Tradução

Google tradutor

Revisão

Alisson Barbosa Ferreira

Apresentação

Este é um tutorial de introdução para obter rapidamente o seu próprio sistema de documentação sphinx. Iremos baixar e instalar o sphinx, personalizando a aparência, usando extensões personalizadas para as parcelas de incorporação, diagramas de herança, sintaxe destaque sessões ipython e mais. Se você acompanhar o tutorial, você vai começar com nada e acabar com este site - é a documentação bootstrapping tutorial que se escreve!

Capitulo 1 - Começar

Instalar o diretório doc

Você pode já ter sphinx instalado - você pode verificar fazendo:

```
python -c 'import sphinx'
```

Se isso falhar baixe a versão mais recente e instale-o com:

```
> sudo easy_install -U Sphinx
```

Agora você está pronto para construir um modelo para seus documentos, utilizando sphinx-quickstart:

```
> sphinx-quickstart
```

aceite a maior parte dos padrões. Eu escolho "sampledoc" como o nome do meu projeto. CD em seu novo diretório e verifique o conteúdo:

```
home:~/tmp/sampledoc> ls
Makefile      _static       conf.py
_build        _templates    index.rst
```

O index.rst é o ReST mestre para seu projeto, mas antes de acrescentar alguma coisa, vamos ver se podemos construir alguns html:

```
make html
```

e agora você aponte seu navegador para _build / html / index.html, você deve ver um site sphinx basico.

Buscando os dados

Agora vamos começar a personalizar os docs. Pegue um par de arquivos do site web ou SVN. Você vai precisar getting_started.rst e _static / basic_screenshot.png. Todos os arquivos devem estar no "concluído" versão deste tutorial, mas uma vez que este é um tutorial, vamos agarrá-los um de cada vez, assim você pode saber o que precisa ser mudado. Uma vez que temos os arquivos, eu vou pegar o diretório SVN todo e copiar apenas os arquivos que eu preciso por enquanto. Primeiro, eu vou até cd de volta para o diretório que contém o meu projeto, confira o final " produto a partir do SVN, e copie apenas os arquivos que eu preciso no meu diretório sampledoc:

```
home:~/tmp/sampledoc> pwd

/Users/jdhunter/tmp/sampledoc
home:~/tmp/sampledoc> cd ..
home:~/tmp> svn co https://matplotlib.svn.sourceforge.net/svnroot/\
matplotlib/trunk/sampledoc_tut
A    sampledoc_tut/cheatsheet.rst
A    sampledoc_tut/_static
A    sampledoc_tut/_static/basic_screenshot.png
A    sampledoc_tut/conf.py
```

```
A    sampledoc_tut/Makefile
A    sampledoc_tut/_templates
A    sampledoc_tut/_build
A    sampledoc_tut/getting_started.rst
A    sampledoc_tut/index.rst
Checked out revision 7449.
home:~/tmp> cp sampledoc_tut/getting_started.rst sampledoc/
home:~/tmp> cp sampledoc_tut/_static/basic_screenshot.png \
sampledoc/_static/
```

O último passo é modificar index.rst para incluir o arquivo getting_started.rst (cuidado com o recuo, o "g" em "Getting_Started" deve alinhar com o ':' em: maxdepth:

```
Contents:
.. toctree::
   :maxdepth: 2

   getting_started.rst
```

e depois reconstruir os documentos:

```
cd sampledoc
make html
```

Quando você recarregar a página pelo seu navegador apontando para _build / html / index.html, você deve ver um link para o "Getting Started" docs, e aí esta página com a imagem. *Voila!*

Note que usamos a diretiva da imagem para incluir a imagem acima com:

```
.. image:: _static/basic_screenshot.png
```

A seguir, vamos personalizar o look and feel do nosso site para dar-lhe um logotipo, alguns css personalizado e atualizar os painéis de navegação para ficar mais parecido a sphinx próprio site.

Capitulo 2 - Personalizando a aparência do site

O site sphinx em si é melhor do que os sites criados com o padrão CSS, então aqui nós vamos chamar Elliots 'Maxim TS " O talento imita, mas o gênio rouba" e pegue o seu css e parte de seu layout. Como antes, você pode obter os arquivos necessários _static / default.css, _templates / layout.html e _static / logo.png a partir do site ou svn (veja *Obtendo os dados*). Uma vez eu fiz um checkout svn antes, vou apenas copiar as coisas que eu preciso de lá:

```
home:~/tmp/sampledoc> cp ../sampledoc_tut/_static/default.css _static/
home:~/tmp/sampledoc> cp ../sampledoc_tut/_templates/layout.html _templates/
home:~/tmp/sampledoc> cp ../sampledoc_tut/_static/logo.png _static/
home:~/tmp/sampledoc> ls _static/ _templates/
_static/:
basic_screenshot.png        default.css             logo.png

_templates/:
layout.html
```

Sphinx irá automaticamente pegar os arquivos HTML e CSS de layout, uma vez que colocá-los nos locais padrão com os nomes padrão, mas temos que incluir manualmente o logotipo em nosso layout.html. Vamos dar uma olhada no layout do arquivo: a primeira parte coloca uma barra de navegação horizontal no topo da nossa página, como você vê nos sites sphinx e matplotlib, a segunda parte inclui um logotipo que quando clicamos sobre ele nos levará *para a home* e a última parte move os painéis de navegação vertical do lado direito da página:

```
{% extends "!layout.html" %}

{% block rootrellink %}
    <li><a href="{{ pathto('index') }}">home</a>| </li>
    <li><a href="{{ pathto('search') }}">search</a>| </li>
   <li><a href="{{ pathto('contents') }}">documentation </a> &raquo;</li>
{% endblock %}
```

```
{% block relbar1 %}

<div style="background-color: white; text-align: left; padding: 10px 10px 15px
15px">
<a href="{{ pathto('index') }}"><img src="{{
pathto("_static/logo.png", 1) }}" border="0" alt="sampledoc"/></a>
</div>
{{ super() }}
{% endblock %}

{# put the sidebar before the body #}
{% block sidebar1 %}{{ sidebar() }}{% endblock %}
{% block sidebar2 %}{% endblock %}
```

Depois de reconstruir o local com um make html e recarregue a página em seu navegador, você deve ver um site parecido com este

Capitulo 3 - Sphinx extensões de terrenos incorporados, de matemática e mais

Sphinx é escrito em python e suporta a habilidade de escrever extensões personalizadas. Nós escrevemos um pouco para a documentação matplotlib, alguns dos quais fazem parte do matplotlib-se no módulo matplotlib.sphinxext, alguns dos quais estão incluídos apenas no diretório doc sphinx, e existem outras extensões escritas por outros grupos, por exemplo, numpy e ipython. Estamos coletando estas neste tutorial e mostrar-lhe como instalar e usá-las para seu próprio projeto. Primeiramente vamos pegar os arquivos de extensão a partir do diretório python sphinxext de svn (veja *Obtendo os dados*), e instalá-los em nosso projeto sampledoc sphinxext Diretório:

```
home:~/tmp/sampledoc> mkdir sphinxext

home:~/tmp/sampledoc> cp ../sampledoc_tut/sphinxext/*.py sphinxext/
home:~/tmp/sampledoc> ls sphinxext/
apigen.py             inheritance_diagram.py
docscrape.py          ipython_console_highlighting.py
docscrape_sphinx.py   numpydoc.py
```

Além das extensões builtin matplotlib para incorporar parcelas pyplot e renderização matemática com o motor de matemática matplotlib, também temos as extensões para o realce de sintaxe, sessões ipython, elaboração de diagramas inhertiance, e muito mais.

Precisamos informar o sphinx de nossas novas extensões no arquivo conf.py adicionando o seguinte. Primeiro vamos dizer-lhe onde encontrar as extensões:

```
# Se as extensões estão em outro diretório, adicioná-lo aqui.
# Se o diretório é relativo à raiz da documentação, use
# os.path.abspath para torná-lo absoluto, como mostrado aqui.
sys.path.append(os.path.abspath('sphinxext'))
```

E, então, dizer-lhe que as extensões de carga:

```
 # Adicione os nomes Sphinx módulo de extensão aqui, como cordas.  Eles podem
# Extensões que vem com o Sphinx (chamado 'sphinx.ext .*') ou o seu
# Mais personalizado.

extensions = ['matplotlib.sphinxext.mathmpl',
```

```
        'matplotlib.sphinxext.only_directives',
        'matplotlib.sphinxext.plot_directive',
        'sphinx.ext.autodoc',
        'sphinx.ext.doctest',
        'ipython_console_highlighting',
        'inheritance_diagram',
        'numpydoc']c
```

Agora vamos olhar para alguns destes em ação. Você pode ver a fonte literal para este arquivo no *Este arquivo* .

Sessões ipython

Michael Droettboom contribuiu com uma extensão de sphinx, que não pygments destaque de sintaxe em ipython sessões. Basta usar ipython como a linguagem em código fonte da diretiva:

```
.. sourcecode:: ipython

   In [69]: lines = plot([1,2,3])

   In [70]: setp(lines)
     alpha: float
     animated: [True | False]
     antialiased or aa: [True | False]
     ...snip
```

e você começará a sintaxe destaque saída abaixo.

```
In [69]: lines = plot([1,2,3])

In [70]: setp(lines)
  alpha: float
  animated: [True | False]
  antialiased or aa: [True | False]
  ...snip
```

Este apoio está incluído neste modelo, mas também será incluído em uma futura

versão do Pygments por padrão.

Usando matemática

No sphinx você pode incluir matemática inline $x \leftarrow y\, x \forall y\, x-y$ ou display math.

$$W^{3\beta}_{\delta_1 \rho_1 \sigma_2} = U^{3\beta}_{\delta_1 \rho_1} + \frac{1}{8\pi 2} \int_{\alpha_2}^{\alpha_2} d\alpha_2' \left[\frac{U^{2\beta}_{\delta_1 \rho_1} - \alpha_2' U^{1\beta}_{\rho_1 \sigma_2}}{U^{0\beta}_{\rho_1 \sigma_2}} \right]$$

Para incluir a matemática em seu documento, basta usar a diretiva de matemática, aqui está uma simples equação:

.. math::

W^{3\beta}_{\delta_1 \rho_1 \sigma_2} \approx U^{3\beta}_{\delta_1 \rho_1}

que é processado como

$$W^{3\beta}_{\delta_1 \rho_1 \sigma_2} \approx U^{3\beta}_{\delta_1 \rho_1}$$

Este quadro documentação inclui uma extensão de sphinx, sphinxext / mathmpl.py, que usa para processar matplotlib equações matemáticas ao gerar HTML e LaTeX-se ao gerar um PDF. Isso pode ser útil em sistemas que tenham matplotlib, mas não LaTeX, instalado. Para usá-lo, adicione mathmpl à lista de extensões em conf.py.

SVN versões atuais do Sphinx agora incluem suporte incorporado para a matemática. Existem dois tipos:

- pngmath: dvipng utiliza para processar a equação
- jsmath: torna a matemática no browser usando Javascript

Para usar essas extensões em vez disso, adicionar ou sphinx.ext.pngmath sphinx.ext.jsmath à lista de extensões em conf.py.

Todas essas três opções para a matemática são concebidos para comportar-se da mesma maneira.

Veja o matplotlib guia mathtext para mais informações sobre como escrever expressões matemáticas no matplotlib.

Inserindo parcelas matplotlib

Inserindo parcelas gerado automaticamente é fácil. Basta colocar o script para gerar o gráfico no diretório pyplots, e se referem a ele usando a diretiva parcela. Primeiro faça um diretório pyplots ao mais alto nível do seu projeto (junto: conf.py) e copie o arquivo `ellipses.py nele:

```
home:~/tmp/sampledoc> mkdir pyplots
home:~/tmp/sampledoc> cp ../sampledoc_tut/pyplots/ellipses.py pyplots/
```

Você pode se referir a esse arquivo na sua documentação esfinge, por padrão ela só vai em linha do enredo com links para a fonte e PF e PNGs de alta resolução. Para incluir o código fonte para a trama no documento, passe a incluir fonte de parâmetros:

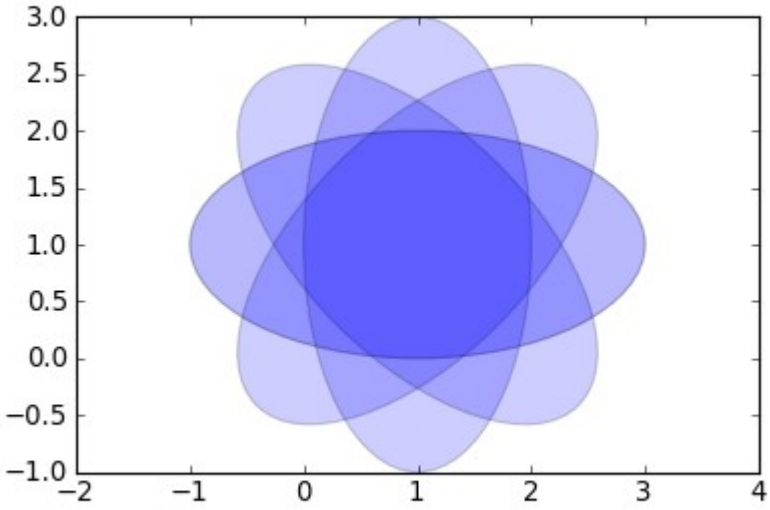
```
.. plot:: pyplots/ellipses.py
   :include-source:
```

Na versão HTML do documento, o lote inclui links para o código fonte original, um PNG de alta resolução e um PDF. Na versão PDF do documento, o enredo é incluída como PDF escalável.

```
from pylab import *
from matplotlib.patches import Ellipse
delta = 45.0 # degrees

angles = arange(0, 360+delta, delta)
ells = [Ellipse((1, 1), 4, 2, a) for a in angles]

a = subplot(111, aspect='equal')

for e in ells:
    e.set_clip_box(a.bbox)
    e.set_alpha(0.1)
    a.add_artist(e)

xlim(-2, 4)
ylim(-1, 3)
```

show()

[source code, hires.png, pdf]

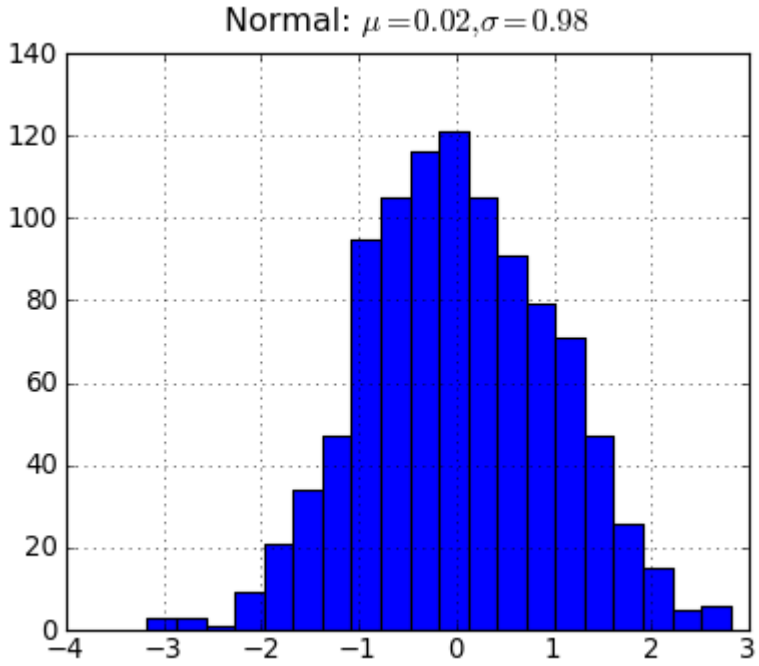


Você também pode inline código para as parcelas diretamente, eo código será executado na documentação do tempo de construção eo valor inserido em seus documentos, o seguinte código:

```
.. plot::

  import matplotlib.pyplot as plt
  import numpy as np
  x = np.random.randn(1000)
  plt.hist( x, 20)
  plt.grid()
  plt.title(r'Normal: $\mu=%.2f, \sigma=%.2f$'%(x.mean(), x.std()))
  plt.show()
```

[hires.png, pdf]

produz esta saída:



Veja o matplotlib pyplot tutorial e galeria de exemplos para os lotes de terrenos matplotlib.

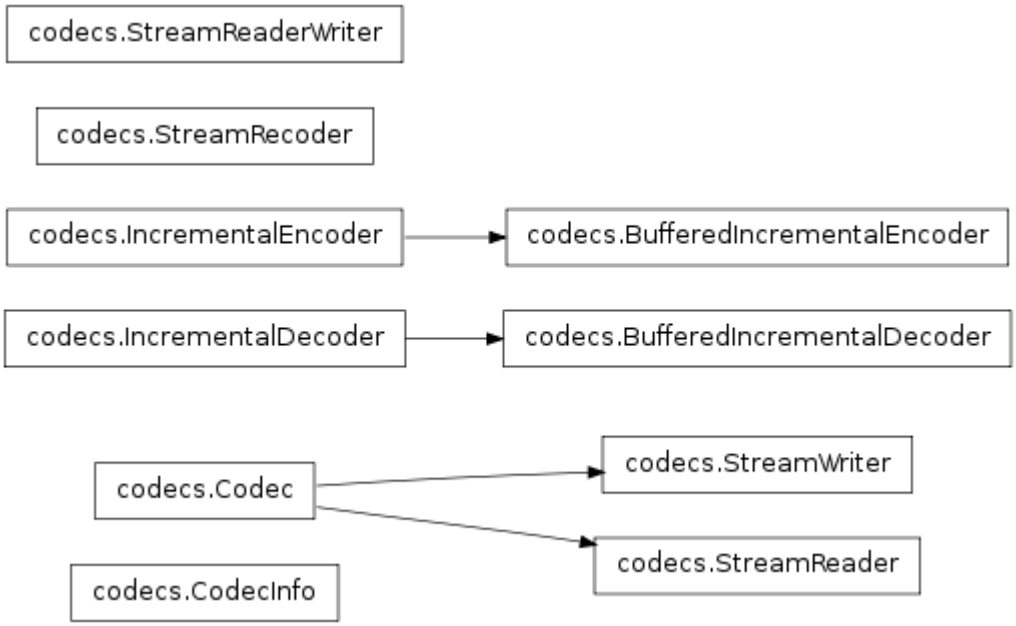
Diagramas herança

Diagramas de herança podem ser inseridos diretamente no documento, fornecendo uma lista de nomes de classe ou módulo para o diagrama de diretiva-herança.

Por exemplo:

.. inheritance-diagram:: codecs

produz:



Este arquivo

.. _extensions:

****
Sphinx extensions for embedded plots, math and more
****

Sphinx is written in python, and supports the ability to write custom extensions. We've written a few for the matplotlib documentation, some of which are part of matplotlib itself in the matplotlib.sphinxext module, some of which are included only in the sphinx doc directory, and there are other extensions written by other groups, eg numpy and ipython. We're collecting these in this tutorial and showing you how to install and use them for your own project. First let's grab the python extension files from the :file:`sphinxext` directory from svn (see :ref:`fetching-the-data`), and install them in our :file:`sampledoc` project :file:`sphinxext` directory::

    home:~/tmp/sampledoc> mkdir sphinxext
    home:~/tmp/sampledoc> cp ../sampledoc_tut/sphinxext/*.py sphinxext/
    home:~/tmp/sampledoc> ls sphinxext/

```
apigen.py               inheritance_diagram.py
docscrape.py            ipython_console_highlighting.py
docscrape_sphinx.py     numpydoc.py
```

In addition to the builtin matplotlib extensions for embedding pyplot plots and rendering math with matplotlib's native math engine, we also have extensions for syntax highlighting ipython sessions, making inhertiance diagrams, and more.

We need to inform sphinx of our new extensions in the :file:`conf.py` file by adding the following. First we tell it where to find the extensions::

```
# If your extensions are in another directory, add it here. If the
# directory is relative to the documentation root, use
# os.path.abspath to make it absolute, like shown here.
sys.path.append(os.path.abspath('sphinxext'))
```

And then we tell it what extensions to load::

```
# Add any Sphinx extension module names here, as strings. They can
# be extensions coming with Sphinx (named 'sphinx.ext.*') or your
# custom ones.
extensions = ['matplotlib.sphinxext.mathmpl',
        'matplotlib.sphinxext.only_directives',
        'matplotlib.sphinxext.plot_directive',
        'sphinx.ext.autodoc',
        'sphinx.ext.doctest',
        'ipython_console_highlighting',
        'inheritance_diagram',
        'numpydoc']
```

Now let's look at some of these in action. You can see the literal source for this file at :ref:`extensions-literal`.

.. _ipython-highlighting:

ipython sessions
================

Michael Droettboom contributed a sphinx extension which does `pygments
<http://pygments.org>`_ syntax highlighting on `ipython
<http://ipython.scipy.org>`_ sessions. Just use ipython as the
language in the ``sourcecode`` directive::

    .. sourcecode:: ipython

        In [69]: lines = plot([1,2,3])

        In [70]: setp(lines)
          alpha: float
          animated: [True | False]
          antialiased or aa: [True | False]
          ...snip

and you will get the syntax highlighted output below.

.. sourcecode:: ipython

    In [69]: lines = plot([1,2,3])

    In [70]: setp(lines)
      alpha: float
      animated: [True | False]
      antialiased or aa: [True | False]
      ...snip

This support is included in this template, but will also be included

in a future version of Pygments by default.

.. _using-math:

Using math
==========

In sphinx you can include inline math :math:`x\leftarrow y\ x\forall
y\ x-y` or display math

.. math::

  W^{3\beta}_{\delta_1 \rho_1 \sigma_2} = U^{3\beta}_{\delta_1 \rho_1} + \frac{1}{8 \pi 2} \int^{\alpha_2}_{\alpha_2} d \alpha^\prime_2 \left[\frac{ U^{2\beta}_{\delta_1 \rho_1} - \alpha^\prime_2U^{1\beta}_{\rho_1 \sigma_2} }{U^{0\beta}_{\rho_1 \sigma_2}}\right]

To include math in your document, just use the math directive; here is
a simpler equation::

   .. math::

     W^{3\beta}_{\delta_1 \rho_1 \sigma_2} \approx U^{3\beta}_{\delta_1 \rho_1}

which is rendered as

.. math::

  W^{3\beta}_{\delta_1 \rho_1 \sigma_2} \approx U^{3\beta}_{\delta_1 \rho_1}

This documentation framework includes a Sphinx extension,
:file:`sphinxext/mathmpl.py`, that uses matplotlib to render math
equations when generating HTML, and LaTeX itself when generating a
PDF. This can be useful on systems that have matplotlib, but not
LaTeX, installed. To use it, add ``mathmpl`` to the list of
extensions in :file:`conf.py`.

Current SVN versions of Sphinx now include built-in support for math.
There are two flavors:

- pngmath: uses dvipng to render the equation

- jsmath: renders the math in the browser using Javascript

To use these extensions instead, add ``sphinx.ext.pngmath`` or
``sphinx.ext.jsmath`` to the list of extensions in :file:`conf.py`.

All three of these options for math are designed to behave in the same
way.

See the matplotlib `mathtext guide
<http://matplotlib.sourceforge.net/users/mathtext.html>`_ for lots
more information on writing mathematical expressions in matplotlib.

.. _pyplots:

Inserting matplotlib plots
==========================

Inserting automatically-generated plots is easy. Simply put the
script to generate the plot in the :file:`pyplots` directory, and
refer to it using the ``plot`` directive. First make a
:file:`pyplots` directory at the top level of your project (next to
:``conf.py``) and copy the :file:`ellipses.py`` file into it::

    home:~/tmp/sampledoc> mkdir pyplots
    home:~/tmp/sampledoc> cp ../sampledoc_tut/pyplots/ellipses.py pyplots/

You can refer to this file in your sphinx documentation; by default it
will just inline the plot with links to the source and PF and high

resolution PNGS. To also include the source code for the plot in the document, pass the ``include-source`` parameter::

```
.. plot:: pyplots/ellipses.py
   :include-source:
```

In the HTML version of the document, the plot includes links to the original source code, a high-resolution PNG and a PDF. In the PDF version of the document, the plot is included as a scalable PDF.

.. plot:: pyplots/ellipses.py
   :include-source:

You can also inline code for plots directly, and the code will be executed at documentation build time and the figure inserted into your docs; the following code::

```
.. plot::

   import matplotlib.pyplot as plt
   import numpy as np
   x = np.random.randn(1000)
   plt.hist( x, 20)
   plt.grid()
   plt.title(r'Normal: $\mu=%.2f, \sigma=%.2f$'%(x.mean(), x.std()))
   plt.show()
```

produces this output:

.. plot::

   import matplotlib.pyplot as plt
   import numpy as np
   x = np.random.randn(1000)

```
   plt.hist( x, 20)
   plt.grid()
   plt.title(r'Normal: $\mu=%.2f, \sigma=%.2f$'%(x.mean(), x.std()))
   plt.show()
```

See the matplotlib `pyplot tutorial
<http://matplotlib.sourceforge.net/users/pyplot_tutorial.html>`_ and
the `gallery <http://matplotlib.sourceforge.net/gallery.html>`_ for
lots of examples of matplotlib plots.

Inheritance diagrams
====================

Inheritance diagrams can be inserted directly into the document by
providing a list of class or module names to the
``inheritance-diagram`` directive.

For example::

```
  .. inheritance-diagram:: codecs
```

produces:

.. inheritance-diagram:: codecs

.. _extensions-literal:

This file
=========

.. literalinclude:: extensions.rst

Capitulo 4 - Sphinx cheat sheet

Aqui está uma rápida cheat sheet para algumas coisas comuns que você quer fazer no sphinx e REST. Você pode ver a fonte literal para este arquivo no *Este arquivo* .

Formatação de texto

Você pode usar a marcação inline fazer texto *em itálico,* **negrito** ou monotipia.

Você pode representar blocos de código facilmente:

```
import numpy as np
x = np.random.rand(12)
```

Ou literalmente incluir o código:

```
from pylab import *
from matplotlib.patches import Ellipse

delta = 45.0 # degrees

angles = arange(0, 360+delta, delta)
ells = [Ellipse((1, 1), 4, 2, a) for a in angles]

a = subplot(111, aspect='equal')

for e in ells:
    e.set_clip_box(a.bbox)
    e.set_alpha(0.1)
    a.add_artist(e)

xlim(-2, 4)
ylim(-1, 3)

show()
```

Fazer uma lista

É fácil fazer listas em repouso

Bullet pontos

Este é um ponto de bala fazendo subseção

- point A
- point B
- point C

Enumerou pontos

Esta é uma subseção de fazer pontos numerados

1. point A
2. point B
3. point C

Fazendo uma tabela

Isso mostra como fazer uma tabela - se você quiser apenas fazer uma lista consulte *Fazer uma lista* .

Nome	**Age**
John D Hunter	40
Cast of Thousands	41
And Still More	42

Fazer ligações

É fácil fazer um link para o Yahoo ou a alguma seção dentro deste documento (ver *Fazendo uma tabela*) ou outro documento.

Você pode também classes de referência, módulos, funções, etc que são documentados usando o sphinx autodoc instalações. Por exemplo, consulte a documentação matplotlib.backend_bases módulo, ou o LocationEvent classe, ou o método mpl_connect ().

Este arquivo

.. _cheat-sheet:

****
Sphinx cheat sheet
****

Here is a quick and dirty cheat sheet for some common stuff you want
to do in sphinx and ReST. You can see the literal source for this
file at :ref:`cheatsheet
-literal`.

.. _formatting-text:

Formatting text
===============

You use inline markup to make text *italics*, **bold**, or ``monotype``.

You can represent code blocks fairly easily::

  import numpy as np
  x = np.random.rand(12)

Or literally include code:

.. literalinclude:: pyplots/ellipses.py

.. _making-a-list:

Making a list
=============

It is easy to make lists in rest

Bullet points

This is a subsection making bullet points

* point A

* point B

* point C

Enumerated points

This is a subsection making numbered points

#. point A

#. point B

#. point C

.. _making-a-table:

Making a table
==========

This shows you how to make a table -- if you only want to make a list see :ref:`making-a-list`.

Name	Age
John D Hunter	40
Cast of Thousands	41

And Still More 42
========== ==========

.. _making-links:

Making links
============

It is easy to make a link to `yahoo <http://yahoo.com>`_ or to some
section inside this document (see :ref:`making-a-table`) or another
document.

You can also reference classes, modules, functions, etc that are
documented using the sphinx `autodoc
<http://sphinx.pocoo.org/ext/autodoc.html>`_ facilites. For example,
see the module :mod:`matplotlib.backend_bases` documentation, or the
class :class:`~matplotlib.backend_bases.LocationEvent`, or the method
:meth:`~matplotlib.backend_bases.FigureCanvasBase.mpl_connect`.

.. _cheatsheet-literal:

This file
=========

.. literalinclude:: cheatsheet.rst

Capitulo 5 - Emacs apoio ReST

Emacs ajudantes

Existe um modo de emacs rst.elque automatiza muitas tarefas ReST importantes como a construção e updateing mesa-de-conteúdo, e promover ou rebaixar títulos da seção. Aqui está a base. Emacs configuração:

```
(require 'rst)
(setq auto-mode-alist
      (append '(("\\.txt$" . rst-mode)
                ("\\.rst$" . rst-mode)
                ("\\.rest$" . rst-mode)) auto-mode-alist))
```

Algumas funções úteis:

C-c TAB - rst-toc-insert

Inserir uma tabela de conteúdos no ponto

C-c C-u - rst-toc-update

Atualizar a tabela de conteúdos no ponto

C-c C-l rst-shift-region-left

região para a esquerda

C-c C-r rst-shift-region-right

região à direita