

COLAB

INTRODUÇÃO

Introdução

- Google tem investido bastante em pesquisas na área de Inteligência Artificial (IA).
- Ferramentas:
 - TensorFlow.
 - Colaboratory.
 - Ambas são ferramentas código-aberto (*open source*).

O que é o COLAB?

- É um ambiente aberto de desenvolvimento de notebook Jupyter que roda totalmente em nuvem.
- Não demanda uma configuração.
- Os notebooks criados podem ser simultaneamente editados por todos os membros da equipe.
- Suporta as principais bibliotecas de aprendizagem de máquina (*machine learning*).

Principais Recursos

- Escrever e executar código em Python.
- Documentar o seu código para suportar equações matemáticas.
- Criar, enviar e compartilhar notebooks.
- Importar e salvar notebooks do/para o Google Drive.
- Importar e publicar notebooks do GitHub.
- Importar external datasets (ex: Kaggle).
- Integrar código do PyTorch, TensorFlow, Keras e OpenCV.
- Serviço de nuvem gratuito com GPU.

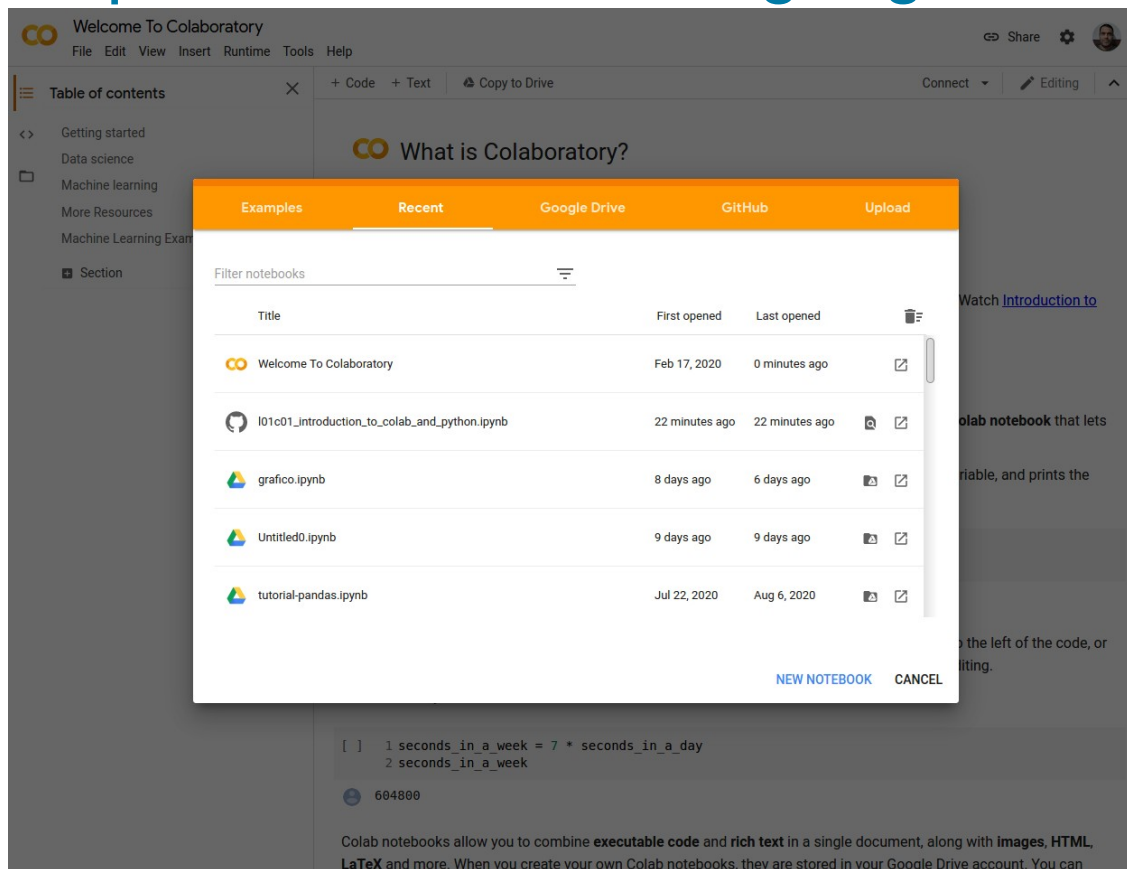
PRIMEIROS PASSOS

Primeiro Notebook

- O usuário precisa estar conectado ao Google.

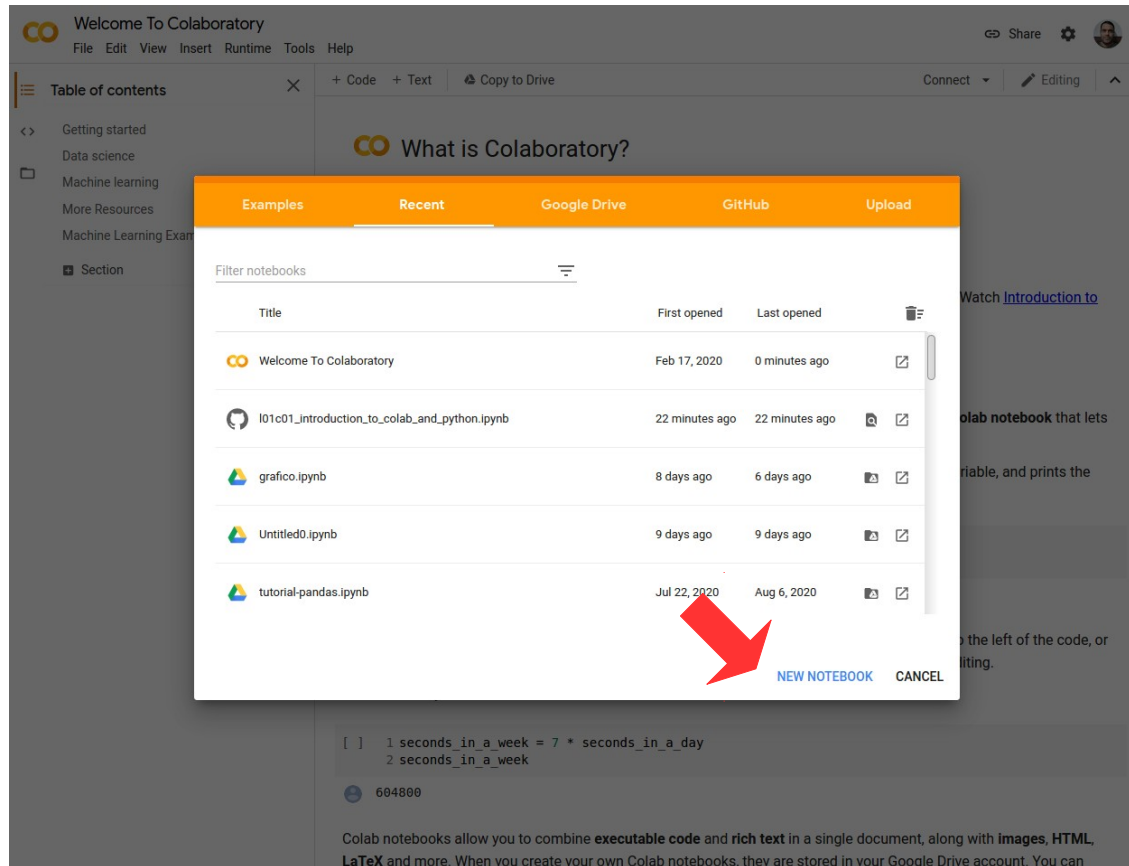
Primeiro Notebook

- **Passo 1:** o usuário deve abrir a URL abaixo no navegador.
 - <https://colab.research.google.com>



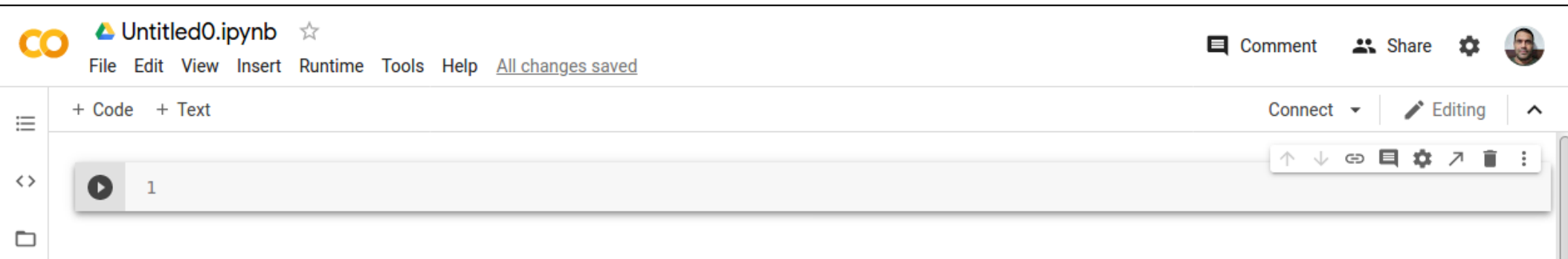
Primeiro Notebook

- **Passo 2:** Clique em New Notebook.



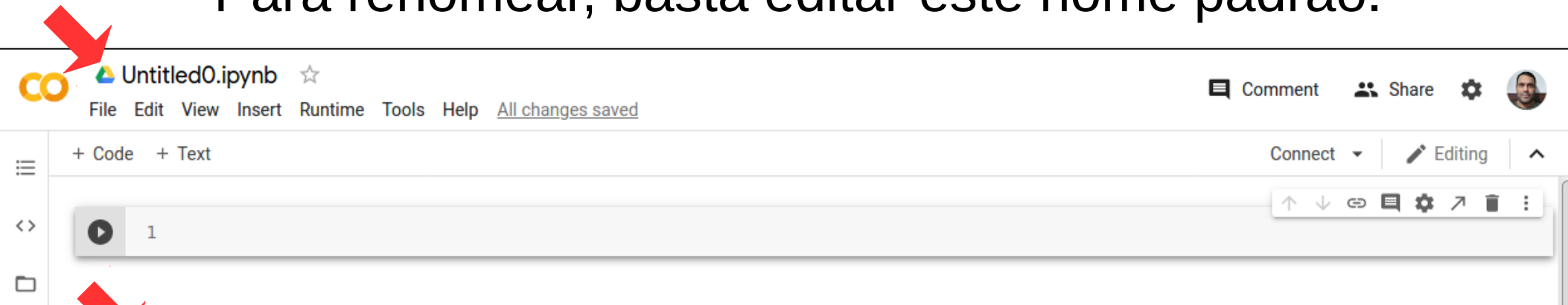
Primeiro Notebook

- **Passo 3:** A interface estará pronta para codificação.

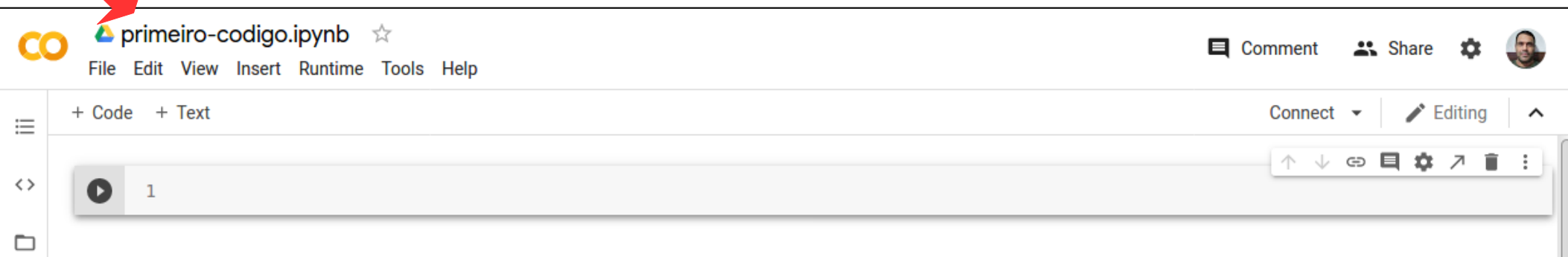


Primeiro Notebook

- **Passo 4:**
 - Por convenção, o notebook terá o nome de Untitled0.ipynb.
 - Para renomear, basta editar este nome padrão.



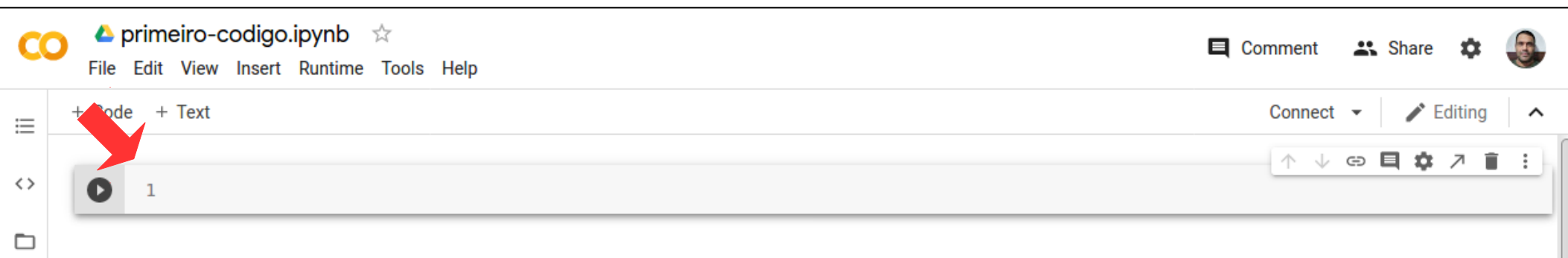
This screenshot shows the top of a Jupyter Notebook interface. The browser tab title is 'Untitled0.ipynb'. The top navigation bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help', with a status message 'All changes saved'. On the right, there are icons for 'Comment', 'Share', and a user profile. Below the navigation bar, there are buttons for '+ Code' and '+ Text'. The main area shows a code cell with a play button icon and the number '1'. A red arrow points to the 'Untitled0.ipynb' text in the browser tab.



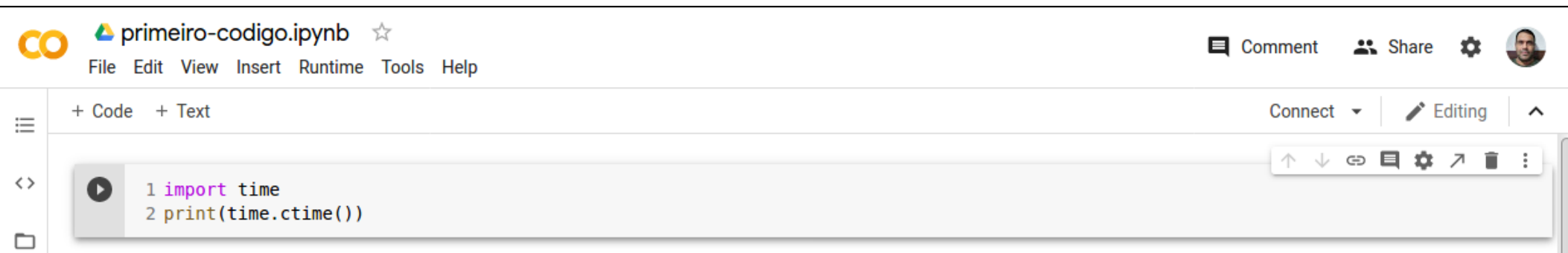
This screenshot shows the top of a Jupyter Notebook interface after renaming. The browser tab title is 'primeiro-codigo.ipynb'. The top navigation bar and right-side icons are identical to the previous screenshot. The main area shows a code cell with a play button icon and the number '1'. A red arrow points to the 'primeiro-codigo.ipynb' text in the browser tab.

Codificação

- Segue abaixo um código trivial em Python.



The screenshot shows the top part of a Jupyter Notebook interface. The title bar reads "primeiro-codigo.ipynb" with a star icon. Below the title bar is a menu: "File Edit View Insert Runtime Tools Help". On the right side, there are icons for "Comment", "Share", and a user profile. The main area shows a code cell with a play button icon on the left and the number "1" in the center. A red arrow points to the play button. To the right of the code cell, there are icons for "Connect", "Editing", and a list of actions (up, down, link, comment, settings, refresh, trash, and a vertical ellipsis).



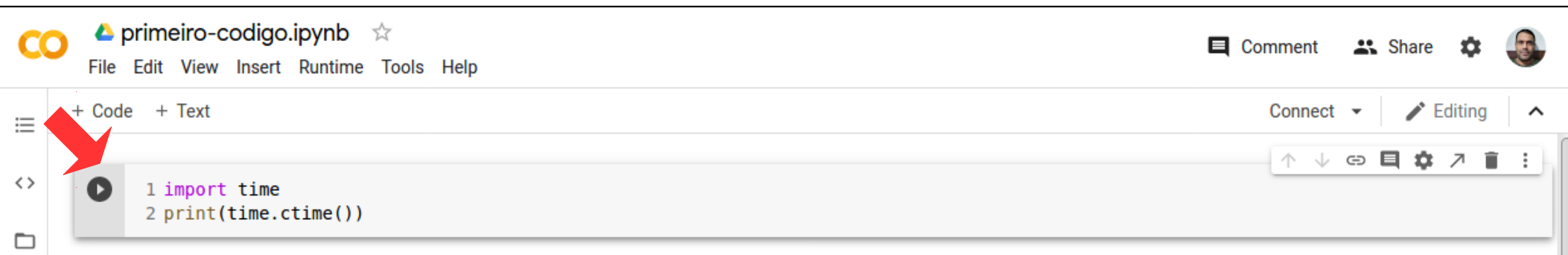
The screenshot shows the same Jupyter Notebook interface as above, but with two lines of Python code in the code cell:

```
1 import time
2 print(time.ctime())
```

The rest of the interface, including the title bar, menu, and right-side icons, is identical to the previous screenshot.

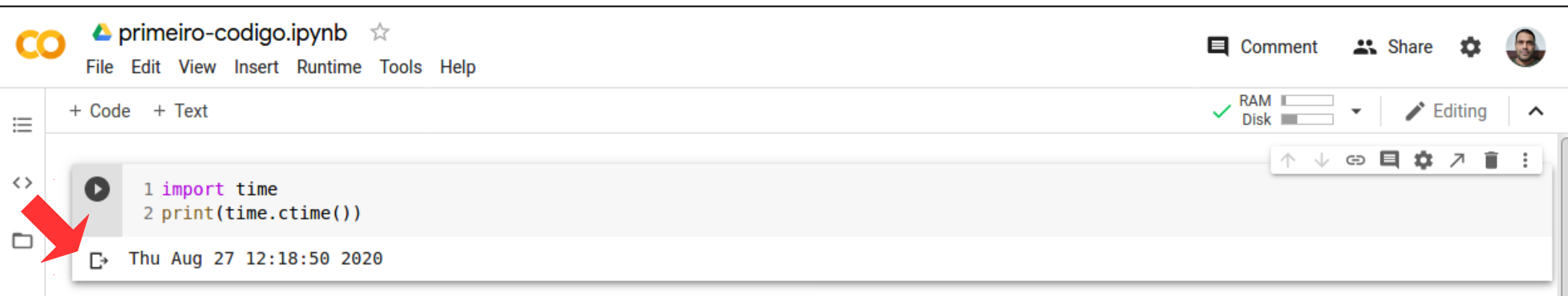
Execução

- Para executar o código, clique na seta à esquerda da janela do código.



The screenshot shows the top part of a Jupyter Notebook interface. The title bar reads "primeiro-codigo.ipynb" with a star icon. Below it is a menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". On the right, there are icons for "Comment", "Share", and a user profile. The main area is a code editor with a toolbar on the left containing "+ Code" and "+ Text". A red arrow points to a play button icon (a right-pointing triangle inside a circle) located to the left of the code. The code in the editor is:

```
1 import time
2 print(time.ctime())
```



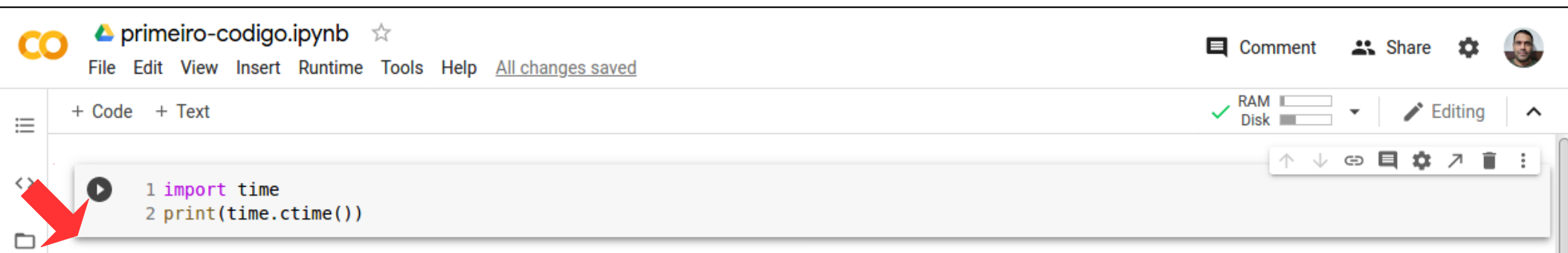
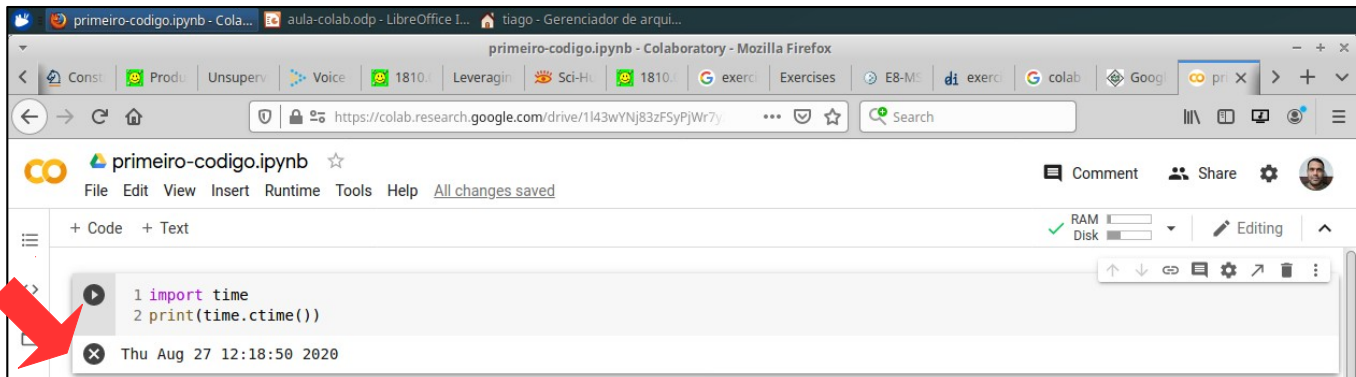
The screenshot shows the same Jupyter Notebook interface as above, but now the code has been executed. The code editor still contains the same code:

```
1 import time
2 print(time.ctime())
```

A red arrow points to the play button icon. Below the code editor, the output area is visible, showing the execution time: "Thu Aug 27 12:18:50 2020". The toolbar on the right now includes a "RAM Disk" indicator with a green checkmark and a progress bar.

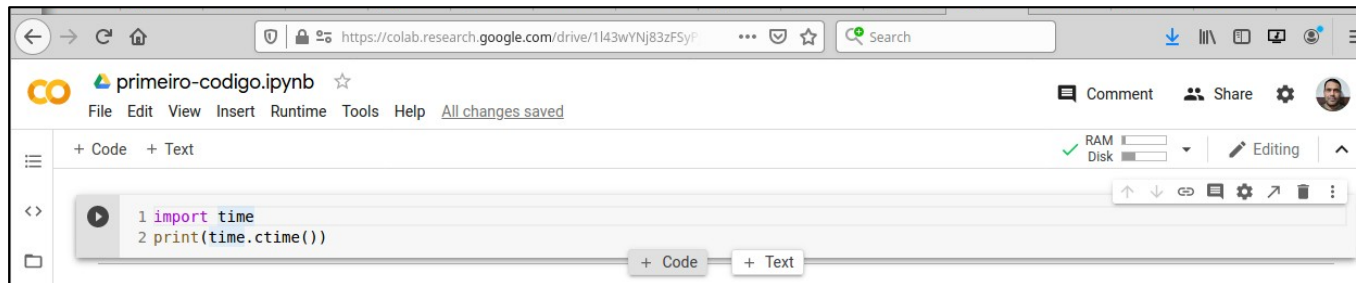
Execução

- Para limpar a saída da execução do programa, basta clicar.



Adicionando Células de Código

- Para inserir uma célula de código:
 - Insert → Code Cell
- Uma outra alternativa:



Adicionando Células de Código

- Digitando o código na nova célula:



The screenshot shows a Jupyter Notebook interface. At the top, there is a header with the Colab logo, the file name "primeiro-codigo.ipynb", and a star icon. Below the header is a menu bar with options: File, Edit, View, Insert, Runtime, Tools, Help, and a link for "All changes saved". On the right side of the header, there are icons for "Comment", "Share", a settings gear, and a user profile picture.

Below the header, there is a toolbar with "+ Code" and "+ Text" buttons. To the right of the toolbar, there are indicators for "RAM" and "Disk" usage, a status "Editing", and an upward arrow.

The main area of the notebook contains two code cells. The first cell is labeled "[2]" and contains the following code:

```
1 import time
2 print(time.ctime())
```

Below the code, there is a timestamp: "Fri Aug 28 00:59:02 2020".

The second cell is a code cell with a play button icon on the left and contains the following code:

```
1 time.sleep(5)
2 print (time.ctime())
```

Below the code, there is a timestamp: "Fri Aug 28 00:59:08 2020".


On the right side of the second cell, there is a small toolbar with icons for up, down, link, comment, settings, refresh, delete, and a vertical ellipsis.

Executando Todo Código

- Para executar o código inteiro (todas as células) sem interrupção, basta usar a opção do menu:
 - Runtime → Reset and run all

Mudando Ordem Células

- Quando o notebook tem muitas células, é possível que o programador precise alterar a ordem de execução.
- Isto pode ser feito selecionando as células para CIMA ou para BAIXO.



The screenshot shows a Jupyter Notebook interface. At the top, there is a header with the logo 'co', the text 'primeiro-codigo.ipynb', and a star icon. Below this is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help', followed by the text 'All changes saved'. On the right side of the header, there are icons for 'Comment', 'Share', a settings gear, and a user profile picture. Below the header, there is a toolbar with '+ Code' and '+ Text' on the left, and 'RAM', 'Disk', 'Editing', and a caret icon on the right. The main area contains two code cells. The first cell has the code:

```
[2] 1 import time
     2 print(time.ctime())
```

 and a timestamp 'Fri Aug 28 00:59:02 2020'. The second cell has the code:

```
1 time.sleep(5)
   2 print (time.ctime())
```

 and a timestamp 'Fri Aug 28 00:59:08 2020'. A red arrow points to the 'Move Up' button (upward arrow) in the toolbar of the second cell.

Apagando Células

- Em muitas situações, o programador pode precisar apagar as células de código.



The screenshot shows a Jupyter Notebook interface for a file named "primeiro-codigo.ipynb". The top navigation bar includes "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help", with a status message "All changes saved". On the right, there are "Comment", "Share", and "Settings" icons, along with a user profile picture. Below the navigation bar, there are "RAM" and "Disk" usage indicators and an "Editing" mode indicator. The main workspace contains two code cells. The first cell is labeled "[2]" and contains the code:

```
1 import time
2 print(time.ctime())
```

 Below the code is a timestamp: "Fri Aug 28 00:59:02 2020". The second cell contains the code:

```
1 time.sleep(5)
2 print (time.ctime())
```

 Below the code is a timestamp: "Fri Aug 28 00:59:08 2020". A toolbar is visible at the bottom right of the second cell, containing icons for up, down, refresh, comment, settings, and trash. A red arrow points to the trash icon, which is circled in red.

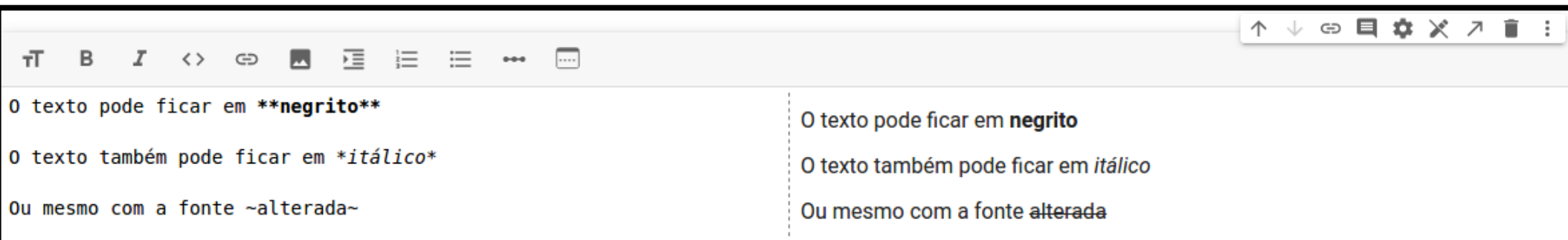
DOCUMENTAÇÃO DO CÓDIGO

Documentação de Código

- Python possui instruções para deixar código comentado.
- Porém, em muitas situações, o programador precisa mais do que texto simples para documentar os algoritmos de Aprendizagem de Máquinas (AM).
- As células de texto usam uma linguagem de marcação (*markdown*).

Exemplos de Marcação

- Algumas maneiras de mudar a formatação do texto:
 - ****negrito****
 - **italico**
 - ~riscada~

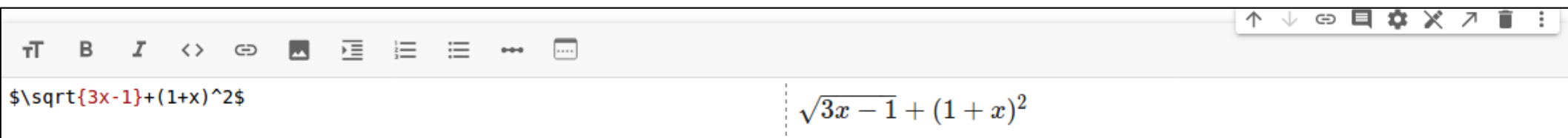


The screenshot shows a rich text editor interface with a toolbar at the top containing icons for text color, bold, italic, code, link, image, list, table, and more options. Below the toolbar, the editor is split into two columns by a vertical dashed line. The left column shows the raw text with formatting codes: "O texto pode ficar em ****negrito****", "O texto também pode ficar em **italico**", and "Ou mesmo com a fonte ~alterada~". The right column shows the rendered output: "O texto pode ficar em **negrito**", "O texto também pode ficar em *italico*", and "Ou mesmo com a fonte alterada".

Equações Matemáticas

- As equações matemáticas são baseadas na sintaxe de Latex.
- Exemplo:
 - Adicione uma célula de texto.
 - Adicione o seguinte texto:

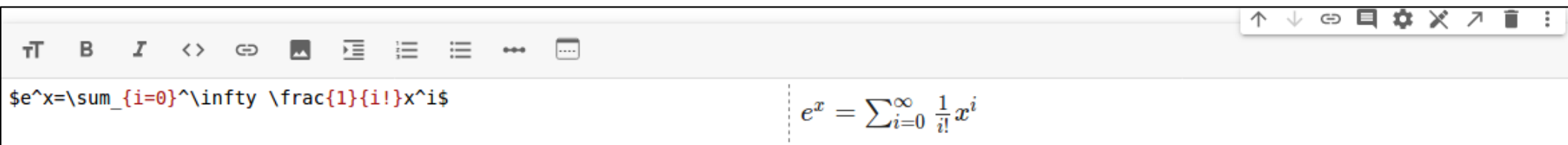
$\sqrt{3x-1}+(1+x)^2$



The screenshot shows a LaTeX editor interface. At the top, there is a toolbar with various icons for text formatting and editing. Below the toolbar, the input field contains the LaTeX code $\sqrt{3x-1}+(1+x)^2$. To the right of the input field, the rendered output is displayed as $\sqrt{3x-1}+(1+x)^2$.

Equações Matemáticas

- Outro exemplo:
 - $e^x = \sum_{i=0}^{\infty} \frac{1}{i!} x^i$



The screenshot shows a LaTeX editor interface. The top toolbar contains icons for undo, redo, bold, italic, link, unlink, insert image, list, table, and other editing tools. The main editing area is split into two sections. The left section contains the LaTeX source code: $e^x = \sum_{i=0}^{\infty} \frac{1}{i!} x^i$. The right section shows the rendered output of this code, which is the mathematical expression $e^x = \sum_{i=0}^{\infty} \frac{1}{i!} x^i$.

Equações Matemáticas

- Existem sites que facilitam a criação dessas fórmulas.
- CODECOGS
 - <https://latex.codecogs.com>

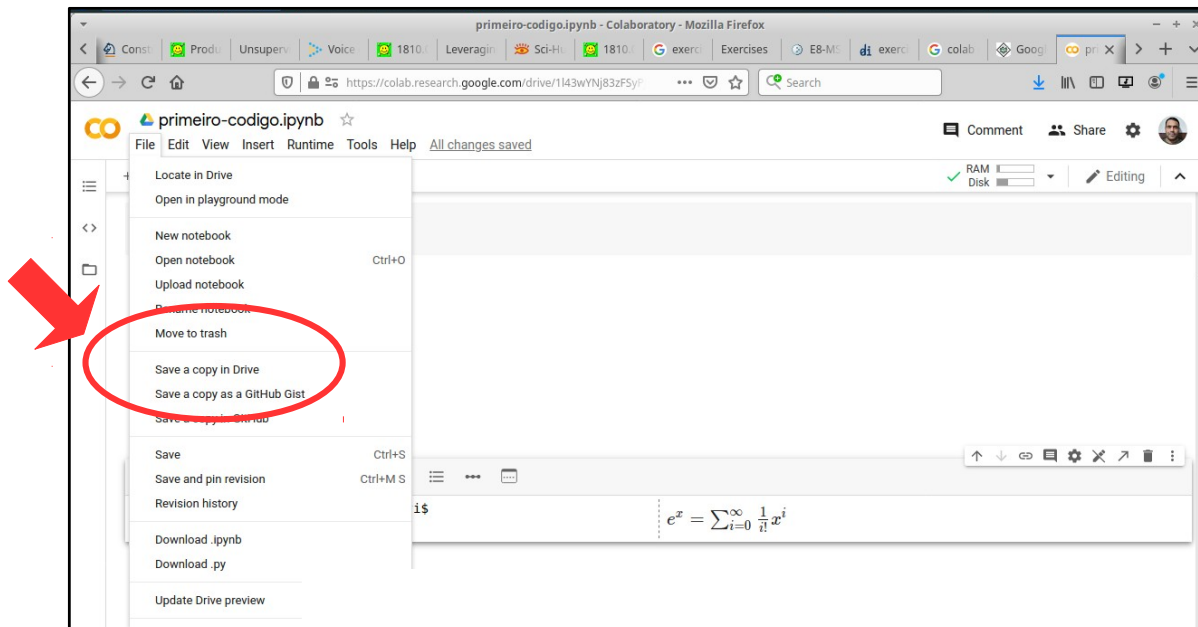
GRAVANDO SEU TRABALHO

Onde Gravar?

- COLAB permite gravar o trabalho no Google Drive ou no GitHub.

Google Drive

- COLAB permite gravar o notebook no Google Drive.
- Opção no menu:
 - File → Save a copy in Drive

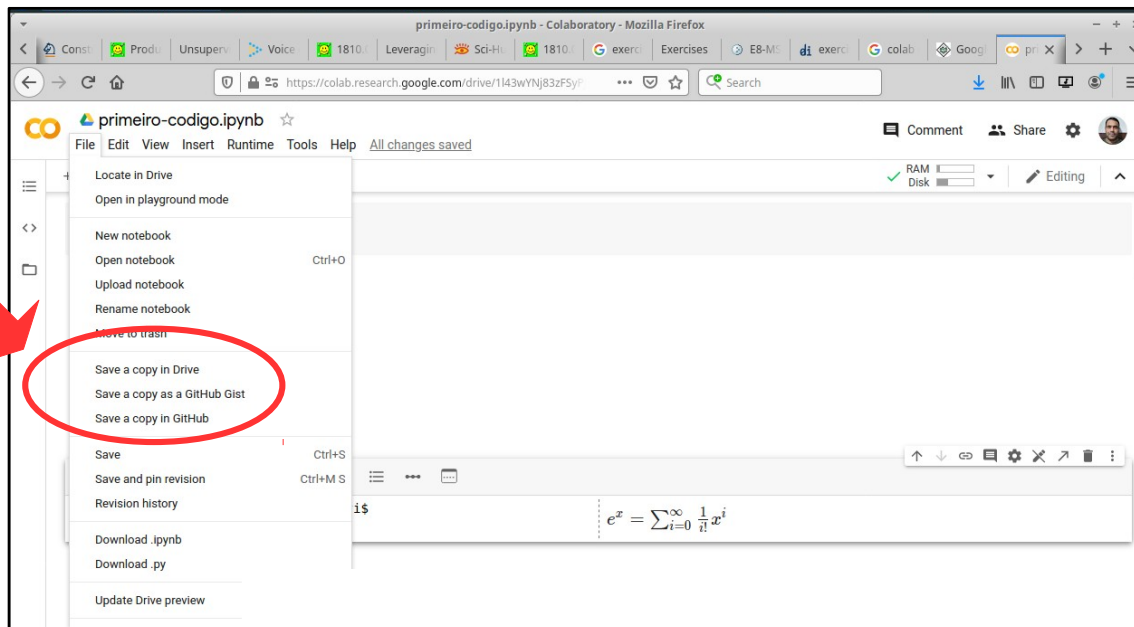


Google Drive

- Salvar no Google Drive permite que você acesse e/ou compartilhe com outras pessoas posteriormente.

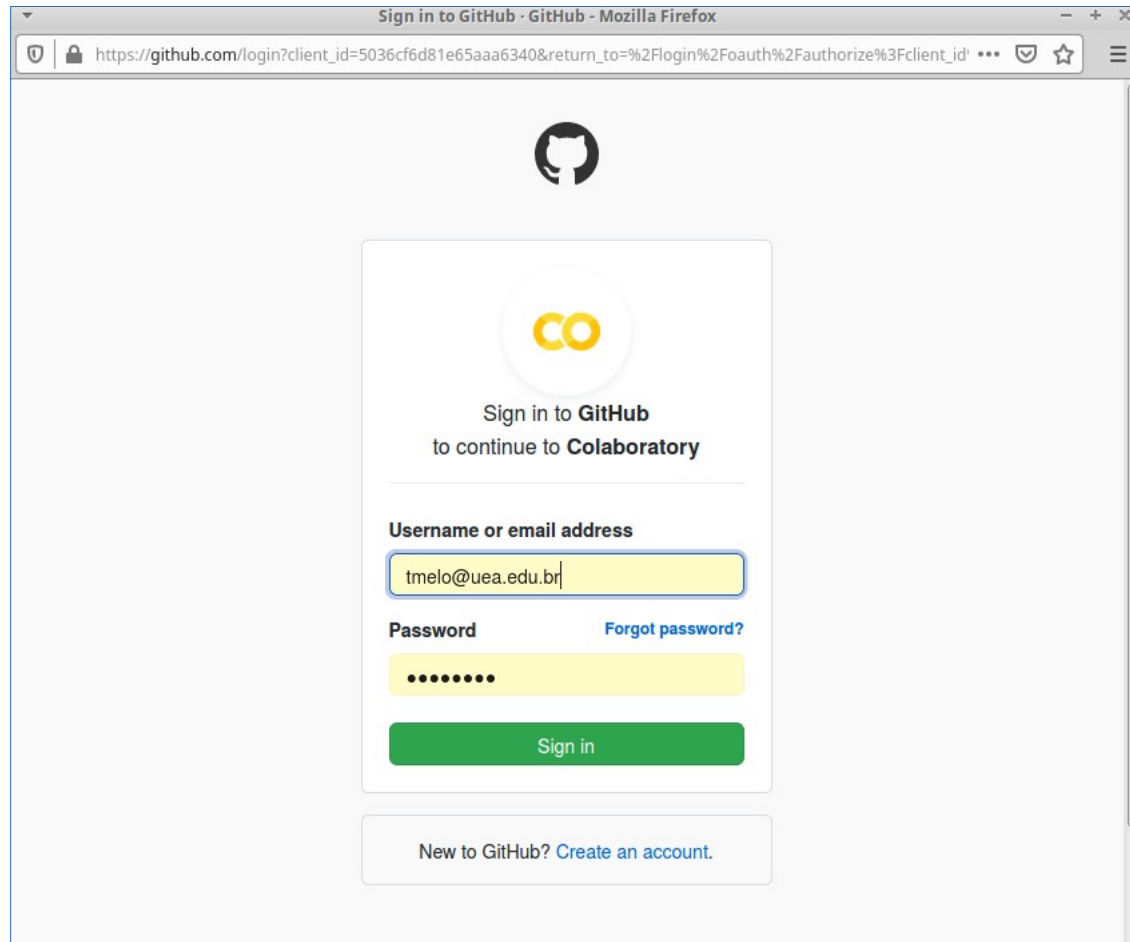
GitHub

- É possível salvar todo o seu trabalho no GitHub.
- Opção no menu:
 - File → Save a copy in GitHub



GitHub

- É necessário ter uma autorização (login) no GitHub.



GitHub

- É necessário criar um repositório no GitHub.

Copy to GitHub

Repository: [🔗](#)
tmelo-uea/covid19

Branch: [🔗](#)
master

File path
primeiro_codigo.ipynb

Commit message
Created using Colaboratory

Include a link to Colaboratory

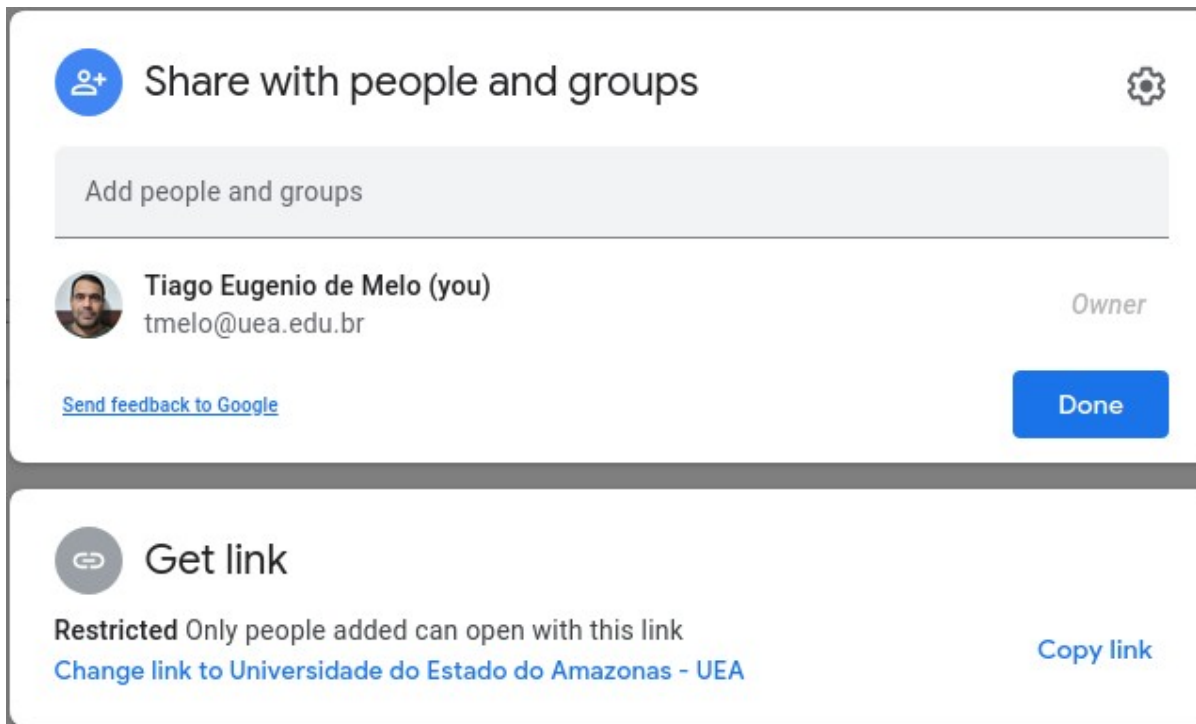
COMPARTILHAMENTO DE NOTEBOOK

Compartilhamento

- O compartilhamento do notebook pode ser feito através do Google Drive.
- Para publicar o notebook para o público em geral, você pode compartilhá-lo através do repositório do GitHub.

Compartilhamento

- O compartilhamento é feito através do botão (canto superior à direita).



Compartilhamento

- Existem várias opções de compartilhamento:
 - Para um grupo específico de pessoas.
 - Colegas da sua empresa (equipe).
 - Qualquer pessoa que possua o *link*.
 - O público em geral.

COMANDOS DE SISTEMA

Comandos de Sistema

- Jupyter permite a execução de vários comandos disponíveis em sistemas operacionais.
- O COLAB dá suporte a esse recurso.
- Esses comandos são iniciados com o sinal de exclamação (!).

Baixando Dados Remotos

- O comando WGET permite baixar (*download*) de arquivos remotos.
- Executar o comando:

```
!wget http://tiagodemelo.info/datasets/dados-curso.csv
```

```
[6] 1 !wget http://tiagodemelo.info/datasets/dados-curso.csv

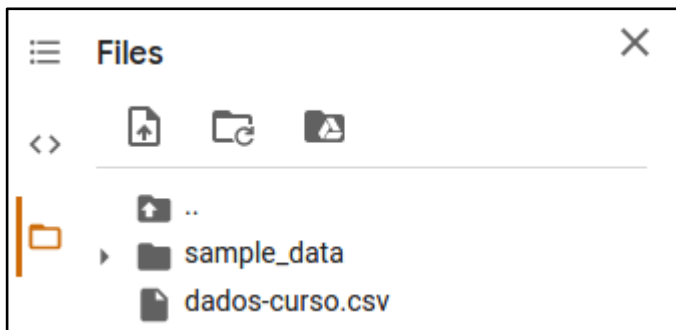
↳ --2020-08-28 02:07:35-- http://tiagodemelo.info/datasets/dados-curso.csv
Resolving tiagodemelo.info (tiagodemelo.info)... 108.167.188.189
Connecting to tiagodemelo.info (tiagodemelo.info)|108.167.188.189|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 102696 (100K) [text/csv]
Saving to: 'dados-curso.csv'

dados-curso.csv      100%[=====>] 100.29K  ---KB/s   in 0.1s

2020-08-28 02:07:35 (833 KB/s) - 'dados-curso.csv' saved [102696/102696]
```

Baixando Dados Remotos

- O arquivo ficará disponível no Google Drive.



- Esse arquivo pode ser lido (usado) através da biblioteca Pandas:

```
1 import pandas as pd
2
3 data = pd.read_csv("/content/dados-curso.csv")
4
5 data.head()
```


Baixando Dados Remotos

- Resultado:

	data	texto	retweet	idioma	lugar	pais	sigla	latitude	longitude
0	2020-05-23 00:21:14	Para voltar tudo ao normal, você precisa fazer...	0	pt	Rio das Ostras	Brazil	BR	-41.937900	-22.522600
1	2020-03-22 22:57:51	14.245 - O que é a hidroxiclороquina? https://...	0	pt	Sao Paulo	Brazil	BR	-46.674739	-23.606067
2	2020-04-14 00:11:33	Quarta morte em Lar de Estarreja associada à C...	1	pt	Lisbon	Portugal	PT	-9.099043	38.747518
3	2020-05-25 20:45:43	COVID-19 Hospital municipal Tide Setúbal, n...	0	pt	Sao Paulo	Brazil	BR	-46.633300	-23.550000
4	2020-04-15 10:34:39	#cenasdocotidiano #santos #distanciamentosocia...	0	pt	Santos	Brazil	BR	-46.293700	-23.975947

Baixando Dados Remotos

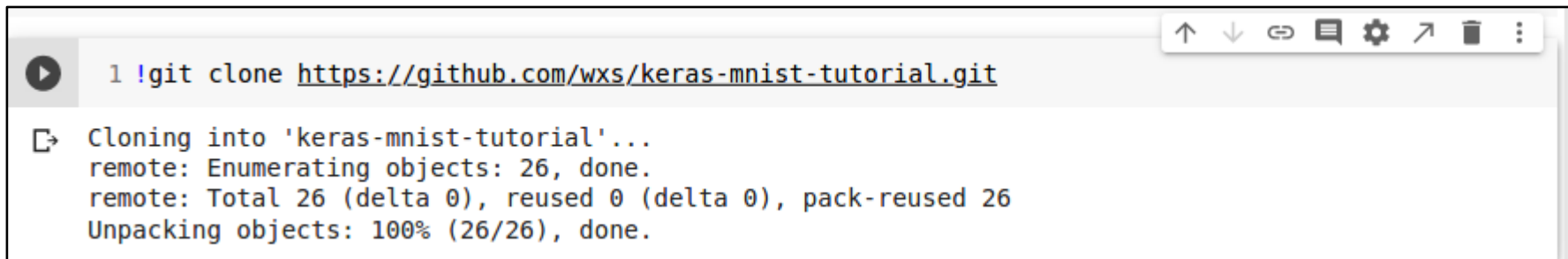
- Uma outra maneira de baixar os arquivos é usar os recursos da biblioteca Pandas para baixar o arquivo diretamente.

```
1 import pandas as pd
2
3 data = pd.read_csv("http://tiagodemelo.info/datasets/dados-curso.csv")
4
5 data.head()
```



Clonando Repositório Git

- É possível clonar o repositório inteiro do GitHub utilizando o comando `git`.



```
1 !git clone https://github.com/wxs/keras-mnist-tutorial.git
↳ Cloning into 'keras-mnist-tutorial'...
  remote: Enumerating objects: 26, done.
  remote: Total 26 (delta 0), reused 0 (delta 0), pack-reused 26
  Unpacking objects: 100% (26/26), done.
```

EXECUTANDO ARQUIVOS EXTERNOS DE PYTHON

Arquivos Externos (Python)

- Suponha que você já tenha desenvolvido alguns programas em Python e que estejam no Google Drive.
- É possível executá-los no COLAB.

Montar Drive

- O primeiro passo é montar o Google Drive.
- Opção no menu:
 - Tools → Command palette

The screenshot shows a Jupyter Notebook interface with the following elements:

- File Explorer (Left):** Shows a directory structure with folders 'keras-mnist-tutorial' and 'sample_data', and a file 'dados-curso.csv'.
- Code Cell (Center):** Contains a code cell with the following content:

```
√3x - 1 + (1 + x)2  
ex = ∑i=0∞ 1/i! xi  
[6] 1 !wget http://tiagodeme...  
--2020-08-28 02:07:35--  
Resolving tiagodemelo.info...  
Connecting to tiagodemelo.info...  
HTTP request sent, awaiting...  
Length: 102696 (100K) [text/csv]  
Saving to: 'dados-curso.csv'  
dados-curso.csv 100%[...]  
2020-08-28 02:07:35 (833 KB/s) - 'dados-curso.csv' saved [102696/102696]  
  
[8] 1 import pandas as pd  
2  
3 data = pd.read_csv("http://tiagodemelo.info/datasets/dados-curso.csv")  
4  
5 data.head()  
  
1 !git clone https://github.com/wxs/keras-mnist-tutorial.git  
Cloning into 'keras-mnist-tutorial'...  
remote: Enumerating objects: 26, done.  
remote: Total 26 (delta 0), reused 0 (delta 0), pack-reused 26  
Unpacking objects: 100% (26/26), done.
```
- Command Palette (Right):** A menu is open with 'Mount Drive' highlighted. Other options include 'Move notebook to trash', 'Move selected cells up', 'Open in playground mode', 'Manage sessions', 'Merge focused cell with next cell', etc.
- Interface Elements:** The top bar shows 'primeiro-codigo.ipynb' and 'All changes saved'. The bottom status bar shows 'RAM' and 'Disk' usage.

Montar Drive

- Uma célula de código será automaticamente criada.

```
1 from google.colab import drive
2 drive.mount('/content/drive')
```

- Se você executar o código acima, será solicitada uma autenticação.

```
1 from google.colab import drive
2 drive.mount('/content/drive')
```

... Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6c

Enter your authorization code:

Montar Drive

- Basta clicar no link e fazer a autenticação:


```
1 from google.colab import drive
2 drive.mount('/content/drive')
```


... Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6c

Enter your authorization code:


Montar Drive


- Autenticação:


 Fazer login com o Google





Escolha uma conta
para prosseguir para [Google Drive File Stream](#)


 **Tiago Melo**
tiago.melo@oceanbrasil.com


 **Tiago Eugenio de Melo**
tmelo@uea.edu.br

 **Tiago Melo**
tiagodemelo@gmail.com

 **Tiago Melo**
tiagoeugeniodemelo@gmail.com

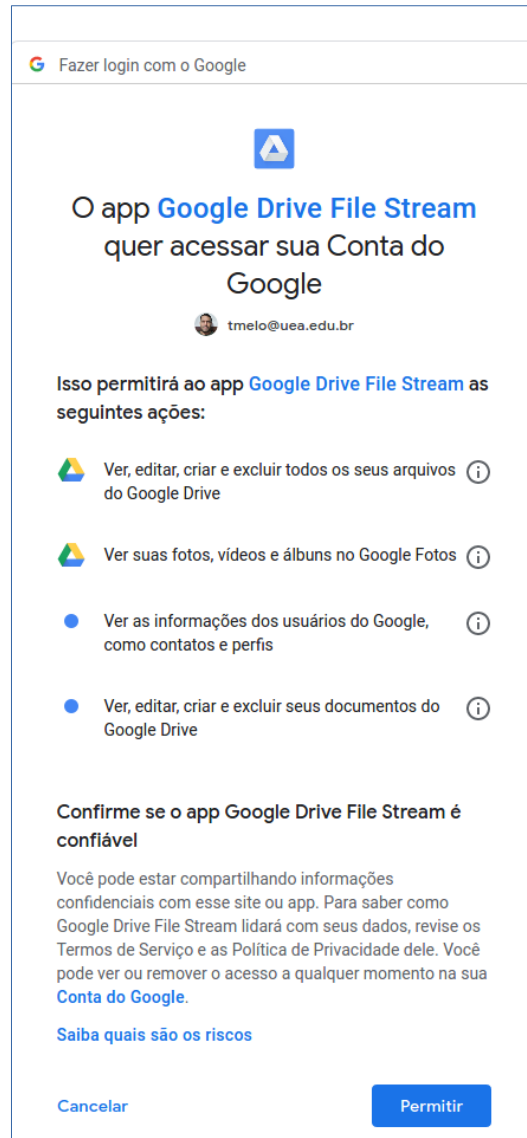
 **Tiago Eugênio de Melo** Desconectado
tiago.melo@icomp.ufam.edu.br

 **Tiago Melo**
tiagodemelo.phd@gmail.com

 Usar outra conta

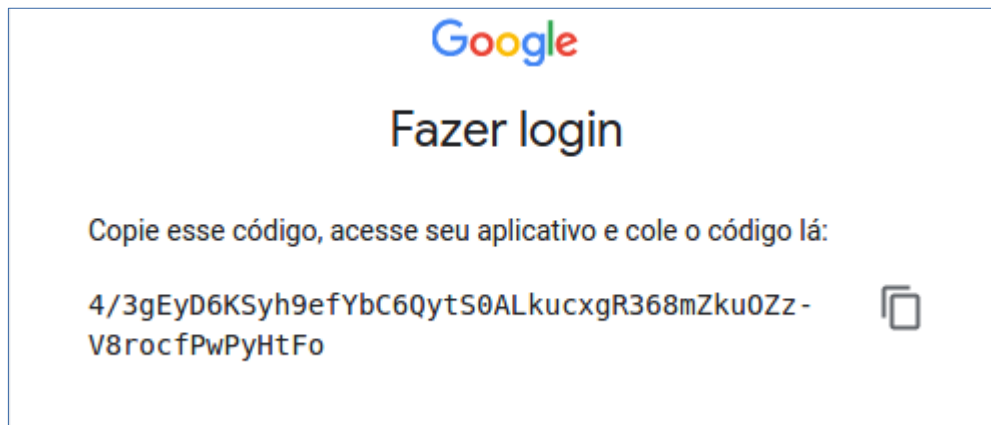
Montar Drive

- Autenticação:

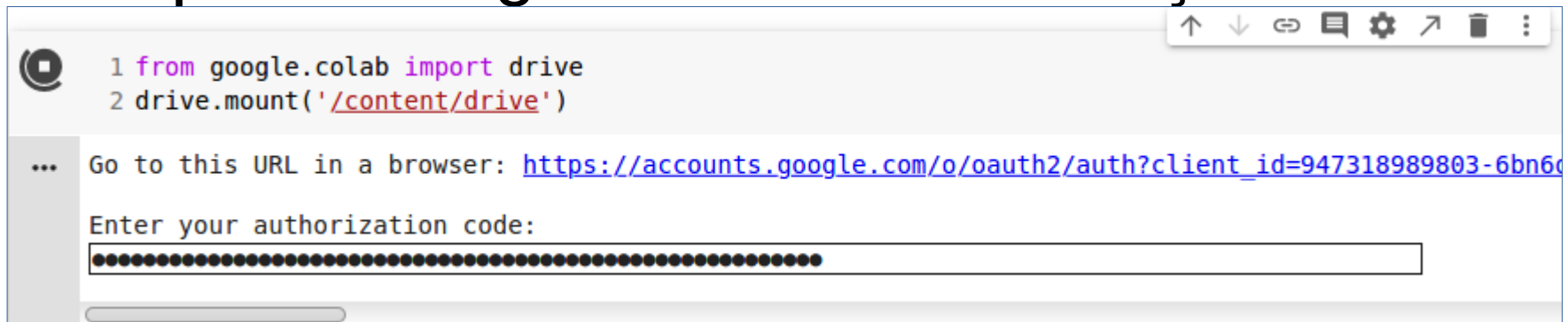


Montar Drive

- Será gerado um código automaticamente:

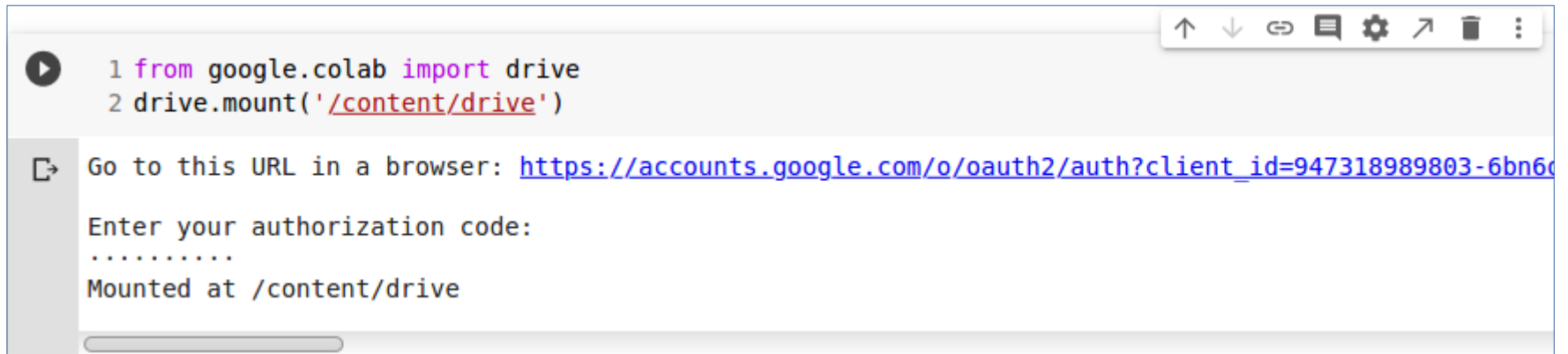


- Copia o código e cola na solicitação:



Montar Drive

- Confirmada a autenticação:



```
1 from google.colab import drive
2 drive.mount('/content/drive')
```

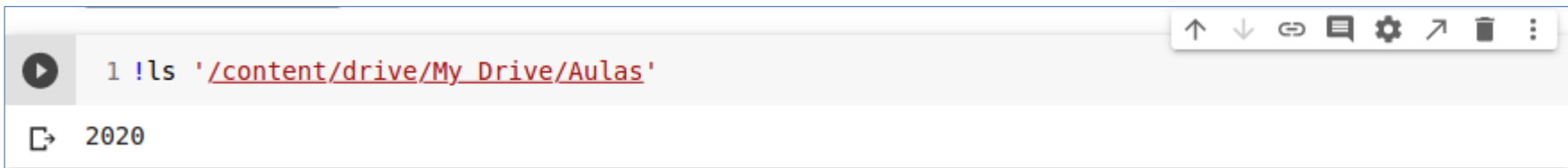
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6c

Enter your authorization code:
.....

Mounted at /content/drive

Execução do Código Python

- É possível listar os arquivos em um determinado diretório:

A screenshot of a terminal window. The command prompt shows a play button icon followed by the command `!ls '/content/drive/My_Drive/Aulas'`. Below the command, the output shows the year `2020`. The terminal has a standard toolbar at the top right with icons for back, forward, search, settings, and other functions.

```
1 !ls '/content/drive/My_Drive/Aulas'  
2020
```

- A execução do programa:

```
!python3 "/content/drive/My Drive/Colab Notebooks/hello.py"
```

GRÁFICOS

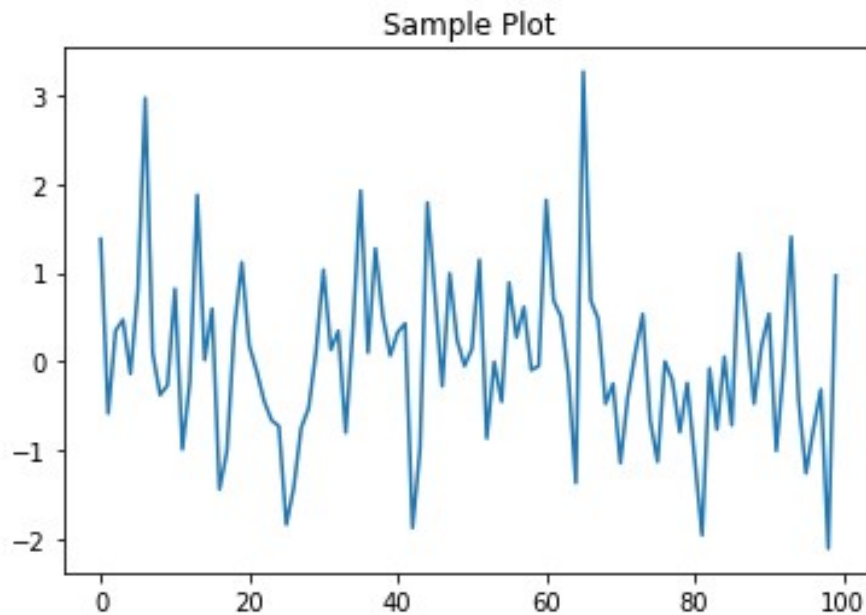
Gráficos

- COLAB dá suporte a sofisticados tipos de gráficos.
- Exemplo:

```
1 import numpy as np
2 from matplotlib import pyplot as plt
3
4 y = np.random.randn(100)
5
6 x = [x for x in range(len(y))]
7
8 plt.plot(x, y, '-')
9
10 plt.fill_between(x, y, 200, where=(y > 195), facecolor='g', alpha=0.6)
11
12 plt.title("Sample Plot")
13
14 plt.show()
```

Gráficos

- Saída do exemplo (código) anterior:



AJUDA (HELP)

Help

- As ferramentas de IDE costumam usar ajuda sensível ao contexto.
- Também recursos como auto-completar.
- COLAB possui todos esses recursos que facilitam a atividade de programação.

Lista de Função

- **Passo 1**

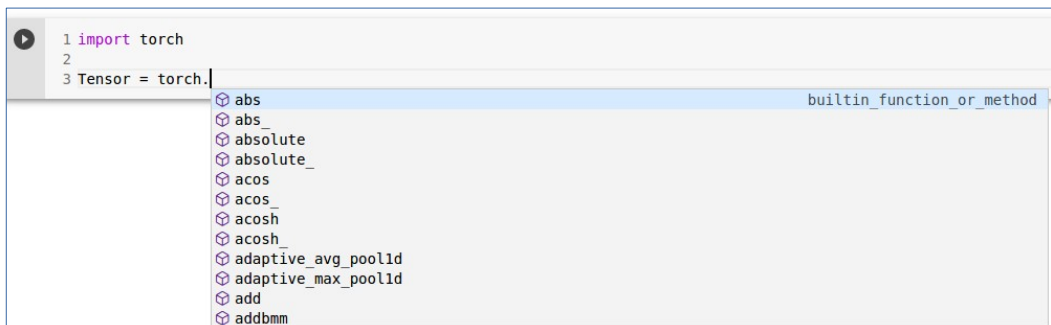
- Abra um novo notebook ou uma nova célula de código e digite o código abaixo:

```
import torch
```

- **Passo 2**

- Digite o código abaixo:

```
Tensor = torch.
```



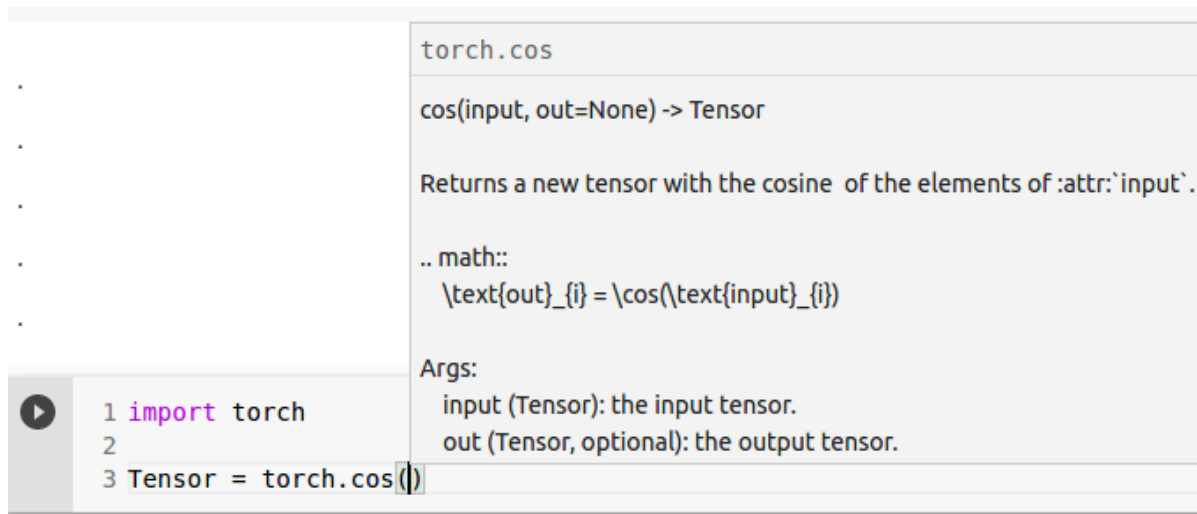
```
1 import torch
2
3 Tensor = torch.
```

The screenshot shows a Jupyter Notebook code cell with three lines of code. The third line, 'Tensor = torch.', is active, and a dropdown menu is open below it, listing various functions from the torch module. The first item in the list is 'abs', which is highlighted in blue. The text 'builtin_function_or_method' is visible at the end of the first item in the dropdown.

Função de Documentação

- COLAB fornece documentação sobre qualquer **função** ou **classe** como ajuda ao contexto.
- Digite o código abaixo:

```
Tensor = torch.cos (
```



```
torch.cos
cos(input, out=None) -> Tensor

Returns a new tensor with the cosine of the elements of :attr:`input`.

.. math::
    \text{out}_{i} = \cos(\text{input}_{i})

Args:
    input (Tensor): the input tensor.
    out (Tensor, optional): the output tensor.
```

```
1 import torch
2
3 Tensor = torch.cos()
```

COMANDOS MÁGICOS

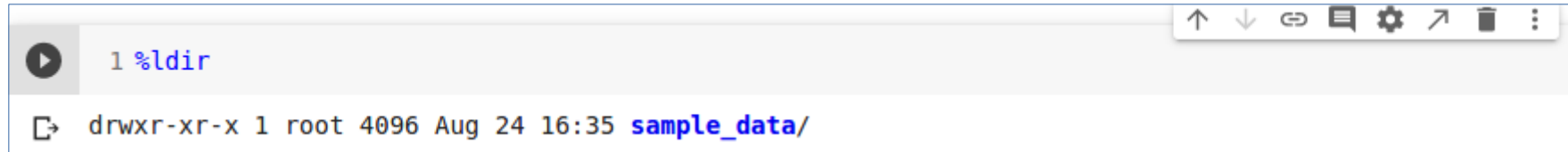
Comandos Mágicos

- COLAB possui um conjunto de comandos de sistema que ampliam as funcionalidades.
- Tipos:
 - Linhas mágicas (*line magics*).
 - Células mágicas (*cell magics*).
- Linhas mágicas:
 - Única linha de comando.
 - Código usa (%) no início do comando.
- Células mágicas:
 - Atua para toda a célula de código.
 - Código usa (%%) no início do comando.

Linhas Mágicas

- Exemplo 1:

%!dir

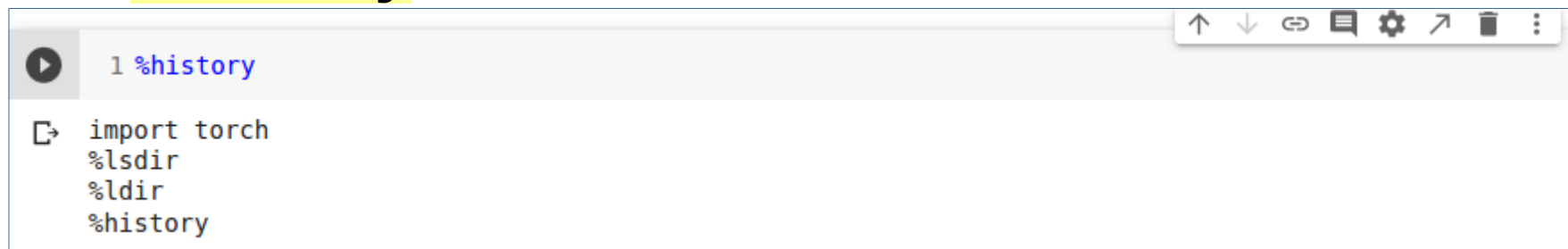


```
1 %!dir
drwxr-xr-x 1 root 4096 Aug 24 16:35 sample_data/
```

A terminal window showing the execution of the magic command `%!dir`. The command is entered in the prompt, and the output displays the directory listing for the current directory, which is `sample_data/`. The terminal window includes a toolbar with navigation and utility icons.

- Exemplo 2:

%!history

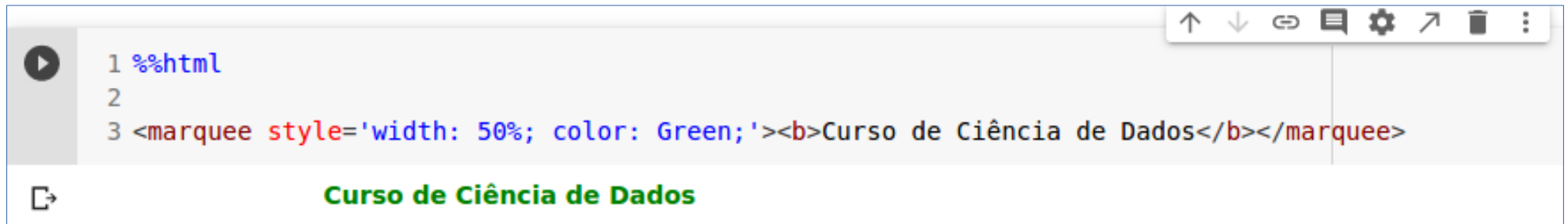


```
1 %!history
import torch
%!lsdir
%!dir
%!history
```

A terminal window showing the execution of the magic command `%!history`. The command is entered in the prompt, and the output displays the history of the current session, listing the commands: `import torch`, `%!lsdir`, `%!dir`, and `%!history`. The terminal window includes a toolbar with navigation and utility icons.

Células Mágicas

- Uso de recursos de HTML.
- Exemplo:



The screenshot shows a Jupyter Notebook cell with the following content:

```
1 %%html
2
3 <marquee style='width: 50%; color: Green;'><b>Curso de Ciência de Dados</b></marquee>
```

Below the code, the rendered output is displayed: **Curso de Ciência de Dados**. The text is bold and green, and is contained within a marquee element that scrolls horizontally.

Lista Mágica

- Comando para verificar a lista de comandos mágicos:

%lsmagic

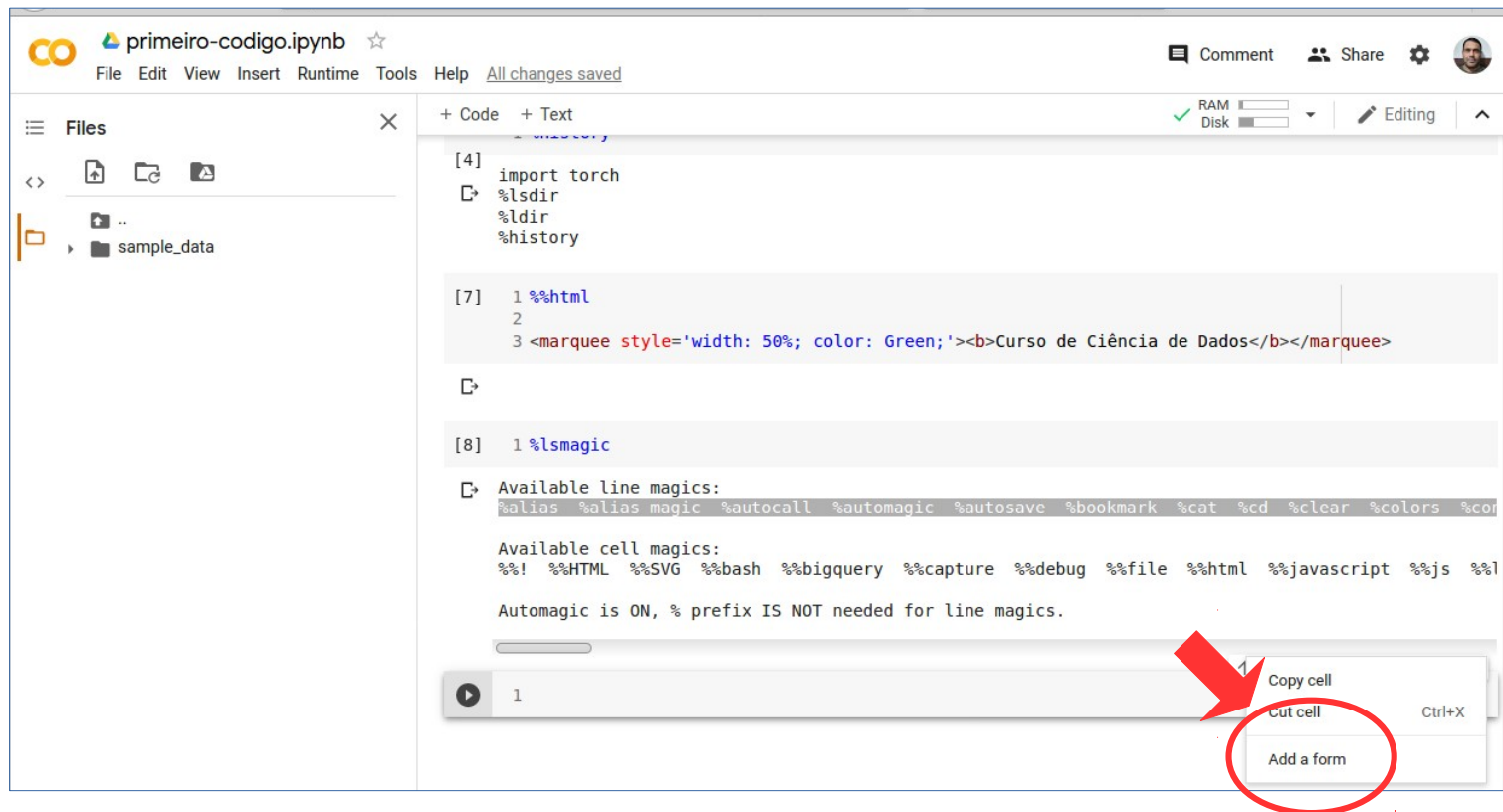
FORMULÁRIOS

Introdução

- COLAB provê um recurso bastante interessante de formulários.
- Os formulários podem ser usados para receber a entrada (*input*) de dados de usuários em tempo de execução.

Inserção de Formulários

- **Passo 1:**
 - Adiciona o formulário.



The screenshot shows a Jupyter Notebook interface with the following content:

- File Explorer on the left showing a folder named `sample_data`.
- Code cell [4] containing:

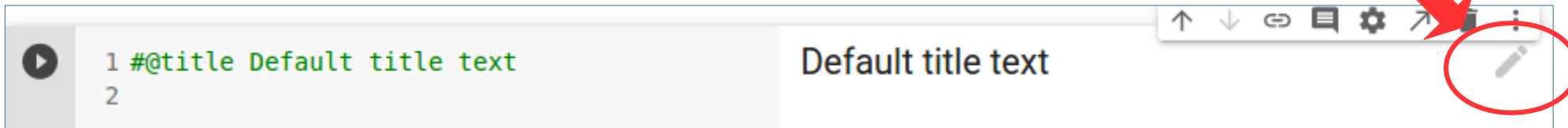
```
import torch
%lsdir
%ldir
%history
```
- Code cell [7] containing:

```
1 %%html
2
3 <marquee style='width: 50%; color: Green;'><b>Curso de Ciência de Dados</b></marquee>
```
- Code cell [8] containing:

```
1 %lsmagic
```
- Output of cell [8] showing available line and cell magics.
- Cell toolbar at the bottom with a red circle around the `Add a form` button.

Inserção de Formulários

- **Passo 2:**
 - Botão de configuração (ícone de caneta).



Edit form attributes

Title text
Default title text

Initial form visibility
last shown (default) ▼

Output height _____ px

Form width _____ px ▼

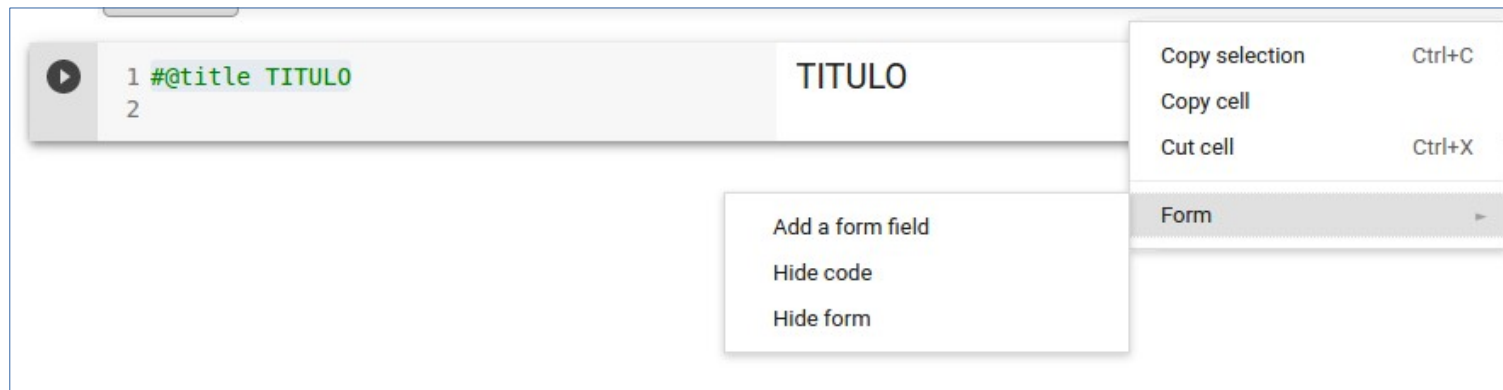
Auto-execute cell when fields change

Display output below form

CANCEL SAVE

Inserção de Formulários

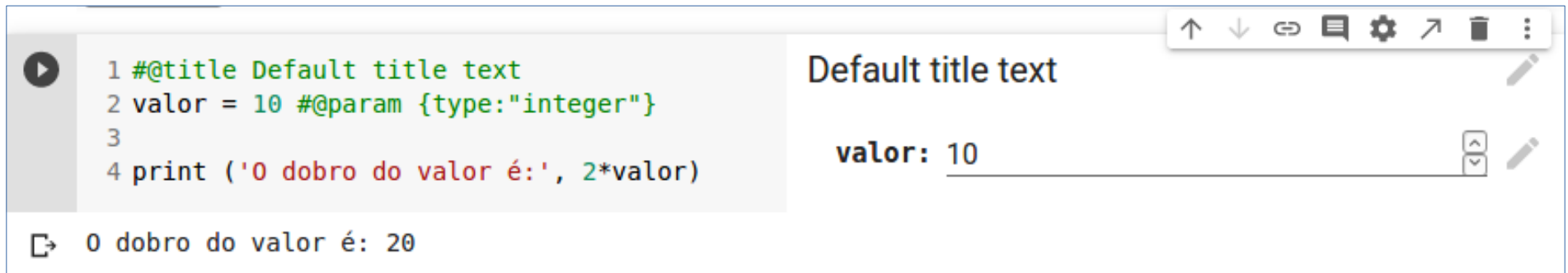
- Passo 3:
 - Adicionar formulário:



The screenshot shows the 'Add new form field' dialog box. It has a title bar 'Add new form field'. There are three input fields: 'Form field type' with a dropdown menu showing 'input', 'Variable type' with a dropdown menu showing 'string', and 'Variable name' with a text input field containing 'variable_name'. Red arrows point to the 'Form field type' dropdown, the 'Variable type' dropdown, and the 'Variable name' text input. At the bottom right, there are 'CANCEL' and 'SAVE' buttons.

Inserção de Formulários

- Uso do formulário:



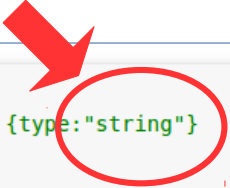
The screenshot displays a code editor interface. On the left, a code block contains the following Python code:

```
1 #@title Default title text
2 valor = 10 #@param {type:"integer"}
3
4 print ('O dobro do valor é:', 2*valor)
```

Below the code, the output is shown: `O dobro do valor é: 20`. On the right, the rendered HTML output is shown, featuring a form with the title "Default title text" and an input field labeled "valor:" containing the value "10". The input field has a spinner control on its right side. The editor interface includes a toolbar at the top right with icons for navigation, search, and other functions.

Entrada de Texto

- Para aceitar a entrada de texto como parâmetro, basta alterar o tipo de dados.
- Exemplo:



```
1 #@title Default title text
2 nome = "Curso de Programa\xE7\xE3o" #@param {type:"string"}
3
4 print (nome)
```

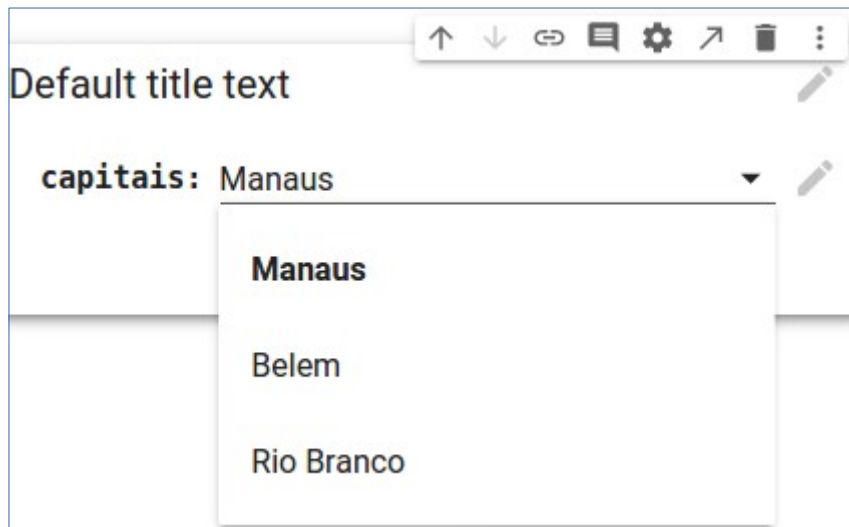
Curso de Programação

Default title text

nome: "Curso de Programa\xE7\xE3o"

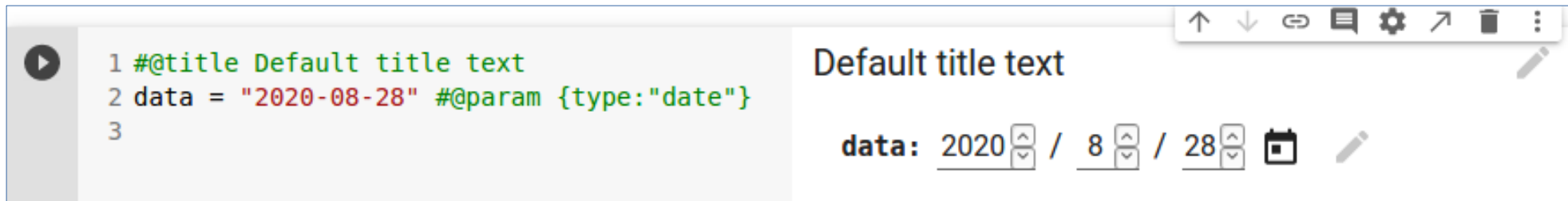
Lista de Dados

- É possível usar o formulário com lista de dados.
- Exemplo:



Entrada de Datas

- É possível usar a entrada de datas.
- Exemplo:



The screenshot shows a code editor interface. On the left, a code block contains three lines of text: `1 #@title Default title text`, `2 data = "2020-08-28" #@param {type:"date"}`, and `3`. On the right, the rendered output is displayed. The title is "Default title text". Below it, the parameter "data" is rendered as a date input field showing "2020 / 8 / 28" with a calendar icon and a pencil icon for editing.

INSTALAÇÃO DE PACOTES AVANÇADOS

Introdução

- COLAB suporta a maioria das bibliotecas de *machine learning* disponíveis.
- A instalação é similar ao ambiente Linux.

!pip install

!apt-get install

Keras

- Escrita em Python e executar usando *frameworks* como TensorFlow ou Theano.
- Permite rápida prototipação de aplicações usando redes neurais.
- Suporta uso de GPU.

```
!pip install -q keras
```

PyTorch

- Pacote bastante usado no desenvolvimento de aplicações com redes neurais.

```
!pip3 install torch torchvision
```

GPU



Introdução

- Google fornece o uso gratuito de GPUs.

Configuração GPUs

- Para usar o recurso de GPU:
 - Runtime → Change runtime type

Notebook settings

Hardware accelerator
GPU  

To get the most out of Colab, avoid using a GPU unless you need one. [Learn more](#)

Omit code cell output when saving this notebook

CANCEL SAVE

Listando Dispositivos

- É possível descobrir (listar) os dispositivos usados pelo seu notebook na nuvem.
- Exemplo (CPU):

```
1 from tensorflow.python.client import device_lib
2 device_lib.list_local_devices()

[name: "/device:CPU:0"
 device_type: "CPU"
 memory_limit: 268435456
 locality {
 }
 incarnation: 13740433489619651683, name: "/device:XLA_CPU:0"
 device_type: "XLA_CPU"
 memory_limit: 17179869184
 locality {
 }
 incarnation: 9018186359868751899
 physical_device_desc: "device: XLA_CPU device"]
```

Listando Dispositivos

- Exemplo (GPU):

```
1 from tensorflow.python.client import device_lib
2 device_lib.list_local_devices()

[{"name": "/device:CPU:0",
  device_type: "CPU",
  memory_limit: 268435456,
  locality {
  },
  incarnation: 2684393887369763698, name: "/device:XLA_CPU:0",
  device_type: "XLA_CPU",
  memory_limit: 17179869184,
  locality {
  },
  incarnation: 15107021630547681460,
  physical_device_desc: "device: XLA_CPU device", name: "/device:XLA_GPU:0",
  device_type: "XLA_GPU",
  memory_limit: 17179869184,
  locality {
  },
  incarnation: 16755329250648886052,
  physical_device_desc: "device: XLA_GPU device", name: "/device:GPU:0",
  device_type: "GPU",
  memory_limit: 15695549568,
  locality {
    bus_id: 1
    links {
    }
  }
}, {"name": "/device:GPU:0",
  device_type: "GPU",
  memory_limit: 15695549568,
  locality {
    bus_id: 1
    links {
    }
  },
  incarnation: 16200912918321920207,
  physical_device_desc: "device: 0, name: Tesla P100-PCIE-16GB, pci bus id: 0000:00:04.0, compute capability: 6.0"]
```

Memória

- É possível checar os recursos de memória através do comando abaixo:

```
!cat /proc/meminfo
```