



Tutorial de Desenvolvimento do *Slicer*: Programando no *Slicer*

Sonia Pujol, Ph.D.

Professora Assistente de Radiologia
Diretora de Treinamento e Educação do 3D Slicer
Brigham and Women's Hospital
Escola de Medicina de Harvard

Steve Pieper, Ph.D.
Arquiteto-chefe do 3D Slicer
Isomics Inc.

Objetivo do Tutorial



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



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



Este tutorial é uma introdução ao *Python interactor* e ao conjunto de *widgets Qt* na versão 5 do 3D Slicer.

Roteiro do Tutorial



Parte 1: Visão Geral dos Módulos do 3D Slicer



Parte 2: Familiarizando-se com o ambiente Python no 3D Slicer

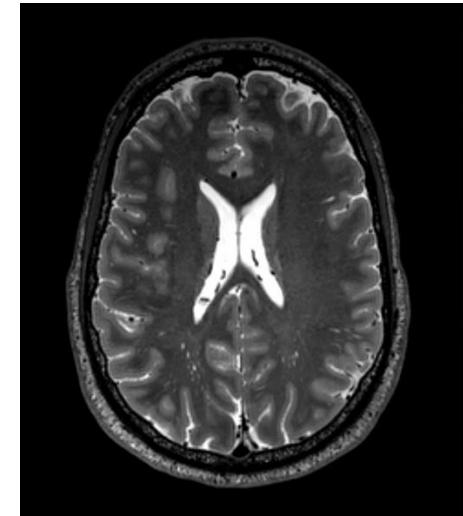
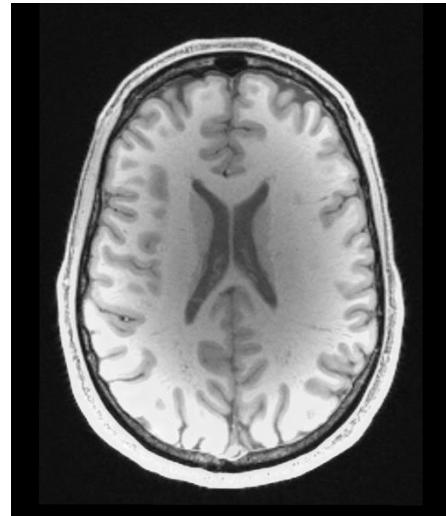
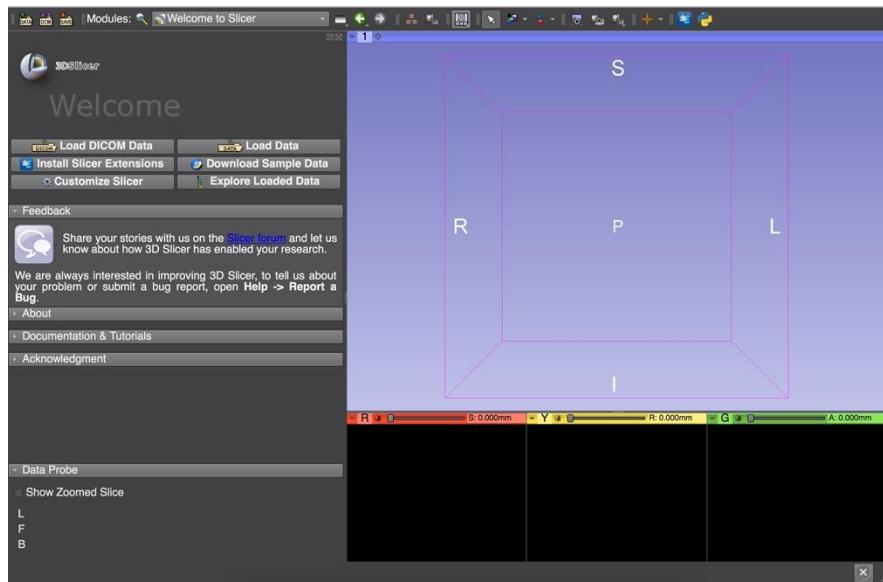


Parte 3: Familiarizando-se com o conjunto de widgets Qt no 3D Slicer

Aviso

- O 3D Slicer é um *software* livre de código aberto distribuído sob uma licença no estilo BSD.
- O *software* não é aprovado pela FDA nem possui marcação CE, sendo destinado apenas para uso em pesquisa.

Materiais de Tutorial

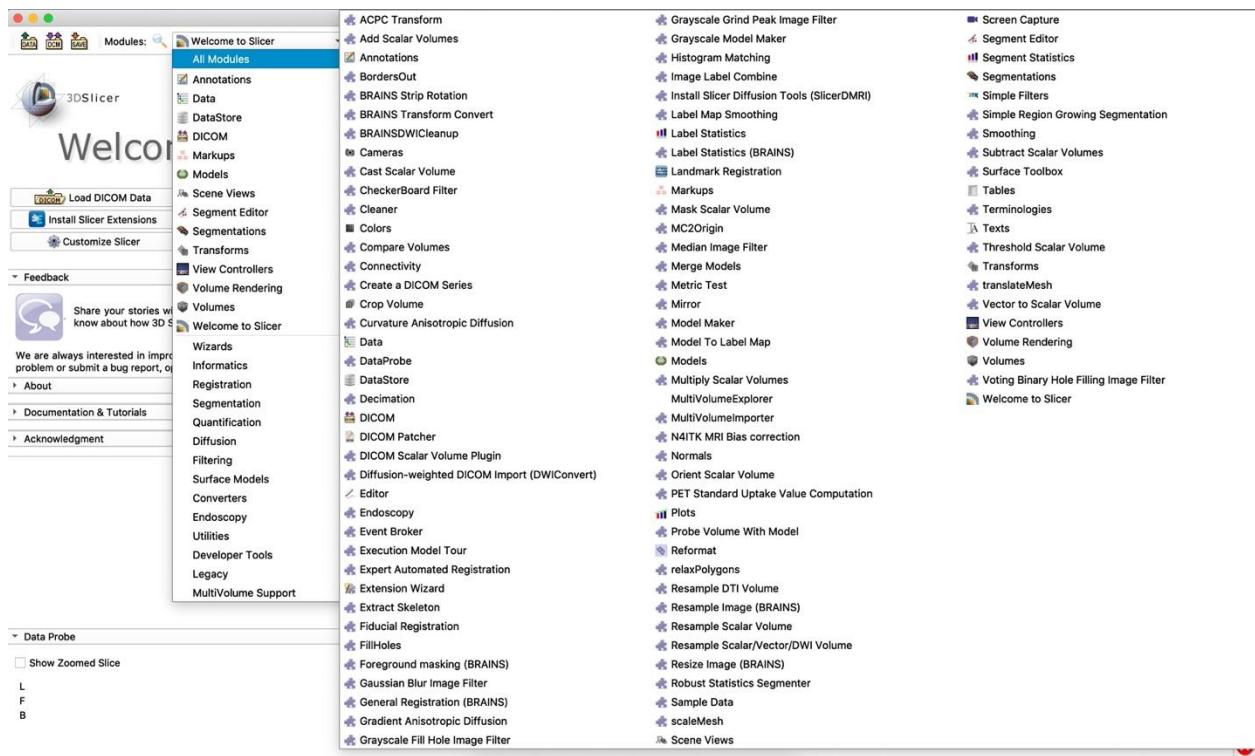


SlicerProgrammingTutorialData.zip

3D Slicer versão 5

Parte 1

Um panorama dos módulos do Slicer



3D Slicer

The screenshot shows the GitHub repository page for 'Slicer'. The top navigation bar includes links for Code, Issues (544), Pull requests (52), Actions, Wiki, Security, and Insights. The repository name 'Slicer' is shown with a public status. Below the header, there are buttons for Watch (43), Fork (575), and Star (1.8k). The main content area displays the repository's structure with 'main' selected, showing 12 branches and 35 tags. A search bar and a 'Go to file' button are available. A green 'Code' button is prominent. On the right, there is an 'About' section describing it as multi-platform, free open source software for visualization and image computing, with a link to www.slicer.org. The 'About' section also lists various technologies and fields: python, c-plus-plus, qt, image-processing, medical-imaging, registration, neuroimaging, segmentation, vtk, itk, national-institutes-of-health, medical-image-computing, 3d-printing, 3d-slicer, tractography, image-guided-therapy, nih, computed-tomography, kitware, and tcia-dac.

- O 3D Slicer é uma plataforma de código aberto para análise e visualização de dados de imagens médicas.
- O 3D Slicer é compilado e testado todos os dias nas plataformas Windows, Mac e Linux.
- O código-fonte está disponível gratuitamente no GitHub em <http://github.com/Slicer/Slicer>.

Módulos do Slicer

O 3D Slicer suporta três tipos de módulos:

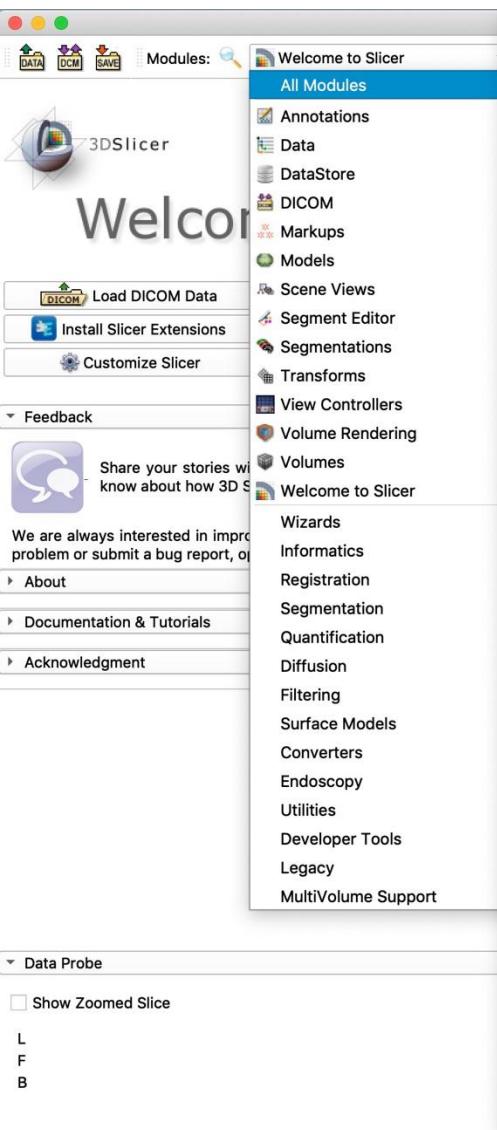
Command-line Interface (CLI) [Interface de Linha de Comando]: executável independente com argumentos de entrada/saída limitados.

Módulos Carregáveis (Plugins em C++): otimizados para computação pesada.

Foco deste tutorial

- Módulos com *scripts* (Python): recomendados para prototipagem rápida e desenvolvimento de fluxos de trabalho.

Módulos do Slicer

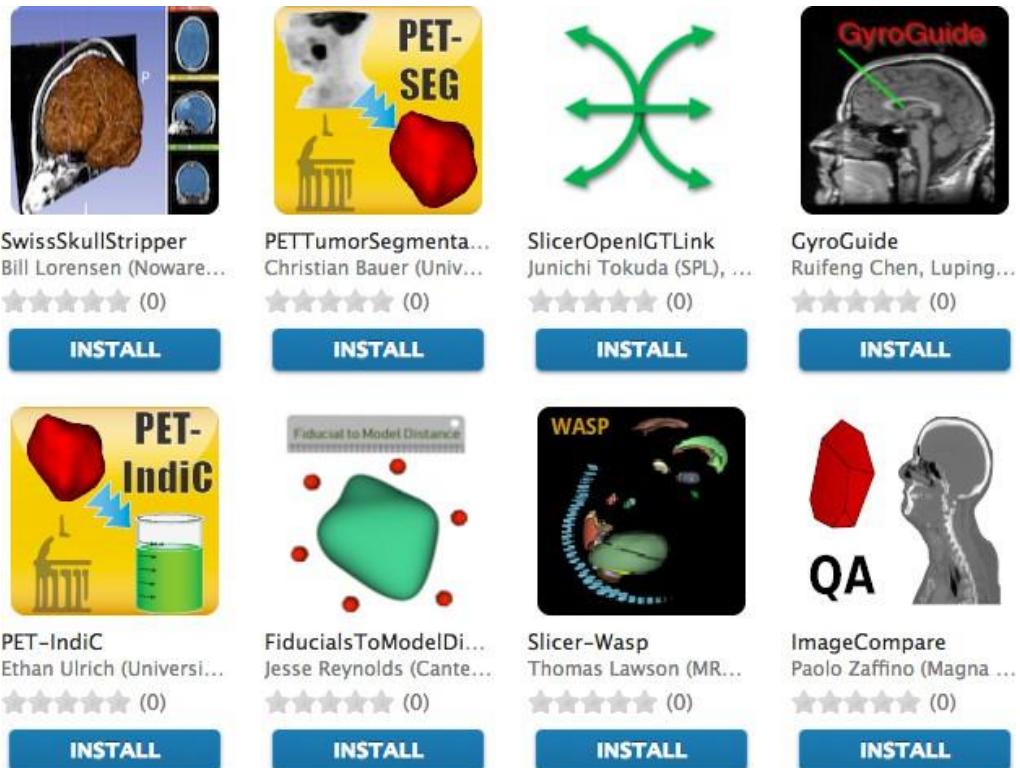


O tipo de módulo é transparente para o usuário final.

- Model Maker
- Model To Label Map
- Models
- Multiply Scalar Volumes
- MultiVolumeExplorer
- MultiVolumeImporter
- N4ITK MRI Bias correction
- Normals
- Orient Scalar Volume
- PET Standard Uptake Value Computation
- Plots
- Probe Volume With Model
- Reformat
- relaxPolygons
- Resample DTI Volume
- Resample Image (BRAINS)
- Resample Scalar Volume
- Resample Scalar/Vector/DWI Volume
- Resize Image (BRAINS)
- Robust Statistics Segmente
- Sample Data
- scaleMesh
- Scene Views
- View Controllers
- Volume Rendering
- Volumes
- Voting Binary Hole Filling Image Filter
- Welcome to Slicer

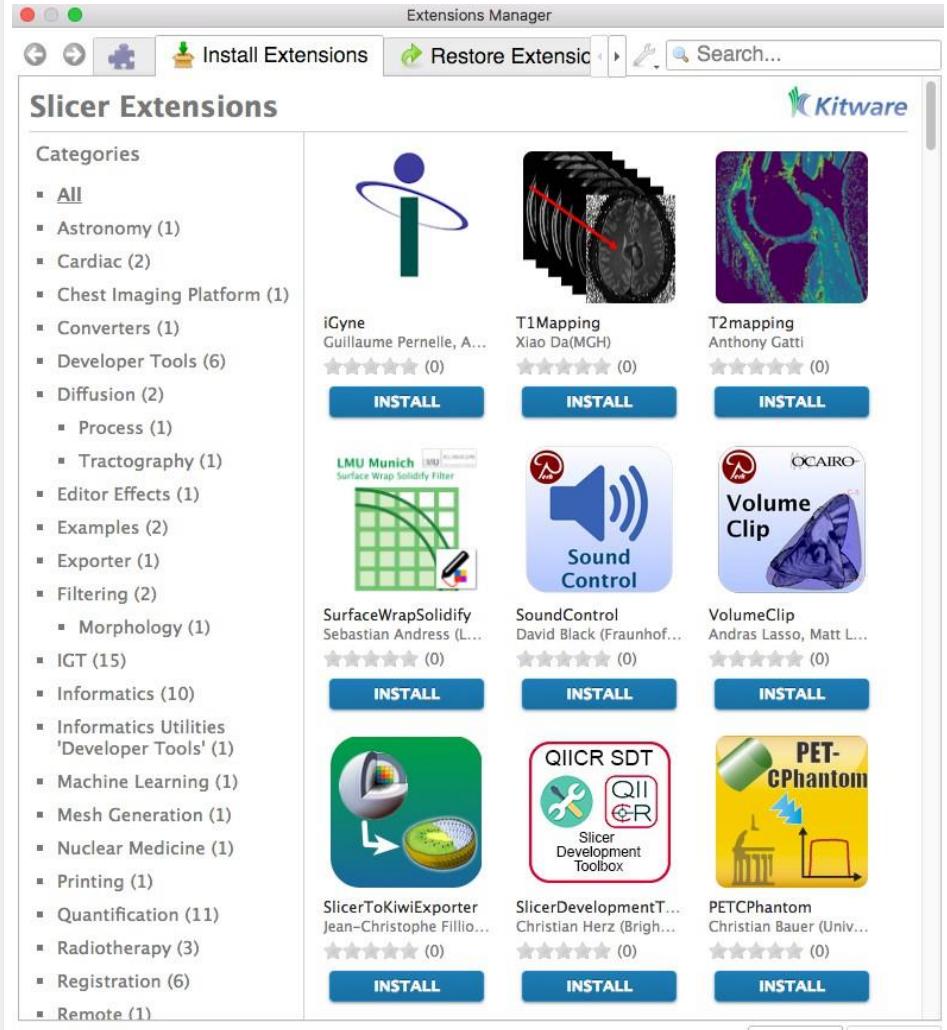
Extensões do Slicer

Uma Extensão do Slicer é um pacote de distribuição que agrupa um ou mais módulos do Slicer.



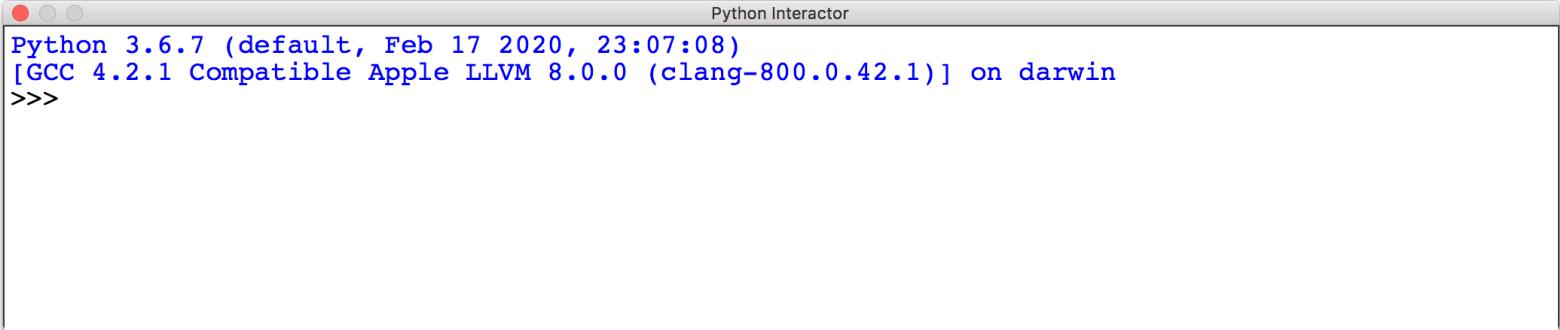
Gerenciador de Extensões do Slicer

- O Gerenciador de Extensões do Slicer oferece uma plataforma do tipo “loja de aplicativos” para o ecossistema do 3D Slicer.
- O Gerenciador de Extensões permite a criação e a instalação fácil de extensões do Slicer.
- A versão de lançamento do Slicer 5 inclui mais de 130 extensões.



Parte 2

Familiarizando-se com o ambiente Python no 3D Slicer



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

```
Python 3.6.7 (default, Feb 17 2020, 23:07:08)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.42.1)] on darwin
>>>
```

Python no Slicer

O Slicer v. 5 funciona com Python 3 e um conjunto rico de bibliotecas padrão.



O NumPy é o pacote fundamental para a computação científica com Python.



O VTK é uma biblioteca de código aberto para manipulação e apresentação de dados científicos.



O ITK é uma biblioteca de código aberto para análise de imagens.



CTK é uma biblioteca de código aberto para a computação de imagens biomédicas.



PythonQT é uma ligação Python para Qt.

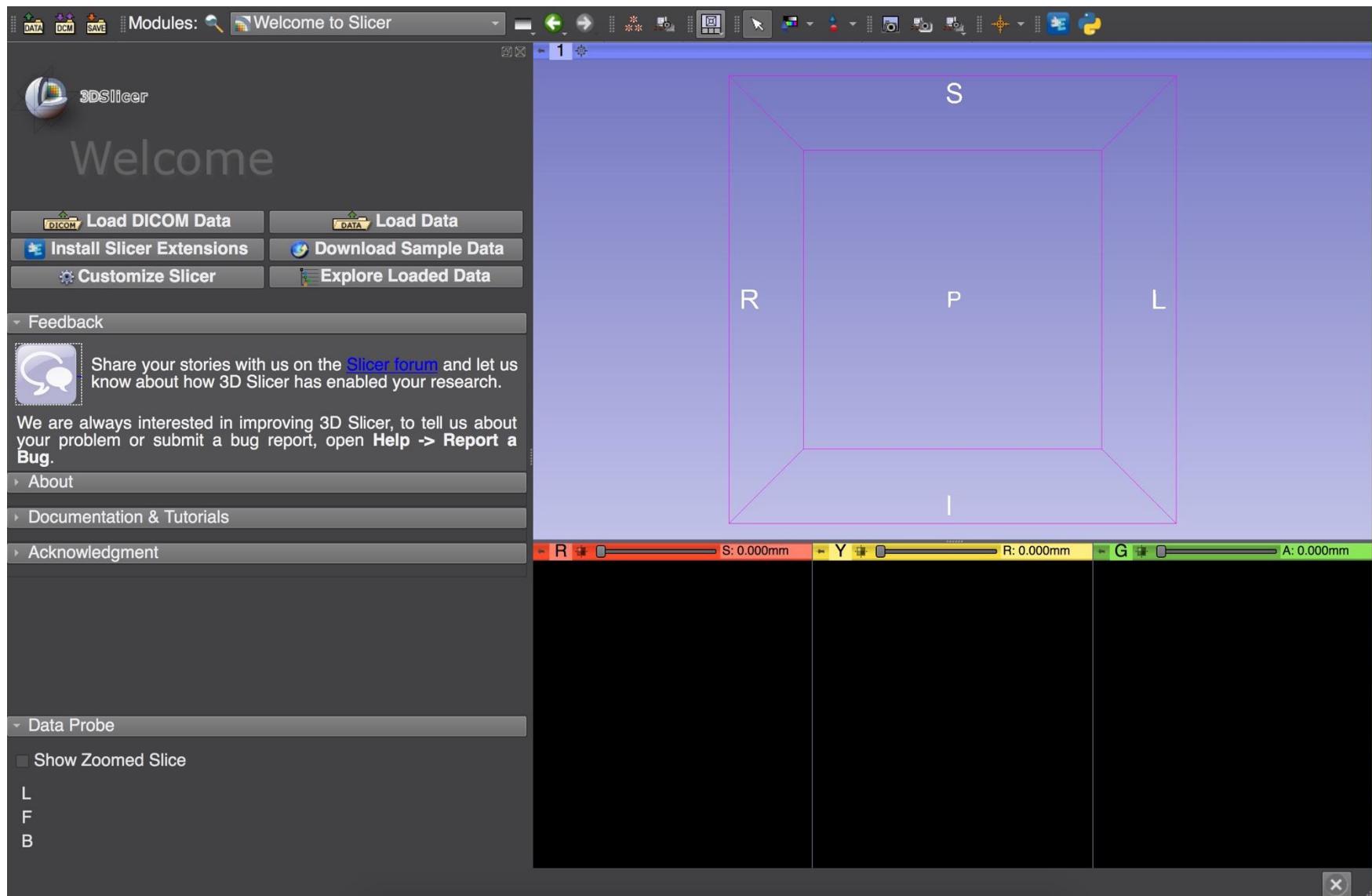


O Qt é uma estrutura multiplataforma utilizada como um conjunto de ferramentas gráficas.



Python no Slicer

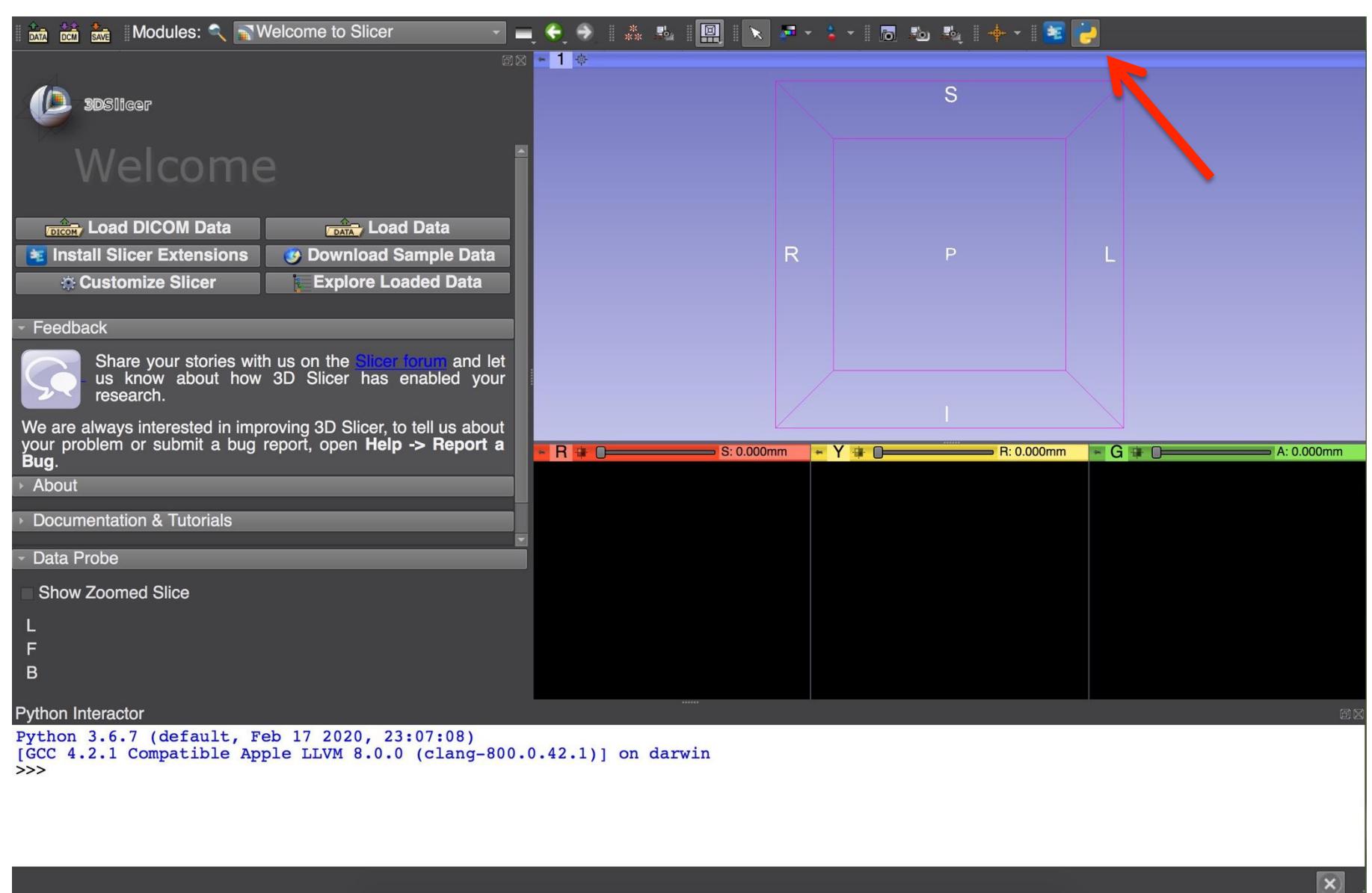
- O índice de pacotes Python (PyPi) dá acesso a mais de 200.000 pacotes Python adicionais (<http://pipy.org>).
- O comando *pip install* no Slicer permite que os programadores instalem as ferramentas de computação científica mais comuns (por exemplo, *TensorFlow*, *SciPy*, *PyTorch*, *Pandas* etc.).
- O Slicer pode ser usado como um *kernel* do *Jupyter Notebook*.
- PyCharm e outras ferramentas de desenvolvimento Python podem ser usadas com o Slicer.



A versão 5 do Slicer integra Python3, VTK5 e ITK5

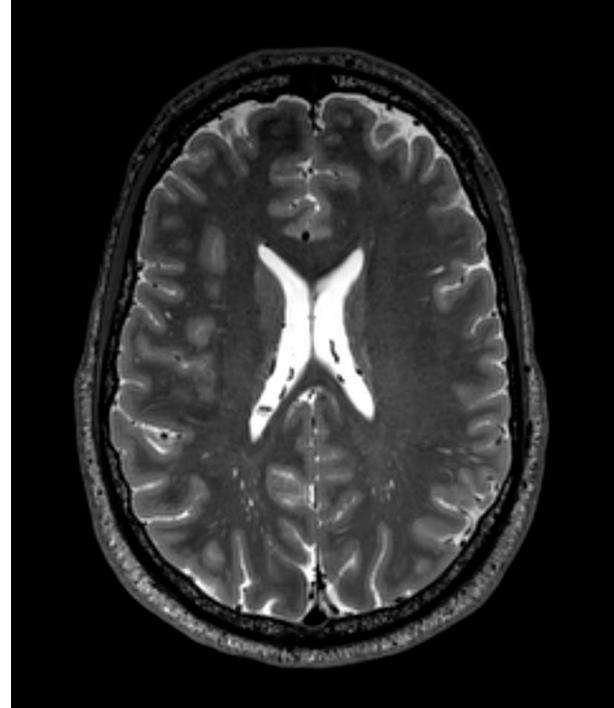
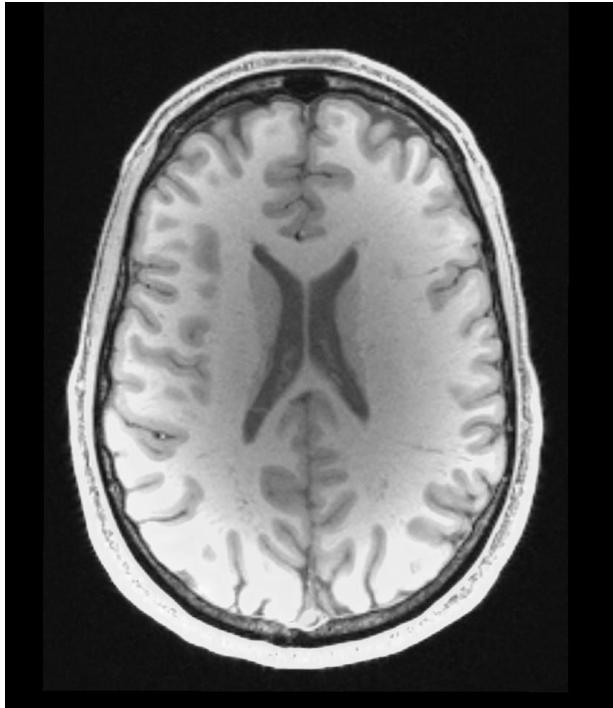
O console Python no Slicer

O Python Interactor é um console baseado em Qt que permite acesso direto aos Nós MRML do Slicer, a bibliotecas (NumPy, VTK, ITK, CTK) e ao Qt.



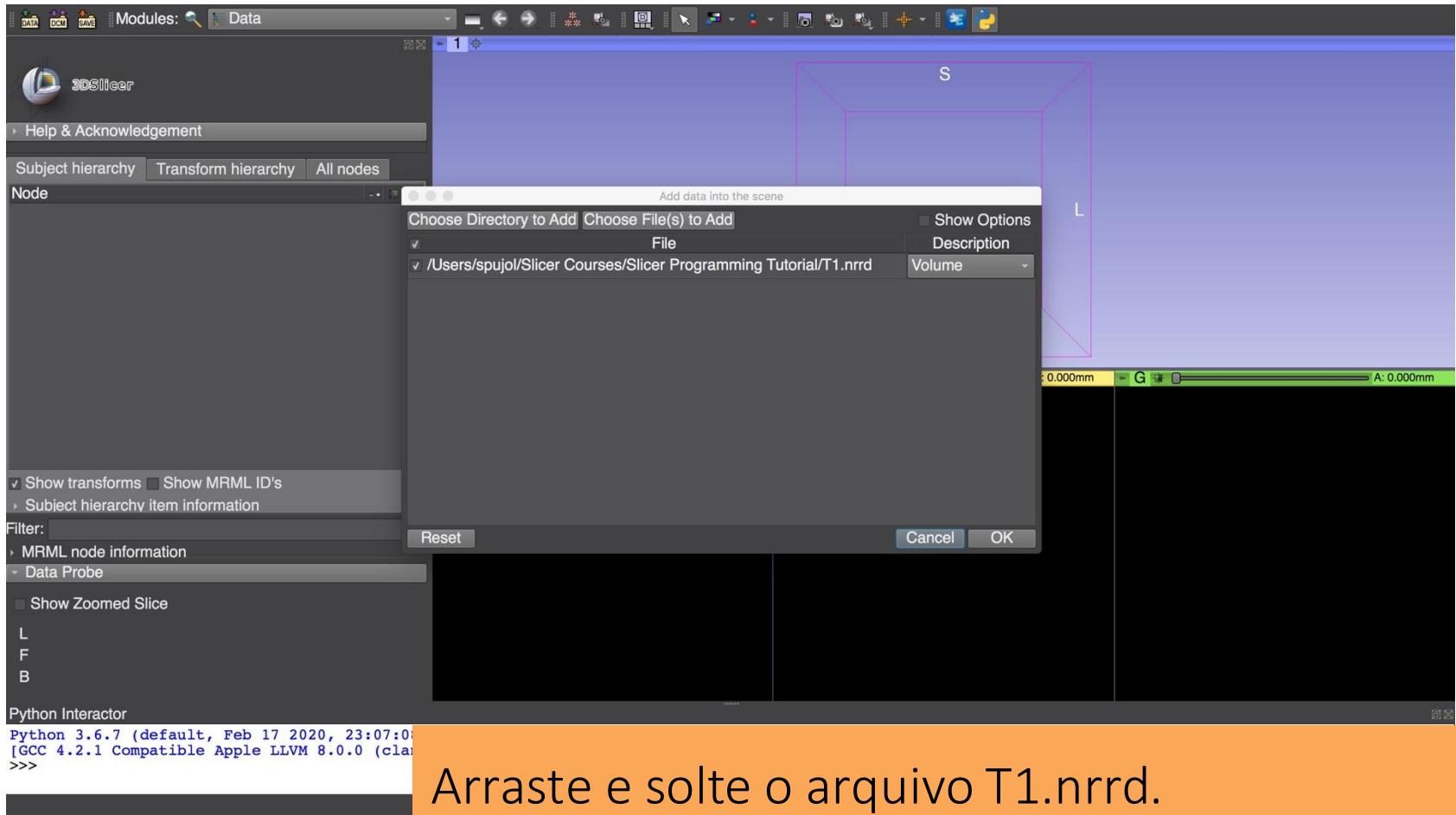
Para acessar o Python Interactor, clique no ícone do Python no menu da barra superior do Slicer.





O conjunto de dados do tutorial de programação do Slicer inclui exames de ressonância magnética ponderados em T1 e T2 de um sujeito saudável.

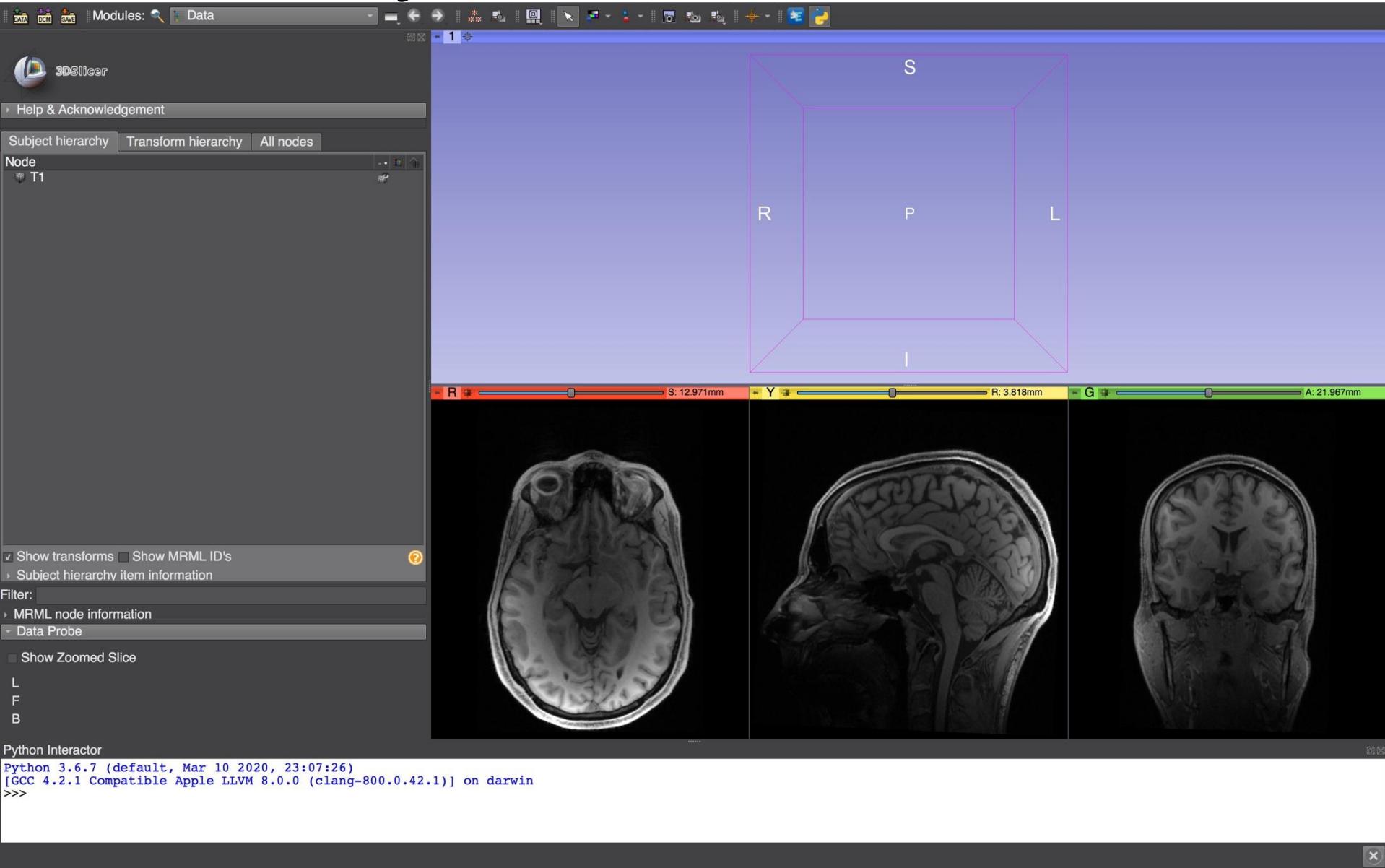
Conjunto de dados do tutorial



Arraste e solte o arquivo T1.nrrd.

Clique em OK para carregar o arquivo no Slicer.

Conjunto de dados do tutorial



Visão Geral

- O Slicer é um *software* livre e de código aberto.
- Há milhares de imagens médicas sofisticadas disponíveis na *internet* que você pode visualizar e analisar com o 3D Slicer.

Modelo de Dados do Slicer



O Modelo de Dados do Slicer é baseado na Estrutura de Dados da Cena do Slicer.



A *Medical Reality Markup Language* (*MRML*) [Linguagem de Marcação da Realidade Médica] é uma linguagem baseada em XML usada para serializar o conteúdo da cena do Slicer no disco (`scene.mrml`).



Uma cena do Slicer é uma coleção de imagens, anotações, modelos 3D, transformações espaciais, marcas de referência e câmeras.



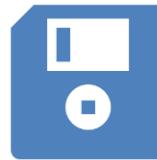
Cada elemento de uma cena é chamado de nó *MRML*.

Nós MRML do Slicer: Tipos Básicos



Nó de dados: Armazena os
dados brutos

Nó de exibição: Descreve
como os dados devem ser
visualizados

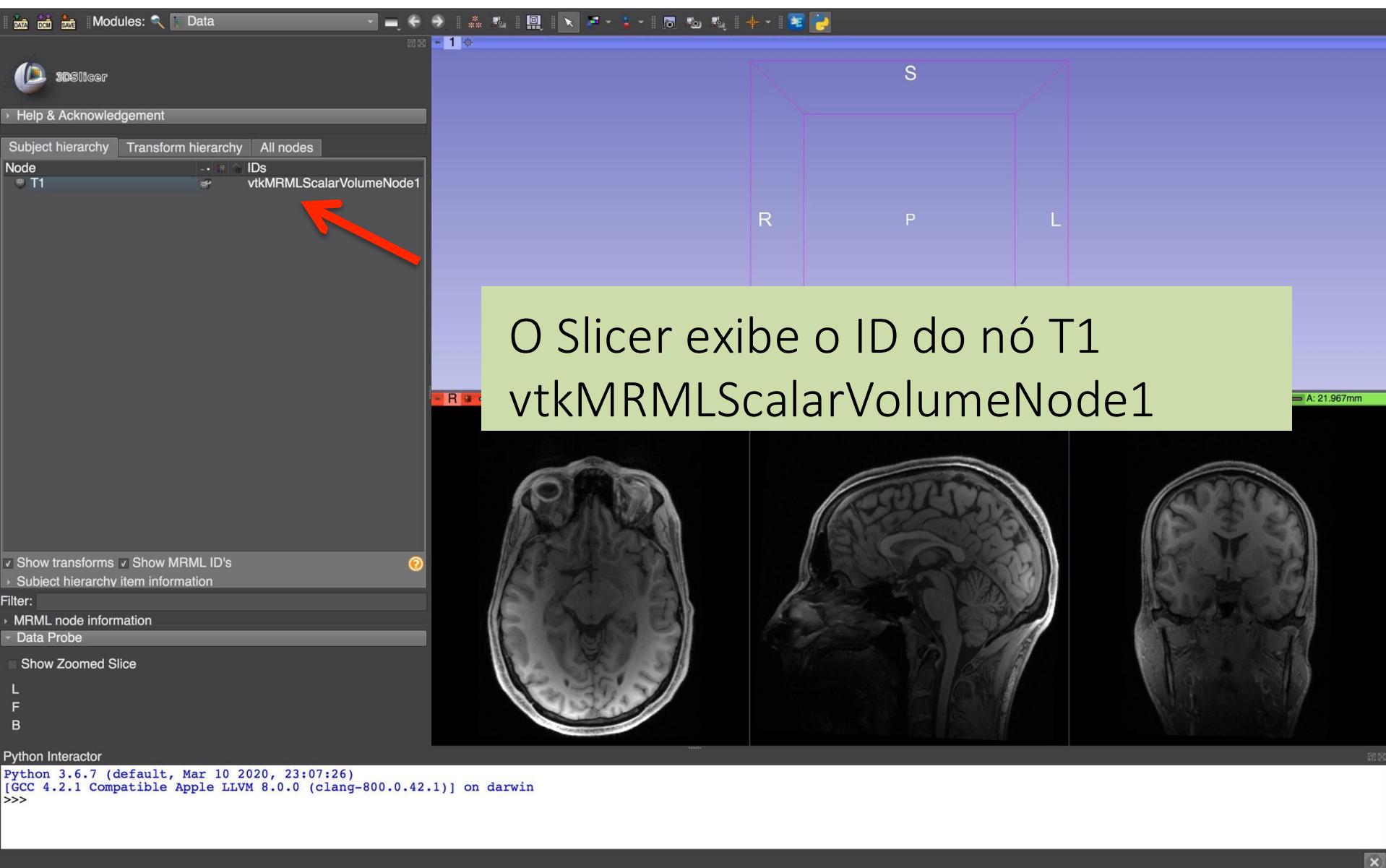


Nó de armazenamento:
Descreve como os dados
devem ser armazenados no
disco

Conjunto de dados do Tutorial

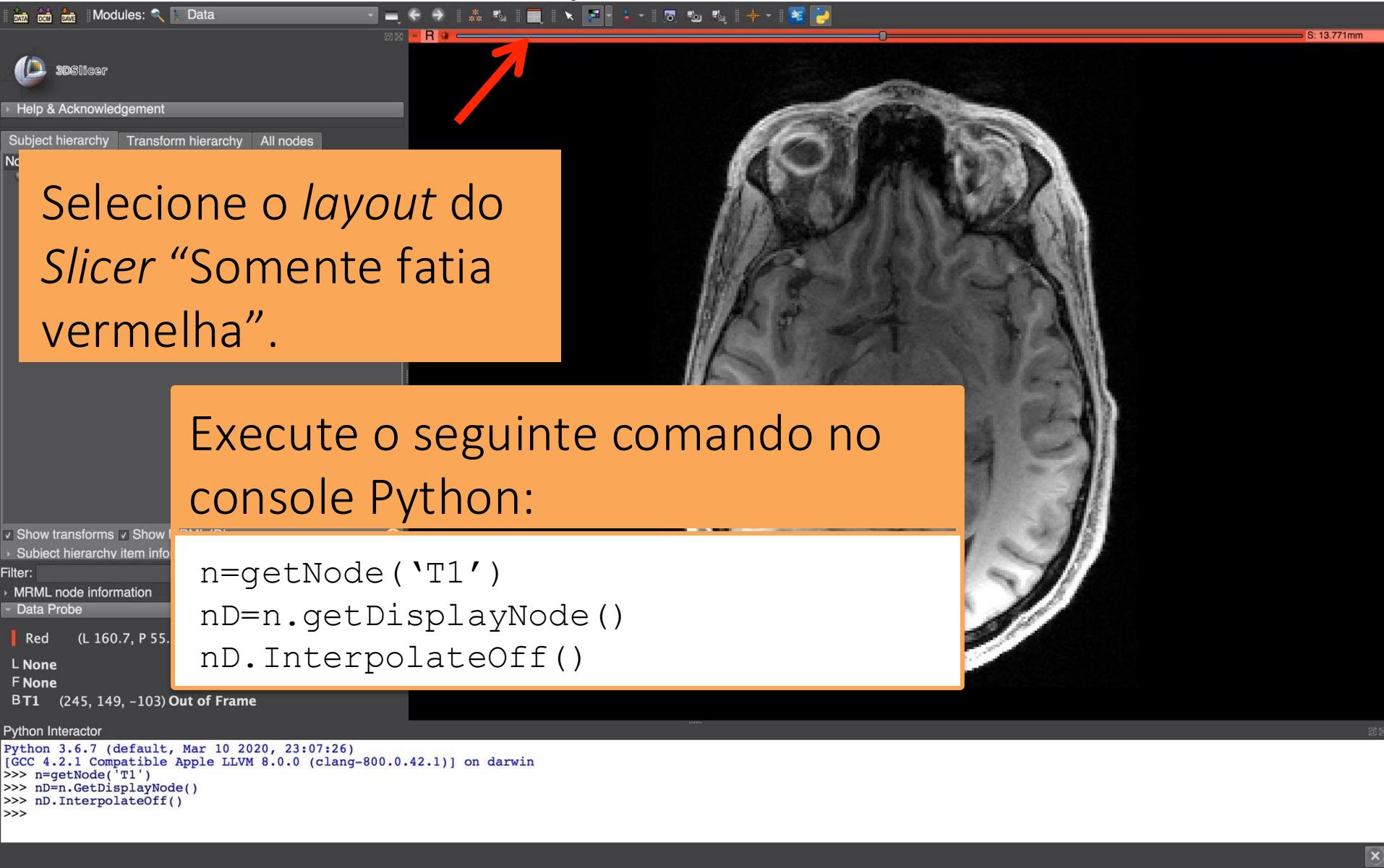


Modelo de Dados do Slicer

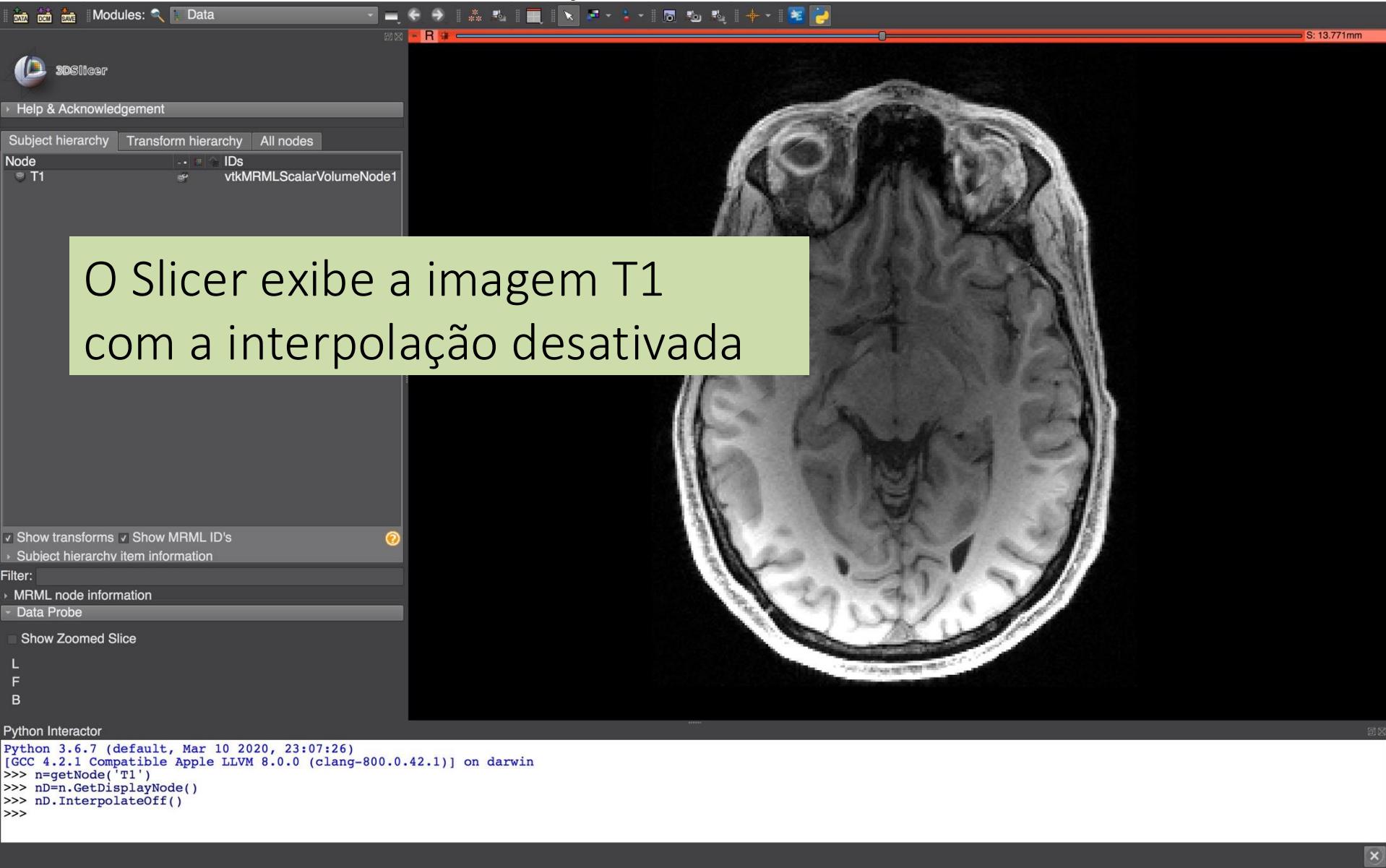


O Slicer exibe o ID do nó T1
vtkMRMLScalarVolumeNode1

Acessando nós MRML a partir do Python interactor

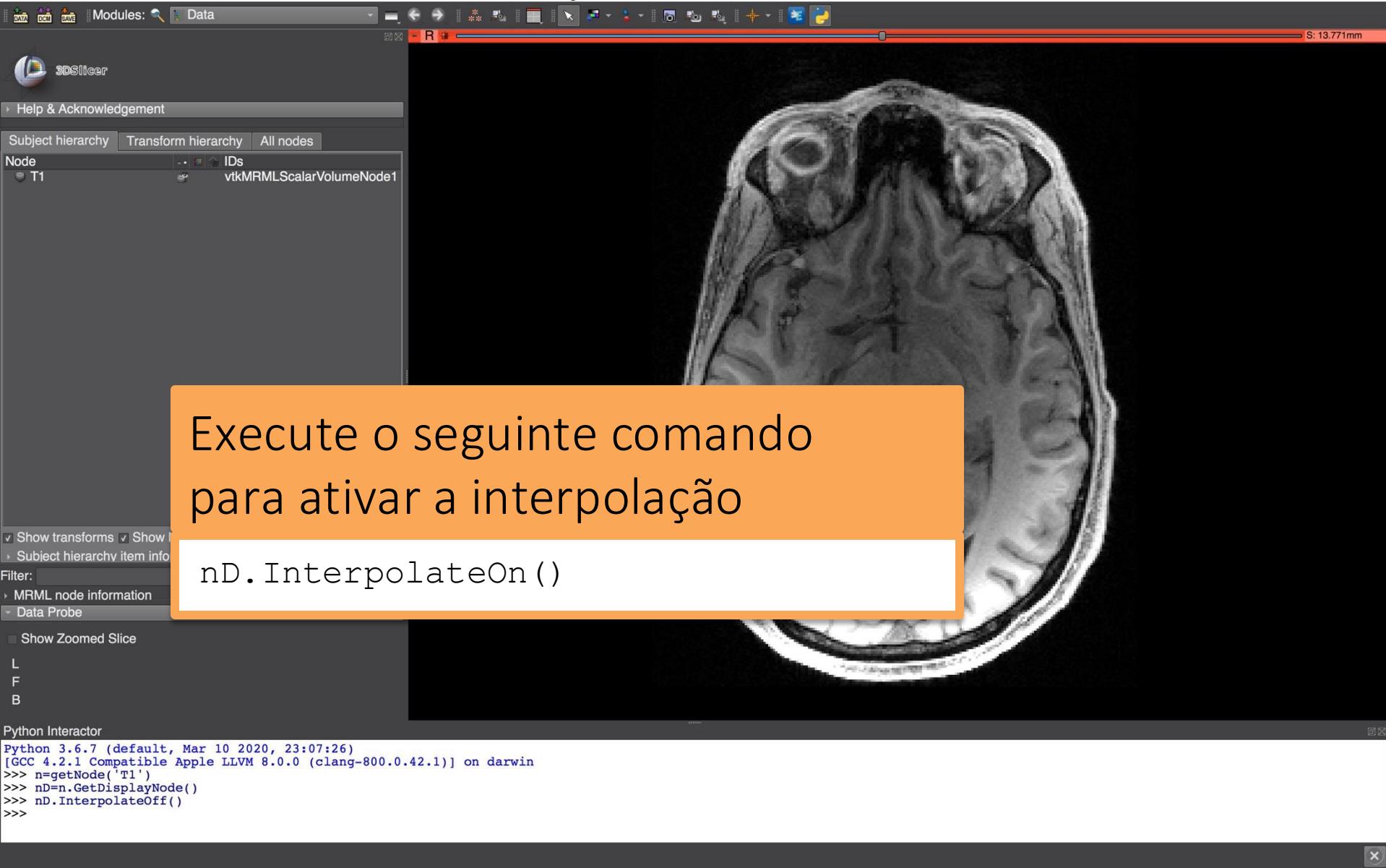


Acessando nós MRML a partir do Python interactor

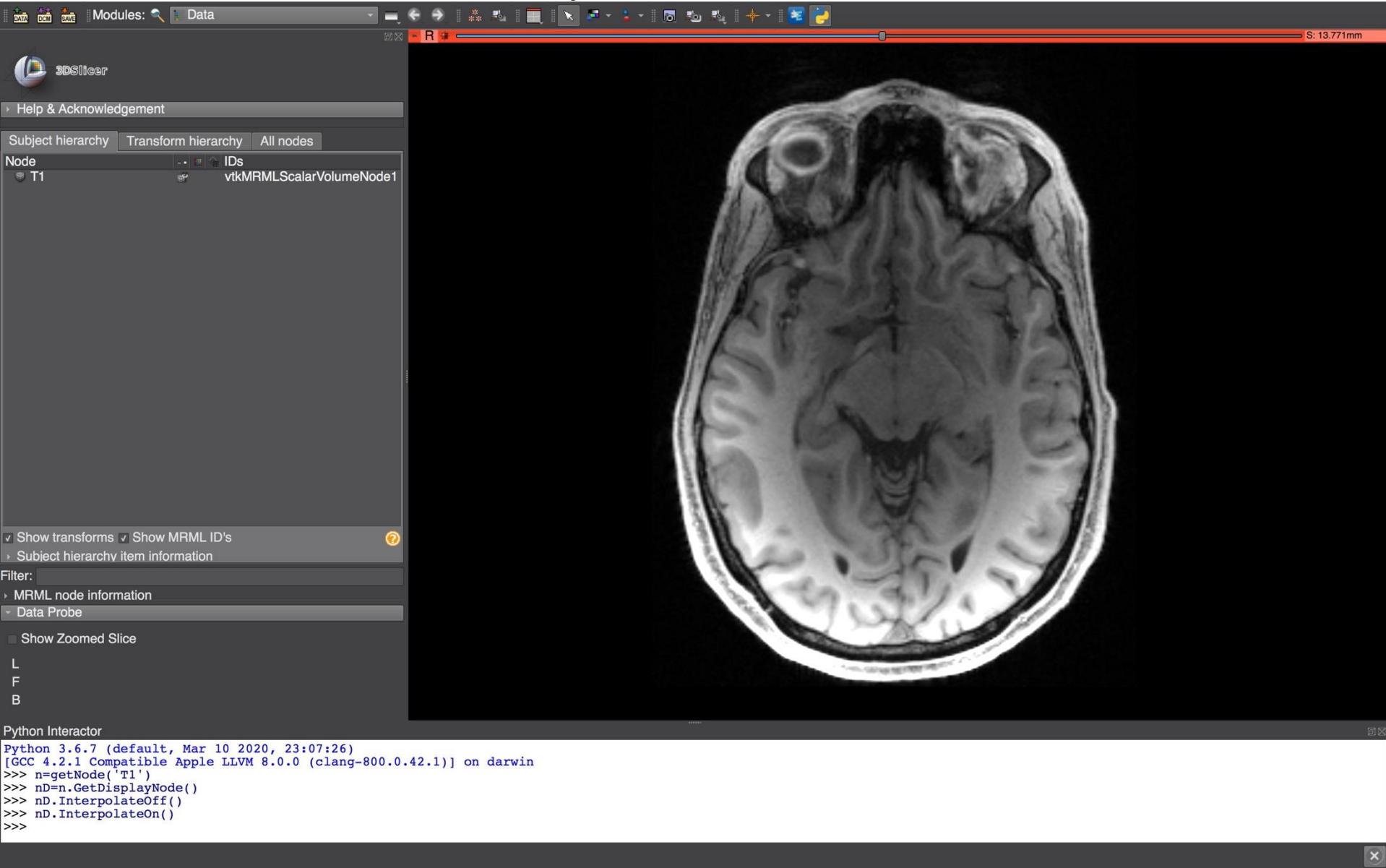


O Slicer exibe a imagem T1
com a interpolação desativada

Acessando nós MRML a partir do Python interactor

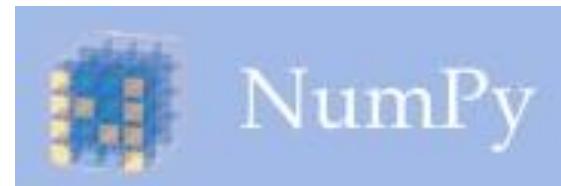


Acessando nós MRML a partir do Python interactor



Acessando vértices em um volume

- O pacote `slicer.util` oferece acesso aos volumes como *arrays* multidimensionais do *NumPy*.
- Os volumes podem ser modificados utilizando métodos padrão do *NumPy*.



Acessando véxeis em um volume

The screenshot shows the 3DSlicer application interface. On the left, the 3D Slicer interface includes a subject hierarchy panel with a node named 'T1' selected. A Python console window is open at the bottom, displaying a script to access the volume data. The main workspace shows a grayscale 3D volume rendering of a brain.

Execute o seguinte comando no console Python:

```
a=slicer.util.array('T1')
print(a)
```

Nota: no console Python, o `slicer.util` é importado automaticamente.

```
a=array('T1') # same as above
print(a)
```

```
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.42.1)] on darwin
>>> n=getNode('T1')
>>> nD=n.GetDisplayNode()
>>> nD.InterpolateOff()
>>> nD.InterpolateOn()
>>>
>>> a = slicer.util.array('T1')
>>> print(a)
```

Acessando véxeis em um volume

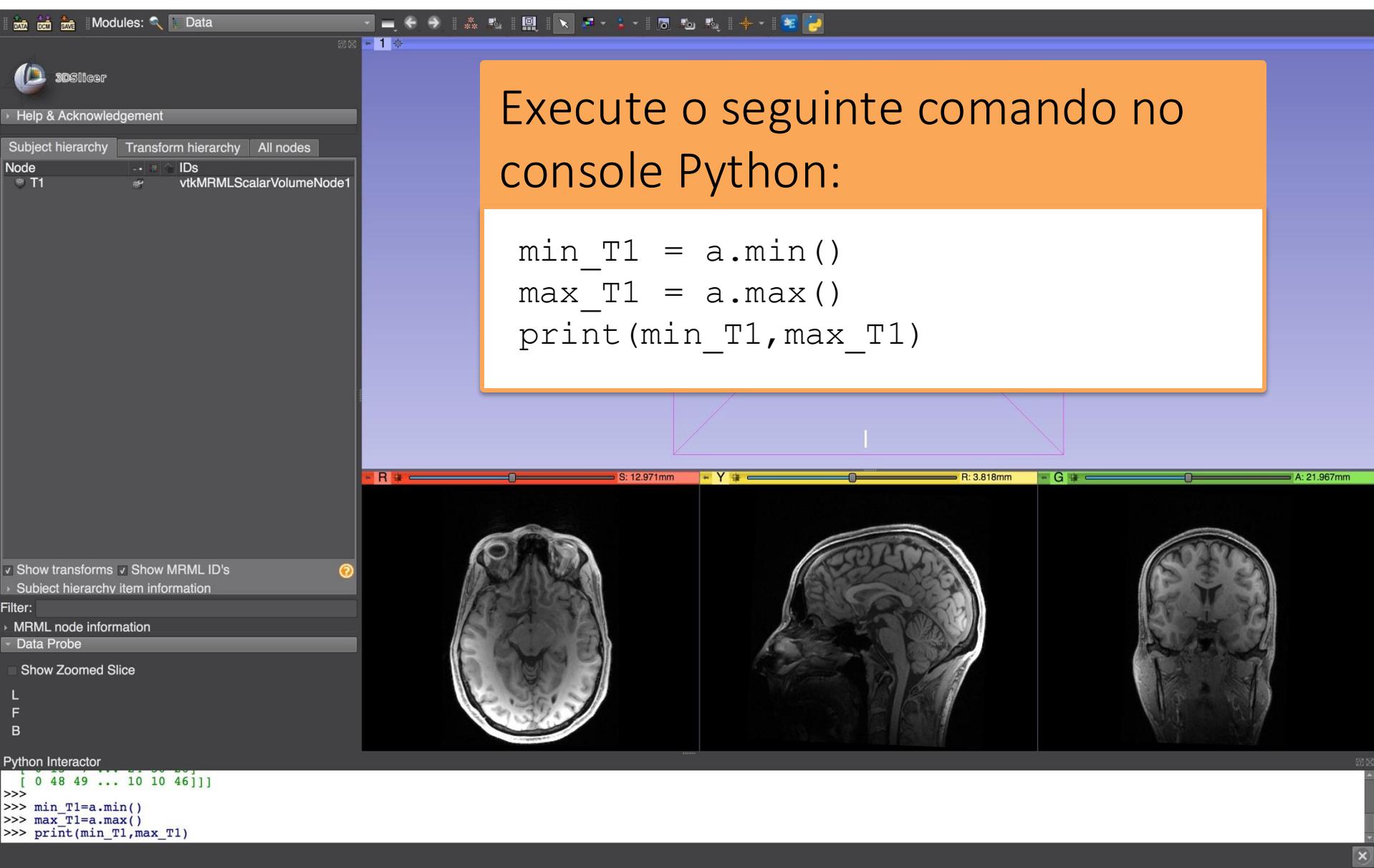
The screenshot shows the Slicer medical image processing application. On the left, the Python Interactor window displays a truncated printout of a 3D array from the T1 image. The array values are shown in green text. On the right, the main Slicer interface shows a 3D volume rendering of a brain scan with a coordinate system (Sagittal, Coronal, Axial) and a 2D slice view at the bottom.

```
>>> a = slicer.util.array('T1')
>>> print(a)
[[[ 0  0  0 ...  0  0  0]
[[ 0 20  6 ... 10 52 27]
[[ 0 24 25 ...  4 32  8]
...
[[ 0 48 14 ... 41 42 21]
[[ 0 15 40 ... 33 38 25]
[[ 0 55 19 ... 21  7 17]]
...
[[ 0  0  0 ...  0  0  0]
[[ 0  4 14 ... 30 17 42]
[[ 0 22  9 ... 11 12 49]
...
[[ 0 86 18 ... 16 66 11]
[[ 0 48 26 ... 14 23 21]
[[ 0 16  3 ... 31 14 33]]
...
[[ 0  0  0 ...  0  0  0]
[[ 0 60 39 ...  7 28 10]
[[ 0 58 19 ... 34 31 29]
...
[[ 0  5 48 ... 39 21 38]
[[ 0 22 55 ... 14 46 15]
[[ 0 17 45 ... 26 20 43]]
...
[[ 0  0  0 ...  0  0  0]
[[ 0  8 26 ... 33 36 44]
[[ 0 27 18 ... 21 21 45]
...
[[ 0 12 22 ... 22 34 14]
[[ 0  2 11 ... 48 65 35]
[[ 0 25  7 ... 7 17 11]]
...
[[ 0  0  0 ...  0  0  0]
[[ 0 34 44 ... 13 41 30]
[[ 0 23 24 ... 28 51 33]
...
[[ 0 18 36 ... 50 14 54]
[[ 0 17 34 ... 42 16 53]
[[ 0 12 30 ... 45 51 36]]
...
[[ 0  0  0 ...  0  0  0]
[[ 0  5 41 ... 11  9 48]
[[ 0 21 64 ... 32 11  9]
...
[[ 0 11 33 ... 30 11 43]
[[ 0 13  7 ... 24 30 26]
[[ 0 48 49 ... 10 10 46]]]
```

O Slicer exibe os valores de intensidade da imagem T1.

O Python interactor mostra uma exibição truncada de um array 3D.

Acessando vóxeis em um volume



Acessando véxeis em um volume

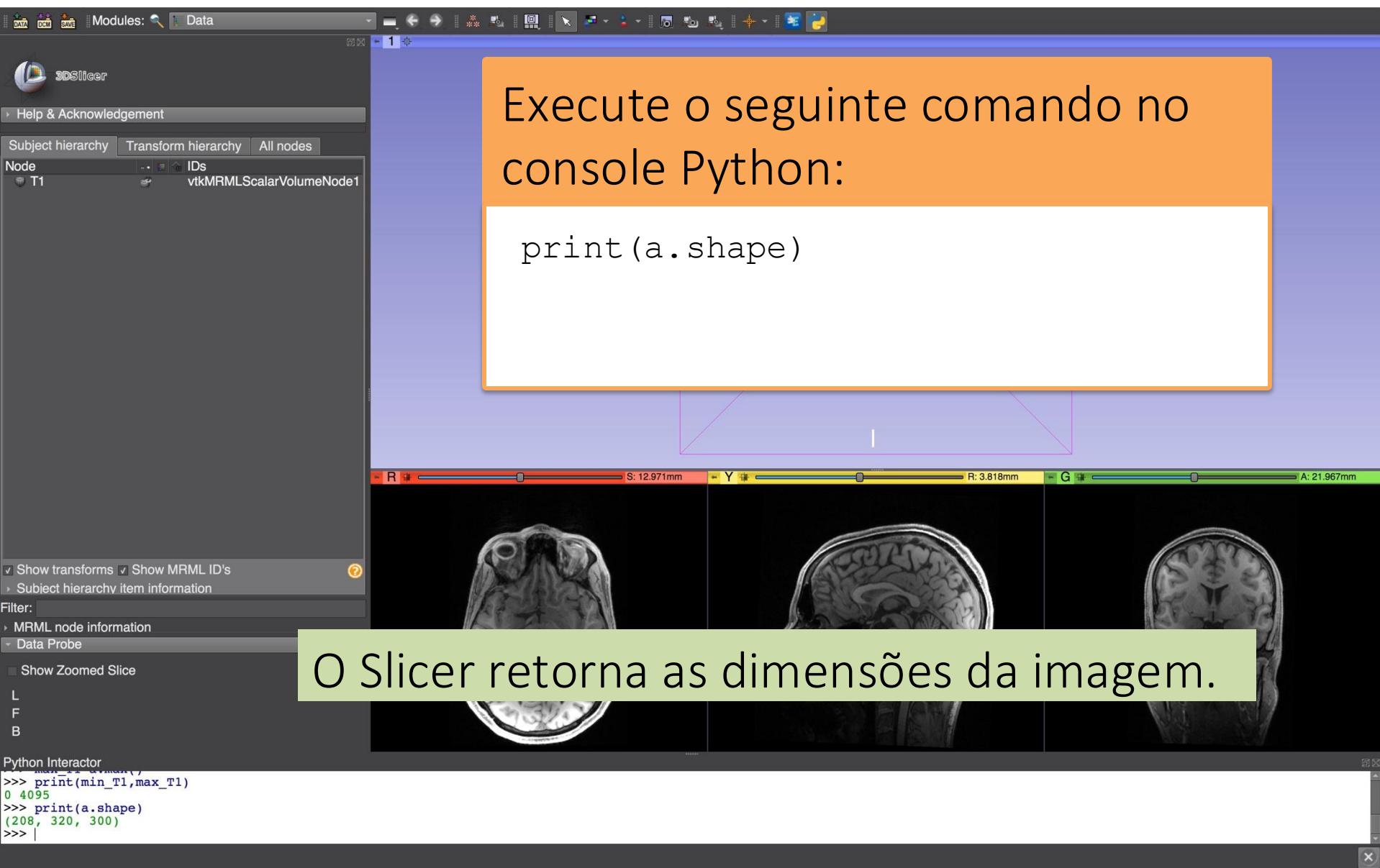


O Slicer retorna os valores mín. e máx. da imagem T1.

Python Interactor

```
>>> min_T1=a.min()  
>>> max_T1=a.max()  
>>> print(min_T1,max_T1)  
0 4095  
>>>
```

Modificando vóxeis em um volume



Modificando véxeis em um volume

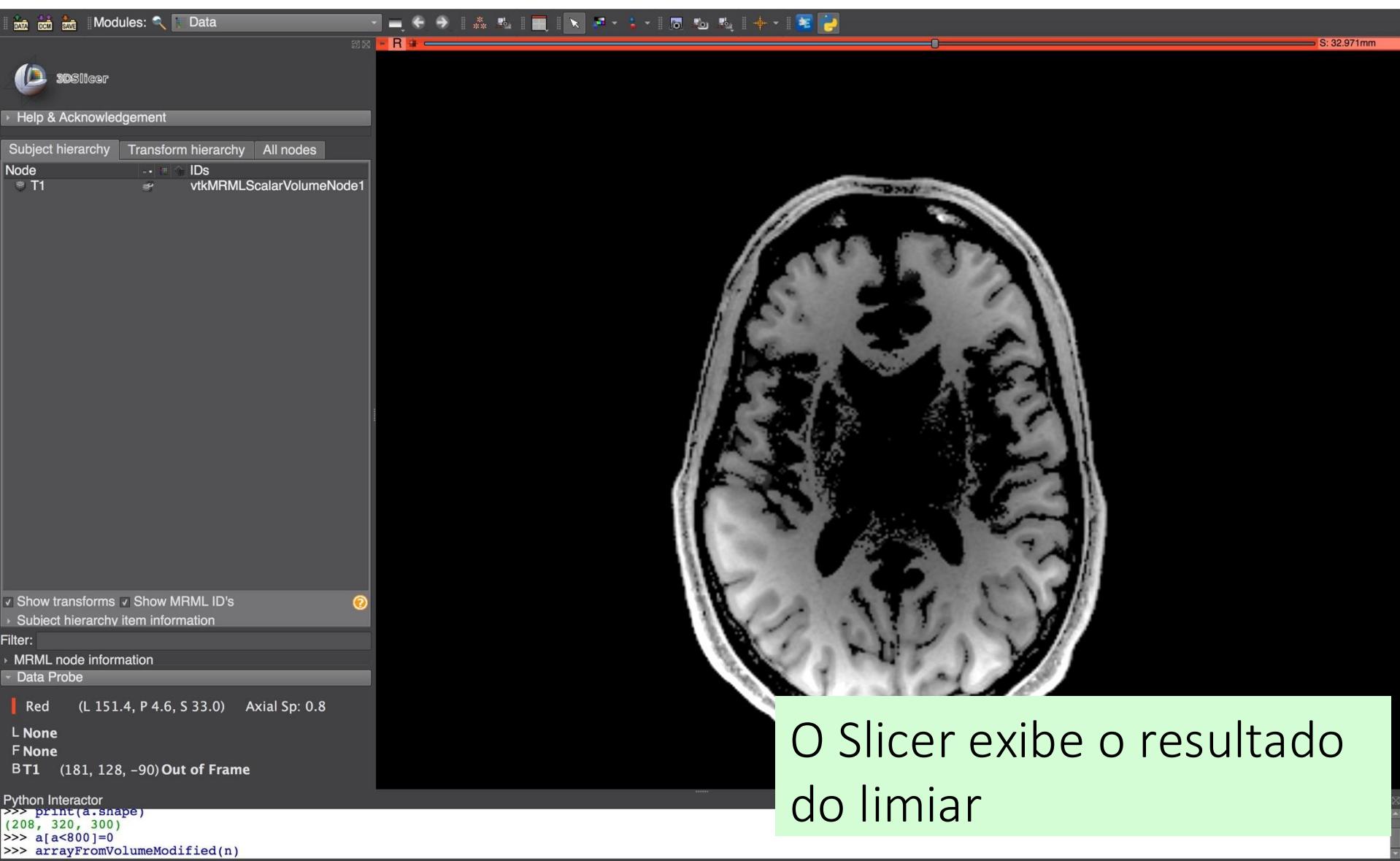
Execute os seguintes comandos no console Python:

```
a[a<800]=0  
arrayFromVolumeModified(n)
```

Este exemplo aplica um limiar de $t=800$ à imagem e notifica o Slicer sobre a modificação.

```
0 4095  
>>> print(a.shape)  
(208, 320, 300)  
>>> a[a<800]=0  
>>> arrayFromVolumeModified(n)
```

Modificando véxeis em um volume



O Slicer exibe o resultado
do limiar

Carregando o volume T2

Carregue a imagem T2 no Slicer:

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

3DSlicer

DATA DCM SAVE Modules: Data

Help & A Subject hierarchy Node T1 T2

L R S: 7.371mm

Show transforms Show MRML ID's
Subject hierarchy item information

Filter:
MRML node information
Data Probe

Show Zoomed Slice

L
F
B

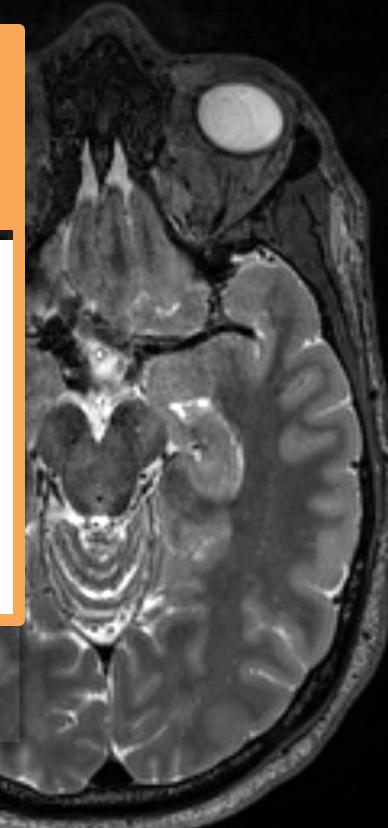
Python Interactor

```
>>>
>>> slicer.util.loadVolume('/Users/spujol/SlicerCourses/SlicerProgrammingTutorial/T2.nrrd')
(MRMLCorePython.vtkMRMLScalarVolumeNode)0x140328c48
>>>
>>>
>>> |
>>>
>>>
>>>
```

Função Python: threshold [limiar]

Crie uma função threshold (t)
no Python interactor:

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



DATA DCM SAVE Modules: Data R S: 7.371mm

3DSlicer

Help & A

Subject hierarchy

Node

T1 T2

Show transforms Show MRML IDs

Subject hierarchy item information

Filter:

MRML node information

Data Probe

Red (R 145.5, P 86.4, S 7.4) Axial Sp: 0.8

L None

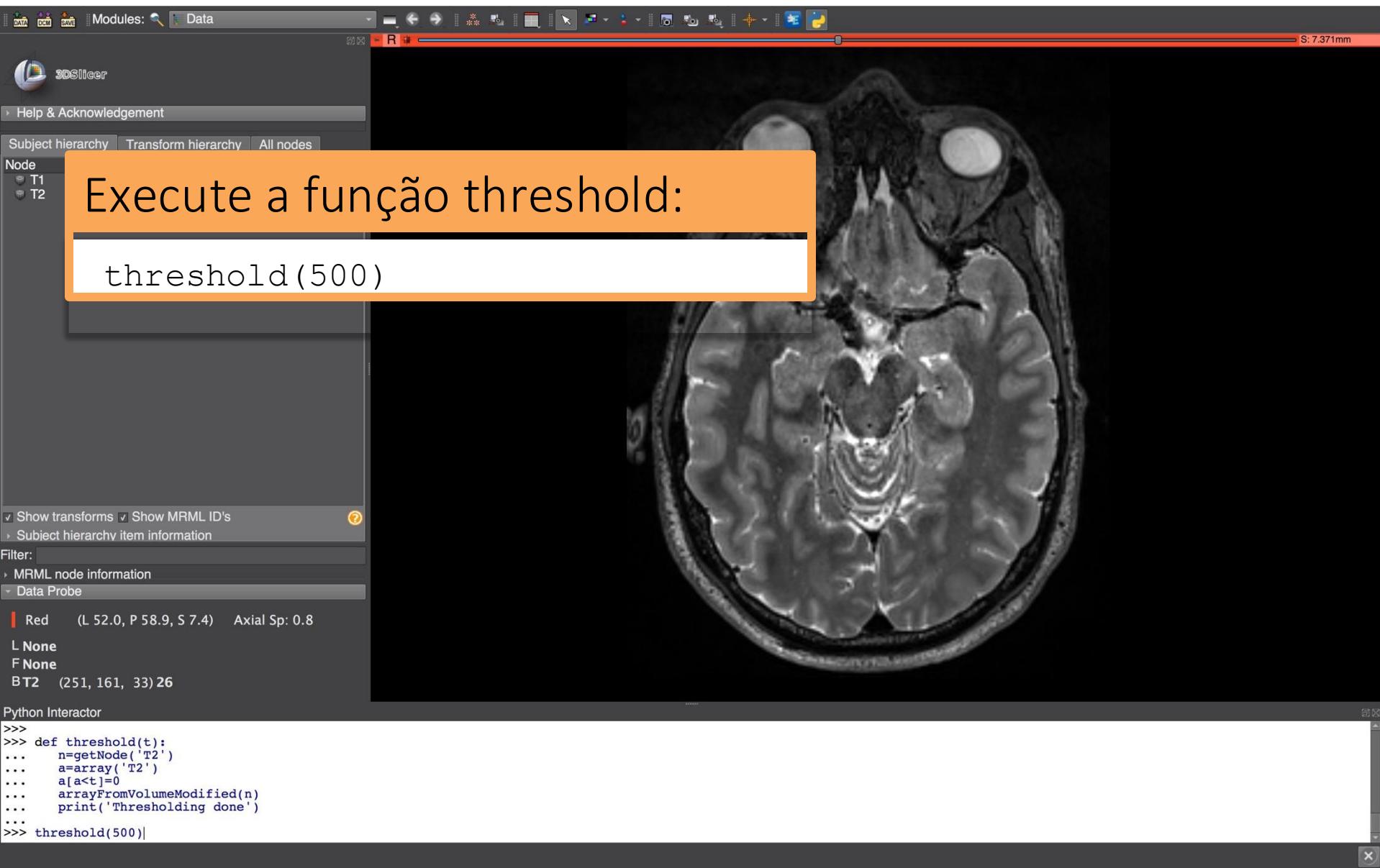
F None

BT2 (286, 167, 280) Out of Frame

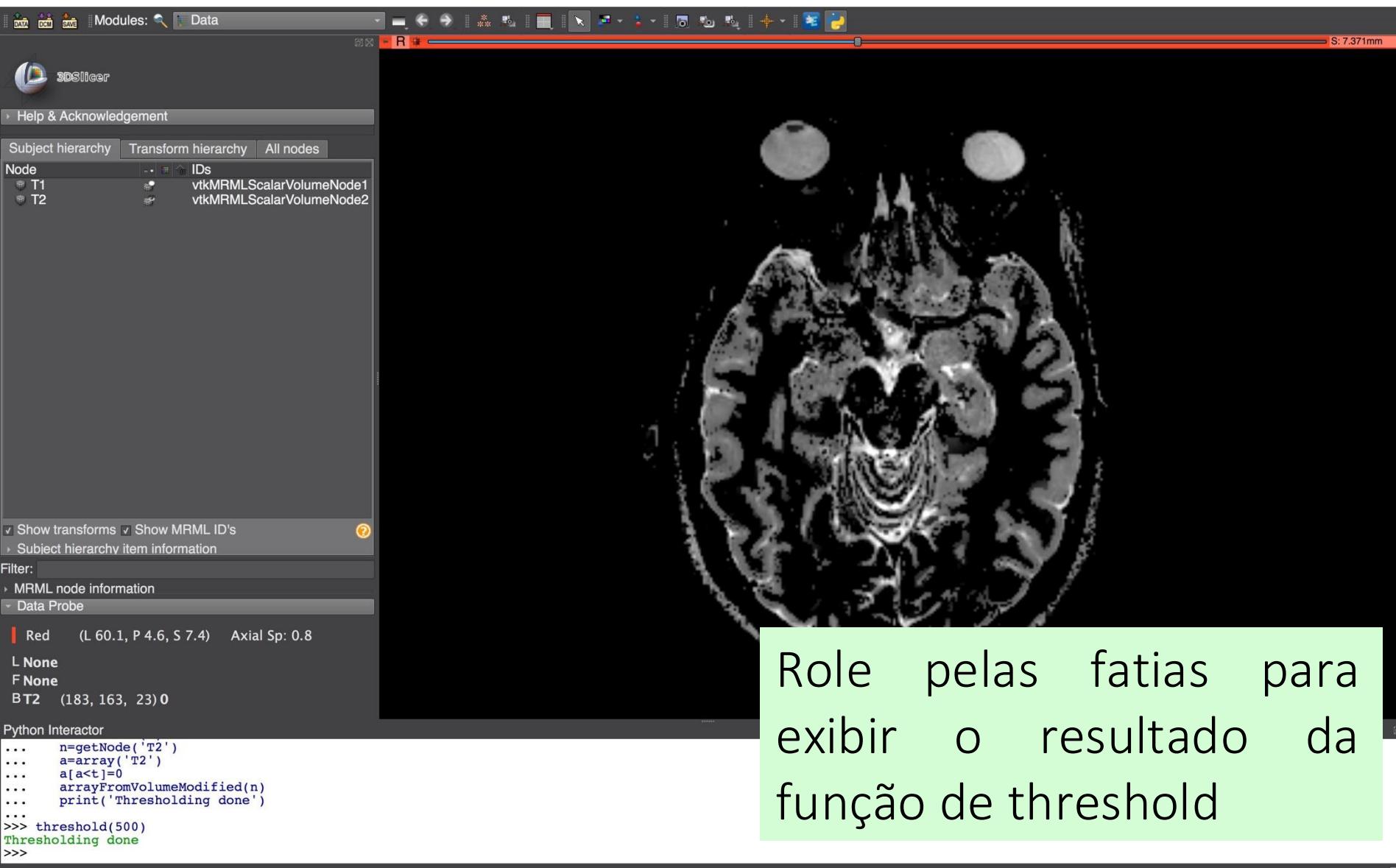
Python Interactor

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

Função Python: threshold



Função Python: threshold



Visão Geral

- O Slicer oferece fácil acesso para analisar e modificar tipos de dados complexos.
- O Slicer é compatível com uma ampla gama de pacotes de computação científica em Python.
- O Slicer é um ambiente de pesquisa para realizar experimentos de imagens médicas.

Parte 3

Familiarizando-se com o Qt no Slicer

Qt & PyQt

- O Qt é a principal ferramenta no Slicer para criar *widgets*, diálogos, campos de texto etc.
- O PyQt expõe a maioria das funcionalidades do Qt e é acessível por meio do *Python interactor* no Slicer.
- Interfaces de usuário podem ser criadas rapidamente para prototipagem e depuração.

Função Python: toggle [alternar]

The screenshot shows the 3DSlicer application interface. On the left, the 3D Slicer node browser displays a hierarchy with nodes T1 and T2 under the Subject hierarchy tab. The Transform hierarchy tab shows vtkMRMLScalarVolumeNode1 and vtkMRMLScalarVolumeNode2. The Python Interactor window on the right contains a code snippet for creating a toggle function.

Crie uma função toggle() no Python interactor:

```
def toggle():
    n=getNode("T1")
    a=array("T1")
    a[a<0] = 0
    a[a>1000] = 700
    a[:]=a.max()-a
    arrayFromVolumeModified(n)
```

```
Python Interactor
>>> def toggle():
...     n=getNode('T1')
...     a=array('T1')
...     a[a<0]=0
...     a[a>1000]=700
...     a[:]=a.max()-a
...     arrayFromVolumeModified(n)
...
>>>
```

Criando um Botão de Push do Qt

The screenshot shows the 3DSlicer interface with the Python Interactor tab selected. On the left, the 3DSlicer module browser displays nodes T1 and T2 under the Subject hierarchy. The main area contains a code snippet for creating a QtPushButton:

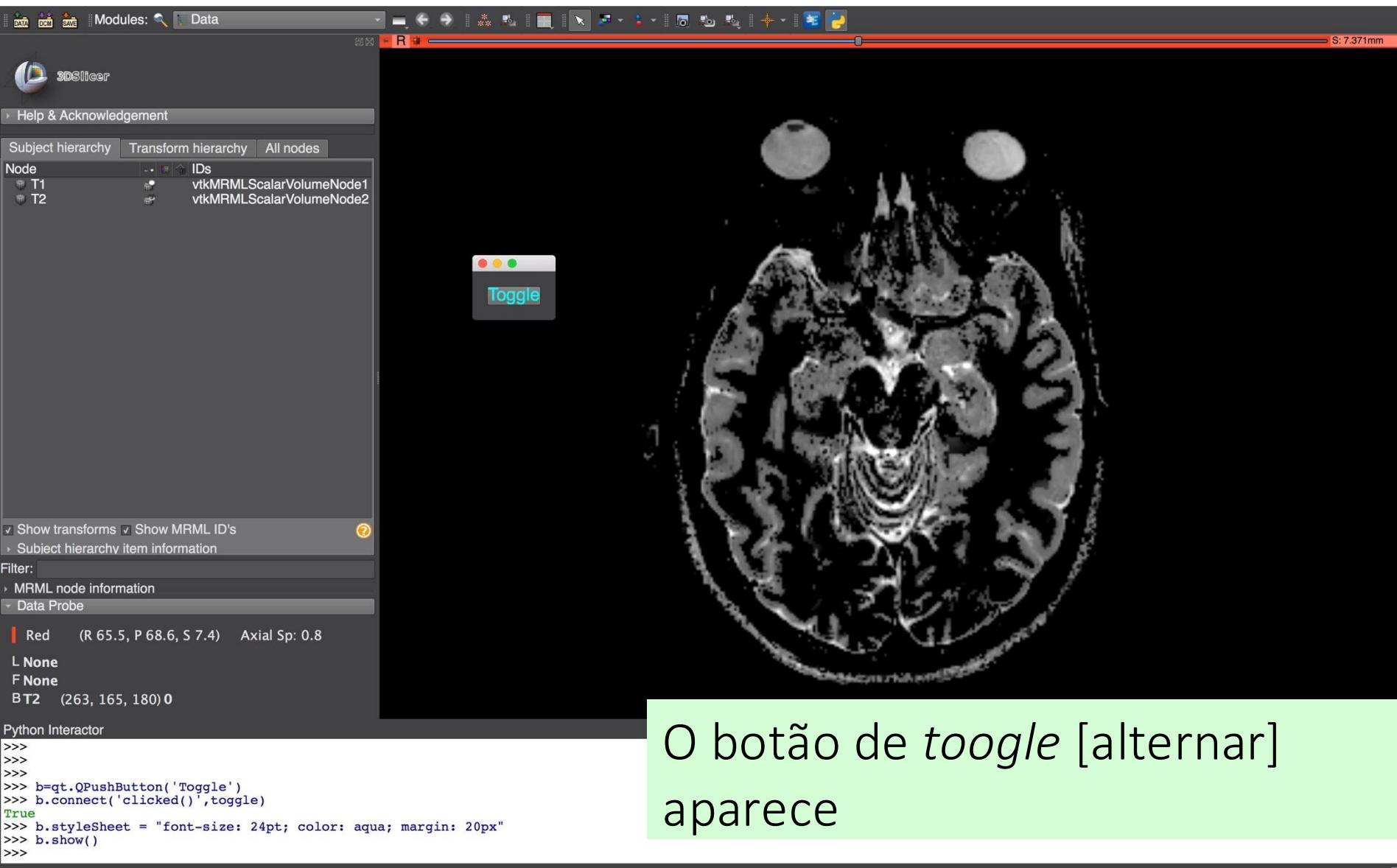
```
b=qt.QPushButton('Toggle')
b.connect('clicked()',toggle)
b.setStyleSheet = "font-size: 24pt;
color: aqua; margin: 20px"
b.show()
```

A callout box highlights the `StyleSheet` assignment, with the text "styleSheet é css" (StyleSheet is css) displayed in an orange box.

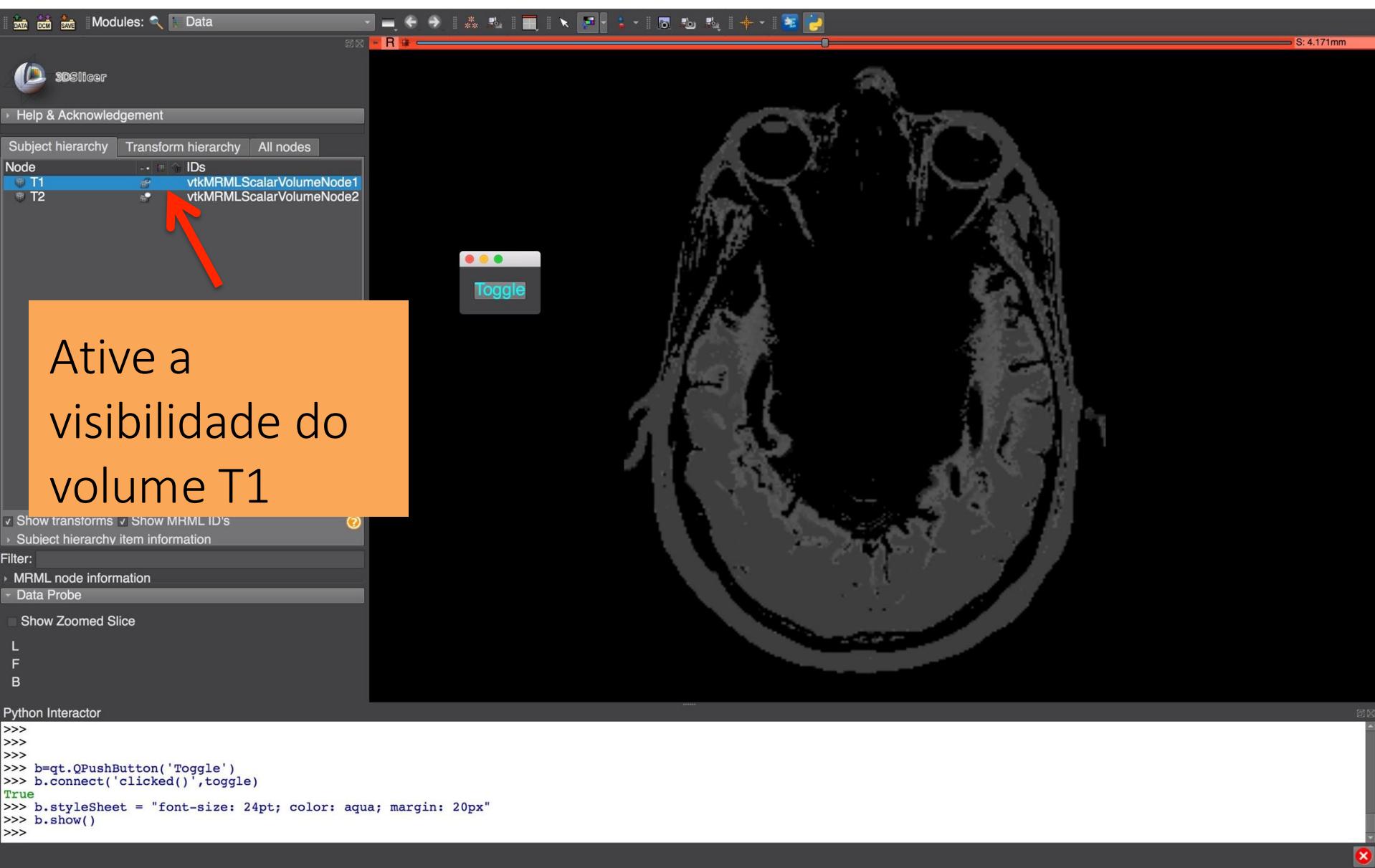
Python Interactor history:

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

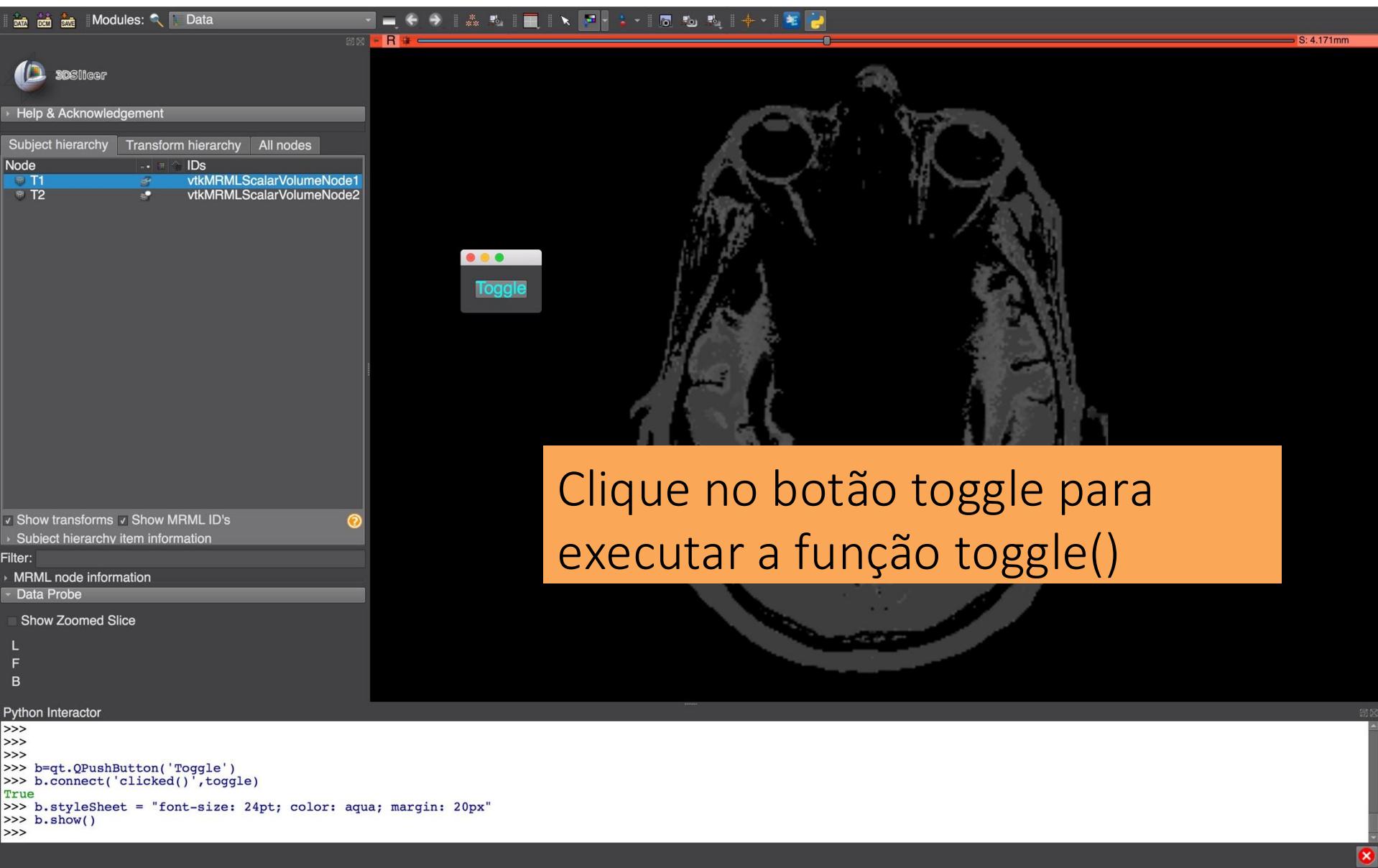
Criando um Botão de Push do Qt



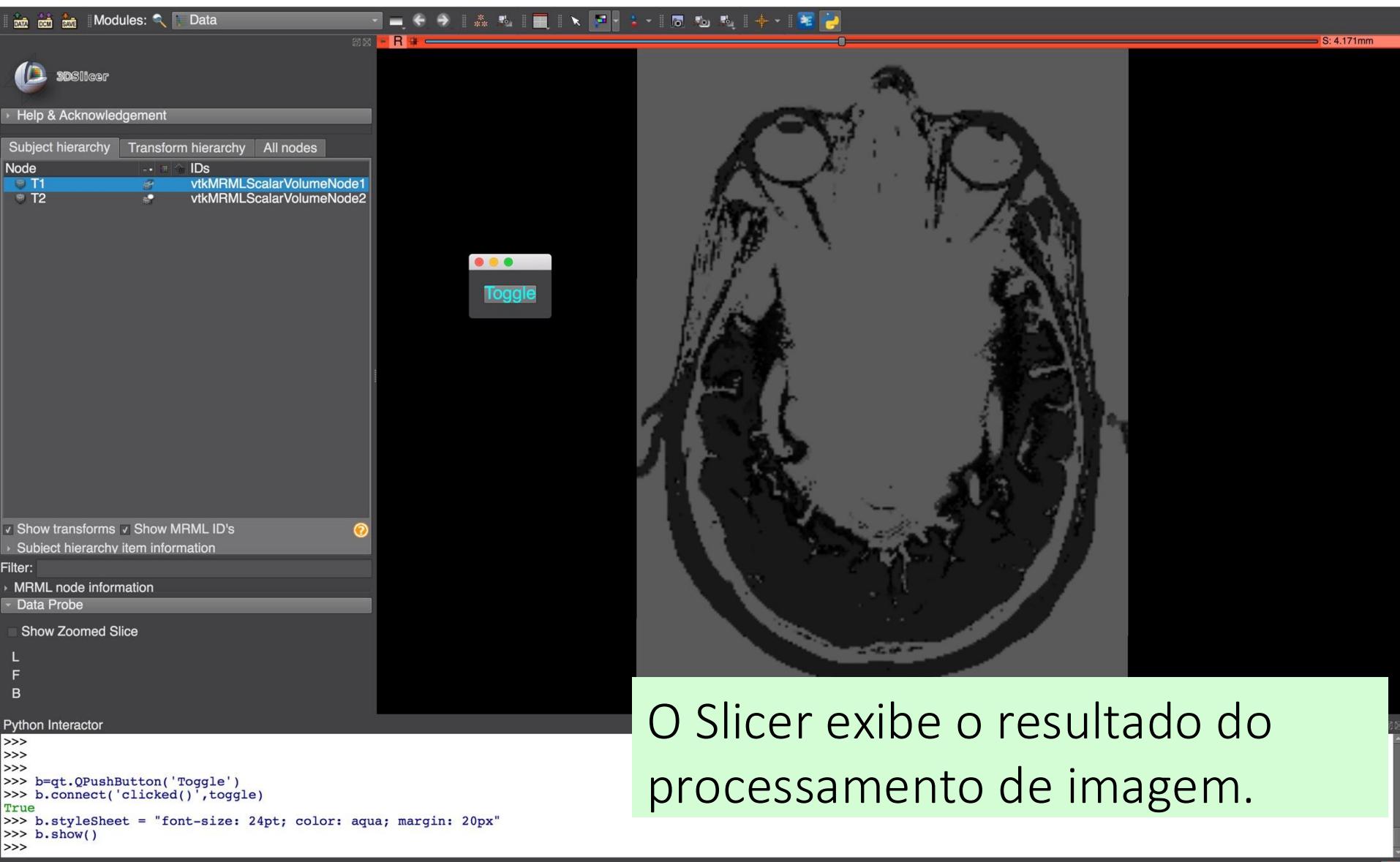
Criando um Botão de Push do Qt



Criando um Botão de Push do Qt



Criando um Botão de Push do Qt



Exemplos de módulos com *scripts*

- O tutorial demonstra como criar uma interface simples em Python.
- O Slicer integra muitos módulos com *scripts* sofisticados, como Segment Statistics, Sample Data, módulo de Endoscopia etc.
- Para mais informações, consulte o Repositório de Scripts do Slicer:
<https://www.slicer.org/wiki/Documentation/Nightly/ScriptRepository>

Conclusão

- O Slicer permite que os desenvolvedores criem interfaces complexas que são otimizadas para os usuários-alvo.
- A plataforma de software oferece possibilidades ilimitadas de personalização.
- O Slicer dá acesso a bibliotecas avançadas subjacentes por meio de um pacote multiplataforma que é fácil de implementar para os usuários finais.

Agradecimentos



Neuroimage Analysis Center

[Centro de Análise de Neuroimagem]

(NIBIB P41 EB015902)



Sylvain Bouix, Ph.D.

Psychiatry Neuroimaging Laboratory

[Laboratório de Neuroimagem Psiquiátrica]