

Introdução a Programação para Geoprocessamento

**TUTORIAL:
TRABALHANDO COM NOTEBOOKS NO GOOGLE
COLABORATORY**



Prof. Dr. Alexandre Gularde Schafer

Janeiro de 2024

SUMÁRIO

1. INTRODUÇÃO.....	1
2. ACESSO AOS NOTEBOOKS.....	4
3. CONFIGURAÇÃO DO TEMA DO NOTEBOOK.....	6
4. TRABALHANDO COM CÉLULAS.....	11
4.1 Células de Código.....	12
4.2 Células de Texto.....	13
4.3 Adicionando células.....	17
5. TRABALHANDO COM ARQUIVOS EXTERNOS NO GOOGLE COLAB.....	18
5.1 Gerenciamento de arquivos e diretórios no Google Colab.....	18
5.2 Montagem do Drive.....	20
5.3 Organização das pastas do curso no Google Drive.....	31
5.4 Upload dos arquivos para o Google Drive.....	41
5.5 Verificação das pastas no ambiente “File” do Google Colab.....	46

1. INTRODUÇÃO

Este tutorial tem como objetivo orientar sobre a utilização dos notebooks no curso de Introdução à Programação para Geoprocessamento no Google Colaboratory. Antes de procedermos às instruções detalhadas, é essencial compreender alguns conceitos fundamentais que nos auxiliarão neste processo de aprendizagem.

Um notebook, no contexto de programação e análise de dados, é um documento digital interativo que permite ao usuário escrever e executar código em partes, chamadas de células, juntamente com a inclusão de textos explicativos, equações, imagens e visualizações.



O notebook é uma ferramenta de grande utilidade para experimentação, documentação e compartilhamento de conhecimento técnico.

O Google Colaboratory, ou "Colab", é um serviço gratuito hospedado na nuvem que permite a qualquer pessoa escrever e executar código Python através do navegador, sem a necessidade de configuração complexa de ambientes. É

baseado no conceito de Jupyter Notebooks, um projeto de código aberto que suporta várias linguagens de programação.

Utilizar notebooks no Google Colaboratory traz uma série de vantagens, principalmente para iniciantes na área de geoprocessamento e/ou programação:

- Acessibilidade: O Colab é acessível através do navegador, o que significa que você pode trabalhar em seus projetos de qualquer lugar e em qualquer dispositivo, sem se preocupar com a instalação de software;
- Recursos computacionais gratuitos: O Google Colab oferece acesso gratuito a GPUs (Unidades de Processamento Gráfico) e TPUs (Unidades de Processamento Tensor), o que pode acelerar significativamente os cálculos e a análise de dados volumosos típicos em geoprocessamento;
- Facilidade de compartilhamento: Com o Colab, você pode facilmente compartilhar seus notebooks, promovendo colaboração e feedback. Além disso, você pode acessar notebooks compartilhados por outros, o que facilita o aprendizado colaborativo;
- Integração com Google Drive: Todos os seus notebooks podem ser salvos diretamente no Google Drive, facilitando o gerenciamento de arquivos e o backup automático;
- Ambiente rico para aprendizado: Além de permitir a execução de código, os notebooks no Colab suportam a inclusão de textos explicativos, o que os

torna perfeitos para criar tutoriais que misturam teoria com prática.

2. ACESSO AOS NOTEBOOKS

Você pode acessar o notebook de cada capítulo individualmente a partir de um link específico no livro texto. Acesse o livro-texto no capítulo 2:

<https://alexandrogschafer.github.io/Programacao-Geoprocessamento/capitulo2.html>

The screenshot shows a Jupyter Notebook interface. On the left is a sidebar with a Python logo icon and a navigation menu:

- Programação para Geoprocessamento
- 1. INTRODUÇÃO
- 2. FUNDAMENTOS DA LINGUAGEM PYTHON
- 3. A BIBLIOTECA PANDAS
- 4. GEOPROCESSAMENTO BÁSICO COM SHAPELY E FIONA
- 5 A BIBLIOTECA GEOPANDAS
- 6. VISUALIZAÇÃO DE DADOS GEOESPACIAIS NO PYTHON
- REFERÊNCIAS BIBLIOGRÁFICAS

The main content area displays Chapter 2: FUNDAMENTOS DA LINGUAGEM PYTHON. The title is "2. FUNDAMENTOS DA LINGUAGEM PYTHON". Below it is a sub-section titled "2.1 Sintaxe básica: variáveis, operadores, expressões". A text block explains the basic syntax of Python for geoprocessing. To the right is a "Contents" sidebar listing chapters 2.1 through 2.10. At the bottom of the main content area is a code block containing Python code:

```
x = 10
y = 8
nome = 'Ana'
idade = 30
altura_metros = 1.75
```

Clique no link para abrir o notebook no Google Colab:

The screenshot shows a section of a Python documentation page. On the left, there's a sidebar with a navigation menu. In the main content area, the title '2. FUNDAMENTOS DA LINGUAGEM PYTHON' is displayed. Below it, a sub-section title '2.1 Sintaxe básica: variáveis, operadores, expressões' is shown. A red arrow points from the 'Abrir no Google Colab' button (which is highlighted with a red box) to the left margin of the page.

Notebook do capítulo 2 no Google colab:

The screenshot shows a Google Colab notebook titled 'capítulo2.ipynb'. The notebook interface includes a top bar with file options like File, Edit, View, Insert, Runtime, Tools, Help, Share, and AI. The main workspace contains the content of the '2.1 Sintaxe básica: variáveis, operadores, expressões' section from the previous screenshot. It includes explanatory text about Python variables, a code block with variable assignments, and a note about dynamic typing. The code block contains the following Python code:

```
x = 10
y = 8
nome = 'Ana'
idade = 30
altura_metros = 1.75
```

3. CONFIGURAÇÃO DO TEMA DO NOTEBOOK

No Google Colaboratory (Colab), personalizar a aparência visual do ambiente de programação é uma maneira de tornar a experiência de codificação não apenas mais agradável, mas também mais ergonômica. A interface do usuário do Colab permite essa personalização através da alteração do tema, o que pode ajudar a reduzir a fadiga visual durante longos períodos de trabalho e adaptar o ambiente ao seu estilo pessoal de codificação.

O Google Colab oferece duas opções principais de tema: claro e escuro. Por padrão, o tema claro é ativado para todos os usuários. Este tema é ideal para ambientes bem iluminados, onde um fundo mais claro pode ajudar a destacar o texto sem causar desconforto visual.

No entanto, para aqueles que preferem uma interface mais suave e menos brilhante, especialmente em ambientes com pouca luz ou para longas sessões de codificação, sugerimos a alteração do tema dos notebooks para "Dark" (Escuro). O tema escuro reduz o brilho da tela, diminuindo a tensão nos olhos e facilitando a concentração no código.

É importante notar que a mudança de tema no Google Colab é puramente estética e não afeta a funcionalidade ou o desempenho do notebook.

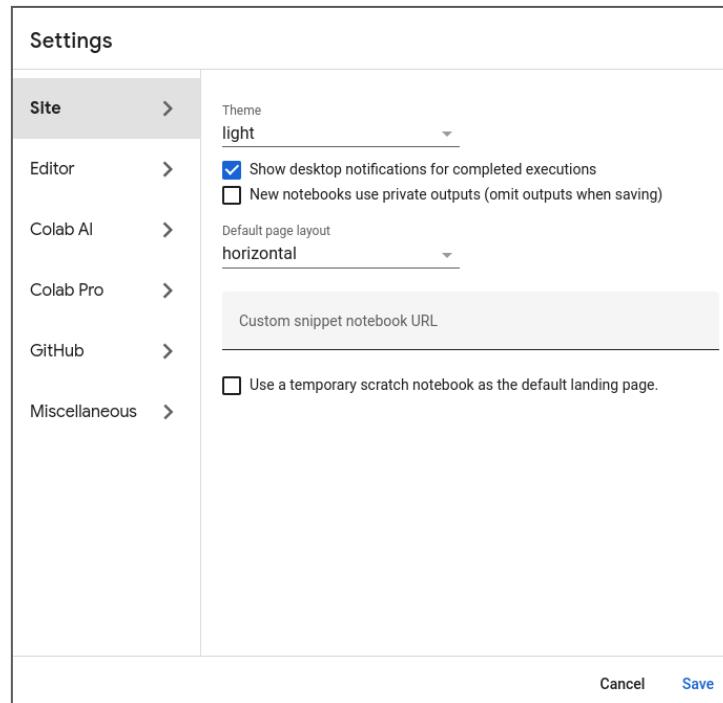
Para alterar o tema no Google Colab, clique em “Tools” no menu superior:

The screenshot shows the Google Colab interface with a notebook titled "capítulo2.ipynb". The "Tools" menu is open, displaying options: Command palette (Ctrl+Shift+P), Settings, Keyboard shortcuts (Ctrl+M H), and Diff notebooks. A red arrow points from the text above to the "Settings" option in the menu.

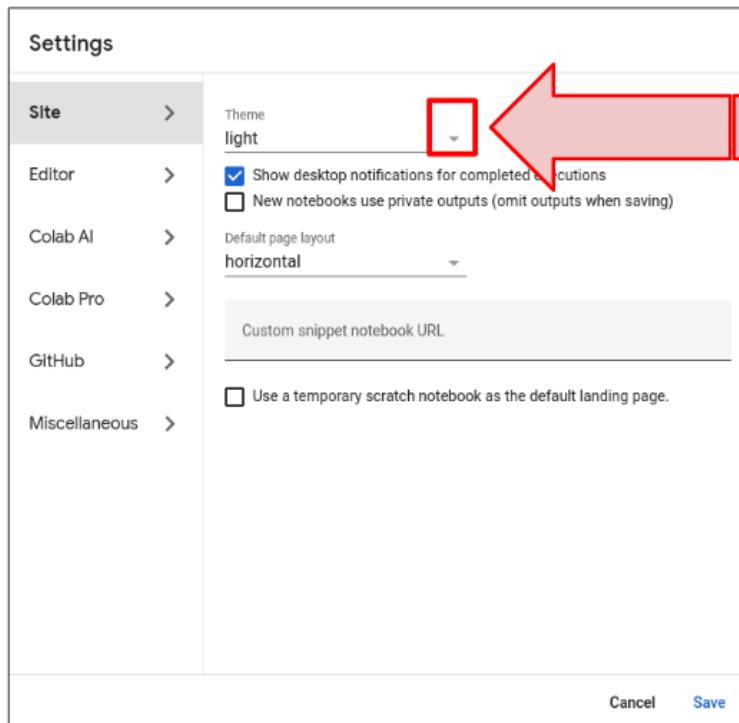
Clique em Settings:

The screenshot shows the Google Colab interface with the "Settings" option in the "Tools" menu highlighted by a red box. A red arrow points from the previous screenshot's instruction to this highlighted option.

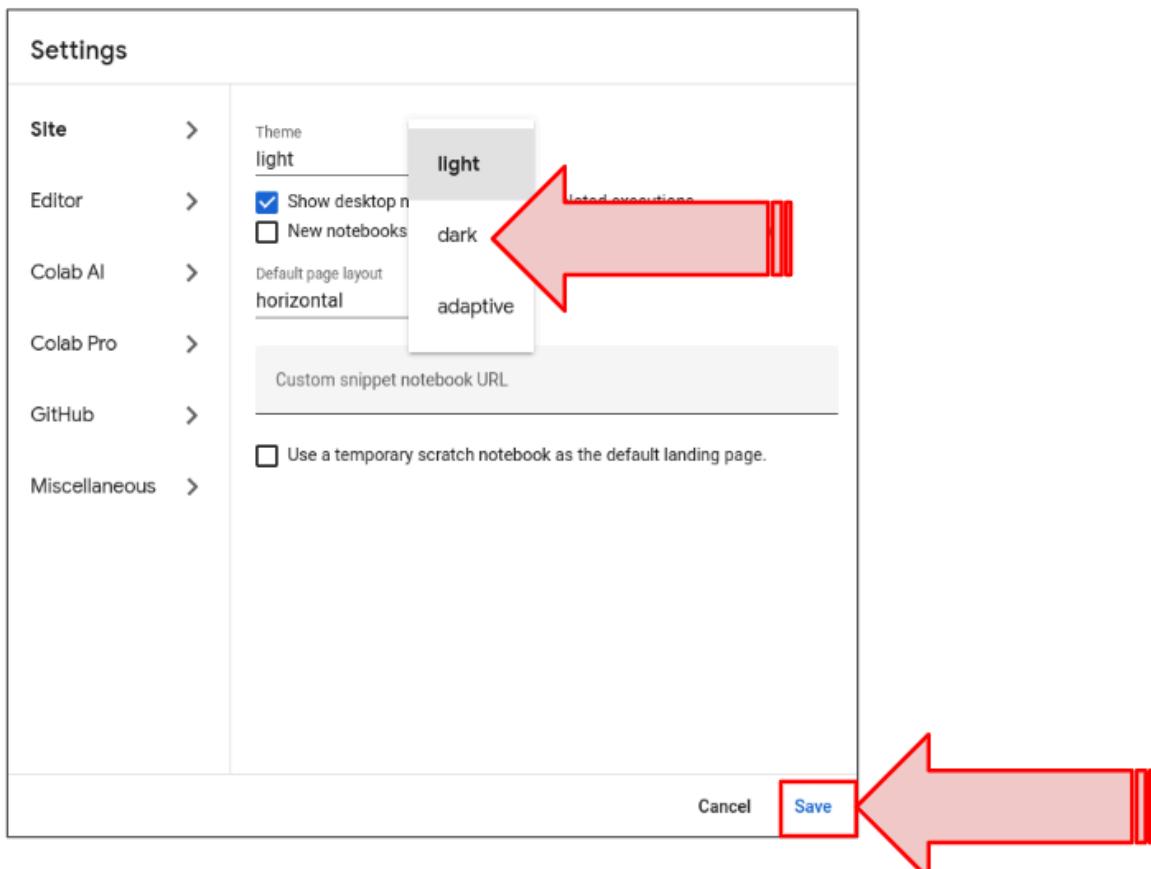
A aba “Settings” abrirá conforme imagem abaixo:



Clique no local indicado na figura abaixo:



Clique em “dark” e em seguida em “Save”:



O tema será alterado para “dark” e o notebook terá a aparência da figura abaixo:

The screenshot shows a Google Colab notebook titled "capítulo2.ipynb". The interface is in dark mode. The notebook structure includes a section titled "2. FUNDAMENTOS DA LINGUAGEM PYTHON" which further branches into "2.1 Sintaxe básica: variáveis, operadores, expressões" and "2.1.1 Variáveis". A code cell displays the following Python assignment statements:

```
[ ] x = 10
y = 8
nome = 'Ana'
idade = 30
altura_metros = 1.75
```

Below the code, there is explanatory text about dynamic typing in Python:

Tipagem dinâmica refere-se ao mecanismo pelo qual o tipo de uma variável é determinado em tempo de execução, ao contrário da tipagem estática, onde o tipo de uma variável é determinado em tempo de compilação. Em linguagens com tipagem dinâmica, o tipo de uma variável pode mudar durante a execução do programa, dependendo do valor que lhe é atribuído.

Cada linguagem de programação possui suas próprias regras e convenções para nomear variáveis, mas existem algumas práticas gerais recomendadas:

- O nome geralmente começa com uma letra ou sublinhado (_);
- Pode conter letras (maiusculas ou minúsculas), números ou sublinhados;
- Não deve conter espaços ou caracteres especiais;

4. TRABALHANDO COM CÉLULAS

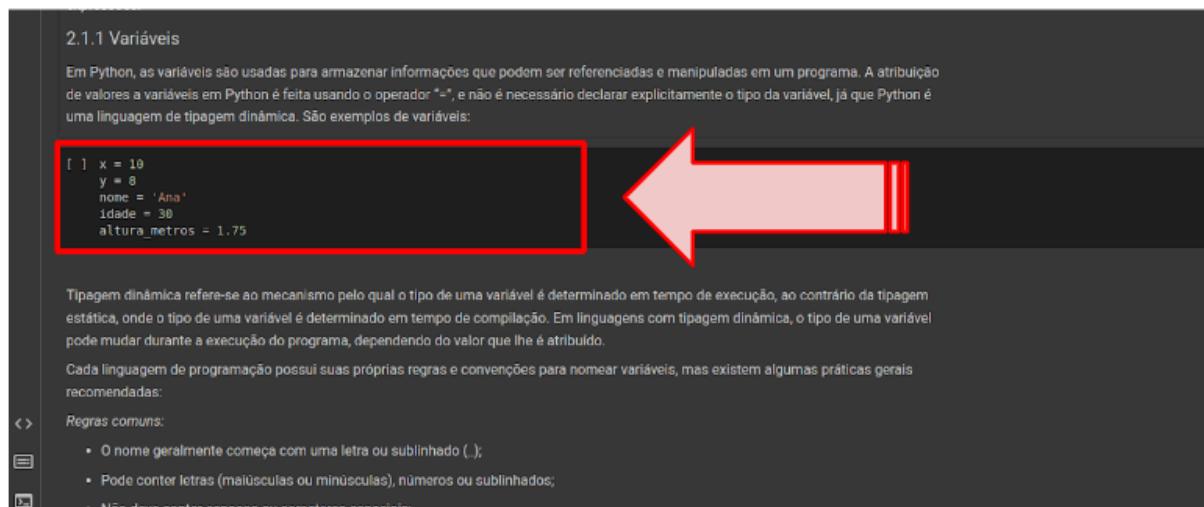
As células constituem componente central dos ambientes de programação em plataformas como Jupyter Notebooks e Google Colaboratory, servindo como unidades para a inserção e execução de código, assim como para a adição de textos explicativos. Elas possibilitam uma interação direta e eficiente entre a execução do código e sua documentação, essencial para a clareza e a compreensão dos projetos de programação e de análise de dados.

Essa integração entre código e comentários textuais facilita a compreensão dos conceitos e processos envolvidos, tornando os projetos mais acessíveis e fáceis de seguir. A estrutura baseada em células promove uma metodologia de trabalho colaborativo, permitindo que projetos sejam desenvolvidos, revisados e compreendidos de maneira mais eficaz.

Existem dois tipos principais de células: células de código e células de texto. Cada tipo desempenha funções distintas e complementares.

4.1 Células de Código

As células de código são o ambiente onde os usuários inserem e executam comandos de programação em linguagens suportadas pela plataforma, como o Python. Essas células permitem a execução imediata do código, exibindo os resultados diretamente abaixo da célula de entrada. São úteis para testar, desenvolver e demonstrar funcionalidades de programação em tempo real.



2.1.1 Variáveis

Em Python, as variáveis são usadas para armazenar informações que podem ser referenciadas e manipuladas em um programa. A atribuição de valores a variáveis em Python é feita usando o operador “=”, e não é necessário declarar explicitamente o tipo da variável, já que Python é uma linguagem de tipagem dinâmica. São exemplos de variáveis:

```
[ ] x = 10
y = 8
nome = 'Ana'
idade = 30
altura_metros = 1.75
```

Tipagem dinâmica refere-se ao mecanismo pelo qual o tipo de uma variável é determinado em tempo de execução, ao contrário da tipagem estática, onde o tipo de uma variável é determinado em tempo de compilação. Em linguagens com tipagem dinâmica, o tipo de uma variável pode mudar durante a execução do programa, dependendo do valor que lhe é atribuído.

Cada linguagem de programação possui suas próprias regras e convenções para nomear variáveis, mas existem algumas práticas gerais recomendadas:

Regras comuns:

- O nome geralmente começa com uma letra ou sublinhado (_);
- Pode conter letras (maiúsculas ou minúsculas), números ou sublinhados;
- Não deve conter espaços ou caracteres especiais;

4.2 Células de Texto

As células de texto utilizam a linguagem de marcação Markdown ou HTML para adicionar anotações, explicações, ou qualquer outro tipo de conteúdo textual que não requer execução como código. Estas células são utilizadas para a documentação do notebook, permitindo que os autores incluam instruções detalhadas, explicações teóricas, links úteis, imagens e gráficos para acompanhar o código. Elas tornam os notebooks mais acessíveis e compreensíveis, facilitando a comunicação de conceitos complexos e a colaboração entre usuários.

The screenshot shows a Jupyter Notebook interface. A red box highlights a text cell containing Python code and explanatory text about dynamic typing. Two large red arrows point from the right side of the image towards this highlighted area, drawing attention to the content.

```
[ ] x = 10
y = 8
nome = 'Ana'
idade = 30
altura_metros = 1.75
```

O entendimento da sintaxe básica do Python é fundamental para começar a programar nesta linguagem, especialmente em aplicações voltadas para o geoprocessamento. Nesta seção abordaremos três elementos essenciais da sintaxe do Python: variáveis, operadores e expressões.

2.1.1 Variáveis

Em Python, as variáveis são usadas para armazenar informações que podem ser referenciadas e manipuladas em um programa. A atribuição de valores a variáveis em Python é feita usando o operador “=”, e não é necessário declarar explicitamente o tipo da variável, já que Python é uma linguagem de tipagem dinâmica. São exemplos de variáveis:

Tipagem dinâmica refere-se ao mecanismo pelo qual o tipo de uma variável é determinado em tempo de execução, ao contrário da tipagem estática, onde o tipo de uma variável é determinado em tempo de compilação. Em linguagens com tipagem dinâmica, o tipo de uma variável pode mudar durante a execução do programa, dependendo do valor que lhe é atribuído.

Cada linguagem de programação possui suas próprias regras e convenções para nomear variáveis, mas existem algumas práticas gerais recomendadas:

Regras comuns:

- O nome geralmente começa com uma letra ou sublinhado (_);
- Pode conter letras (maiúsculas ou minúsculas), números ou sublinhados;
- Não deve conter espaços ou caracteres especiais.

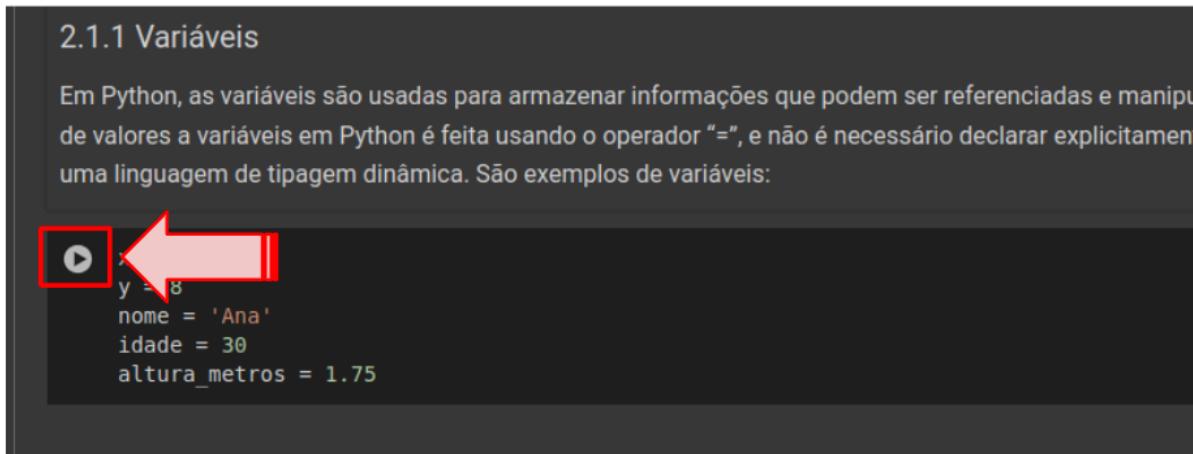
Executando a primeira célula de código:

The screenshot shows a Jupyter Notebook interface. A red box highlights a code cell containing Python code. A play button icon is visible to the left of the code. Two large red arrows point from the right side of the image towards this highlighted area, drawing attention to the content.

```
▶ x = 10
y = 8
nome = 'Ana'
idade = 30
altura_metros = 1.75
```

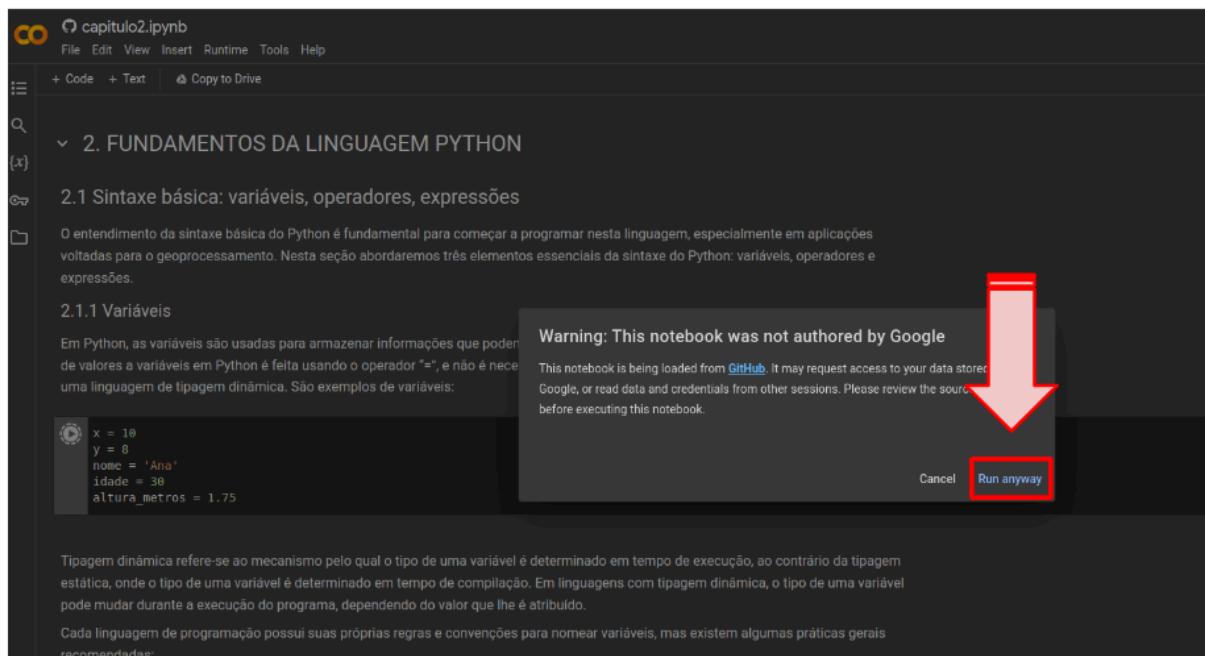
Em Python, as variáveis são usadas para armazenar informações que podem ser referenciadas e manipuladas em um programa. A atribuição de valores a variáveis em Python é feita usando o operador “=”, e não é necessário declarar explicitamente o tipo da variável, já que Python é uma linguagem de tipagem dinâmica. São exemplos de variáveis:

Clique no botão de execução de código:



```
y = 8
nome = 'Ana'
idade = 30
altura_metros = 1.75
```

Clique em *Run anyway*:



Warning: This notebook was not authored by Google

This notebook is being loaded from [GitHub](#). It may request access to your data stored in Google, or read data and credentials from other sessions. Please review the source before executing this notebook.

Cancel Run anyway

Essa célula não vai gerar nenhuma saída, pois estamos apenas criando variáveis. Execute as demais células do Notebook em que são criadas outras variáveis.

O código da célula abaixo irá gerar saídas. Clique no botão de execução da célula:

The screenshot shows a Jupyter Notebook interface with the title "capítulo2.ipynb". In the code cell, the following Python code is written:

```
x = 7  
y = 12  
x, y = y, x  
print('O valor de x é:', x)  
print('O valor de y é:', y)
```

A red arrow points to the play button icon in the toolbar above the code cell. Below the code cell, the output is displayed:

Neste caso, x receberá o valor 15, y receberá o valor 25, e z receberá o valor 35.
Uma das vantagens da atribuição múltipla é a facilidade com que você pode trocar valores entre duas variáveis.

A atribuição múltipla torna o código mais conciso e, em muitos casos, mais legível, especialmente quando usada de maneira apropriada e não excessiva.

2.1.2 Operadores: aritméticos, comparação, lógicos

Podemos ler as saídas abaixo da célula:

The screenshot shows a Jupyter Notebook interface with the title "capítulo2.ipynb". In the code cell, the same Python code is written:

```
x = 7  
y = 12  
x, y = y, x  
print('O valor de x é:', x)  
print('O valor de y é:', y)
```

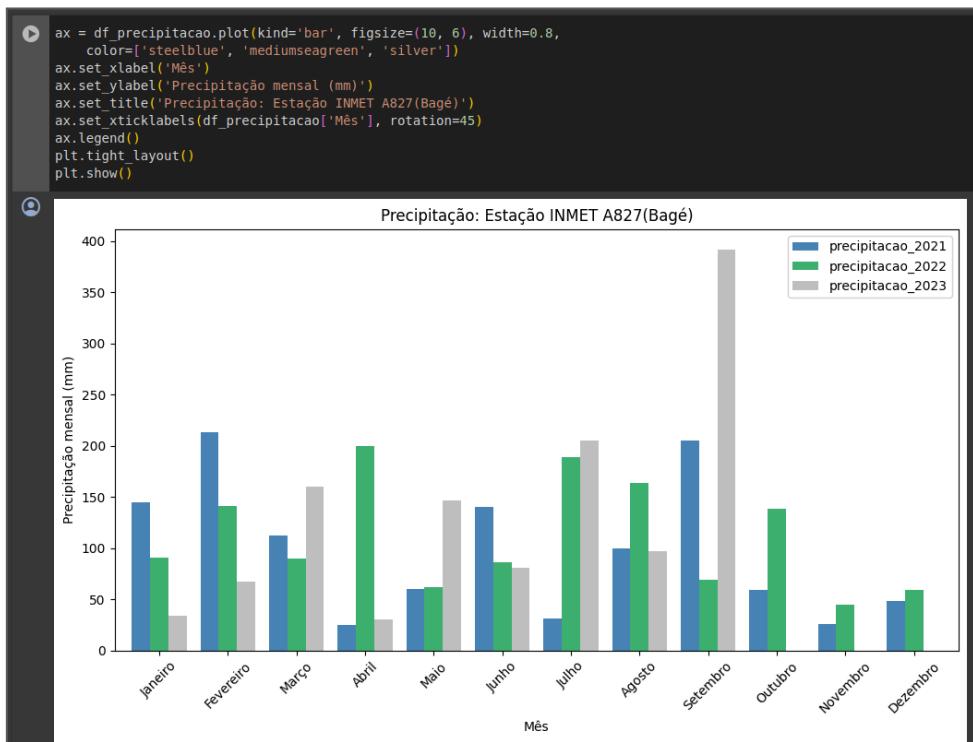
A red arrow points to the play button icon in the toolbar above the code cell. Below the code cell, the output is displayed:

Neste caso, x receberá o valor 15, y receberá o valor 25, e z receberá o valor 35.
Uma das vantagens da atribuição múltipla é a facilidade com que você pode trocar valores entre duas variáveis.

O valor de x é: 12
O valor de y é: 7

A atribuição múltipla torna o código mais conciso e, em muitos casos, mais legível, especialmente quando usada de maneira apropriada e não excessiva.

Abaixo temos um exemplo de saída de gráfico:

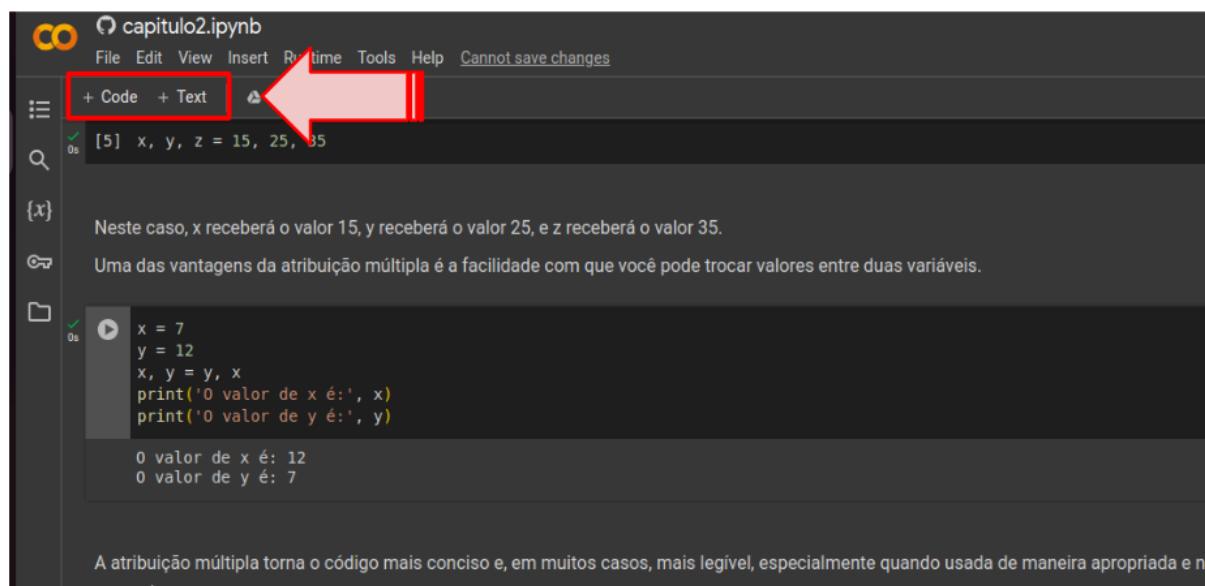


Na figura abaixo, temos um exemplo de plot de mapa:



4.3 Adicionando células

Você pode adicionar células clicando nos botões indicados na figura abaixo. o botão “+ Code” insere células de código e o botão “+ Text” insere células de texto:



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

- Title Bar:** Shows the notebook file name "capítulo2.ipynb".
- Toolbar:** Features a "File" menu, followed by "Edit", "View", "Insert", "Runtime", "Tools", and "Help". A status message "Cannot save changes" is displayed.
- Buttons:** Two buttons are highlighted with a red box and arrow: "+ Code" and "+ Text".
- Code Cell:** Contains the code:

```
[5] x, y, z = 15, 25, 35
```
- Text Cell:** Contains the text:

{x} Neste caso, x receberá o valor 15, y receberá o valor 25, e z receberá o valor 35.
Uma das vantagens da atribuição múltipla é a facilidade com que você pode trocar valores entre duas variáveis.
- Output Cell:** Shows the execution results:

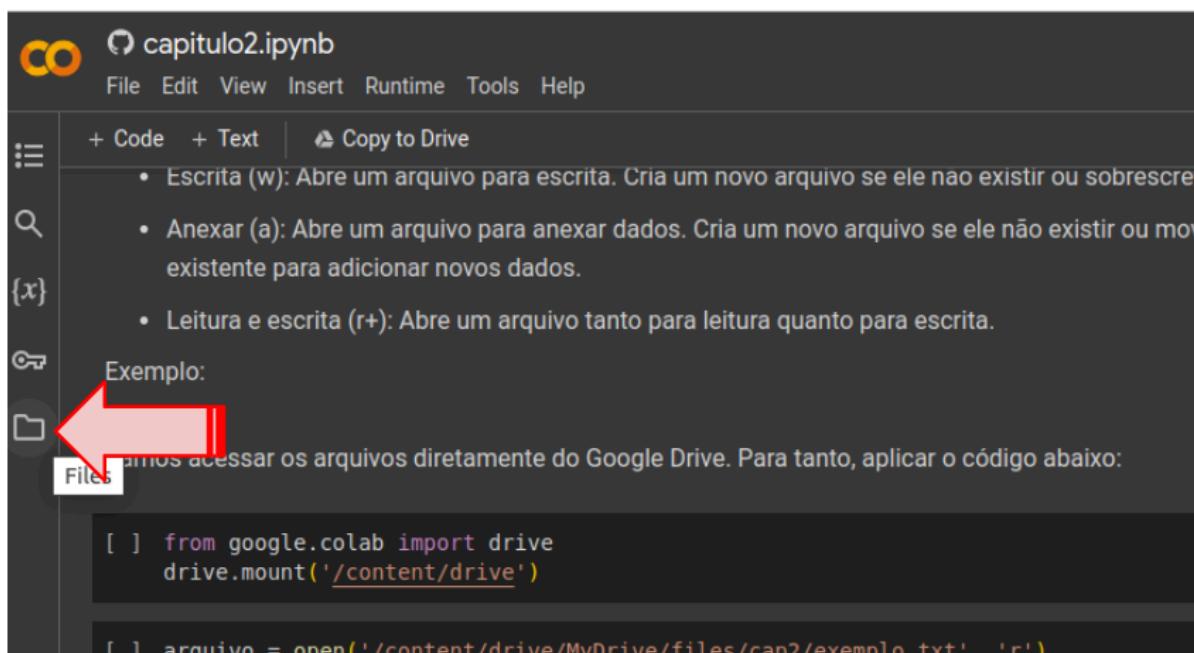
```
x = 7  
y = 12  
x, y = y, x  
print('O valor de x é:', x)  
print('O valor de y é:', y)
```

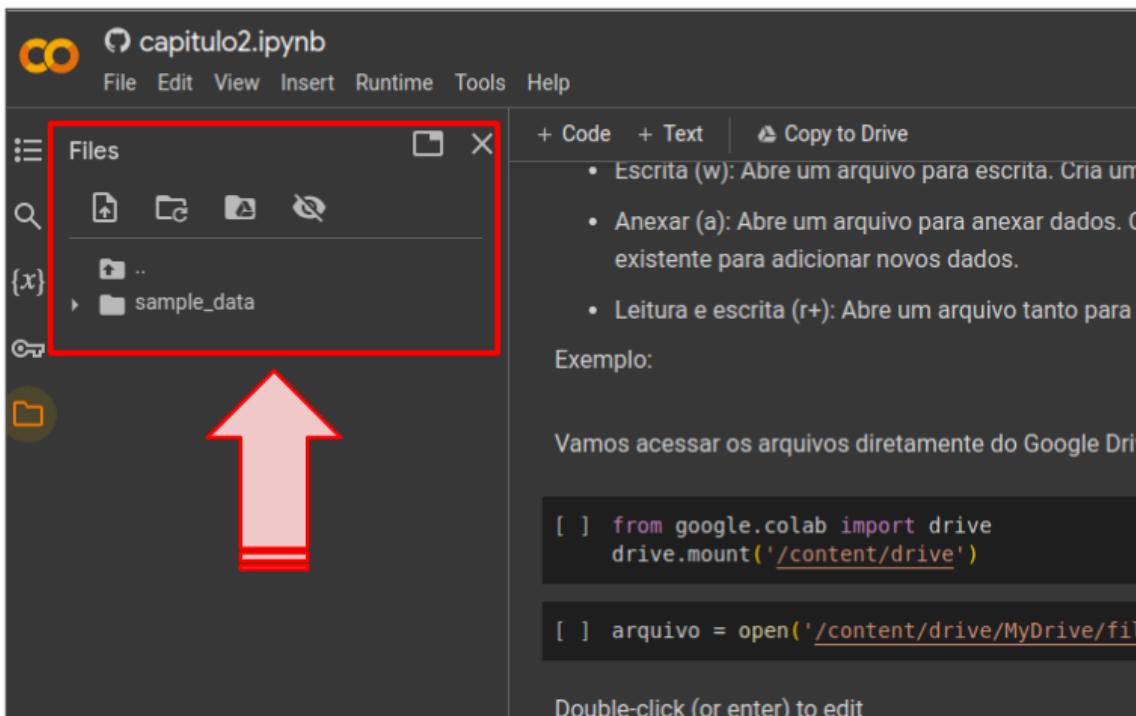
Output:
0 valor de x é: 12
0 valor de y é: 7
- Notebook Sidebar:** Includes a search icon, a cell list, and a folder icon.
- Footnote:** At the bottom, there is a note: "Atribuição múltipla torna o código mais conciso e, em muitos casos, mais legível, especialmente quando usada de maneira apropriada e não excessiva."

5. TRABALHANDO COM ARQUIVOS EXTERNOS NO GOOGLE COLAB

5.1 Gerenciamento de arquivos e diretórios no Google Colab

No Google Colaboratory, o ambiente "Files" (Arquivos) é uma interface integrada que permite aos usuários gerenciar arquivos e diretórios dentro de seus notebooks. Esse ambiente facilita o upload, o download e a organização de arquivos.





Dentro do ambiente “File”, os usuários têm à disposição uma visão clara dos arquivos atualmente disponíveis no ambiente de execução do seu notebook. Isso facilita a identificação e o acesso a *scripts*, conjuntos de dados e outros recursos necessários para o desenvolvimento de projetos de análise de dados e de programação.

A interface permite o gerenciamento de arquivos, incluindo a capacidade de mover, renomear ou excluir arquivos dentro do ambiente de execução.

A funcionalidade de upload permite aos usuários transferir arquivos do seu computador local para o ambiente Colab, possibilitando a utilização de dados pessoais ou específicos do projeto. Da mesma forma, é possível baixar arquivos do ambiente Colab para o computador local, facilitando a exportação de resultados, dados processados ou scripts desenvolvidos durante a sessão.

Ao utilizar o Google Colaboratory para o desenvolvimento de seus projetos, é importante estar ciente das limitações relacionadas à persistência de dados no ambiente de execução. Os ambientes de execução do Colab são efêmeros, o que significa que qualquer dado armazenado temporariamente será perdido ao final da sessão, ou quando o ambiente for reiniciado. Isso inclui arquivos carregados diretamente no ambiente de execução e quaisquer dados gerados durante a sessão.

5.2 Montagem do Drive

Existem diferentes métodos para incorporar arquivos externos ao seu ambiente de trabalho no Google Colaboratory, cada um adequado a diferentes necessidades e escalas de projeto.

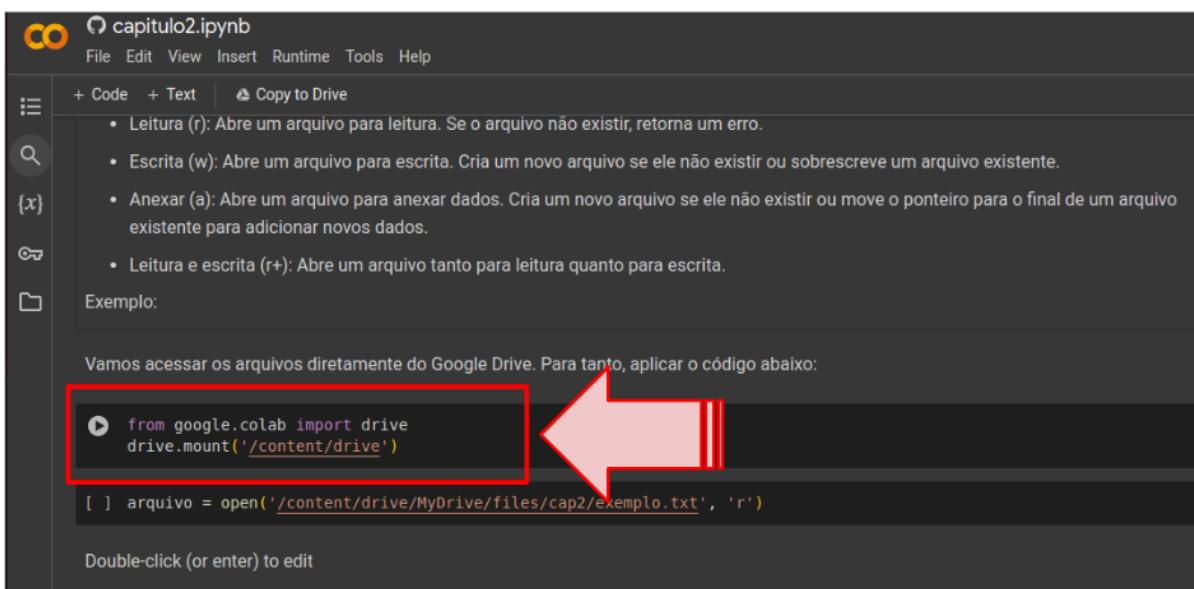
- a) Upload de arquivos: Para arquivos de tamanho reduzido, o método mais direto é o upload para o ambiente de trabalho. Essa abordagem é rápida e eficiente para casos de uso que envolvem conjuntos de dados pequenos ou arquivos de configuração específicos do projeto;
- b) Montar Google Drive: Para acessar arquivos maiores ou garantir que eles permaneçam disponíveis por um período prolongado, a montagem do Google Drive no ambiente Colab é a solução recomendada. Esse método permite o acesso direto aos arquivos armazenados no seu Google Drive, facilitando a manipulação de conjuntos de dados grandes sem necessidade de *uploads* frequentes. Além disso, oferece a conveniência de manter todos os seus arquivos importantes em um local centralizado e acessível;
- c) Uso do GitHub: Quando o trabalho envolve projetos colaborativos ou a necessidade de manter um rigoroso controle de versão, integrar arquivos a partir do GitHub torna-se uma opção interessante. Utilizar o GitHub permite não apenas acessar a versão mais atualizada dos arquivos do projeto, mas também contribuir com projetos de código aberto ou gerenciar as mudanças no código de maneira estruturada e transparente;
- d) Acessar arquivos via APIs: Em situações em que é necessário acessar dados dinâmicos de APIs ou servidores remotos, o uso de requisições HTTP é o método apropriado. Este processo envolve enviar requisições HTTP para APIs externas para recuperar dados em tempo real, permitindo a integração de informações atualizadas diretamente no seu ambiente de programação.

Em nosso curso, optamos por trabalhar com a montagem do Google Drive.

Montar o Google Drive no Google Colaboratory significa estabelecer uma conexão direta entre o seu ambiente de notebooks Colab e seu Google Drive. Essa ação permite que você acesse, leia e escreva arquivos armazenados no seu Drive diretamente a partir dos notebooks do Colab. Essa integração facilita a manipulação de conjuntos de dados grandes, o armazenamento persistente de arquivos e a colaboração em projetos.

A montagem é feita por meio de um código específico no notebook do Colab, que geralmente solicita que você autorize o acesso do Colab ao seu Google Drive. Após essa autorização, uma pasta do Drive é vinculada ao ambiente do Colab, e você pode começar a acessar seus arquivos diretamente no notebook.

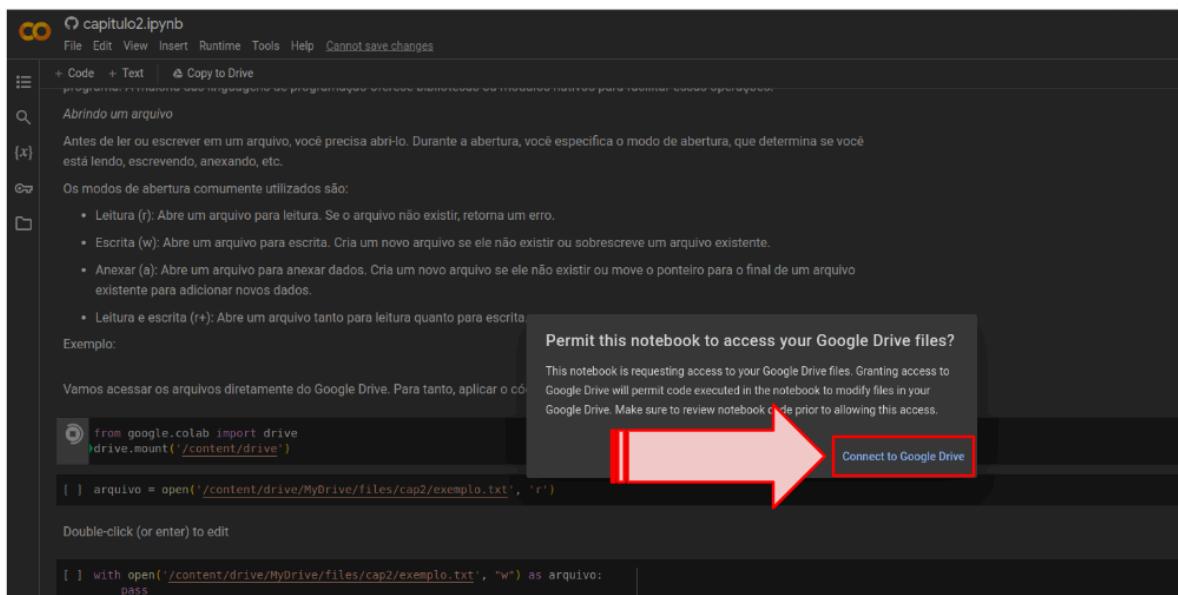
No notebook do capítulo 2, encontre o código de montagem do Google Drive:



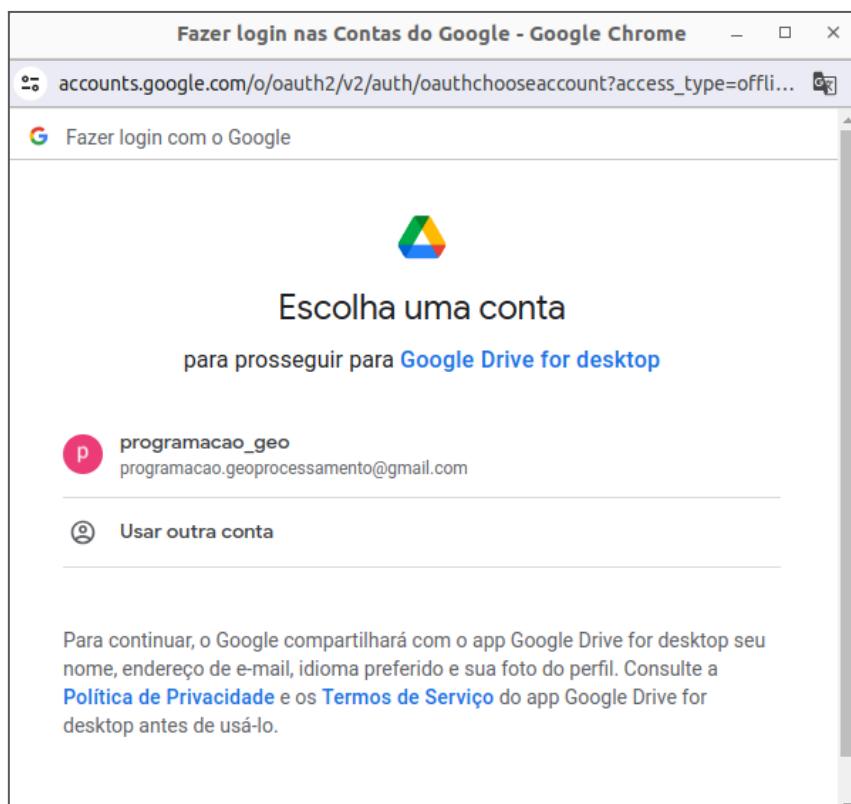
```
from google.colab import drive
drive.mount('/content/drive')

[ ] arquivo = open('/content/drive/MyDrive/files/cap2/exemplo.txt', 'r')
```

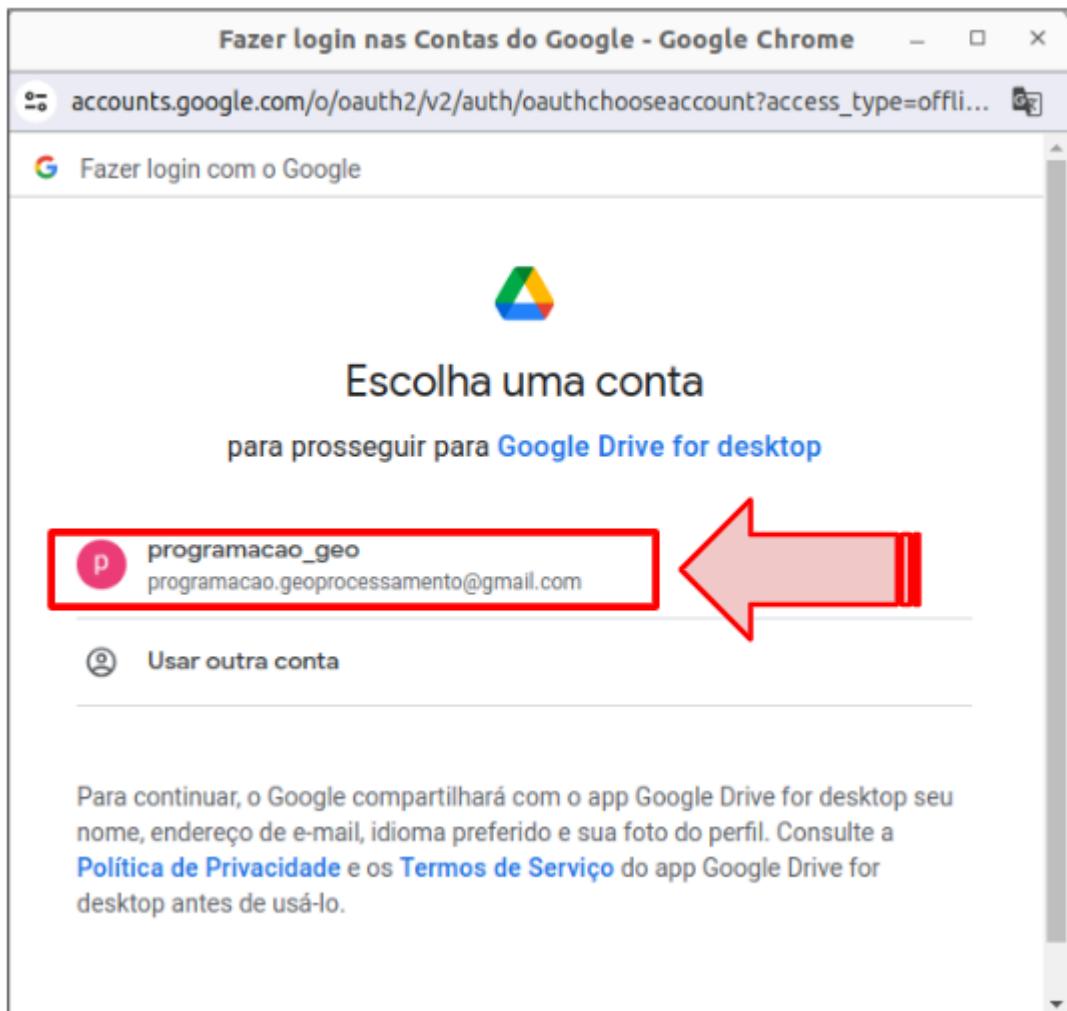
Na janela que abrirá, clique em “Connect to Google Drive”:



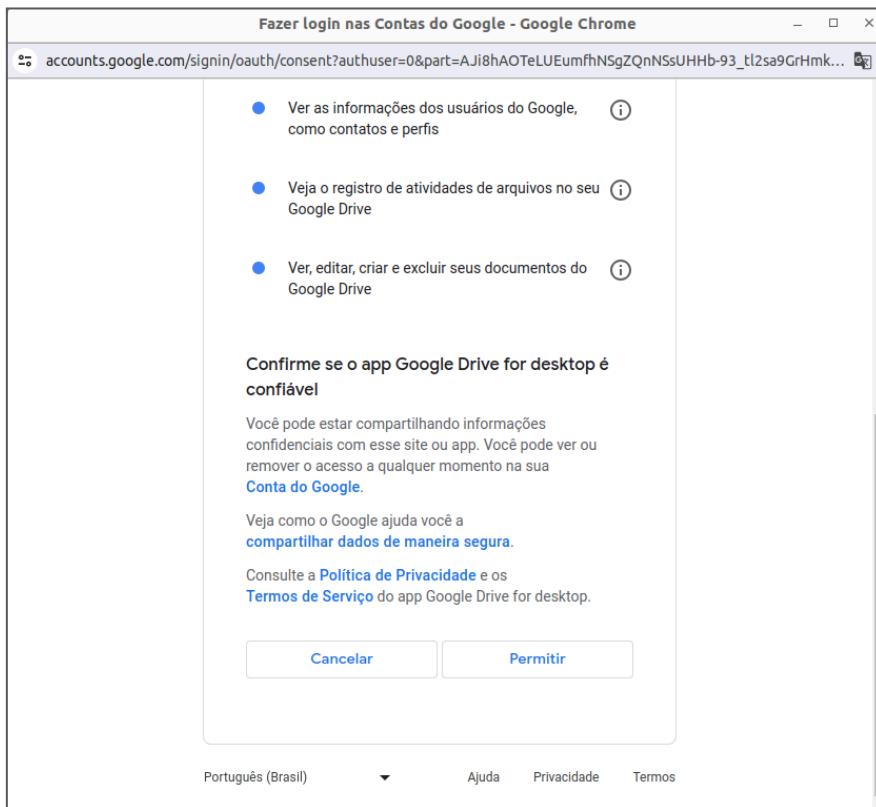
Abrirá uma janela solicitando o *login* em sua conta do Google. Caso você ainda não tenha uma conta, aproveite para criar uma.



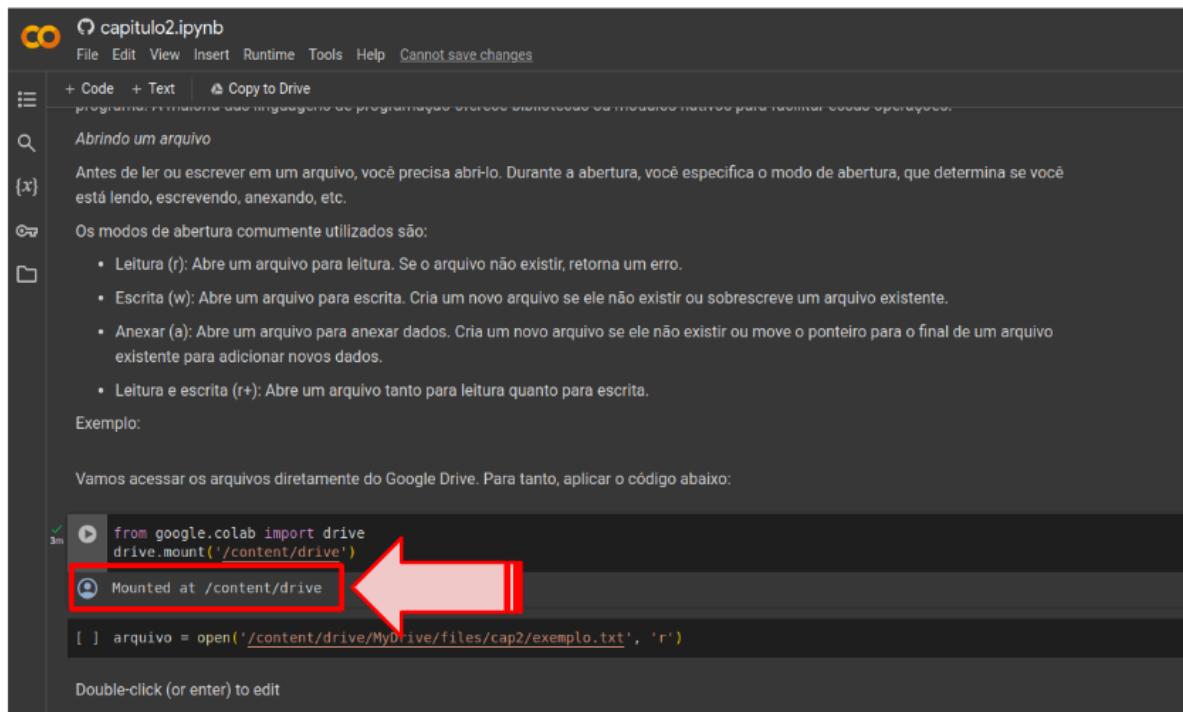
Selecione a sua conta do google:



Clique em “Permitir”:



Após a permissão concedida, o drive se encontrará montado em seu notebook.

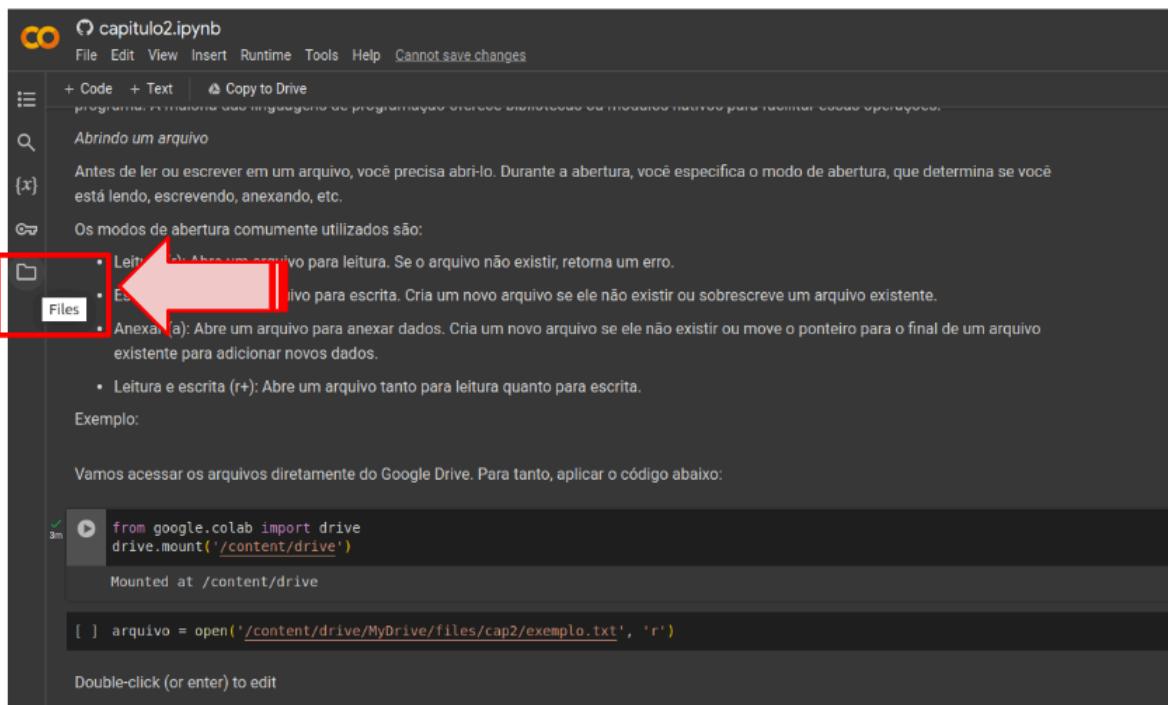


The screenshot shows a Google Colab notebook titled "capítulo2.ipynb". The code cell contains the following Python code:from google.colab import drive
drive.mount('/content/drive')

```
3m [ ]
```

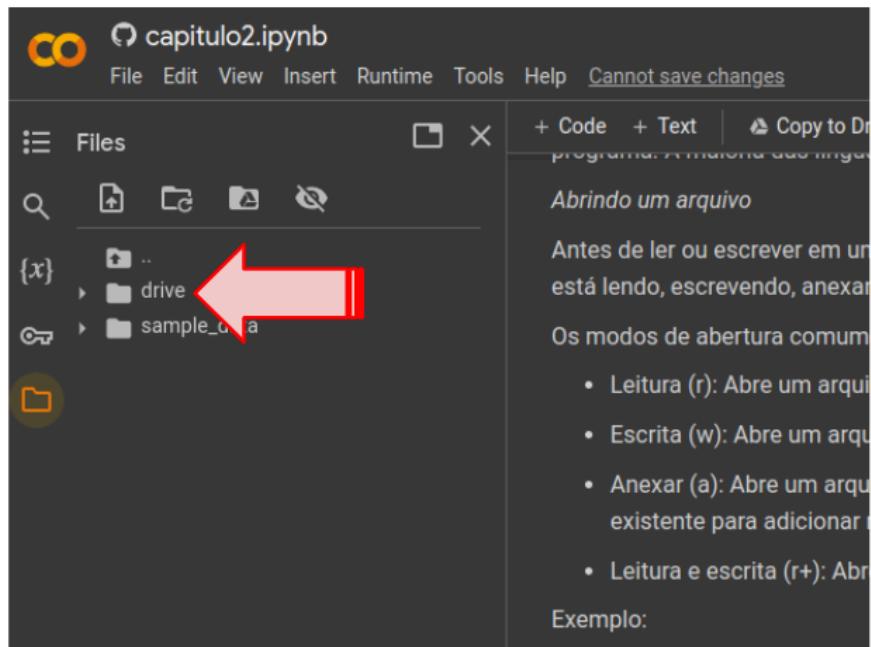
A red arrow points from the text "Mounted at /content/drive" back to the line of code where it was printed. Below the code cell, there is a message: "Double-click (or enter) to edit".

Para verificar se o drive foi montado corretamente, acesse o ambiente “Files”:

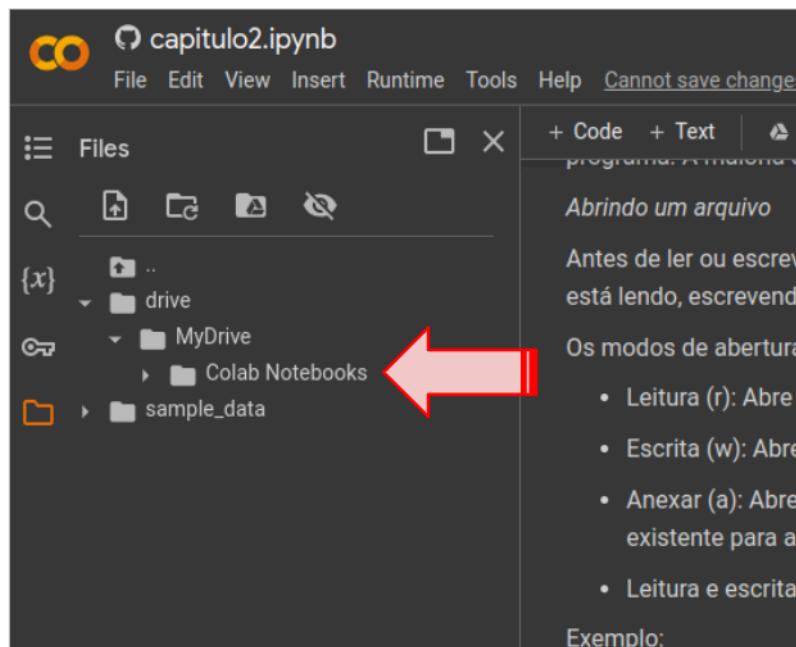


The screenshot shows a Google Colab notebook titled "capítulo2.ipynb". The sidebar on the left has a "Files" icon highlighted with a red box and a red arrow pointing to it. The main content area is identical to the previous screenshot, showing the mounted drive information and the code cell for mounting Google Drive.

Caso o processo tenha ocorrido conforme o esperado, você verá a pasta “drive” no ambiente “Files”:

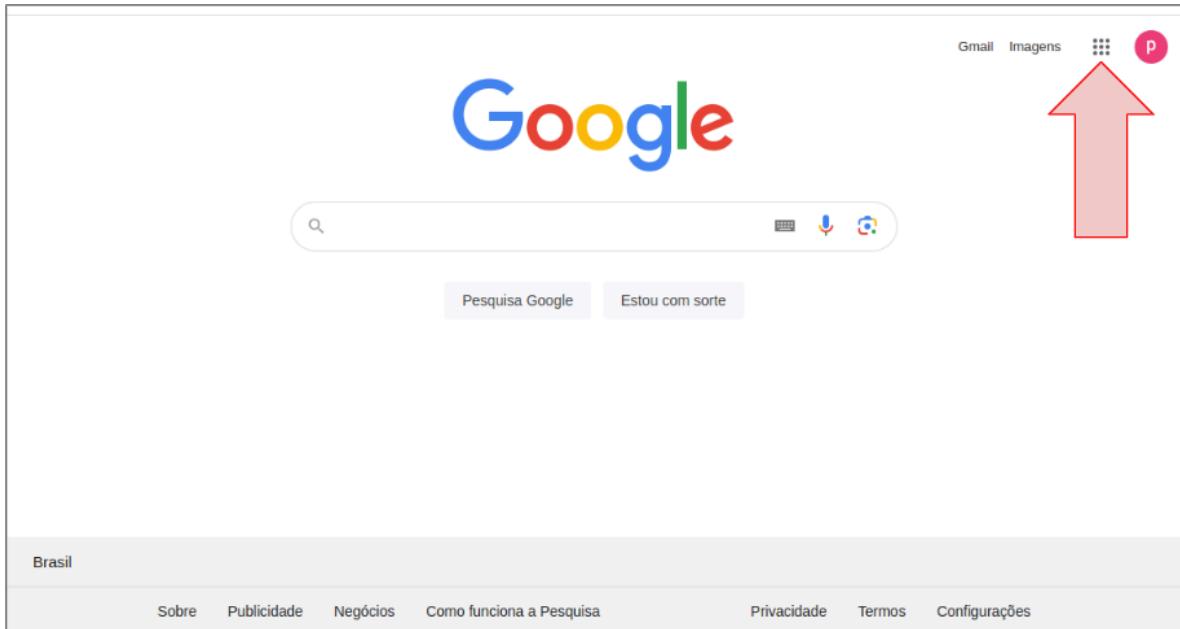


Clique nessa pasta e, em seguida, na pasta “MyDrive”. Dentro dessa pasta, haverá a pasta “Colab Notebooks”, em que os arquivos externos serão armazenados.

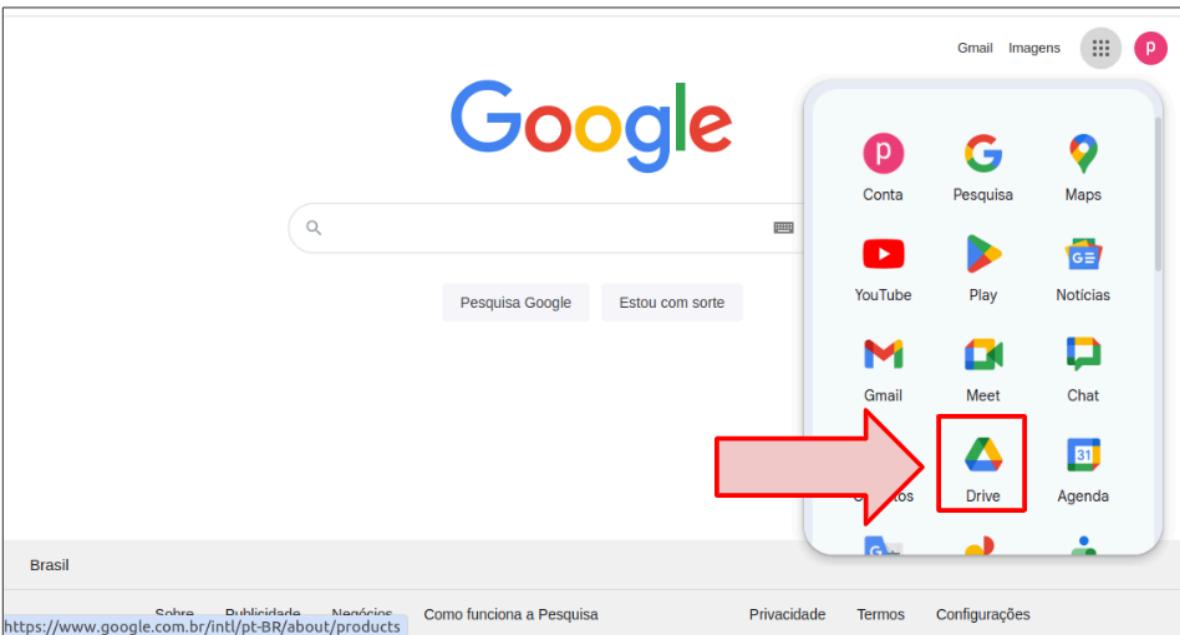


Você pode visualizar as pastas criadas no seu ambiente do Google Drive.

Para isso, acesse o Google Chrome:



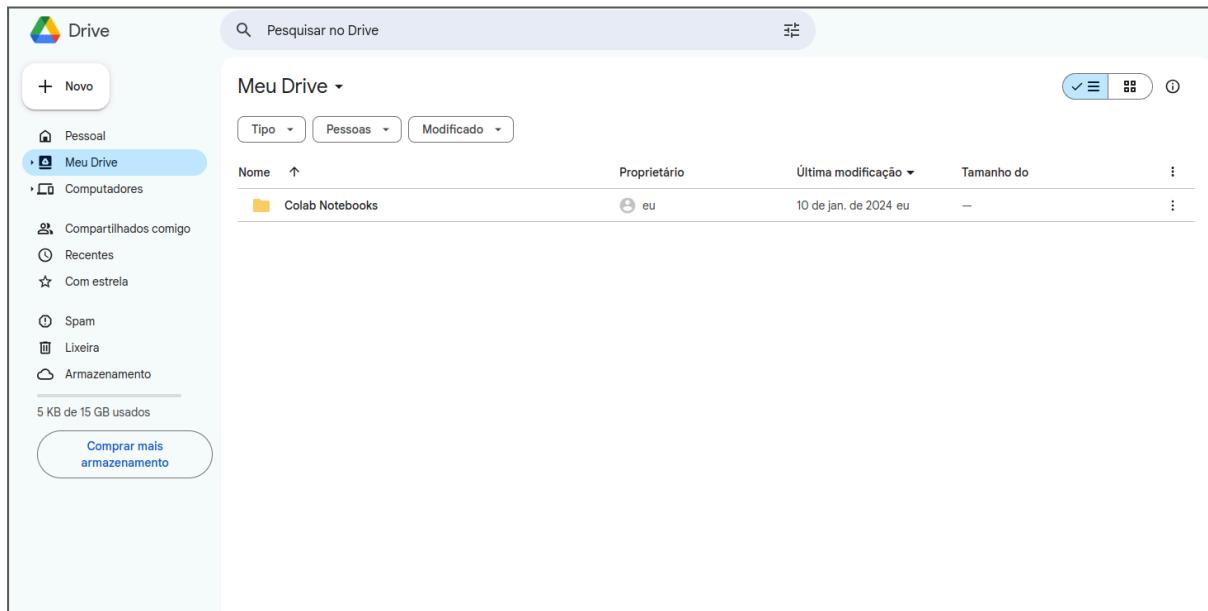
Acesse o Google Drive:



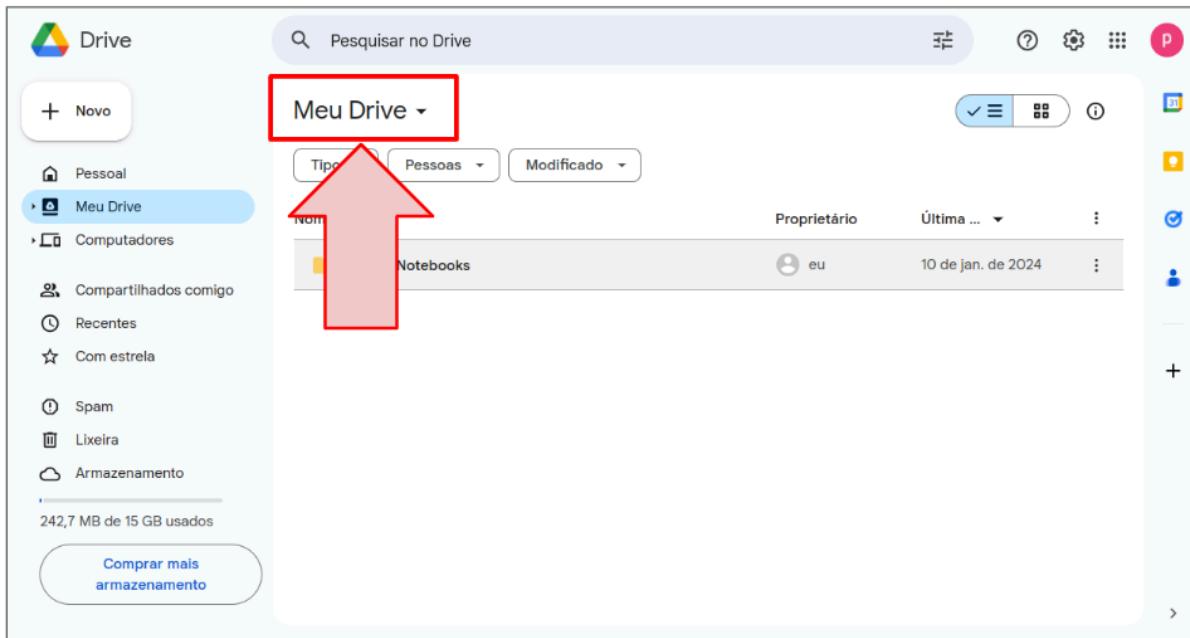
Clique em “Meu Drive”:



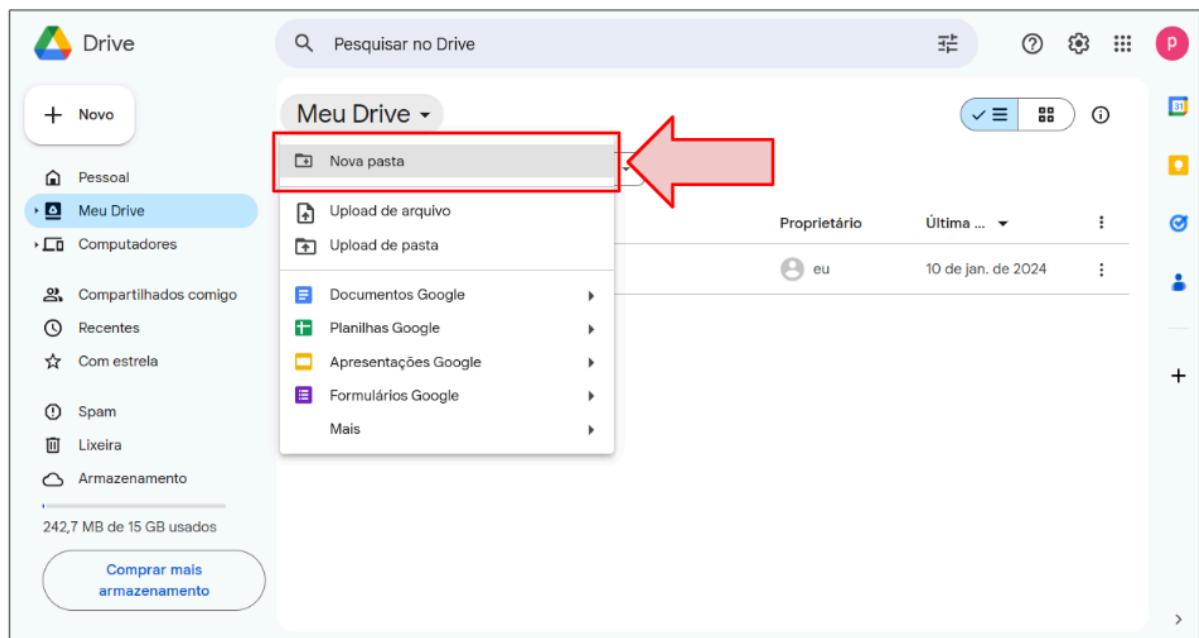
A pasta “Colab Notebooks” estará no seu Drive:



Vamos criar uma nova pasta, dentro da pasta “Colab Notebooks” para armazenarmos os arquivos externos de nosso curso. Para tanto, clique em “Meu Drive”:

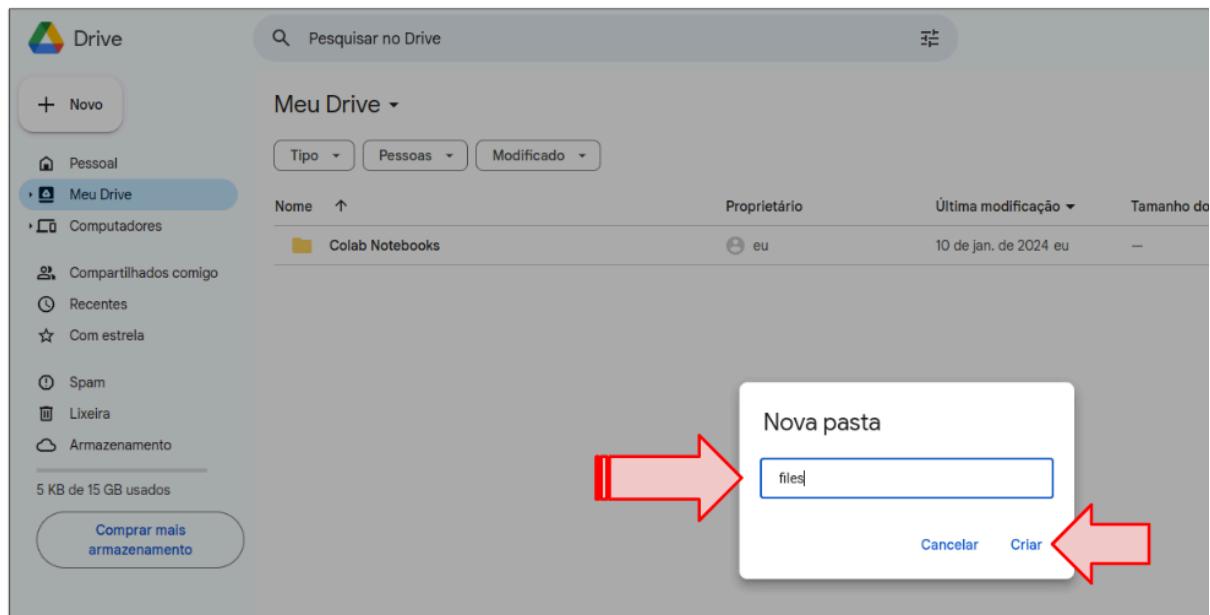


Clique em “Nova Pasta”:

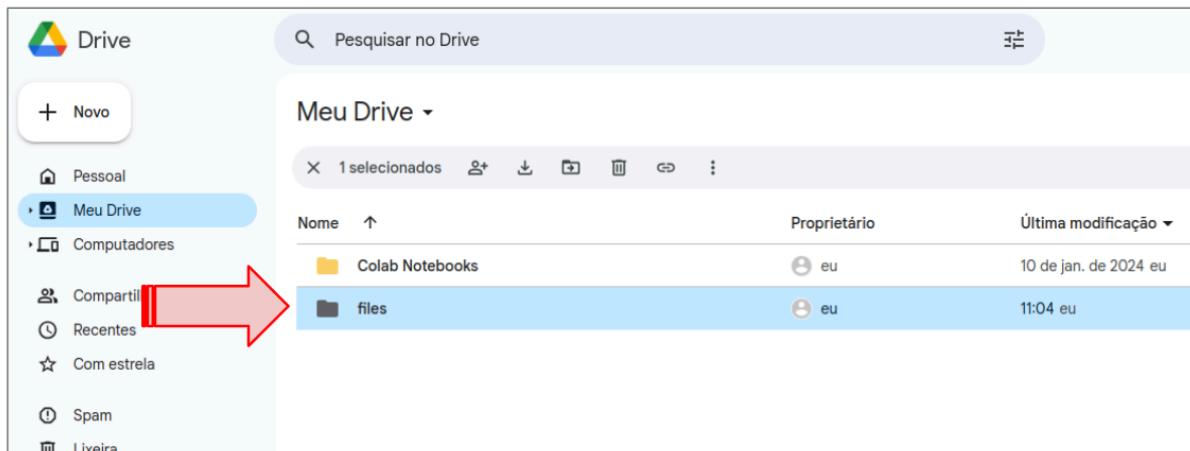


Informe o nome da pasta. A pasta deve se chamar “files” para que os códigos que estão em nossos notebooks funcionem sem precisar realizar alterações. Cuide com a grafia do nome do arquivo, que é todo em letras minúsculas.

Logo após, clique em “criar”:



Após criar a pasta “files”, seu ambiente do “Meu Drive” estará conforme a imagem abaixo:



5.3 Organização das pastas do curso no Google Drive

Inicialmente vamos realizar o download dos arquivos que estão disponíveis no repositório Programacao-Geoprocessamento no GitHub. Acesse o repositório no endereço:

<https://github.com/Alexandrogschafer/Programacao-Geoprocessamento/tree/gh-pages/files>

Na página do repositório, identifique os arquivos compactados (.zip). São esses arquivos que devem ser baixados:

The screenshot shows the GitHub repository interface for 'Programacao-Geoprocessamento'. On the left, there's a sidebar with a tree view of the repository structure, including 'ipynb_checkpoints', 'images', 'sources', 'sphinx_design_static', 'static', and a 'files' folder. Inside 'files', there are subfolders for 'cap2', 'cap3', 'cap4/f4', 'cap5', 'cap6', and 'cap6'. Under 'cap6', there are several zip files: 'cap2.zip', 'cap3.zip', 'cap4.zip', 'cap5.1.zip', 'cap5.2.zip', 'cap5.3.zip', 'cap6.1.zip', 'cap6.2.zip', and 'cap6.3.zip'. A red arrow points from the left sidebar towards this list. To the right of the sidebar is a table showing commit history for each file. The commits are as follows:

Name	Last commit message
..	
cap2	Update documentation
cap3	Update documentation
cap4/f4	Update documentation
cap5	Update documentation
cap6	Update documentation
~lock.paises.xlsx#	Update documentation
cap2.zip	Add files in .zip format
cap3.zip	Add files in .zip format
cap4.zip	Add files in .zip format
cap5.1.zip	Add zip shapefiles cap 5 and 6
cap5.2.zip	Adicionando arquivos cap5_2 e cap6_2
cap5.3.zip	Add zip shapefiles cap 5 and 6
cap6.1.zip	Add zip shapefiles cap 5 and 6
cap6.2.zip	Adicionando arquivos cap5_2 e cap6_2
cap6.3.zip	Add zip shapefiles cap 5 and 6

Vamos baixar o arquivo cap2.zip. Clique no arquivo:

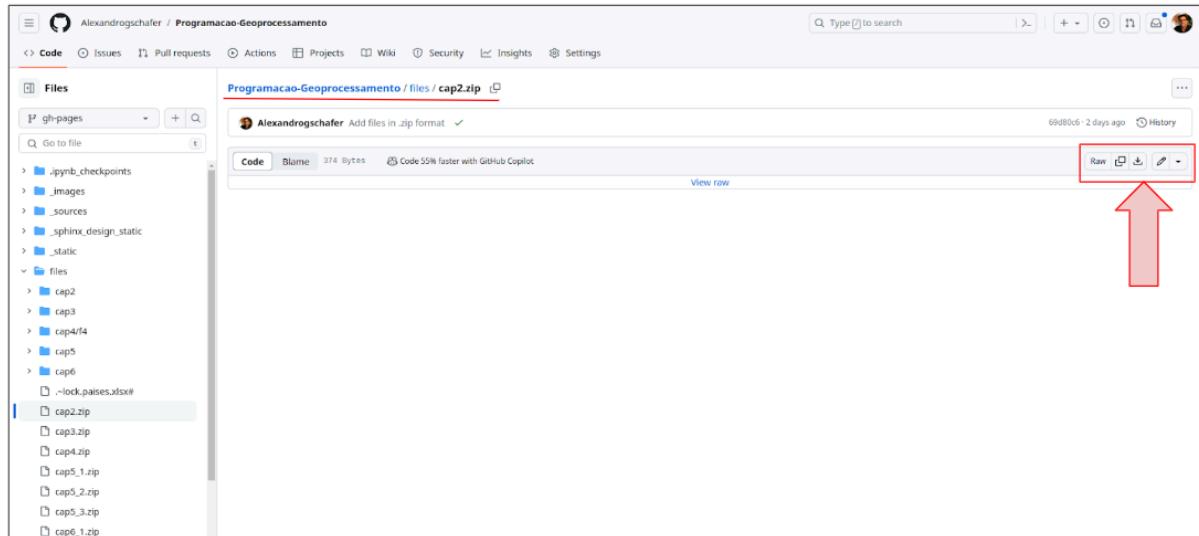
A screenshot of a GitHub repository interface. On the left, there's a sidebar with navigation links like 'Files', 'gh-pages', and a search bar. The main area shows a list of files under the commit 'Adicionando arquivos cap5_2 e cap6_2'. One file, 'cap2.zip', is highlighted with a red arrow pointing to it from the left.

Name	Last commit message
..	
cap2	Update documentation
cap3	Update documentation
cap4/f4	Update documentation
cap5	Update documentation
cap6	Update documentation
.-lock.paises.xlsx#	Update documentation
cap2.zip	Add files in .zip format
cap3.zip	Add files in .zip format
cap4.zip	Add files in .zip format
cap5_1.zip	Add zip shapefiles cap 5 and 6
cap5_2.zip	Adicionando arquivos cap5_2 e cap6_2
cap5_3.zip	Add zip shapefiles cap 5 and 6
cap6_1.zip	Add zip shapefiles cap 5 and 6
cap6_2.zip	Adicionando arquivos cap5_2 e cap6_2
cap6_3.zip	Add zip shapefiles cap 5 and 6

O arquivo foi aberto. Verifique o caminho destacado (em vermelho) na figura abaixo:

A screenshot of the GitHub file viewer for 'cap2.zip'. The URL 'Programacao-Geoprocessamento / files / cap2.zip' is highlighted with a red box. The page shows basic file metadata: 'Code' (374 Bytes), 'Blame', and a note about GitHub Copilot.

Identifique o ícone para download de arquivos, no canto direito da área de trabalho:



Clique no ícone e o download iniciará automaticamente. O arquivo estará em seu computador, na pasta “Downloads” ou na pasta que você especificou como padrão para receber downloads de arquivos.

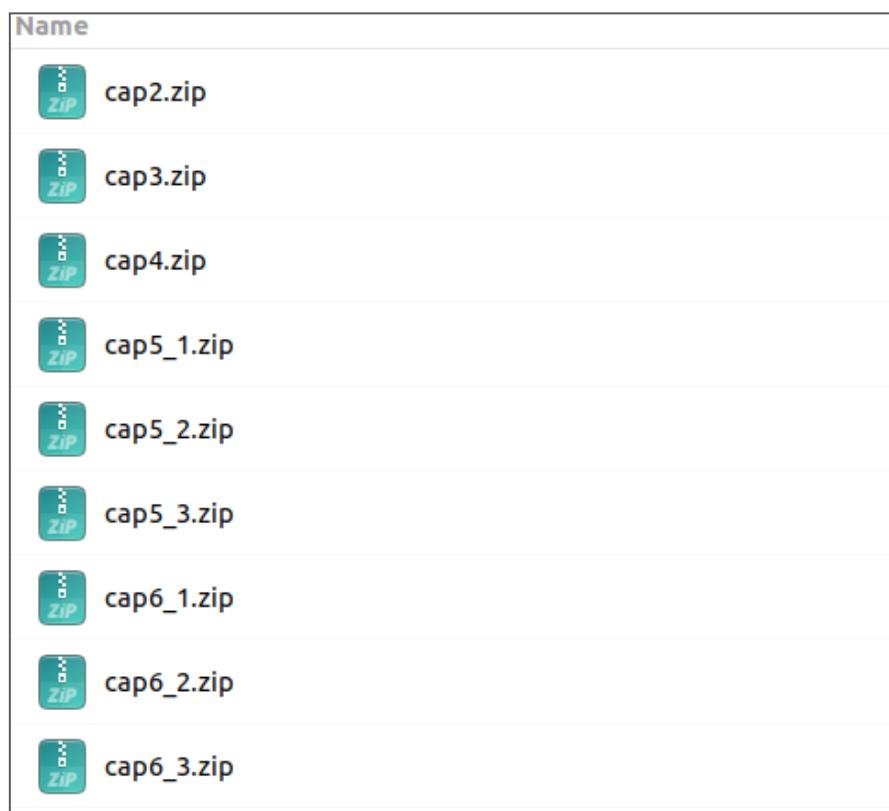


Repita esse procedimento para os arquivos:

- cap3.zip;

- cap4.zip;
- cap5_1.zip;
- cap5_2.zip;
- cap5_3.zip;
- cap6_1.zip;
- cap6_2.zip;
- cap6_3.zip.

Após o download de todos os arquivos, acesse a pasta em que eles se encontram e os descompacte.

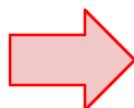


Cada arquivo descompactado dará origem a uma pasta, que conterá os arquivos que utilizaremos em nossos notebooks.

Name
cap2
cap3
cap4
cap5_1
cap5_2
cap5_3
cap6_1
cap6_2
cap6_3

Os arquivos dos capítulos 5 e 6 foram divididos em várias pastas, devido a limitação de tamanho de arquivo do GitHub. Vamos reorganizar essas pastas.

Name
cap2
cap3
cap4
cap5_1
cap5_2
cap5_3
cap6_1
cap6_2
cap6_3



Vamos criar uma pasta única para todos os arquivos do capítulo 5. Em seguida, copiaremos os arquivos de “**cap5_1**”, “**cap5_2**” e “**cap5_3**” para a nova pasta, denominada “**cap5**”.

Verifique o conteúdo de cada pasta do capítulo 5. A figura abaixo apresenta o conteúdo da pasta cap5_1:

Name	Size	Modified	
f4	7 items	17 out 2023	☆
f5	1 item	19 out 2023	☆
f6	1 item	25 nov 2023	☆
f7	1 item	19 out 2023	☆
f8	6 items	2 dez 2023	☆

Conteúdo da pasta cap5_2:

Name	Size	Modified	
f9	7 items	21 out 2023	☆

Conteúdo da pasta cap5_3:

Name	Size	Modified	
f10	35 items	2 dez 2023	☆
f11	7 items	21 out 2023	☆
f12	1 item	21 out 2023	☆
f13	1 item	20 out 2023	☆
f14	1 item	20 out 2023	☆
f15	7 items	19 out 2023	☆
f16	7 items	19 out 2023	☆
f17	7 items	21 out 2023	☆
f18	7 items	21 out 2023	☆
f19	5 items	11 dez 2023	☆

Crie a pasta denominada cap5 e copie ou move todos os arquivos de cap5_1, cap5_2 e cap5_3 para essa nova pasta. O seu conteúdo final deve ser:

Name	Size	Modified
f4	7 items	17 out 2023
f5	1 item	19 out 2023
f6	1 item	25 nov 2023
f7	1 item	19 out 2023
f8	6 items	2 dez 2023
f9	7 items	21 out 2023
f10	35 items	2 dez 2023
f11	7 items	21 out 2023
f12	1 item	21 out 2023
f13	1 item	20 out 2023
f14	1 item	20 out 2023
f15	7 items	11:41
f16	7 items	19 out 2023
f17	7 items	21 out 2023
f18	7 items	21 out 2023
f19	5 items	11 dez 2023

Vamos realizar o mesmo procedimento para as pastas do capítulo 6.

Name
cap2
cap3
cap4
cap5_1
cap5_2
cap5_3
cap6_1
cap6_2
cap6_3



Vamos criar uma pasta única para todos os arquivos do capítulo 6. Em seguida, copiaremos os arquivos de “**cap6_1**”, “**cap6_2**” e “**cap6_3**” para a nova pasta, denominada “**cap6**”.

Conteúdo da pasta cap6_1:

Name	Size	Modified	
f4	7 items	17 out 2023	☆
f18	7 items	21 out 2023	☆
f20	1 item	21 out 2023	☆
f21	15 items	22 out 2023	☆

Conteúdo da pasta cap6_2:

Name	Size	Modified	
f22	7 items	1 dez 2023	☆

Conteúdo da pasta cap6_3:

Name	Size	Modified	
f23	17 items	8 dez 2023	☆
f24	7 items	11 dez 2023	☆
f25	8 items	22 out 2023	☆
f26	7 items	22 out 2023	☆
f27	8 items	22 out 2023	☆

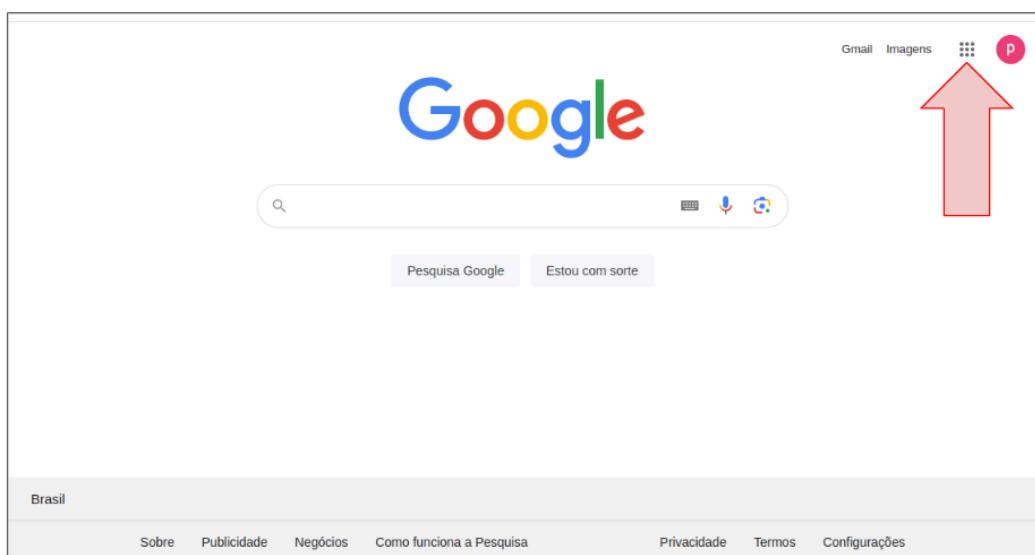
Conteúdo da nova pasta, cap6:

Name	Size	Modified	
f4	7 items	17 out 2023	☆
f18	7 items	21 out 2023	☆
f20	1 item	21 out 2023	☆
f21	15 items	22 out 2023	☆
f22	7 items	1 dez 2023	☆
f23	17 items	8 dez 2023	☆
f24	7 items	11 dez 2023	☆
f25	8 items	22 out 2023	☆
f26	7 items	22 out 2023	☆
f27	8 items	22 out 2023	☆

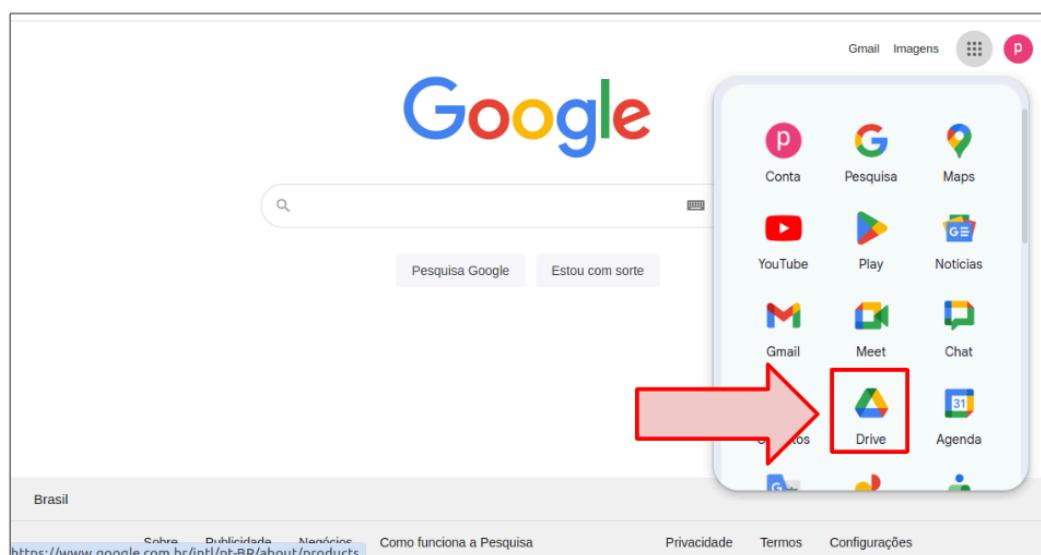
5.4 Upload dos arquivos para o Google Drive

Após fazer o *download* dos arquivos e descompactá-los, a próxima etapa consiste em fazer *upload* desses arquivos para a pasta “files” que você criou anteriormente no seu Google Drive.

Abra o Google Chrome.



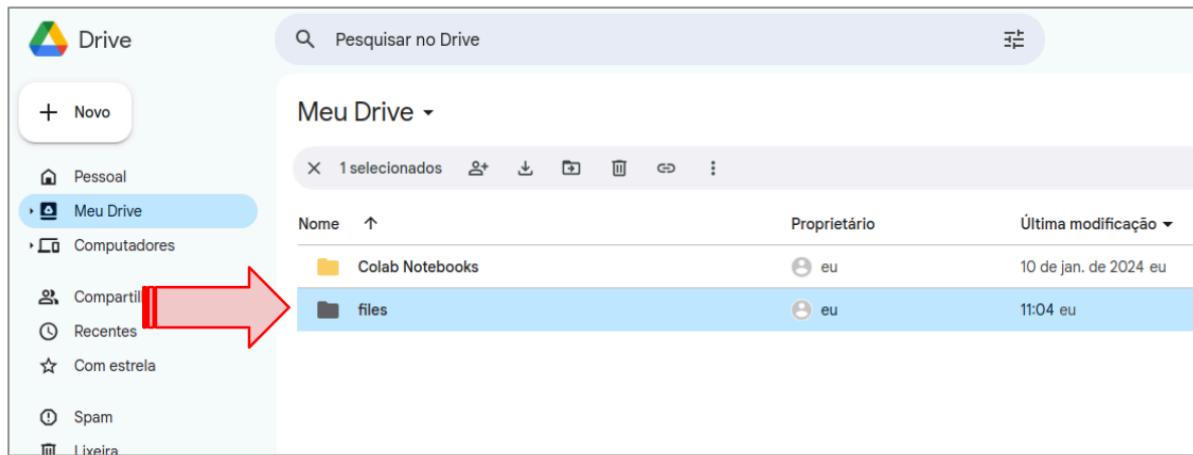
Acesse o Google Drive:



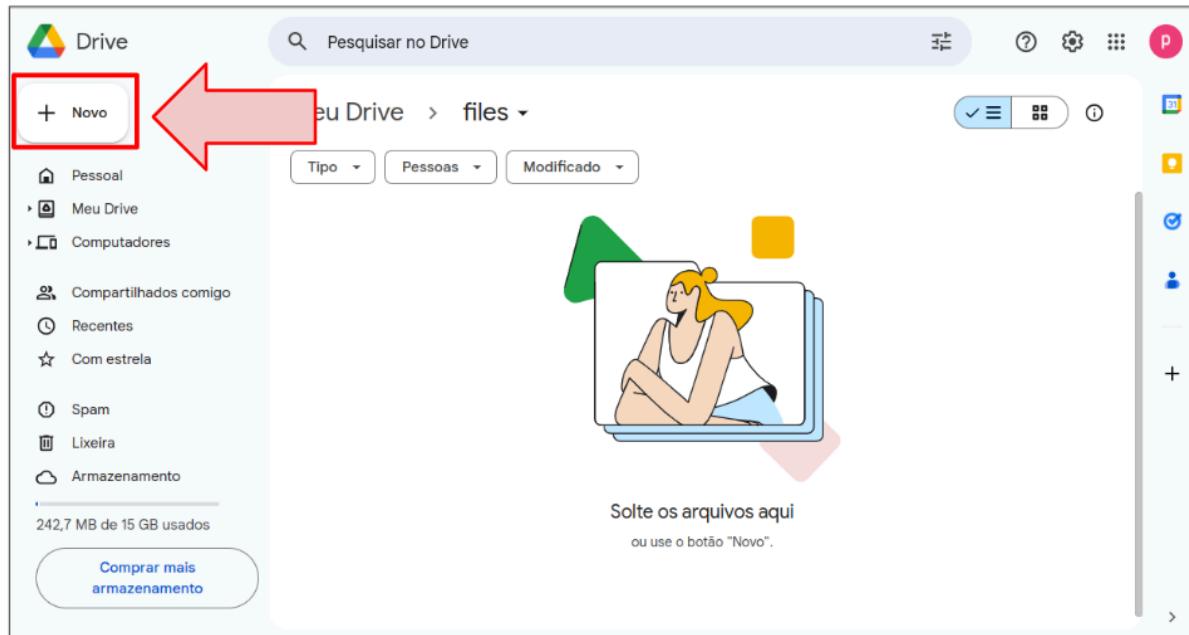
Clique em “Meu Drive”:



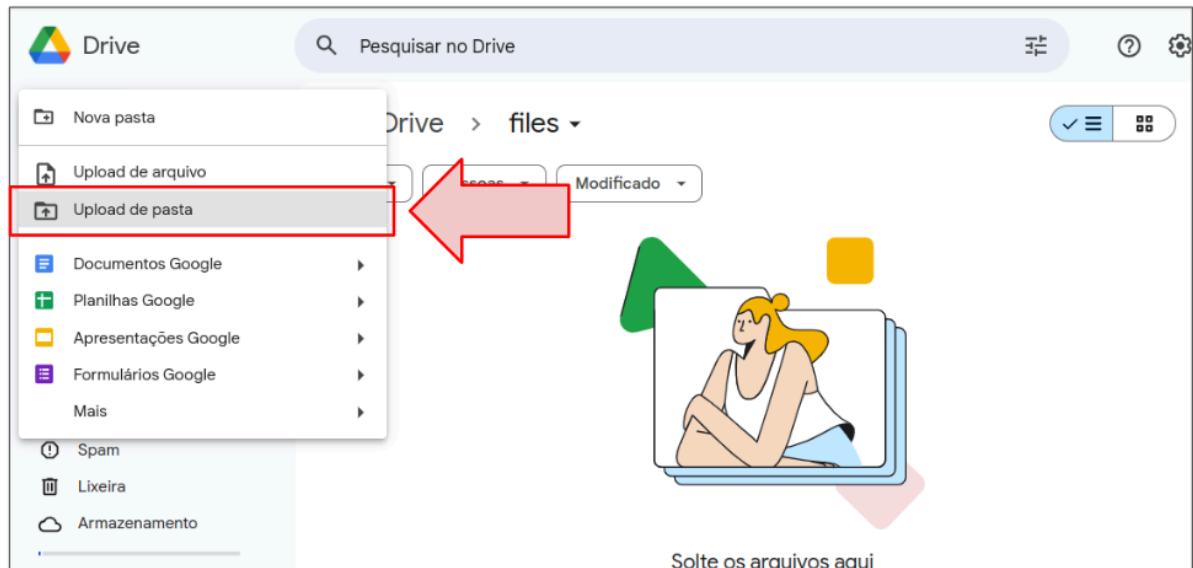
Acesse a pasta files:



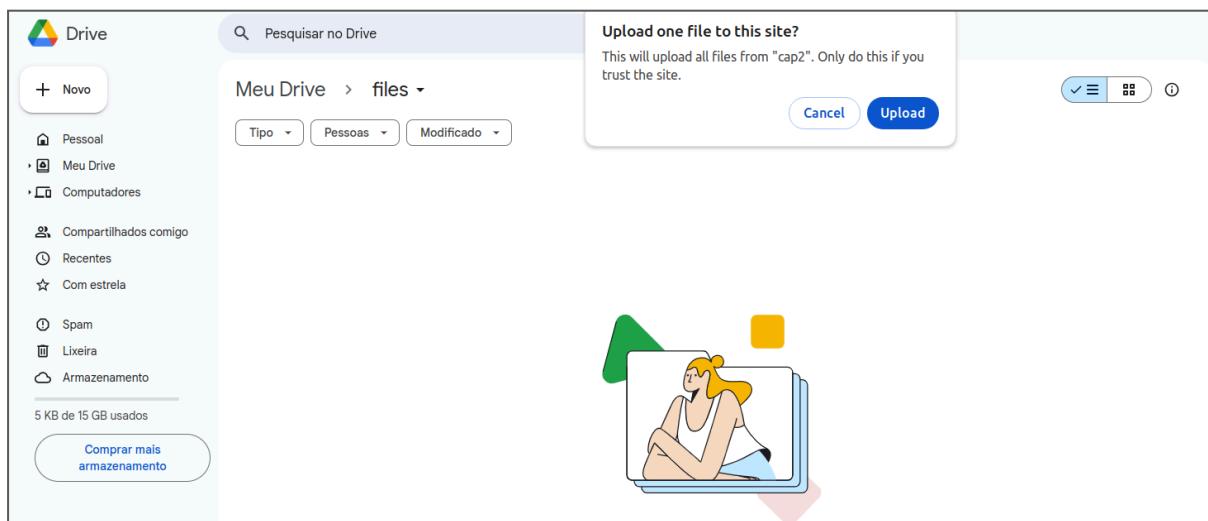
Clique em “Novo”:



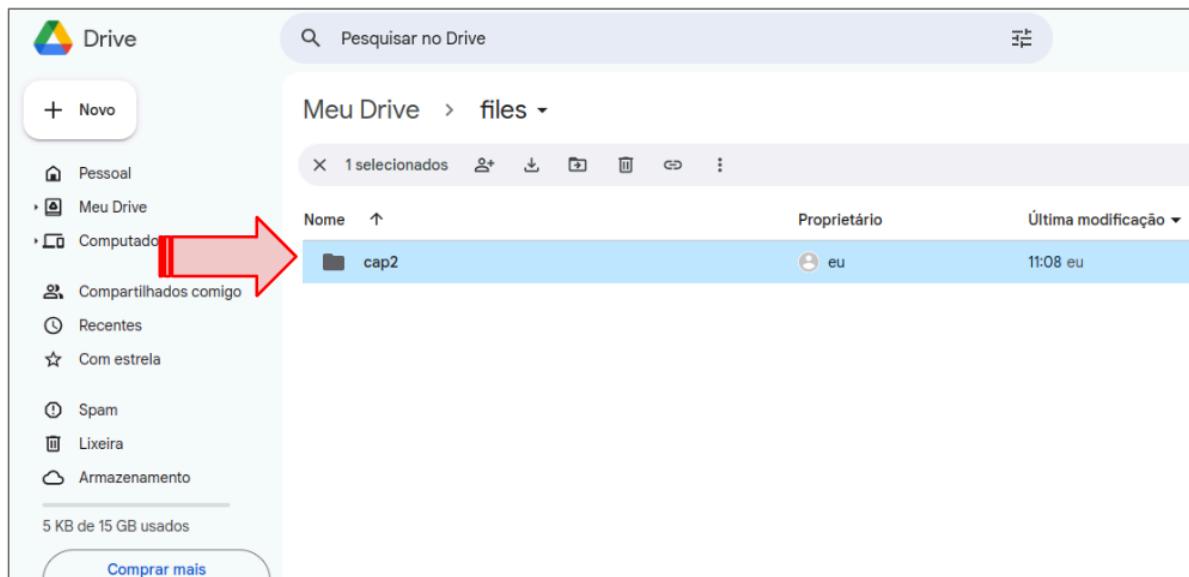
Selecione “Upload de pasta”:



Navegue até a pasta onde estão os arquivos do curso que você baixou.
Selecione “cap2” e clique em upload:



Após a operação, você pode verificar que a pasta cap2 está no google Drive, dentro da pasta “files”.



The screenshot shows the Google Drive interface. On the left, there's a sidebar with links like 'Pessoal', 'Meu Drive', 'Computadores', etc. The main area shows a folder named 'cap2' in the 'files' folder. A red box highlights the text: 'Repita o procedimento para as pastas "cap3", "cap4", "cap5" e "cap6"'.

Nome	Proprietário	Última modificação
cap2	eu	11:08 eu

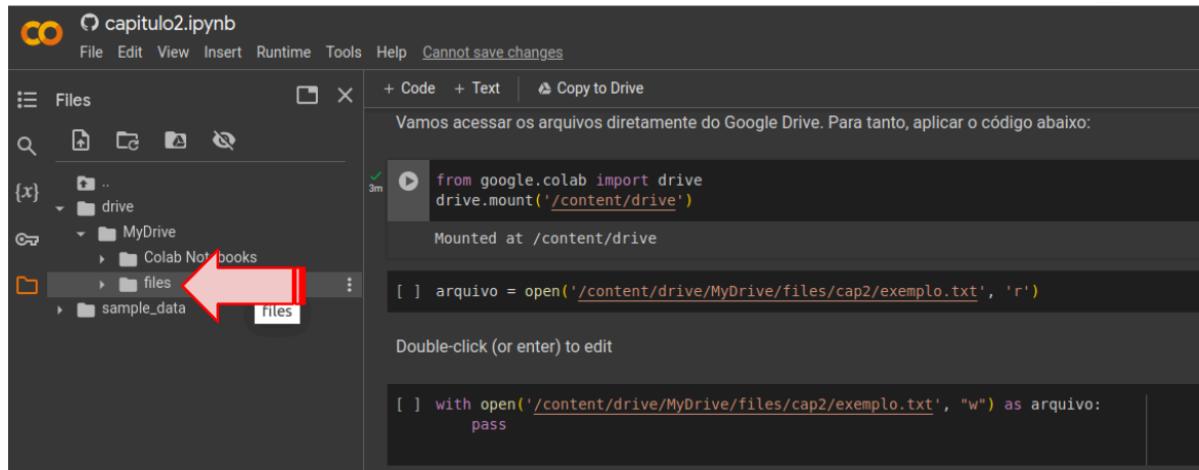
Após realizar o upload de todas as pastas, a sua pasta “files” deve ter o seguinte conteúdo:

The screenshot shows the Google Drive interface with a search bar and filter buttons ('Tipo', 'Pessoas', 'Modificado'). The main area lists six folders: 'cap2', 'cap3', 'cap4', 'cap5', 'cap6', and another 'cap2'. All are owned by 'eu' and were modified at different times between 11:08 and 12:08.

Nome	Proprietário	Última modificação
cap2	eu	11:08 eu
cap3	eu	12:07 eu
cap4	eu	12:08 eu
cap5	eu	12:08 eu
cap6	eu	11:37 eu
cap2	eu	11:08 eu

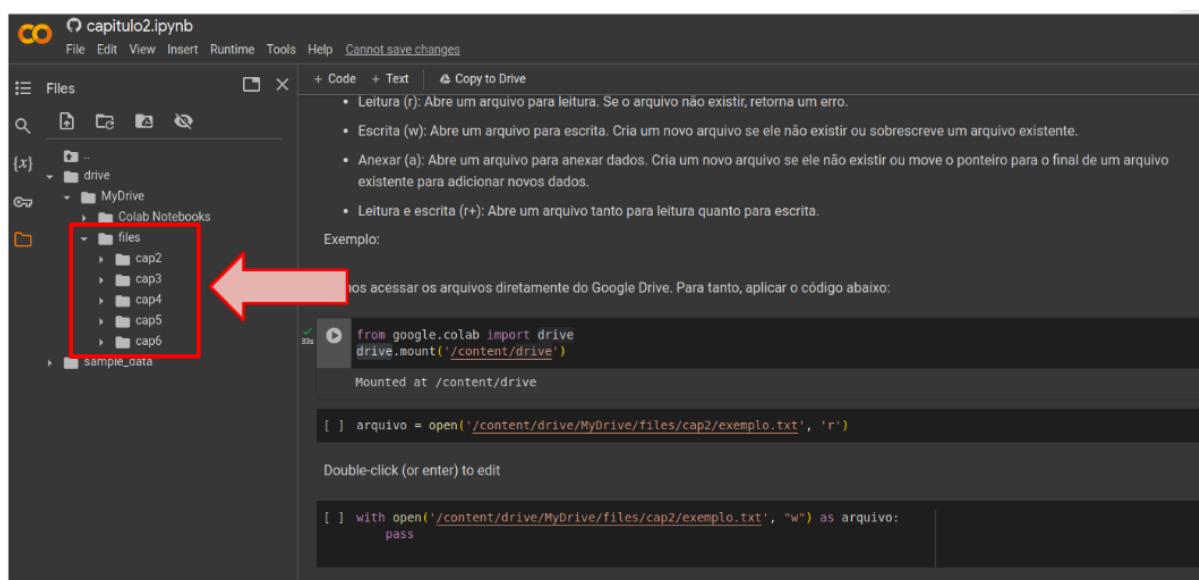
5.5 Verificação das pastas no ambiente “File” do Google Colab

Vamos visualizar as pastas e os arquivos que estão na pasta “files” do Google Drive no ambiente “Files” do Google Colab. Acesse novamente o Google Colab, acesse o ambiente “File”, navegue por drive/Mydrive. Clique na pasta “files”:



The screenshot shows the Google Colab interface with the title "capítulo2.ipynb". In the top menu, "File" is selected. On the left, the "Files" sidebar shows a tree structure: a root folder with "drive" and "MyDrive" (which contains "Colab Notebooks" and "files"). A red arrow points to the "files" folder. The main workspace on the right displays code for mounting Google Drive and opening a file named "exemplo.txt".

Verifique que a pasta foi atualizada com os novos arquivos:



The screenshot shows the Google Colab interface with the title "capítulo2.ipynb". In the top menu, "File" is selected. On the left, the "Files" sidebar shows a tree structure: a root folder with "drive" and "MyDrive" (which contains "Colab Notebooks" and "files"). A red box highlights the "files" folder, and a large red arrow points to it. The main workspace on the right displays code for mounting Google Drive and opening a file named "exemplo.txt".

Verifique o conteúdo das pastas, clicando em cada uma. A figura abaixo apresenta a visualização do conteúdo da pasta cap2:

The screenshot shows the Google Colab interface with a notebook titled 'capítulo2.ipynb'. On the left, there's a 'Files' sidebar showing a directory structure: 'drive' (containing 'MyDrive' and 'files'), 'sample_data', and several 'cap' folders (cap1 through cap6). A red arrow points from the 'files' folder to the 'cap2' folder, which contains a file named 'exemplo.txt'. On the right, a code cell is displayed with the following content:

```
+ Code + Text | ⚙ Copy to Drive
• Leitura (r): Abre um arquivo para leitura. Se o arquivo não existir, retorna um erro.
• Escrita (w): Abre um arquivo para escrita. Cria um novo arquivo se ele não existir ou sobrescreve um arquivo existente.
• Anexar (a): Abre um arquivo para anexar dados. Cria um novo arquivo se ele não existir ou move o ponteiro para o final de um arquivo existente para adicionar novos dados.
• Leitura e escrita (r+): Abre um arquivo tanto para leitura quanto para escrita.

Exemplo:
Vamos acessar os arquivos diretamente do Google Drive. Para tanto, aplicar o código abaixo:

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

Mounted at /content/drive

[ ] arquivo = open('/content/drive/MyDrive/files/cap2/exemplo.txt', 'r')

Double-click (or enter) to edit

[ ] with open('/content/drive/MyDrive/files/cap2/exemplo.txt', "w") as arquivo:
    pass
```

Com os arquivos externos no ambiente do Google Colab, execute a célula:

The screenshot shows the Google Colab interface with the same notebook and file structure. A red arrow points from the 'files' folder to the 'cap2' folder. In the code cell, the run button (a play icon) is highlighted with a red box. The cell content is identical to the previous screenshot:

```
+ Code + Text | ⚙ Copy to Drive
• Leitura (r): Abre um arquivo para leitura. Se o arquivo não existir, retorna um erro.
• Escrita (w): Abre um arquivo para escrita. Cria um novo arquivo se ele não existir ou sobrescreve um arquivo existente.
• Anexar (a): Abre um arquivo para anexar dados. Cria um novo arquivo se ele não existir ou move o ponteiro para o final de um arquivo existente para adicionar novos dados.
• Leitura e escrita (r+): Abre um arquivo tanto para leitura quanto para escrita.

Exemplo:
Vamos acessar os arquivos diretamente do Google Drive. Para tanto, aplicar o código abaixo:

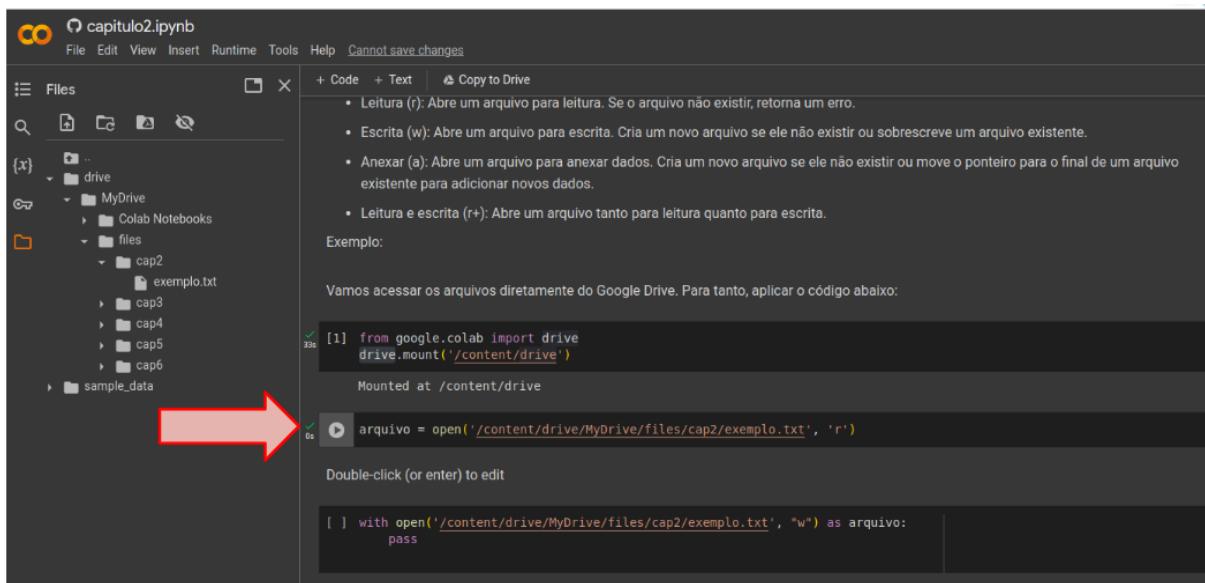
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

[ ] arquivo = open('/content/drive/MyDrive/files/cap2/exemplo.txt', 'r')

Run cell (Ctrl+Enter)
Do cell has not been executed in this session

[ ] with open('/content/drive/MyDrive/files/cap2/exemplo.txt', "w") as arquivo:
    pass
```



The screenshot shows a Google Colab notebook titled "capítulo2.ipynb". The left sidebar displays a file tree with a red arrow pointing to the "Files" section. The main area shows a code cell with the following content:

```
[1]: from google.colab import drive  
drive.mount('/content/drive')  
  
Mounted at /content/drive  
  
[2]: arquivo = open('/content/drive/MyDrive/files/cap2/exemplo.txt', 'r')  
  
Double-click (or enter) to edit  
  
[ ] with open('/content/drive/MyDrive/files/cap2/exemplo.txt', "w") as arquivo:  
    pass
```

O arquivo foi aberto para leitura e você pode continuar executando os códigos em seu notebook.