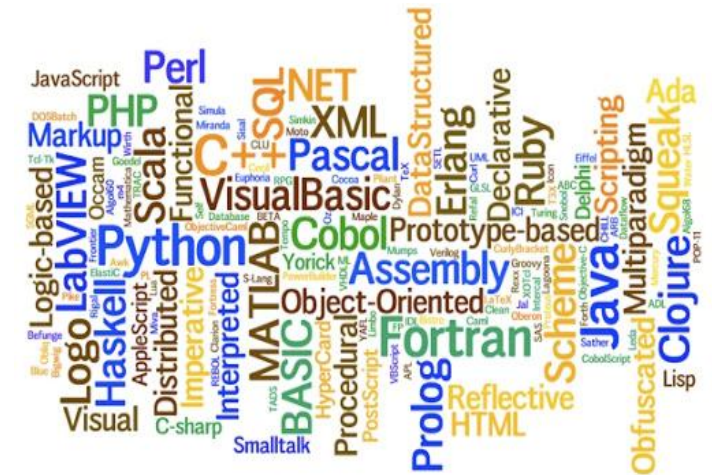




Linguagem de Programação - LP

Prof. Ms. Massaki de O. Igarashi



Propósito desta Aula!

CONCEITOS INTRODUTÓRIOS DE PYTHON

- Diferença entre Compilador e Interpretador
- Primeiros passos com o Interpretador **Colaboratory** da Google
- Conceito de Variável
- Entendendo o que é sistema: ENTRADA, PROCESSAMENTO E SAÍDA
- Variável tipo inteira e tipo real
- Narrativa, Fluxograma, Algoritmo ou pseudocódigo e código

Propósito da Componente LP

Estudo dos conceitos básicos de informática. Descrição de **algoritmos: Narrativa, Pseudocódigo, Fluxogramas e Linguagem de programação C++**. Desenvolvimento de **Lógica** de Programação. Estudo dos Elementos básicos de programação: **variáveis e tipos; entrada e saída de dados; estrutura sequencial; estruturas condicionais; estruturas repetitivas; funções predefinidas e funções de usuário**. Simulação de algoritmos (teste de mesa). Elaboração de **funções com passagem de parâmetros por valor e por referência. Criação de Unidades independentes (bibliotecas de funções)**. Manipulação de arranjos estáticos (**vetores e matrizes**). Noções de interfaces gráficas de usuário.

DEFINIÇÃO



A linguagem de Programação é um conjunto de **instruções e símbolos** escritas em um **código fonte** que permite a nós humanos traduzir nossos pensamentos em **instruções que os computadores possam entender** (já que a eletrônica é essencialmente binária). Este código pode ser compilado e transformado em um programa de computador, ou usado como script interpretado; que informará **instruções de processamento ao computador.**



Fonte: <https://news.codecademy.com/programming-languages/>

Aula Introdução!

O que é um programa?

- Um programa é uma seqüência de instruções destinada a resolver um problema.
- Um programa é um algoritmo!



Fonte: Barros (2020).



Compilador

Compiladores fazem a ação de **compilar** um código C++ e construir os programas para serem executados no computador.

Os computadores entendem apenas conjuntos de instruções feitas de 1 e 0. Essa linguagem de computador é apropriadamente chamada de *linguagem de máquina*, ou *linguagem binária*. Mas programar um computador diretamente em linguagem de máquina usando apenas 1 e 0 é muito tedioso e propenso a erros.

Compilar significa traduzir uma programa escrito em uma linguagem de programação para formato no qual o computador entenda!



Fonte: <https://www.codigosinformaticos.com/cuales-son-los-compiladores-mas-utilizados-en-el-mundo/>

Para facilitar a programação, foram desenvolvidas linguagens de alto nível. Programas de alto nível também tornam mais fácil para os programadores inspecionarem e entenderem os programas uns dos outros.

Interpretador

Interpretador Python

O intérprete é um programa que compreende as instruções que você vai escrever na linguagem Python.

Sem o intérprete, o computador não vai entender as instruções e os programas não funcionam.



Vantagens das linguagens compiladas

Os programas compilados em código de máquina nativo tendem a ser mais rápidos que o código interpretado. Isso ocorre porque o processo de traduzir o código em tempo de execução aumenta o tempo do processo, podendo fazer com que o programa seja, em geral, mais lento.

Desvantagens das linguagens compiladas

Tempo adicional necessário para concluir toda a etapa de compilação antes dos testes. Dependência da plataforma do código binário gerado.

****Linguagens interpretadas****

Os interpretadores passam por um programa linha por linha e executam cada comando. Aqui, se o autor decidir que quer usar um tipo diferente de óleo de oliva, só precisaria remover o antigo e adicionar o novo. Seu amigo tradutor poderia informar isso a você quando a mudança acontecesse. Linguagens interpretadas, antigamente, eram significativamente mais lentas do que as linguagens compiladas. Porém, com o desenvolvimento da compilação just-in-time, essa distância vem diminuindo. Exemplos de linguagens interpretadas comuns são o PHP, o Ruby, o Python e o JavaScript. Um pequeno detalhe. A maioria das linguagens de programação pode ter implementações compiladas e interpretadas – a linguagem em si não é necessariamente compilada ou interpretada. Porém, para fins de simplicidade, elas são normalmente referidas deste modo. Python, por exemplo, pode ser executado como um programa compilado ou como uma linguagem interpretada em modo interativo. Por outro lado, a maioria das ferramentas de linha de comando, ou CLIs, e shells podem, em teoria, ser classificadas como linguagens interpretadas.

Vantagens das linguagens interpretadas

As linguagens interpretadas tendem a ser mais flexíveis, geralmente oferecendo recursos como digitação dinâmica e tamanho reduzido de programa. Além disso, como os interpretadores executam o código fonte do programa por conta própria, o código não depende da plataforma.

Desvantagens das linguagens interpretadas

A desvantagem mais notável é a velocidade típica de execução em comparação com as linguagens compiladas.



Por que aprender Python?



- ✓ **Linguagem mais utilizada** hoje globalmente
- ✓ Fácil de aprender
- ✓ **Interoperabilidade** (comunica-se de forma transparente com outras linguagens:

Java, .NET e bibliotecas C/C ++);

- ✓ Permite **integração e desenvolvimento web**;
- ✓ Tem muitos recursos e **bibliotecas para visualização de dados**;
- ✓ **Interpreta scripts** (não requer compilação já que interpreta o código diretamente);

Rank	Language	Type	Score
1	Python	  	100.0
2	Java	  	95.4
3	C	  	94.7
4	C++	  	92.4
5	JavaScript		88.1
6	C#	   	82.4
7	R		81.7
8	Go	 	77.7
9	HTML		75.4
10	Swift	 	70.4

Um resumo sobre Python

Python é uma **linguagem de scripts** que permite executar e testar um código imediatamente depois de escrevê-lo, facilitando bastante as atualizações. Em outras palavras, linguagens de script são linguagens interpretadas. O **interpretador executa o programa apenas traduzindo comandos em uma série de uma ou mais sub-rotinas** que depois são traduzidas em outras linguagens.

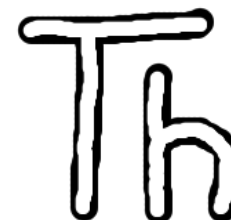
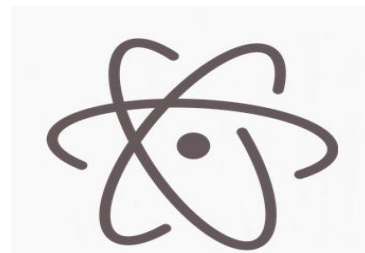
Um script é uma **coleção de comandos em um arquivo** projetada **para ser executada como um programa** e não pelo processador do computador, como acontece com linguagens compiladas. **O arquivo pode conter funções e módulos variáveis**, mas **a ideia central é que ele possa rodar e cumprir uma tarefa específica a partir de uma linha de comando**. Um exemplo clássico disso são as linguagens para prompts de comando, como no arquivo batch Windows.

Em geral, é mais rápido e fácil programar usando uma linguagem de script do que uma mais estruturada e compilada, como C ou C++.

Principais interpretadores PYTHON

O acrônimo **IDE (Integrated Development Environment)** é usado para definir um software ou ambiente de desenvolvimento integrado que une ferramentas de desenvolvimento em uma única interface gráfica do usuário (GUI) para escrever e testar códigos escrito em diferentes linguagens de programação.

Principais IDE's PYTHON:



Introdução ao Google Colab

1

- **Introdução**

2

- **Passo a Passo**

3

- **Referências**

Introdução ao Google Colab

“

O Google Colab é um ambiente **semelhante a plataforma Jupyter notebook**. É um ambiente **gratuito** que funciona **inteiramente na nuvem**. Não requer uma configuração e **os blocos de notas** que você cria **podem ser editados simultaneamente por todos os membros de sua equipe** - da mesma forma que você edita documentos no **Google Docs**. Colab oferece suporte a muitas bibliotecas de aprendizado de máquina populares que podem ser carregadas em seu notebook. Usando o Google Colab você pode: **escrever e executar código em Python**; documentar seu código com suporte a equações matemáticas; **criar / Carregar / Compartilhar blocos de anotações**; **Importar / Salvar** notebooks de / **para o Google Drive**; **Importar / Publicar notebooks do GitHub**; Importar conjuntos de dados externos, por exemplo de Kaggle; Integrar PyTorch, TensorFlow, Keras, OpenCV; Serviço de nuvem grátis com GPU grátis.

”

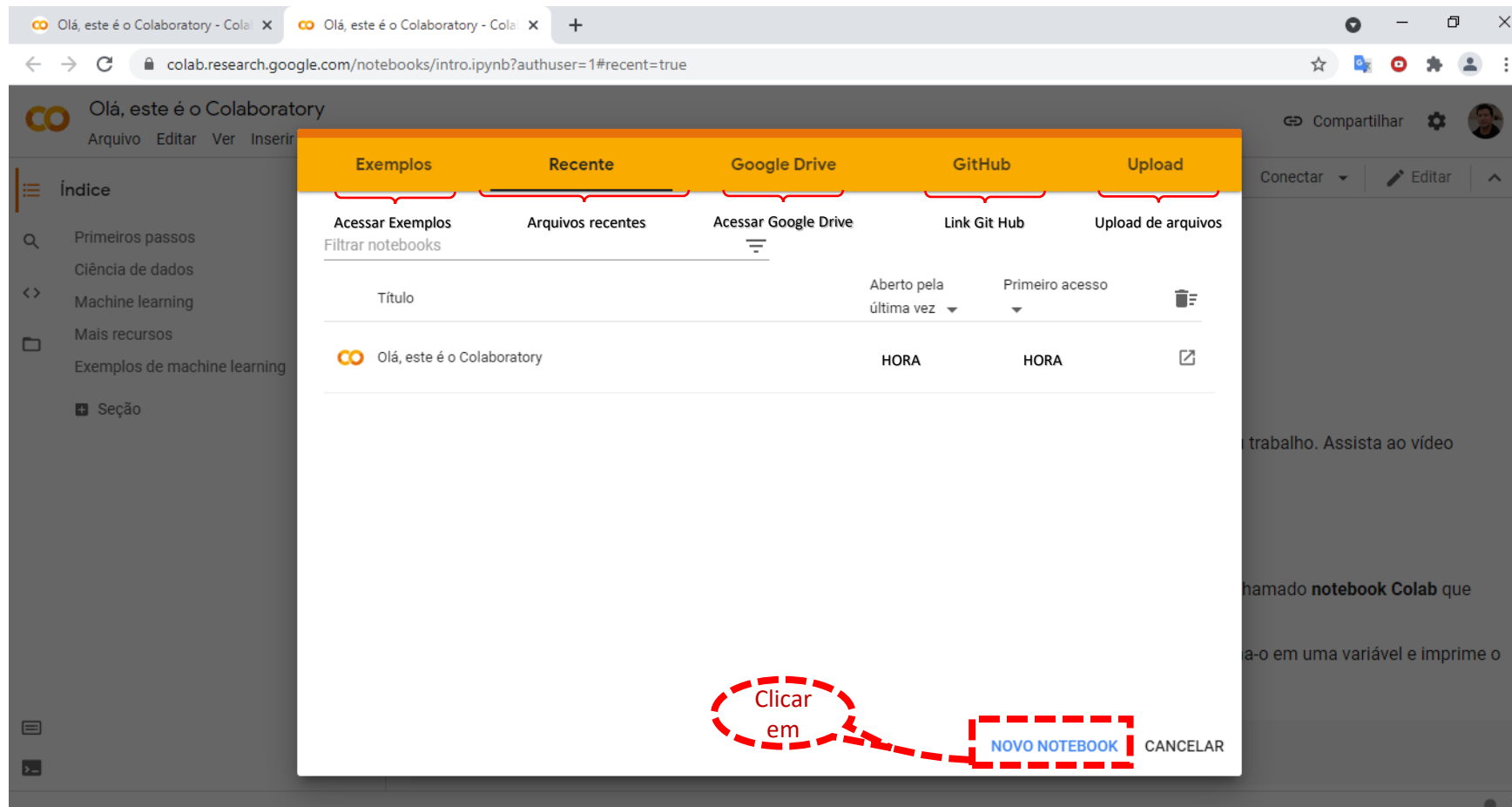


<https://www.youtube.com/watch?v=inN8seMm7UI>

Google Colab em 10 + 2 Passos

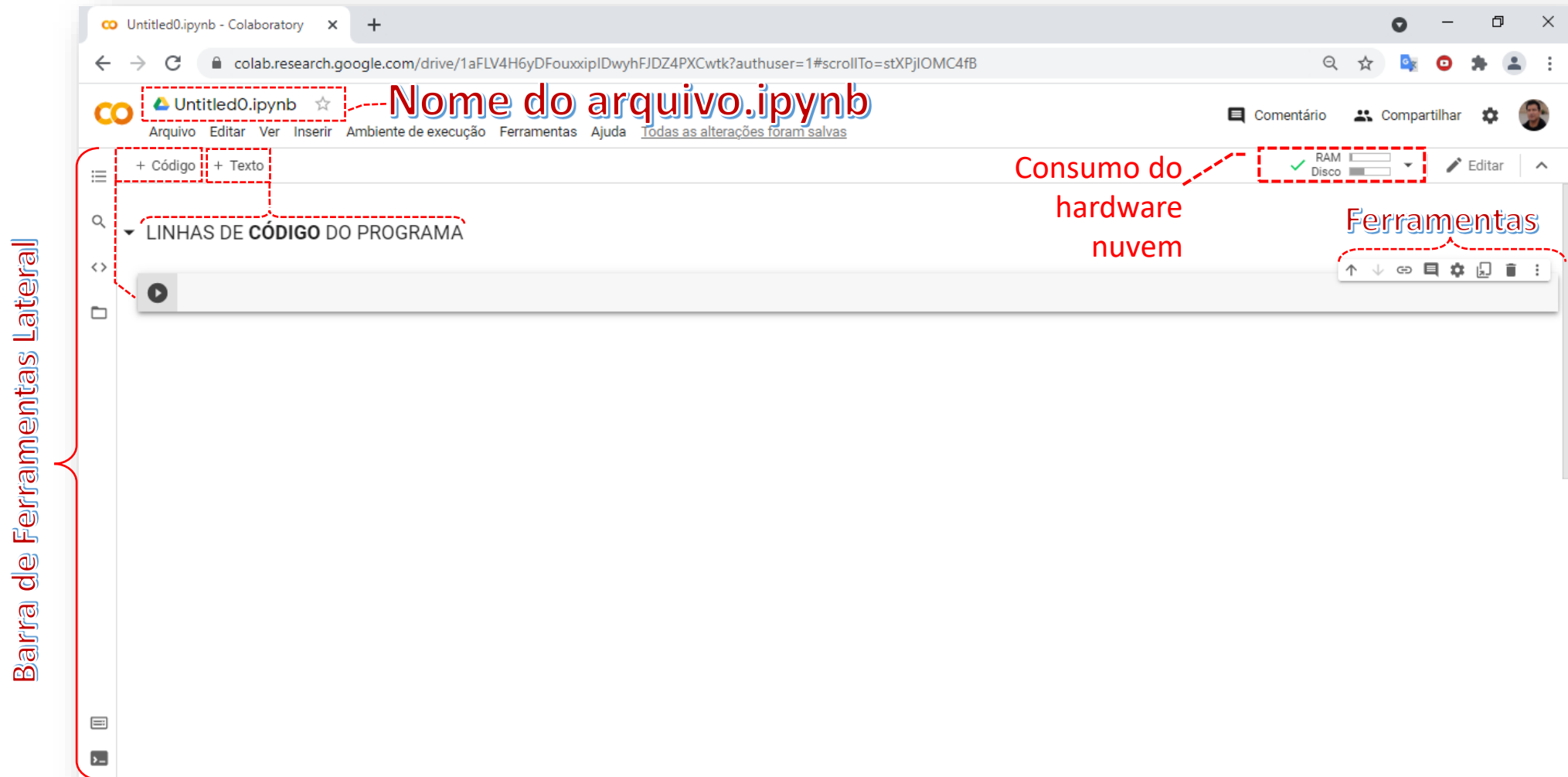
Passo 1: utilize seu navegador web para acessar a URL <https://colab.research.google.com>

OBS: Seu navegador exibiria a seguinte tela (desde que você esteja logado em seu Google Drive e usando navegador Chrome):



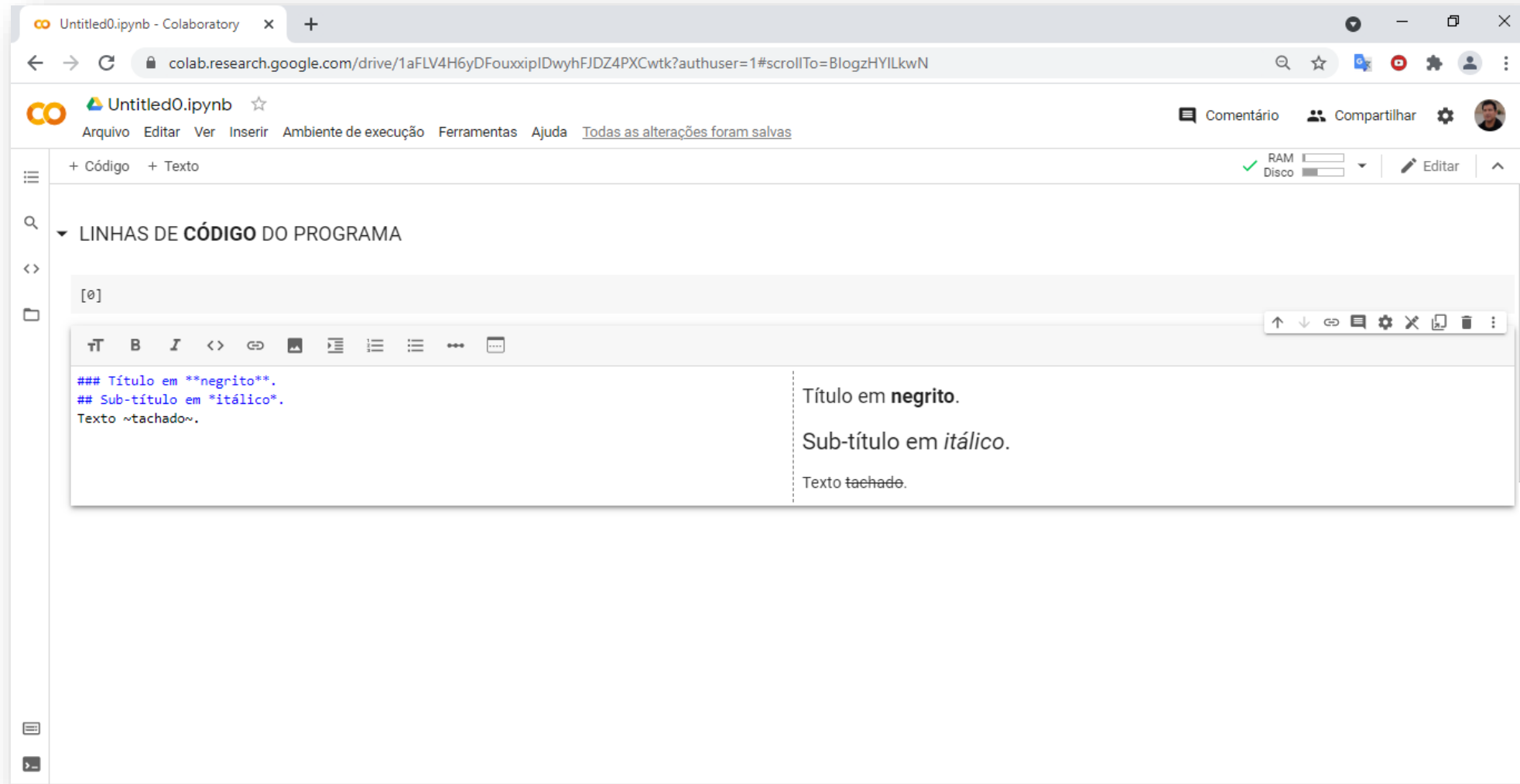
Google Colab em 10 + 2 Passos

Passo 2: ao Clicar em NOVO NOTEBOOK um novo bloco de anotações do colab abrirá:



Google Colab em 10 + 2 Passos

Passo 2: entendendo a ferramenta Markdown para formatação de texto.



Google Colab em 10 + 2 Passos

Passo 3: inserindo equações matemáticas no seu texto Markdown.

Passo 03) Inserindo equações matemáticas nos textos de anotações

```
## Exemplo 1:
 $x \in [-5, 5]$ 

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

## Exemplo 3:
 $e^x = \sum_{i=0}^{\infty} \frac{1}{i!} x^i$ 

## Exemplo 4:

$$\begin{aligned} -3x_1 + 6x_2 + x_3 &\leq 28 \\ -7x_1 + 3x_2 + 2x_3 &\leq 37 \\ -4x_1 + 5x_2 + 2x_3 &\leq 19 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$


## Exemplo 5: Vetores

$$\begin{aligned} u_i(t) &= x_i(t) + \beta(\hat{x}(t) - x_i(t)) + \beta \sum_{k=1}^n \\ &\quad \cdot (x_{i1,k}(t) - x_{i2,k}(t)) \end{aligned}$$



$$f(x_1, x_2) = 20 + e - 20\exp(-0.2 \sqrt{\frac{1}{n}} (x_1^2 + x_2^2)) - \exp(\frac{1}{n} (\cos(2\pi x_1) + \cos(2\pi x_2)))$$


## Exemplo 6: Matriz

$$A_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

```

Exemplo 1:

$$x \in [-5, 5]$$

Exemplo 2:

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

Exemplo 3:

$$e^x = \sum_{i=0}^{\infty} \frac{1}{i!} x^i$$

Exemplo 4:

- $3x_1 + 6x_2 + x_3 \leq 28$
- $7x_1 + 3x_2 + 2x_3 \leq 37$
- $4x_1 + 5x_2 + 2x_3 \leq 19$
- $x_1, x_2, x_3 \geq 0$

Exemplo 5: Vetores

$$u_i(t) = x_i(t) + \beta(\hat{x}(t) - x_i(t)) + \beta \sum_{k=1}^n (x_{i1,k}(t) - x_{i2,k}(t))$$

$$f(x_1, x_2) = 20 + e - 20\exp(-0.2 \sqrt{\frac{1}{n}} (x_1^2 + x_2^2)) - \exp(\frac{1}{n} (\cos(2\pi x_1) + \cos(2\pi x_2)))$$

Exemplo 6: Matriz

$$A_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

Google Colab em 10 + 2 Passos

Passo 4: Salvando seu bloco de notas python

The screenshot shows the Google Colab interface for a notebook named 'Aula01.ipynb'. The 'File' menu is open, displaying various options. The 'Save' option is highlighted, with the keyboard shortcut 'Ctrl+S' shown next to it. Other options include 'Localizar no Drive', 'Abrir no modo Playground', 'Novo notebook', 'Abrir notebook', 'Fazer upload de notebook', 'Rename', 'Mover para a lixeira', 'Salvar uma cópia no Drive', 'Salvar uma cópia como Gist do GitHub', 'Salvar uma cópia no GitHub', 'Salvar e fixar revisão', 'Histórico de revisões', 'Fazer download', and 'Imprimir'. The main editor area shows a code cell with a matrix definition using LaTeX notation. The right sidebar contains a 'Comentário' section and a 'Compartilhar' section. The bottom status bar shows 'RAM' and 'Disco' usage.

File menu options:

- Localizar no Drive
- Abrir no modo Playground
- Novo notebook
- Abrir notebook
- Fazer upload de notebook
- Rename
- Mover para a lixeira
- Salvar uma cópia no Drive
- Salvar uma cópia como Gist do GitHub
- Salvar uma cópia no GitHub
- Salvar (Ctrl+S)
- Salvar e fixar revisão (Ctrl+M S)
- Histórico de revisões
- Fazer download
- Imprimir (Ctrl+P)

Code cell content:

```
=  
matrix(  
    & a_{1,2} & \cdots & a_{1,n} \\  
    & a_{2,2} & \cdots & a_{2,n} \\  
    & \vdots & \ddots & \vdots \\  
    & a_{m,2} & \cdots & a_{m,n} \\  
)$
```

Exemplo 5: Vetores

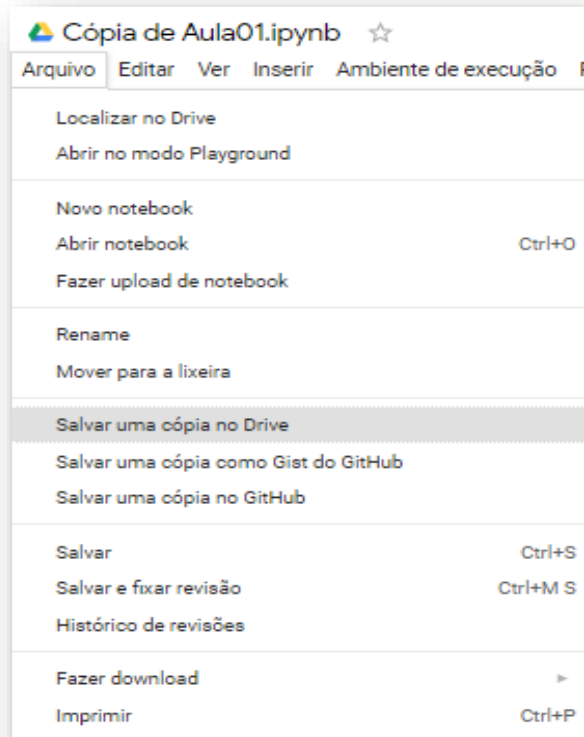
$$u_i(t) = x_i(t) + \beta(\hat{x}(t) - x_i(t)) + \beta \sum_{k=1}^{n_k} (x_{i1,k}(t) - x_{i2,k}(t))$$
$$f(x_1, x_2) = 20 + e - 20 \exp(-0.2 \sqrt{\frac{1}{n} (x_1^2 + x_2^2)}) - \exp(\frac{1}{n} (\cos(2\pi x_1) + \cos(2\pi x_2)))$$

Exemplo 6: Matriz

$$A_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

Google Colab em 10 + 2 Passos

Passo 5: Compartilhar seu Bloco de notas Colab



A screenshot of the Google Colab interface. The top bar shows the file name 'Cópia de Aula01.ipynb' and various menu items. A red dashed box highlights the 'Compartilhar' (Share) button. A large red arrow points from the text 'Clicar em Compartilhar' to this button. The main area displays a notebook with the title 'LINHAS DE CÓDIGO DO PROGRAMA' and some text. A sharing dialog box is open, showing the option to 'Copiar link' (Copy link). The dialog also shows the link 'https://colab.research.google.com/drive/1VYGZCYy0zG4--z3H90c-eDPAs...' and a dropdown menu for sharing permissions, with 'Editor' selected.

8:35

Google Colab em 10 + 2 Passos

Passo 6: princípios básicos do Python

Aula01.ipynb
Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda Não é possível salvar as alterações

Índice

LINHAS DE CÓDIGO DO PROGRAMA
Título em negrito.
Passo 03) Inserindo equações matemáticas nos textos de anotações
Exemplo 1:
Passo 09) Princípios básicos do Python
Seção

+ Código + Texto Copiar para o Drive

$$u_i(t) = x_i(t) + \beta(x(t) - x_i(t)) + \beta \sum_{k=1}^n (x_{i,k}(t) - x_{i2,k}(t))$$

$$f(x_1, x_2) = 20 + e - 20\exp(-0.2\sqrt{\frac{1}{n}(x_1^2 + x_2^2)}) - \exp(\frac{1}{n}(\cos(2\pi x_1) + \cos(2\pi x_2)))$$

Exemplo 6: Matriz

$$A_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

▼ Passo 09) Princípios básicos do Python

```
n1 = int(input("Digite o valor 1: "))
n2 = int(input("Digite o valor 2: "))
n3 = 20
n4 = n1 + n2 + n3
print("Soma: " + str(n4))
```

Digite o valor 1: 3
Digite o valor 2: 2
Soma: 25

RAM
Disco

Compartilhar Editar

No C++ seria:

```
int main(){
    int n1, n2;
    cout << "Digite o valor 1: ";
    cin >> n1;
    cout << "Digite o valor 2: ";
    cin >> n2;
    n3=20;
    n4 = n1+n2+n3;
    cout << "Soma: " << n4 << endl; }
```

No Python para concatenar texto (*string*) com variável numérica é preciso usar a função **str()**

```
print ("Soma: " + str(n4))
```


Google Colab em 10 + 2 Passos

Passo 7: Executando seu Código Python

Para executar um código python, clique na célula de código e depois **pressione o botão Play à esquerda do código** ou use o atalho do teclado "**Command/Ctrl+Enter**". Para editar o código, basta clicar na célula e começar a editar.

▼ Passo 09) Princípios básicos do Python



```
n1 = int(input("Digite o valor 1: "))
n2 = int(input("Digite o valor 2: "))
n3 = 20
n4 = n1 + n2 + n3
print ("Soma: " + str(n4))
```

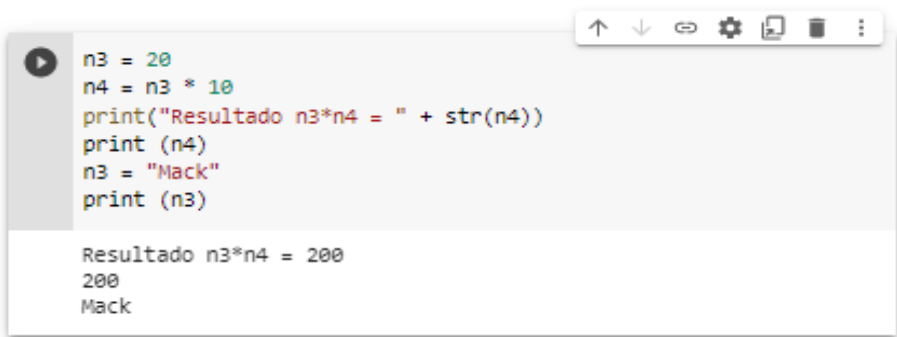


```
Digite o valor 1: 3
Digite o valor 2: 2
Soma: 25
```

Google Colab em 10 + 2 Passos

Passo 8: Declaração de variáveis no Python

A linguagem de programação Python, assim como a linguagem R, é fracamente tipificada; ou seja, a sintaxe usada para a declaração de uma variável é a mesma utilizada para fazer uma atribuição. Assim como na linguagem R na Python não existe a declaração do tipo da variável (inteira, ponto flutuante, String, etc.)? Diferentemente de outras linguagens, em Python, o tipo da variável é definido no momento da sua atribuição inicial. Porém, nada impede que você atribua um novo tipo à variável. Veja a sequência de comandos:



```
n3 = 20
n4 = n3 * 10
print("Resultado n3*n4 = " + str(n4))
print (n4)
n3 = "Mack"
print (n3)
```

Resultado n3*n4 = 200
200
Mack

Nesse exemplo:

A variável **n3** foi definida com o valor inteiro 20, portanto, **numérica**.

Foi feito um cálculo utilizando esse valor

Em seguida foi atribuída uma String para n3.

A partir dessa instrução, **n3** **passa a ser** tratada no programa como **String**.

As regras para definição do nome das variáveis são:

- ❖ Apenas uso de letras, números e *underscore* (`_`)
- ❖ O primeiro caractere não pode ser um número
- ❖ Não pode ser uma palavra reservada da linguagem (comando, função, etc.)

Convenção para nomenclatura de variáveis:

Nomes de variáveis devem ser escritos em letra minúscula, utilizando o *underscore* (`_`) para a separação de palavras ou ainda, iniciar a outra palavra com letra maiúscula. Exemplos:

```
nome_aluno  nomeAluno
salario_base salarioBase
nota_trabalho_final notaTrabalhoFinal
```

Google Colab em 10 + 2 Passos

Passo 9: Declaração de variáveis no Python e funções de conversão

```
x = 2 #tipo numérico
p = 3.1415 #tipo ponto flutuante
verdadeiro = True #tipo booleano
texto = 'isto é uma string' #tipo string
texto1 = "isto também é uma string" #tipo string
c = 3 + 2j #tipo número complexo

print(x)
print(p)
print(verdadeiro)
print(texto)
print(texto1)
```

```
2
3.1415
True
isto é uma string
isto também é uma string
```

Google Colab em 10 + 2 Passos

Passo 10: O Python tem algumas funções prontas que executam determinadas instruções. Uma função nada mais é que um nome, seguido de argumentos que são enviados como parâmetro de entrada para a função. Uma função pode ter mais de um argumento, que são separados por vírgula. Veja o código abaixo e indique quais são as funções utilizadas:



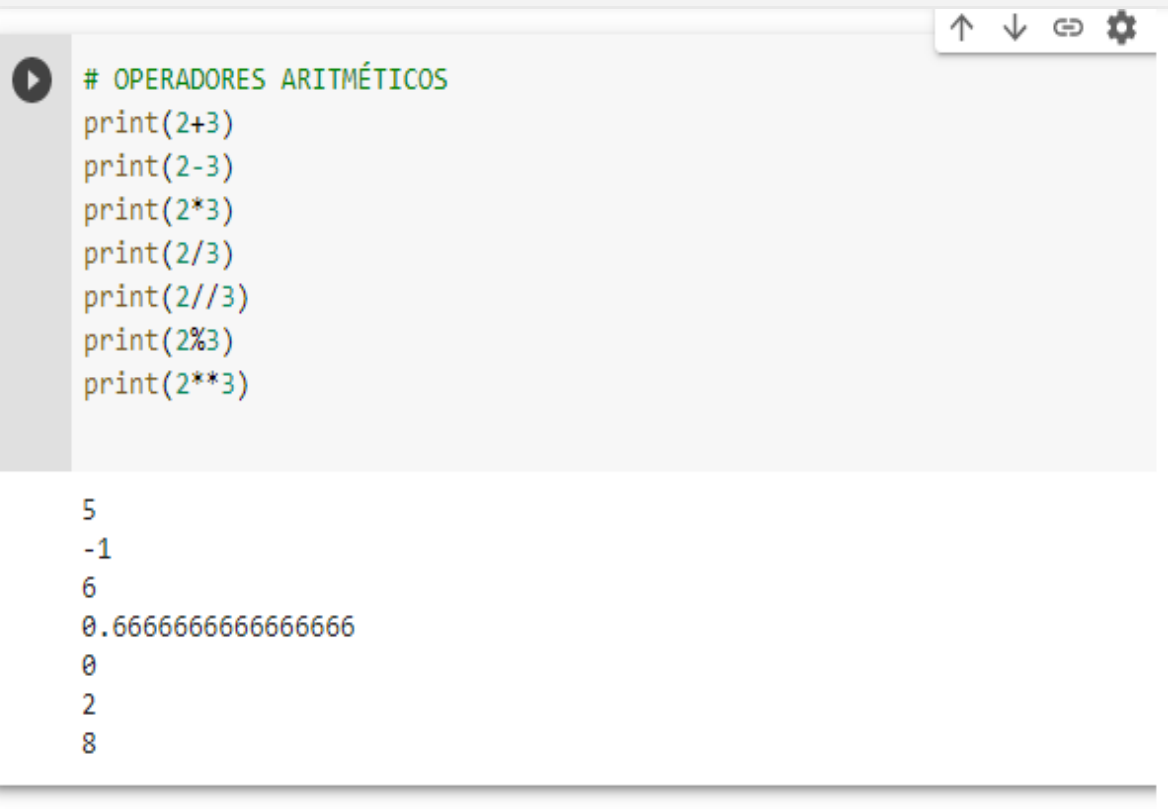
```
v1 = 10.5
v2 = 10
v3 = "true"
v4 = True
v5 = int(v1)
v6 = float(v2)
v7 = type(v3)
v8 = type(v4)

print(v1)
print(v2)
print(v3)
print(v4)
print(v5)
print(v6)
print(v7)
print(v8)
```

```
10.5
10
true
True
10
10.0
<class 'str'>
<class 'bool'>
```

Google Colab em 10 + 1 Passo

Lembre-se da ordem de precedência da análise de expressões que utilizam diversos operadores: primeiro os **aritméticos** (**exponenciação**, depois **multiplicação** e **divisão**, em seguida a **divisão truncada** e o **módulo** e finalmente a **soma** e a **subtração**), em seguida, os **relacionais** e por último, os **lógicos** (primeiro **not**, depois **and** e por último **or**).



```
# OPERADORES ARITMÉTICOS
print(2+3)
print(2-3)
print(2*3)
print(2/3)
print(2//3)
print(2%3)
print(2**3)
```

5
-1
6
0.6666666666666666
0
2
8

ARITMÉTICOS

+ soma

- subtração

* multiplicação

/ divisão

// divisão truncada

% módulo (resto)

** exponenciação

RELACIONAIS

== igual

!= não igual

< menor

> maior

<= menor ou igual

>= maior ou igual

LÓGICOS

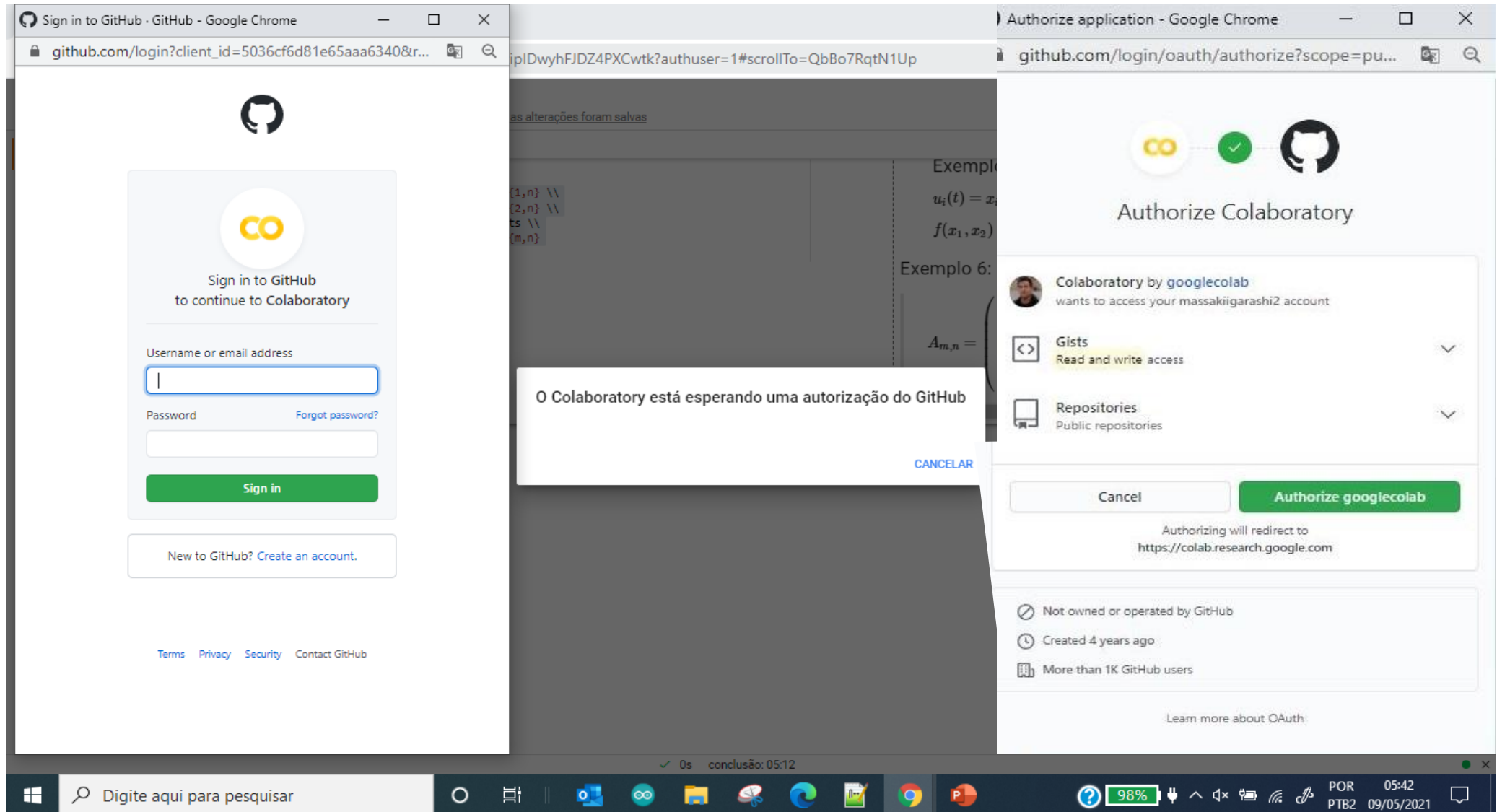
not negação

and e

or ou

Google Colab em 10 + 2 Passos

Passo +1: Salvando seu bloco de notas python No Git Hub



Google Colab em 10 + 2 Passos

Passo +2: Configurando seu repositório Git Hub

The image shows the GitHub 'Create a new repository' page. A blue thought bubble with the text 'Após criar seu repositório aguarde 1 min...' is positioned over the page. A red dashed line connects a box in a modal dialog to the 'Novo repositório' link in the GitHub page. The modal dialog, titled 'Copiar para o GitHub', contains the following text: 'Repositório: [link]', 'Não foi possível encontrar repositórios públicos. Novo repositório', 'Caminho do arquivo: Aula01.ipynb', 'Mensagem de confirmação: Criado usando o Colaboratory', and a checked checkbox 'Incluir um link para o Colaboratory'. The GitHub page shows the 'Create a new repository' form with fields for Owner (massakiigarashi2), Repository name (Aula01_Colab_Python), Description (Aula introdutória de Python utilizando ambiente Colab do Google), and options for Public/Private visibility and repository initialization (Add a README file, Add .gitignore, Choose a license).

Após criar seu repositório aguarde 1 min...

Copiar para o GitHub

Repositório: [link]
Não foi possível encontrar repositórios públicos. [Novo repositório](#)

Caminho do arquivo
Aula01.ipynb

Mensagem de confirmação
Criado usando o Colaboratory

☒ Incluir um link para o Colaboratory

CANCELAR OK

Create a new repository
A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * massakiigarashi2 / Repository name * Aula01_Colab_Python ✓

Great repository names are short and memorable. Need inspiration? How about [glowing-sniffle?](#)

Description (optional)
Aula introdutória de Python utilizando ambiente Colab do Google

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)

☒ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

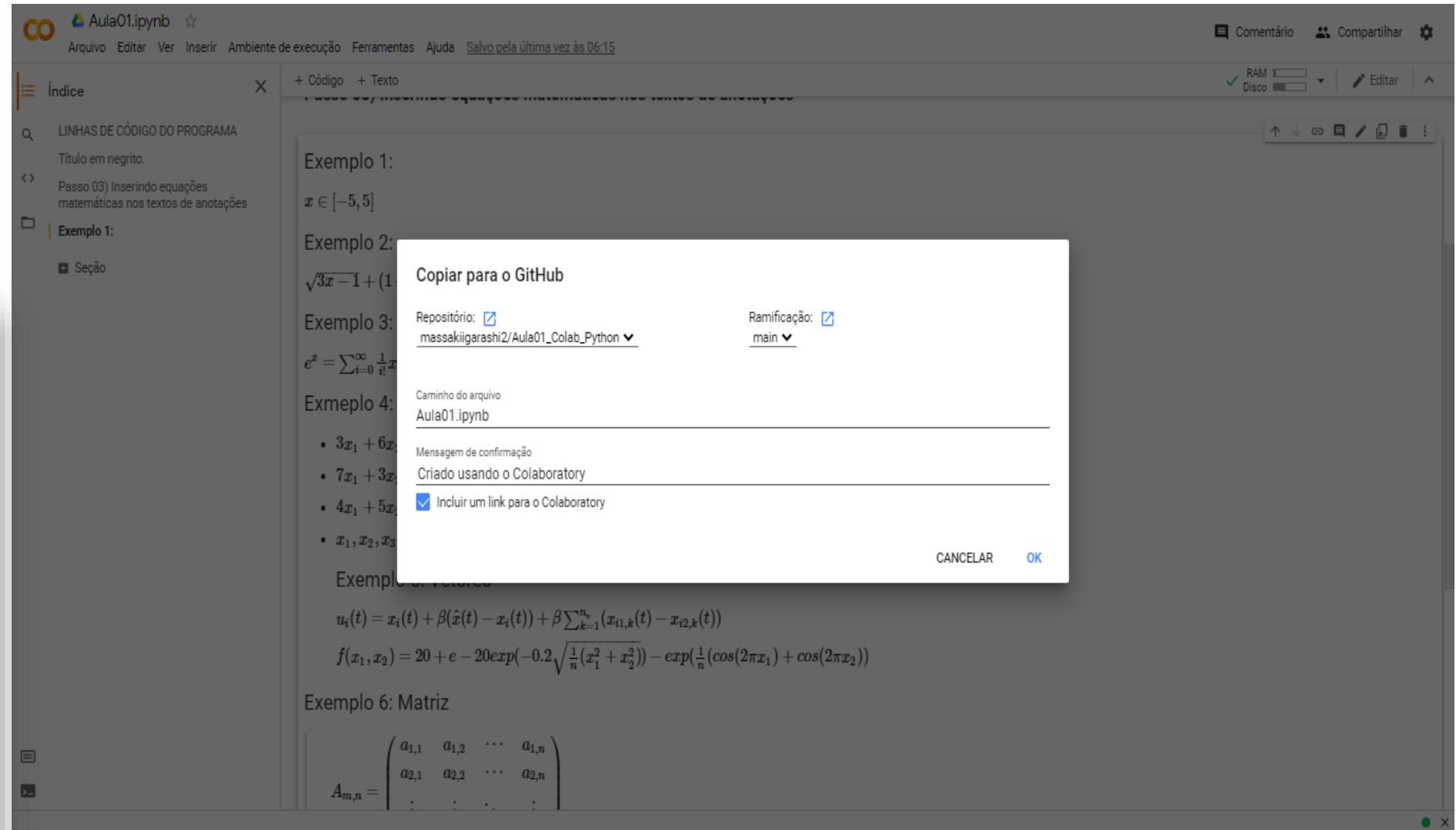
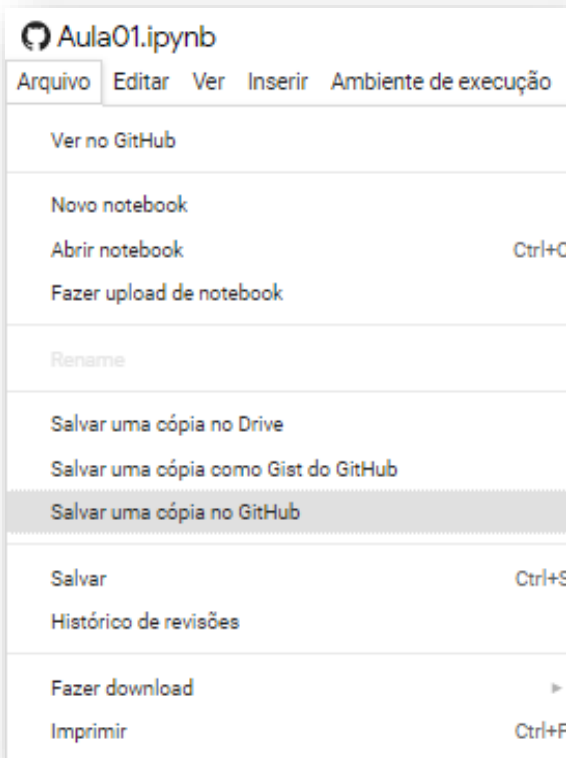
License: [Creative Commons Z...](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

Create repository

Google Colab em 10 + 2 Passos

Configurando seu repositório Git Hub



https://github.com/massakiigarashi2/Aula01_Colab_Python

Crie seu próprio Git Hub



The image shows the GitHub sign-up page. At the top, there's a navigation bar with the GitHub logo and links for 'Join GitHub · GitHub'. Below this, the main heading reads 'É muito simples criar um Git Hub!' in a large, stylized font, followed by 'Crie o seu agora!'. To the right of the heading, there's a link 'Already have an account?' and two buttons: 'Sign in' and 'Sign up'. A red arrow points from the bottom right towards the 'Sign up' button, with the word 'Clique!' written vertically inside the arrow. In the center, there's a dark blue box with the text 'Welcome to GitHub! Let's begin the adventure' and a prompt 'Enter your email' with a red arrow pointing to an input field. A 'Continue' button is located to the right of the input field. At the bottom, there's a small paragraph of text about terms of service and privacy.

Join GitHub · GitHub

github.com/signup?source=login

GitHub

É muito simples criar um Git Hub!

Already have an account? Sign in Sign up

Clique!

Welcome to GitHub!
Let's begin the adventure

Enter your email

Continue

By creating an account, you agree to the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#). We'll occasionally send you account-related emails.

Aula Passada: Desafio de Pratica!

Ex1 - Qual é a diferença entre o símbolo = e ==?

Ex2 - Analise o código abaixo:

```
peso = 120  
altura = 1.80  
ponto = \.'
```

Ex2- Para cada um dos comandos abaixo, indique o resultado da expressão e o tipo de cada um deles.

```
peso/2  
peso/2.0  
altura/3  
1 + 2 * 5  
ponto * 5
```

Ex3 - O que será exibido na tela?

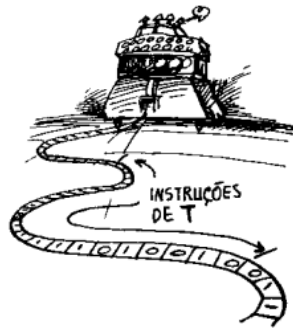
```
x = 'aa'  
y = x * 12  
print(y)
```

Você aprendeu o que é um Sistema!

Uma super “dica”!

Toda linguagem de programação tem:

1. Comandos de ENTRADA
2. OPERADORES MATEMÁTICOS
3. Comandos de CONDIÇÃO
4. Comandos de REPETIÇÃO
5. Comandos de SAÍDA



Processando um programa

Entrada(s)

```
a = input("Digite a: ")  
b = input("Digite b: ")
```

Processamento

```
soma = int(a) + int(b)
```

Saída

```
print(soma)
```

The screenshot shows a Jupyter Notebook titled "LPO1_INTRODUÇÃO_À_LINGUAGEM_DE_PROGRAMAÇÃO.ipynb". The interface includes a top bar with icons for file operations, editing, and execution, along with a "Compartilhar" (Share) button and a user profile picture. Below the top bar, there are tabs for "Código" (Code) and "Texto" (Text), and a "Copiar para o Drive" (Copy to Drive) button. The main content area displays a code cell labeled "#Exemplo 01" with the following Python code:

```
a = input("Digite a:")  
b = input("Digite b:")  
soma = int(a) + int(b)  
print(soma)
```

Below the code cell, the output is shown, indicating successful execution in 7 seconds:

```
Digite a:3  
Digite b:5  
8
```

The bottom status bar shows a green checkmark, the execution time "7s", and the conclusion time "conclusão: 22:34".

INSERINDO COMENTÁRIOS

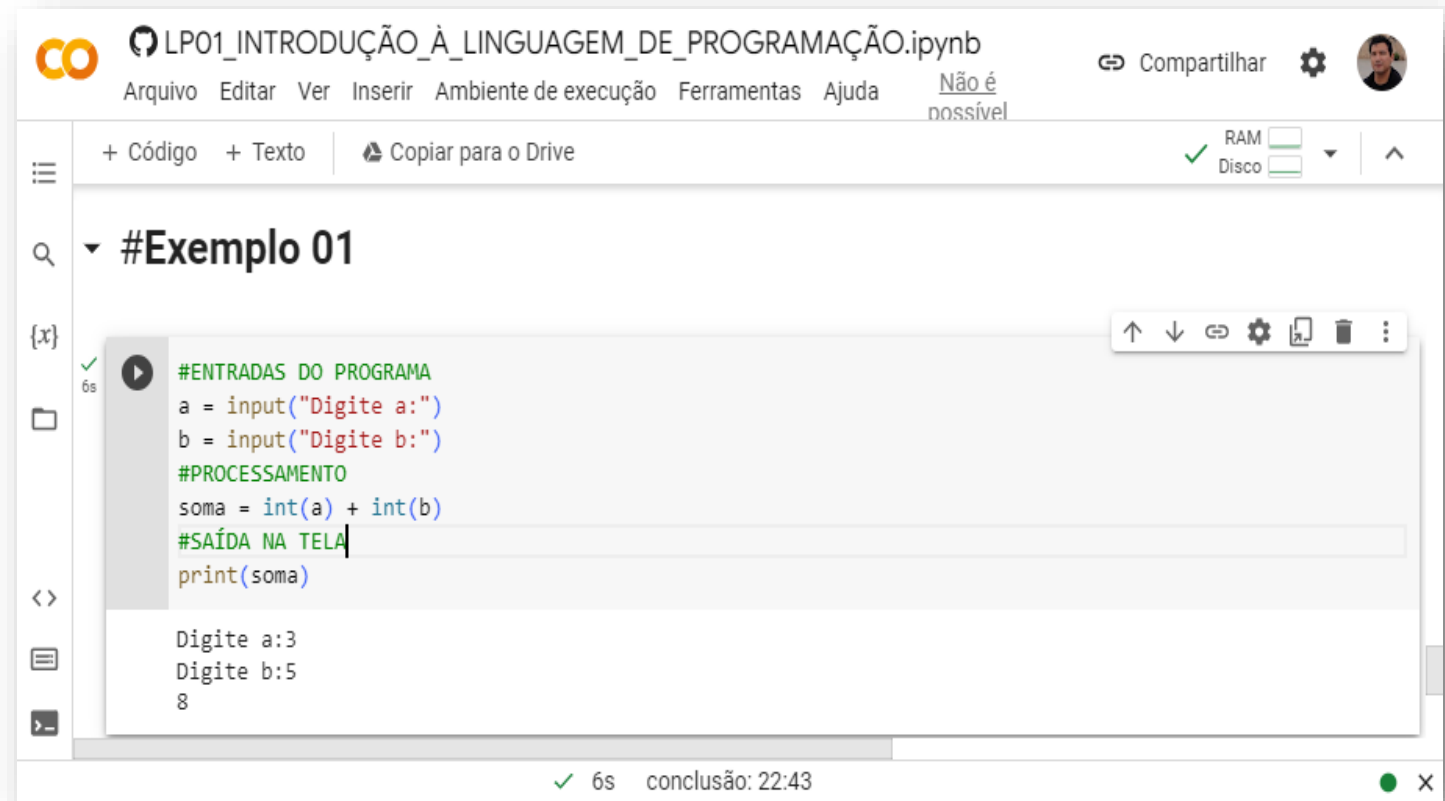
Por mais que até pareça algo desprezível o entendimento sobre como criar um comentário num programa, este talvez seja uma das premissas mais importantes em qualquer Linguagem de Programação; e python não poderia ser por menos.

O símbolo de **#** é utilizado na linguagem Python para realizar comentários nos programas; isto permite ao programador **documentar a funcionalidade(s)** de cada linha **ou inserir observações importantes ao longo do código**.

Mas talvez a aplicação mais importante de saber comentar um programa é que **quando você deseja fazer alguma modificação no seu código mas não tem certeza de que aquilo de fato funcionará, o que se faz é comentar o código escrito até o momento e fazer a modificação numa outra linha de código**; isto permitirá você voltar atrás caso sua modificação não funcione!

Símbolo **#**

#Comentário sem função de execução



The screenshot shows a Jupyter Notebook titled "LP01_INTRODUÇÃO À LINGUAGEM DE PROGRAMAÇÃO.ipynb". The code cell, labeled "#Exemplo 01", contains the following Python code with comments:

```
#ENTRADAS DO PROGRAMA
a = input("Digite a:")
b = input("Digite b:")
#PROCESSAMENTO
soma = int(a) + int(b)
#SAÍDA NA TELA
print(soma)
```

The output of the code execution is displayed below the code cell:

```
Digite a:3
Digite b:5
8
```

The interface also shows a file explorer on the left, a toolbar with icons for running, undo, redo, and other actions, and a status bar at the bottom indicating the execution time as 6s and the conclusion time as 22:43.

EXEMPLO 02

#ENTRADAS DO PROGRAMA

```
a = input("Digite a:")
```

```
b = input("Digite b:")
```

#PROCESSAMENTOS

```
soma = int(a) + int(b)
```

```
subtracao = int(a) - int(b)
```

```
multiplicacao = int(a) * int(b)
```

```
divisao = int(a)/int(b)
```

```
resto = int(a)%int(b)
```

```
potencia = int(a)**int(b)
```

#SAÍDAS NA TELA

```
print("Soma = " + str(soma))
```

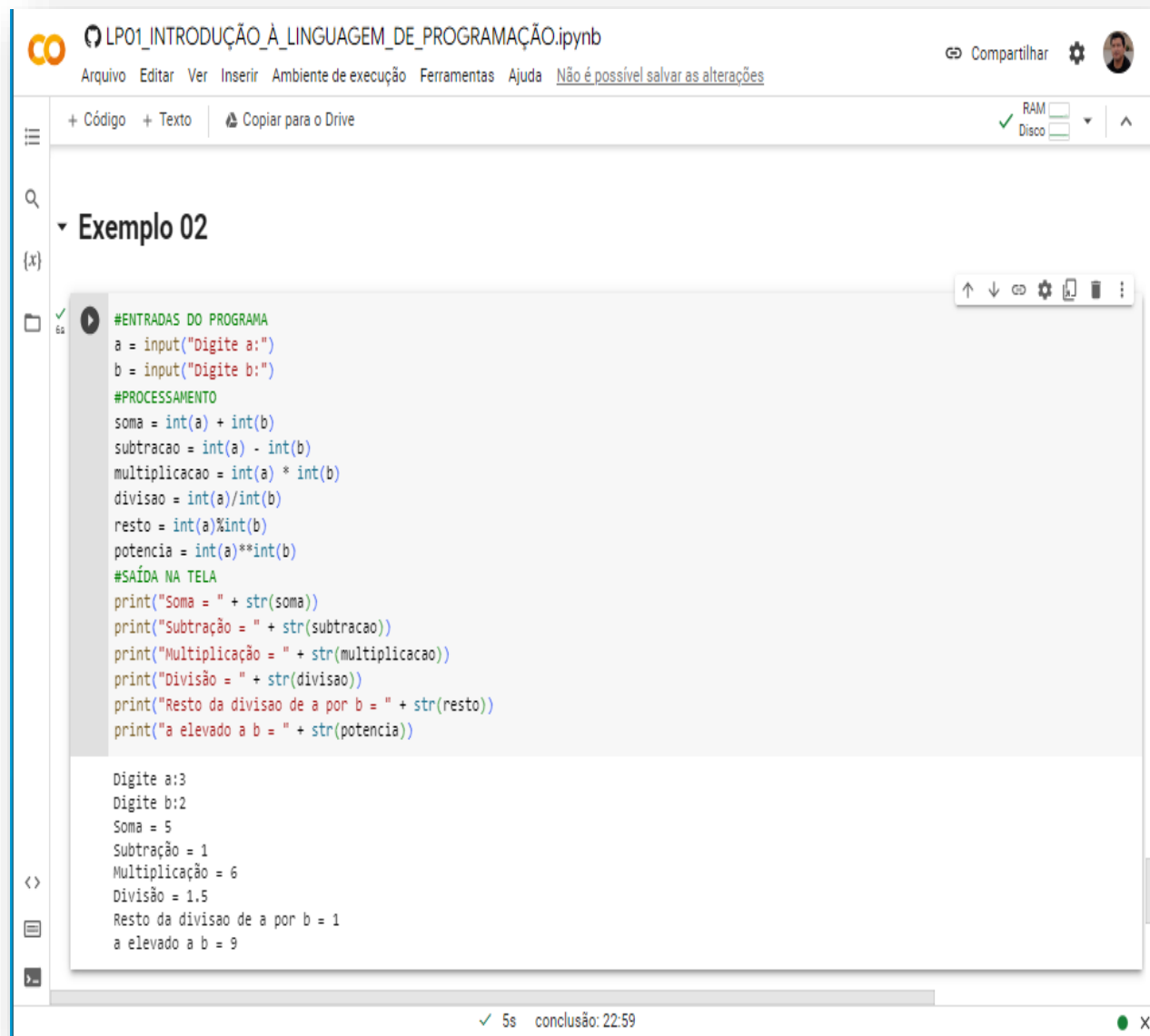
```
print("Subtração = " + str(subtracao))
```

```
print("Multiplicação = " + str(multiplicacao))
```

```
print("Divisão = " + str(divisao))
```

```
print("Resto da divisao de a por b = " + str(resto))
```

```
print("a elevado a b = " + str(potencia))
```



```
LP01_INTRODUÇÃO À LINGUAGEM DE PROGRAMAÇÃO.ipynb
Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda Não é possível salvar as alterações

+ Código + Texto Copiar para o Drive


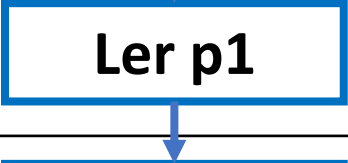

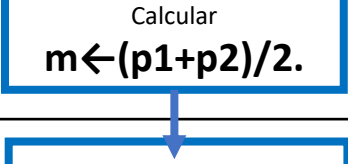
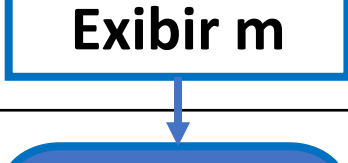

Exemplo 02

#ENTRADAS DO PROGRAMA
a = input("Digite a:")
b = input("Digite b:")
#PROCESSAMENTO
soma = int(a) + int(b)
subtracao = int(a) - int(b)
multiplicacao = int(a) * int(b)
divisao = int(a)/int(b)
resto = int(a)%int(b)
potencia = int(a)**int(b)
#SAÍDA NA TELA
print("Soma = " + str(soma))
print("Subtração = " + str(subtracao))
print("Multiplicação = " + str(multiplicacao))
print("Divisão = " + str(divisao))
print("Resto da divisao de a por b = " + str(resto))
print("a elevado a b = " + str(potencia))

Digite a:3
Digite b:2
Soma = 5
Subtração = 1
Multiplicação = 6
Divisão = 1.5
Resto da divisao de a por b = 1
a elevado a b = 9

5s conclusão: 22:59
```

EXEMPLO 03

NARRATIVA	FLUXOGRAMA	ALGORITMO	LINGUAGEM PYTHON
		Início real: p1, p2, m	
Ler nota na prova p1		Ler (p1)	p1 = float(input("Digite nota p1:"))
Ler nota na prova p2		Ler (p2)	p2 = float(input("Digite nota p2:"))
Calcular a Média (Onde $m = (p1+p2)/2$)		$m \leftarrow (p1+p2)/2.$	m = (p1 + p2)/2.0
Exibir a média calculada		Escrever (m)	print (m)
		Fim	

EXEMPLO 03

```
#ENTRADAS DO PROGRAMA
p1 = float(input("Digite nota p1:"))
p2 = float(input("Digite nota p2:"))
#PROCESSAMENTO
m = (p1+p2)/2.
#SAÍDA NA TELA
print("Média = " + str(m))
```



8:35

CO LPO2_INTRODUÇÃO À LINGUAGEM DE PROGRAMAÇÃO - Parte 1.ipynb ☆

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda Todas as alterações foram salvas

+ Código + Texto

RAM ☐ Disco ☐

Exemplo 03

Crie um código de programa em linguagem Python para Solicitar dois valores de nota p1 e p2 e em seguida calcular e exibir o resultado da média destas notas.

```
#ENTRADAS DO PROGRAMA
p1 = float(input("Digite nota p1:"))
p2 = float(input("Digite nota p2:"))
#PROCESSAMENTO
m = (p1+p2)/2.
#SAÍDA NA TELA
print("Média = " + str(m))
```

<> {x}

Digite nota p1:8.5
Digite nota p2:7.5
Média = 8.0

✓ 8s conclusão: 23:32

36

VAMOS PRATICAR! Crie programas Python para:

1. Dado um ângulo em radianos, elaborar um programa para converter este valor para graus.
2. Dado um ângulo em radianos, elaborar um programa para converter este valor para grados. Lembrar que 400 grados equivalem a 2π .
3. Dada uma medida em polegadas, elaborar um programa para converter o valor dado para milímetros. ($1''=25,4\text{mm}$)
4. Dada uma medida em milímetros, elaborar um programa para converter o valor dado para polegadas.
5. Dada uma temperatura em graus Celsius (c), elaborar um programa para converter a temperatura para graus Fahrenheit (f).
6. Dada uma temperatura em graus Fahrenheit, elaborar um programa para converter a temperatura para graus Celsius.
7. Dada uma temperatura em graus Celsius (c), elaborar um programa para converter a temperatura para graus Kelvin (k).
8. Dada uma temperatura em graus Fahrenheit, elaborar um programa para converter a temperatura para graus Kelvin.
9. São conhecidas as notas de um determinado aluno em uma determinada disciplina durante um semestre letivo: $p1$, $p2$, $t1$ e $t2$ com pesos respectivamente 3, 5, 1 e 1. São conhecidos também o total de aulas desta disciplina e a quantidade de aulas que o aluno assistiu. Elaborar um programa para calcular e exibir a média do aluno e a sua frequência.
10. Uma experiência foi realizada para determinar a aceleração da gravidade: uma bola caiu, a partir do repouso, do alto de um edifício e o tempo gasto para atingir o solo foi registrado. Dados a altura (a) e o tempo (t), determinar a aceleração da gravidade (g).

REFERÊNCIAS BIBLIOGRÁFICAS

<https://www.upgrad.com/blog/why-learn-python/>

<https://spectrum.ieee.org/top-programming-languages-2021#toggle-gdpr>

REFERÊNCIAS BIBLIOGRÁFICAS PYTHON e GOOGLE COLABORATORY

BORGES, Luiz Eduardo. **Python para desenvolvedores: aborda Python 3.3**. Novatec Editora, 2014.

VANDERPLAS, Jake. **Python data science handbook: Essential tools for working with data**. " O'Reilly Media, Inc.", 2016.

<https://colab.research.google.com/notebooks/intro.ipynb>