

# ART|IEEEFICIAIS



# **Introdução à Machine Learning com Python**

**Dos primeiros passos até o desenvolvimento avançado em  
Machine Learning**



**Carine Gottschall**

**Lucas Alves**



# Conteúdo Programático

1. Gestão de pacotes e ambientes em Python
  - I. Anaconda
  - II. Jupyter Notebook
  - III. Google Colab
2. Pacotes essenciais ao desenvolvimento de RNA com Python
  - I. Numpy
  - II. Pandas
  - III. Tratamento de Dados
3. Machine Learning
  - I. Regressão Linear
  - II. Classificação
  - III. Clustering (K-means)

ARTIFICIAIS

# Conteúdo Programático

## 4. Deep Learning

### I. Perceptron multicamadas

- i. Perceptron básico
- ii. Backpropagation e otimização
- iii. Treinamento da Rede
- iv. Criando do zero

### II. Principais Arquiteturas de Deep Learning

- i. Redes densas
- ii. Convolucionais
- iii. Recorrentes
- iv. Auto codificadoras
- v. GAN (Generative Adversarial Network)
- vi. Aprendizado por reforço

ARTIFICIAIS

# Conteúdo Programático

## 4. Deep Learning

### III. Redes convolucionais

- i. Camadas convolutivas
- ii. Camadas de reagrupamento
- iii. Visualização filtros característicos

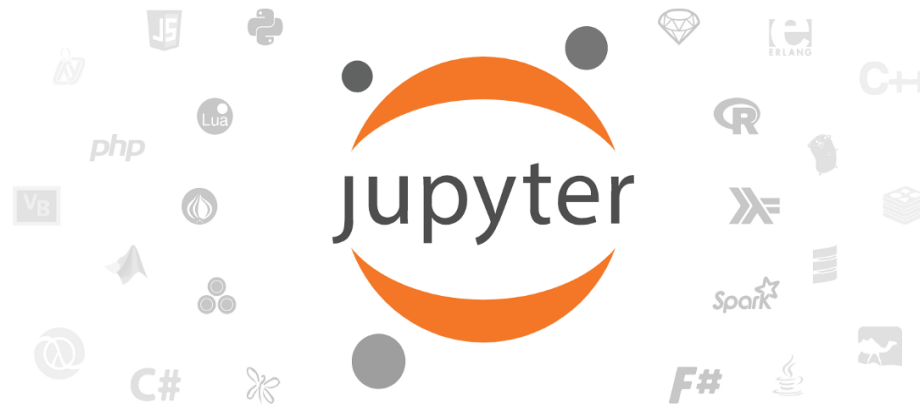
### IV. Transferência de aprendizado

- i. Aprendizado Fino
- ii. Aprendizado Completo
- iii. Mudança de domínio

### V. Projetos

ART|EEE|FICIAIS

# Gestão de Pacotes e Ambientes em Python



# O que é o Anaconda?

O [Anaconda](#) é uma distribuição de pacotes construída para análise de dados. É onde os cientistas de dados compartilham seu trabalho. Você pode pesquisar e baixar pacotes e notebooks populares Python e R para iniciar seu trabalho de ciência de dados.

ART|EEE|FICIAIS



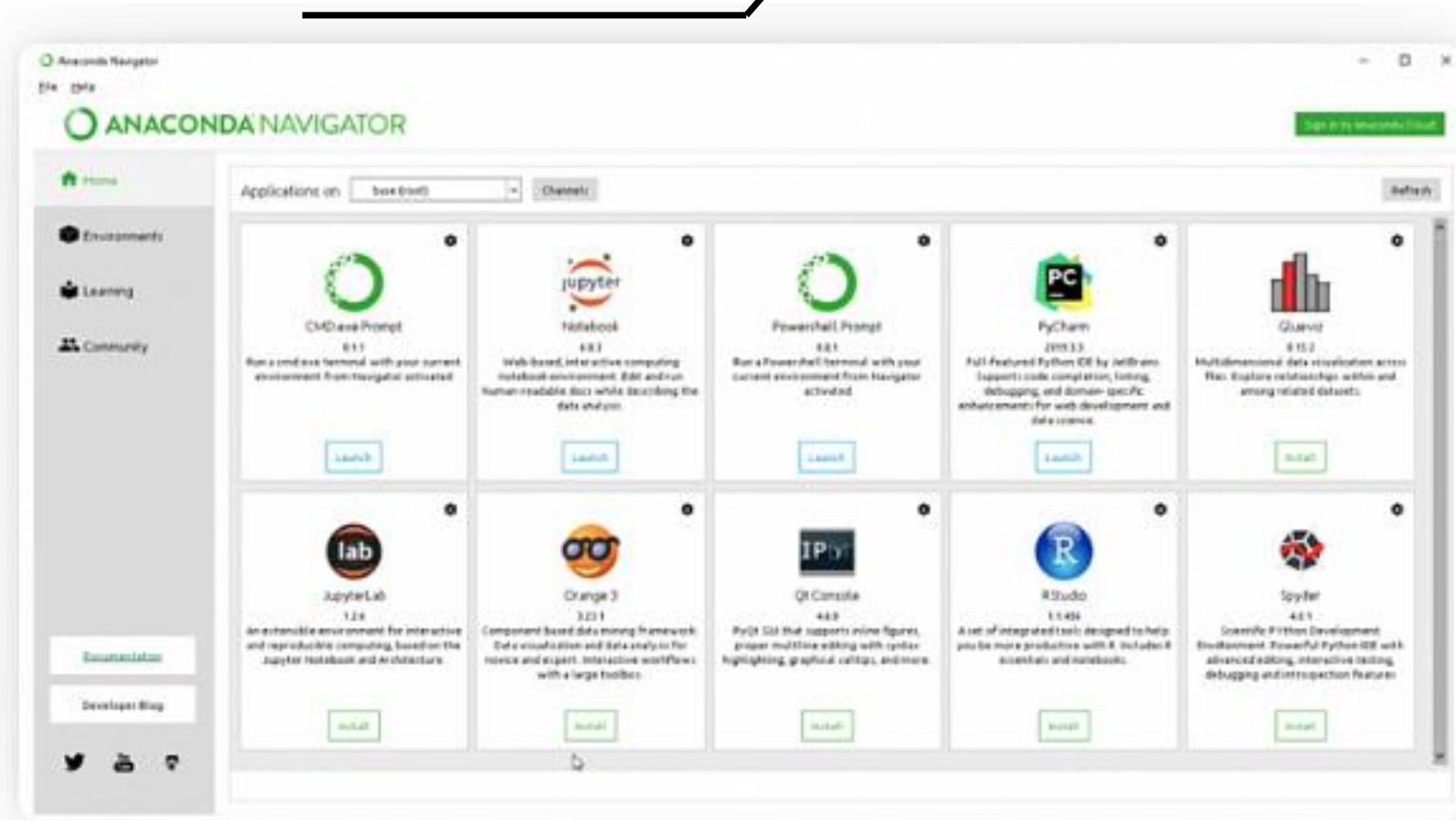
O que a distribuição inclui?

- ✓ O Python e mais 150 pacotes científicos e suas dependências (~500MB download);
- ✓ Os pacotes mais comuns de Data Science do Python.

❑ Se você precisa economizar espaço, para embarcar a distribuição em um minicomputador como o Raspberry Pi, por exemplo, há uma distribuição menor que inclui apenas o conda e o Python.




# Anaconda Navigator



# Instalando o Anaconda

O Anaconda está disponível para Windows, Mac OS X e Linux. É possível encontrar os instaladores e as instruções de instalação no site <https://www.continuum.io/downloads>.



Caso já tenha o Python instalado em seu computador, essa instalação não estragará nada. Em vez disso, o Python padrão usado nos scripts e programas passará a ser o que vem dentro da distribuição Anaconda.

# Gerenciando pacotes

## conda

- Gerenciador de pacotes e ambientes do Anaconda;
- É capaz de instalar pacotes não específicos do python;
- Mais voltado para Data Science.

**X**

## pip

- Gerenciador de pacotes padrões do python;
- Mais generalista.

ARTIFICIAIS

# Gerenciando pacotes

- ❑ Instalando pacotes:

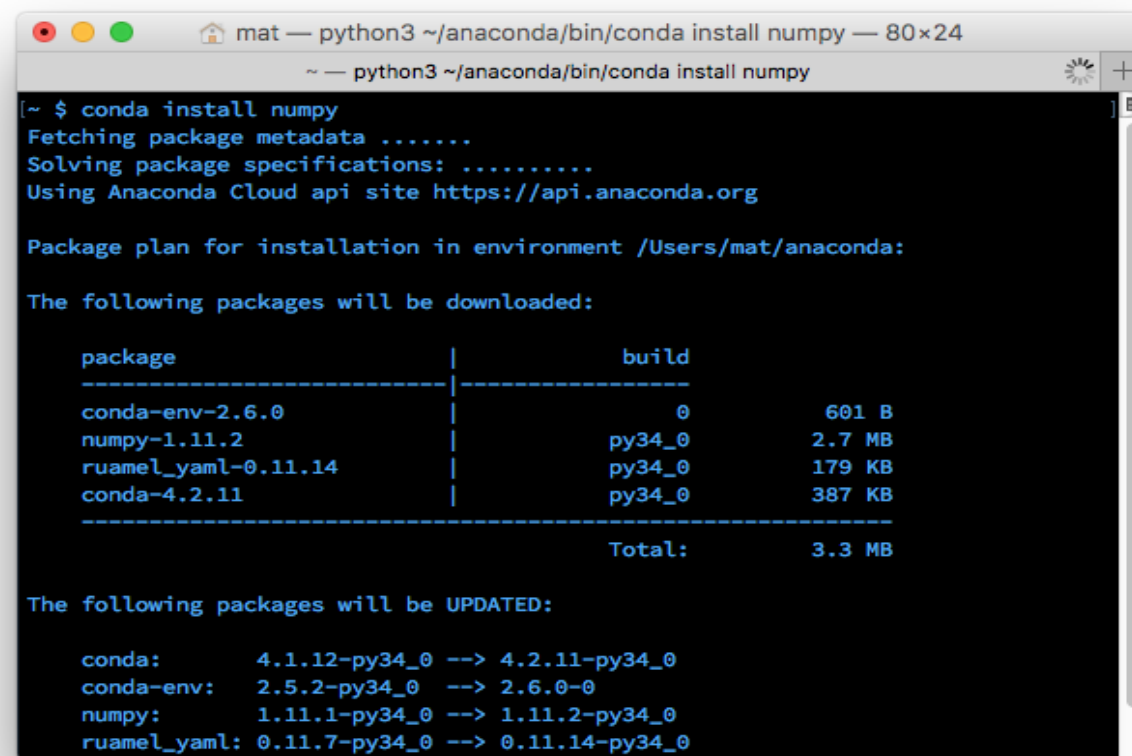
`conda install nome_do_pacote`

- ❑ Especificando versão do pacote:

`conda install nome_do_pacote=x.x`

O gerenciador **conda** é responsável por procurar, empacotar e instalar a biblioteca no ambiente.

- ❖ Caso não seja especificada a versão do pacote, ocorrerá a instalação da versão mais recente compatível com sua versão Python.



```
mat — python3 ~/anaconda/bin/conda install numpy — 80×24
~ — python3 ~/anaconda/bin/conda install numpy

[~ $ conda install numpy
Fetching package metadata .....
Solving package specifications: .....
Using Anaconda Cloud api site https://api.anaconda.org

Package plan for installation in environment /Users/mat/anaconda:

The following packages will be downloaded:

package | build
-----|-----
conda-env-2.6.0 | 0 601 B
numpy-1.11.2 | py34_0 2.7 MB
ruamel_yaml-0.11.14 | py34_0 179 KB
conda-4.2.11 | py34_0 387 KB
-----|-----
Total: 3.3 MB

The following packages will be UPDATED:

conda: 4.1.12-py34_0 --> 4.2.11-py34_0
conda-env: 2.5.2-py34_0 --> 2.6.0-0
numpy: 1.11.1-py34_0 --> 1.11.2-py34_0
ruamel_yaml: 0.11.7-py34_0 --> 0.11.14-py34_0
```

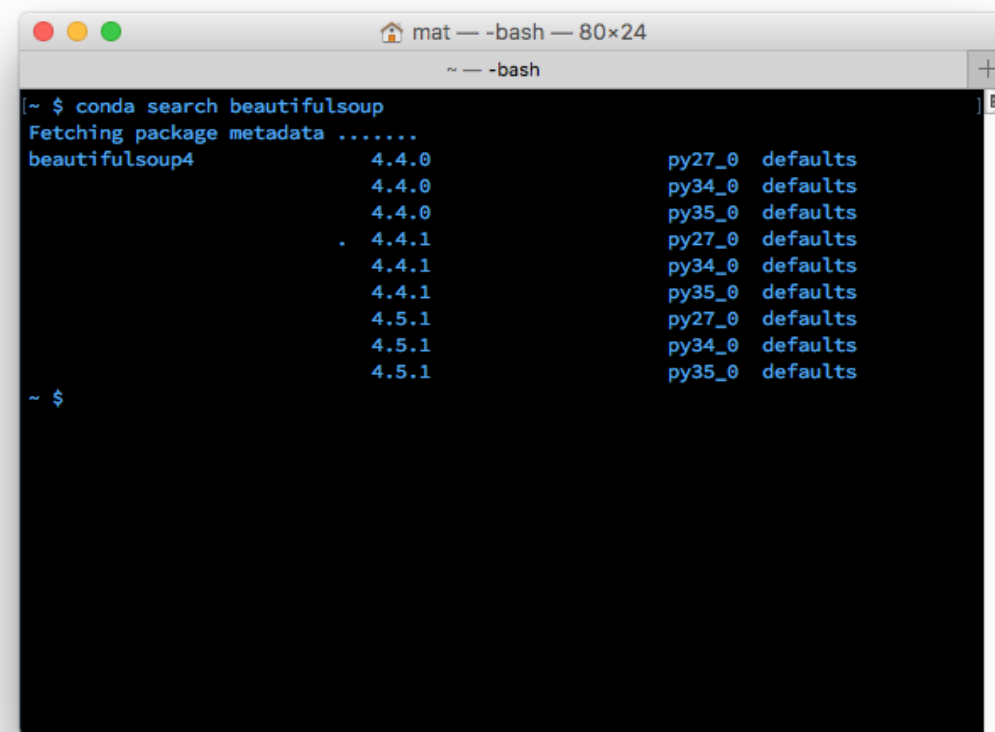
# Gerenciando pacotes

Caso você não saiba exatamente o nome do pacote que está buscando, é possível tentar encontrá-lo com o comando:

- **conda search termo\_de\_busca**

Por exemplo, eu sei que quero instalar o pacote [Beautiful Soup](#), porém, não tenho certeza do nome exato, então, eu digito:

- **conda search beautifulsoup**



```
mat — -bash — 80x24
~ — -bash
[~ $ conda search beautifulsoup
Fetching package metadata .....
beautifulsoup4      4.4.0      py27_0 defaults
                   4.4.0      py34_0 defaults
                   4.4.0      py35_0 defaults
                   . 4.4.1      py27_0 defaults
                   4.4.1      py34_0 defaults
                   4.4.1      py35_0 defaults
                   4.5.1      py27_0 defaults
                   4.5.1      py34_0 defaults
                   4.5.1      py35_0 defaults
~ $
```

❖ O programa retorna uma lista com os pacotes Beautiful Soup disponíveis com o nome apropriado do pacote, **beautifulsoup4**.

# Gerenciando pacotes

- ❑ Listando pacotes:

`conda list`

- ❑ Removendo pacotes:

`conda remove nome_do_pacote`

- ❑ Atualizando pacotes:

`conda upgrade nome_do_pacote`

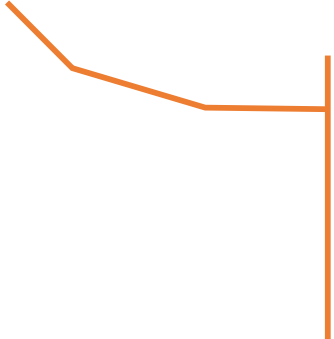
- ❑ Para atualizar todos os pacotes:

`conda upgrade conda`  
`conda upgrade --all`

ART|EEE|FICIAIS

# Gerenciando Ambientes

Ambientes permitem que você separe e isole pacotes que estão sendo utilizados para projetos diferentes. Códigos que dependem de versões diferentes de uma mesma biblioteca, por exemplo, é possível ter código que use aspectos novos do Numpy e outros que usem aspectos antigos que foram removidos das versões novas. Nesse caso, cada versão do Numpy deve estar presente em um ambiente diferente.



Além de administrar pacotes, o conda também gerencia ambientes virtuais, similar ao [virtualenv](#) e ao [pyenv](#), outros gerenciadores de ambientes famosos.

# Gerenciando Ambientes

- ❑ Criando um novo ambiente:

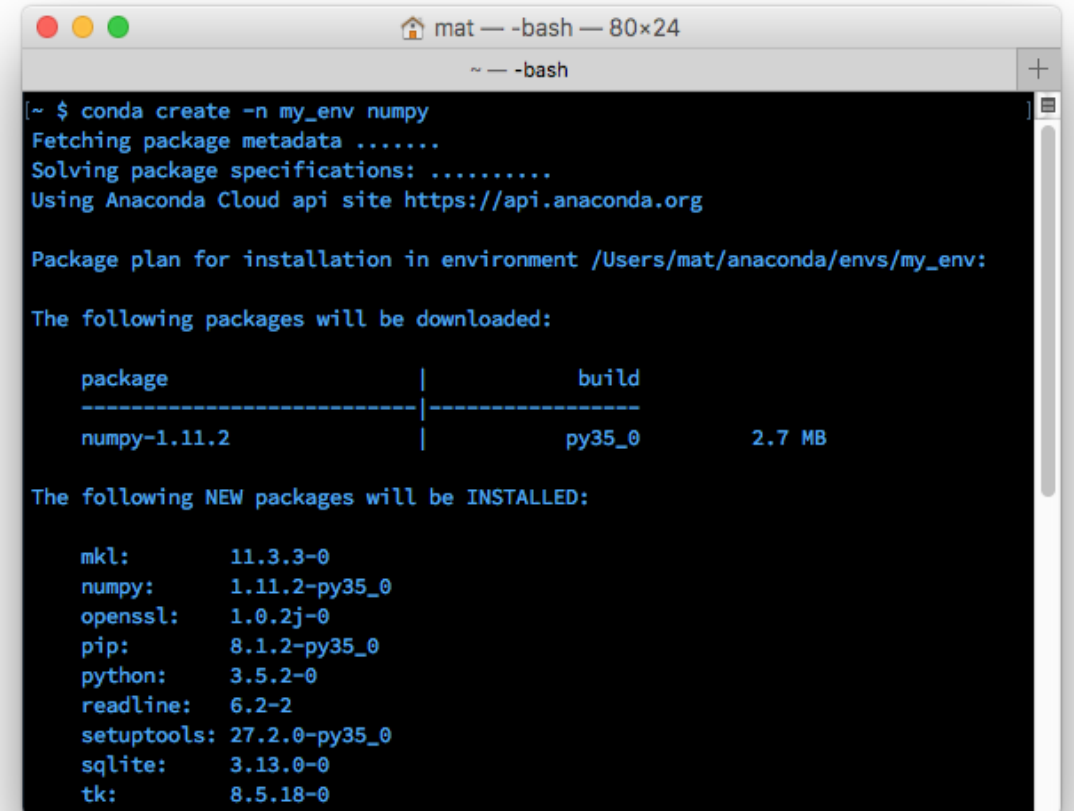
`conda create -n nome_do_ambiente`

- ❑ Criando ambientes com uma versão específica de python:

`conda create -n nome_do_ambiente  
python=3.6`

- ❑ Criando ambientes com pacotes já instalados:

`conda create -n nome_do_ambiente  
nome_do_pacote`

A terminal window titled 'mat - bash - 80x24' showing the execution of 'conda create -n my\_env numpy'. The output displays the fetching of package metadata, solving specifications, and the use of the Anaconda Cloud API. It then shows the package plan for installation in the environment '/Users/mat/anaconda/envs/my\_env', listing the packages to be downloaded: numpy-1.11.2 (py35\_0, 2.7 MB). Finally, it lists the new packages to be installed: mkl, numpy, openssl, pip, python, readline, setuptools, sqlite, and tk.

```
mat - bash - 80x24
~ - bash

[~ $ conda create -n my_env numpy
Fetching package metadata .....
Solving package specifications: .....
Using Anaconda Cloud api site https://api.anaconda.org

Package plan for installation in environment /Users/mat/anaconda/envs/my_env:

The following packages will be downloaded:

package | build
-----|-----
numpy-1.11.2 | py35_0 2.7 MB

The following NEW packages will be INSTALLED:

mkl:      11.3.3-0
numpy:    1.11.2-py35_0
openssl:  1.0.2j-0
pip:      8.1.2-py35_0
python:   3.5.2-0
readline: 6.2-2
setuptools: 27.2.0-py35_0
sqlite:   3.13.0-0
tk:       8.5.18-0
```



# Gerenciando Ambientes

❑ Listando ambientes:

`conda env list`

❑ Removendo um ambiente:

`conda env remove -n nome_do_ambiente`

ART|E|E|E|FICIAIS

# Gerenciando Ambientes

## Entrando em um ambiente

❑ Comando Windows:  
`activate nome_do_ambiente`

❑ Comando OSX/Linux:  
`source activate nome_do_ambiente`

## Saindo de um ambiente

❑ Comando Windows:  
`deactivate nome_do_ambiente`

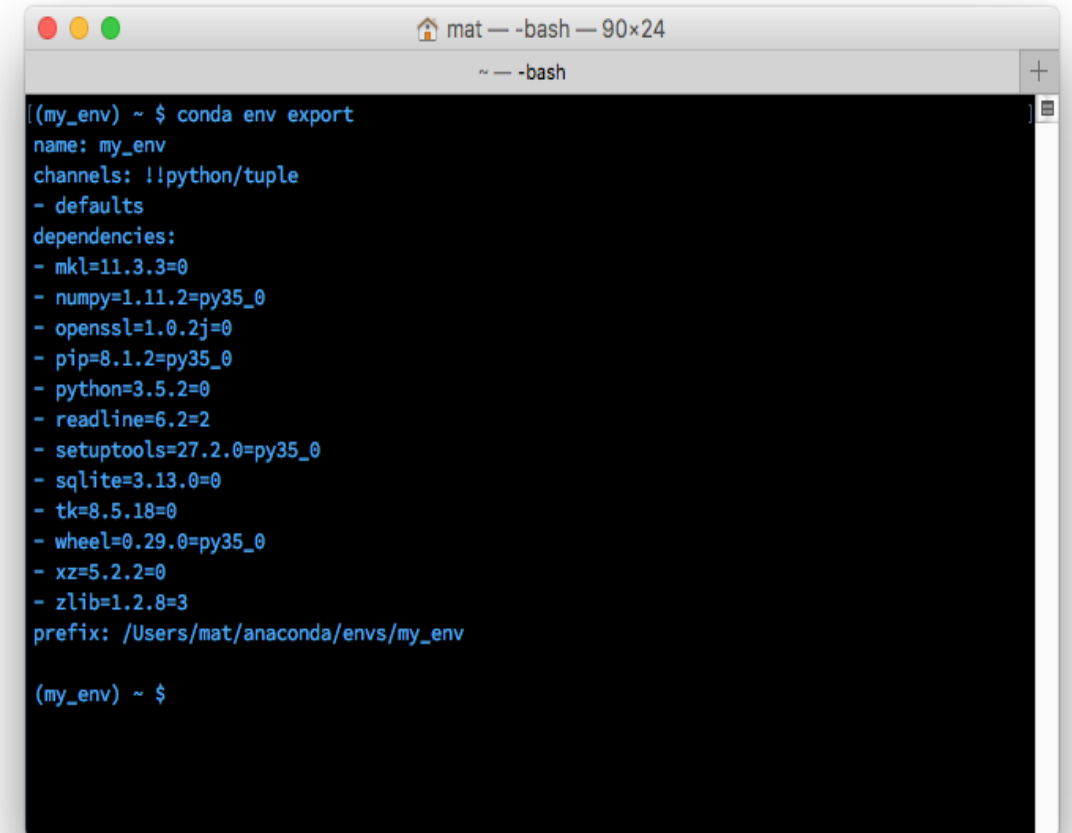
❑ Comando OSX/Linux:  
`source deactivate nome_do_ambiente`

# Gerenciando Ambientes

Também é possível salvar e carregar um ambiente para compartilhá-lo, salvando os pacotes em um arquivo YAML :

- `conda env export > environment.yaml`

A primeira parte, **conda env export**, escreve todos os pacotes no ambiente, incluindo a versão Python. A segunda parte do comando de exportar, **> environment.yaml**, escreve o texto exportado em um arquivo YAML chamado **environment.yaml**.

A terminal window titled 'mat - bash - 90x24' showing the output of the 'conda env export' command. The output is a YAML configuration for an environment named 'my\_env'. It lists channels as '!!python/tuple', defaults, and a list of dependencies including mkl, numpy, openssl, pip, python, readline, setuptools, sqlite, tk, wheel, xz, and zlib. The prefix is set to '/Users/mat/anaconda/envs/my\_env'. The prompt '(my\_env) ~ \$' is visible at the bottom.

```
(my_env) ~ $ conda env export
name: my_env
channels: !!python/tuple
- defaults
dependencies:
- mkl=11.3.3=0
- numpy=1.11.2=py35_0
- openssl=1.0.2j=0
- pip=8.1.2=py35_0
- python=3.5.2=0
- readline=6.2=2
- setuptools=27.2.0=py35_0
- sqlite=3.13.0=0
- tk=8.5.18=0
- wheel=0.29.0=py35_0
- xz=5.2.2=0
- zlib=1.2.8=3
prefix: /Users/mat/anaconda/envs/my_env

(my_env) ~ $
```

Para restaurá-lo, use o comando **conda env create -f environment.yaml**. Isso criará um novo ambiente com o mesmo nome contido no arquivo `environment.yaml`.

# Challenge

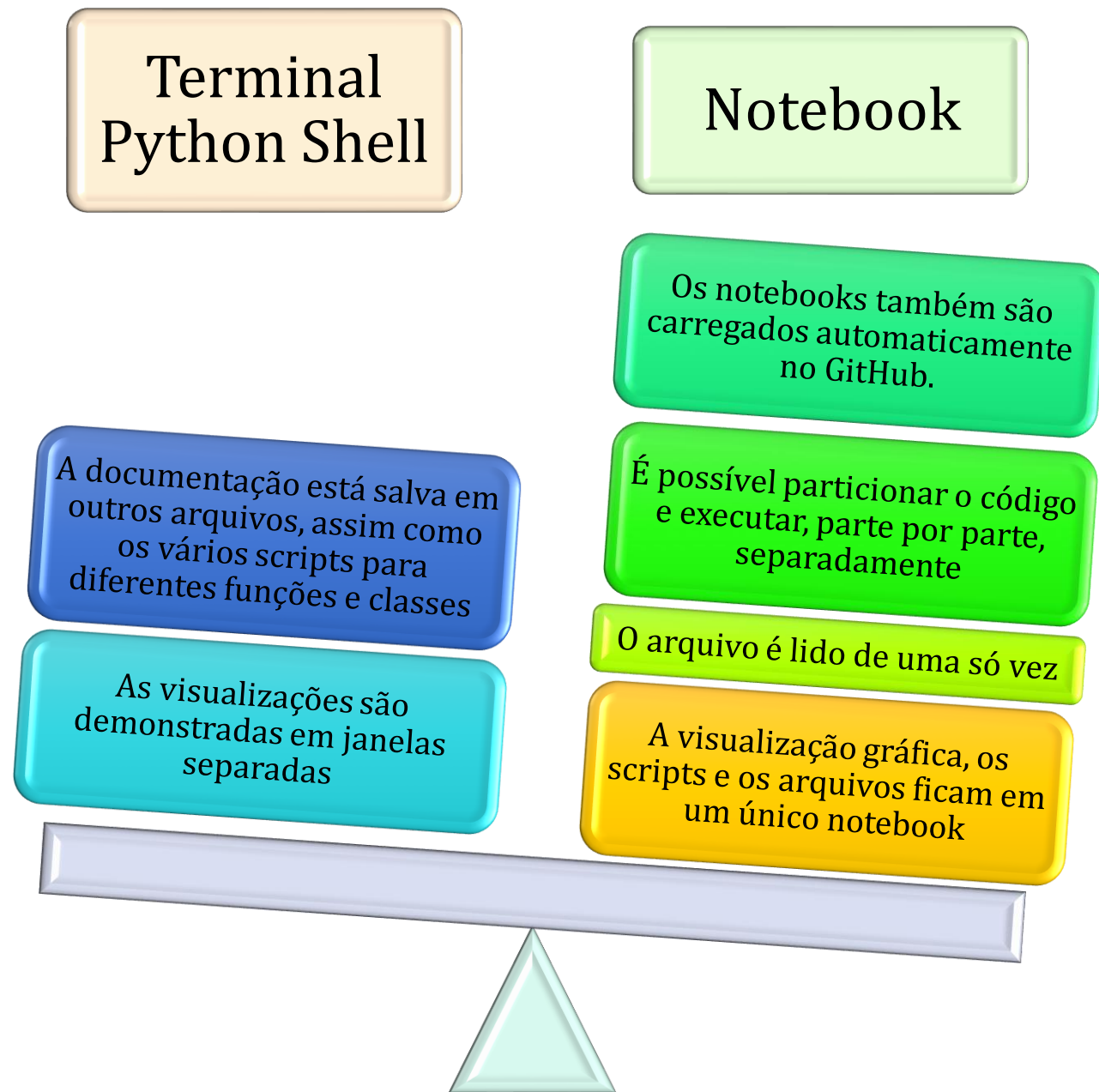
The background features a stylized illustration of a human brain. The left hemisphere is colored purple and the right hemisphere is blue. Grey lines representing neural connections or circuitry extend from the brain towards the top and right edges of the frame. A large, blue, three-dimensional ribbon banner is draped across the middle of the image, serving as a container for the challenge text.

Utilizando o *Anaconda prompt*  
crie um ambiente python 3.6 e  
instale as bibliotecas numpy e  
pandas

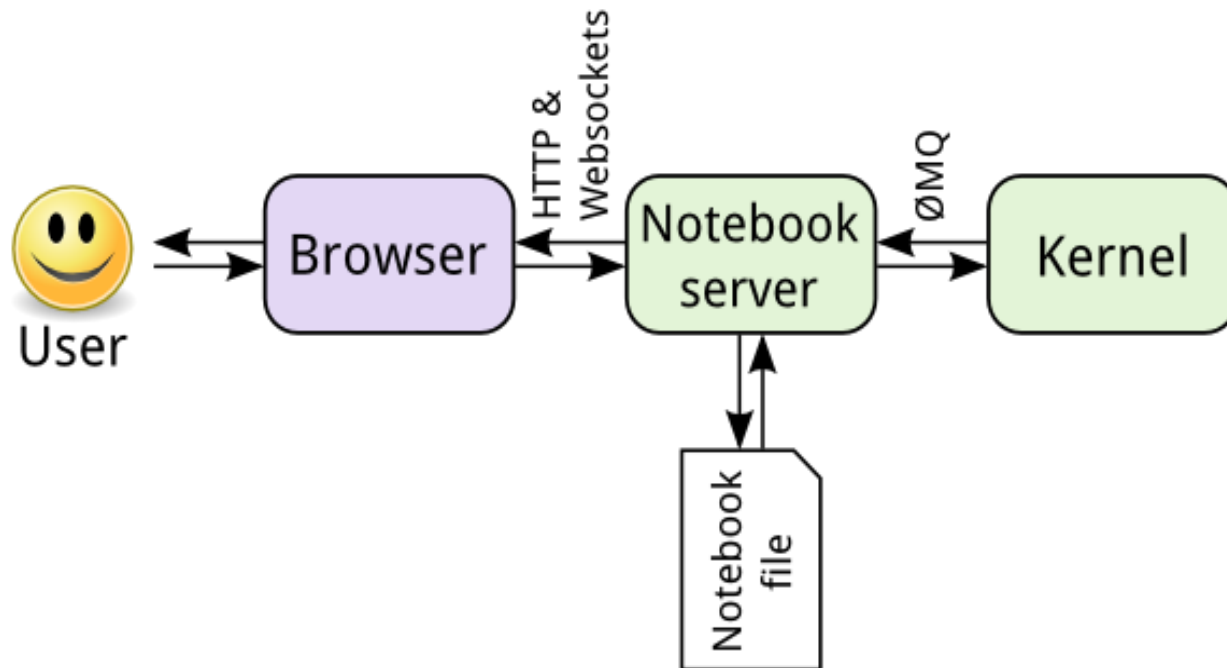
# Jupyter Notebook

O notebook é uma aplicação web que permite que você combine texto explicativo, equações matemáticas, código e visualizações em um único documento facilmente compartilhável. Os notebooks se tornaram rapidamente uma ferramenta essencial para trabalhar com dados. Você os verá sendo usados para a limpeza e exploração de dados, visualização, machine learning e até análise de big data.

Qual a  
diferença  
entre o Jupyter  
e um terminal  
Python Shell  
ou Ipython?



# Como os notebooks funcionam?



O ponto central é o servidor do notebook. A conexão é feita no servidor por seu navegador, e o notebook é carregado como um aplicativo web. O código escrito nesse aplicativo é mandado pelo servidor para o núcleo. O núcleo roda o código e o manda de volta para o servidor, então, o output é carregado no navegador. Ao salvar um notebook, ele é escrito no servidor como um arquivo JSON com a extensão `.ipynb`.

# Instalando o Jupyter Notebook

O jeito mais fácil de instalar o Jupyter é baixando o Anaconda. Os notebooks Jupyter vêm embutidos na distribuição. É possível usar os notebooks já no ambiente padrão.

Para instalar os notebooks Jupyter em um ambiente do conda, use:

- `conda install jupyter notebook`

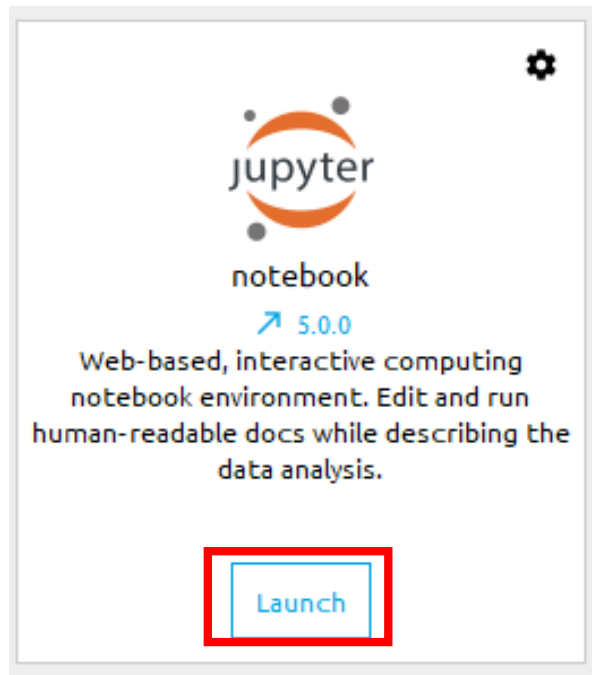
Os notebooks Jupyter também estão disponíveis no pip, digitando:

- `pip install jupyter notebook`



# Iniciando o Jupyter

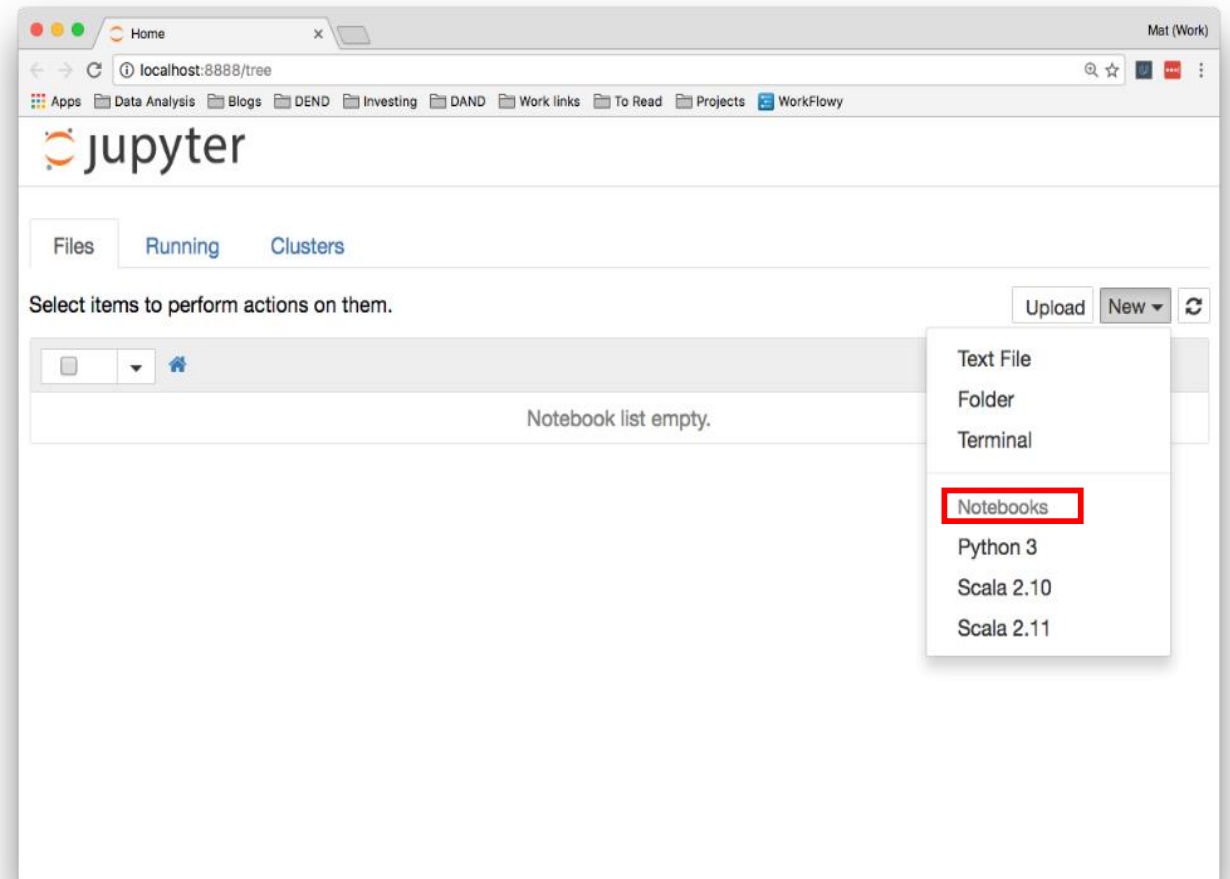
- Através do Anaconda Navigator
- Através do prompt de comando



- Utilize o comando:  
`jupyter notebook`

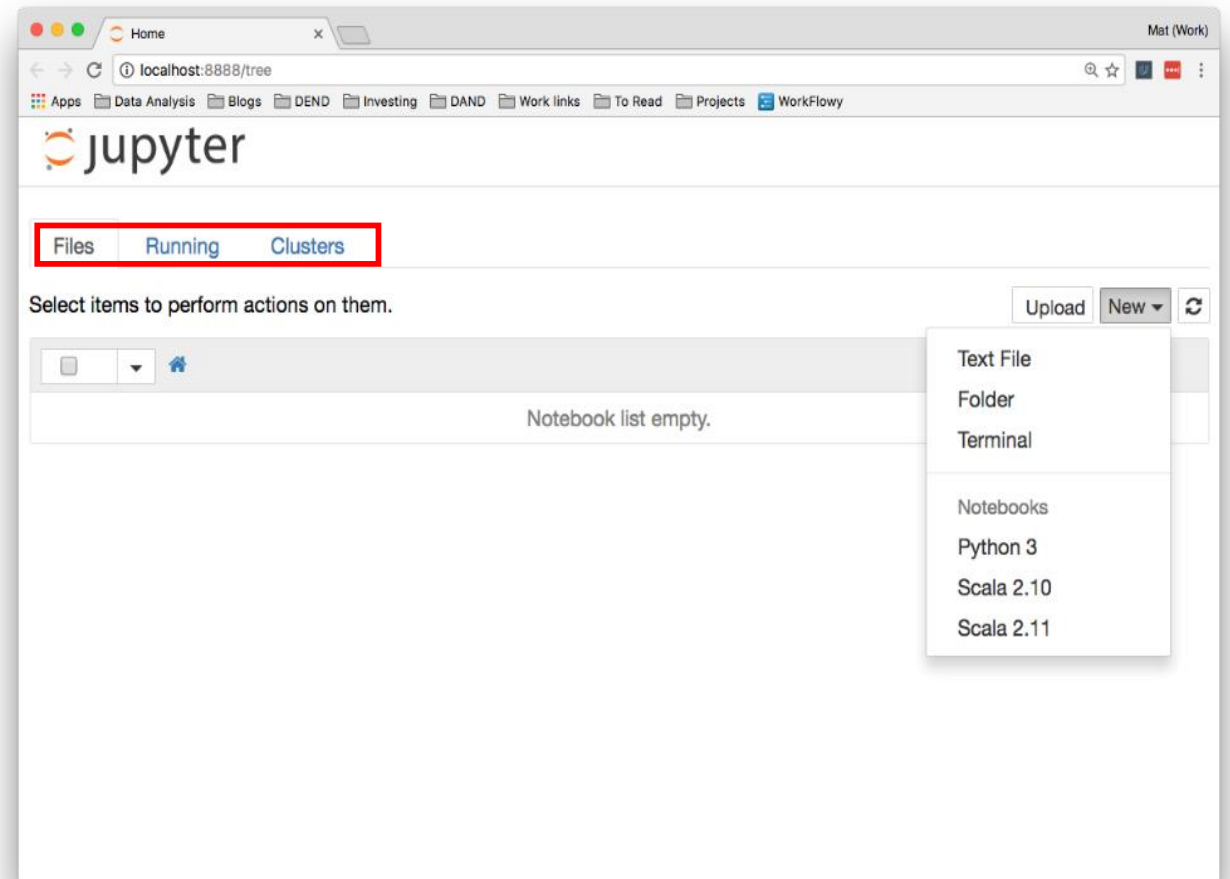
# Criando um novo notebook

No canto direito superior, você pode clicar em "**New**" para criar um notebook, arquivo de texto, pasta ou terminal novo. A lista abaixo de "**Notebooks**" mostra os núcleos (kernels) que você tem instalados. Neste caso, os núcleos disponíveis são Python 3, Scala 2.10 e 2.11, que também aparecem na lista.



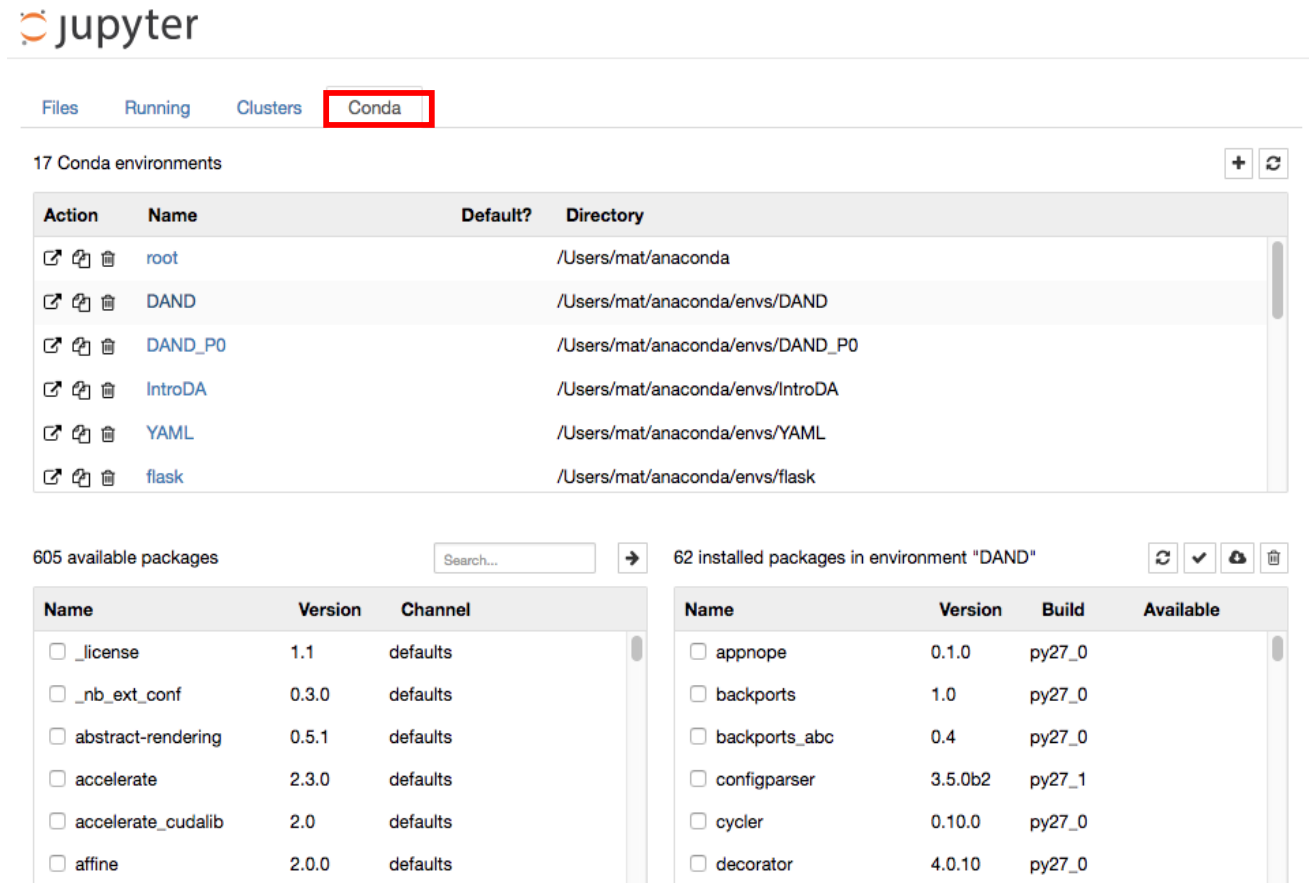
# Criando um novo notebook

As abas no topo mostram Files, Running e Cluster. Files mostra todos os arquivos e pastas do diretório atual. Clicar na aba Running listará todos os notebooks atualmente ativos. Neste ponto, é possível gerenciá-los. Clusters era onde antes você podia criar núcleos múltiplos para usar em computação paralela. Agora, isso foi tomado pelo ipyparallel, então, não há nada demais a ser feito aqui.



# Criando um novo notebook

Caso esteja rodando o servidor do notebook de um ambiente conda, você também terá acesso a uma aba nomeada "**Conda**", como mostraremos abaixo. Aqui, é possível administrar os ambientes de dentro do Jupyter. É possível criar ambientes, instalar pacotes, atualizar pacotes, exportar ambientes e muito mais.





















The screenshot displays the Jupyter web interface with the 'Conda' tab selected. It shows a list of 17 Conda environments, including 'root', 'DAND', 'DAND\_P0', 'IntroDA', 'YAML', and 'flask'. Below this, there are two panels: '605 available packages' and '62 installed packages in environment "DAND"'. The 'available packages' panel lists various packages like '\_license', '\_nb\_ext\_conf', 'abstract-rendering', 'accelerate', 'accelerate\_cudalib', and 'affine'. The 'installed packages' panel lists packages like 'appnope', 'backports', 'backports\_abc', 'configparser', 'cycler', and 'decorator'.

**Jupyter**

Files Running Clusters **Conda**

17 Conda environments

Action	Name	Default?	Directory
  	root		/Users/mat/anaconda
  	DAND		/Users/mat/anaconda/envs/DAND
  	DAND_P0		/Users/mat/anaconda/envs/DAND_P0
  	IntroDA		/Users/mat/anaconda/envs/IntroDA
  	YAML		/Users/mat/anaconda/envs/YAML
  	flask		/Users/mat/anaconda/envs/flask

605 available packages

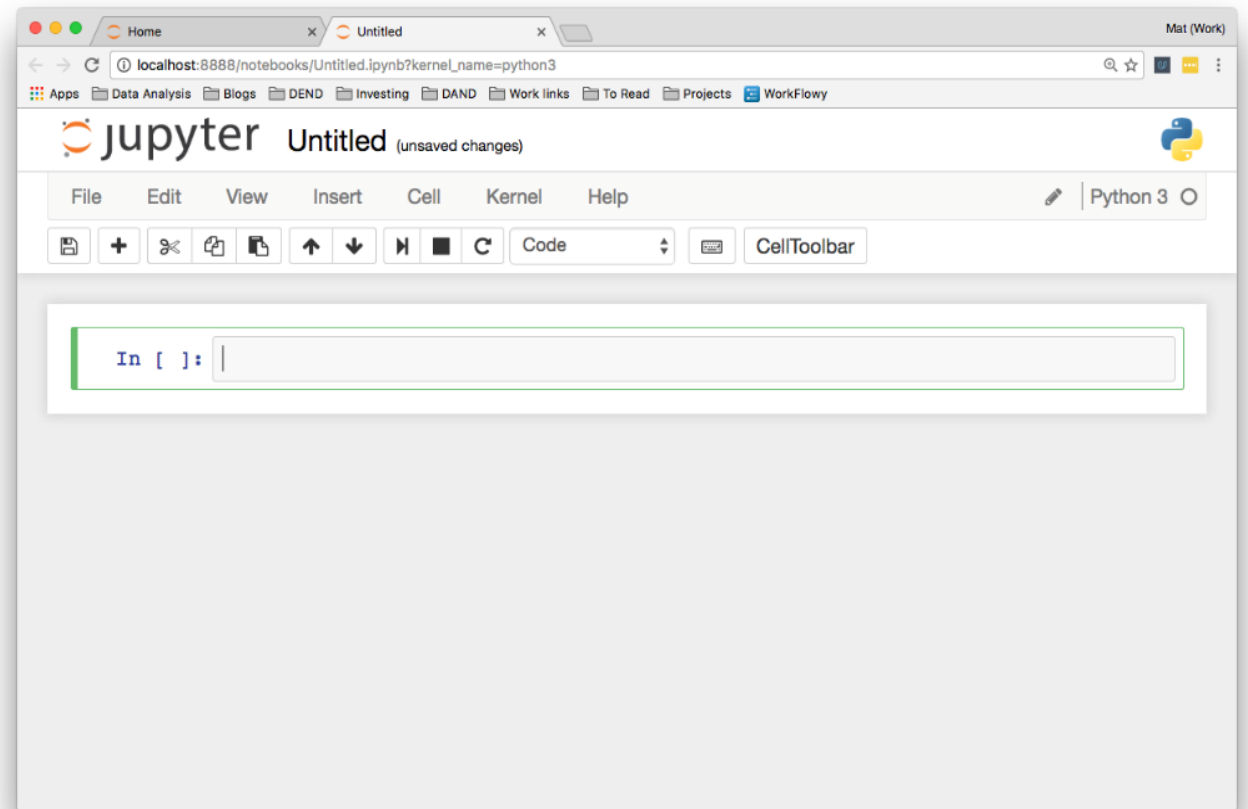
Name	Version	Channel
<input type="checkbox"/> _license	1.1	defaults
<input type="checkbox"/> _nb_ext_conf	0.3.0	defaults
<input type="checkbox"/> abstract-rendering	0.5.1	defaults
<input type="checkbox"/> accelerate	2.3.0	defaults
<input type="checkbox"/> accelerate_cudalib	2.0	defaults
<input type="checkbox"/> affine	2.0.0	defaults

62 installed packages in environment "DAND"

Name	Version	Build	Available
<input type="checkbox"/> appnope	0.1.0	py27_0	
<input type="checkbox"/> backports	1.0	py27_0	
<input type="checkbox"/> backports_abc	0.4	py27_0	
<input type="checkbox"/> configparser	3.5.0b2	py27_1	
<input type="checkbox"/> cycler	0.10.0	py27_0	
<input type="checkbox"/> decorator	4.0.10	py27_0	

# Interface do notebook

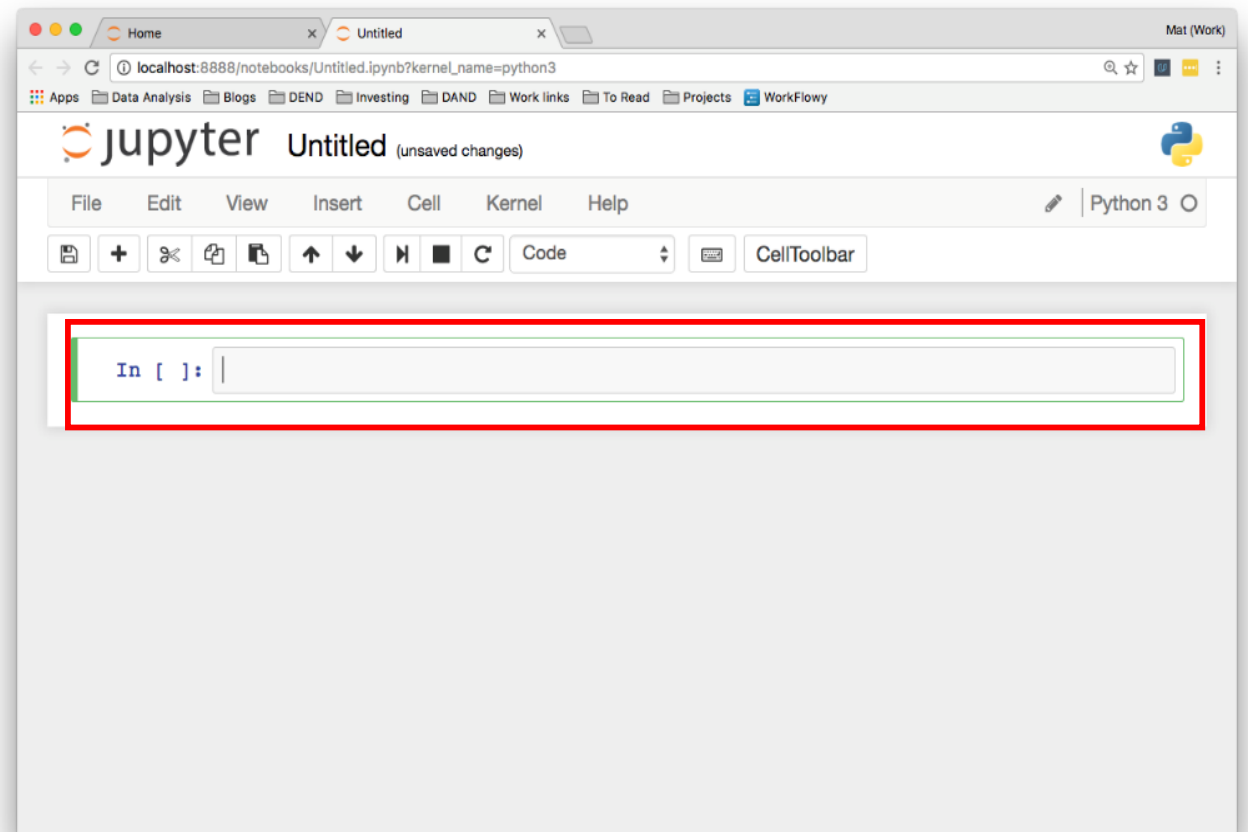
É possível ver uma pequena caixa com um destaque verde. Isso é chamado de célula. Células são onde você escreve e roda seu código. Você também pode modificá-la para que ela seja markdown, um formato popular de escrever conteúdo web. Na barra de ferramentas, clique em "code" para mudar para markdown. O pequeno botão de play roda a célula, e as setas para cima e para baixo movem a célula para cima ou para baixo.



Ao rodar uma célula de código, o output dela é mostrado abaixo da célula. A célula também recebe um número, no caso o In [1];, que aparece à esquerda. Isso permite que você veja que o código foi rodado em ordem, caso você rode várias células. Rodar uma célula no modo markdown carregará o markdown como texto.

# Células de código

A maioria do seu trabalho nos notebooks será feita em células de código. É nelas que você escreve e executa o código. Nessas células, é possível escrever qualquer tipo de código, declarando variáveis, definindo funções e classes, importando pacotes e muito mais. Qualquer código executado em uma célula fica disponível para todas as outras.



# Células Markdown

As células também podem ser usadas para texto escrito em markdown. Markdown é uma sintaxe de formatação que permite a inclusão de links, textos estilizados como negrito ou itálico, assim como código formatado. Assim como nas células de código, ao apertar **Shift + Enter** ou **Control + Enter** para rodar a célula de markdown, o lugar onde ela está carregará o Markdown como texto formatado.

É possível escrever cabeçalhos usando o símbolo jogo da velha **#** antes do texto. Um **#** gera um cabeçalho h1, dois **#**s geram um h2 e assim por diante.

```
# Cabeçalho 1  
## Cabeçalho 2  
### Cabeçalho 3
```

gera

## Cabeçalho 1

## Cabeçalho 2

### Cabeçalho 3alho

Para mais estilos: <https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet>

# Atalhos de Teclado

**Ctrl-Enter**: executa a célula;

**Shift-Enter**: executa a célula e cria uma nova célula abaixo no modo de comando;

**Alt-Enter**: executa a célula e cria uma nova célula abaixo no modo de edição;

**Y**: muda a célula para o tipo código;

**M**: muda a célula para o tipo markdown;

**A**: insere célula acima;

**B**: insere célula abaixo;

**X**: recorta a célula;

**C**: copia a célula;

**V**: cola a célula do clipboard abaixo;

**shift-V**: cola a célula do clipboard acima;

**D, D**: deleta uma célula;

**Z**: desfaz o apagar célula; e

**L**: ativa a enumeração das linhas.





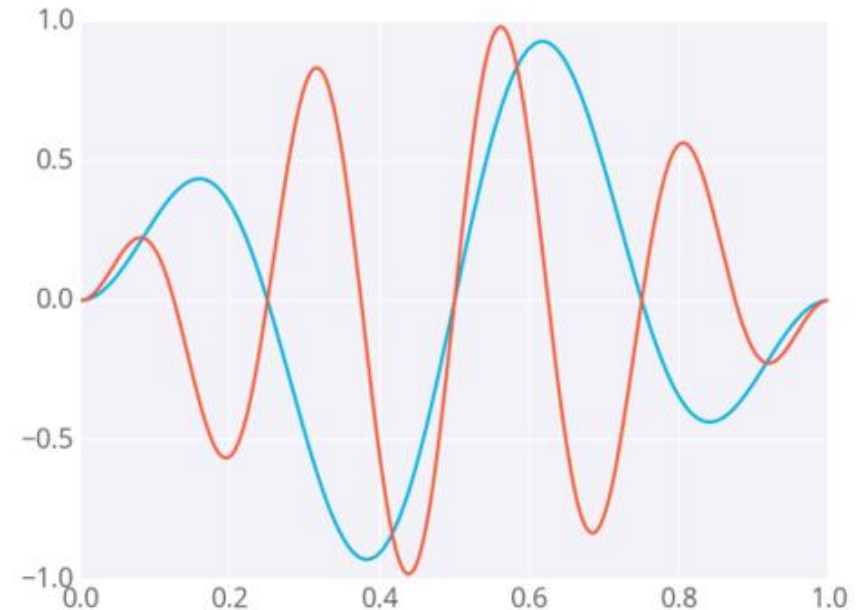
# Embutindo visualizações em notebooks

É possível incluir imagens ao longo do texto e código. Isso é muito útil, especialmente quando se usa o **matplotlib** ou outros pacotes de visualização para criar gráficos e imagens. É possível usar o comando **%matplotlib** para carregar o pacote **matplotlib** de modo interativo no notebook.

```
In [103]: %matplotlib inline
          %config InlineBackend.figure_format = 'retina'

          import matplotlib.pyplot as plt
          import numpy as np
```

```
In [134]: x = np.linspace(0, 1, 300)
          for w in range(2, 6, 2):
              plt.plot(x, np.sin(np.pi*x)*np.sin(2*w*np.pi*x))
```



# Google Colab

O Google Colab ou “Colaboratório” é um serviço de nuvem gratuito hospedado pelo Google para incentivar a pesquisa de Aprendizado de Máquina e Inteligência Artificial, onde muitas vezes a barreira para o aprendizado e o sucesso é a exigência de um tremendo poder computacional.



# Benefícios do Google Colab

Suporte para Python 2.7 e Python 3.6

Aceleração de GPU grátis

Construído com base no Jupyter Notebook

Suporta comandos bash

Recurso de colaboração: o Google Colab permite que os desenvolvedores usem e compartilhem o Jupyter notebook entre si sem precisar baixar, instalar ou executar qualquer coisa que não seja um navegador

Bibliotecas pré-instaladas:  
Todas as principais bibliotecas Python, como o TensorFlow, o Scikit-learn, o Matplotlib, entre muitas outras, estão pré-instaladas e prontas para serem importadas

Os notebooks do Google Colab são armazenados no drive

# Criando um notebook com o Colab

1. Abra o [Google Colab](#);
2. Clique em “**novο notebook**” e selecione o notebook **Python 2** ou o notebook **Python 3**.

**OU**

1. Abra o [Google Drive](#);
2. Crie uma nova pasta para o projeto;
3. Clique em ‘**Novo**’ > ‘**Mais**’> ‘**Colaboratório**’.

# Configurando o acelerador de GPU

1. O hardware padrão do Google Colab é a CPU ou pode ser GPU.
2. Clique em 'Editar' > 'Configurações do notebook' > 'Acelerador de hardware' > 'GPU'.

**OU**

1. Clique em 'Runtime' > 'Hardware Accelerator' > 'GPU'.

# Comandos Bash

Os comandos de *bash* podem ser executados prefixando o comando com “!”.

- ❑ Clonando um repositório git:

```
!git clone [git clone url]
```

- ❑ Comandos de diretório !ls, !mkdir:

```
!ls
```

Este comando gera as pastas / conteúdo e / drive (se ele foi montado).

- ❑ Para alterar a pasta atual execute o trecho a seguir :

```
import sys  
sys.path.append('[nome da pasta]')
```

- ❑ Download da web

```
!wget [url] -p drive/[nome da pasta]
```

# Instalando Bibliotecas

Embora a maioria das bibliotecas Python comumente usadas seja pré-instalada, novas bibliotecas podem ser instaladas usando os pacotes abaixo:

```
!pip install [nome do pacote]
```

**OU**

```
!apt-get install [nome do pacote]
```

A stylized graphic of a brain, split vertically. The left half is purple and the right half is blue. Both halves have a lighter, semi-transparent version of the same color behind them. Grey circuit lines with small circular nodes extend from the top and sides of the brain.

# **OBRIGADA!**

**Repositório GitHub:**

**<https://github.com/Skyzenho/ArtIEEEficiais>**

**ARTIEEEFICIAIS**