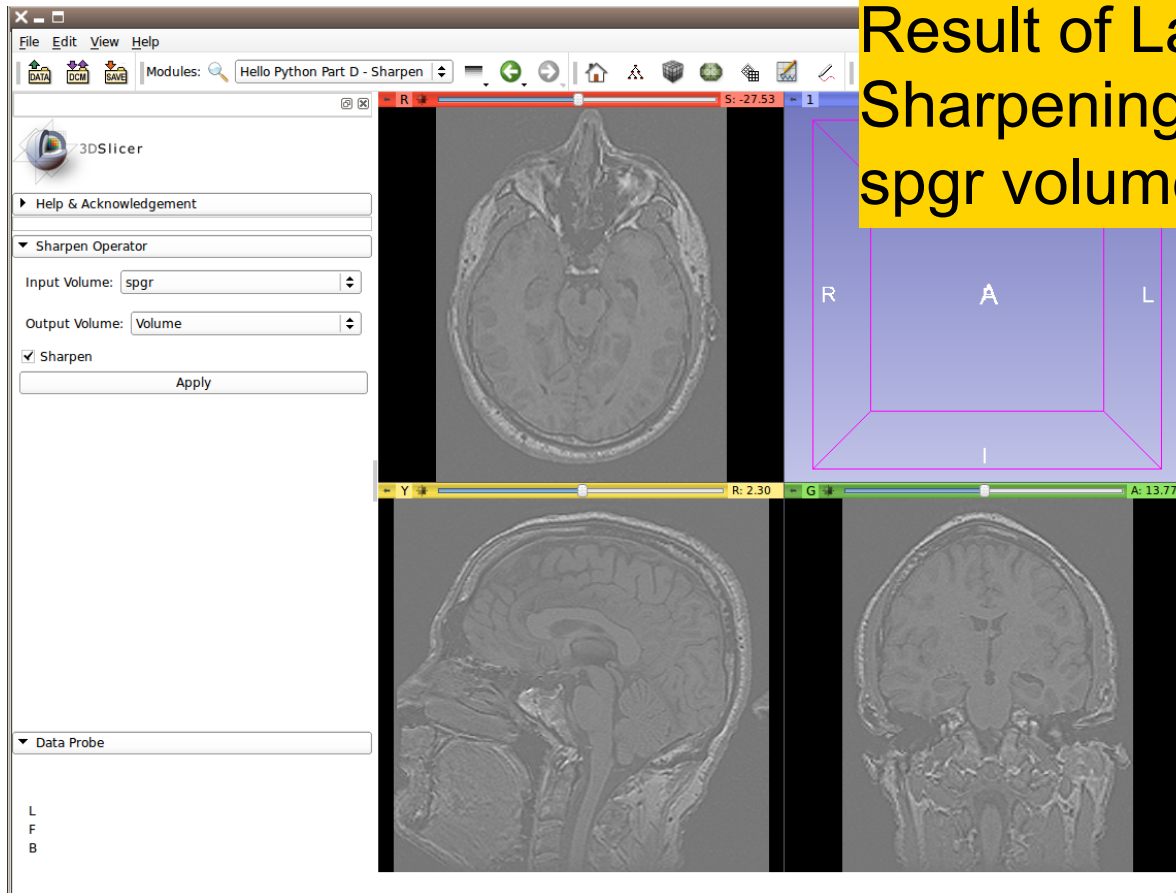
Sharpen

Apply

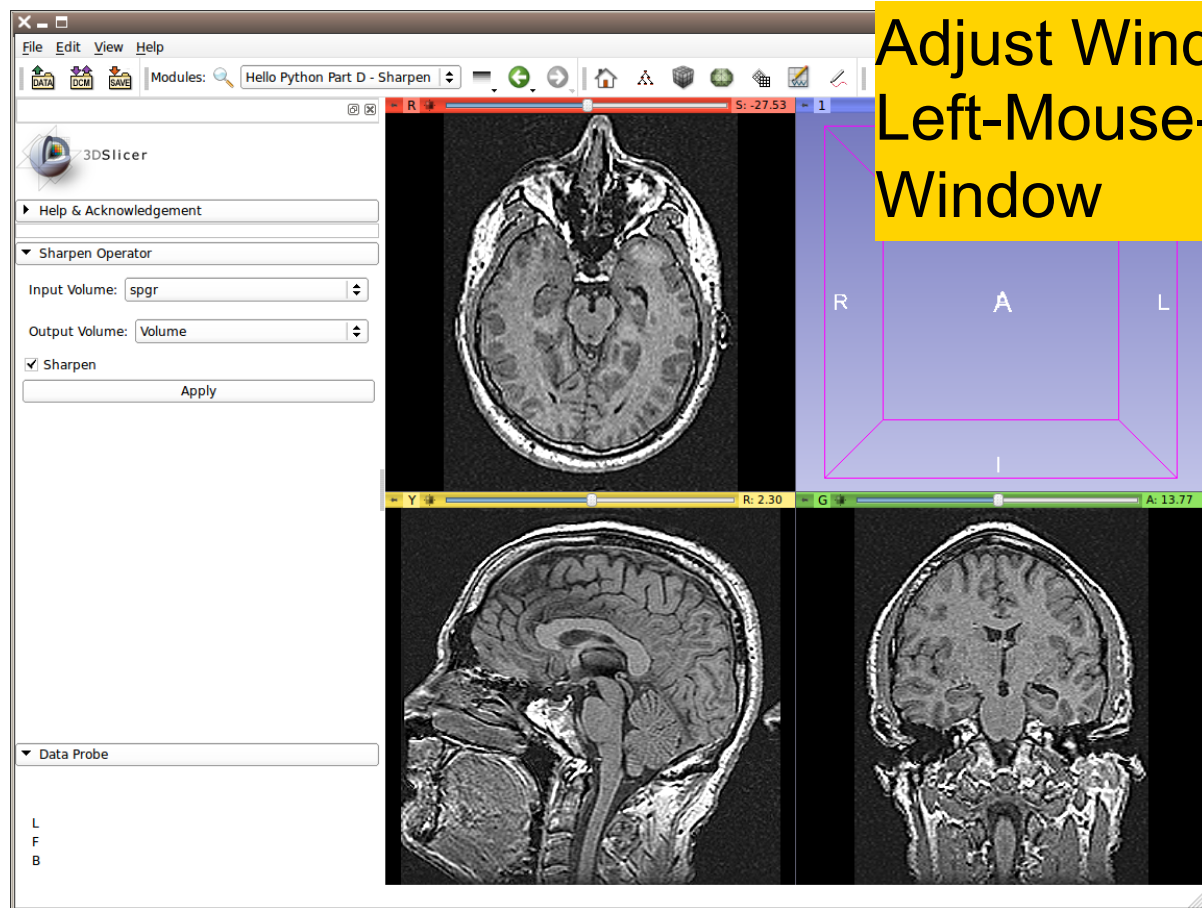
(3) Run the module in Sharpen mode

Run the Laplace or Sharpen Operator.

# Sharpen Module



# Sharpen Module



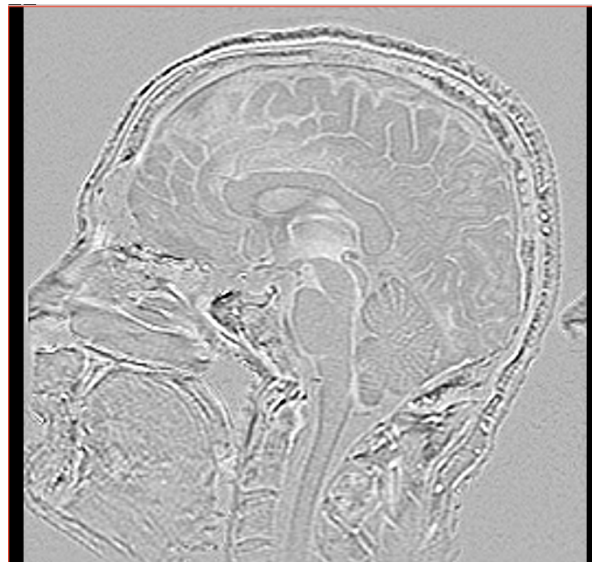
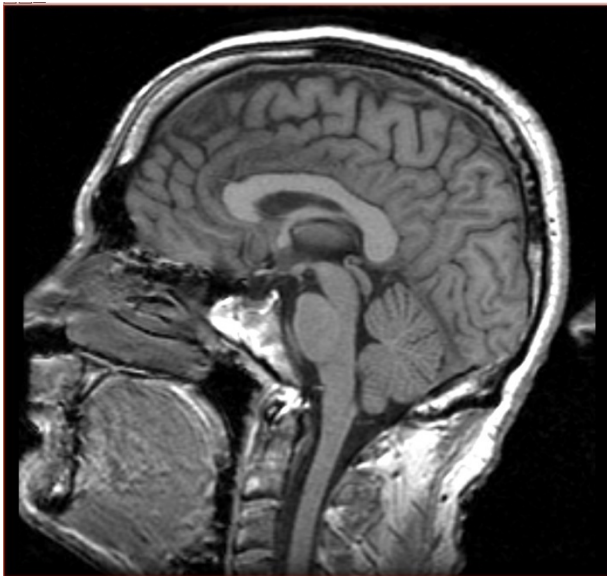
Adjust Window/Level with Left-Mouse-Drag in Slice Window

# 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 HelloWorld C++ tutorial for instructions on writing CLI modules like the Subtract Scalar Volumes
- 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





# Acknowledgments

---



**National Alliance for Medical Image Computing**

NIH U54EB005149



**Neuroimage Analysis Center**

NIH P41RR013218

---