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

Programming in Slicer4

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

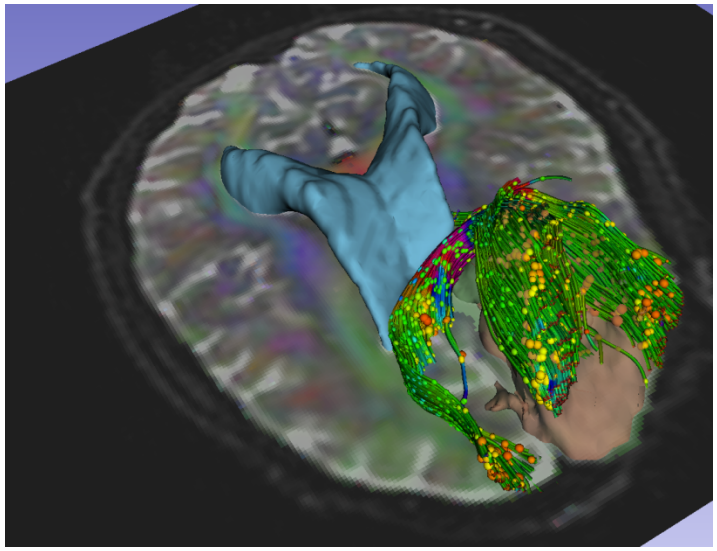
Steve Pieper, Ph.D.
Isomics Inc.

The NA-MIC Kit





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 Modules

- **Command Line Interface (CLI):** standalone executables with limited input/ output arguments
- **Scripted Modules (Python):** recommended for fast prototyping
- **Loadable Modules (C++ Plugins)** optimized for heaving computation

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

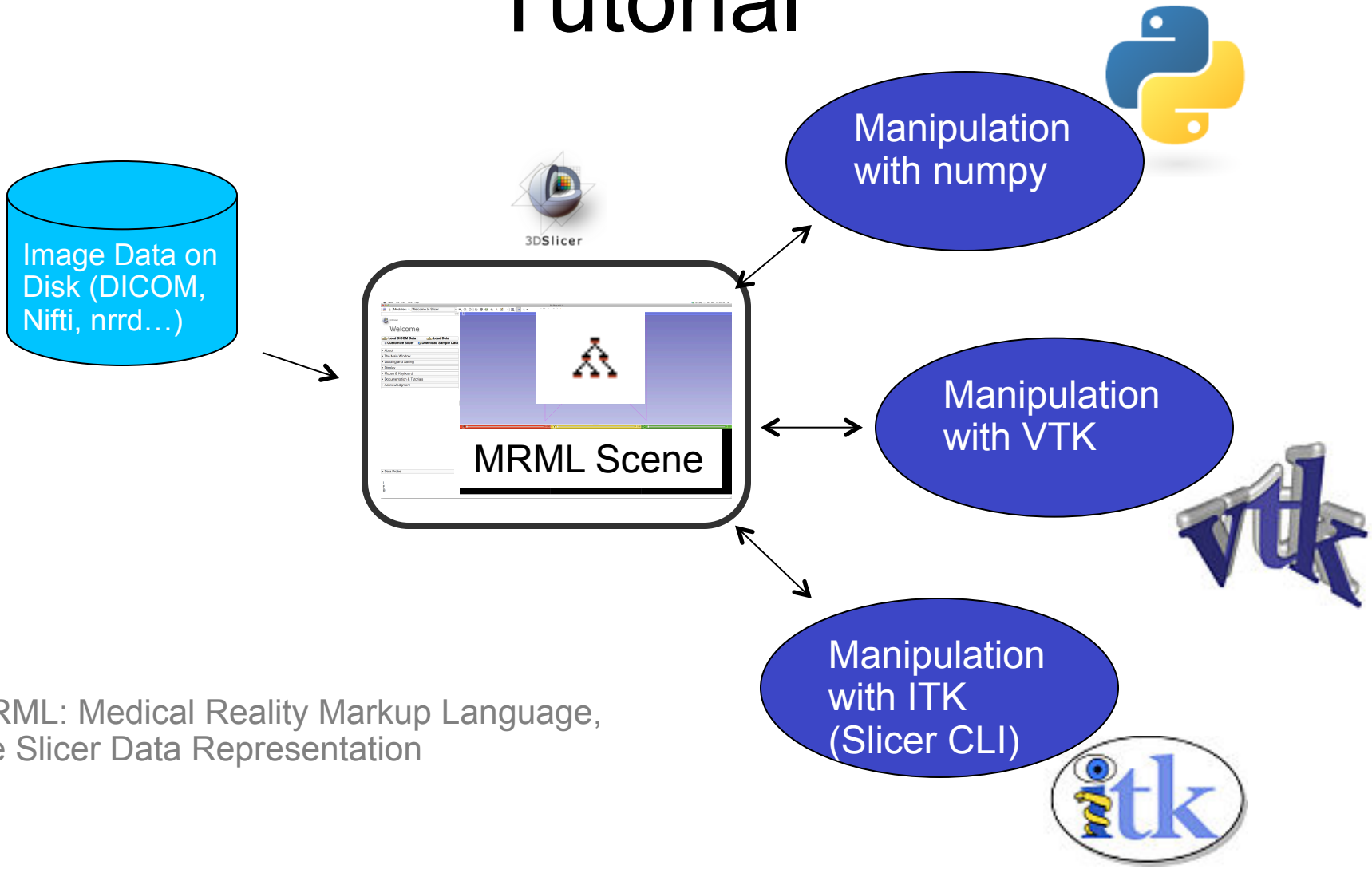
Tutorial Goal

- 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.

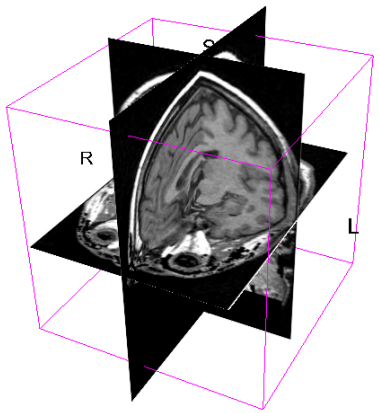
- The tutorial is available on the Slicer4 101 compendium:

http://www.slicer.org/slicerWiki/index.php/Training/4.0

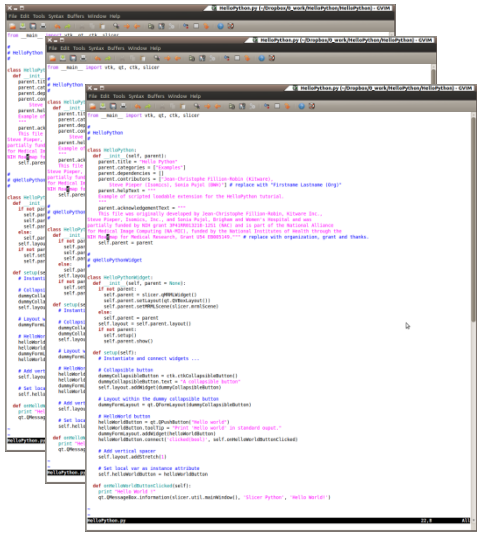
- Programming experience is required, and some familiarity with Python is essential.

Course Material

Slicer4.1 version available at www.slicer.org



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

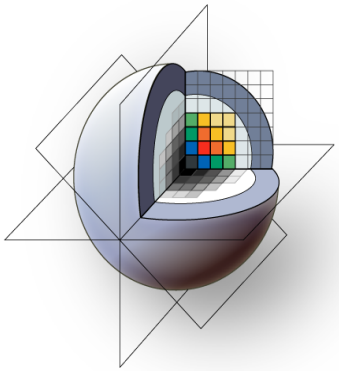


HelloPython.py
HelloLaplace.py
HelloSharpen.py

Unzip the HelloPython.zip archive

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



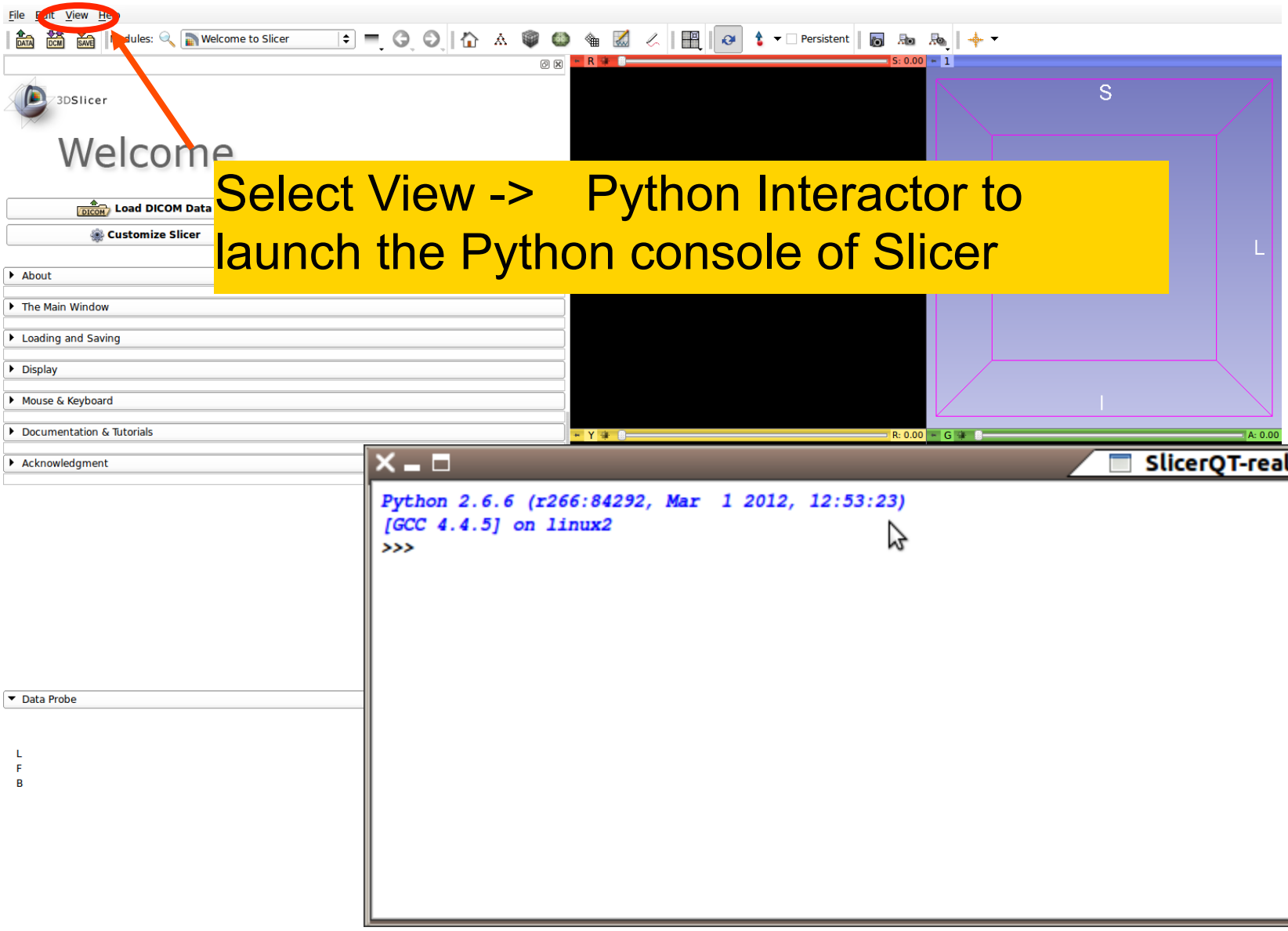
Part A: EXPLORING SLICER VIA PYTHON

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 ploting library),
    - ipython (interactive python)

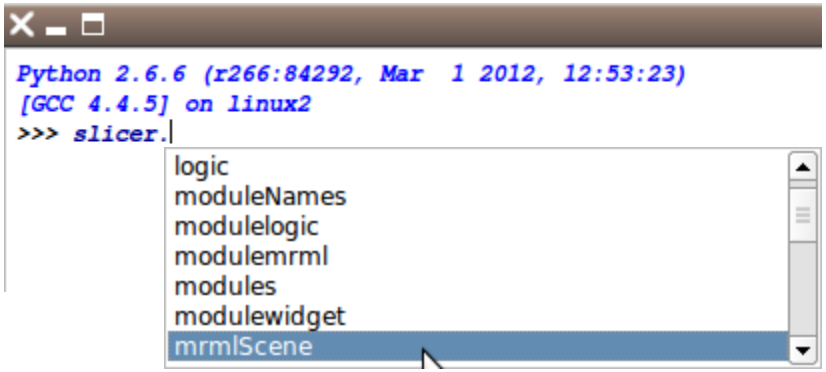
    and some other popular packages that we have found difficult to package for distribution

Python Console in Slicer

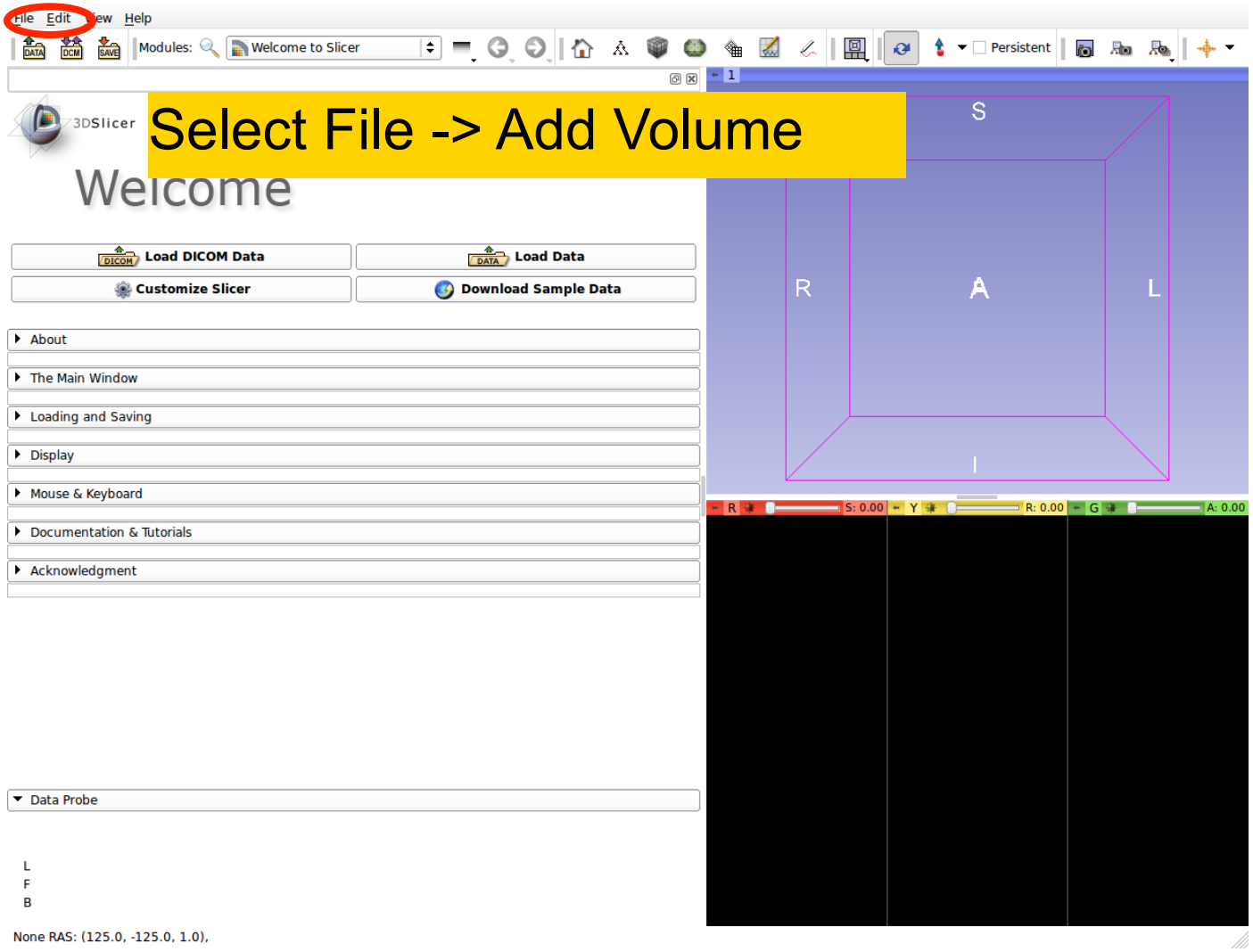


General Python Console Features

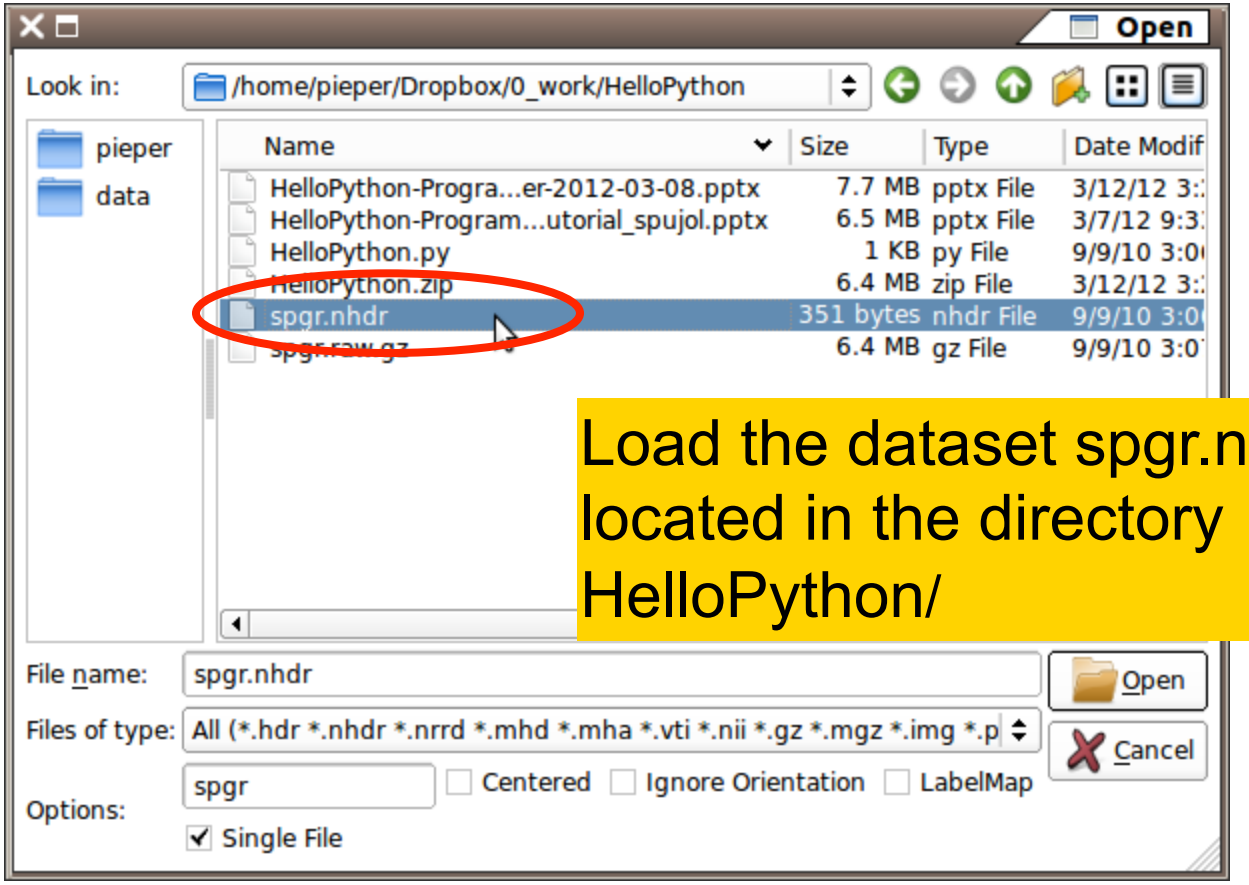
- Command Line Editing:
  - Left/Right Arrow Keys, Home, End
  - Delete (Control-D)
- Input History
  - Up/Down Arrow Keys
- Command Completion
  - Tab Key



Add Volume Dialog

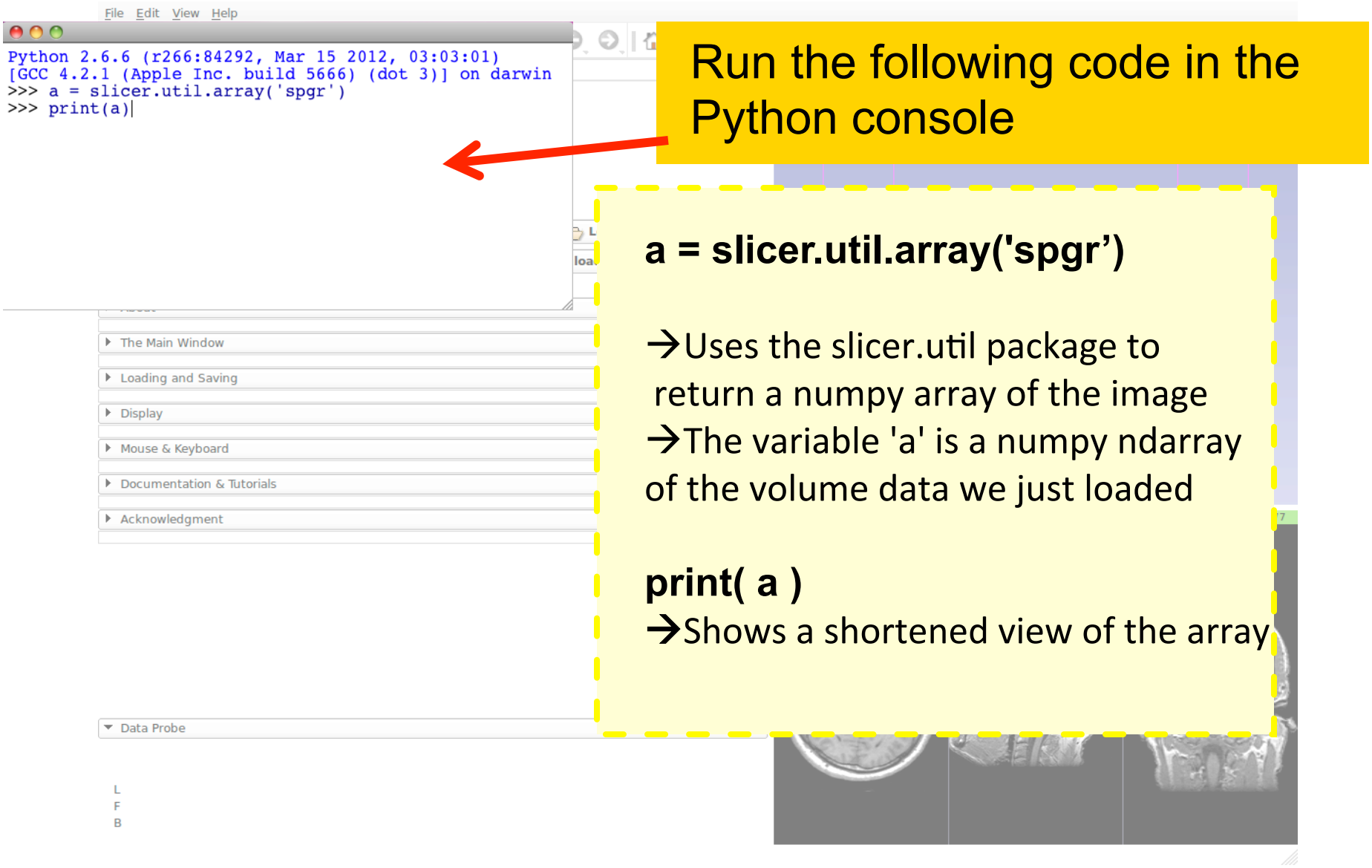


Add spgr.nhdr



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

Access to MRML and Arrays



Run the following code in the Python console

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

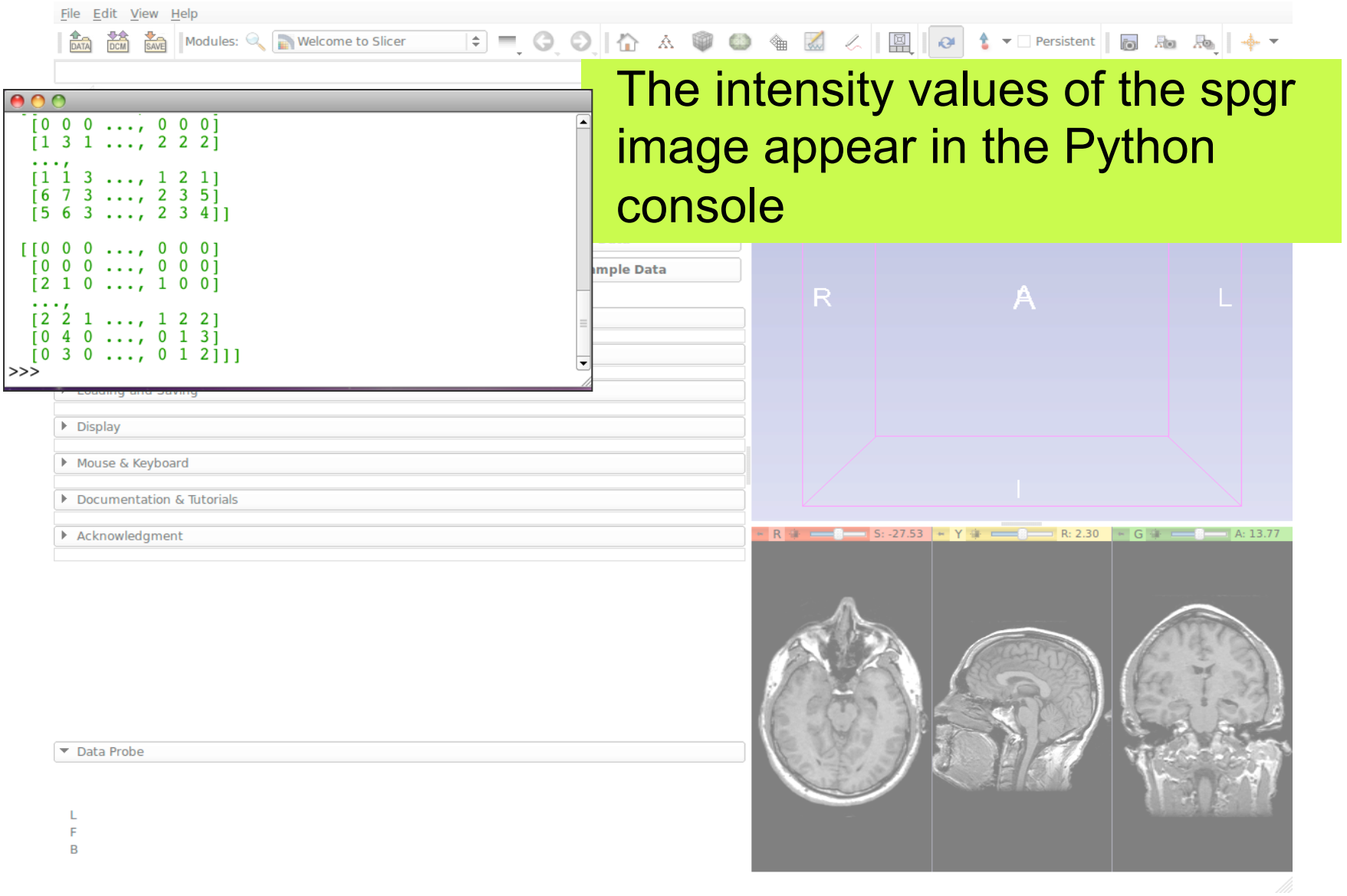
→Uses the slicer.util package to return a numpy array of the image

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

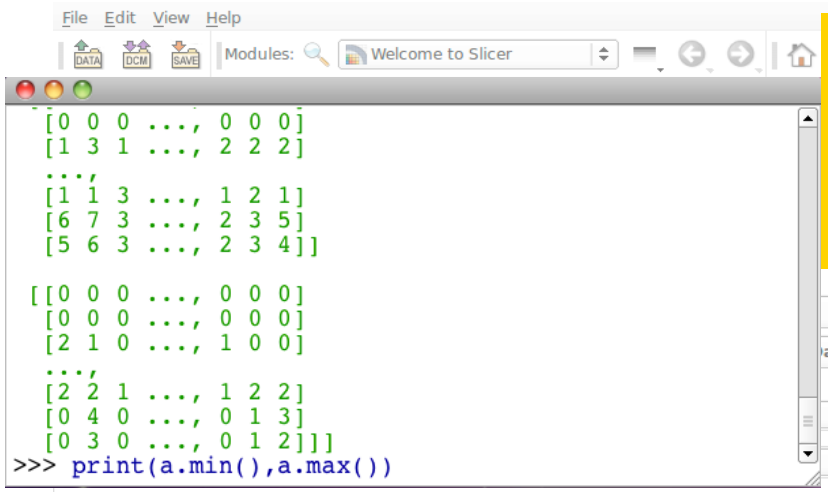
print(a)

→Shows a shortened view of the array

Access to MRML and Arrays



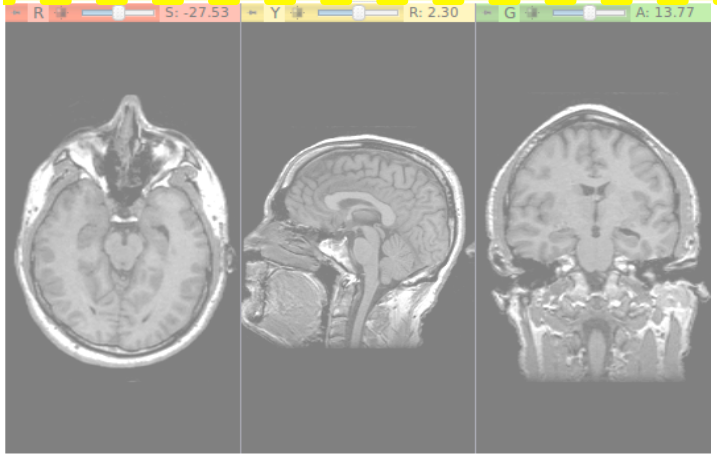
Access to MRML and Arrays



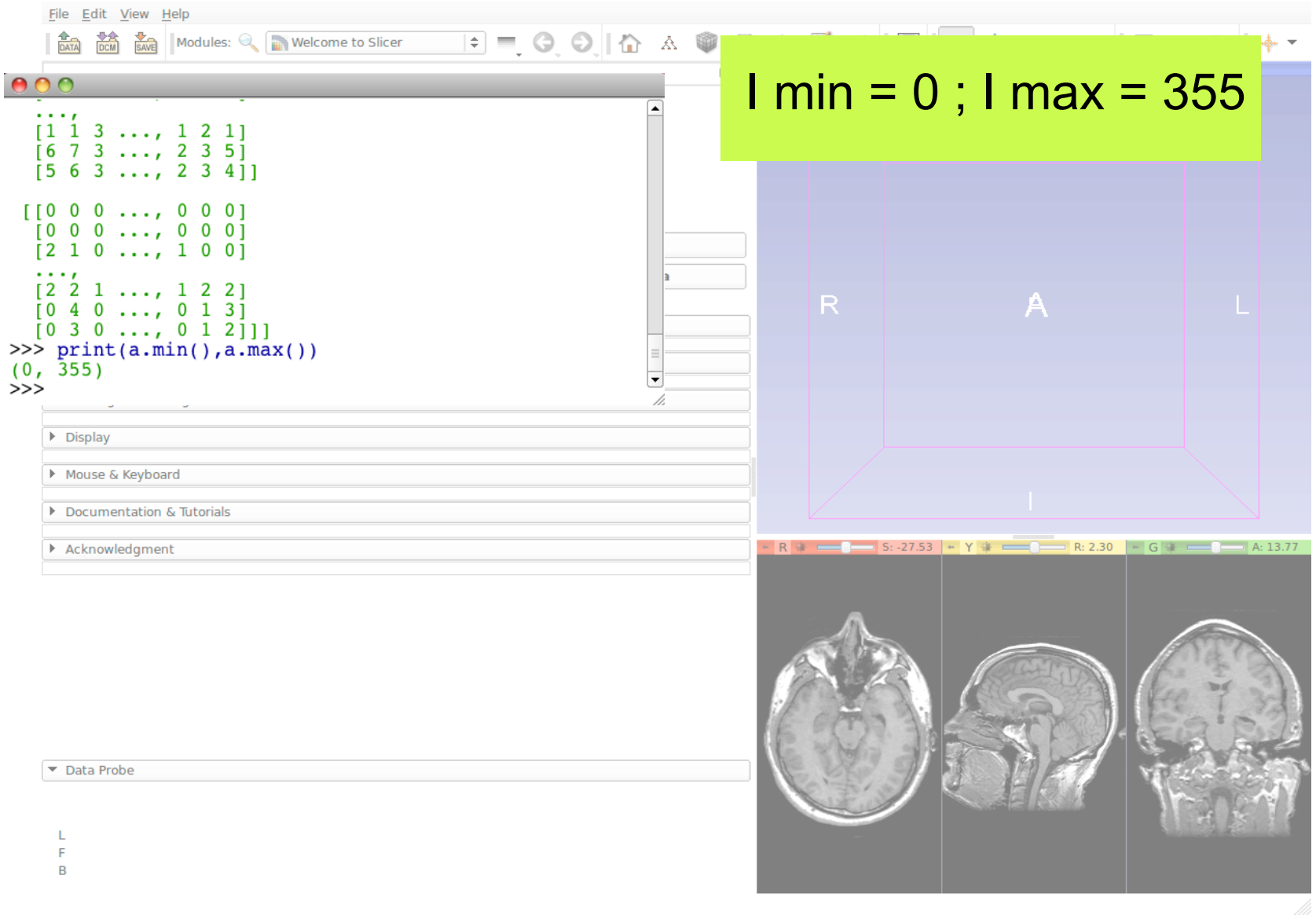
Type the following command to display the min and max intensity value of the spgr image

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

→Use numpy array methods to analyze the data

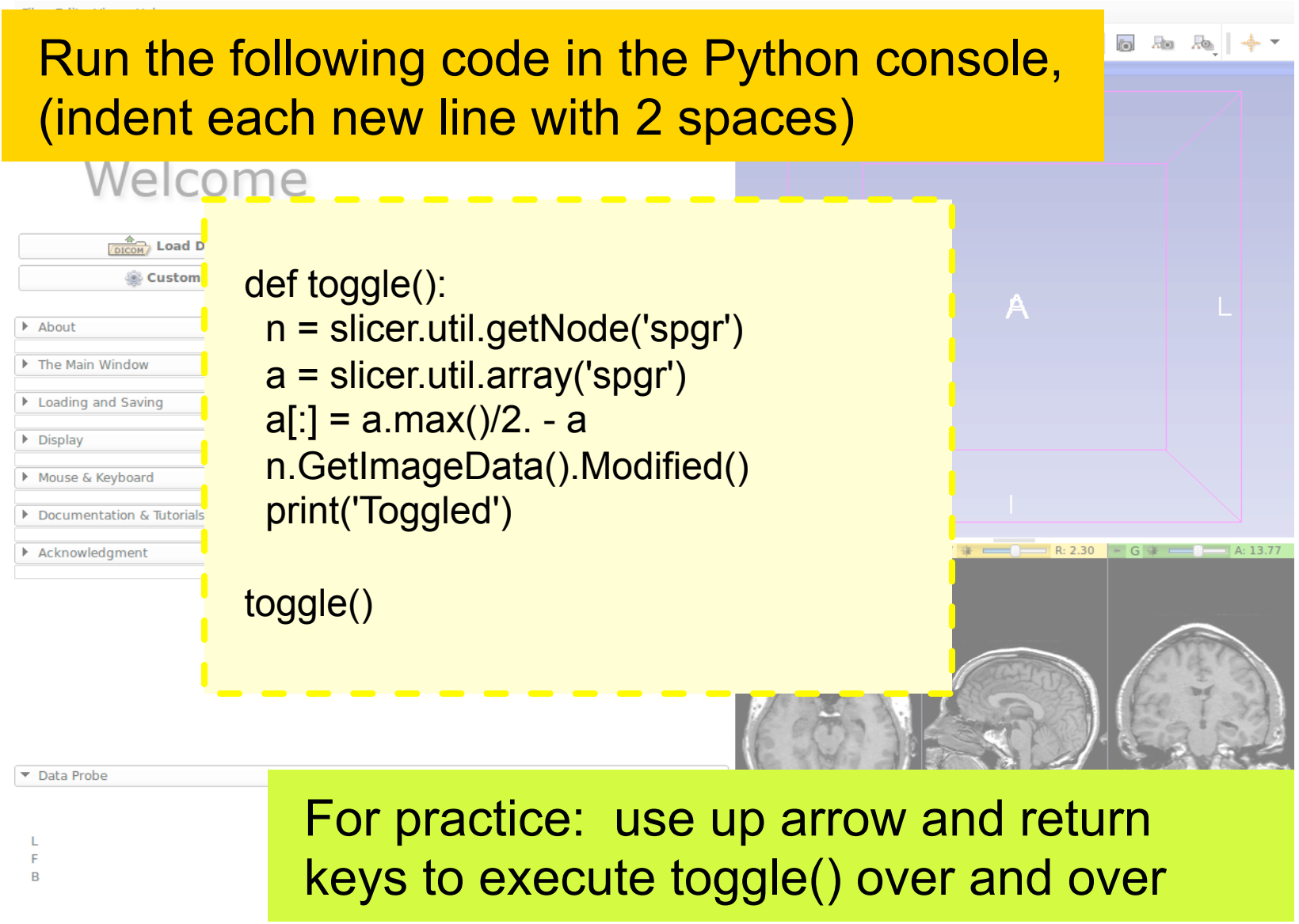


Access to MRML and Arrays



Manipulating Arrays

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

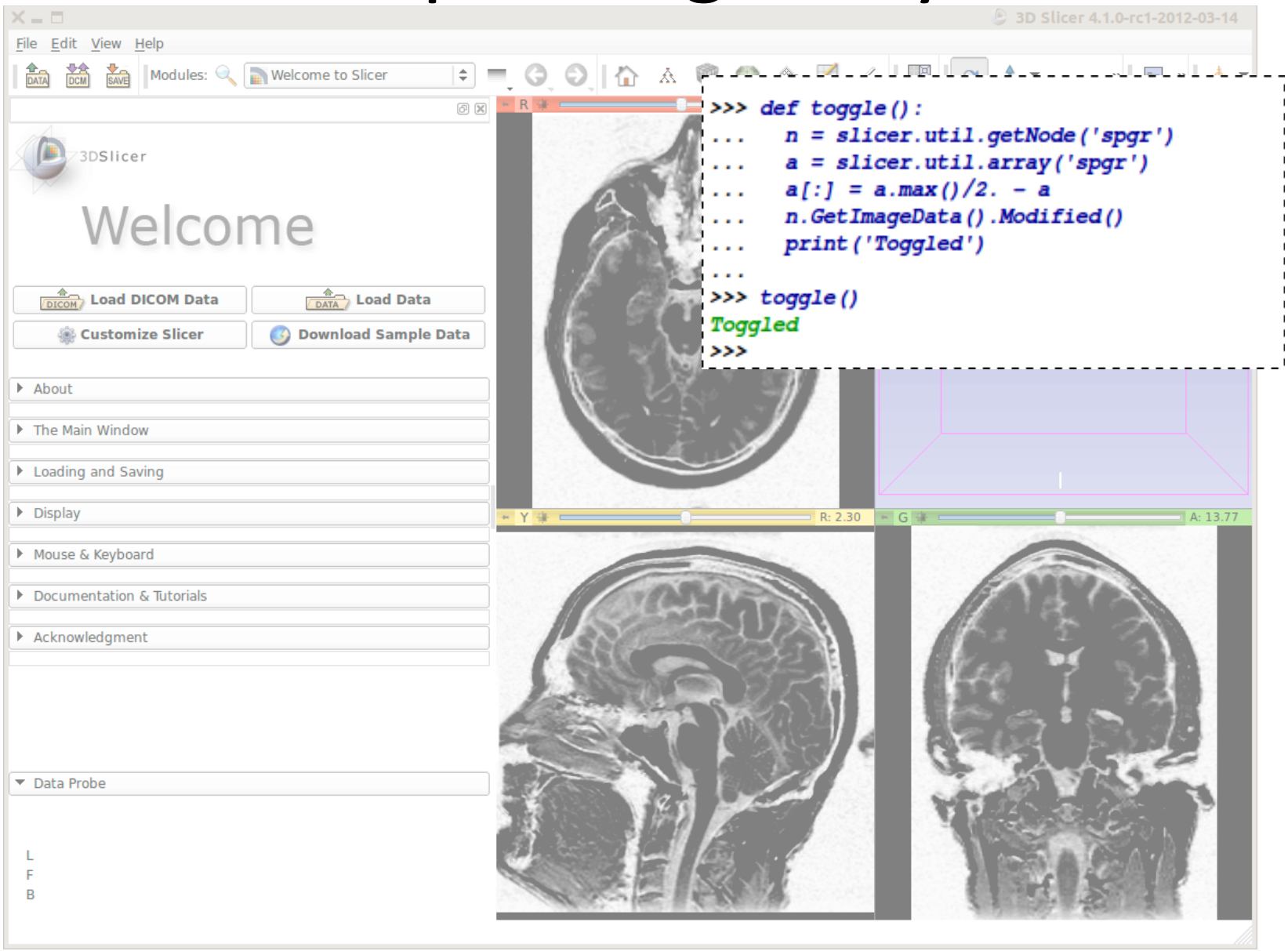



```
def toggle():
  n = slicer.util.getNode('spgr')
  a = slicer.util.array('spgr')
  a[:] = a.max()/2. - a
  n.GetImageData().Modified()
  print('Toggled')

toggle()
```

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

Manipulating Arrays



The 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

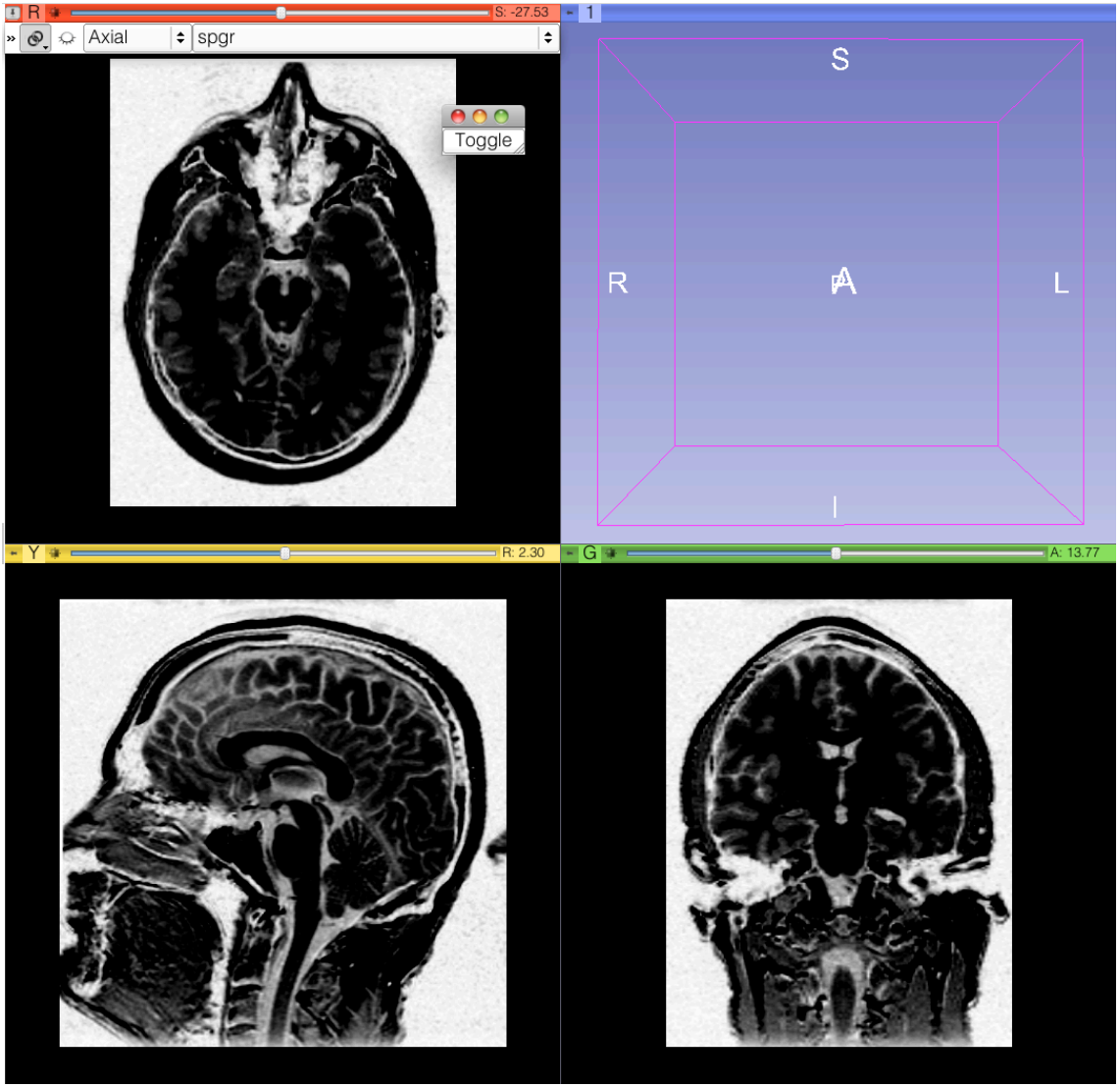
Qt GUI in Python

Run the following code in the Python console

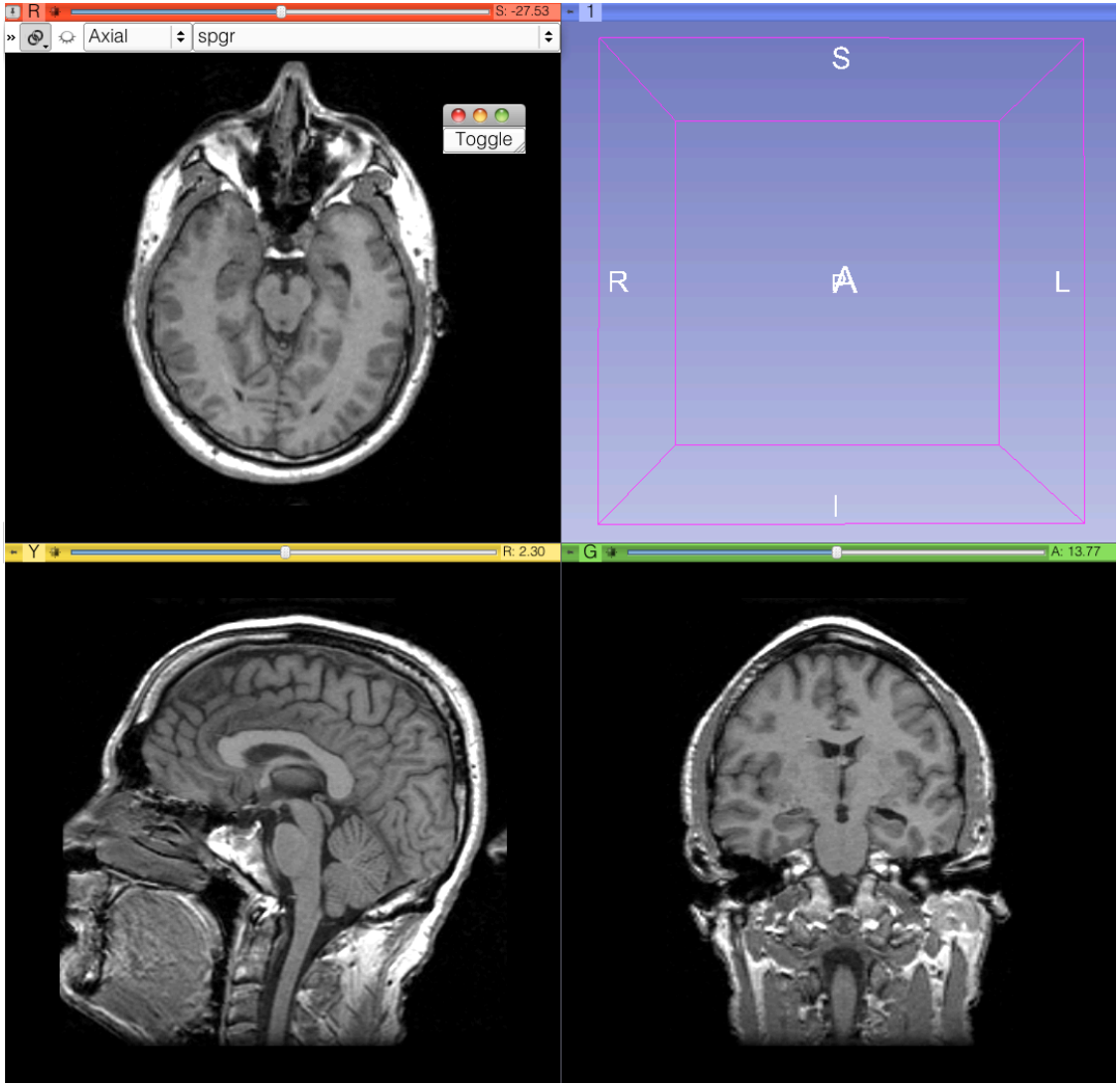
```
b = qt.QPushButton('Toggle')
b.connect('clicked()', toggle)
b.show()
```

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

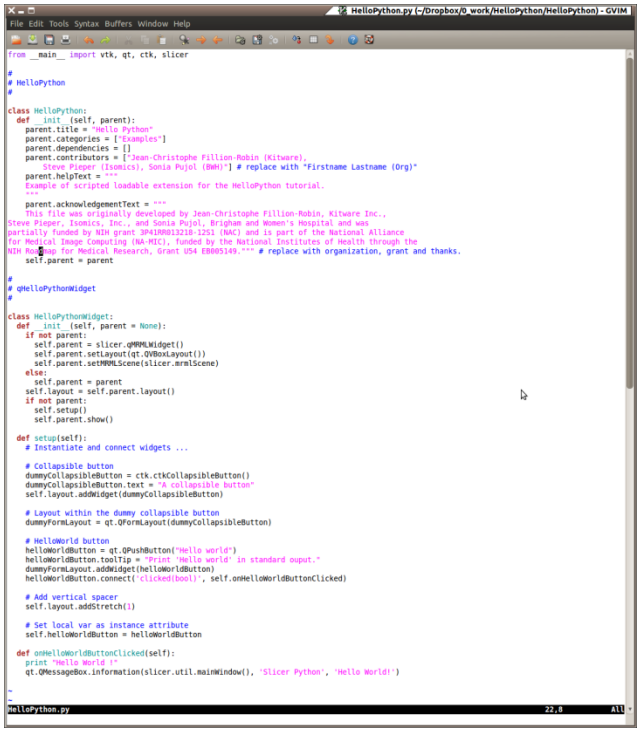


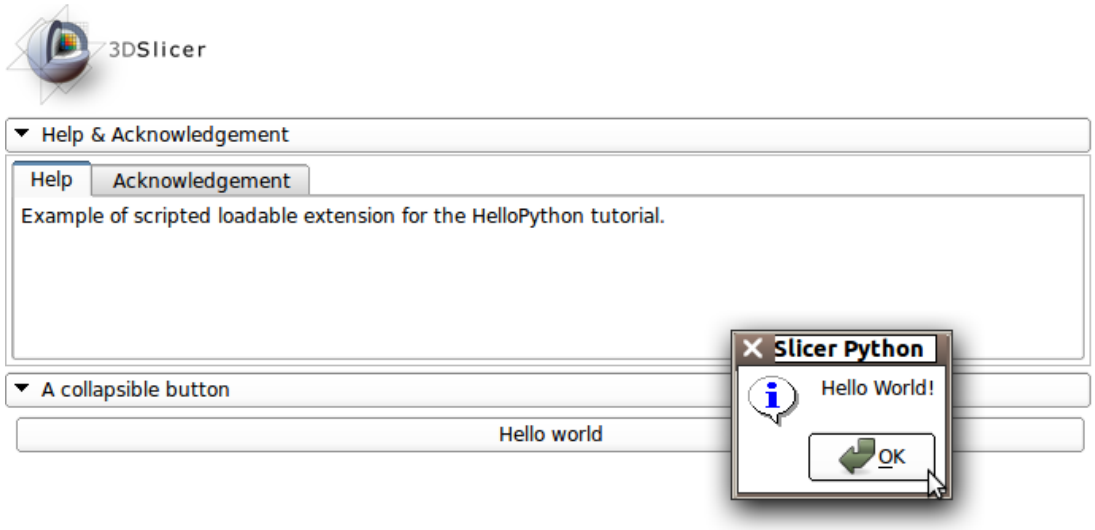
In More Detail

- 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)

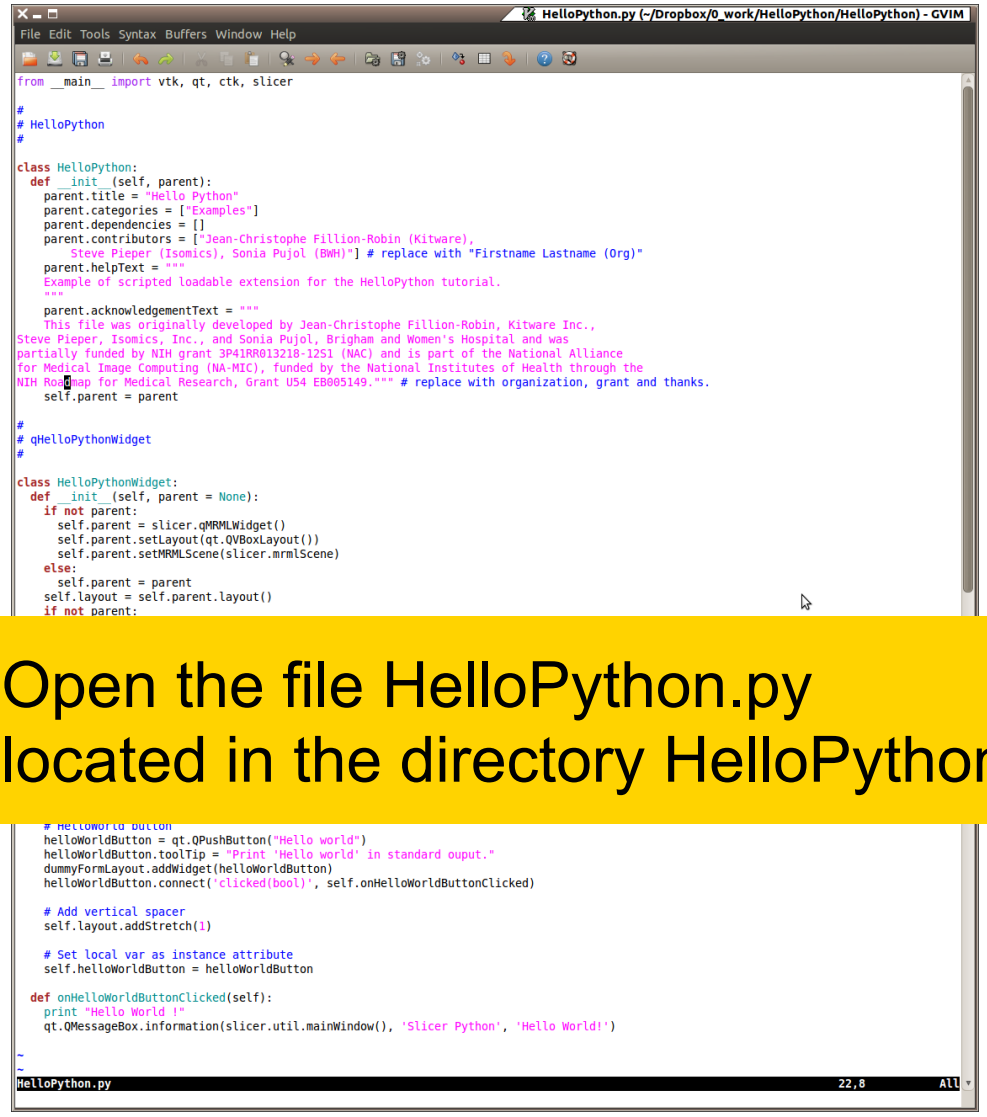
(*) Qt: http://qt.nokia.com
(**) PythonQt: http://pythonqt.sf.net /F.Link (MeVis)





PART B: INTEGRATION OF THE HELLOPYTHON TUTORIAL TO SLICER4

HelloPython.py



Open the file HelloPython.py
located in the directory HelloPython

HelloPython.py

Module Description

Module GUI

Processing Code

HelloPython.py (~/Dropbox/0_work/HelloPython/HelloPython) - GVIM

File Edit Tools Syntax Buffers Window Help

```
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()
      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.toolTip = "Print 'Hello world' in standard ouput."
    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

This code is provided in the template

```
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
```

constructor

Module GUI

Add this Text in section A

```
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.toolTip = "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
```

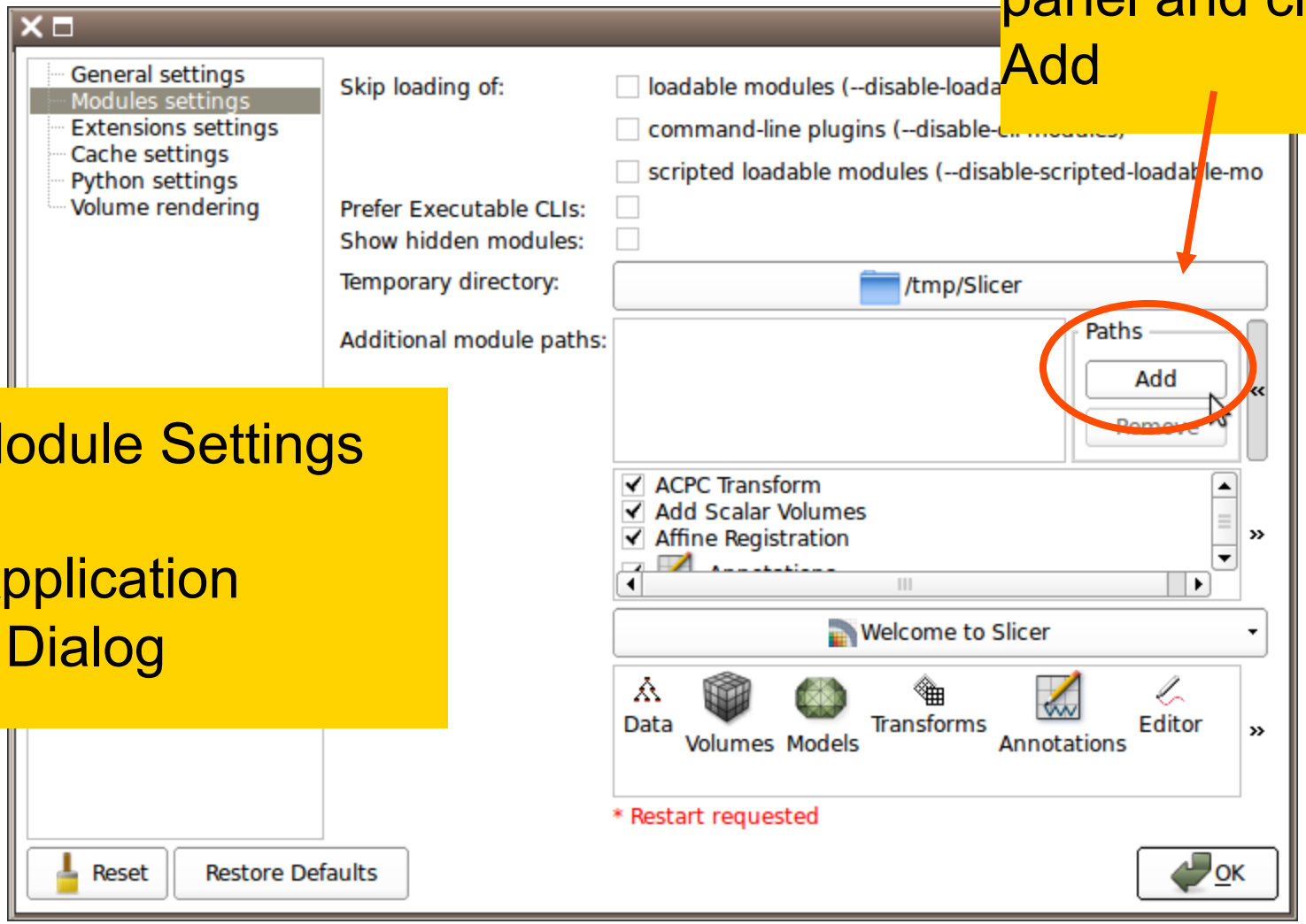
Processing Code

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

Add this Text in section B

Integrating HelloPython

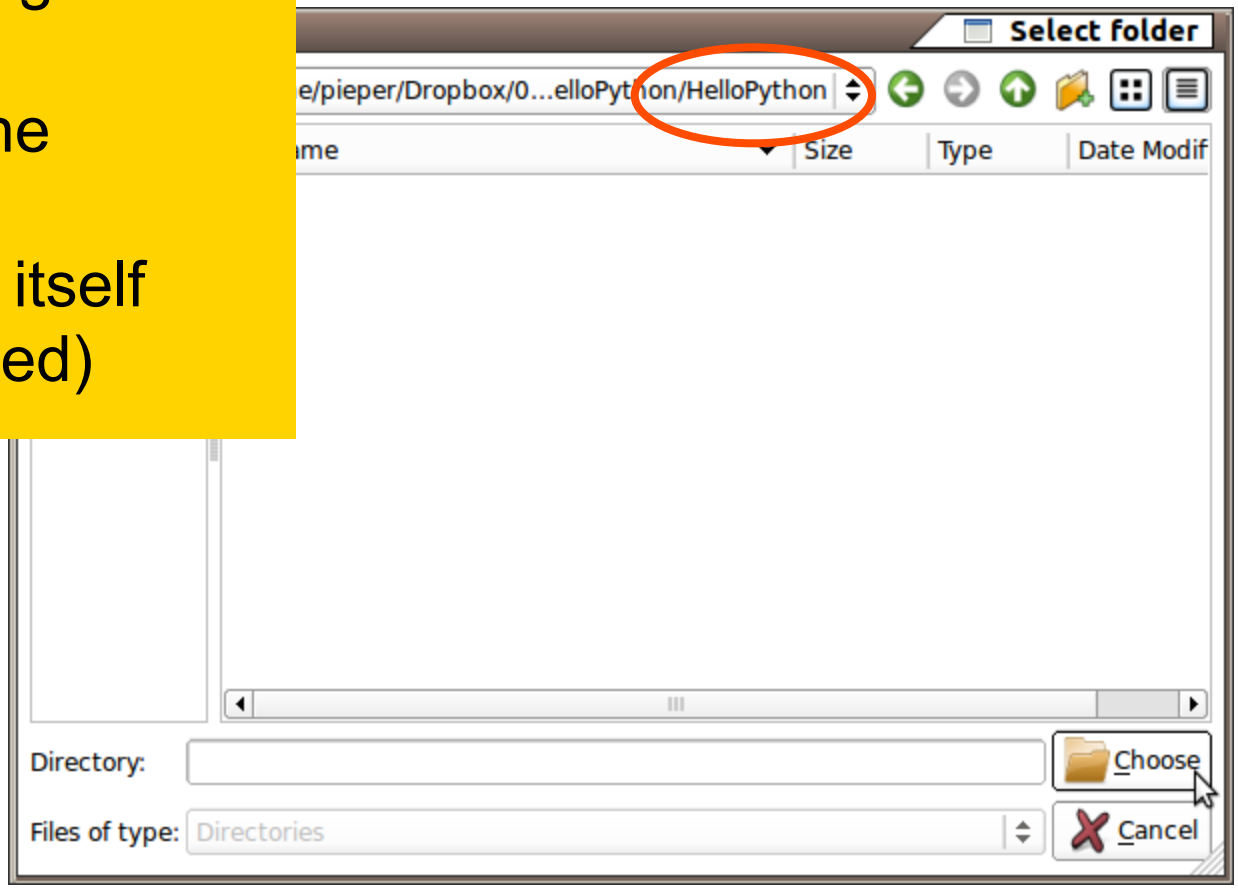
Open the side panel and click Add



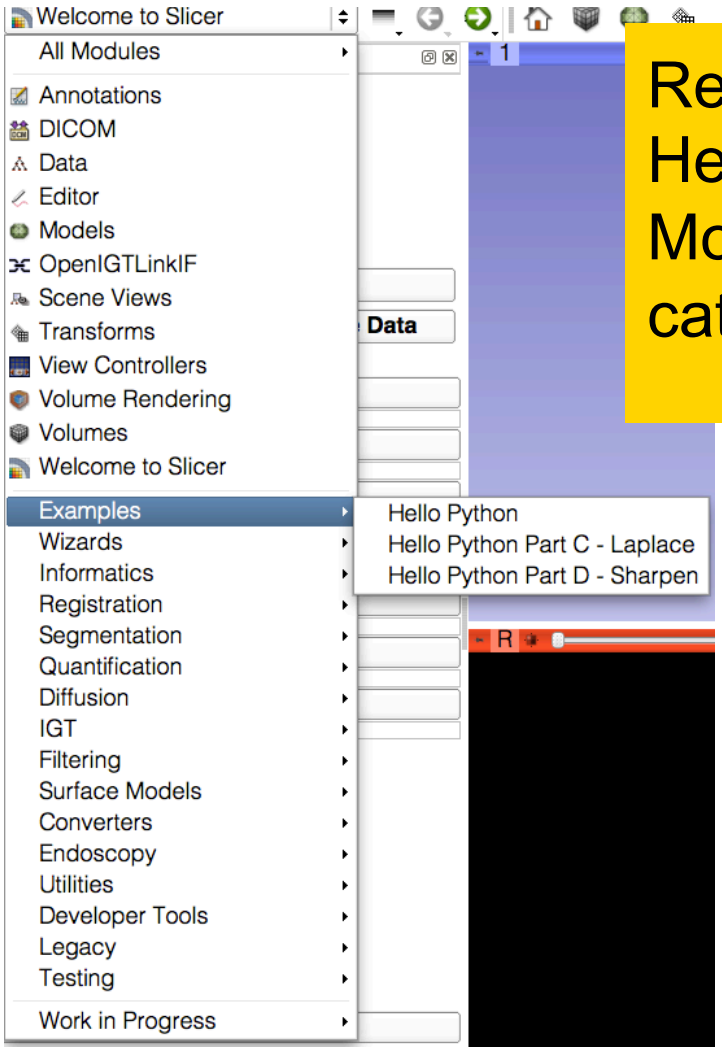
Select Module Settings from the Edit -> Application Settings Dialog

Integrating HelloPython

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

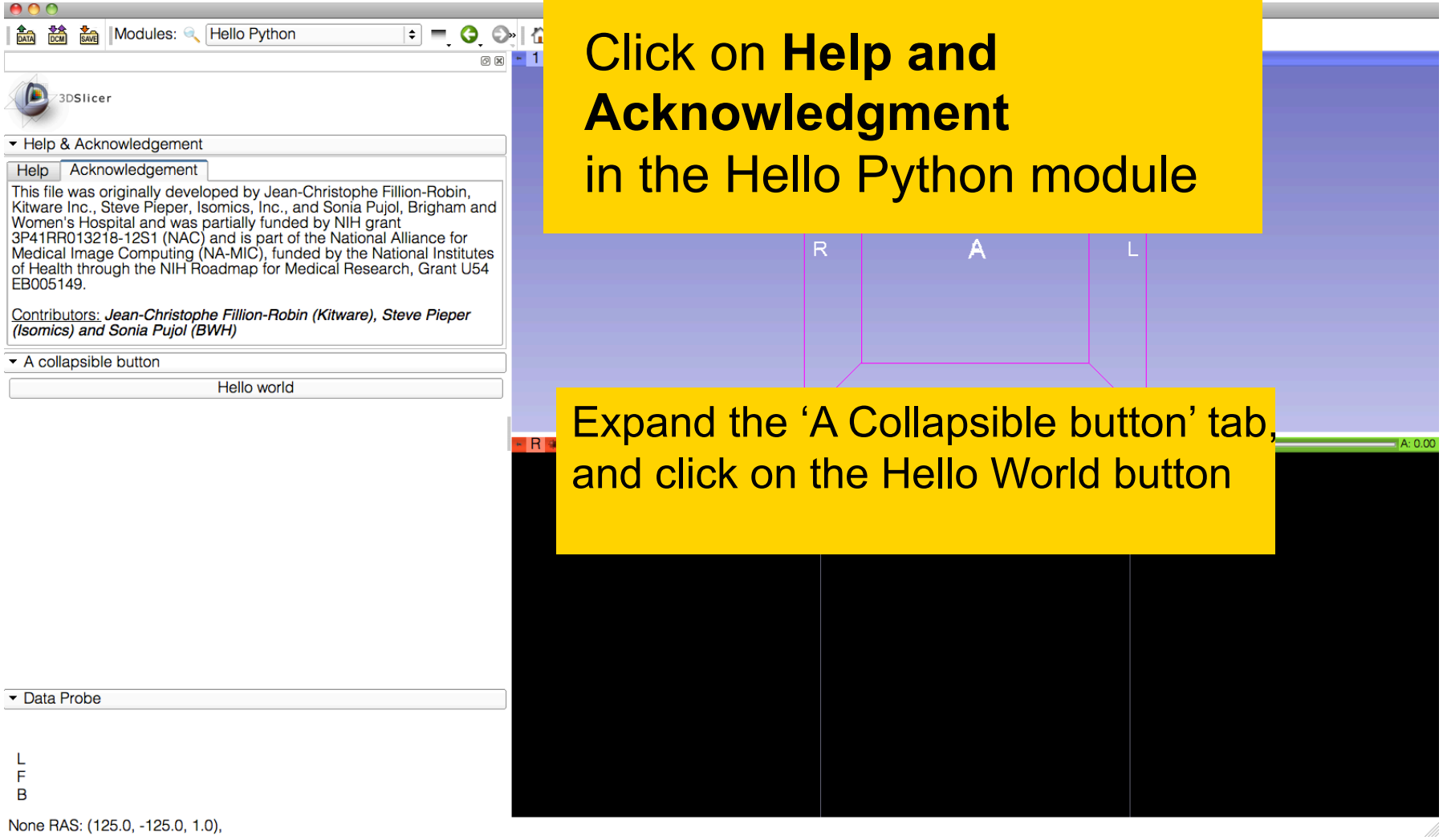


HelloPython in Slicer



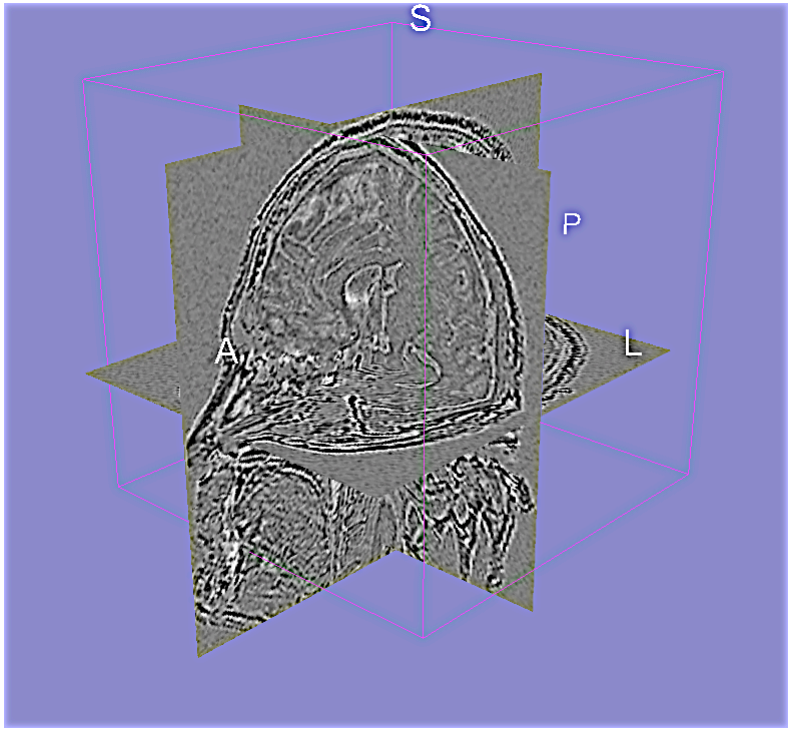
Restart Slicer when prompted. Hello Python is now in the Modules Menu, under the category **Examples**

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



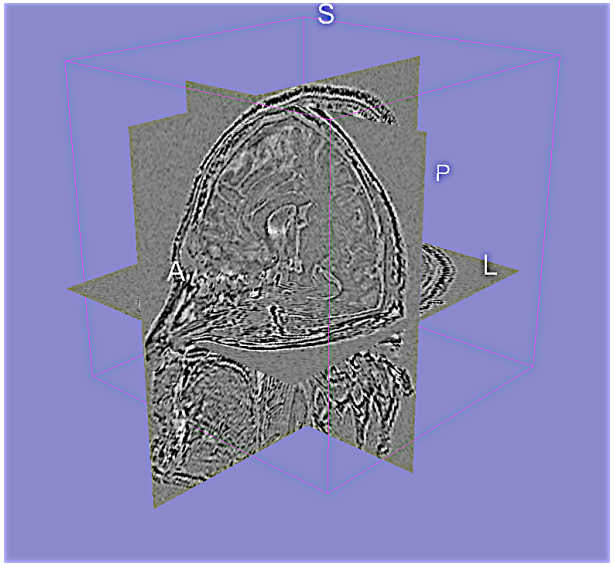
Part C:

Implementing the Laplace* Operator

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

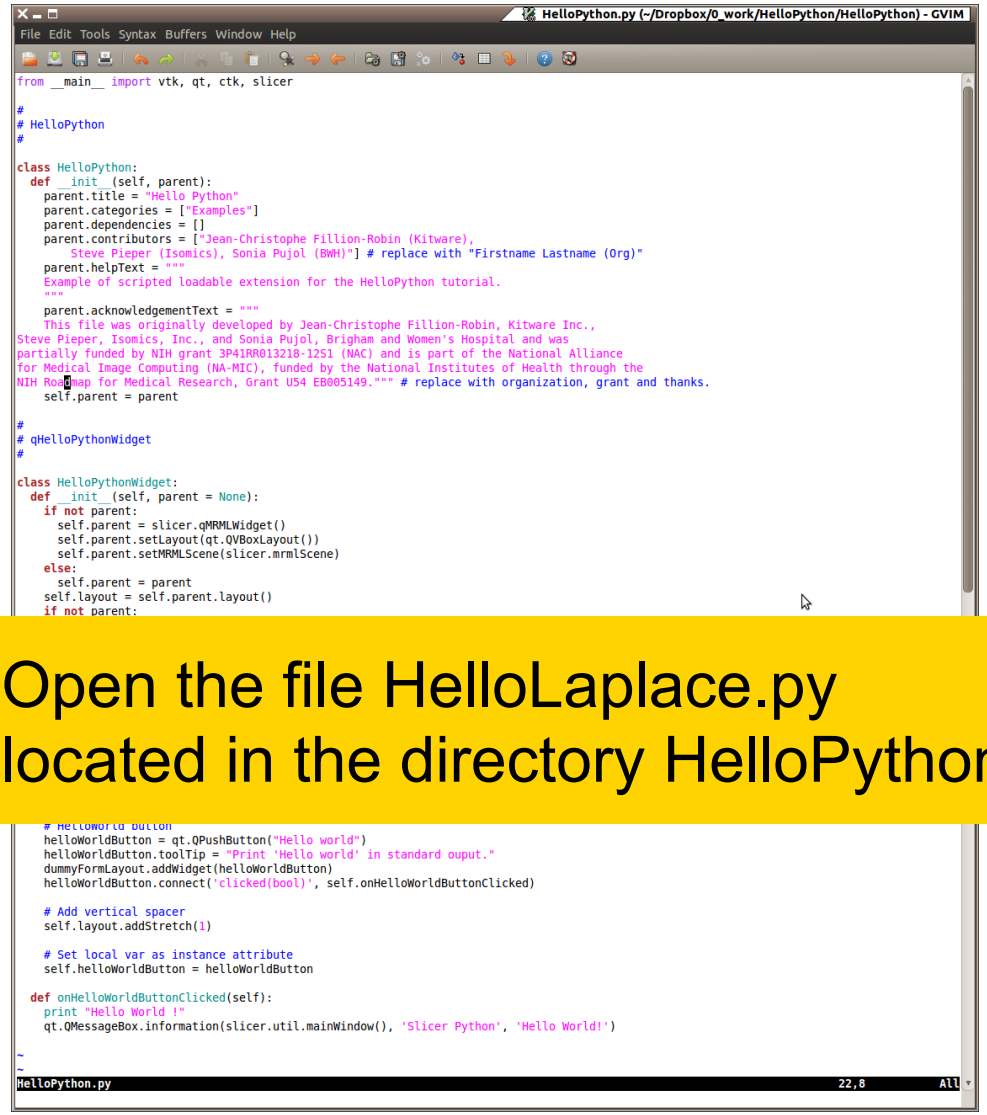
Overview

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



Open the file HelloLaplace.py
located in the directory HelloPython

Module GUI (Part 1)

This code is provided in the template

```
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.nodeTypes = ( ("vtkMRMLScalarVolumeNode"), "" )
  self.inputSelector.addEnabled = False
  self.inputSelector.removeEnabled = False
  self.inputSelector.setMRMLScene( slicer.mrmlScene )
  self.inputFrame.layout().addWidget(self.inputSelector)
```

Module GUI (Part 2)

This code is provided in the template

```
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.nodeTypes = ( ("vtkMRMLScalarVolumeNode"), "" )
self.outputSelector.setMRMLScene( slicer.mrmlScene )
self.outputFrame.layout().addWidget(self.outputSelector)

# Apply button
laplaceButton = qt.QPushButton("Apply Laplace")
laplaceButton.toolTip = "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
```

In More Detail

- **CTK i**s 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

Add this 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)
```

In More Detail

- **vtkImageLaplacian** is a vtkImageAlgorithm operates on vtkImageData (see http://vtk.org)

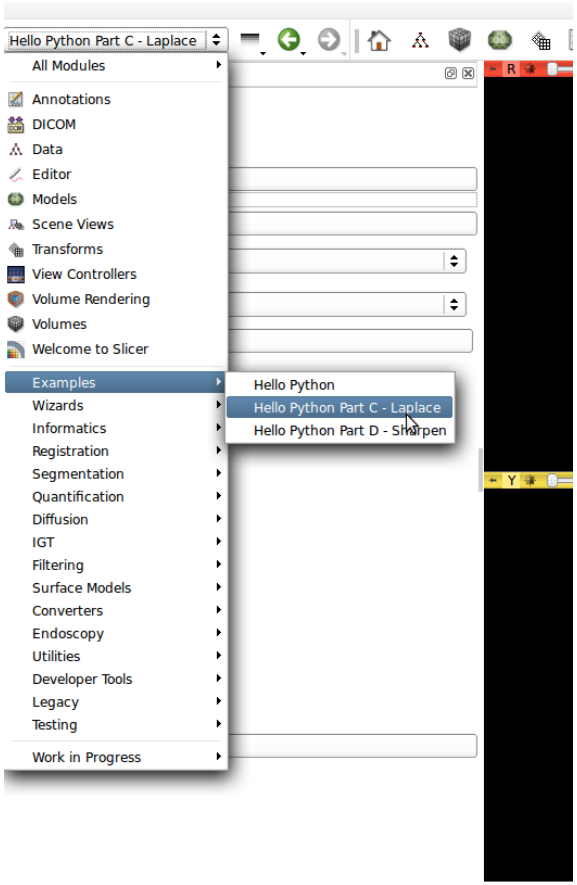
- **vtkMRMLScalarVolumeNode** is a Slicer MRML class that contains vtkImageData, plus orientation information ijkToRAS matrix (see http://www.slicer.org/slicerWiki/index.php/Coordinate_systems)

In More Detail (Continued)

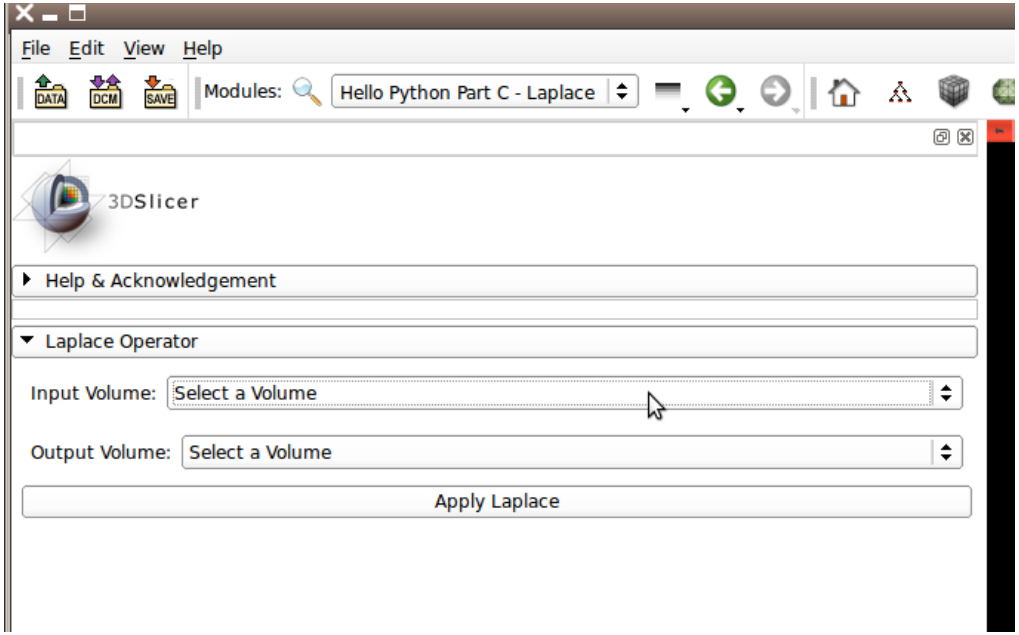
Global **slicer** package gives python access to:
1- GUI (via **slicer.app**)
2- modules (via **slicer.modules**)
3- data (via **slicer.mrmlScene**)

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

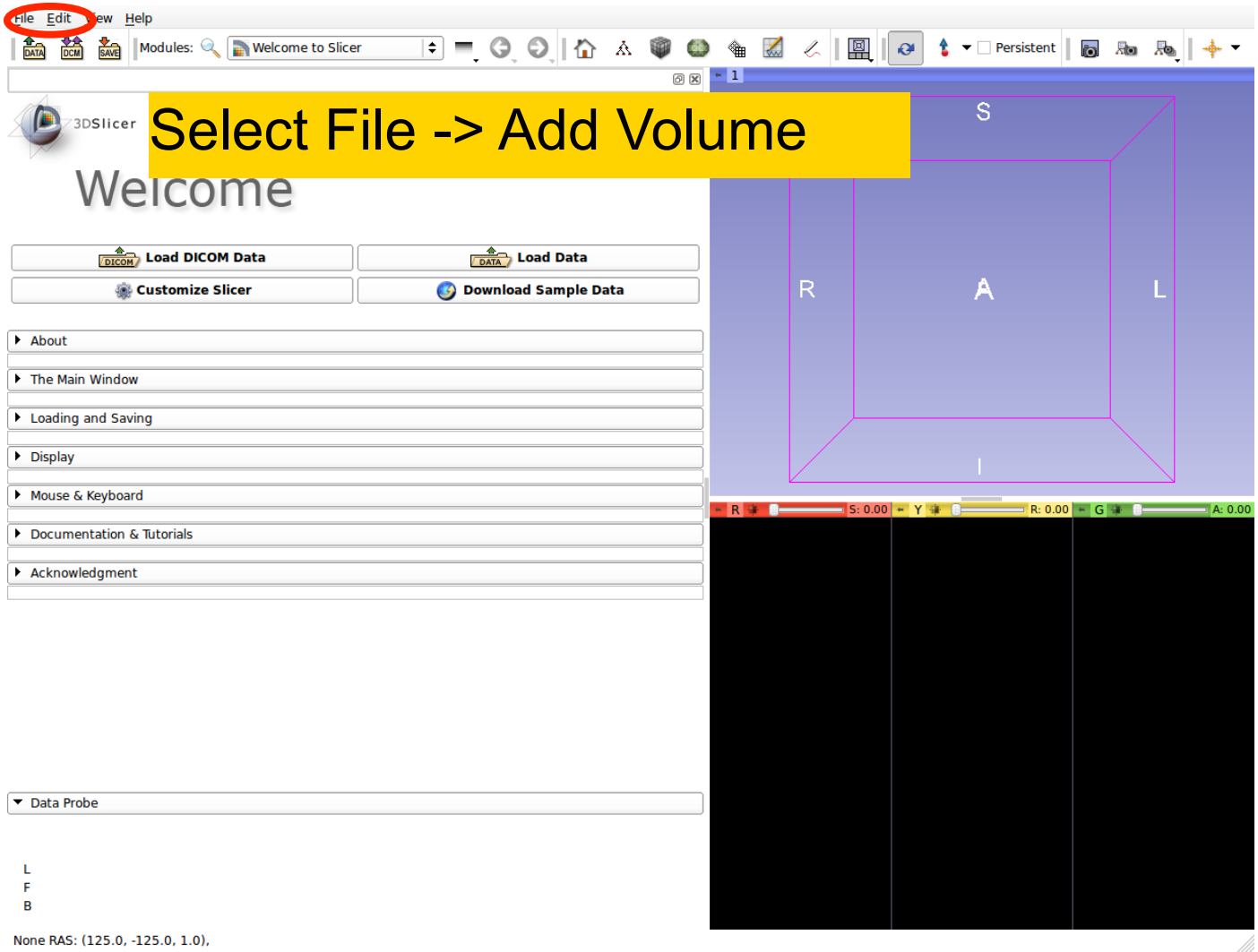
Go To Laplace Module



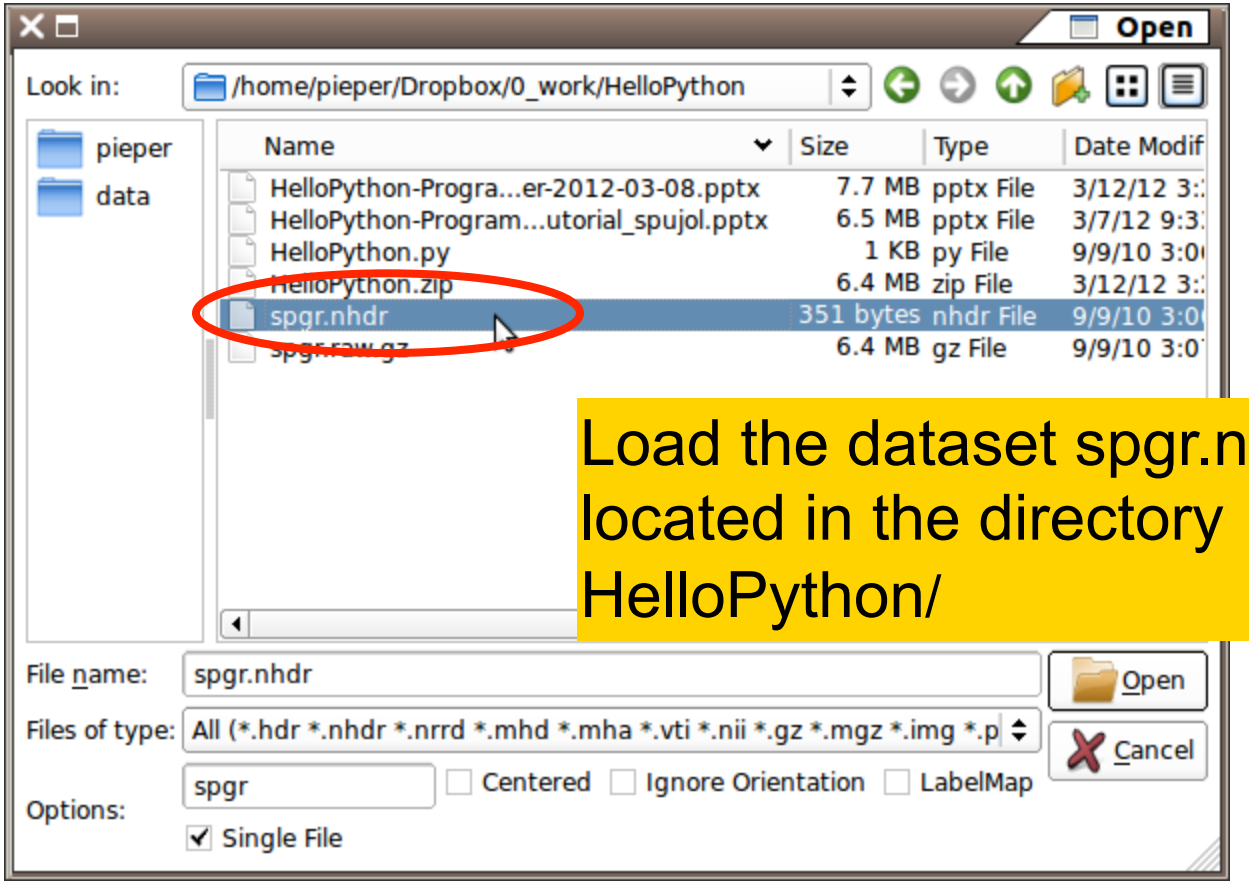
Re-start Slicer and select module. Note that combobox is empty



Add Volume Dialog

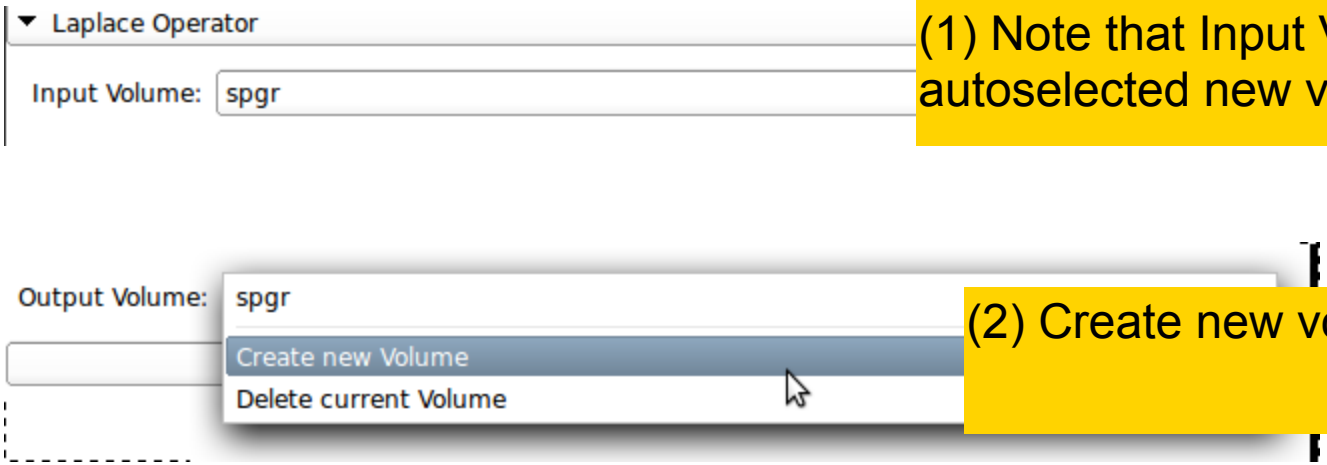


Add spgr.nhdr



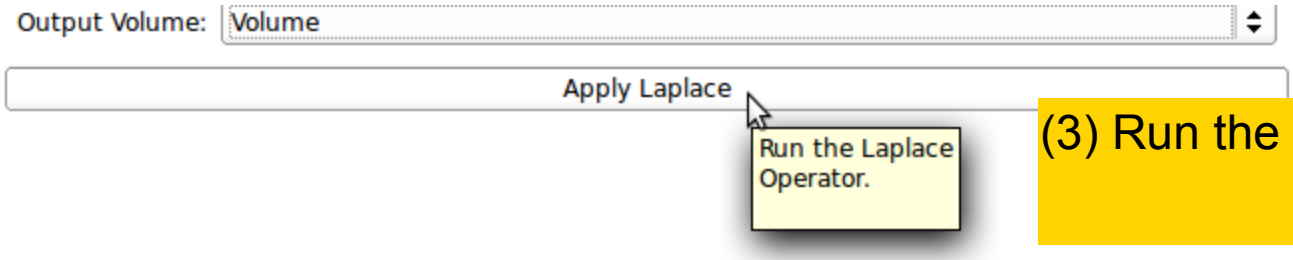
Load the dataset spgr.nhdr located in the directory HelloPython/

After Adding Volume



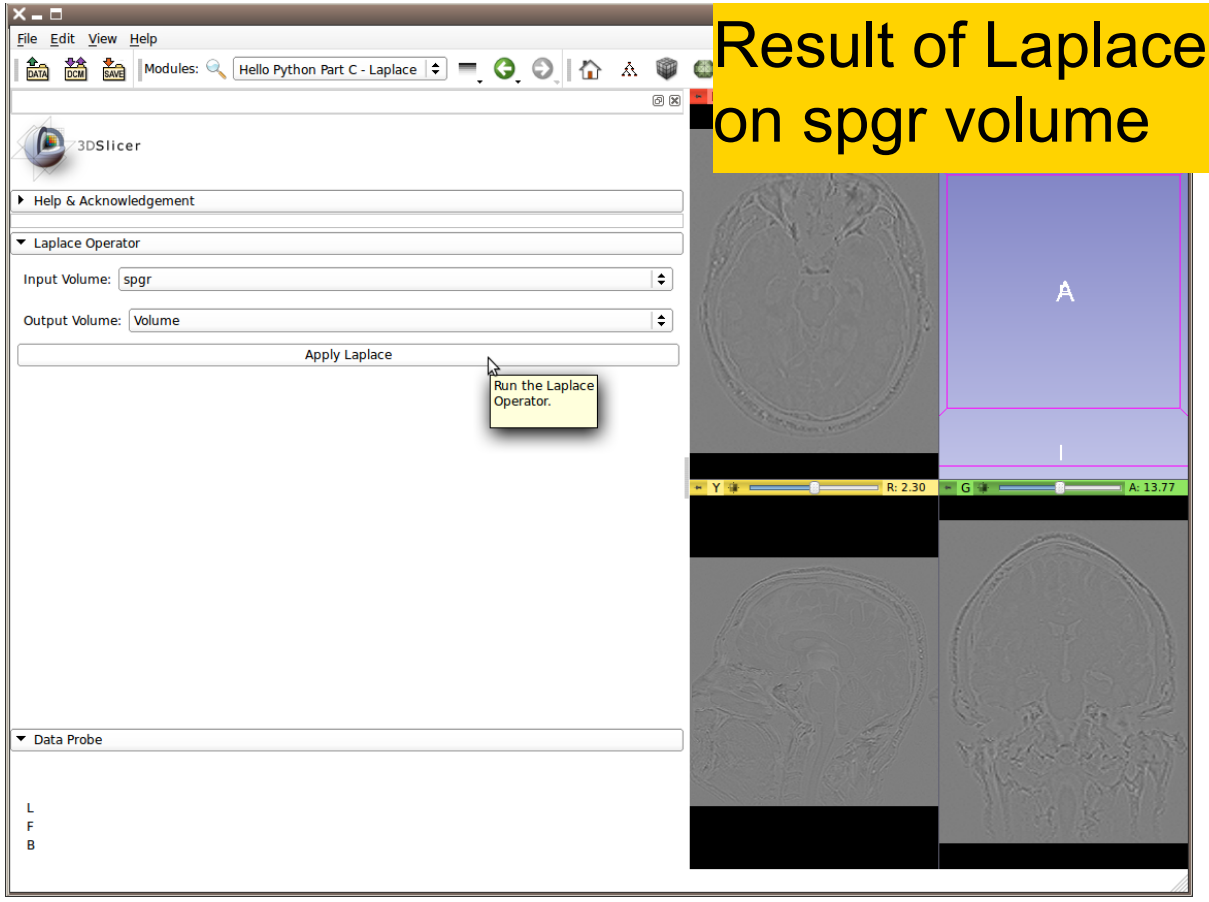
(1) Note that Input Volume combobox autoselected new volume

(2) Create new volume for output

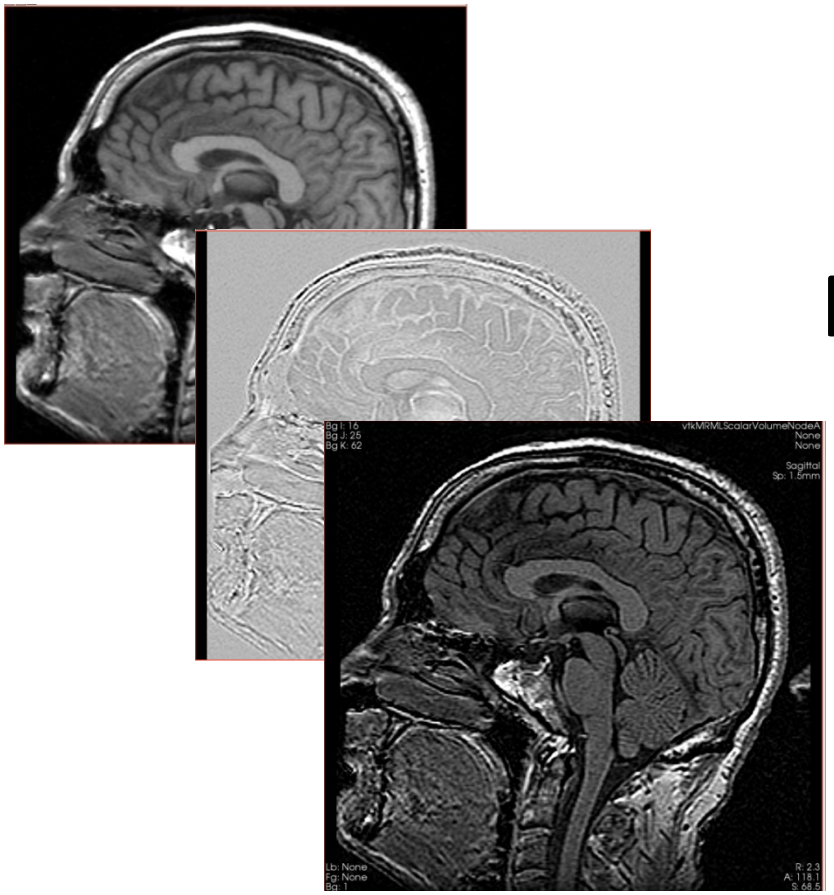


(3) Run the module

Laplace Module



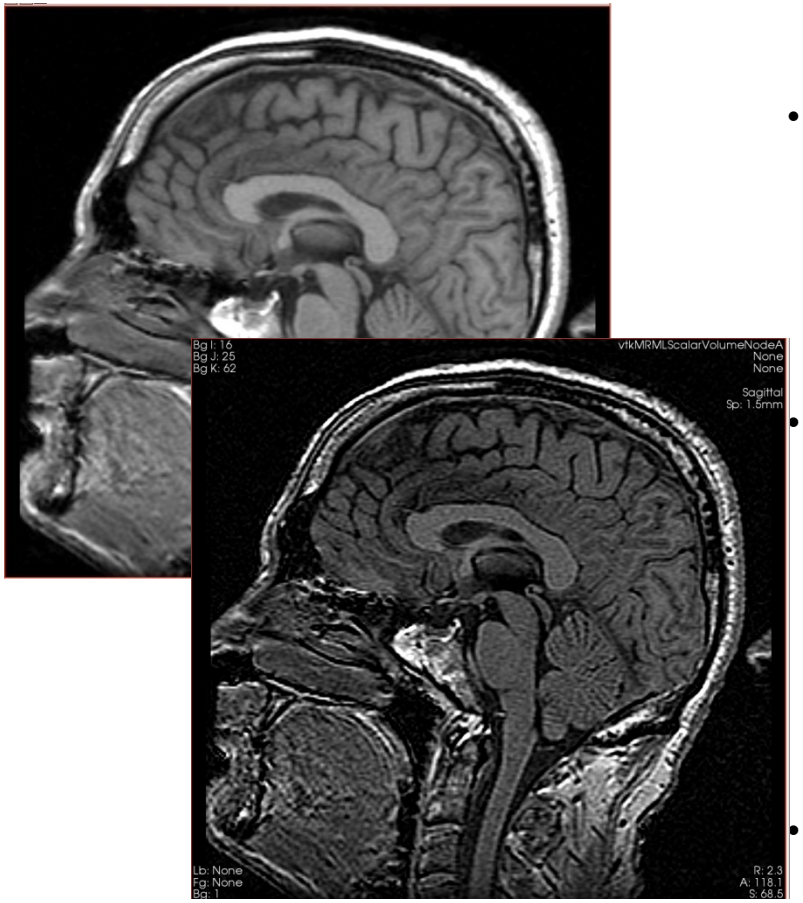
Result of Laplace Operator on spgr volume



Part D:

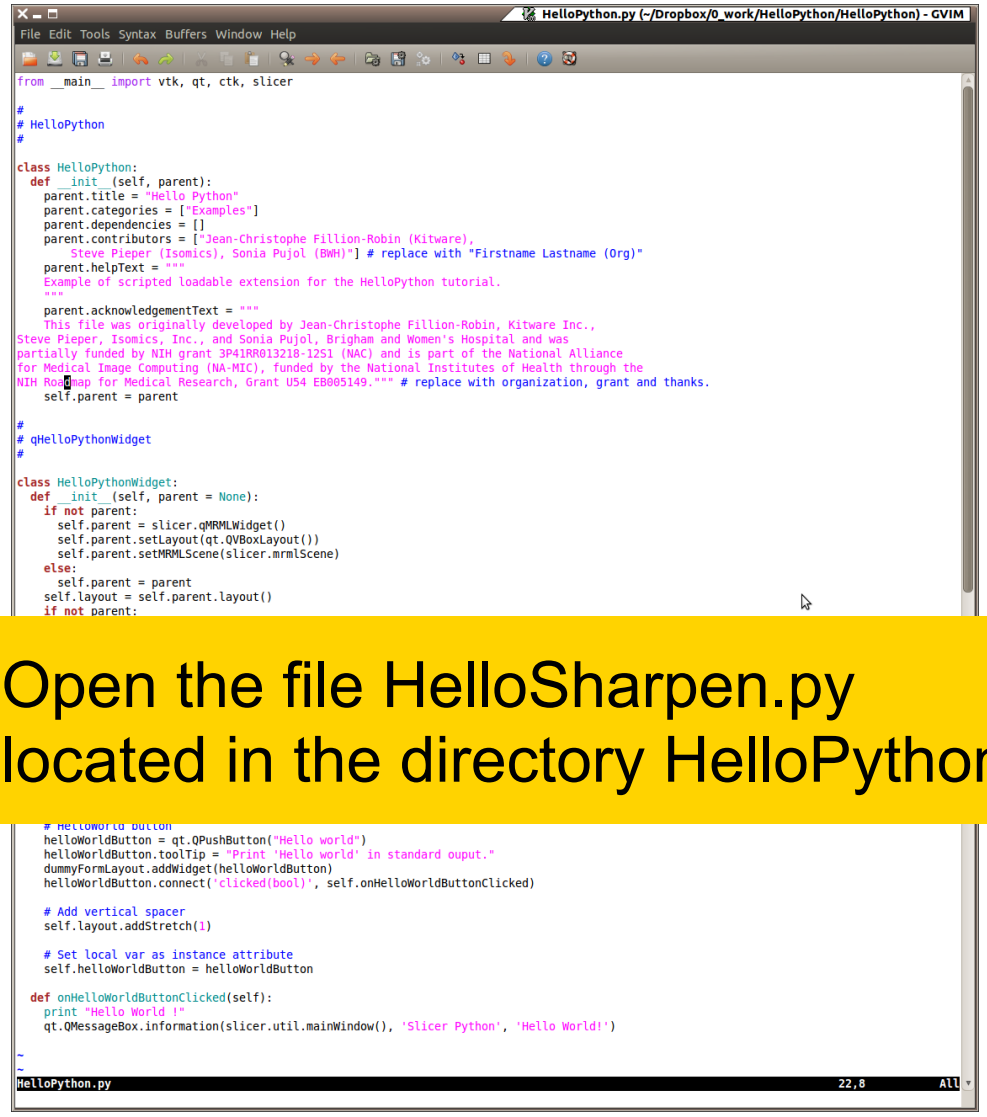
Image Sharpening with the Laplace Operator

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



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.toolTip = "When checked, subtract laplacian from input volume"
self.sharpen.checked = True
self.laplaceFormLayout.addWidget(self.sharpen)

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

Add to Processing Code

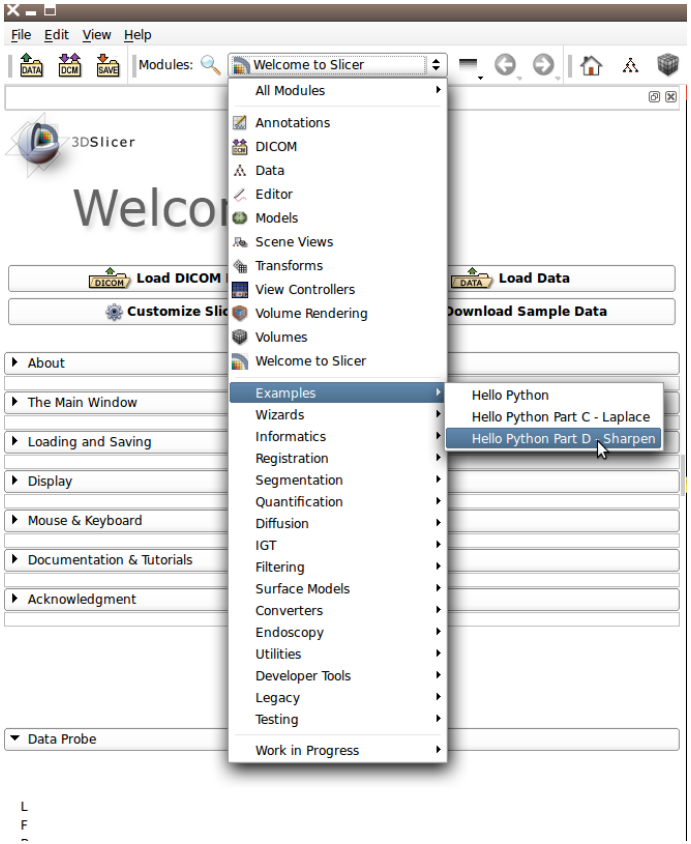
Add this Text in section B

```
    ...
    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)
```

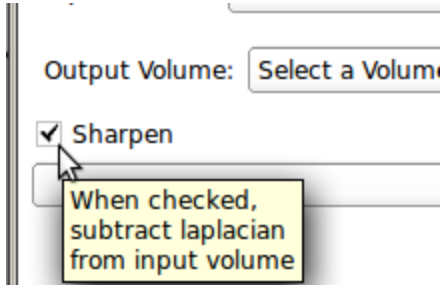
In More Detail

- **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

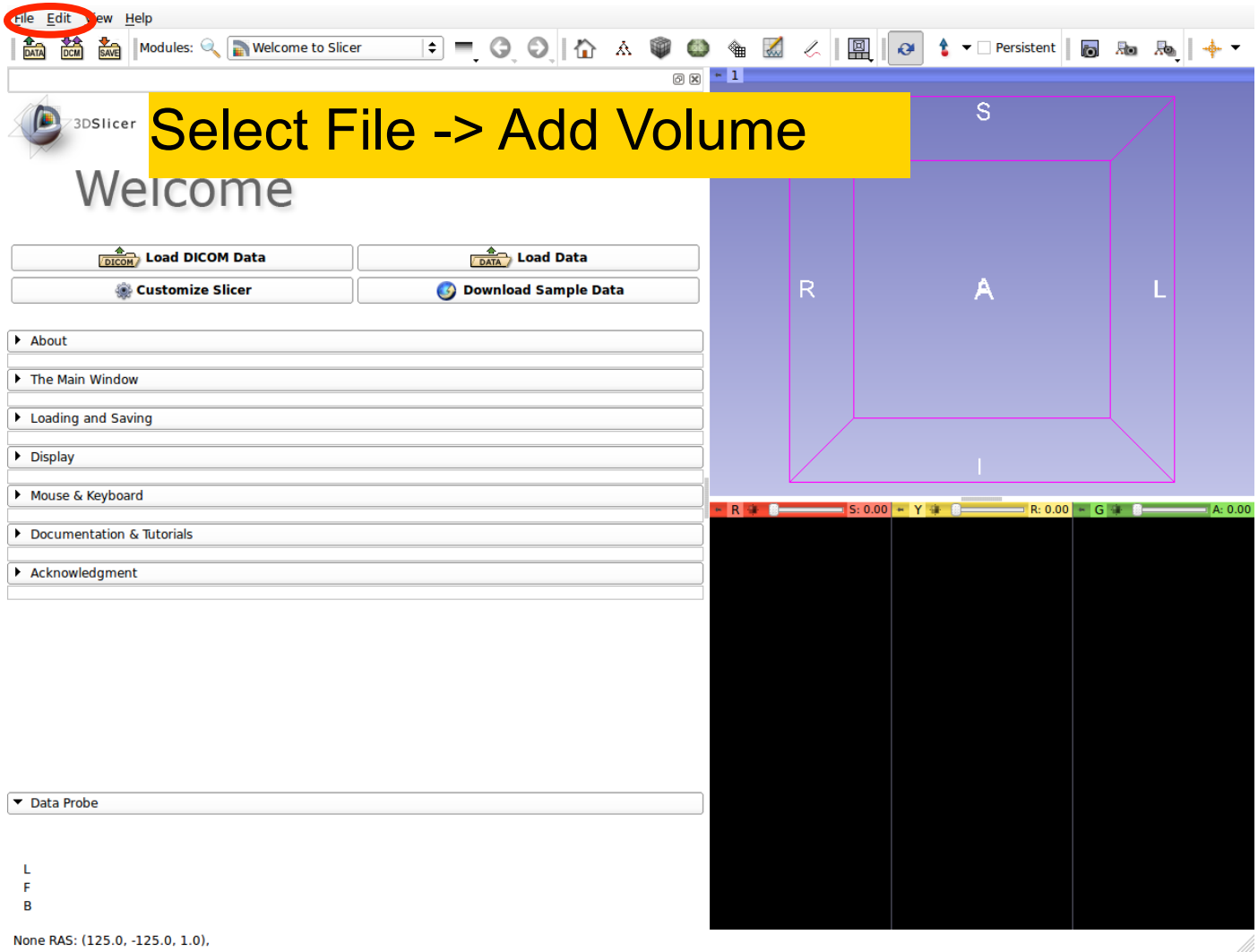
Go To Sharpen Module



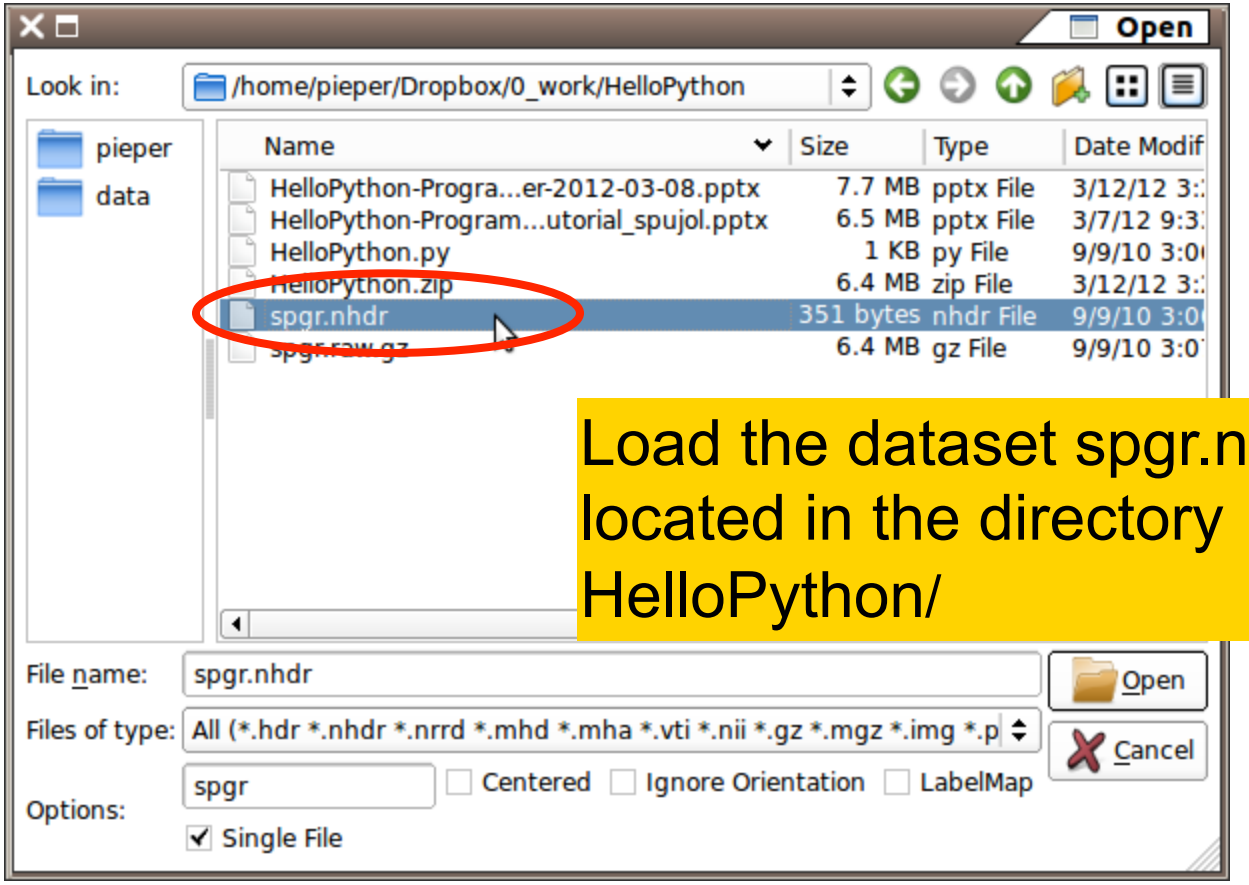
Re-start Slicer and select module. Note the new sharpen check box



Add Volume Dialog



Add spgr.nhdr

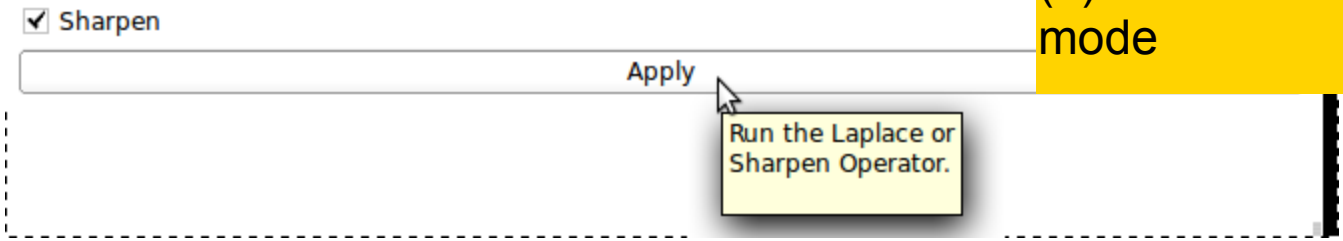


Load the dataset spgr.nhdr located in the directory HelloPython/

After Adding Volume

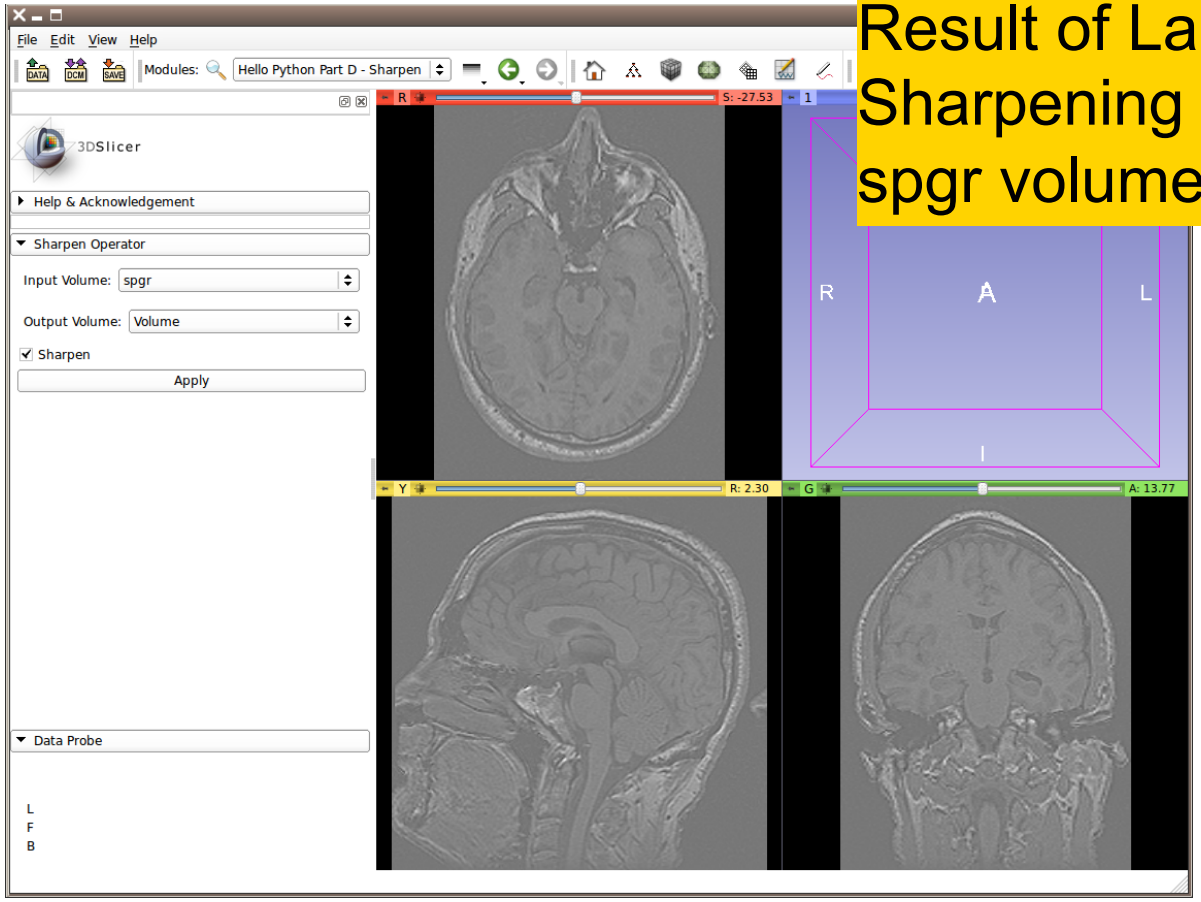
(1) Note that Input Volume combobox autoselected new volume

(2) Create new volume for output



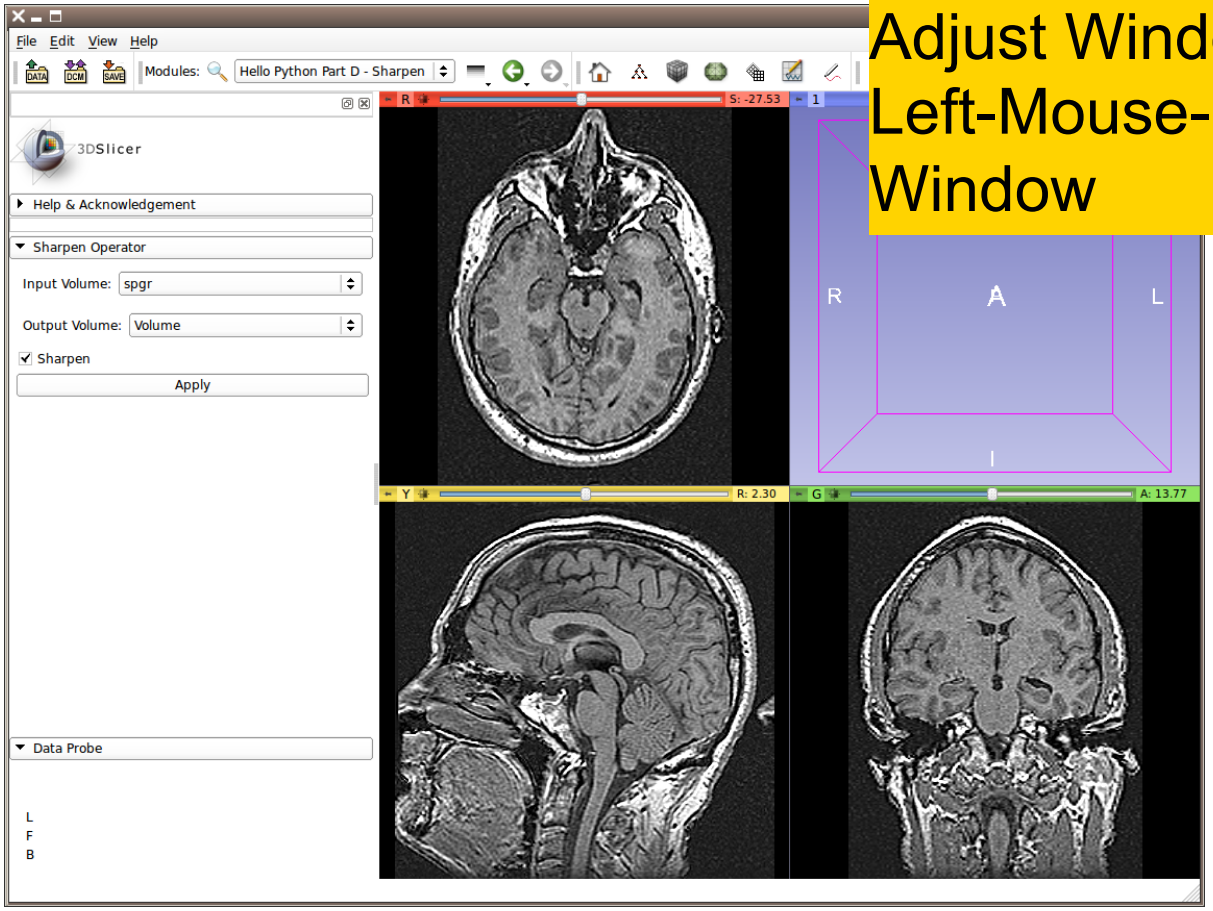
(3) Run the module in Sharpen mode

Sharpen Module



Result of Laplacian Sharpening Operator on spgr volume

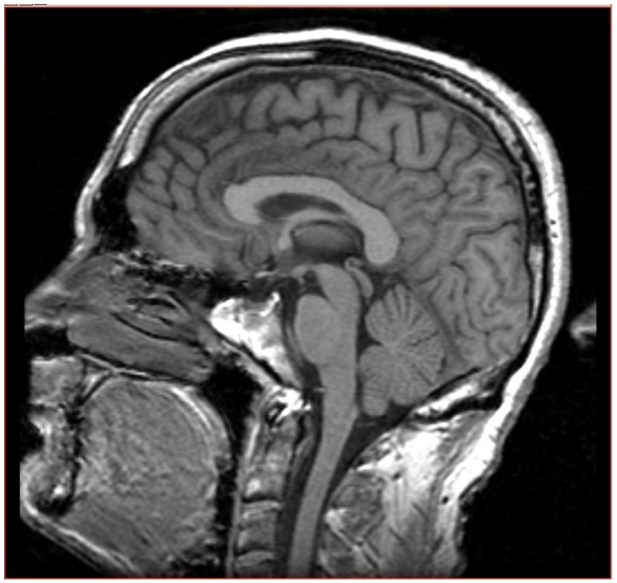
Sharpen Module



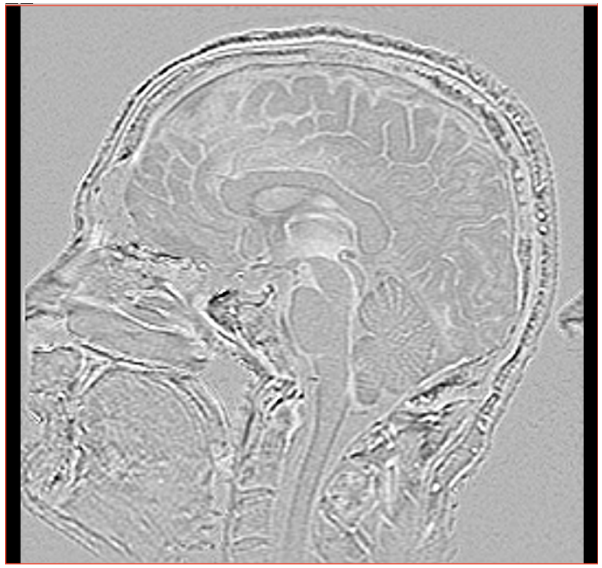
Adjust Window/Level with Left-Mouse-Drag in Slice Window

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

Acknowledgments

National Alliance for Medical Image Computing
NIH U54EB005149

Neuroimage Analysis Center
NIH P41RR013218