

# Manipulação de Dados em Arquivo Texto



**Conteudista:** Prof. Dr. Cleber Silva Ferreira da Luz

**Revisão Textual:** Esp. Jéssica Dante

## Objetivos da Unidade:

- Apresentar recursos importantes do *Python* para trabalhar com arquivos textos;
- Compreender técnicas de manipulação de arquivos textos;
- Conseguir realizar processamento utilizando arquivos textos.



Material Teórico



Material Complementar



Referências



# Material Teórico

---

## Introdução

A busca pelo conhecimento sempre foi a força motriz no avanço da humanidade. Vivemos em uma era tecnológica, na qual a informação digital predomina sobre a humanidade. É notório que, atualmente, a tecnologia desempenha um importante papel na vida da humanidade, ela permite impulsionar o progresso e que nações estejam conectadas, melhora a qualidade de vida e abre novas oportunidades em diversos setores. Podemos facilmente observar também que a tecnologia molda e transforma a maneira de como vivemos, trabalhamos e nos relacionamos. Sua importância é inegável e abrange praticamente todos os aspectos da sociedade moderna.

Hoje, realizamos os mais diversos serviços através de nossos *smartphones*. Isso porque cada dia mais surgem novos aplicativos poderosos que trazem muitos benefícios para as pessoas. Não só os aplicativos, mas, também, todo e qualquer tipo de *software* (*Web, Desktop, Mobile* etc.) traz inúmeros benefícios para a população

A utilização desses *softwares* só é possível porque “Programadores” com bastante *expertises* os desenvolveram com as mais atuais tecnologias. Cada dia mais a programação vem se tornando uma habilidade essencial para pessoas que queiram ingressar na área de tecnologia. Na verdade, a importância da programação vai além do desenvolvimento de *softwares* e sistemas complexos. A verdadeira importância da programação consiste na capacidade das pessoas em resolver problemas e impulsionarem a inovação.

Para você ser um excelente programador, você precisa ter bom raciocínio lógico, conseguir resolver problemas, dominar uma linguagem de programação, conhecer IDEs eficientes, bem como técnicas de programações avançadas que permitirão a você criar algoritmos eficientes e elegantes.

O principal objetivo desta Unidade é apresentar técnicas e ferramentas para a manipulação de arquivo texto na programação. Todo *software* manipula dados, sejam dados de entrada, ou de saídas (CORMEN, 2012). Muitas vezes, é necessário armazenar tais dados em banco de dados, ou até mesmo, em arquivos de texto. Imagine que você precisa desenvolver um *software* que

mantenha um registro em “Log” que armazena o horário de *login* no sistema. Geralmente, arquivos de “log” utilizam arquivo “.txt” para armazenar os dados.

Assim, é visto a grande importância de aprender manipulação de dados em arquivos textos. Nesta Unidade de ensino, focamos na leitura, processamento e escrita em arquivos textos. Utilizaremos a linguagem *Python* e IDE *Colab*. Entretanto, você pode ficar à vontade para usar a IDE que mais goste. Usamos o *Colab* porque é uma eficiente IDE e é *on-line*, assim não precisamos nos preocupar com instalações e configurações. Apenas praticar os conceitos apresentados aqui. Bons estudos.

## Colab

Uma IDE é uma plataforma de desenvolvimento que permite aos seus usuários criarem *software* em determinadas linguagens de programação. Nesta apostila, iremos adotar a IDE *Colab* para apresentarmos técnicas de manipulação de arquivo texto. Utilizaremos, também, a linguagem de programação *Python*. Escolhemos o Colaboratório (*Colaboratory*) ou abreviadamente, *Colab*, para ser a nossa IDE porque ela possui grande facilidade de uso. O *Colab* é uma IDE *on-line*, assim não será necessária a instalação de nada.

O *Colab* é uma IDE desenvolvida e mantida pelo *Google*, esse ambiente possui diversas bibliotecas de funções previamente instaladas que facilitarão nosso trabalho. A IDE *Colab* é um serviço de nuvem gratuito hospedado pelo *Google* para incentivar a programação em *Python*, que permite que você escreva e execute *Python* em seu navegador sem necessidade alguma de configuração, acesso gratuito à aceleração da GPU e recursos de colaboração. Além disso, tem pré-instalado uma série de bibliotecas de nosso interesse.

É importante observar também que o *Google Colab* é constituído de máquinas virtuais. Na verdade, o *Google* disponibiliza uma máquina virtual para que possamos programar e executar os nossos códigos.

---

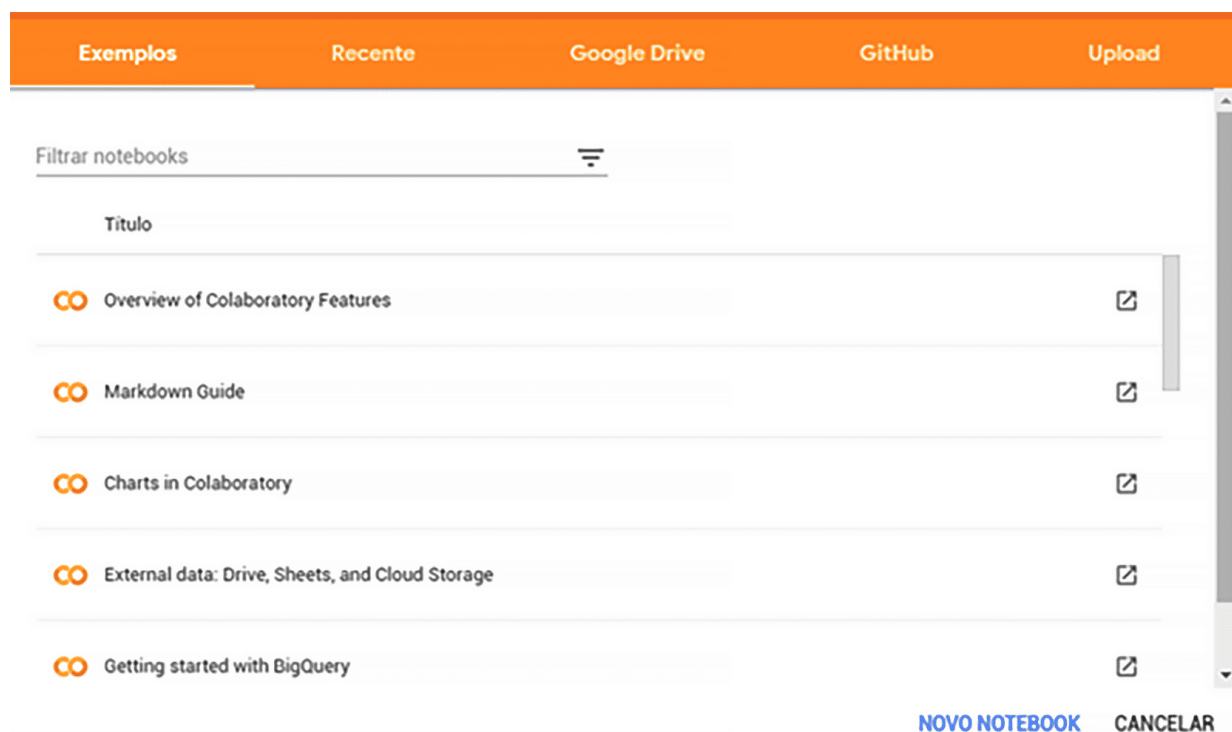
# Leitura

## Colab

Clique no botão para conferir o conteúdo.

ACESSE

Depois que acessar o *link* anterior, você será direcionado para a tela inicial do *Google Colab*, onde constará a seguir com a opção “**Recente**” selecionada. Note que no menu superior você possui as opções conforme a Figura 1.



## Figura 1 – Tela inicial do Colab

Fonte: Reprodução

**#ParaTodosVerem:** foto ilustrando a tela inicial do *Colab*. Foto com fundo em branco e letras em cinza e detalhes em laranja. Ilustrando as opções que existem no *Google Colab*. Fim da descrição.

---

Observe que na Figura 1, na parte laranja, há algumas opções, tais como:

- **Exemplos:** nessa opção, você encontra vários exemplos de utilização dos recursos do *Google Colab*;
- **Recente:** essa opção mostra os *Notebooks* que você criou recentemente;
- **Google Drive:** já essa opção possibilita o armazenamento do seu *notebook* virtual no seu *Google Drive*;
- **GitHub:** agora, essa opção permite que você utilize o *GitHub* como repositório;
- **Upload:** essa última opção permite que você faça *upload* de arquivos *Python* para dentro do seu *notebook*.

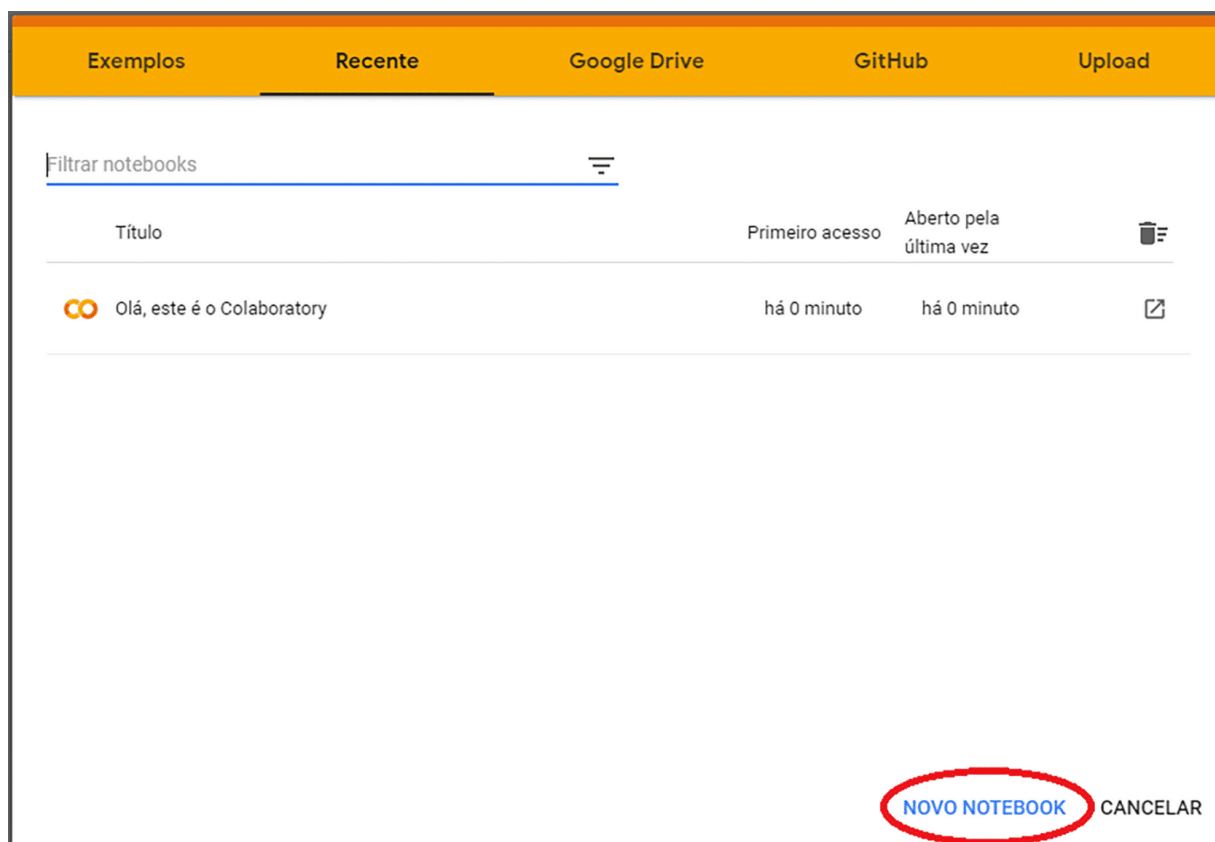
Como mencionado anteriormente, os *Notebooks Colab* são máquinas virtuais e são *Notebooks Jupyter*, que são armazenados no *Google Drive* que permitem combinar código executável e texto em um único documento, junto com imagens, HTML e muito mais.

É importante observar que, para poder instalar um *notebook*, você precisa fazer *login* na sua conta *Google*. Isso permitirá que todos os *notebooks* criados no *Colab* sejam armazenados no seu *Google Drive*.

O *Google Colab* é uma IDE fantástica, ela permite compartilhar *notebooks* com seus amigos. Além do acesso compartilhado, seus amigos poderão editar, alterar e fazer comentários no seu código.

Dessa forma, o *Google Colab* funciona muito bem quando o trabalho é em equipe.

Agora, vamos para a nossa primeira missão, ela consiste na criação de um *Notebook Colab*. Para isso, você deve clicar em “Novo Notebook” que se encontra na parte inferior da telinha laranja do *Colab*.



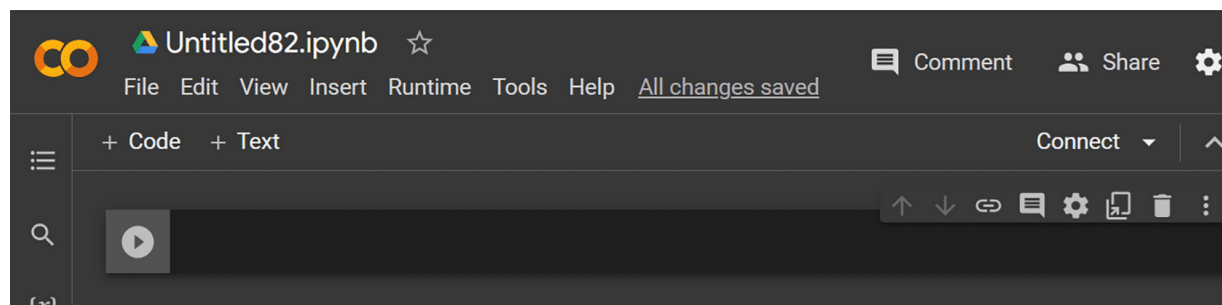
## Figura 2 – Criando um novo *notebook*

Fonte: Reprodução

**#ParaTodosVerem:** foto ilustrando a abertura de um novo *notebook* no *Colab*. Foto com fundo em branco e letras em cinza. A foto possui um menu superior na cor laranja com algumas opções que podem ser acessadas. Fim da descrição.

---

Após criar um *notebook*, você será direcionado para a tela principal do *Google Colab*. Nela, podemos codificar e executar códigos em *Python*. Aqui, você pode codificar o seu código e vê-lo executando logo em seguida.

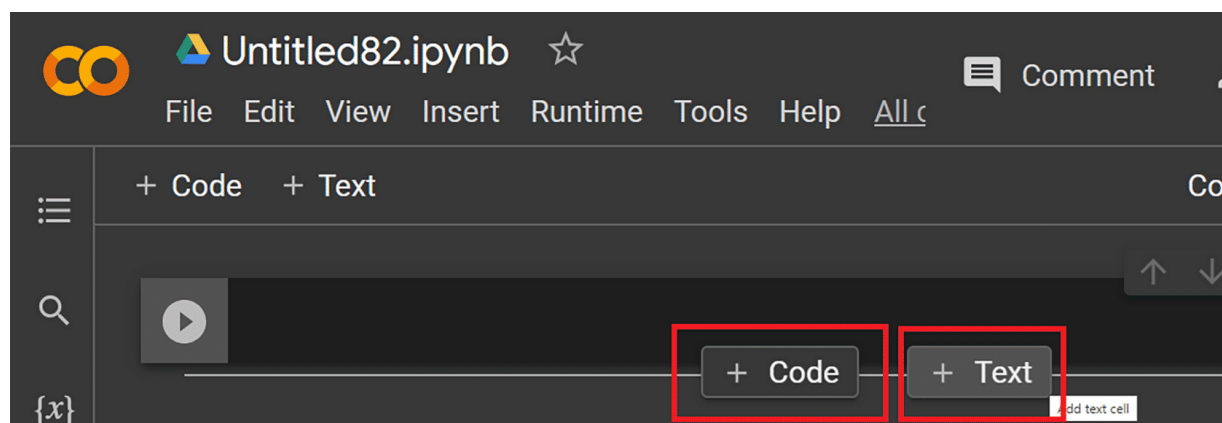


### Figura 3 – Tela de programação do Colab

Fonte: Reprodução

**#ParaTodosVerem:** figura ilustrando a tela de programação do *Colab*. Foto com fundo em preto e letras em brancos. A figura contém célula para codificação, que é representado por um retângulo, contém também mecanismos de execução que é representado por um triângulo dentro de um círculo. Fim da descrição.

Opções são exibidas quando passamos o mouse abaixo da célula, essas opções permitem adicionar uma nova célula, ou ainda, adicionar um texto, isto é, um comentário na célula. Tal situação, é ilustrada a seguir:





## Figura 4 – Célula de programação do Colab

Fonte: Reprodução

**#ParaTodosVerem:** figura ilustrando a Célula de programação do Colab. Foto com fundo em preto e letras em branco. A figura contém um retângulo em vermelho alertando a área de programação. Contém, também, dois retângulos vermelhos apresentando as opções onde criar células e inserir textos. Fim da descrição.

---

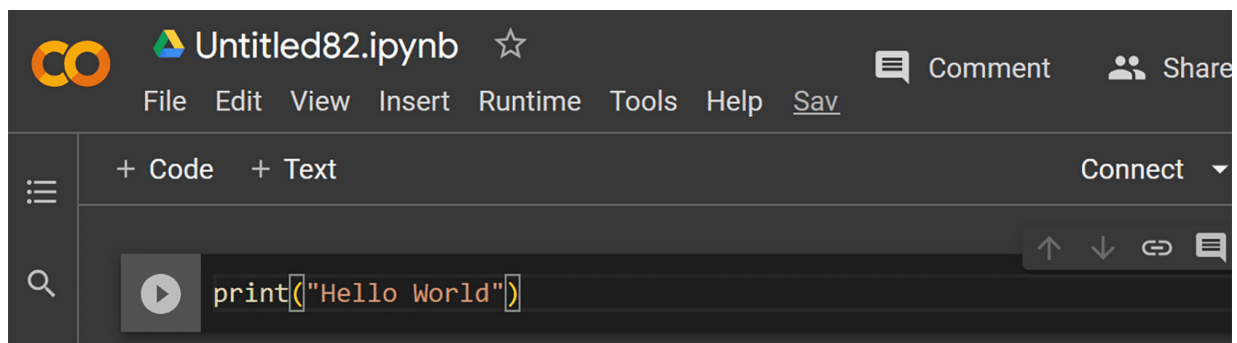
A criação de células permite você ir criando e testando códigos e inserindo comentários sobre o código criado.

A segunda missão que devemos completar consiste em executar um “*Hello World*” no Colab para se familiarizar com o ambiente que iremos trabalhar ao longo desta Unidade de ensino. Para isso, basta colocar o seguinte código:

```
print("Hello World")
```

Dentro da célula criada e, depois, clicar na seta dentro de um círculo no lado esquerdo para executar.

Para executar, você também pode pressionar as teclas “Ctrl + Enter” que o programa será executado.



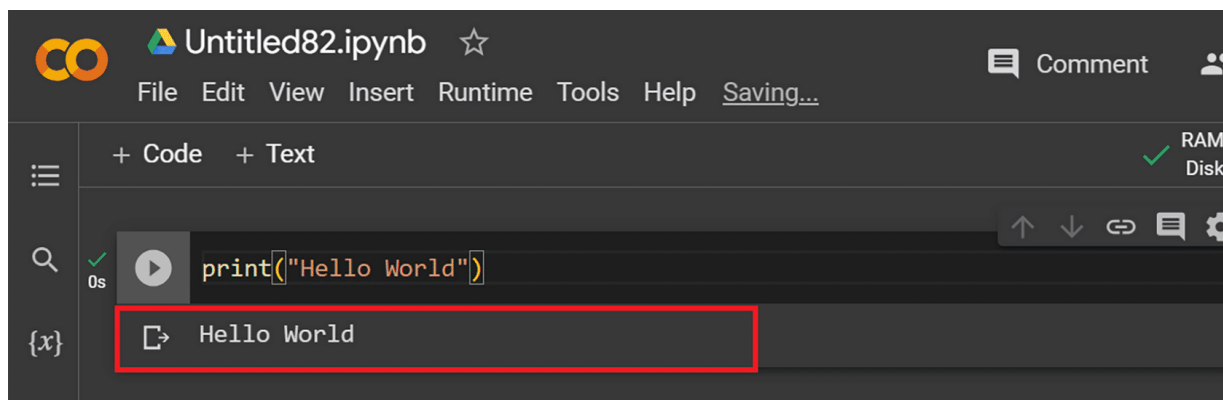
**Figura 5 – Hello World no Colab**

Fonte: Reprodução

**#ParaTodosVerem:** figura ilustrando um *Hello World* no Colab. Foto com fundo em preto e letras em brancas. A Figura contém, também, o `print("Hello World")`. Fim da descrição.

---

A seguir, mostrando o resultado do comando `print("Hello World")`:



**Figura 6 – Resultado do Hello World no Colab**

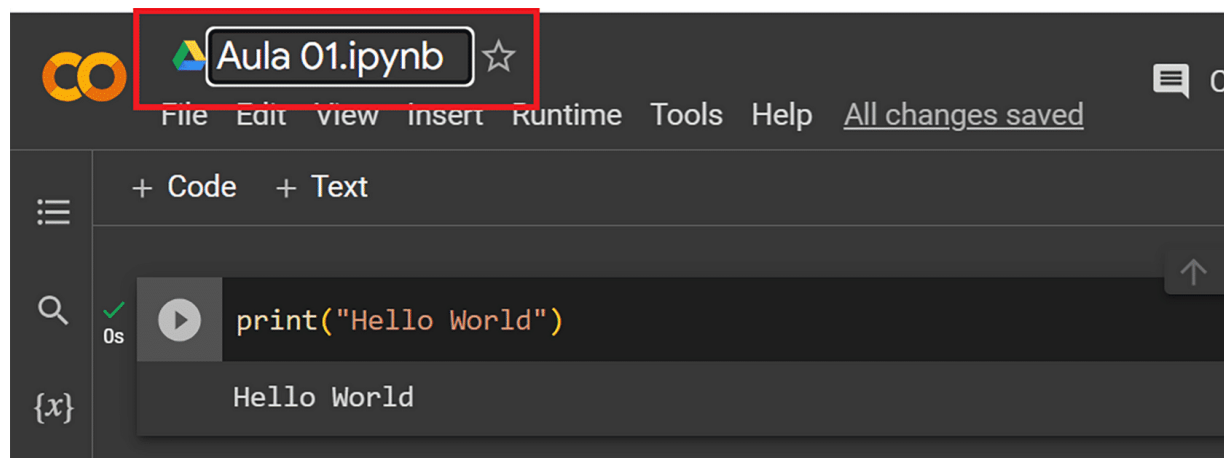
Fonte: Reprodução

**#ParaTodosVerem:** figura ilustrando o resultado do *Hello World* no Colab. Figura com fundo em preto e letras em brancas. A Figura contém também um retângulo em vermelho mostrando o resultado do `print("Hello World")`. Fim da descrição.

---

Dessa forma é que iremos executar todos os códigos apresentados aqui.

Vamos, ainda, apresentar mais um recurso interessante do *Google Colab*, o *Colab* permite que você altere o nome do seu *notebook*, isso permite que você organize melhor as suas aplicações.



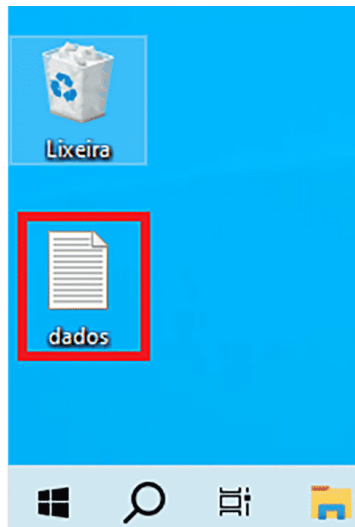
**Figura 7 – Alteração do nome do *notebook* no Colab**

Fonte: Reprodução

**#ParaTodosVerem:** figura ilustrando a alteração de nome no *Google Colab*. Figura com o fundo em preto e com as letras em brancos. Figura contém, também, um retângulo em vermelho que apresenta onde trocar o nome do *notebook*. Fim da descrição.

## Lendo um Arquivo em Texto

O primeiro passo é criar um arquivo “.txt”. Crie um arquivo “.txt” com o nome de “dados.txt”. Conforme ilustrado na Figura a seguir:



## Figura 8 – Criação do arquivo “txt”

Fonte: Reprodução

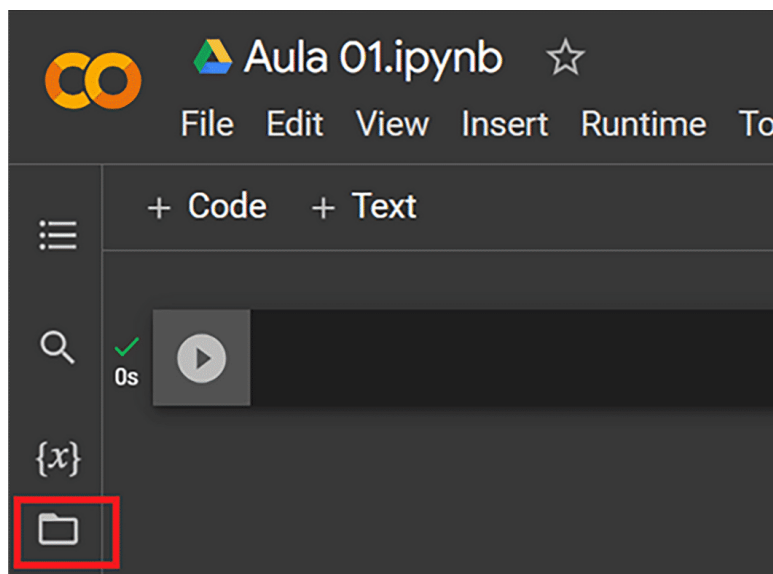
**#ParaTodosVerem:** figura com o fundo azul demonstrando uma área do *Desktop*, onde tem o arquivo “dados.txt” e uma lixeira. A figura também possui um menu abaixo. Fim da descrição.

---

Agora, abra o arquivo “dados.txt” e escreva a seguinte frase: “Meu primeiro arquivo txt”. Salve e feche o arquivo.

Após criar o arquivo “dados.txt”, precisamos enviá-lo para o *notebook*. Isto porque o *Google Colab* é uma IDE *on-line* que está na nuvem, assim, precisamos enviar o arquivo “dados.txt” para a nuvem.

Clique no ícone de uma pasta do lado esquerdo, conforme ilustrado na Figura a seguir:



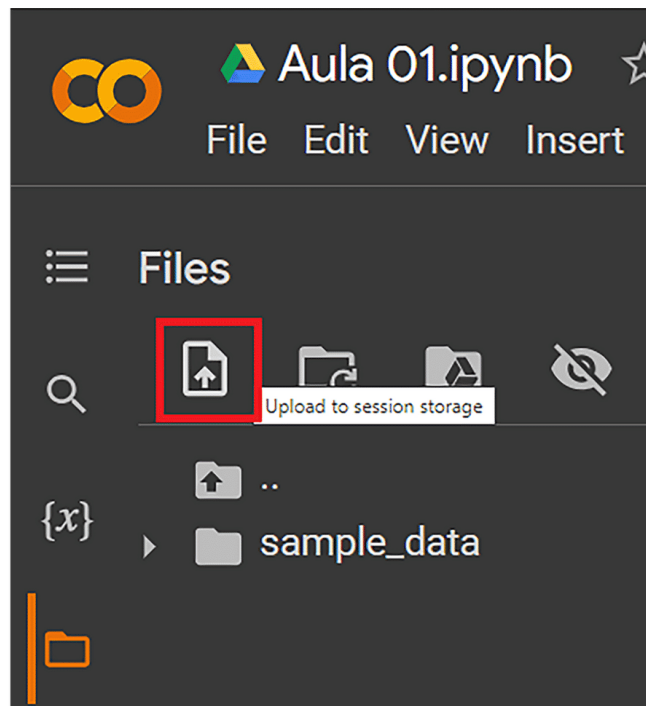
**Figura 9 – Pasta de upload de arquivos no Colab**

Fonte: Reprodução

**#ParaTodosVerem:** figura com o fundo preto demonstrando o ícone da pasta para fazer *upload* do arquivo “txt”. A Figura ainda contém um retângulo em vermelho demonstrando onde está o ícone. Fim da descrição.

---

Após clicar no ícone da pasta, clique em “*upload*” conforme ilustrado na Figura a seguir:



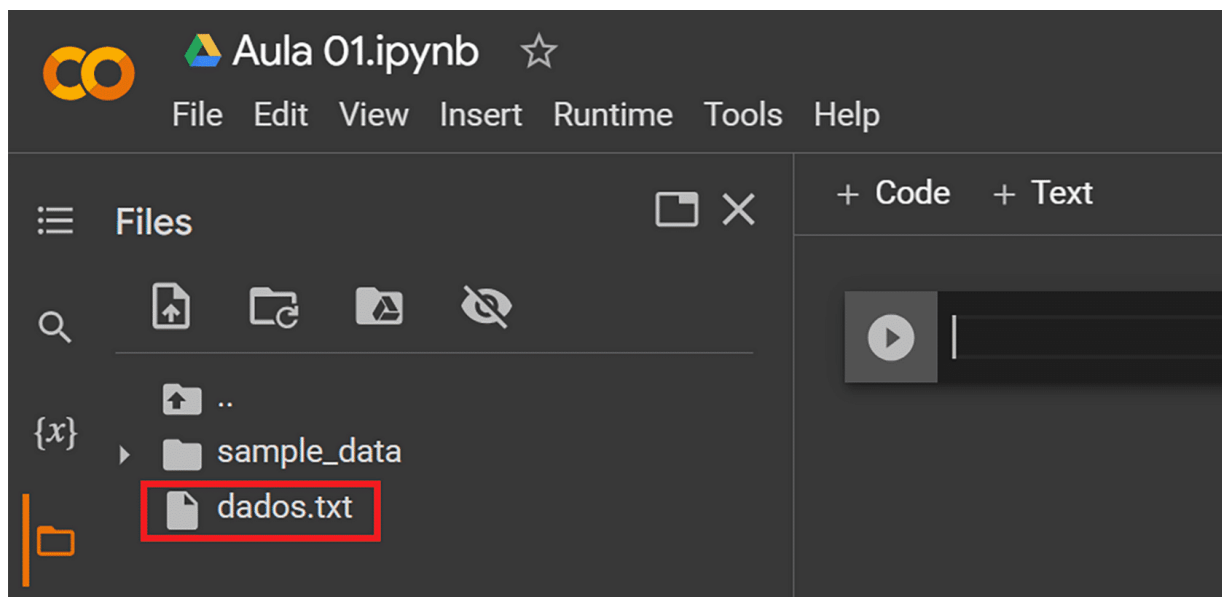
**Figura 10 – Ícone de *upload* de arquivo**

Fonte: Reprodução

**#ParaTodosVerem:** figura com o fundo preto demonstrando o ícone de *upload* para fazer *upload* do arquivo “txt”. A Figura ainda contém um retângulo em vermelho demonstrando onde está o ícone. Fim da descrição.

---

Após fazer o *upload* do arquivo, ele aparecerá no diretório, conforme a Figura a seguir.



**Figura 11 – Arquivo dado no Colab**

Fonte: Reprodução

**#ParaTodosVerem:** figura com o fundo preto demonstrando onde o arquivo “dados.txt” fica após ser feito o *upload*. A Figura ainda contém um retângulo em vermelho demonstrando onde está o arquivo. Fim da descrição.

---

## Lendo um Arquivo Texto

Agora, vamos à programação. O primeiro passo na programação consiste em ler o arquivo e carregá-lo na memória. Para ler um arquivo em “txt” no *Python*, utilizaremos a função “open()”. A função “open()” recebe dois parâmetros, o primeiro é o nome do arquivo e o segundo parâmetro consiste na ação que iremos fazer com o arquivo. Como iremos abrir o arquivo para uma leitura, precisamos usar a opção “r” de *read*.

```
dados = open("dados.txt", "r")
```

Após a execução do código anterior, a variável “dados” será uma referência para o arquivo “dados.txt” que está sendo carregado na memória.

Após, é necessário ler o conteúdo do arquivo. Realizamos essa operação com a função “read”. Escreva o seguinte trecho de código no *Colab*:

```
conteudo = dados.read()
```

Agora, a variável “conteudo” contém todos os dados que estavam dentro de “dados.txt”. Para imprimir os dados que estão em “conteudo”, utilize a função “print”.

```
print(conteudo)
```

E por último, devemos fechar o arquivo com o seguinte código:

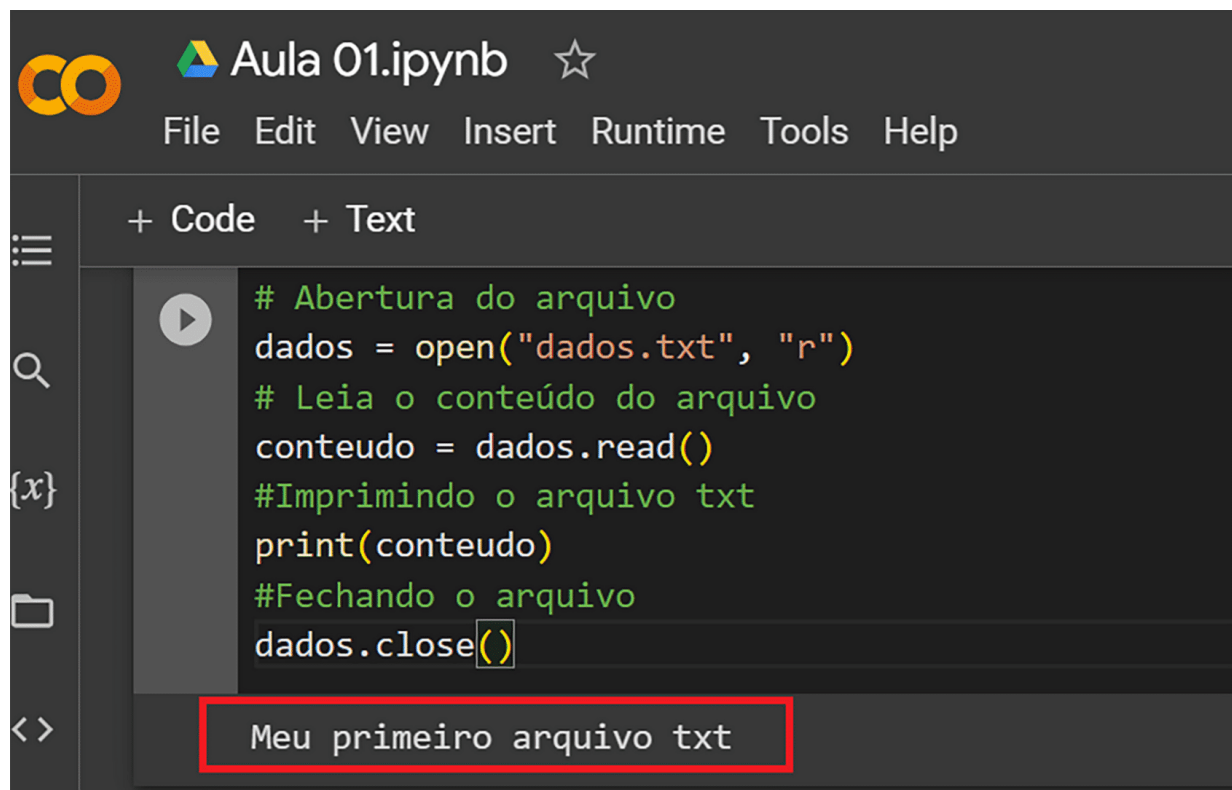
```
dados.close()
```

Neste momento, o seu código deve estar parecido com o código a seguir:



```
1. # Abertura do arquivo
2. dados = open("dados.txt", "r")
3. # Leia o conteúdo do arquivo
4. conteudo = dados.read()
5. #Imprimindo o arquivo txt
6. print(conteudo)
7. #Fechando o arquivo
8. dados.close()
```

Agora, execute o código clicando na seta de execução. A saída do seu programa deve ser igual ao ilustrado na Figura a seguir:



The screenshot shows a Jupyter Notebook window titled "Aula 01.ipynb". The interface includes a menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". Below the menu bar, there are tabs for "+ Code" and "+ Text". The code cell contains the following Python code:

```
# Abertura do arquivo
dados = open("dados.txt", "r")
# Leia o conteúdo do arquivo
conteudo = dados.read()
#Imprimindo o arquivo txt
print(conteudo)
#Fechando o arquivo
dados.close()
```

Below the code cell, the output is displayed in a box with a red border: "Meu primeiro arquivo txt".

**Figura 12 – Saída da execução do código**

Fonte: Reprodução

**#ParaTodosVerem:** figura com o fundo preto demonstrando a saída do código. Os comentários possuem a cor verde, e o código está na cor branca. A Figura

ainda contém um retângulo em vermelho demonstrando a saída. Fim da descrição.

---

## Escrevendo em um Arquivo txt

Para escrever em um arquivo texto, devemos abri-lo com o parâmetro “w” de *write*, ao invés de “r”. Dessa forma:

```
dados = open("dados.txt", "w")
```

Dessa forma, o arquivo será aberto para escrita. Para escrever dados no arquivo, podemos usar a função “write()”. Da seguinte maneira:

```
dados.write("\n Escrevendo no arquivo texto \n")
```

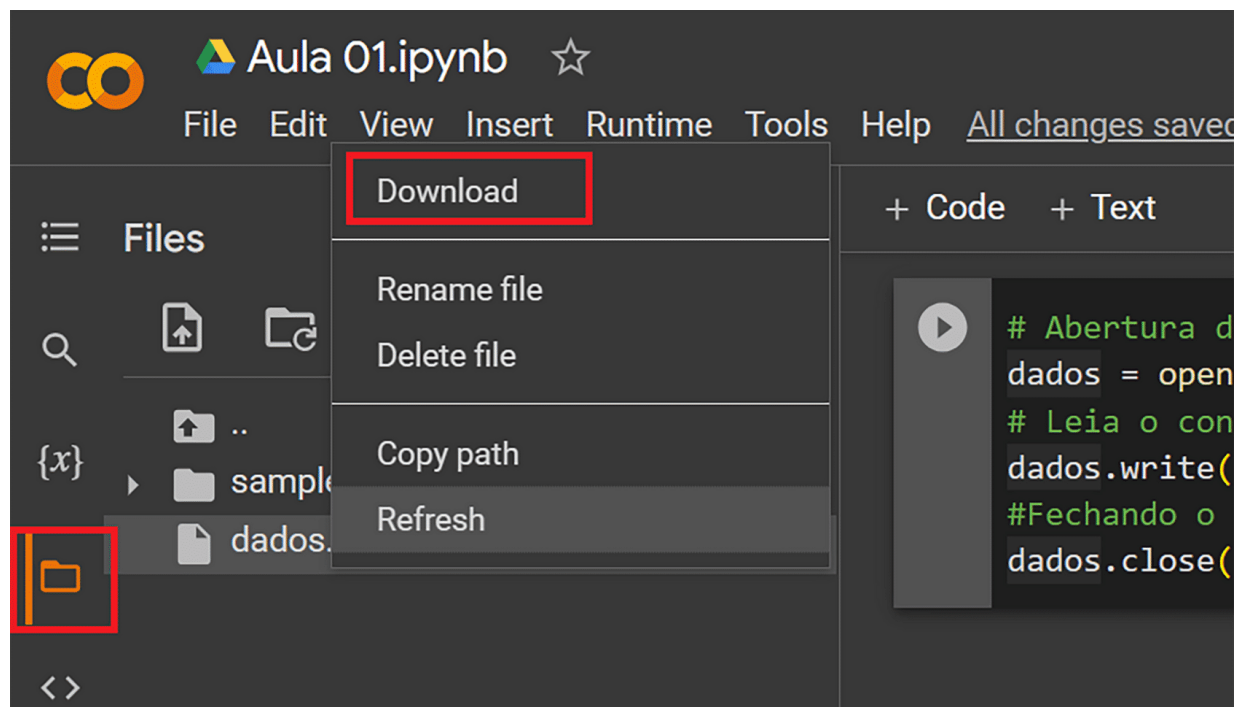
A frase dentro das aspas será escrita no arquivo texto. Após, podemos fechar o arquivo, com a função “close()”.

```
dados.close()
```

Até o momento, o seu código deve estar semelhante ao apresentado a seguir:

```
1. # Abertura do arquivo
2. dados = open("dados.txt", "w")
3. # Escrevendo o conteúdo do arquivo
4. dados.write("\n Escrevendo no arquivo texto \n")
5. #Fechando o arquivo
6. dados.close()
```

Para visualizar o que foi escrito no arquivo, devemos abri-lo. Entretanto, como o arquivo está na nuvem, devemos fazer o *download* dele. Para isso, clique novamente no ícone da pasta, depois clique nos três pontinhos à direita do arquivo e selecione “Download”. Assim, o seu arquivo texto será baixado para o seu computador e você poderá ver o que foi escrito no arquivo. A Figura a seguir ilustra esse procedimento.



## Figura 13 – *Download* do arquivo alterado

Fonte: Reprodução

**#ParaTodosVerem:** figura com o fundo preto demonstrando as opções de *download* do arquivo alterado. A Figura ainda contém um retângulo em vermelho demonstrando os passos que devem ser feitos. Fim da descrição.

---

## Lendo Apenas uma Linha do Arquivo

Como visto anteriormente, a função “`read()`” lê o conteúdo todo do arquivo. Entretanto, muitas vezes, é interessante que você faça a leitura de apenas uma linha do arquivo. Podemos ler apenas uma linha do arquivo usando a função:

```
arquivo.readline()
```

onde “arquivo” é o nome da variável de referência para o arquivo texto no seu código.

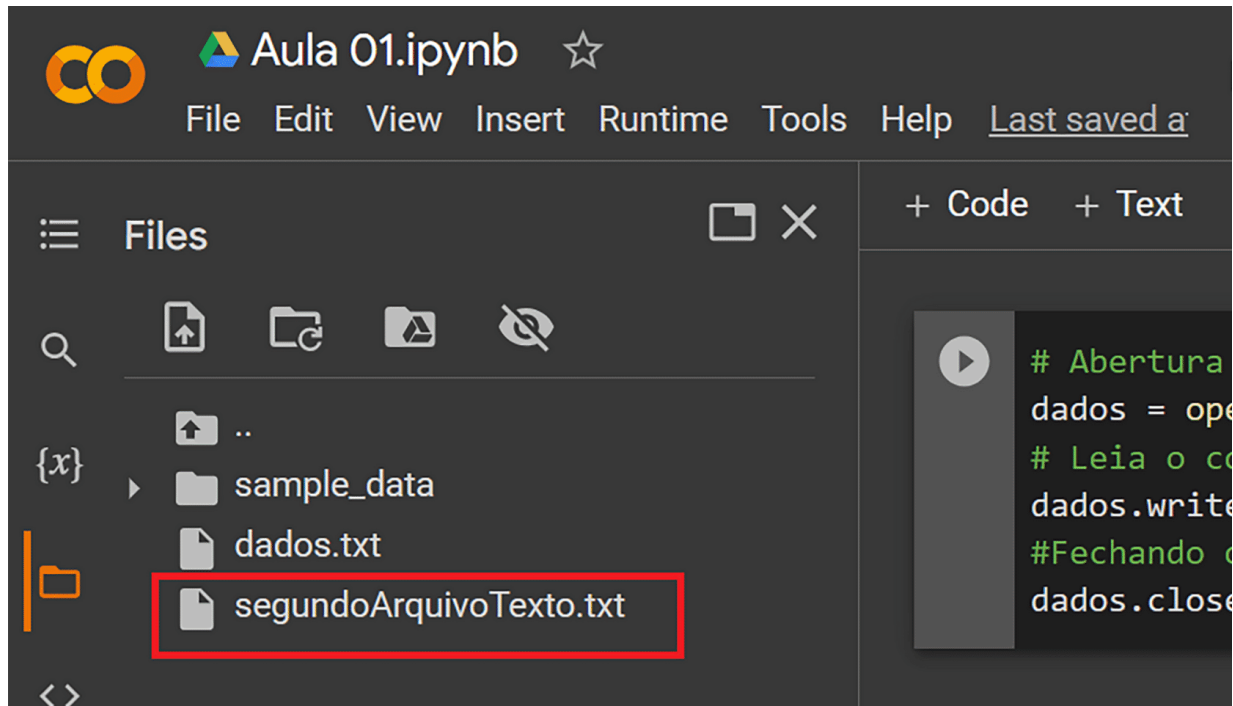
Agora, crie um novo arquivo texto e nomeie esse arquivo como “segundoArquivoTexto.txt”.

Agora, abra esse arquivo e coloque as seguintes frases: “Este é meu segundo exemplo de arquivo texto e *Python*.”

*Python* é uma linguagem fantástica!

É uma linguagem muito simples de programar.

É importante que você digite as frases em linhas separadas. Agora, feche o arquivo e realize o mesmo procedimento anterior para subi-lo para a nuvem.



**Figura 14 – Upload do arquivo**

Fonte: Reprodução

**#ParaTodosVerem:** figura com o fundo preto demonstrando o *upload* do arquivo alterado. A Figura ainda contém um retângulo em vermelho demonstrando onde o arquivo está. Fim da descrição.

---

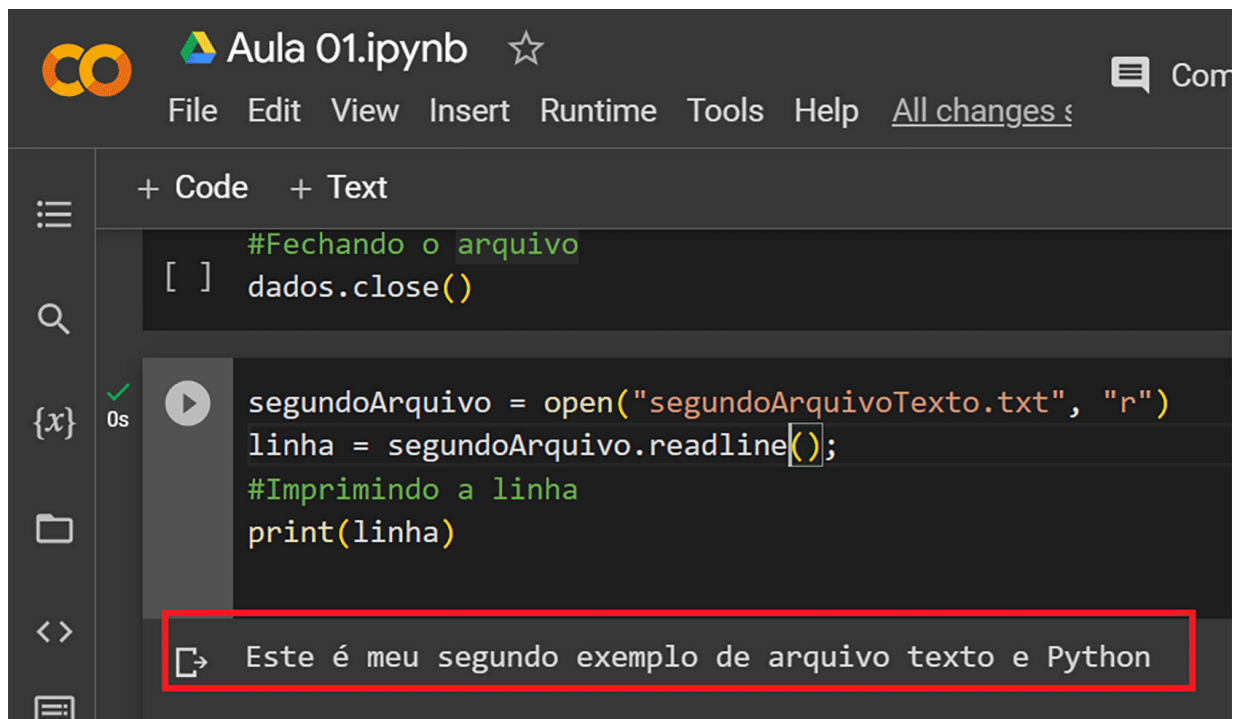
Agora crie uma nova célula para programar.

Antes, é importante observar que existe a função “`readline()`” que lê apenas uma única linha do arquivo e existe, também, a função “`readlines()`” que lê todas as linhas do arquivo e o concatena em uma única variável. Neste exemplo, vamos usar o `readline()`.

Digite o seguinte código na nova célula:

```
1. segundoArquivo = open("segundoArquivoTexto.txt", "r")
2. linha = segundoArquivo.readline();
3. #Imprimindo a linha
4. print(linha)
```

A saída desta execução será semelhante a apresentada na Figura a seguir:

The image shows a Jupyter Notebook interface with a dark theme. At the top, the title bar says 'Aula 01.ipynb' with a star icon. Below it is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and 'All changes'. The left sidebar contains icons for a menu, search, variables, files, and input/output. The main area has a toolbar with '+ Code' and '+ Text'. The code cell contains: 

```
#Fechando o arquivo
[ ] dados.close()
```

 Below this is a run button (play icon) and a code cell with: 

```
segundoArquivo = open("segundoArquivoTexto.txt", "r")
linha = segundoArquivo.readline();
#Imprimindo a linha
print(linha)
```

 The output cell below shows: 

```
>>> Este é meu segundo exemplo de arquivo texto e Python
```

 The output text is enclosed in a red rectangular box.

**Figura 15 – Saída do Código**

Fonte: Reprodução

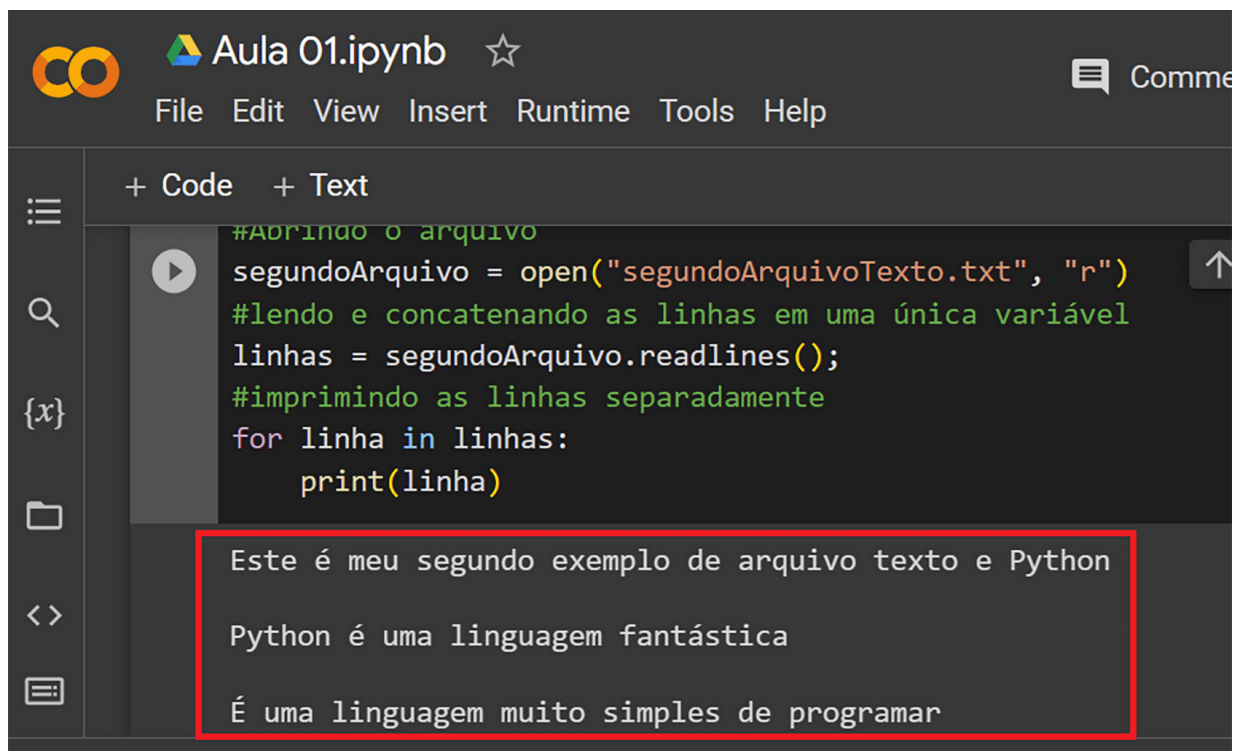
**#ParaTodosVerem:** figura com o fundo preto demonstrando a saída do código. A Figura ainda contém um retângulo em vermelho demonstrando a saída. São apresentados, ainda, códigos que estão na cor branca e comentários na cor verde. Fim da descrição.

Perceba que foi impresso apenas a primeira linha, isso porque a função “`readline()`” lê apenas uma linha por vez. Se você deseja ler apenas uma linha por vez e imprimi-las cada uma separada, também, você utiliza a função “`readline()`” dentro de um `for`, ou ainda, usa a função “`readlines()`”.

No código a seguir, usamos a função “`readlines()`”.

```
1. #Abrindo o arquivo
2. segundoArquivo = open("segundoArquivoTexto.txt", "r")
3. #lendo e concatenando as linhas em uma única variável
4. linhas = segundoArquivo.readlines();
5. #imprimindo as linhas separadamente
6. for linha in linhas:
7.     print(linha)
```

A saída desse programa será:



The screenshot shows a Jupyter Notebook window titled "Aula 01.ipynb". The interface includes a top menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". Below the menu is a toolbar with icons for file operations and a "Code" button. The main area is divided into two sections: a code editor and an output area. The code editor contains the following Python code:

```
#Abrindo o arquivo
segundoArquivo = open("segundoArquivoTexto.txt", "r")
#lendo e concatenando as linhas em uma única variável
linhas = segundoArquivo.readlines();
#imprimindo as linhas separadamente
for linha in linhas:
    print(linha)
```

The output area below the code editor displays the text read from the file, which is enclosed in a red rectangular box:

```
Este é meu segundo exemplo de arquivo texto e Python
Python é uma linguagem fantástica
É uma linguagem muito simples de programar
```

**Figura 16 – Saída do Código 2**

Fonte: Reprodução

**#ParaTodosVerem:** figura com o fundo preto demonstrando a saída do código com o comando “`readlines()`”. A Figura ainda contém um retângulo em vermelho demonstrando a saída. São apresentados, ainda, códigos que estão na cor branca e comentários na cor verde. Fim da descrição.

## Abrindo um Arquivo para Leitura e Escrita ao Mesmo Tempo

Para abrir um arquivo para leitura e escrita ao mesmo tempo, devemos abri-lo com “`r+`”. Assim, podemos abrir um arquivo, ler e depois escrever nele, no mesmo código. Observe o código a seguir:

```
1. # Abertura do arquivo
2. dados = open("dados.txt", "r+")
3. # Leia o conteúdo do arquivo
4. conteudo = dados.read()
5. # Imprimindo o conteudo
```



```
6. print(conteudo)
7. # Escrevendo no arquivo
8. dados.write("\n Phyton s2 \n")
9. #Fechando o arquivo
10. dados.close()
```

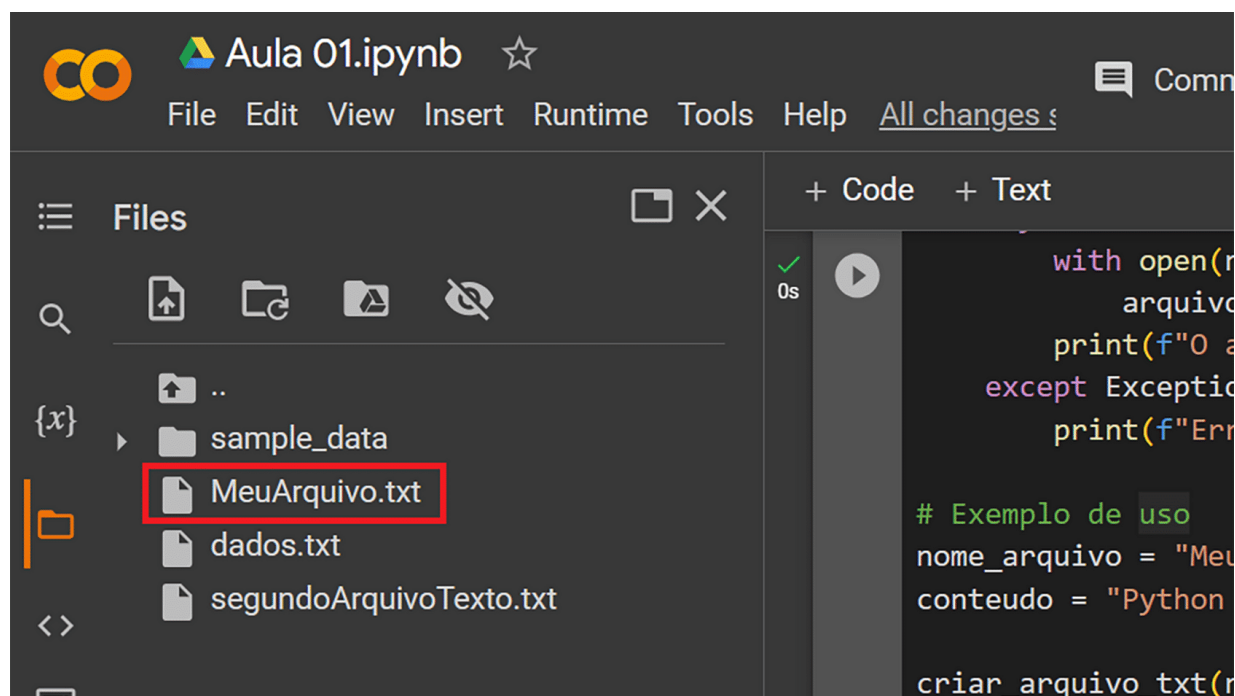
No código anterior, abrimos o arquivo, lemos o seu conteúdo e logo em seguida, imprimimos o conteúdo que estava no arquivo. Posteriormente, escrevemos mais conteúdo no arquivo e depois o fechamos. É interessante observar que o conteúdo escrito será escrito abaixo do conteúdo lido anteriormente, isso porque como lemos todo o conteúdo do arquivo primeiro, o cursor está no final do arquivo e, assim, irá adicionar o conteúdo da linha oito no arquivo. Dependendo da ordem que lemos e escrevemos no arquivo, o conteúdo pode ser sobreposto.

## Como Criar um Arquivo “txt” em Python

Em muitas situações, o programa desenvolvido terá que criar um arquivo texto, ao invés de abrir um arquivo já existente. Para criar um arquivo do zero e adicionar conteúdo nele, você pode usar o seguinte código:

```
1. def criar_arquivo_txt(nome_arquivo, conteudo):
2.     try:
3.         with open(nome_arquivo, 'w') as arquivo:
4.             arquivo.write(conteudo)
5.             print(f"O arquivo '{nome_arquivo}' foi criado com sucesso!")
6.     except Exception as e:
7.         print(f"Erro ao criar o arquivo: {e}")
8.
9. nome_arquivo = "MeuArquivo.txt"
10. conteudo = "Python s2."
11.
12. criar_arquivo_txt(nome_arquivo, conteudo)
```

No código anterior, criamos um método chamado de “criar\_arquivo\_txt()”. Esse método será responsável por criar o arquivo “txt”. Posteriormente, na linha 12, chamamos o método e ele criará o nosso arquivo. Para você verificar se o arquivo realmente foi criado, vá no ícone da pasta ao lado esquerdo.



**Figura 17 – Saída do Código 3**

Fonte: Reprodução

**#ParaTodosVerem:** figura com o fundo preto demonstrando a criação do arquivo. A Figura ainda contém um retângulo em vermelho demonstrando onde o arquivo foi criado. São apresentados, ainda, códigos que estão na cor branca e comentários na cor verde. Fim da descrição.

---

## Em Síntese

Já está mais do que comprovado que a tecnologia possui um papel importante na sociedade. Com a tecnologia podemos ir mais longe na busca do conhecimento e obter ideias inovadoras. Atualmente, a tecnologia traz muitos benefícios e conforto para as pessoas. Hoje realizamos quase todas as nossas rotinas na palma da mão com os nossos *smartphones*. Isso só é possível porque algum desenvolvedor criou excelentes aplicativos através da programação. A programação possibilita a criação de diversos *softwares*, seja *mobile*, *web* ou *desktop*.

Na programação, muitas vezes, é necessário a manipulação de dados, sejam dados de entrada ou de saída. Esta Unidade de ensino traz um estudo sobre manipulação de arquivo texto em *Python*. Aqui, estudamos os principais recursos sobre a criação, leitura e escrita em arquivo texto.



## Material Complementar

---

Indicações para saber mais sobre os assuntos abordados nesta Unidade:

---

### Vídeos

#### *Python* Arquivos txt em 7 Minutos [Fácil] + Desafios

Este vídeo complementa o conhecimento sobre manipulação de arquivo “txt” em *Python*.



## Como Criar, Ler e Escrever Arquivos com *Python*

Este vídeo apresenta um estudo sobre manipulação de dados em *Python*.



## Como Ler, Editar e Criar Arquivos de Texto com *Python* – [Trabalhando com Textos no *Python*]

Este vídeo apresenta um estudo sobre manipulação de dados em *Python*.

## Manipulando Arquivos com *Python*

Este vídeo apresenta um estudo sobre manipulação de dados em *Python*.



## Referências

---

ALVES, W. P. **Programação Python**: aprenda de forma rápida. São Paulo: Editora Saraiva, 2021. (*e-book*)

BARRY, P. **Use a Cabeça! Python**. 2. ed. Rio de Janeiro: Editora Alta Books, 2018. (*e-book*)

CORMEN, T. **Algoritmos**: teoria e prática. 3. ed. Rio de Janeiro: Elsevier, 2012. (*e-book*)

SHAW, Z. A. **Aprenda Python 3 do jeito certo**. Rio de Janeiro: Editora Alta Books, 2019.