Instalação e configuração do IPython/Jupyter

Prof. Dr. Salviano A. Leão
Instituto de Física – UFG
email:salviano@ufg.br

22 de novembro de 2017

Sumário

1 Python e o Jupyter/IPython Notebook

1.1 Porque Python

Python é a linguagem de programação de escolha para muitos cientistas em grande medida porque oferece um grande poder para analisar e modelar dados científicos com relativamente pouca

sobrecarga em termos de aprendizagem, instalação ou tempo de desenvolvimento. É uma linguagem que você pode aprender o básico em um fim de semana, e usa-lá pelo resto de sua vida.

1.1.1 Documentação

O Python é uma linguagem muito bem documentada e há muitas opções em português:

- Python página oficial. Aqui ele disponibiliza um shell iterativo no qual voê pode rodar seu códgio sem precisar instalar ele em sua máquina.
- Python Brasil
- O livro "Python para desenvolvedores"
- Como Pensar Como um Cientista da Computação
- O livro "Think Python How to Think Like a Computer Scientist", pode ser baixado aqui versão em PDF
- O livro A byte of Python foi traduzido pelo Rodrigo Amaral.
- O Python Tutorial é um ótimo lugar para começar a ter uma idéia da linguagem.
- Aprendendo a Programar. Note entretanto que ela usa a versão 2.7 do Python
- The Runestone Interactive tools

1.2 O projeto Jupyter/IPython

Com o Projeto IPython montou-se uma interface de notebook que acho incrivelmente valiosa, por permite em um único local escrever um documento, analisar a os dados, fazer os gráfios, etc. Entretanto o projeto inical cresceu muito e ele então foi dividido em duas partes o Projeto IPython o qual ficou basicamente com o kernel do Python e o Jupyter, o qual inclui o kernel das outras quase 40 linguagens de programação conforme Jupyter kernels.

De acordo com o site do projeto, o IPython/Jupyter é uma aplicação de código aberto (BSD license) que permite criar e compartilhar documentos que contenham um código iterativo, equações e formulas matemáticas renderizadas pelo LaTeX, visualizações de imagens (via scikit) e gráficos (matplotlib) e textos explicativos. Atualmente o Jupyter notebook é um dos principais ambientes para a análise de dados, onde você pode usar não apenas Python, mas várias outras linguagens como você pode ver aqui.

O IPython fornece uma arquitetura rica para uma computação iterativa com: - Um poderoso shell iterativo. - Um kernel para o Jupyter. - Um suporte para visualização de dados iterativa e o use de GUI toolkits. - etc.

O IPython é um projeto crescente, com componentes cada vez mais agnósticas de linguagem. O IPython 3.x foi o último lançamento monolítico do IPython, contendo o servidor notebook, qtconsole, etc. A partir do IPython 4.0, as partes agnósticas linguísticas do projeto: o formato do notebook, o protocolo da mensagem, o qtconsole, a aplicação web do notebook, etc. mudaram-se para novos projetos sob o nome de Jupyter. O IPython ficou focado somente no Python interativo, parte do qual está fornecendo um kernel Python para o Jupyter.

Atualmente no GitHub em uma pesquisa rápida pode-se encontrar centenas de notebooks, pois há uma comunidade de usuários muito grade dipostos a compartilhar seu trabalho. Um grande número de pessoas têm disponibilizado muitos Jupyter/IPython Notebooks os quais serviram de apoio na montagem deste. Usei um grande número de material para a construção deste, particularmente destaco os seguintes:

- Hans Fangohr Python for Computational Science and Engineering Uma execelente página e

contém diversos elementos de programção em python e algumas dicas de bibliotecas para cáculo e visualização.
- Rob Johansson's excellent notebooks, incluindo Scientific Computing with Python e Computational Quantum Physics with QuTiP lectures;
- XKCD style graphs in matplotlib;
- A collection of Notebooks for using IPython effectively
- A gallery of interesting IPython Notebooks

1.3 O que você precisa instalar

Existem atualmente duas versões do Python que estão convivendo entre-si: a sintaxe mais antiga Python 2 e a sintaxe mais recente Python 3. Essa esquizofrenia é em grande parte intencional: quando ficou claro que algumas mudanças não compatíveis com a linguagem eram necessárias, a Python dev-team decidiu passar por uma transição de cinco anos (ou assim), durante o qual os novos recursos de linguagem seria introduzida e a versão 2 da linguagem ainda seria ativamente mantida, para fazer essa transição tão fácil quanto possível.

Com isso em mente, essas notas assumem que você tem uma distribuição Python que inclui:

- Python versão $\geq$ 3.5;
- Numpy, o núcleo de extensões numéricas para álgebra linear e arrays multidimensionais;
- Scipy, bibliotecas adicionais para programação científica;
- Matplotlib, excelente traçado e bibliotecas de gráficos;
- Pandas, é uma biblioteca de código aberto (BSD-licensed), fácil de usar na análise e estruturas de dados, para a lingugem python.
- StatsModels é um módulo Python que fornece classes e funções para estimativas de muitos modelos estatísticos diferentes, assim como testes estatísctícos e exploração estatística de dados.
- scikit-learn é uma ferramente simples e eficiente para mineração e análise de dados em Python.
- Jupyter/IPython, com as bibliotecas adicionais necessárias para a interface do notebook. O projeto do IPythone evolui e agor é chamado de Jupyter

2 Instalando o Jupyter Notebook

A seguir será explicado como instalar o Jupyter Notebook e o kernel do IPython em uma máquina Linux baseada no Debian, como Debian, Ubuntu, LinuxMint, etc.

2.1 Pré-requisito: Python

O IPython/Jupyter executa códigos de muitas linguagens de programação, entretanto, o Python é um requisito necessário (Python 3.5 ou superior, ou Python 2.7) para instalar o Jupyter Notebook. Nas distribuições GNU/Linux baseadas no Debian, assim como o Ubuntu e familia, o Linux Mint, etc, a instalação dos seguintes pacotes é essencial para o desenvolvimento do curso:

```
sudo apt-get update
sudo apt-get install -y python3 python3-all python3-pip python3-setuptools python3-markdown
sudo apt-get install -y ipython3 ipython3-notebook ipython3-qtconsole python3-all-dev
sudo apt-get install -y pandoc pandoc-data python3-pandocfilters python3-sphinx
```

Eu gosto de instalar a documentação, exemplos e os pacotes para debugs:

```
sudo apt-get install -y python3-doc python3-examples python3-all-dbg
```

Aqui vai um rápida descrição das bibliotecas acima: - *python3-setuptools* é pacote que permite emapcotar/instalar um aplicativo/biblioiteca python - *python3-markdown* é a liguagem usada nos IPython/Jupyter notebooks - *python3-pandocfilters* é uma ligação python direta com o conversor de linguagem de marcação o *pandoc* - *python3-sphinx* é um gerador de documentação para os projetos python, usando textos restruturados com liguagens de marcação.

OBS.: O IPython/Jupyter podem exportar os notebooks no formato pdf, gerando um artigo elegante, entretanto para tal ele usa o LaTeX, logo para quem deseja usar dessa funcionalidade recomenda-se que instale o texlive no Linux, no sistema operacional Windows recomenda-se o MikTeX ou o próprio TeX LIve.

2.2 PyPi

As duas características mais relevantes do Python são a sua capacidade de integração com outras linguagens e seu sistema de pacote maduro que é bem incorporado pelo PyPI (O Índice do Pacote Python), um repositório comum Para a maioria dos pacotes do Python. O pacote *python3-pip* é que nos permite termos acesso a esse repositório. Pois nem todos os pacotes Python possuem uma versão no repositório da sua distribuição Linux.

Digitando somente `pip` no terminal,

```
> pip3
Usage:
  pip <command> [options]

Commands:
  install                     Install packages.
  download                    Download packages.
  uninstall                   Uninstall packages.
  freeze                      Output installed packages in requirements format.
  list                        List installed packages.
  show                        Show information about installed packages.
  check                       Verify installed packages have compatible dependencies.
  search                      Search PyPI for packages.
  wheel                       Build wheels from your requirements.
  hash                        Compute hashes of package archives.
  completion                  A helper command used for command completion.
  help                        Show help for commands.

General Options:
  -h, --help                  Show help.
  --isolated                  Run pip in an isolated mode, ignoring
                              environment variables and user configuration.
  -v, --verbose               Give more output. Option is additive, and can be
                              used up to 3 times.
  -V, --version               Show version and exit.
  -q, --quiet                 Give less output. Option is additive, and can be
```

```
                              used up to 3 times (corresponding to WARNING,
                              ERROR, and CRITICAL logging levels).
  --log <path>                Path to a verbose appending log.
  --proxy <proxy>             Specify a proxy in the form
                              [user:passwd@]proxy.server:port.
  --retries <retries>         Maximum number of retries each connection should
                              attempt (default 5 times).
  --timeout <sec>             Set the socket timeout (default 15 seconds).
  --exists-action <action>    Default action when a path already exists:
                              (s)witch, (i)gnore, (w)ipe, (b)ackup, (a)bort.
  --trusted-host <hostname>   Mark this host as trusted, even though it does
                              not have valid or any HTTPS.
  --cert <path>               Path to alternate CA bundle.
  --client-cert <path>        Path to SSL client certificate, a single file
                              containing the private key and the certificate
                              in PEM format.
  --cache-dir <dir>           Store the cache data in <dir>.
  --no-cache-dir              Disable the cache.
  --disable-pip-version-check
                              Don't periodically check PyPI to determine
                              whether a new version of pip is available for
                              download. Implied with --no-index.
```

Então por exemplo, para pesquisar o pacote `qutitp`

```
> pip3 search qutip
qutip (4.2.0)  - QuTiP: The Quantum Toolbox in Python
  INSTALLED: 4.2.0 (latest)
```

Esse como pode-se ver já está instalado, mas o pacote `statsmodel`

```
> pip3 search statsmodel
scikits.statsmodels (0.3.1)  - Statistical computations and models for use with SciPy
statsmodels (0.8.0)          - Statistical computations and models for Python
```

Mostra que há dois pocatoes disponíveis os quais não estão instalados. Para instalar use

```
> sudo -H pip3 install statsmodels
```

e ele irá instalar o pacote para todos os usuários do sistema, a opção `-H` do sudo. Note que o comando

```
> pip3 install statsmodels
```

irá instalar o pacote localmente na área do usuário.

OBS: Na instalação use a opção `-U` ou `--upgrade` para força todos os pacotes a fazerem um *upgrade* para a versão mais nova.

Para desinstalar o pacote `statsmodels` proceda da seguinte forma:

```
> sudo -H pip3 uninstall statsmodels
```

Tanto na instalação

```
> sudo -H pip3 install statsmodels
```

Para obter informações do pacote use a opção `show`

```
> pip3 show qutip
Name: qutip
Version: 4.2.0
Summary: QuTiP: The Quantum Toolbox in Python
Home-page: http://qutip.org
Author: Alexander Pitchford, Paul D. Nation, Robert J. Johansson, Chris Granade, Arne Grimsmo
Author-email: alex.pitchford@gmail.com, nonhermitian@gmail.com, jrjohansson@gmail.com, cgranade@
License: BSD
Location: /usr/local/lib/python3.5/dist-packages
Requires: numpy, scipy, cython
```

2.3 Instalação

Para os usuários menos experientes, a página de documentação do Jupyter Notebook recomenda que se faça uso da distribuição Anaconda para instalar o Python e o Jupyter, por ser fácil de instalar em qualquer que seja seu sistema operacional. Uma outra boa opção que também é fácil de instalar e que suporta Mac, Windows e Linux, e que tem todos esses pacotes (e muito mais) é a Entought Python Distribution, também conhecido como EPD, que parece estar mudando seu nome para Enthought Canopy. Enthought é uma empresa comercial que suporta um monte de muito bom trabalho em desenvolvimento e aplicação científica Python. Você pode adquirir uma licença para usar EPD, ou há também uma versão gratuita que você pode baixar e instalar.

A seguir descrevemos como instalar o jupyter.

2.3.1 Via o Anaconda e conda

Os usuários de outros sistemas operacionais e também do linux podem optar por uma distribuição do python e suas bibliotecas chamada Anaconda. A página oficial do Jupyter Notebook recomenda, que se instale o Anaconda. Isso deve-se ao fato de que o Anaconda convenientemente instala Python, o Jupyter Notebook, e outros pacotes comumente usados para computação científica e nas ciências de análise de dados.

Use as seguintes etapas de instalação:

- Faça o download do Anaconda. Recomenda-se que você baixe a última versão do Python 3 da Anaconda (atualmente Python 3.6).
- Instale a versão do Anaconda que você baixou, seguindo as instruções na página de download.
- Ao término da etapa anterior, você terá instalado o Jupyter Notebook. Para executar o notebook, digite na linha de comando:

  jupyter-notebook

Aqui temos alguns sites que detalham como instalar o anaconda

- Como instalar o anaconda
- Como instalar Python e Jupyter Notebook usando Anaconda
- How to install anaconda on ubuntu 16.04

2.3.2 Usuários Linux: Instalar o Jupyter com o pip

Importante A instalação do Jupyter requer Python 3.3 ou superior, ou Python 2.7. IPython 1.x, que incluiu as partes que mais tarde se tornou Jupyter, foi a última versão para suportar Python 3.2 e 2.6.

Como um usuário mais experiente do Python, você pode querer instalar o Jupyter usando o gerenciador de pacotes do Python, pip, em vez de Anaconda.

Nos ambientes GNU/Linu, primeiro, certifique-se de que você tem o pip mais recente; Versões mais antigas podem ter problemas com algumas dependências:

```
sudo -H pip3 install --upgrade pip3
```

ou

```
sudo -H pip3 install -U pip3
```

Em seguida, instale o Jupyter Notebook usando:

```
sudo -H pip3 install jupyter
```

(Use pip se estiver usando o Python 2.)
Para instalar algumas extensões do jupyter notebook consulte:

- Jupyter notebook extensions
- Notebook extensions
- Installing Jupyter Notebook Extensions
- An easy way to install Jupyter Notebook extensions
- Installing jupyter_contrib_nbextensions
- Distributing Jupyter Extensions as Python Packages

2.4 Bibliotecas essenciais

Bibliotecas científicas que todos devem instalar:

```
sudo apt-get install -y python3-matplotlib python3-numpy python3-numpydoc
sudo apt-get install -y python3-pandas python3-pandas-lib python3-mpmath
sudo apt-get install -y python3-numexpr python3-scipy python3-sympy
sudo apt-get install -y python3-pyfftw python3-opengl python3-seaborn
sudo apt-get install -y python3-matplotlib-dbg python3-numpy-dbg python3-numexpr-dbg
```

seus pacotes para permitir debugs:

```
sudo apt-get install -y python3-matplotlib-dbg python3-numpy-dbg python3-numexpr-dbg
```

Alguns pacotes úteis para tratar imagens, textos html, etc:

```
sudo apt-get install -y python3-html2text python3-html5lib python3-urllib3 python3-bs4
sudo apt-get install -y python3-pdfrw python3-pil python3-pil.imagetk python3-pilkit
sudo apt-get install -y python3-img2pdf python3-skimage-lib python3-sklearn-lib
sudo apt-get install -y python3-pil-dbg python3-pil.imagetk-dbg python3-pyfftw-dbg
```

2.4.1 Descrição das bibliotecas mais importantes

NumPy NumPy, que é a criação de Travis Oliphant, é o verdadeiro laboratório analítico da linguagem Python. Ele fornece ao usuário arrays multidimensionais, juntamente com um grande conjunto de funções para lidar com uma multiplicidade de operações matemáticas sobre matrizes ("arrays"). Arrays são blocos de dados dispostos ao longo de múltiplas dimensões, que implementam vetores matemáticos e matrizes. As matrizes são úteis não apenas para armazenar dados, mas também para operações de matriz rápida (vectorization), que são indispensáveis quando você deseja resolver problemas ad hoc de ciência de dados.

Site: http://www.numpy.org/ Versão atual: 1.9.1

> Nota: Como uma convenção amplamente adotada pela comunidade Python, ao importar NumPy, sugere-se que você o alias como np:
>
> import numpy as np
>
> Estaremos fazendo isso ao longo deste.

SciPy Um projeto original de Travis Oliphant, Pearu Peterson e Eric Jones, SciPy completa as funcionalidades do NumPy, oferecendo uma maior variedade de algoritmos científicos para álgebra linear, matrizes esparsas, processamento de sinais e imagens, otimização, transformada rápida de Fourier e muito mais.

Site: http://www.scipy.org/ Versão atual: 0.14.0

Matplotlib Originalmente desenvolvido por John Hunter, matplotlib é a biblioteca que contém todos os blocos de construção que são necessários para criar gráficos de qualidade a partir de arrays e visualizá-los interativamente.

Você pode encontrar todas as estruturas de plotagem de tipo MATLAB dentro do módulo pylab.

Website: http://matplotlib.org/ Versão atual: 1.4.2

> Você pode simplesmente importar o que você precisa para suas finalidades de visualização com o seguinte comando:
>
> import matplotlib.pyplot as plt

SymPy SymPy é uma biblioteca Python para matemática simbólica. Ele pretende tornar-se um sistema completo de álgebra computacional (CAS), mantendo o código o mais simples possível para ser compreensível e facilmente extensível. A SymPy foi escrita inteiramente em Python. Você não precisa instalar o SymPy para iniciar o seu uso. Você pode usar o shell on-line em http://live.sympy.org.

Página: http://www.sympy.org/

Pandas O pacote pandas lida com tudo o que NumPy e SciPy não pode fazer. Graças às suas estruturas de dados de objetos específicos, DataFrames e Series, pandas permite manipular tabelas complexas de dados de diferentes tipos (que é algo que as matrizes de NumPy não podem fazer) e séries temporais. Graças à criação de Wes McKinney, você será capaz de facilmente e sem problemas carregar dados de uma variedade de fontes. Você pode então cortar, fragmentar, manipular elementos ausentes, adicionar, renomear, agregar, reformular e, finalmente, visualizar esses dados em sua vontade.

Website: http://pandas.pydata.org/ Versão atual: 0.15.2

Convencionalmente, os pandas são importados como pd:

import pandas as pd

Beautiful Soup Beautiful Soup, uma criação de Leonard Richardson, é uma ótima ferramenta para tratar dados de arquivos HTML e XML obtidos da Internet. Funciona incrivelmente bem, mesmo no caso das sopas de tags (daí o nome), que são coleções de etiquetas malformadas e contraditórias e incorretas. Depois de escolher o analisador (basicamente, o analisador HTML incluído na biblioteca padrão do Python funciona bem), graças à Beautiful Soup, você pode navegar pelos objetos na página e extrair texto, tabelas e qualquer outra informação que possa ser útil.

Site: http://www.crummy.com/software/BeautifulSoup/ Versão atual: 4.3.2 O pacote python: python3-bs4

Nota: Observe que o módulo importado é denominado bs4. from bs4 import BeautifulSoup

Scikit-learn Iniciado como parte do SciKits (SciPy Toolkits), Scikit-learn é o núcleo das operações de ciência de dados em Python. Ele oferece tudo o que você pode precisar em termos de pré-processamento de dados, aprendizado supervisionado e não supervisionado, seleção de modelos, validação e métricas de erro. O Scikit-learn começou em 2007 como um projeto Google Summer of Code de David Cournapeau. Desde 2013, foi retomado pelos investigadores do INRA (Instituto Francês de Investigação em Informática e Automação).

Website: http://scikit-learn.org/stable/ Versão atual: 0.15.2 instalar via pip3 e o nome do pacote é: scikit-learn

Nota: Observe que o módulo importado é denominado sklearn.

Statsmodels Anteriormente parte de SciKits, statsmodels foi pensado para ser um complemento para SciPy funções estatísticas. Ele apresenta modelos lineares generalizados, modelos de escolha discreta, análise de séries temporais e uma série de estatísticas descritivas, bem como testes paramétricos e não paramétricos.

Site: http://statsmodels.sourceforge.net/ Versão atual: 0.6.0 Instalação via pip3 e o nome do pacote é: - scikits.statsmodels (0.3.1) - Statistical computations and models for use with SciPy - statsmodels (0.8.0) - Statistical computations and models for Python

Seaborn É uma biblioteca para visualização de dados estatísticos, baseada na matplotlib.

Site:https://seaborn.pydata.org/ Versão atual: 0.8.1 Instalada via pip, mas possui um pacote debian

NetworkX Desenvolvido pelo Laboratório Nacional de Los Alamos, NetworkX é um pacote especializado na criação, manipulação, análise e representação gráfica de dados de rede da vida real (pode facilmente operar com gráficos compostos de um milhão de nós e bordas). Além de estruturas de dados especializadas para gráficos e métodos de visualização fina (2D e 3D), ele fornece ao usuário muitas medidas de gráfico padrão e algoritmos, como o caminho mais curto, centralidade, componentes, comunidades, cluster e PageRank.

Site: https://networkx.github.io/ Versão atul: 1.9.1 pacote: python3-networkx

Convencionalmente, o NetworkX é importado como nx:

import networkx as nx

```
In [ ]: %%writefile install_python.sh
        #!/bin/sh
        sudo apt-get update
        sudo apt-get install -y python3 python3-doc python3-all python3-pip python3-bs4 python3-
skimage-lib python3-sklearn-lib python3-seaborn python3-sphinx
```

```
In [1]: ! cat install_python.sh
```

```
#!/bin/sh
sudo apt-get update
sudo apt-get install -y python3 python3-doc python3-all python3-pip python3-bs4 python3-all-dev
```

Agora para instalar o script abra um terminal na pasta onde enoctra-se o scpript e digite:

```
sh ./install_python.sh
```

2.5 Ajustando a configuração do Linux

Uma das habilidades do Jupyter Notebook reside no fato de que não é necessário deixar o ambiente para realizar muitas das tarefas do sistema, mesmo a instalação de softwares. Entretanto nesse caso é necessário fazer um pequeno ajuste no sistema. Vamos permitir que o administrador da máquina que é quem está usando ela não precise mais de digitar o seu password ao usar o sudo. Para isso edite o arquivo /etc/sudoers e acrescente uma linha contendo

```
login_do_usuario  ALL=(ALL:ALL) NOPASSWD:ALL
```

Devendo ficar conforme o exemplo abaixo, no qual o meu login é salviano.
Blog do Da2k Esse é um pequeno texto em *azul*

```
In [2]: !sudo cat /etc/sudoers
```

```
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/

# Host alias specification

# User alias specification
```

```
# Cmnd alias specification

# User privilege specification
root        ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo        ALL=(ALL:ALL) ALL
salviano ALL=(ALL:ALL) NOPASSWD:ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d
```

Para ver o seu login:

```
In [3]: ! echo $USER

salviano
```

Considerando que o resultado do comando acima, ou seja o login do usuário, seja `pedro`, então para ajustar o arquivo `/etc/sudoers` abra um terminal e digite:

```
sudo sed -i "/^%sudo/ a\pedro  ALL=(ALL:ALL) NOPASSWD:ALL" /etc/sudoers
```

ou

```
sudo sed -i "/^%sudo/ a\ $USER  ALL=(ALL:ALL) NOPASSWD:ALL" /etc/sudoers
```

A seguir verifique se o comando rodou corretamente. Para isso abra um terminal e digite:

```
sudo cat /etc/gshadow
```

Se o comando acima não te pedir a senha e mostar a listagem do arquivo `/etc/gshadow`, então pode continuar e executar a próxima célula.

Uma vez feito isso teste verificando se o sistema não está pedindo o password e para isso abra um terminal e digite:

```
sudo apt update
```

Se o comando rodou sem te pedir a senha então, pode continuar e executar a próxima célula.

```
In [4]: !sudo -H pip3 install version_information

Requirement already satisfied: version_information in /usr/local/lib/python3.5/dist-packages
```

2.6 Verificando a instalação

ISe você instalou tudo corretamente, então abra um terminal e digite

jupyter-notebook

Ele então vai abrir o navegador e você terá o jupyter funcionando.

Para verificar a versão atual instalada dos pacotes selecionados, considere por exemplo: o numpy, scipy, matplotlib, sympy, pandas, ipython, jupyter. Digite o comando abaixo na célula do Jupyter

```
In [5]: %load_ext version_information
        %version_information numpy, scipy, matplotlib, sympy, pandas, ipython, jupyter
```

Out[5]:

Software	Version
Python	3.5.2 64bit [GCC 5.4.0 20160609]
IPython	6.2.1
OS	Linux 4.10.0 40 generic x86_64 with LinuxMint 18.2 sonya
numpy	1.13.3
scipy	1.0.0
matplotlib	2.1.0
sympy	1.1.1
pandas	0.21.0
ipython	6.2.1
jupyter	1.0.0
Wed Nov 22 07:48:14 2017 -02	

3 Jupyter/IPython

Jupyter/IPython (Python interativo) é um shell Python aprimorado que fornece um ambiente de desenvolvimento mais robusto e produtivo para os usuários. Existem várias características-chave que o diferenciam do shell padrão do Python.

3.0.1 História

No Jupyter/IPython, todas as suas entradas e saídas são salvas. Há duas variáveis chamadas `In` e`Out` que são atribuídas à medida que você trabalha com seus resultados. Todas as saídas são salvas automaticamente em variáveis da forma `_N`, onde `N` é o número do prompt, e as entradas para `_iN`. Isso permite que você recupere rapidamente o resultado de uma computação anterior, referindo-se a seu número, mesmo se você se esqueceu de armazená-lo como uma variável.

```
In [6]: from math import sqrt
        sqrt(15.0)
```

```
Out[6]: 3.872983346207417
```

Para referenciar o resultado da célula anterior, faça

```
In [7]: _6 * 2.0

Out[7]: 7.745966692414834
```

Você também pode referenciar a entrada, nesse caso considere a entrada da célula 3 anteriror, e nesse caso observe que ela é uma string, logo

```
In [8]: _i6

Out[8]: 'from math import sqrt\nsqrt(15.0)'

In [9]: _6 / 4.

Out[9]: 0.9682458365518543

In [10]: import math
         math.sin(3.0)  # Note o uso. Aqui carregou-se toda a biblioteca

Out[10]: 0.1411200080598672

In [12]: sqrt(13.0)

Out[12]: 3.605551275463989
```

Os notebooks são iterativos

```
In [13]: a=input("Entre com o seu nome")
         print("O nome entrado foi: ",a)

Entre com o seu nomeSalviano
O nome entrado foi:  Salviano
```

3.0.2 A saída é assíncrona

Toda a saída é exibida de forma assíncrona à medida que é gerada no Kernel. Se você executar a próxima célula, você verá a saída uma peça de cada vez, nem todas no final.

```
In [14]: import time
         inicio = 1
         fim = 9
         passo = 1
         for i in range(inicio,fim,passo):
             print("i = ", i)
             time.sleep(0.5) # usado para fornecer um pequeno delay

i =  1
i =  2
i =  3
i =  4
i =  5
i =  6
i =  7
i =  8
```

3.0.3 Introspecção

Se desejar detalhes sobre as propriedades e funcionalidades de qualquer objeto Python atualmente carregado no Jupyter/IPython, você pode usar o ? Para revelar quaisquer detalhes disponíveis:

```
In [15]: i?
```

```
In [16]: algum_dic = {}
         algum_dic?
```

Se estiverem disponível, os detalhes adicionais serão fornecidos com dois pontos de interrogação, incluindo o código-fonte do próprio objeto.

```
In [ ]: from numpy.linalg import cholesky
        cholesky??
```

Essa sintaxe também pode ser usada para pesquisar *namespaces* com curingas (*).

```
In [17]: %matplotlib inline
         import pylab as plt
         plt.*plot*?
```

```
In [18]: %matplotlib --list
```

```
Available matplotlib backends: ['notebook', 'qt4', 'agg', 'ipympl', 'qt', 'osx', 'gtk', 'widget'
```

As opções de visualização da matplotlib são:
'auto', 'gtk', 'gtk3', 'inline', 'nbagg', 'notebook', 'osx', 'qt', 'qt4', 'qt5', 'tk', 'wx'
%matplotlib --list Available matplotlib backends: ['osx', 'qt4', 'qt5', 'gtk3', 'notebook', 'wx', 'qt', 'nbagg', 'gtk', 'tk', 'inline']
positional arguments: gui Name of the matplotlib backend to use ('agg', 'gtk', 'gtk3', 'inline', 'ipympl', 'nbagg', 'notebook', 'osx', 'qt', 'qt4', 'qt5', 'tk', 'wx'). If given, the corresponding matplotlib backend is used, otherwise it will be matplotlib's default (which you can set in your matplotlib config file).
optional arguments:

3.0.4 Completar com Tab

Como o Jupyter/IPython permitem a introspecção, ele é capaz de dar ao usuário a capacidade de completar comandos que foram parcialmente digitados com um tab. Isso é feito pressionando a tecla <tab> em qualquer ponto durante o processo de digitar um comando:

```
In [ ]: plt.p
```

Isso pode até ser usado para ajudar com a especificação dos argumentos de algumas funções, o que às vezes pode ser difícil de lembrar:

```
In [ ]: plt.hist
```

3.0.5 Comandos do sistema

No Jupyter/IPython, você pode digitar `ls` para ver seus arquivos ou `cd` para alterar diretórios, assim como você faria em um prompt do sistema, mas para isso adicione o caracter `!`:

```
In [ ]: !ls /home/salviano/Cursos/FisComp/Python/ipython/Cursos/
```

Praticamente qualquer comando do sistema pode ser acessado prepending `!`, o qual passa qualquer comando subsequente diretamente para o sistema operacional.

```
In [ ]: !locate python3 | grep pdf
```

Você pode até usar variáveis Python em comandos enviados para o sistema operacional:

```
In [19]: file_type = 'ipynb'
         !ls ./*$file_type

./01-Instalacao-e-configuracao-do-IPython-e-Jupyter.ipynb
./Exemplo-de-notebook-em-c.ipynb
./Fortran_Exemplo-notebook.ipynb
```

A saída de um comando do sistema usando a sintaxe do ponto de exclamação pode ser atribuída a uma variável Python.

```
In [20]: notebooks = !ls ./

In [21]: notebooks

Out[21]: ['01-Instalacao-e-configuracao-do-IPython-e-Jupyter.ipynb',
          'Cab_IF_Red.png',
          'contour_frontpage.png',
          'Exemplo-de-notebook-em-c.ipynb',
          'Fortran_Exemplo-notebook.ipynb',
          'funcs.mod',
          'install_python.sh',
          'LICENSE',
          'README.md',
          'README.md~']

In [22]: type(notebooks)

Out[22]: IPython.utils.text.SList
```

3.1 Console Qt

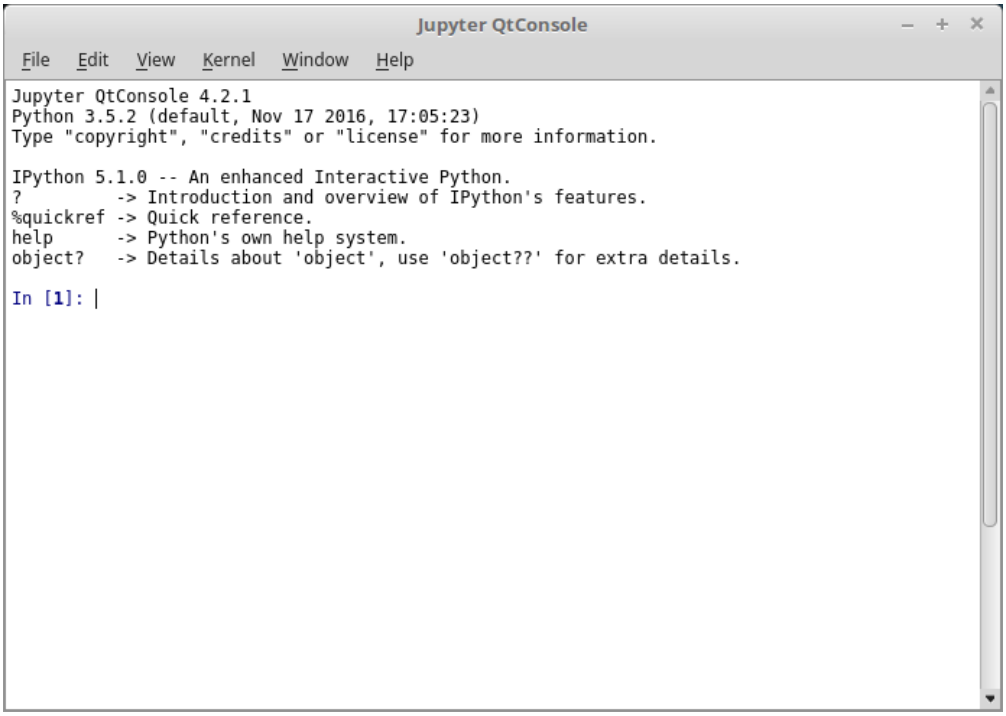
Se você digitar no prompt do sistema:

$ ipython3 qtconsole

ou

$ jupyter-qtconsole

Em vez de abrir em um terminal, IPython irá iniciar um console gráfico que à primeira vista aparece como um terminal, mas que é, de fato, muito mais capaz do que um terminal somente de texto. Este é um terminal especializado projetado para o trabalho científico interativo, e suporta a edição de multi-linha cheia com realce de cor e calltips gráficos para funções, pode manter múltiplas sessões de IPython abertas simultaneamente em abas, e quando scripts funcionam pode exibir os números diretamente Na área de trabalho.



qtconsole

4 Jupyter Notebook

Ao longo do tempo, o projeto IPython cresceu para incluir vários componentes, incluindo:

- Um shell interativo
- Um protocolo REPL
- Um documento de caderno fromat
- Uma ferramenta de conversão de documentos do notebook
- Uma ferramenta de criação de notebooks baseada na web
- Ferramentas para a construção de interface interativa (widgets)
- Python paralelo interativo

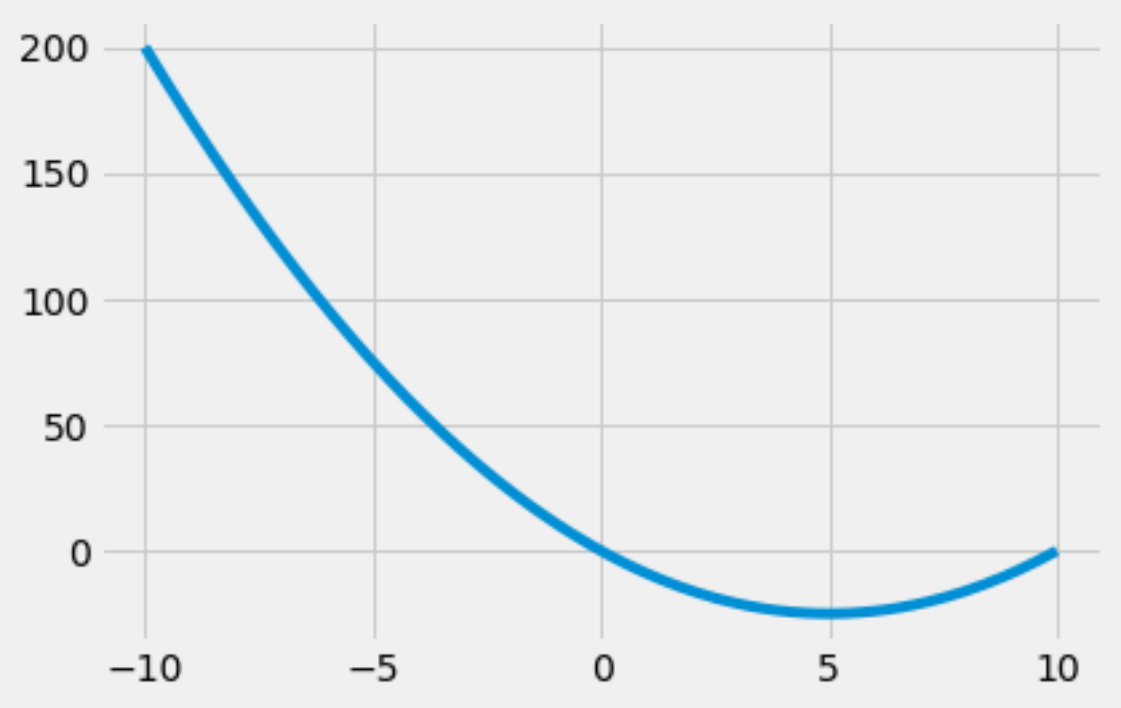
Como cada componente evoluiu, vários tinham crescido ao ponto que eles warrented projetos de seus próprios. Por exemplo, peças como o notebook e o protocolo não são nem mesmo específicas do Python. Como resultado, a equipe do IPython criou o Projeto Jupyter, que é a nova casa

de projetos agnósticos de linguagem que começou como parte do IPython, como o notebook no qual você está lendo este texto.

O notebook HTML que faz parte do projeto Jupyter suporta ** visualização de dados interativos ** e fácil de alto desempenho ** computação paralela **.

In [23]:
```
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')

def f(x):
    return x**2-10*x
#(x-3)*(x-5)*(x-7)+85

import numpy as np
x0 = -10         # origem
xf = 10          # fim do intervalo
n  = 200         # número de pontos que se discretiza o intervalo (x0,xf)
x = np.linspace(x0, xf, n)
y = f(x)
plt.plot(x,y)
```

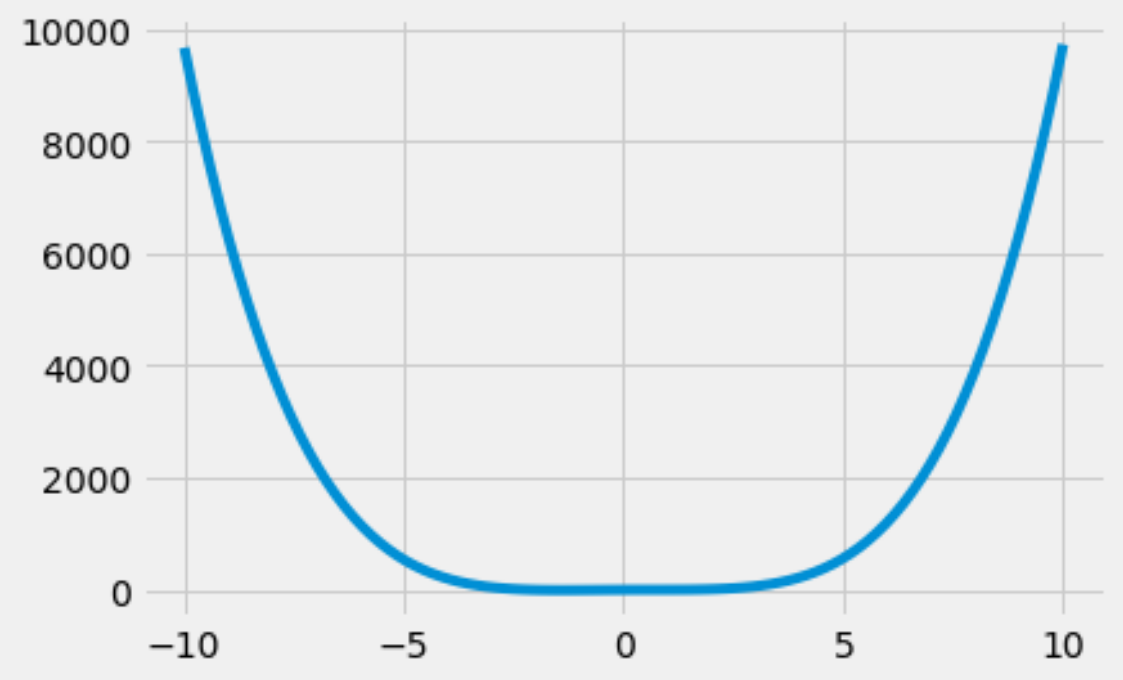
Out[23]: [<matplotlib.lines.Line2D at 0x7f255c18e550>]



In [24]:
```
def g(x):
    return x**4 -3*x**2 +4*x -x +2
```

```
x = np.linspace(-10, 10, 100)
y = g(x)
plt.plot(x,y)
```

```
Out[24]: [<matplotlib.lines.Line2D at 0x7f255a0f7860>]
```



O notebook permite documentar seu fluxo de trabalho usando HTML ou Markdown.
O Jupyter Notebook consiste em dois componentes relacionados:

- Um formato de documento JSON baseado em Notebook para gravação e distribuição de código Python e texto rico.
- subb1
- sub2
- Uma interface de usuário baseada na web para a criação e execução de documentos do notebook.

O Notebook pode ser usado iniciando o servidor Notebook com o comando:

$ ipython3 notebook

Isso inicia um **mecanismo IPython**, que é uma instância Python que leva comandos Python através de uma conexão de rede.

O **controlador IPython** fornece uma interface para trabalhar com um conjunto de mecanismos, aos quais um ou mais **clientes IPython** podem se conectar.

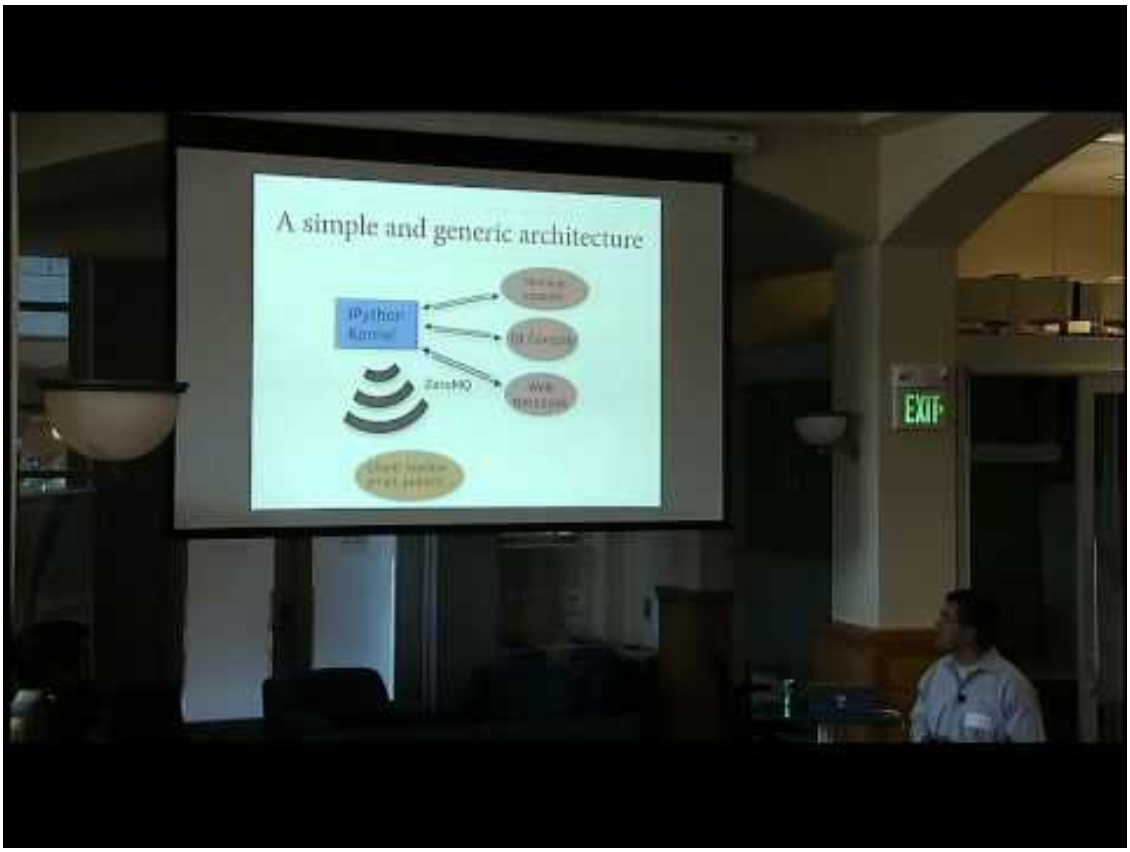
O Notebook dá-lhe tudo o que um navegador lhe dá. Por exemplo, você pode incorporar imagens, vídeos ou sites inteiros.

```
In [25]: from IPython.display import HTML
         HTML("<iframe src=http://jupyter.org/ width=700 height=350></iframe>")
```

```
Out[25]: <IPython.core.display.HTML object>
```

```
In [26]: from IPython.display import YouTubeVideo
         YouTubeVideo("rl5DaFbLc60")
```

Out[26]:

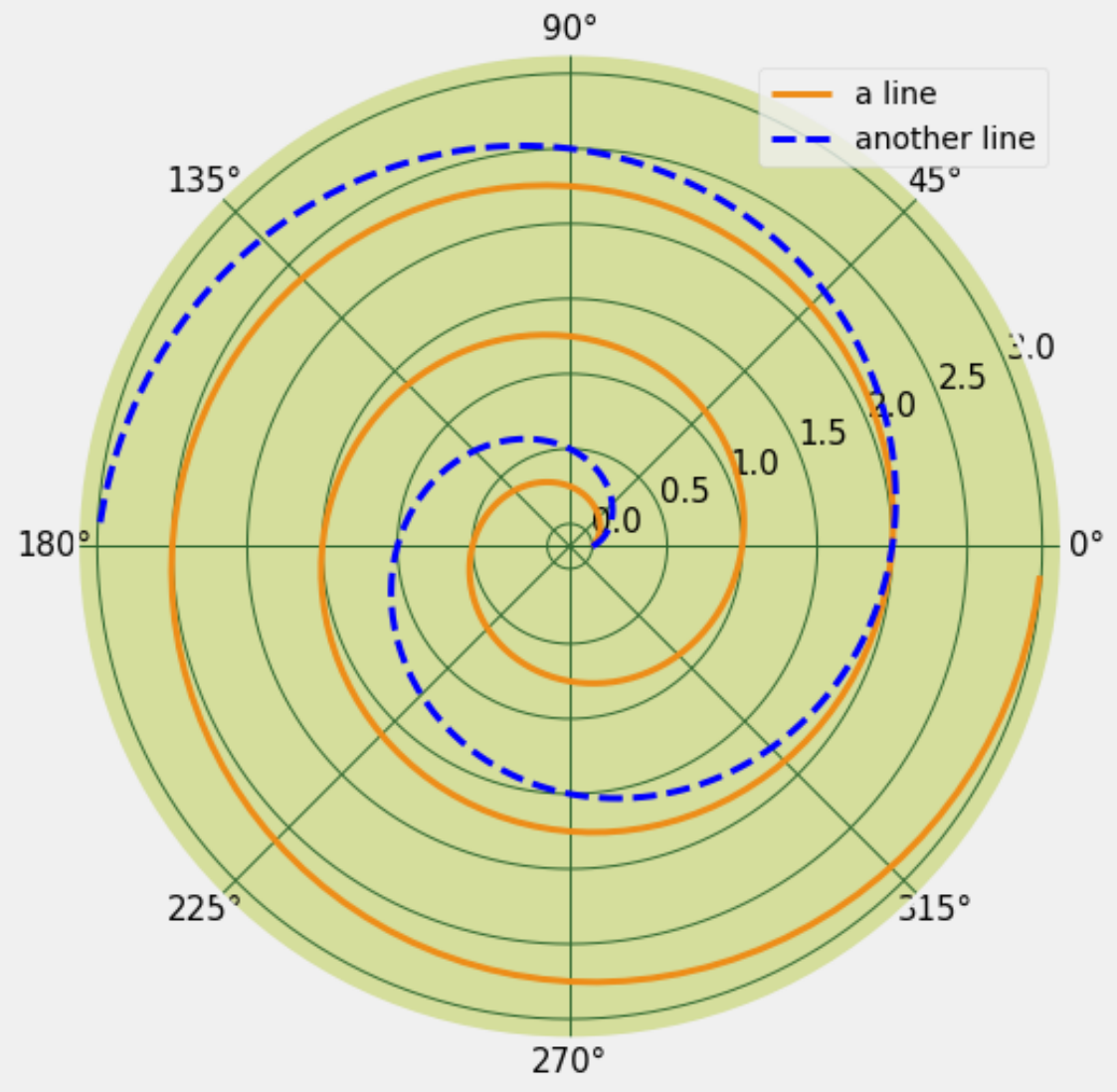


4.0.1 Código Remoto

Use `%load` para adicionar um código remoto

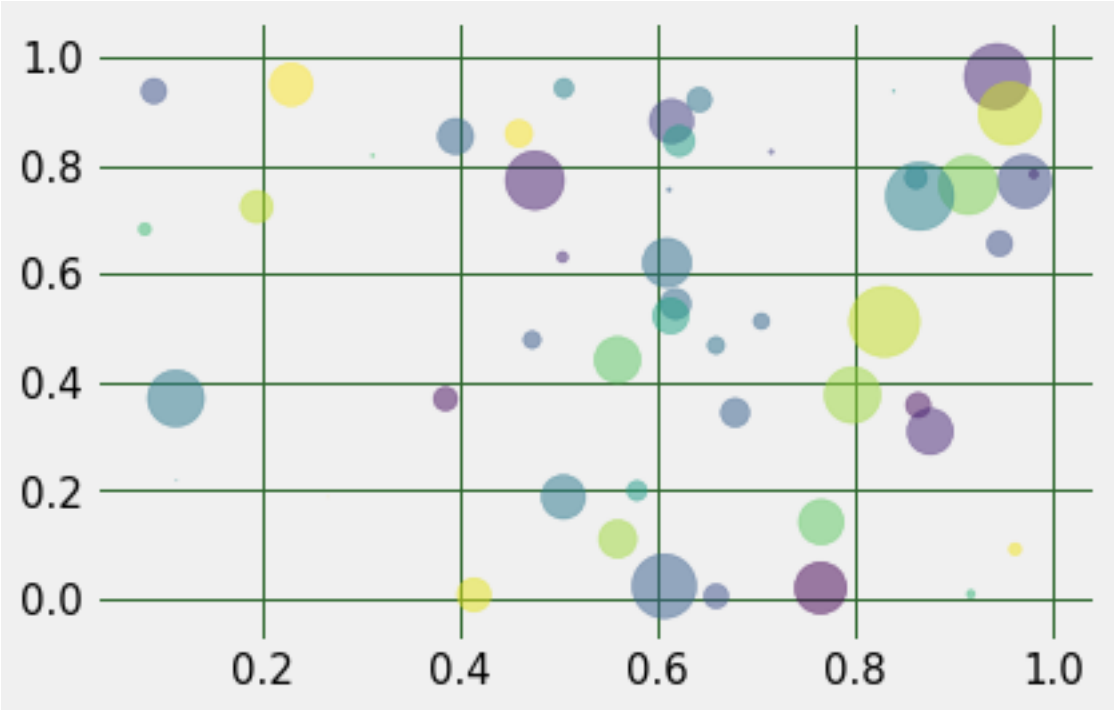
```
In [27]: # %load http://matplotlib.org/_downloads/polar_legend.py
         """
         ============
         Polar Legend
         ============

         Demo of a legend on a polar-axis plot.
         """
```

```
import numpy as np
from matplotlib.pyplot import figure, show, rc

# radar green, solid grid lines
rc('grid', color='#316931', linewidth=1, linestyle='-')
rc('xtick', labelsize=15)
rc('ytick', labelsize=15)

# force square figure and square axes looks better for polar, IMO
fig = figure(figsize=(8, 8))
ax = fig.add_axes([0.1, 0.1, 0.8, 0.8],
                  projection='polar', facecolor='#d5de9c')

r = np.arange(0, 3.0, 0.01)
theta = 2 * np.pi * r
ax.plot(theta, r, color='#ee8d18', lw=3, label='a line')
ax.plot(0.5 * theta, r, color='blue', ls='--', lw=3, label='another line')
ax.legend()

show()
```



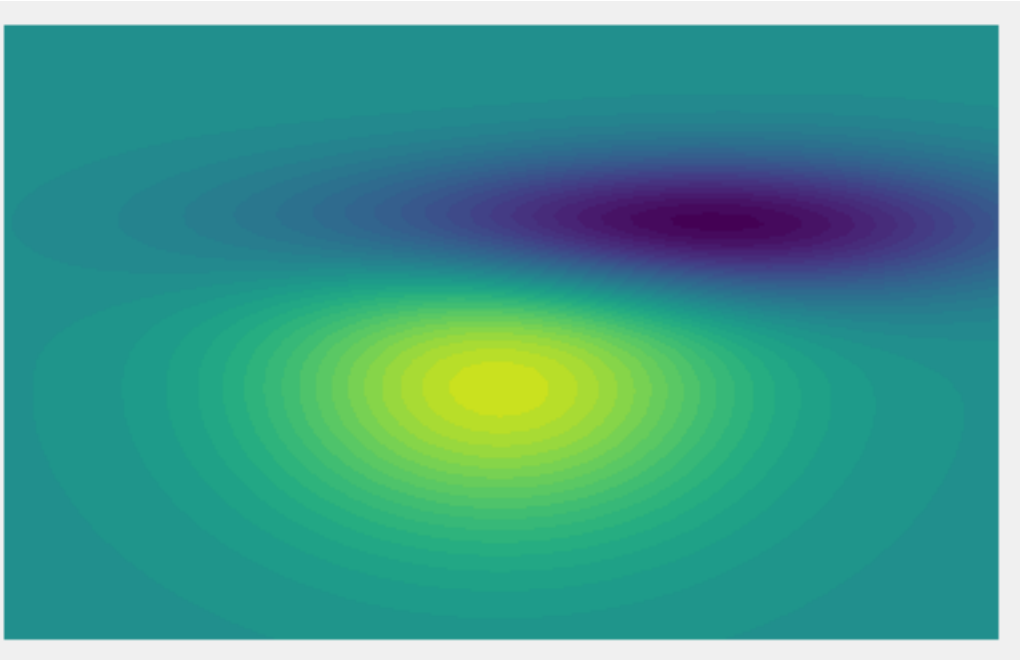
In [28]:
```
# %load http://matplotlib.org/mpl_examples/shapes_and_collections/scatter_demo.py
"""
Simple demo of a scatter plot.
"""
import numpy as np
import matplotlib.pyplot as plt


N = 50
x = np.random.rand(N)
y = np.random.rand(N)
colors = np.random.rand(N)
area = np.pi * (15 * np.random.rand(N))**2  # 0 to 15 point radii
```

```
plt.scatter(x, y, s=area, c=colors, alpha=0.5)
plt.show()
```



In [29]:
```
# %load http://matplotlib.org/examples/frontpage/plot_contour.py
"""
==========================
Frontpage contour example
==========================

This example reproduces the frontpage contour example.
"""

import matplotlib.pyplot as plt
import numpy as np
from matplotlib import mlab, cm

extent = (-3, 3, -3, 3)

delta = 0.1
x = np.arange(-3.0, 4.001, delta)
y = np.arange(-4.0, 3.001, delta)
X, Y = np.meshgrid(x, y)
Z1 = mlab.bivariate_normal(X, Y, 1.0, 1.0, 0.0, -0.5)
Z2 = mlab.bivariate_normal(X, Y, 1.5, 0.5, 1, 1)
Z = (Z1 - Z2) * 10
```

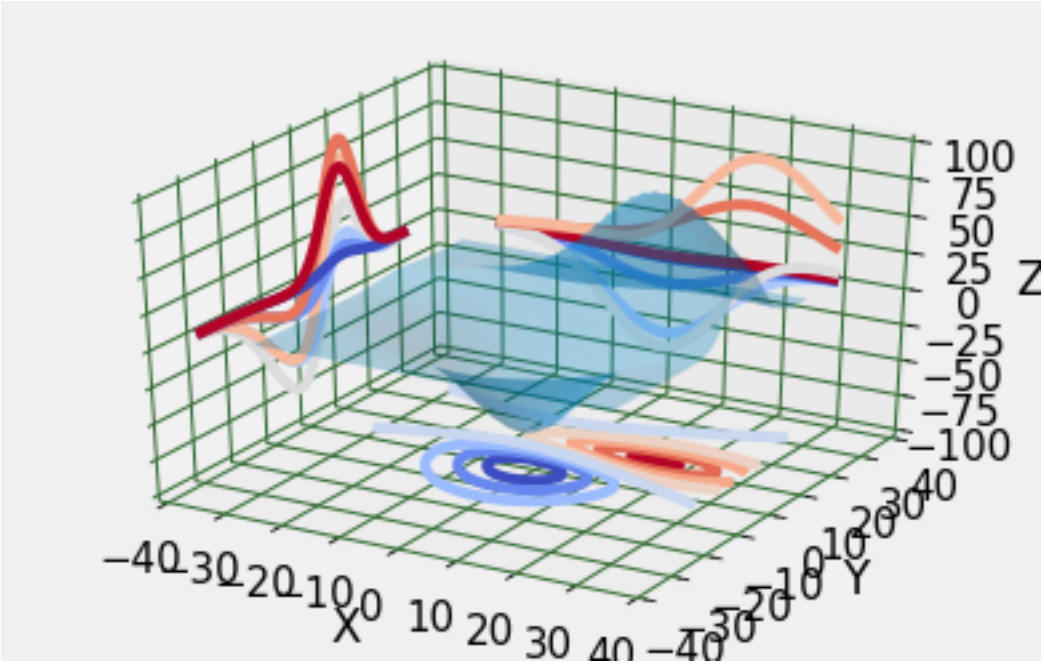
```
levels = np.linspace(-2.0, 1.601, 40)
norm = cm.colors.Normalize(vmax=abs(Z).max(), vmin=-abs(Z).max())

fig, ax = plt.subplots()
cset1 = ax.contourf(
    X, Y, Z, levels,
    norm=norm)
ax.set_xlim(-3, 3)
ax.set_ylim(-3, 3)
ax.set_xticks([])
ax.set_yticks([])
fig.savefig("contour_frontpage.png", dpi=300)  # results in 160x120 px image
```



```
In [ ]: %matplotlib qt5
```

```
In [30]: # %load http://matplotlib.org/mpl_examples/mplot3d/contour3d_demo3.py
         # Para fazer o gráfico em uma janela habilite:
         # %matplotlib qt
         from mpl_toolkits.mplot3d import axes3d
         import matplotlib.pyplot as plt
         from matplotlib import cm

         fig = plt.figure()
         ax = fig.gca(projection='3d')
         X, Y, Z = axes3d.get_test_data(0.02)
```

```
ax.plot_surface(X, Y, Z, rstride=10, cstride=10, alpha=0.3)
cset = ax.contour(X, Y, Z, zdir='z', offset=-100, cmap=cm.coolwarm)
cset = ax.contour(X, Y, Z, zdir='x', offset=-40, cmap=cm.coolwarm)
cset = ax.contour(X, Y, Z, zdir='y', offset=40, cmap=cm.coolwarm)

ax.set_xlabel('X')
ax.set_xlim(-40, 40)
ax.set_ylabel('Y')
ax.set_ylim(-40, 40)
ax.set_zlabel('Z')
ax.set_zlim(-100, 100)

plt.show()
```



4.0.2 Suporte ao Mathjax

Mathjax é uma implementação javascript α do LaTeX que permite que as equações sejam incorporadas em HTML. Por exemplo, esta marcação:

```
"""$$ \int_{a}^{b} f(x)\, dx \approx \frac{1}{2} \sum_{k=1}^{N} \left( x_{k} - x_{k-1} \right) \
```

torna-se:

$$\int_a^b f(x)\,dx \approx \frac{1}{2}\sum_{k=1}^{N} \left(x_k - x_{k-1}\right)\left(f(x_k) + f(x_{k-1})\right).$$

4.1 Suporte SymPy

SymPy é uma biblioteca Python para cálculo simbólico mathematics. Ela suporta:

- polinômios
- cálculo
- resolvendo equações
- matemática discreta
- matrizes

```
In [31]: from sympy import *
         init_printing()
         x, y = symbols("x y")
```

```
In [32]: eq = ((x+y)**3 * (x+1))
         eq
```

Out[32]:

$$(x+1)\,(x+y)^3$$

```
In [33]: expand(eq)
```

Out[33]:

$$x^4 + 3x^3y + x^3 + 3x^2y^2 + 3x^2y + xy^3 + 3xy^2 + y^3$$

```
In [34]: (1/cos(x)).series(x, 0, 12)
```

Out[34]:

$$1 + \frac{x^2}{2} + \frac{5x^4}{24} + \frac{61x^6}{720} + \frac{277x^8}{8064} + \frac{50521x^{10}}{3628800} + \mathcal{O}\left(x^{12}\right)$$

```
In [35]: limit((sin(x)-x)/x**3, x, 0)
```

Out[35]:

$$-\frac{1}{6}$$

```
In [36]: diff(cos(x**2)**2 / (1+x), x)
```

Out[36]:

$$-\frac{4x\cos\left(x^2\right)}{x+1}\sin\left(x^2\right) - \frac{\cos^2\left(x^2\right)}{(x+1)^2}$$

```
In [37]: integrate(x/sqrt(x**2 +y**2),x)
```

Out[37]:

$$\sqrt{x^2+y^2}$$

4.1.1 Funções mágicas

O IPython tem um conjunto de "funções mágicas"predefinidas que você pode chamar com uma sintaxe de estilo de linha de comando. Esses incluem:

- %run
- %edit
- %debug
- %timeit
- %paste
- %load_ext

```
In [ ]: %lsmagic
```

4.2 Exemplo: O tutor

Para executar o código abaixo você precisa de uuma conexão com a internet, pois ele irá executar o comando abaixo no site Python Tutor. Esse site é muito bom para aqueles que estão aprendendo uma linguagem de programação, pois ele linha por linha o que ocorrer na execução de um código. Usando essa ferramenta é possível visualizar códigos escritos nas seguintes linguagens: Python2, Python3, Java, JavaScript, TypeScript, Ruby, C, and C++.

Como exemplo, considere o seguinte código:

```
In [38]: %%tutor
         inicio=4
         fim= 13 # ele sempre vai terminar um passo antes, ou seja, em 12
         passo = 2
         for i in range(inicio,fim,passo):
             print(i)
```

```
<IPython.lib.display.IFrame at 0x7f2553f60e10>
```

Determinando o tempo de execução do código; O comando magic `timeit` existe tanto na forma de linha como de célula:

```
In [39]: import numpy as np
         %timeit np.linalg.eigvals(np.random.rand(100,100))
```

```
13.8 ms ± 1.3 ms per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

```
In [40]: %%timeit a = np.random.rand(100, 100)
         np.linalg.eigvals(a)
```

```
12.5 ms ± 149 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

O IPython também cria *aliases* para alguns intérpretes comuns, como bash, ruby, perl, etc. Estes são todos equivalentes a %%script <nome>

```
In [41]: %%ruby
         puts "Hello from Ruby #{RUBY_VERSION}"

Hello from Ruby 2.3.1


In [42]: %%bash
         echo "hello from $BASH"

hello from /bin/bash
```

IPython tem uma extensão `rmagic` que contém algumas funções mágicas para trabalhar com R via rpy2. Esta extensão pode ser carregada usando a magia `%load_ext` da seguinte forma:

```
In [ ]: %load_ext rpy2.ipython
```

Se o comando anterior gerar um erro, é provável que você não tenha o módulo `rpy2` instalado. Você pode instalar isso agora, em uma nova célula, usando o seguinte código:

```
!sudo apt install python3-rpy2
```

Caso não encontre o pacote, vocẽ pode instalar ele usando o seguinte código:

```
!sudo -H pip3 install rpy2
```

```
In [43]: %load_ext watermark

In [44]: %watermark

2017-11-22T07:52:40-02:00

CPython 3.5.2
IPython 6.2.1

compiler   : GCC 5.4.0 20160609
system     : Linux
release    : 4.10.0-40-generic
machine    : x86_64
processor  : x86_64
CPU cores  : 4
interpreter: 64bit


In [45]: %watermark -t -n -u -z

last updated: Wed Nov 22 2017 07:52:41 -02


In [46]: %watermark -v -m -p numpy,matplotlib,scipy -g
```

```
CPython 3.5.2
IPython 6.2.1

numpy 1.13.3
matplotlib 2.1.0
scipy 1.0.0

compiler   : GCC 5.4.0 20160609
system     : Linux
release    : 4.10.0-40-generic
machine    : x86_64
processor  : x86_64
CPU cores  : 4
interpreter: 64bit
Git hash   :
```

```
In [ ]: %watermark?
```

O comando acima retorna a saída:

```
 %watermark [-a AUTHOR] [-d] [-n] [-t] [-i] [-z] [-u] [-c CUSTOM_TIME]
                 [-v] [-p PACKAGES] [-h] [-m] [-g] [-r] [-w] [-iv]

IPython magic function to print date/time stamps
and various system information.

optional arguments:
  -a AUTHOR, --author AUTHOR
                        prints author name
  -d, --date            prints current date as YYYY-mm-dd
  -n, --datename        prints date with abbrv. day and month names
  -t, --time            prints current time as HH-MM-SS
  -i, --iso8601         prints the combined date and time including the time
                        zone in the ISO 8601 standard with UTC offset
  -z, --timezone        appends the local time zone
  -u, --updated         appends a string "Last updated: "
  -c CUSTOM_TIME, --custom_time CUSTOM_TIME
                        prints a valid strftime() string
  -v, --python          prints Python and IPython version
  -p PACKAGES, --packages PACKAGES
                        prints versions of specified Python modules and
                        packages
  -h, --hostname        prints the host name
  -m, --machine         prints system and machine info
  -g, --githash         prints current Git commit hash
  -r, --gitrepo         prints current Git remote address
  -w, --watermark       prints the current version of watermark
```

```
  -iv, --iversions      prints the name/version of all imported modules
File:      /usr/local/lib/python3.5/dist-packages/watermark/watermark.py
```

```
In [ ]: x,y = np.arange(10), np.random.normal(size=10)
        %R print(lm(rnorm(10)~rnorm(10)))
```

Exemplo de uso do R

```
In [ ]: %%R -i x,y -o XYcoef
        lm.fit <- lm(y~x)
        par(mfrow=c(2,2))
        print(summary(lm.fit))
        plot(lm.fit)
        XYcoef <- coef(lm.fit)
```

```
In [ ]: XYcoef
```

4.2.1 LaTeX

Além do suporte MathJax, você pode declarar uma célula LaTeX usando o comando magic `%latex`:

```
In [47]: %%latex
         \begin{align}
         \nabla \times \vec{\mathbf{B}} -\, \frac1c\, \frac{\partial\vec{\mathbf{E}}}{\partial t
         \nabla \cdot \vec{\mathbf{E}} & = 4 \pi \rho \\
         \nabla \times \vec{\mathbf{E}}\, +\, \frac1c\, \frac{\partial\vec{\mathbf{B}}}{\partial
         \nabla \cdot \vec{\mathbf{B}} & = 0
         \end{align}
```

$$\nabla \times \vec{\mathbf{B}} - \frac{1}{c}\frac{\partial \vec{\mathbf{E}}}{\partial t} = \frac{4\pi}{c}\vec{\mathbf{j}} \tag{1}$$

$$\nabla \cdot \vec{\mathbf{E}} = 4\pi\rho \tag{2}$$

$$\nabla \times \vec{\mathbf{E}} + \frac{1}{c}\frac{\partial \vec{\mathbf{B}}}{\partial t} = \vec{\mathbf{0}} \tag{3}$$

$$\nabla \cdot \vec{\mathbf{B}} = 0 \tag{4}$$

4.2.2 A biblioteca matplotlib2tikz

A biblioteca matplotlib2tikz do Python, é uma ferramenta a qual irá converter as figuras feitas pela biblioteca Section **??** em figura PGFPlots (TikZ).

This is matplotlib2tikz, a Python tool for converting matplotlib figures into Section **??** (Section **??**).

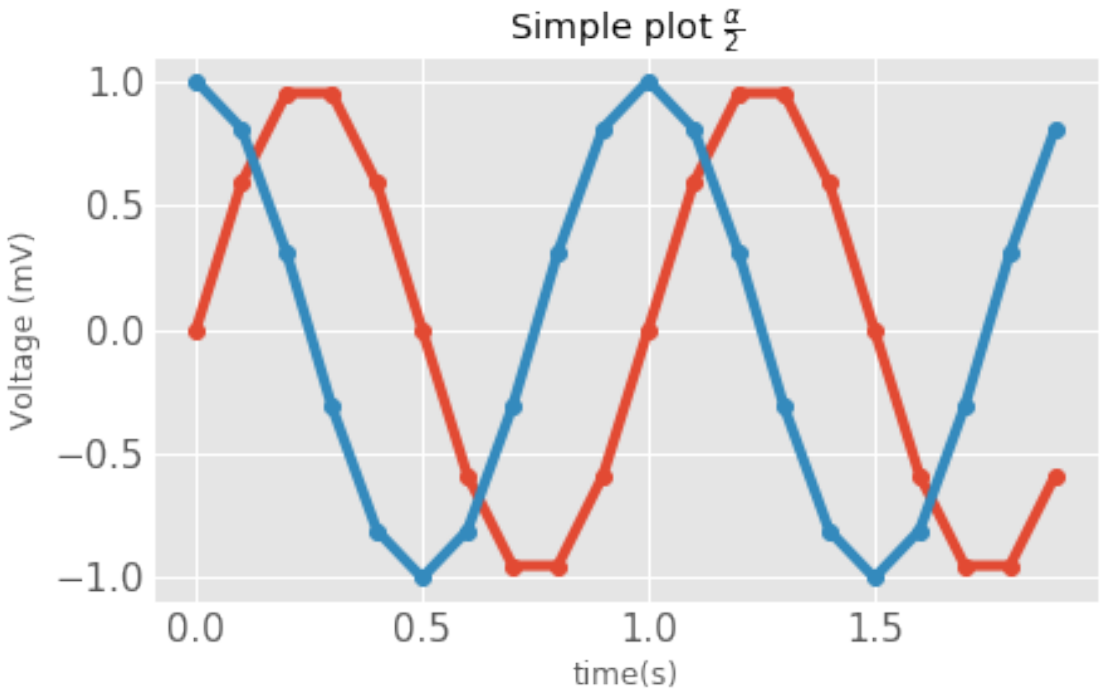
Para instalar a biblioteca faça o seguinte:

```
In [ ]: !sudo -H pip3.5 install --upgrade matplotlib2tikz
```

```
In [48]: %matplotlib inline
```

In [49]:
```
#Agora vamos testar com o seguinte código
from matplotlib import pyplot as plt
from matplotlib import style
import numpy as np
fig = plt.figure()
style.use('ggplot')
t = np.arange(0.0, 2.0, 0.1)
s = np.sin(2*np.pi*t)
s2 = np.cos(2*np.pi*t)
plt.plot(t, s, 'o-', lw=4.1)
plt.plot(t, s2, 'o-', lw=4.1)
plt.xlabel('time(s)')
plt.ylabel('Voltage (mV)')
plt.title('Simple plot $\\frac{\\alpha}{2}$')
plt.grid(True)

from matplotlib2tikz import save as tikz_save
tikz_save('test.pgf')
plt.show()
```

```
Upgrade to   matplotlib2tikz 0.6.14    available! (installed: 0.6.13)

To upgrade matplotlib2tikz with pip, type

   pip install -U matplotlib2tikz

To upgrade _all_ pip-installed packages, use

   pipdate/pipdate3

To disable these checks, set SecondsBetweenChecks in /home/salviano/.config/pipdate/config.ini t
============================================================
Please add the following lines to your LaTeX preamble:

\usepackage[utf8]{inputenc}
\usepackage{fontspec} % This line only for XeLaTeX and LuaLaTeX
\usepackage{pgfplots}
============================================================
Horizontal alignment will be ignored as no 'x tick label text width' has been passed in the 'ext
Horizontal alignment will be ignored as no 'y tick label text width' has been passed in the 'ext
```

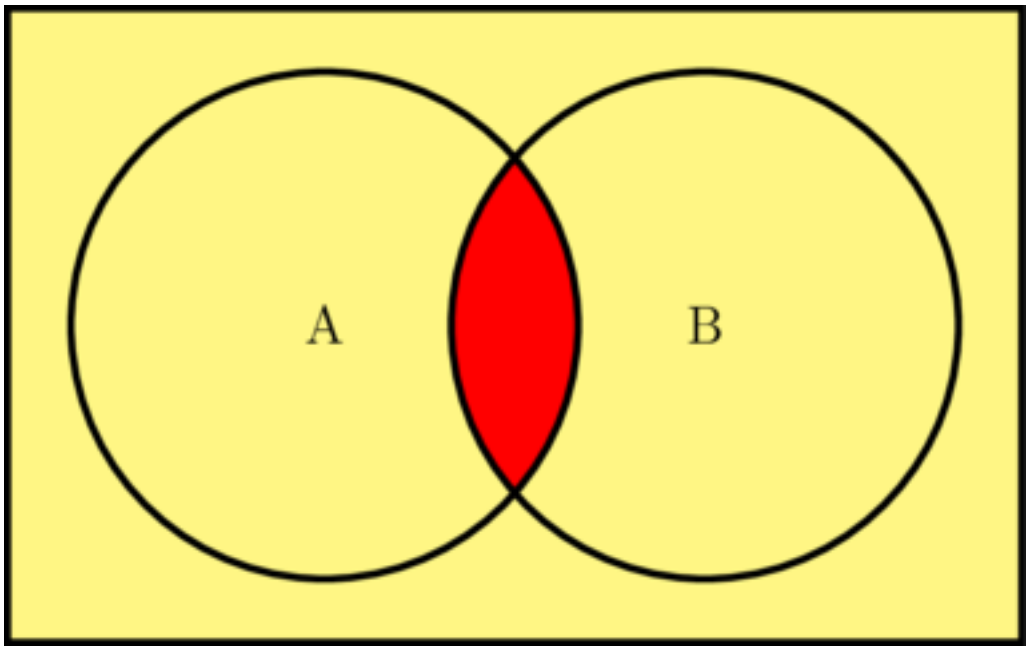



```
In [50]: !cat test.pgf
```

```
% This file was created by matplotlib2tikz v0.6.13.
\begin{tikzpicture}

\definecolor{color1}{rgb}{0.203921568627451,0.541176470588235,0.741176470588235}
\definecolor{color0}{rgb}{0.886274509803922,0.290196078431373,0.2}

\begin{axis}[
title={Simple plot $\frac{\alpha}{2}$},
xlabel={time(s)},
ylabel={Voltage (mV)},
xmin=-0.095, xmax=1.995,
ymin=-1.1, ymax=1.1,
tick align=outside,
tick pos=left,
xmajorgrids,
x grid style={white},
ymajorgrids,
y grid style={white},
axis line style={white},
axis background/.style={fill=white!89.803921568627459!black}
]
\addplot [line width=1.64pt, color0, mark=*, mark size=3, mark options={solid}, forget plot]
```

```
table {%
0 0
0.1 0.587785252292473
0.2 0.951056516295154
0.3 0.951056516295154
0.4 0.587785252292473
0.5 1.22464679914735e-16
0.6 -0.587785252292473
0.7 -0.951056516295154
0.8 -0.951056516295154
0.9 -0.587785252292473
1 -2.44929359829471e-16
1.1 0.587785252292474
1.2 0.951056516295154
1.3 0.951056516295154
1.4 0.587785252292473
1.5 3.67394039744206e-16
1.6 -0.587785252292473
1.7 -0.951056516295154
1.8 -0.951056516295154
1.9 -0.587785252292473
};
\addplot [line width=1.64pt, color1, mark=*, mark size=3, mark options={solid}, forget plot]
table {%
0 1
0.1 0.809016994374947
0.2 0.309016994374947
0.3 -0.309016994374948
0.4 -0.809016994374947
0.5 -1
0.6 -0.809016994374947
0.7 -0.309016994374948
0.8 0.309016994374947
0.9 0.809016994374947
1 1
1.1 0.809016994374947
1.2 0.309016994374947
1.3 -0.309016994374947
1.4 -0.809016994374947
1.5 -1
1.6 -0.809016994374948
1.7 -0.309016994374946
1.8 0.309016994374947
1.9 0.809016994374947
};
\end{axis}

\end{tikzpicture}
```

```
In [51]: %load_ext tikzmagic

In [52]: %%tikz
         \draw[ultra thick,fill=yellow!60] (1,1) rectangle (9,6);
         \begin{scope}
           \clip (6.5,3.5) circle (2);
           \fill[red,ultra thick] (3.5,3.5) circle (2);
         \end{scope}
         \draw[ultra thick] (3.5,3.5) circle (2) node{\large A};
         \draw[ultra thick] (6.5,3.5) circle (2) node{\large B};
```



Uso Gere seu gráfico com a matplotlib como sempre. Agora ao invez de `pyplot.show()`, deve-se chamar a `matplotlib2tikz` da seguinte forma:

```
tikz_save('meu_tikz.pgf')
```

para a criar um arquivo TikZ com o nome de "meu_tikz.pgf". Para isso carregue a biblioteca com:

```
from matplotlib2tikz import save as tikz_save
```

Opcional Os scripts aceitam várias opções, por exemplo `height`, `width`, `encoding`, e algumas outras. Chame elas da seguinte forma:

```
tikz_save('meu_tikz.pgf', figureheight='4cm', figurewidth='6cm')
```

Observe que altura e largura devem ser definidas grandes o suficiente; Configurá-lo muito baixo pode resultar em uma falha de compilação do LaTeX ao longo das linhas de dimensões muito grande ou um `Arithmetic Overflow`; Consulte informações sobre esses erros no manual PGFPlots. Para especificar a dimensão do gráfico a partir do documento LaTeX, tente

```
tikz_save(
    'meu_tikz.pgf',
    figureheight = '\\figureheight',
    figurewidth = '\\figurewidth'
    )
```

e no código fonte LaTeX escreva

```
\newlength\figureheight
\newlength\figurewidth
\setlength\figureheight{4cm}
\setlength\figurewidth{6cm}
\input{mytikz.tex}
```

Adicione o conteúdo do arquivo "meu_tikz.pgf"em seu código-fonte LaTeX; uma maneira conveniente de fazê-lo é através de `\input{/path/to/meu_tikz.pgf}`. Também certifique-se de que no cabeçalho do documento os pacotes para PGFPlots e suporte a Unicode adequada e estão incluídos:

```
\usepackage[utf8]{inputenc}
\usepackage{pgfplots}
```

Adicionalmente, com o LuaLaTeX

```
\usepackage{fontspec}
```

será necessário para compor os caracteres Unicode. Opcionalmente, para usar os recursos PGFPlots mais recentes,

```
\pgfplotsset{compat=newest}
```

4.3 Javscript

O Jupyter também permite que os objetos declarem uma representação JavaScript. No início, isso pode parecer estranho como saída é inerentemente visual e JavaScript é uma linguagem de programação. No entanto, isso abre a porta para a saída rica que aproveita toda a potência do JavaScript e bibliotecas associadas, como D3 para a saída.

```
In [53]: %%javascript

         alert("Hello world!");
```

```
<IPython.core.display.Javascript object>
```

5 Instalando o kernel do Fortran

Para instalar o kernel do fortran siga os passos a seguir:

5.0.1 Instalação Manual

Certifique-se que o seu sistemas tem o seguintes programas instalados:

- gfortran
- jupyter
- python 3
- pip3

5.0.2 Instalação passo a passo

Siga as instruções na sequência a seguir:

```
> git clone https://github.com/ZedThree/jupyter-fortran-kernel.git
> sudo -H pip3 install jupyter-fortran-kernel
> cd jupyter-fortran-kernel
> sudo -H jupyter-kernelspec install fortran_spec/
> jupyter-notebook.
```

Agora é só testar. Reinicie o jupyter notebook

6 Instalando o kernel do C

Para instalar o kernel do C siga os passos a seguir:

6.0.1 Instalação Manual

Certifique-se que o seu sistemas tem o seguintes programas instalados:

- gcc
- jupyter
- python 3
- pip3

6.0.2 Instalação passo a passo

Siga as instruções na sequência a seguir:

```
> git clone https://github.com/brendan-rius/jupyter-c-kernel.git
> sudo -H pip3 install -U jupyter-c-kernel
> cd jupyter-c-kernel
> sudo -H jupyter-kernelspec install jupyter_c_kernel/
> sudo mv resources /usr/local/lib/python3.5/dist-packages/.
> sudo chown -cR root.root /usr/local/lib/python3.5/dist-packages/resources
> jupyter-notebook.
```

Agora é só testar. Reinicie o jupyter notebook

```
In [ ]: # Vamos instalar o kernel do C Manualmente
        !sudo -H pip3 install jupyter-c-kernel
```

É necessário informar ao python onde está o kernel e para isso faremos

```
In [ ]: ! git clone https://github.com/brendan-rius/jupyter-c-kernel.git
```

```
In [ ]: ! sudo pip3.5 install jupyter-c-kernel/
```

```
In [ ]: %%bash
        sudo cat <<FIM > jupyter-c-kernel/jupyter_c_kernel/kernel.json
        {
          "argv": [
            "python3",
            "-m",
            "jupyter_c_kernel",
            "-f",
            "{connection_file}"
          ],
          "display_name": "C",
          "language": "c"
        }
        FIM
```

```
In [ ]: ! cd jupyter-c-kernel && sudo -H jupyter-kernelspec install jupyter_c_kernel/
```

```
In [ ]: %%bash
        if [ -d /usr/local/lib/python3.5/dist-packages/ ]
        then
           echo "O diretório já existe e foi movido o seu conteúdo"
           sudo mv -f jupyter-c-kernel/resources/* /usr/local/lib/python3.5/dist-packages/resour
        else
           echo "O diretório foi movido com o seu conteúdo"
           sudo mv -f resources /usr/local/lib/python3.5/dist-packages/.
        fi
```

```
In [ ]: !sudo chown -cR root.root /usr/local/lib/python3.5/dist-packages/resources
```

```
In [ ]: # Se tudo ocorreu bem você já pode remover a pasta do jupyte-c-kernel
        ! rm -rf jupyter-c-kernel
```

O kernel deve estar instalado no diretório: /usr/local/share/jupyter/kernels/c_spec

```
In [ ]: # Para verificar liste o diretório
        ! ls -alh /usr/local/share/jupyter/kernels/
```

```
In [ ]: # Para verificar se o arquivo foi criado e o seu conteúdo
        ! cat /usr/local/share/jupyter/kernels/jupyter_c_kernel/kernel.json
```

7 Instalando os kernels do Gnuplot e do Scilab

Ele é um kernel do Jupyter/IPython para o Gnuplot e para o Scilab.

7.0.1 Instalação Manual

Certifique-se que o seu sistemas tem o seguintes programas instalados:

- gnuplot
- scilab
- jupyter
- python 3
- pip3
- metakernel

7.0.2 Instalação passo a passo

```
sudo -H pip3 install --upgrade metakernel
sudo -H pip3 install --upgrade gnuplot_kernel
sudo -H pip3 install --upgrade scilab_kernel
```

Será necessário gera as especificações do metakernel

```
> mkdir metakernel_spec
```

crie dentro desse diretório o arquivo "kernel.json"com o seguinte conteúdo:

```
{
  "argv": [
    "python3",
    "-m",
    "metakernel_kernel",
    "-f",
    "{connection_file}"
  ],
  "display_name": "Metakernel",
  "language": "metakernel"
}
```

Será necessário gera as especificações do gnuplot_kernel e para isso faça

```
> mkdir metakernel_spec
> mkdir gnuplot_spec
```

crie dentro desse diretório o arquivo "kernel.json"com o seguinte conteúdo:

```
{
  "argv": [
    "python3",
    "-m",
```

```
    "gnuplot_kernel",
    "-f",
    "{connection_file}"
  ],
  "display_name": "Gnuplot",
  "language": "gnuplot"
}
```

Agora ative os dois kernels fazendo:

```
> sudo -H jupyter-kernelspec install metakernel_spec/
> sudo -H jupyter-kernelspec install gnuplot_spec/
> jupyter-notebook.
```

O metakernel exige que se adicione o seguinte arquivo

```
# /etc/ipython/ipython_config.py
c = get_config()
startup = [
   'from metakernel import register_ipython_magics',
   'register_ipython_magics()',
]
c.InteractiveShellApp.exec_lines = startup
```

Agora é só testar. Reinicie o jupyter notebook

Para instalar execute as células abaixo Para realizar a instalação do kernel do gnuplot e scilab é necessário instalar o metakernel. Antes vá ao menu Kernel --> Change Kernel e veja quais são os kernels que já se encontram instalados.

Portanto siga os passos das células abaixo:

```
In [ ]: !sudo -H pip3 install --upgrade metakernel

In [ ]: %%file ipython_config.py
        c = get_config()
        startup = [
           'from metakernel import register_ipython_magics',
           'register_ipython_magics()',
        ]
        c.InteractiveShellApp.exec_lines = startup

In [ ]: !sudo -p mkdir /etc/ipython

In [ ]: !sudo mv ipython_config.py /etc/ipython/.

In [ ]: !ls -alh /etc/ipython/

In [ ]: ! sudo chown -c root.root /etc/ipython/ipython_config.py

In [ ]: !sudo -H pip3 install --upgrade gnuplot_kernel
```

```
In [ ]: !sudo -H pip3 install --upgrade scilab_kernel

In [ ]: !mkdir -p metakernel_spec

In [ ]: %%file metakernel_spec/kernel.json
        {
          "argv": [
            "python3",
            "-m",
            "metakernel_kernel",
            "-f",
            "{connection_file}"
          ],
          "display_name": "Metakernel",
          "language": "metakernel"
        }

In [ ]: !sudo -H jupyter-kernelspec install metakernel_spec/

In [ ]: !rm -rf metakernel_spec

In [ ]: !mkdir -p gnuplot_spec

In [ ]: %%file gnuplot_spec/kernel.json
        {
          "argv": [
            "python3",
            "-m",
            "gnuplot_kernel",
            "-f",
            "{connection_file}"
          ],
          "display_name": "Gnuplot",
          "language": "gnuplot"
        }

In [ ]: !cat gnuplot_spec/kernel.json

In [ ]: !sudo -H jupyter-kernelspec install gnuplot_spec/

In [ ]: !rm -rf gnuplot_spec

In [ ]: !mkdir -p scilab_spec

In [ ]: %%file scilab_spec/kernel.json
        {
          "argv": [
            "python3",
            "-m",
            "scilab_kernel",
```

```
            "-f",
            "{connection_file}"
          ],
          "display_name": "Scilab",
          "language": "scilab"
        }
```

```
In [ ]: !sudo -H jupyter-kernelspec install scilab_spec/
```

```
In [ ]: !rm -rf scilab_spec
```

```
In [ ]: !sudo -H pip3 install --upgrade bash_kernel
```

```
In [ ]: !mkdir -p bash_spec
```

```
In [ ]: %%file bash_spec/kernel.joson
        {
          "argv": [
            "python3",
            "-m",
            "bash_kernel",
            "-f",
            "{connection_file}"
          ],
          "display_name": "Bash",
          "language": "bash"
        }
```

```
In [ ]: !sudo -H jupyter-kernelspec install bash_spec/
```

```
In [ ]: !rm -rf bash_spec
```

Agora é só testar. Reinicie o jupyter notebook e depois vá no menu Kernel --> Change Kernel e veja se aparece os kernels instalados? Se apareceu está tudo ok.

7.1 Kernel do Gnuplot

Ao iniciar um novo notebook vocẽ agora deve seelcionar o kernel a ser usado. No caso da selecção ter sido o do gnuplo, todas as células de código serão interpretadas como declarações para gnuplot a menos que a célula magic esteja sendo usada. Da mesma forma, cada linha é uma declaração para gnuplot a menos que seja uma linha magic. Vamos ver o que tudo isso significa.

Entretanto aqui o kernel que está sendo usado é do Python 3, conforme mostra o canto superior direito da linha que contém o menu deste notebook.

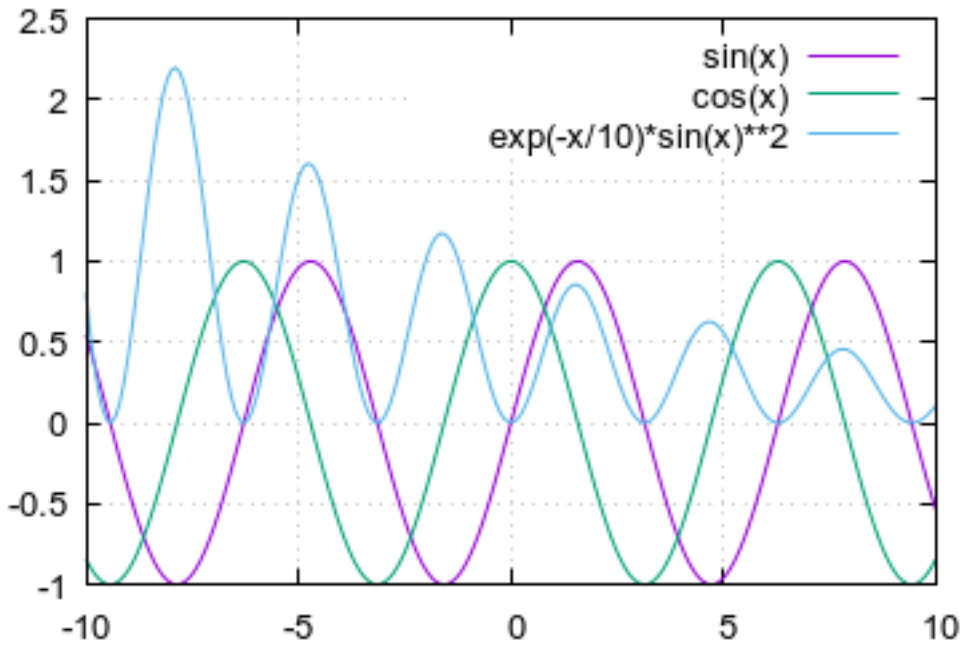
Pode-se fazer alguns gráficos usando o gnuplot, para isso deve-se usar as `magics` do notebook. Para carregar a extensão do gnuplot faça `%load_ext gnuplot_kernel`

```
In [54]: # Esse código as magics para o gnuplot
         %load_ext gnuplot_kernel
```

Agora coloque o seguinte código `%%gnuplot` no início de toda célula que contém um código do gnuplot.

Primeiro, um gráfico e um cálculo simples.

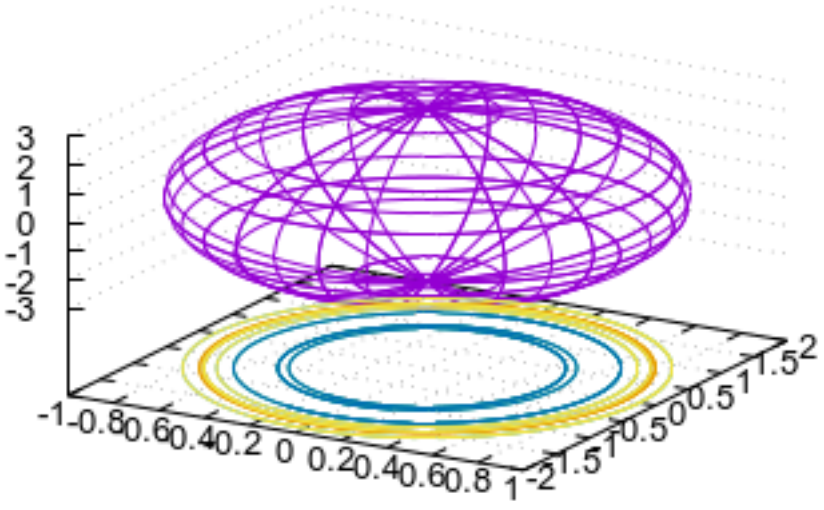
```
In [55]: %%gnuplot
         set grid
         set samples 200
         plot sin(x), cos(x),  exp(-x/10)*sin(x)**2
```




```
set grid
set samples 200
set output '/tmp/gnuplot-inline-1511344442.2007608.12986944765.png'
plot sin(x), cos(x),  exp(-x/10)*sin(x)**2
```

```
gnuplot> gnuplot> gnuplot> gnuplot> unset output
```

```
In [56]: %%gnuplot
         set parametric
         set contour base
         set grid xtics ytics ztics
         unset key
         splot cos(u)*cos(v),2*sin(u)*cos(v),3*sin(v)
```




```
set parametric

dummy variable is t for curves, u/v for surfaces
set contour base
set grid xtics ytics ztics
unset key
set output '/tmp/gnuplot-inline-1511344448.213879.121646573712.png'
splot cos(u)*cos(v),2*sin(u)*cos(v),3*sin(v)
unset output
```

8 Instalando o kernel do C ++ para o notebook Jupyter: Cling

A seguir mostramos como instalar o kernel do c++ no jupyter e para isso será usado o interpretador cling.

O cling é um intérpretador interativo do C++; Ele nos permite digitar e executar código C ++ dinamicamente, como Python ou Julia. Os desenvolvedores fornecem instantâneos binários para Cling em https://root.cern.ch/download/cling/. Baixe e extraia o associado à sua plataforma e sistema operacional. Considere que o nome do caminho para a pasta extraída é `/home/ubuntu_user/cling_ubuntu`. Haverá uma pasta chamada `bin`, que contém o binário Cling. Precisamos adicionar o caminho, ou seja, `/home/ubuntu_user/cling_ubuntu/bin` no caminho (*PATH*). Então, abra um terminal e digite o seguinte.

```
export PATH=/home/ubuntu_user/cling_ubuntu/bin:$PATH
```

Agora precisamos instalar o kernel Cling para o Jupyter. Para fazer isso, adicione o caminho para o kernel do Jupyter chamado `/home/ubuntu_user/cling_ubuntu/share/cling/Jupyter/kernel`. No terminal, alteramos o diretório para ele e instalamos o kernel da seguinte forma.

```
cd /cling-install-prefix/share/cling/Jupyter/kernel
```

```
sudo -H pip3 install -e .
```

Agora, para finalizar devemos registrar para o kernelspec:
sudo -H jupyter-kernelspec install cling.
Se você ainda não o fez, instale um compilador C++ como g++ e para isso use o terminal ou synaptic.
Certo, já terminamos. Para iniciar um notebook jupyter com kernel C++, basta digitar o seguinte no terminal:

jupyter-notebook

Uma página da Web denominada Início será aberta. No canto superior direito, teremos um botão intitulado novo, clique nele e selecione C ++.

8.1 Exportando e Convertendo Notebooks

No Jupyter, é possível converter um arquivo de documentos do notebook `.ipynb` em vários formatos estáticos através da ferramenta `nbconvert`. Atualmente, nbconvert é uma ferramenta de linha de comando, executado como um script usando Jupyter.

```
In [ ]: !jupyter nbconvert --to html 01-Instalacao-e-configuracao-do-IPython-e-Jupyter.ipynb
```

Atualmente, `nbconvert` suporta HTML (padrão), LaTeX, Markdown, reStructuredText, Python e Slides HTML5 para apresentações. Alguns tipos podem ser pós-processados, como LaTeX para PDF (isso requer Pandoc para ser instalado, no entanto).

```
In [ ]: !jupyter nbconvert --to latex 01-Instalacao-e-configuracao-do-IPython-e-Jupyter.ipynb
```

```
In [ ]: !evince 01-Instalacao-e-configuracao-do-IPython-e-Jupyter.pdf
```

Um serviço on-line muito útil é o IPython Notebook Viewer que permite que você exiba seu notebook como uma página HTML estática, que é útil para compartilhar com outras pessoas:

```
In [63]: %%html
         <iframe src=http://nbviewer.ipython.org/2352771 width=700 height=300></iframe>
```

```
<IPython.core.display.HTML object>
```

Além disso, o GitHub suporta renderização de Notebooks Jupyter armazenado em seus repositórios.

8.1.1 Coverter código python em notebook

Uma API IPythonpossui funções para ler e escrever arquivo Notebooks. Você deve usar esta API e não criar JSON diretamente. Por exemplo, o seguinte fragmento de código converte um script test.py em um notebook test.ipynb.

```
import IPython.nbformat.current as nbf
nb = nbf.read(open('test.py', 'r'), 'py')
nbf.write(nb, open('test.ipynb', 'w'), 'ipynb')
```

veja maiores detalhes em Converting to (not from) ipython Notebook format
Aqui temos um código completa para fazer isso:

```
In [ ]: import nbformat
        from nbformat.v4 import new_code_cell,new_notebook

        import codecs

        sourceFile = "codigo-fonte.py"       # <<<< Código a ser convertido em ipynb
        destFile   = "codigo-fonte.ipynb"    # <<<< Código convertido em ipynb


        def parsePy(fn):
            """ Generator that parses a .py file exported from a IPython notebook and
        extracts code cells (whatever is between occurrences of "In[*]:").
        Returns a string containing one or more lines
        """
            with open(fn,"r") as f:
                lines = []
                for l in f:
                    l1 = l.strip()
                    if l1.startswith('# In[') and l1.endswith(']:') and lines:
                        yield "".join(lines)
                        lines = []
                        continue
                    lines.append(l)
                if lines:
                    yield "".join(lines)

        # Create the code cells by parsing the file in input
        cells = []
        for c in parsePy(sourceFile):
            cells.append(new_code_cell(source=c))

        # This creates a V4 Notebook with the code cells extracted above
        nb0 = new_notebook(cells=cells,
                           metadata={'language': 'python',})
```

```
    with codecs.open(destFile, encoding='utf-8', mode='w') as f:
        nbformat.write(nb0, f, 4)
```

O código abaixo irá converter todos os arquivos .py em notebooks. Escolha o diretório e excute o código.

```
In [ ]: import nbformat
        from nbformat.v4 import new_code_cell,new_notebook
        import codecs
        import os


        def parsePy(fn):
            """ Generator that parses a .py file exported from a IPython notebook and
        extracts code cells (whatever is between occurrences of "In[*]:").
        Returns a string containing one or more lines
        """
            with open(fn,"r") as f:
                lines = []
                for l in f:
                    l1 = l.strip()
                    if l1.startswith('# In[') and l1.endswith(']:') and lines:
                        yield "".join(lines)
                        lines = []
                        continue
                    lines.append(l)
                if lines:
                    yield "".join(lines)

        def genNb(sourceFile,destFile):
            # Create the code cells by parsing the file in input
            cells = []
            for c in parsePy(sourceFile):
                cells.append(new_code_cell(source=c))

            # This creates a V4 Notebook with the code cells extracted above
            nb0 = new_notebook(cells=cells,
                               metadata={'language': 'python',})

            with codecs.open(destFile, encoding='utf-8', mode='w') as f:
                nbformat.write(nb0, f, 4)


        dir='.'  # quando vazio significa diretório local

        # Aqui encontramos todos os arquivo python no diretório
        arq_python = [x for x in os.listdir(dir) if x.endswith(".py")]
```

```
for x in arq_python :
    sourcefile=x
    destfile=x.split('.')[0]+'.ipynb'
    print("Source= ", sourcefile, " Destino = ", destfile)
    genNb(sourcefile,destfile)

# Para verificar o resultado
arq_conv = [x for x in os.listdir(dir) if x.endswith(".py") or x.endswith(".ipynb")]
print(arq_conv)
```

8.2 IPython paralelo

Em um nível alto, há três componentes básicos para paralelizar IPython:

- Máquina(s) - os processos remotos ou distribuídos onde seu código é executado.
- Cliente - sua interface para executar o código em motores.
- Controlador - a coleção de processos que coordenam Motores e Clientes.

Esses componentes vivem no pacote `IPython.parallel`, que foi lançado em seu próprio modelo que requer instalação.

8.2.1 Computação Paralela e Iterativa Com o IPython

ipyparallel é o novo nome do IPython.parallel. ipyparallel é um pacote Python e uma coleção de scripts CLI para controlar clusters para Jupyter.

ipyparallel contém os seguintes scripts CLI:

- ipcluster - inicia/para um cluster
- ipcontroller - inicia um agendador
- ipengine - inicia um motor

8.2.2 Instalação

Para instalar o ipyparallel digite na linha de comando:

```
sudo -H pip3 install -U ipyparallel
```

ou via `conda`:

```
sudo -H conda install ipyparallel
```

Para habilitar a tab do `IPython Clusters` no Jupyter Notebook:

```
sudo -H ipcluster nbextension enable
```

Para desabilitar ela novamente:

```
sudo -H ipcluster nbextension disable
```

Para instalar a guia IPython Clusters no Jupyter Notebook, adicione isso ao seu `jupyter_notebook_config.py`:

```
c.NotebookApp.server_extensions.append('ipyparallel.nbextension')
```

Este arquivo reside no subdiretório `~/.jupyter` do seu diretório pessoal e deve ser criado se ele ainda não existir.

Veja o "documentation on configuring the notebook server" para achar seu arquivo de configuração ou sua configuração inicial `jupyter_notebook_config.py`.

8.2.3 Instalando o JupyterHub

Para instalar o JupyterHub para todos os usuários, digite:

```
sudo -H jupyter nbextension install --sys-prefix --py ipyparallel
sudo -H jupyter nbextension enable --sys-prefix --py ipyparallel
sudo -H jupyter serverextension enable --sys-prefix --py ipyparallel
```

8.3 Rodando

Inicia um cluster:

```
ipcluster start
```

Usado ele a partir do Python:

```
import os
import ipyparallel as ipp
rc = ipp.Client()
ar = rc[:].apply_async(os.getpid)
pid_map = ar.get_dict()
```

8.4 Documentação

Para maiores detalhes consulte o manual "Using IPython for parallel computing".

Antes de executar a próxima célula, certifique-se de ter iniciado o cluster pela primeira vez, você pode usar a guia clusters no painel para fazer isso.

```
In [57]: from ipyparallel import Client
         client = Client()
         dv = client.direct_view()

In [58]: import os
         import ipyparallel as ipp
         rc = ipp.Client()
         ar = rc[:].apply_async(os.getpid)
         pid_map = ar.get_dict()

In [59]: len(dv)

   Out[59]:
```

```
In [60]: def where_am_i():
             import os
             import socket

             return "No processo com pid {0} no nó: '{1}'".format(
                 os.getpid(), socket.gethostname())
```

```
In [61]: where_am_i_direct_results = dv.apply(where_am_i)
         where_am_i_direct_results.get()
```

```
Out[61]: ["No processo com pid 30710 no nó: 'polaron'",
          "No processo com pid 30711 no nó: 'polaron'",
          "No processo com pid 30713 no nó: 'polaron'",
          "No processo com pid 30729 no nó: 'polaron'"]
```

9 Células Markdown

O texto pode ser adicionado aos notebooks IPython usando células Markdown. Markdown é uma linguagem de marcação popular que é um superconjunto de HTML. Sua especificação pode ser encontrada aqui:

http://daringfireball.net/projects/markdown/

9.1 Noções básicas do Markdown

Para adicionar um texto em itálico usa-se `*itálico*` como em *itálico* já para adicionar um em negrito usa-se `**negrito**` como em **negrito**.

9.2 Listas no Markdown

Pode-se criar facilmente listas enumeradas ou enumeradas aninhadas:

- Item um
  - Sublista tópico 1 do item um
    - subitem aninhado do tópico 1
  - Sublista tópico 2 do item um
    - primeiro subitem aninhado do tópico 2
    - segundo subitem aninhado do tópico 2
- Item dois
- Sublista do item 2
- Item três
- Sublista do item 3

Agora uma outra lista, mas agora numerada:

1. Esse é o primeiro item da lista
   1. Sublista do primeiro item
   2. Sublista do primeiro item
2. Essse é o segudo item da lista
3. Essse é o terceiro item da lista

9.3 Listas de tarefas

Para criar uma lista de tarefaz, inicie a lista com `[]`. Mareque a tarefa completada com `[x]`, conforme o exemplo a seguir

- [x] Terminei minhas alterações nesse arquivo
- [ ] Levar minhas alterações para o GitHub
- [ ] Trabalhar no novo texto

Pode-se adicionar linhas horizontais:

Aqui está o bloco de citação:

> Bonito é melhor do que feio.
>
> Explicito é melhor do que implícito.
>
> Simples é melhor do que complexo.
>
> Complexo é melhor do que complicado.
>
> Liso é melhor do que aninhado.
>
> Esparso é melhor do que denso.
>
> A legibilidade conta.
>
> Casos especiais não são especiais o suficiente para quebrar as regras.
>
> Embora a praticidade supera a pureza.
>
> Erros nunca devem passar em silêncio.
>
> A menos que seja explicitamente silenciado.
>
> Em face da ambiguidade, recusar a tentação de adivinhar.
>
> Deve haver uma - e preferivelmente somente uma - maneira óbvia de fazê-lo.
>
> Embora essa maneira não pode ser óbvia no início a menos que você seja holandês.
>
> Agora é melhor do que nunca.
>
> Embora nunca seja melhor do que *agora*.
>
> Se a implementação é difícil de explicar, então é uma má idéia.
>
> Se a implementação é fácil de explicar, então pode ser uma boa idéia.
>
> Namespaces são uma boa idéia- vamos fazer mais uso deles!

A seguir mostramso uma maneira simples de criar links externos:
O site do Jupyter/IPython
Se desejar, você pode adicionar títulos usando a sintaxe do Markdown:

10 Cabeçalho 1

11 Cabeçalho 2

11.1 Cabeçalho 2.1

11.2 Cabeçalho 2.2

11.2.1 Cabeçalho 3

Na maior parte do tempo deve-se usar as células de cabeçalho do notebook para organizar o conteúdo do seu notebook e manter seu documento estruturado, pois fornecem estrutura significativa que podem ser interpretada por outras ferramentas, não apenas grandes fontes em negrito.

Pode-se incorporar código destinado à ilustração em vez de execução em Python:

```
def f(x):
    """a docstring"""
    return x**2
```

ou em outras linguagens:

```
if (i=0; i<n; i++) {
  printf("hello %d\n", i);
  x += 4;
}
```

11.3 Equaçõe LaTeX

Para escrever equações matemáticas em uma célula do markdown, usa-se o mesma sintaxe do LaTeX, na qual para escrever uma equação *"inline"* no meio do texto deve-se usar o par de símbolos $ equação $. Para escrever uma equação que seja mostrada sozinha em uma única linha, deve-se usar em uma linhas separadas o par de símbolos $$ equação $$.

Considere por exemplo a seguinte equação no meio do texto $e^{i\pi} + 1 = 0$, e q seguinte equação em uma linha sozinha:

$$e^x = \sum_{i=0}^{\infty} \frac{1}{i!} x^i \tag{5}$$

11.3.1 Numerando equações LaTeX no notebook

Para maiores detalhes colsulte o link.

Pode-se ativar a numeração executando a célula abaixo com o seguinte contéudo

```
In [62]: %%javascript MathJax.Hub.Config({ TeX: { equationNumbers: { autoNumber:
           "AMS" } } });
```

```
<IPython.core.display.Javascript object>
```

Há também uma extensão para a numeração de uma equação

Para instalar a extensão faça o seguinte:

```
sudo -H pip3 install jupyter_contrib_nbextensions
jupyter contrib nbextension install --user
jupyter nbextension enable equation-numbering/main
```

Usando o MathJax, pode-se incluir expressões matemáticas no seu texto, usando a nomenclatura LaTeX, conforme exemplo a seguir: $e^{i\pi} + 1 = 0$ e mostrado como:

$$e^x = \sum_{i=0}^{\infty} \frac{1}{i!} x^i$$

11.4 Tabelas no Markdown

Há uma interface web a qual gera uma tabela Markdown para você de forma simples: http://www.tablesgenerator.com/markdown_tables

Porque Markdown é um superconjunto de HTML você pode até mesmo adicionar coisas como tabelas HTML:

<table>
<tr><th>Cabeçalho 1</th><th>Cabeçalho 2</th></tr>
<tr><td>linha 1, celula 1</td><td>linha 1, celula 2</td></tr>
<tr><td>linha 2, celula 1</td><td>linha 2, celula 2</td></tr>
</table>

Uma outra forma de fazer tabelas é usando o seguinte código

```
| H1-C1 | H2-C2 | H3-C3 | H4-C4 | H5-C5 |
|-------|-------|-------|-------|-------|
| L2-C1 | L2-C2 | L2-C3 | L2-C4 | L2-C5 |
| L3-C1 | L3-C2 | L3-C3 | L3-C4 | L3-C5 |
| L4-C1 | L4-C2 | L4-C3 | L4-C4 | L4-C5 |
```

obtemos como resultado a seguinte tabela:

H1-C1	H2-C2	H3-C3	H4-C4	H5-C5
L2-C1	L2-C2	L2-C3	L2-C4	L2-C5
L3-C1	L3-C2	L3-C3	L3-C4	L3-C5
L4-C1	L4-C2	L4-C3	L4-C4	L4-C5

Quanto ao alinhamento das coulnas, eles podem ser controlados usando o seguinte código

```
| Esquerda     | Centralizado   | Direita       |
| :---         |     :---:      |          ---: |
| git status   | git status     | git status    |
| git diff     | git diff       | git diff      |
```

o qual resulta na seguinte tabela

Esquerda	Centralizado	Direita
git status	git status	git status

Esquerda	Centralizado	Direita
git diff	git diff	git diff

11.5 Incluindo arquivos locais: imagens, vídeos, etc.

Se você tiver arquivos locais no diretório do Notebook, você pode consultar e/ou inserir esses arquivos diretamente nas células do Markdown:

```
[subdiretorio/]<arquivo>
```

Por exemplo, na pasta de Figs, temos o logotipo Python:

```
<img src="Figs/python_logo.svg" />
```

e um vídeo com uma tag de vídeo HTML5:

```
<video controls src="Figs/animation.m4v" />
```

Esses não incorporam os dados no arquivo do notebook e exigem que os arquivos existam quando você estiver visualizando o notebook.

Note que isso significa que o servidor de notebooks IPython também atua como um servidor de arquivos genérico para arquivos dentro da mesma árvore que seus notebooks. O acesso não é concedido fora da pasta do caderno de modo que você tenha controle estrito sobre quais arquivos são visíveis, mas por esse motivo é altamente recomendável que você não execute o servidor de notebook com um diretório de notebook em um nível alto em seu sistema de arquivos diretório).

Quando você executa o notebook de maneira protegida por senha, o acesso a arquivos locais é restrito a usuários autenticados, a menos que as exibições somente leitura estejam ativas.

11.6 Comentários no Markdown

Em geral é sempre muito útil poder introduzir alguns comentários em um texto:
http://daringfireball.net/projects/markdown/syntax#link

```
[comment]: <> (This is a comment, it will not be included)
[comment]: <> (in  the output file unless you use it in)
[comment]: <> (a reference style link.)
```

Para aperfeiçoar a compatibilidade da platforma tamém, é possível usar # (o qual é uma taga para um hiperlink legitimo) em vez de <>:

```
[//]: # (This may be the most platform independent comment)
```