

Introdução TensorFlow

Prof. Luiz Bordignon

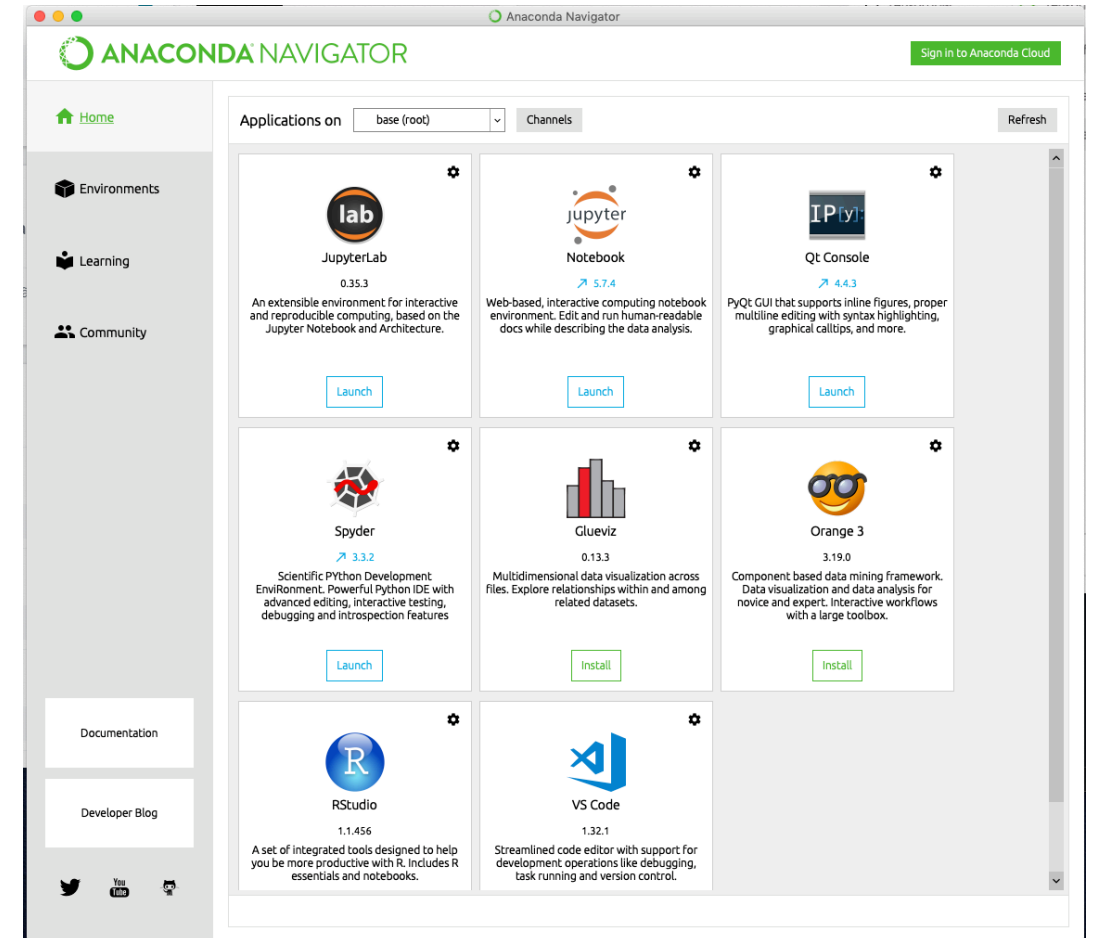
Ferramentas

- Para o desenvolvimento das nossas aplicações, será utilizado o software Anaconda
- <https://www.anaconda.com/distribution/>

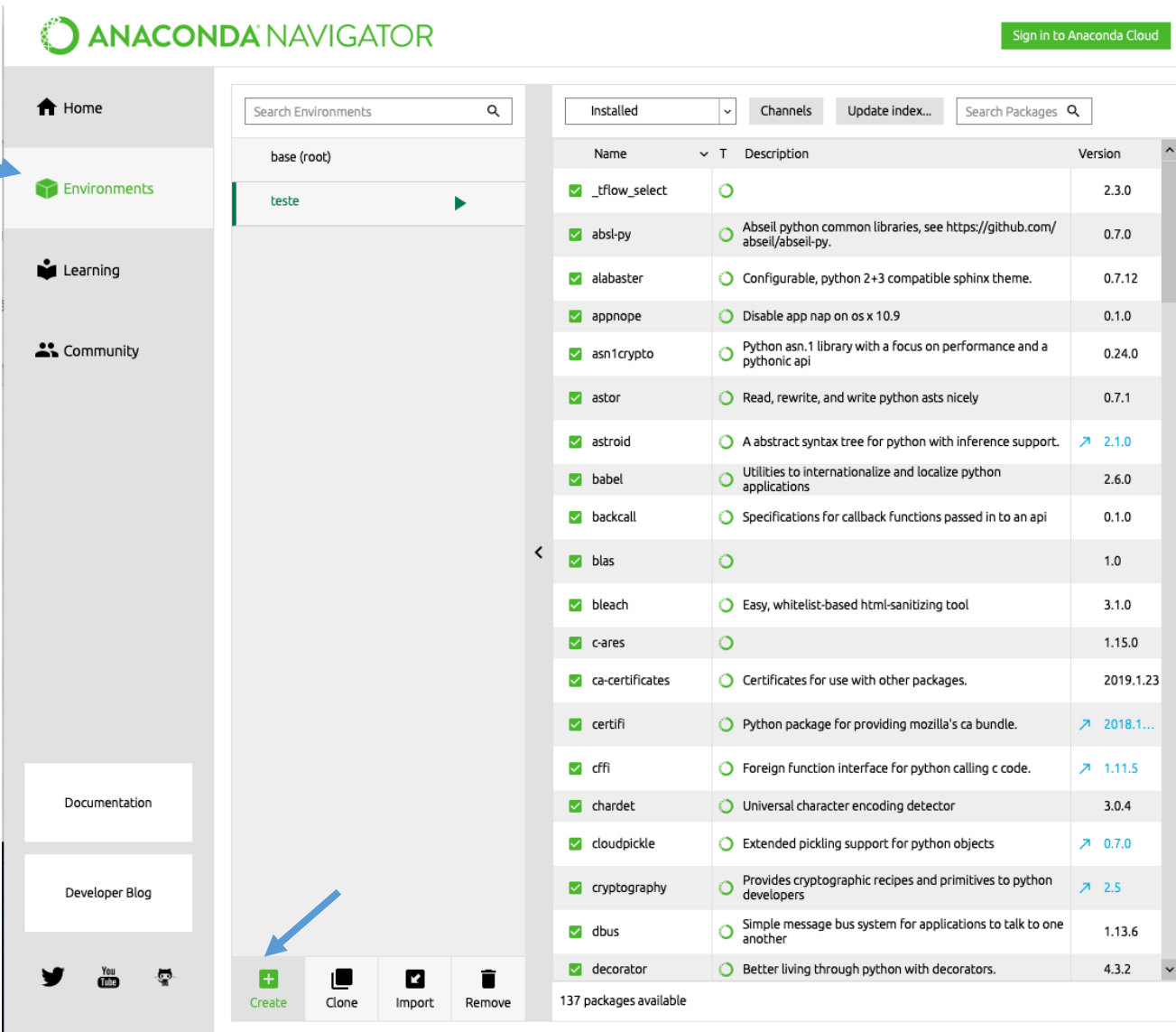


Configuração do ambiente

- Environments.
- Criação de ambientes de trabalho.
- Instalação de bibliotecas.



Criação de um ambiente de trabalho



The screenshot displays the Anaconda Navigator application interface. On the left sidebar, the 'Environments' tab is selected, indicated by a blue arrow. The main panel shows a list of installed packages for the 'base (root)' environment. A blue arrow points to the 'Create' button at the bottom left of the package list.

ANACONDA NAVIGATOR Sign in to Anaconda Cloud

Home Environments Learning Community

Documentation Developer Blog

Search Environments

base (root)

teste

Installed Channels Update index... Search Packages

Name	T	Description	Version
✓ _tflow_select	○		2.3.0
✓ absl-py	○	Abseil python common libraries, see https://github.com/abseil/abseil-py.	0.7.0
✓ alabaster	○	Configurable, python 2+3 compatible sphinx theme.	0.7.12
✓ appnope	○	Disable app nap on os x 10.9	0.1.0
✓ asn1crypto	○	Python asn.1 library with a focus on performance and a pythonic api	0.24.0
✓ astor	○	Read, rewrite, and write python asts nicely	0.7.1
✓ astroid	○	A abstract syntax tree for python with inference support.	2.1.0
✓ babel	○	Utilities to internationalize and localize python applications	2.6.0
✓ backcall	○	Specifications for callback functions passed in to an api	0.1.0
✓ blas	○		1.0
✓ bleach	○	Easy, whitelist-based html-sanitizing tool	3.1.0
✓ c-ares	○		1.15.0
✓ ca-certificates	○	Certificates for use with other packages.	2019.1.23
✓ certifi	○	Python package for providing mozilla's ca bundle.	2018.1...
✓ cffi	○	Foreign function interface for python calling c code.	1.11.5
✓ chardet	○	Universal character encoding detector	3.0.4
✓ cloudpickle	○	Extended pickling support for python objects	0.7.0
✓ cryptography	○	Provides cryptographic recipes and primitives to python developers	2.5
✓ dbus	○	Simple message bus system for applications to talk to one another	1.13.6
✓ decorator	○	Better living through python with decorators.	4.3.2

137 packages available

Create Clone Import Remove

Instalação de pacotes

The screenshot shows the Anaconda environment manager interface. On the left, a sidebar lists environments: 'base (root)' and 'Teste'. The 'Teste' environment is selected. The main panel displays a list of packages available for installation. At the top, there's a search bar and a filter dropdown set to 'Not installed'. A search for 'tensorflow' has been performed, and the results are shown in a table. The 'tensorflow' package is highlighted, and a blue arrow points to the 'Apply' button at the bottom right.

Search Environments

base (root)

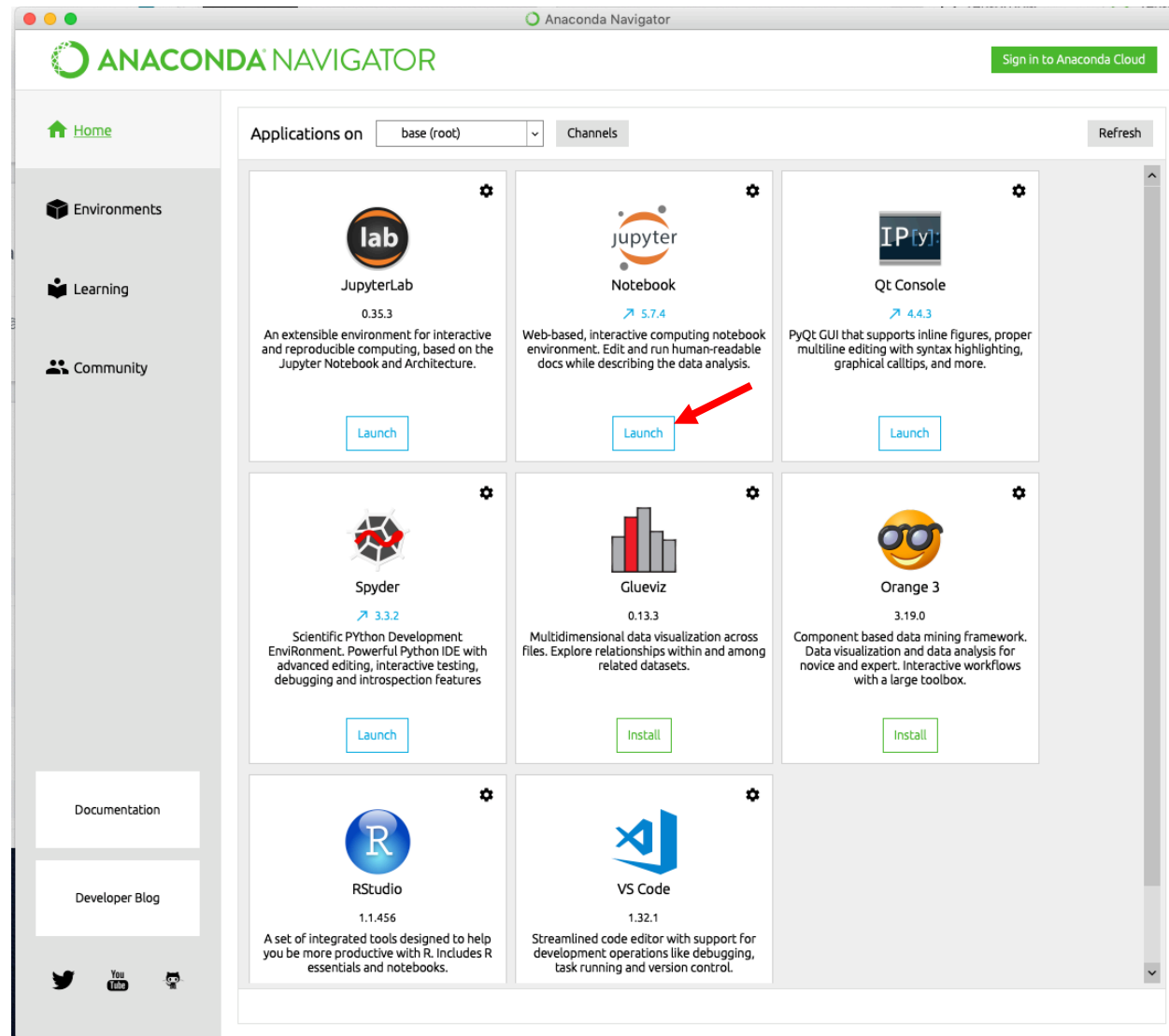
Teste

Not installed Channels Update index... tensorflow X

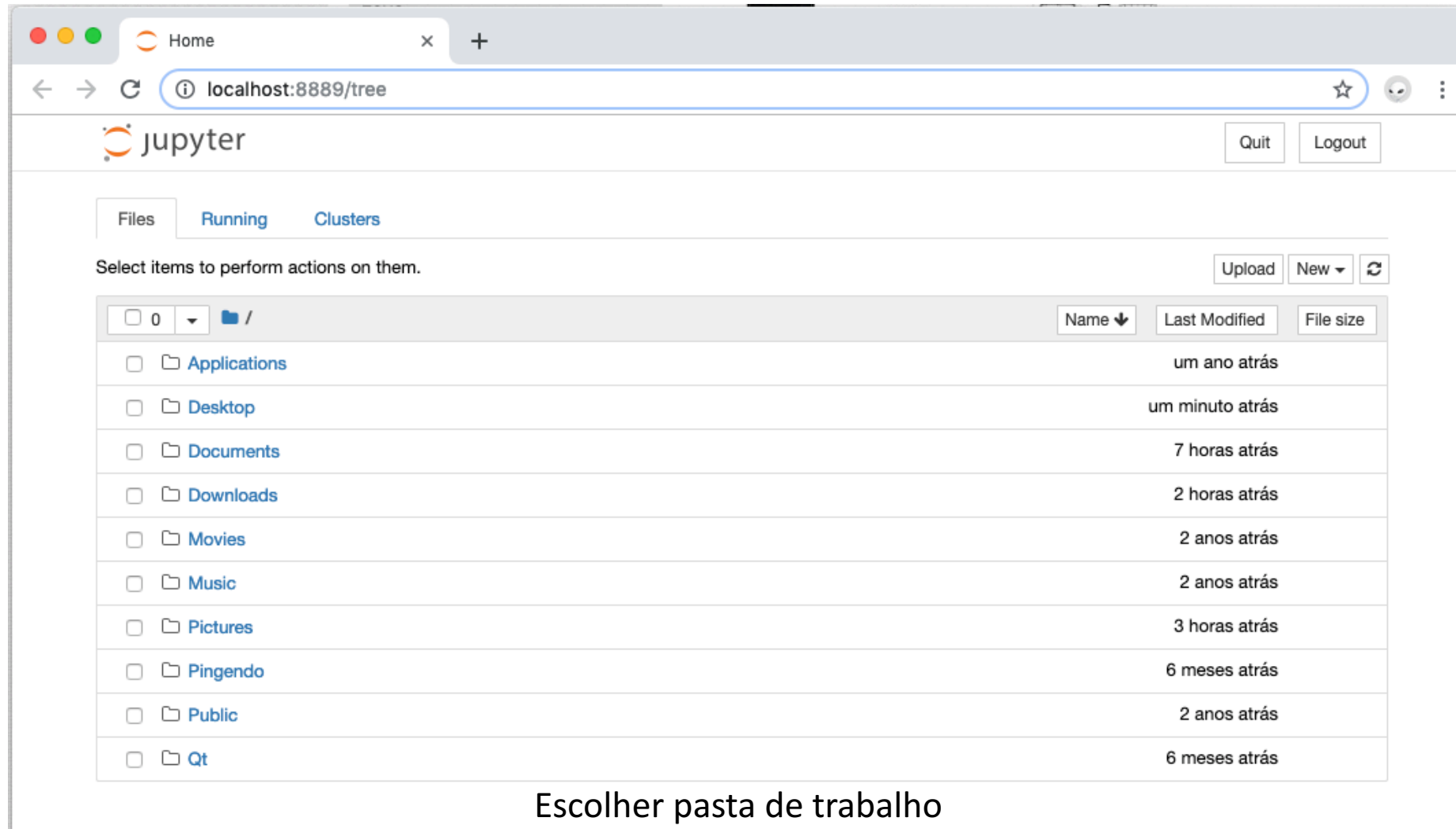
Name	T	Description	Version
<input type="checkbox"/> keras	○	Deep learning library for theano and tensorflow	2.2.4
<input type="checkbox"/> keras-gpu	○	Deep learning library for theano and tensorflow	2.2.2
<input type="checkbox"/> r-tensorflow	○		1.8
<input checked="" type="checkbox"/> tensorflow	○	Tensorflow is a machine learning library.	1.9.0
<input type="checkbox"/> tensorflow-base	○	Tensorflow is a machine learning library, base package contains only tensorflow.	1.9.0
<input type="checkbox"/> tensorflow-eigen	○	Metapackage for selecting a tensorflow variant.	1.9.0
<input type="checkbox"/> tensorflow-mkl	○	Metapackage for selecting a tensorflow variant.	1.9.0

Apply Clear

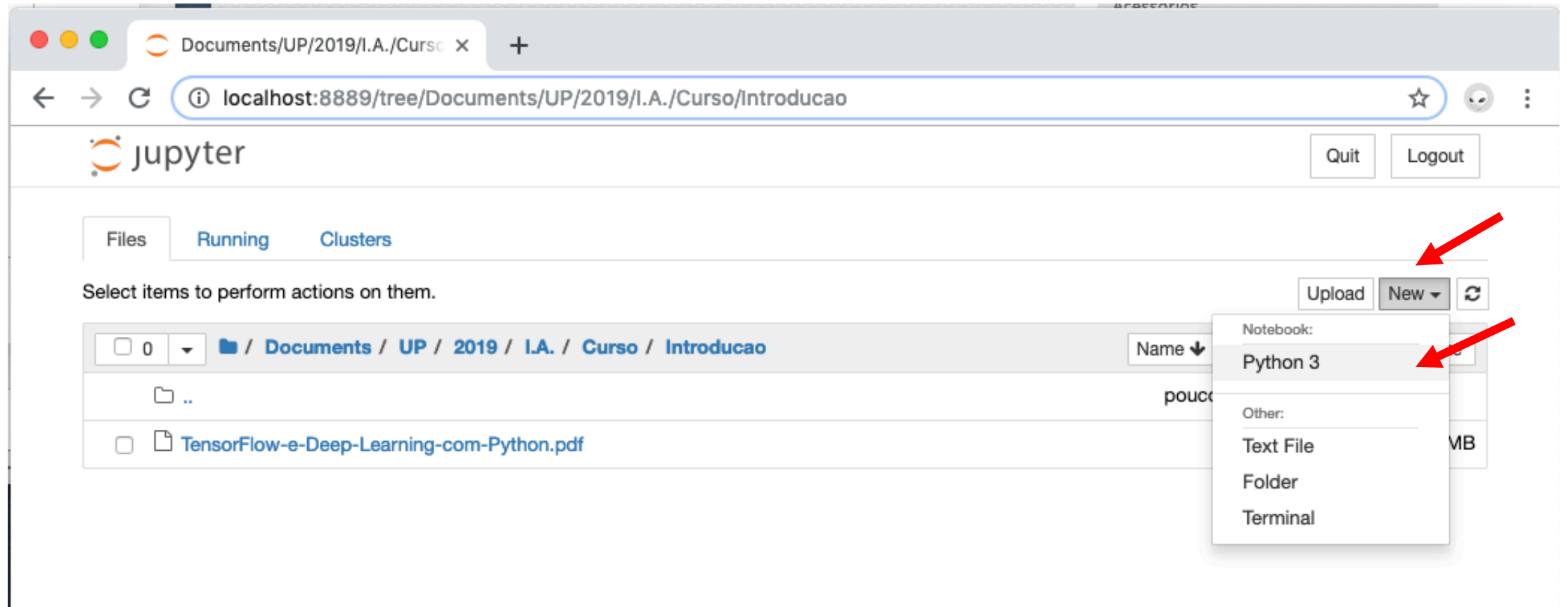
Instalação da IDE Jupyter Notebook



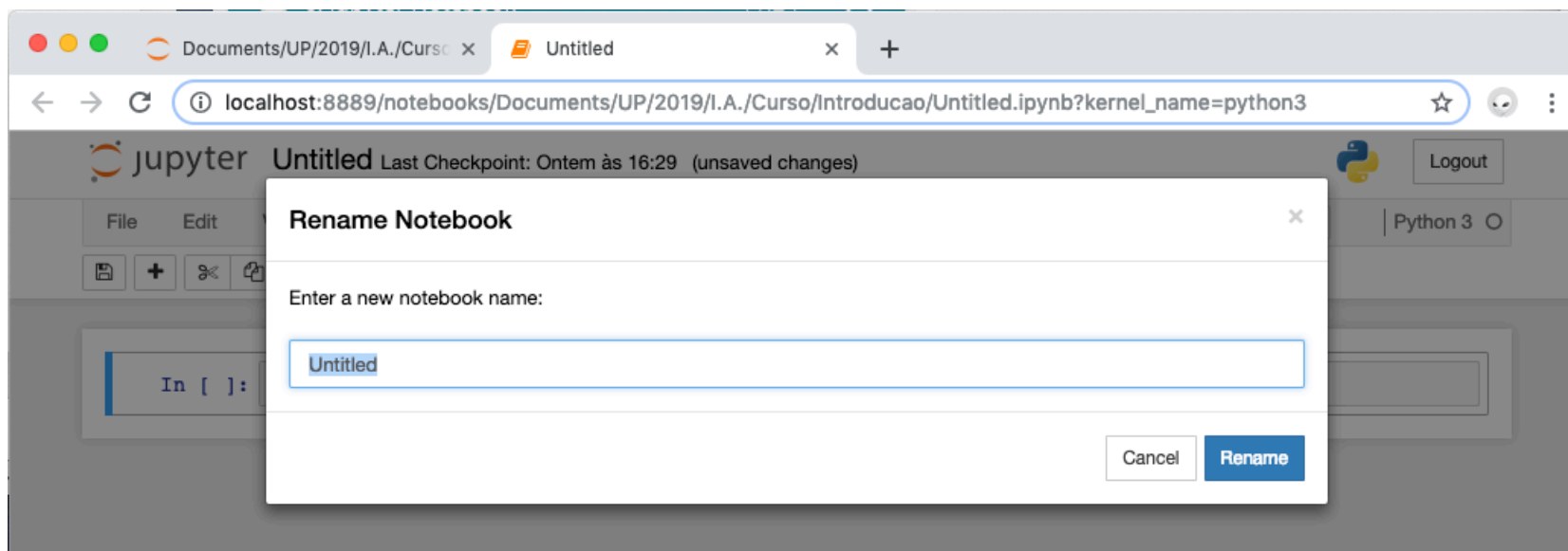
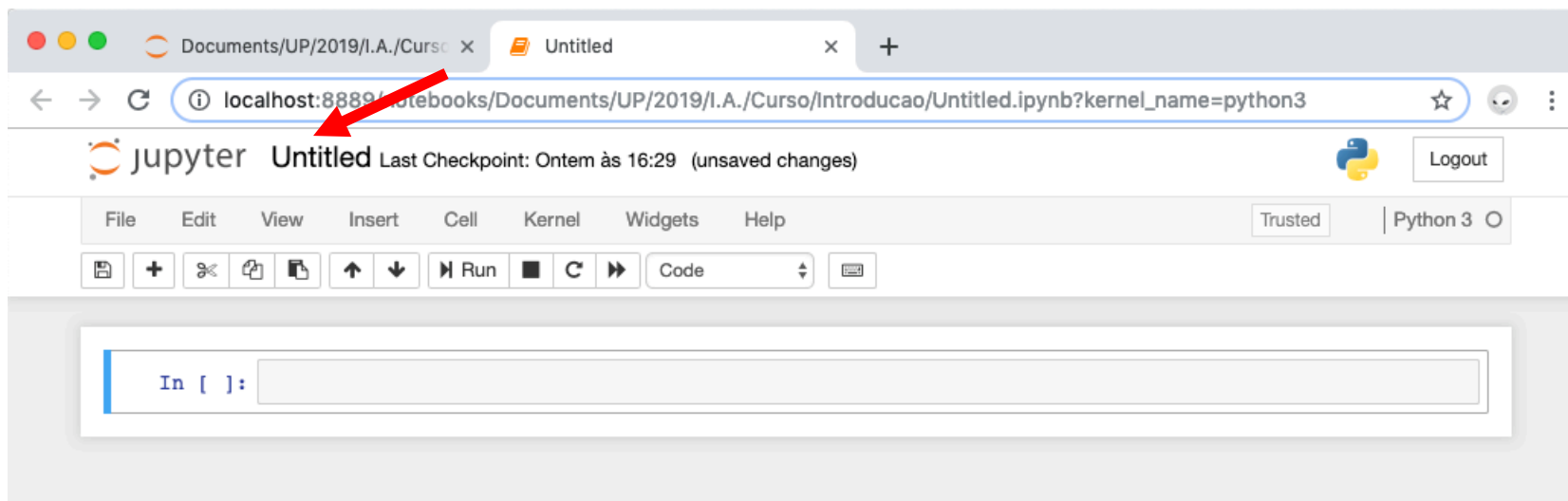
Iniciando com Jupyter Notebook



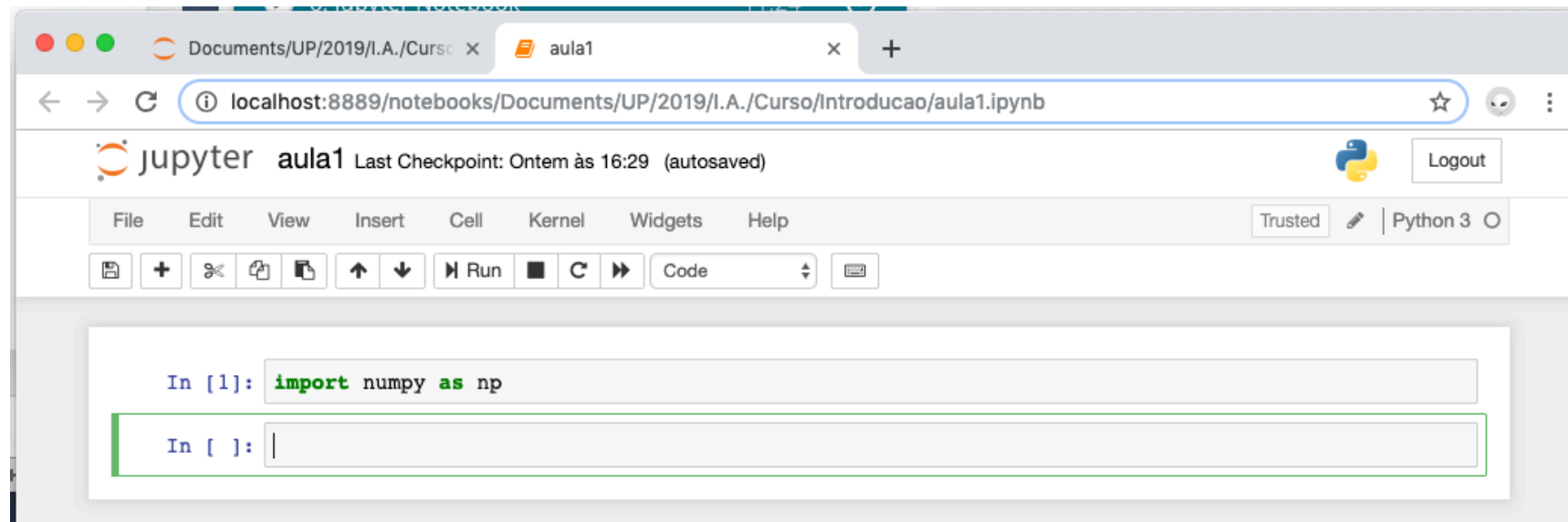
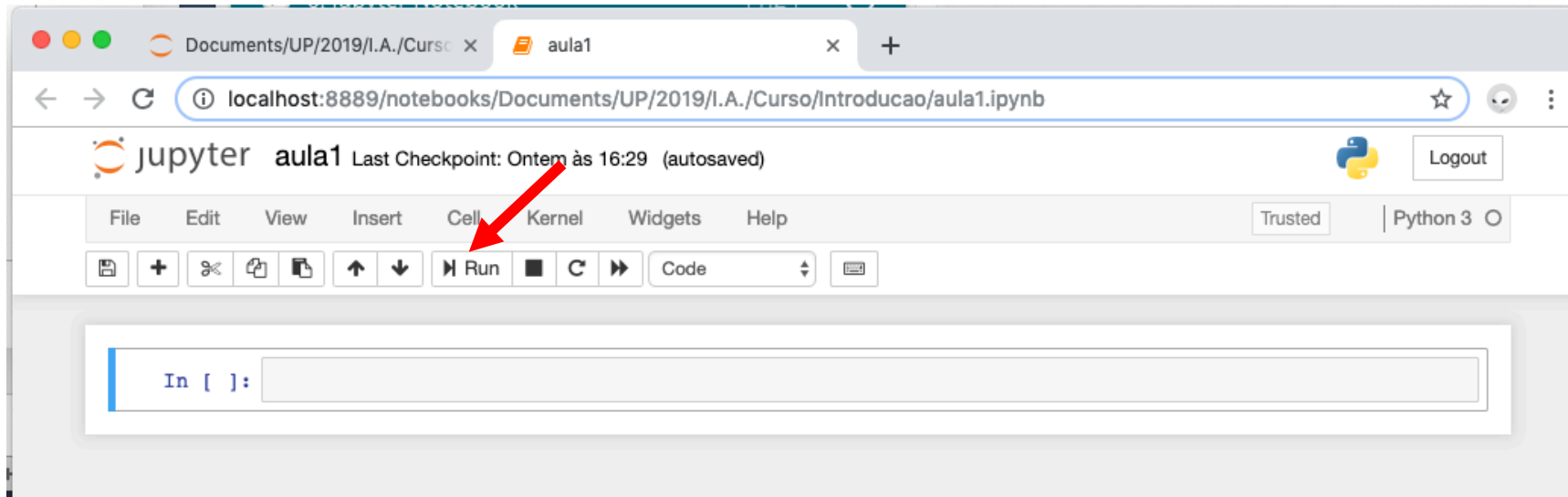
Criar um arquivo de trabalho



Renomear o arquivo



Executando comandos



Criando um variável

- Vamos criar um variável a = [2, 3, 4]

```
In [1]: import numpy as np
```

```
In [3]: a = [1, 2, 3]  
a
```

```
Out[3]: [1, 2, 3]
```

O atalho para a execução é o ctrl+enter

Criando um array

- Vamos criar um array utilizando recursos da lib numpy, para isso precisamos informar o valor inicial, valor final e intervalo que queremos para nosso array, exemplo:
- Criar um vetor com 10 posições iniciando em 1 e finalizando em 10.

```
In [5]: array = np.linspace(1, 10, 10)
```

```
In [6]: array
```

```
Out[6]: array([ 1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.])
```

Atalho: para verificar as opções de um determinado comando, basta clicar em SHIFT+TAB

Comando reshape()

- Um comando bastante útil para trabalhar com RNA's, tem como característica mudar a dimensionalidade de uma variável, exemplo, vamos alterar o vetor criado anteriormente para uma matriz 2x5.

```
In [8]: array = array.reshape(2,5)
```

```
In [9]: array
```

```
Out[9]: array([[ 1.,  2.,  3.,  4.,  5.],  
               [ 6.,  7.,  8.,  9., 10.]])
```

Atalho: para executar e na sequência já criar um nova linha SHIFT+ENTER

Pegar elemento da matriz

- Para ler um determinado elemento de uma matriz basta chamar a posição que ele ocupa na mesma, lembrando que linha e coluna se iniciam em 0. Exemplo, pegar o valor 4 da matriz criada anteriormente:

```
In [10]: array[0,3]
```

```
Out[10]: 4.0
```

Pegar elemento da matriz

- Para ler um conjunto de elementos de uma matriz podemos utilizar a seguinte sintaxe:
 - Exemplo, ler todas as linhas da matriz, porém só as duas primeiras colunas:

```
In [11]: array[:,0:2]
```

```
Out[11]: array([[1., 2.],  
                [6., 7.]])
```

Gerando um gráfico com a lib matplotlib

- Primeiramente vamos importar a lib:

```
In [ ]: import matplotlib.pyplot as plt
```

- Depois vamos criar os eixos do nosso gráfico:

```
In [14]: x = np.arange(1,11)  
x
```

```
Out[14]: array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

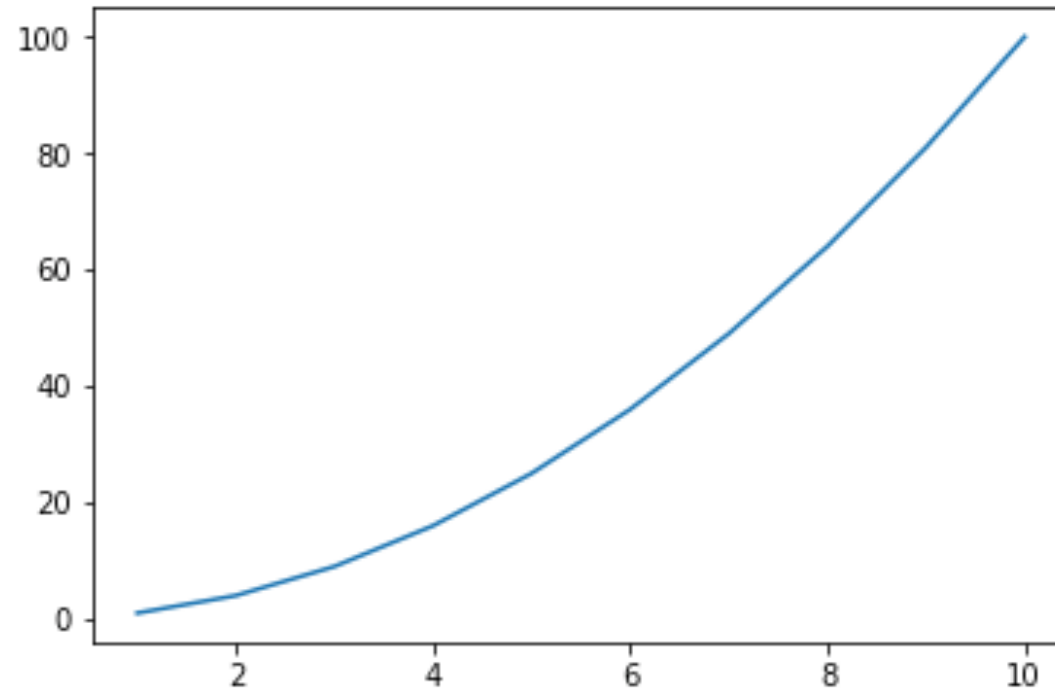
```
In [15]: y = x**2  
y
```

```
Out[15]: array([ 1,  4,  9, 16, 25, 36, 49, 64, 81, 100])
```


Gerando um gráfico com a lib matplotlib

```
In [16]: import matplotlib.pyplot as plt  
%matplotlib inline  
plt.plot(x,y)
```

```
Out[16]: [<matplotlib.lines.Line2D at 0x11d48ec50>]
```



TensorFlow

- Biblioteca de código aberto para computação numérica
- Desenvolvido pelos engenheiros e pesquisadores do Google Brain
- Inclui o XLA que é um compilador de álgebra linear para execução em CPUs, GPUs e TPUs
- Considerada a biblioteca mais eficiente para Deep Learning



Algumas aplicações

- Sistemas de tradução
- Entendimento do contexto de mensagens do Gmail
- Diagnóstico de diabetes
- Geração de músicas e filmes
- Descoberta de cura para doenças
- Carros autônomos

Documentação da API

- <https://www.tensorflow.org/guide>

Conceito de tensor

- Da algebra...

- Escalar

5

- Vetor

[1, 2, 3]

- Matriz

[1, 2, 3]

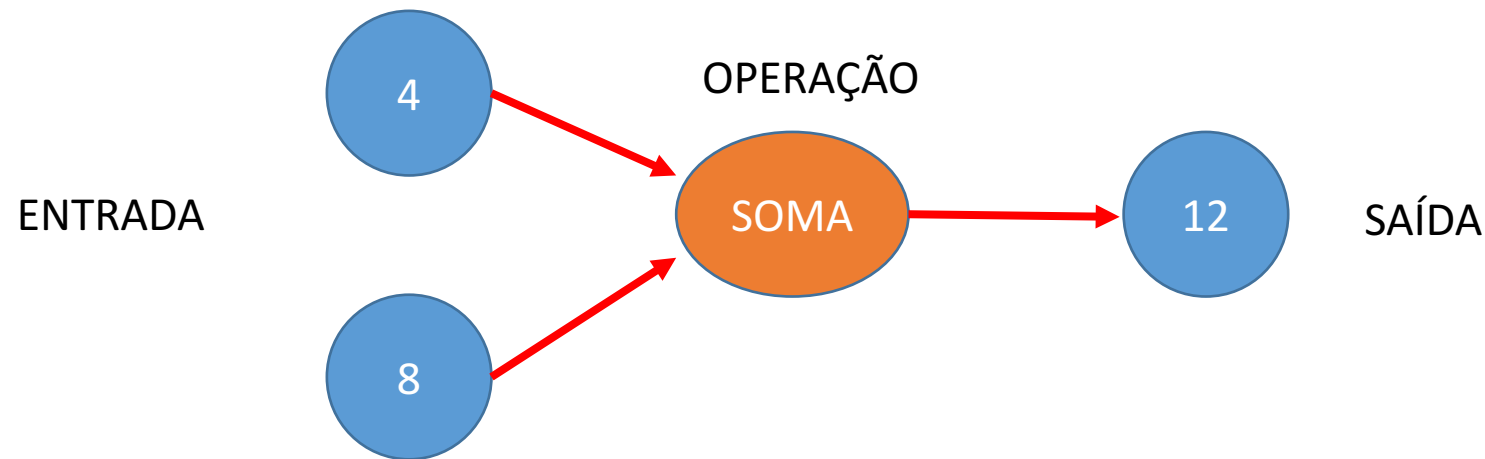
[4, 5, 6]

- Tensor

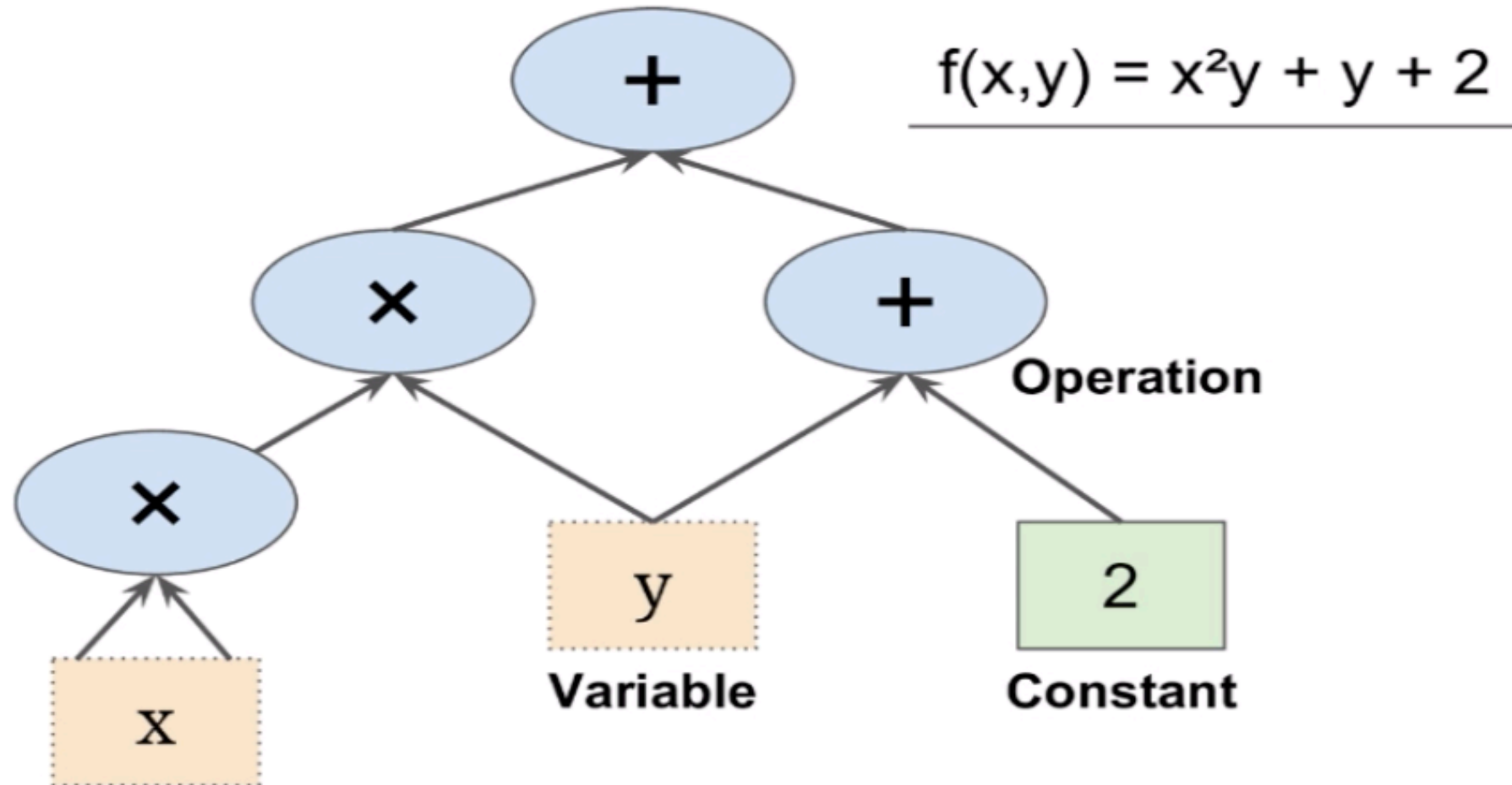
Um escalar, um vetor, uma matriz bidimensional, uma matriz 3D (ou de maior dimensão)

Grafos

- TensorFlow é baseado em estrutura de grafos, as operações não são realizadas conforme são definidas por código e sim, apenas quando são executados os grafos formados por essas operações.



Exemplo de operação



Importando a lib TensorFlow

- Inicialmente precisamos importar a lib no nosso ambiente, para isso:

```
import tensorflow as tf
```

Lembrando que o módulo deverá estar instalado no ambiente de desenvolvimento.

Operações

- Diferentemente de outras linguagens de programação, o tensorflow não retorna diretamente o valor de uma determinada operação.
- Trabalhando com o modelo de grafos, ele primeiro cria a operação, mas não há execução com valores.
- Para que possamos executar uma determinada operação, é necessário executar uma sessão.

Operações

- Para ficar mais claro, vamos criar duas constantes, e uma operação de soma entre elas.

```
In [2]: valor1 = tf.constant(2)
```

```
In [3]: valor2 = tf.constant(3)
```

```
In [4]: type(valor1)
```

```
Out[4]: tensorflow.python.framework.ops.Tensor
```

```
In [5]: soma = valor1 + valor2
```

```
In [6]: print(soma)
```

```
Tensor("add:0", shape=(), dtype=int32)
```

Operações

- Agora que temos uma operação criada, podemos criar e executar uma sessão, nesta sessão o grafo correspondente a essa operação será executado.

```
In [7]: with tf.Session() as sess:  
        s = sess.run(soma)
```

```
In [9]: print(s)
```

5

Tudo que é criado fora de uma sessão gera um grafo, e não é necessariamente executado.

Variáveis

- Seguindo o modelo de tensores, o TensorFlow trata variáveis de forma um pouco diferente de como estamos acostumados.
- Para que possamos usar as variáveis em nosso código precisamos inicializa-las. Isso não é o mesmo que atribuir um valor.
- No TensorFlow ao inicializar uma variável, dizemos ao compilador que o grafo deverá ser executado e o tensor correspondente desta variável poderá fluir por esse grafo.

Variáveis exemplos

Sintaxe Python

```
In [10]: x = 35  
        y = x + 20
```

```
In [11]: print(y)
```

55

Sintaxe TensorFlow

```
In [12]: valor1 = tf.constant(15, name = 'valor1')  
        print(valor1)
```

Tensor("valor1:0", shape=(), dtype=int32)

```
In [13]: soma = tf.Variable(valor1 + 5, name = 'valor_soma')  
        print(soma)
```

<tf.Variable 'valor_soma:0' shape=() dtype=int32_ref>

Inicializando a variável

- Para conseguir usar a variável em uma determinada operação, precisamos inicializa-la:

computa as dependências entre as variáveis

```
In [14]: init = tf.global_variables_initializer()
```

```
In [15]: with tf.Session() as sess:  
         sess.run(init)  
         s = sess.run(soma)
```

```
In [16]: print(s)
```

20

Variáveis - vetor

- Lembrando que a codificação é muito parecida com Python normal, no TensorFlow temos apenas que inicializar as variáveis.

```
vetor = tf.constant([5,10,14], name = 'vetor')  
print(vetor)
```

```
Tensor("vetor:0", shape=(3,), dtype=int32)
```

```
soma = tf.Variable(vetor + 5, name = 'soma')  
init = tf.global_variables_initializer()
```

```
with tf.Session() as sess:  
    sess.run(init)  
    print(sess.run(soma))
```

```
[10 15 19]
```

Variáveis - for

- Vamos montar um contador simples utilizando um for.

```
contador = tf.Variable(0, name = 'contador')  
init = tf.global_variables_initializer()
```

```
with tf.Session() as sess:  
    sess.run(init)  
    for i in range(5):  
        contador = contador + 1  
        print(sess.run(contador))
```

1
2
3
4
5

Adição vetores e matrizes

- Funções que são muito importantes para trabalhar com redes neurais.
- Inicialmente vamos fazer a soma de dois vetores:

```
a = tf.constant([9, 8, 7], name = 'a')  
b = tf.constant([1, 2, 3], name = 'b')
```

```
soma = a + b
```

```
type(a)
```

```
tensorflow.python.framework.ops.Tensor
```

```
print(a)
```

```
Tensor("a_1:0", shape=(3,), dtype=int32)
```

```
with tf.Session() as sess:  
    print(sess.run(soma))
```

```
[10 10 10]
```

Vetores de mesma ordem

Matriz

- Para criar uma matriz basta adicionar uma dimensão no exemplo do vetor:

```
a1 = tf.constant([[1, 2, 3], [4, 5, 6]], name = 'a1')
```

```
type(a1)
```

```
tensorflow.python.framework.ops.Tensor
```

```
print(a1)
```

```
Tensor("a1_1:0", shape=(2, 3), dtype=int32)
```

```
a1.shape
```

```
TensorShape([Dimension(2), Dimension(3)])
```

Matriz - soma

```
b1 = tf.constant([[1, 2, 3], [4, 5, 6]], name = 'b1')
```

```
somal = tf.add(a1, b1)
```

```
with tf.Session() as sess:  
    print(sess.run(a1))  
    print('\n')  
    print(sess.run(b1))  
    print('\n')  
    print(sess.run(somal))
```

```
[[1 2 3]  
 [4 5 6]]
```

```
[[1 2 3]  
 [4 5 6]]
```

```
[[ 2  4  6]  
 [ 8 10 12]]
```

shift + tab expande uma função

Matriz - soma

```
a2 = tf.constant([[1,2,3], [4,5,6]])  
b2 = tf.constant([[1], [2]])
```

```
with tf.Session() as sess:  
    print(sess.run(a2))  
    print('\n')  
    print(sess.run(b2))  
    print('\n')
```

```
[[1 2 3]  
 [4 5 6]]
```

```
[[1]  
 [2]]
```

soma ?

Matriz - soma

```
a2 = tf.constant([[1,2,3], [4,5,6]])  
b2 = tf.constant([[1], [2]])  
soma2 = tf.add(a2, b2)
```

```
with tf.Session() as sess:  
    print(sess.run(a2))  
    print('\n')  
    print(sess.run(b2))  
    print('\n')  
    print(sess.run(soma2))
```

```
[[1 2 3]  
 [4 5 6]]
```

```
[[1]  
 [2]]
```

```
[[2 3 4]  
 [6 7 8]]
```

Sim!

Matriz – multiplicação

- Lembrando que multiplicação de matrizes, segue as regras da álgebra linear.

```
import tensorflow as tf
```

```
a = tf.constant([[1, 2], [3, 4]])  
b = tf.constant([[-1, 3], [4, 2]])
```

```
multiplicacao = tf.matmul(a, b)
```

Para usar as regras corretamente, usaremos a função **matmul**, e não o operador `*` como fizemos na soma.

Matriz – multiplicação

- Criando uma sessão e executando os cálculos:

```
with tf.Session() as sess:  
    print(sess.run(a))  
    print('\n')  
    print(sess.run(b))  
    print('\n')  
    print(sess.run(multiplicacao))
```

```
[[1 2]  
 [3 4]]
```

```
[[ -1  3]  
 [ 4  2]]
```

```
[[ 7  7]  
 [13 17]]
```

Relembrando....

- 1ª linha e 1ª coluna

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} -1 & 3 \\ 4 & 2 \end{bmatrix} = \begin{bmatrix} \boxed{1 \cdot (-1) + 2 \cdot 4} & \\ & \end{bmatrix} \quad c_{11}$$

- 1ª linha e 2ª coluna

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} -1 & 3 \\ 4 & 2 \end{bmatrix} = \begin{bmatrix} 1 \cdot (-1) + 2 \cdot 4 & \boxed{1 \cdot 3 + 2 \cdot 2} \\ & \end{bmatrix} \quad c_{12}$$

- 2ª linha e 1ª coluna

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} -1 & 3 \\ 4 & 2 \end{bmatrix} = \begin{bmatrix} 1 \cdot (-1) + 2 \cdot 4 & 1 \cdot 3 + 2 \cdot 2 \\ \boxed{3 \cdot (-1) + 4 \cdot 4} & \end{bmatrix} \quad c_{21}$$

- 2ª linha e 2ª coluna

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} -1 & 3 \\ 4 & 2 \end{bmatrix} = \begin{bmatrix} 1 \cdot (-1) + 2 \cdot 4 & 1 \cdot 3 + 2 \cdot 2 \\ 3 \cdot (-1) + 4 \cdot 4 & \boxed{3 \cdot 3 + 4 \cdot 2} \end{bmatrix} \quad c_{22}$$

$$\text{Assim, } A \cdot B = \begin{bmatrix} 7 & 7 \\ 13 & 17 \end{bmatrix}.$$

Atividades

- Utilizando os conceitos visto na aula, resolva os seguintes exercícios:
 - Crie um vetor de 100 posições com intervalos de 0.5;
 - Faça a soma dos vetor $a = [12, 4, 46, 76, 18]$ e $b = [-10, 21, 0, 19, 99]$;
 - Faça a multiplicação das matrizes $A = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}$ e $B = \begin{pmatrix} 4 & -1 \\ 5 & 0 \end{pmatrix}$. E também $B \times A$;
 - Crie um vetor de 20 posições e transforme-o em uma matriz 4x5;
 - Crie duas matrizes de 10x10 e faça sua multiplicação;

Atividade próxima aula....

- Pesquisar em **artigos científicos** aplicações para IA.
- Escolher **um** artigo para apresentar um resumo sobre as **técnicas utilizadas, contexto do problema e resultados apresentados**.
- Pesquisa e apresentação em até 3 alunos.
- Apresentar na próxima aula (17/03/2020).
- Vale nota.