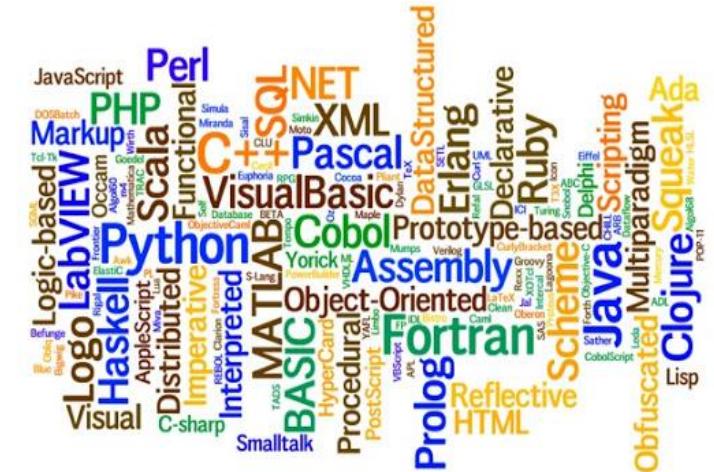




Algoritmos e Ling. de Programação

Prof. Me. Massaki de O. Igarashi

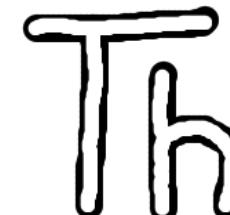
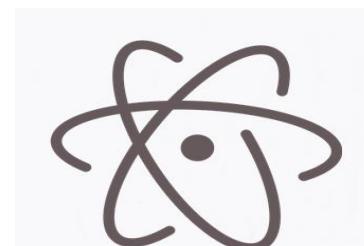
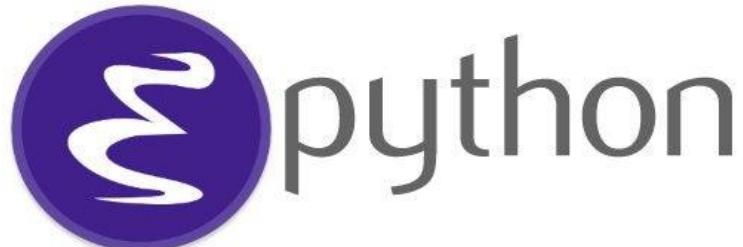
massaki.igarashi@anchieta.br



Dicas para o desenvolvimento do trabalho!

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 escritos em diferentes linguagens de programação.

Principais IDE's PYTHON:



Dicas para o desenvolvimento do trabalho!

1º) Fazer download e instalar o Anaconda Python:

<https://www.anaconda.com/products/distribution>

Anaconda Distribution

Download



For Windows

Python 3.9 • 64-Bit Graphical Installer • 594 MB

Get Additional Installers



streamlit lembreteAgora

2º) Após instalar o pcte Anaconda
Vá em Anaconda Prompt
e digite: pip install streamlit

Anaconda Prompt (Anaconda3) - streamlit hello

C:\Users\massaki\ pip install streamlit

```
general Options:
-h, --help
--isolated
-v, --verbose
-V, --version
-q, --quiet
--log <path>
--no-input
--proxy <proxy>
--retries <retries>
--timeout <sec>
--exists-action <act>
--trusted-host <host>
--cert <path>
--client-cert <path>
--cache-dir <dir>
--no-cache-dir
--disable-pip-version-check
--no-color
--no-python-version-warning
--use-feature <feature>
```

Show help.
Run pip in an isolated mode, ignoring environment variables. Option is additive.
Give pip full control over the environment. Option is additive.
Append log file. Option is additive.
Wait for input.
Proxy in the form [user:password@]host[:port]. Number of retries each connection attempt.
Set timeout (default 15 seconds).
Default action when a path already exists. Options are: ignore, replace, and error.
Mark this host or host:port pair as trusted.
Path to PEM-encoded CA certificate for more information.
Path to SSL client certificate, a certificate chain, or a private key.
Store the cache data in <dir>.
Disable the cache.
Don't periodically check PyPI to determine if packages have new versions.
Suppress colored output.
Silence deprecation warnings for upcoming functionality.
Enable new functionality, that may not yet be fully tested.

Digite aqui para pesquisar



Dicas para o desenvolvimento do trabalho!

RPAcourse_Tutorial_video_installation

vimeo.com/724995785

<https://vimeo.com/724995785>

vimeo Gerenciar Videos Recursos Funções Assistir Upgrade Pesquisar vídeos, pe M Novo vídeo

Anaconda | Anaconda Distribution

anaconda.com/products/distribution

ANACONDA Products Pricing Solutions Resources Partners Blog Company Contact Sales

Individual Edition is now
ANACONDA DISTRIBUTION

The world's most popular open-source Python distribution platform

Download For Windows Python 3.9 • 64-Bit Graphical Installer • 594 MB

Get Additional Installers

Windows | macOS | Linux

Aponte seu celular para ter acesso ao vídeo de instrução p/ instalação da Distribuição ANACONDA PYTHON



Dica

01

Dicas para o desenvolvimento do trabalho!

Anaconda Prompt (Anaconda3) - streamlit hello

(base) C:\Users\massaki\ pip install streamlit

General Options:

- h, --help Show help.
- isolated Run pip in an isolated mode, ignoring system site packages.
- v, --version Show version.
- q, --quiet Show quiet output.
- log-level Set log level.
- nolog Don't log anything.
- path Set path to configuration file.
- proxy Set proxy URL.
- cert Set certificate bundle.
- client-timeout Set client timeout.
- no-color Suppress colored output.
- no-pygments-warnings Silence deprecation warnings for upcoming changes.
- use-feature <feature> Enable new functionality, that may be experimental.
- use-deprecated <feature> Enable deprecated functionality, that may be removed.

3º) Após instalar o pkte Streamlit
Usando o **pip install streamlit**
Ou
pip3 install streamlit-webrtc

Vá no prompt e digite agora
Streamlit hello

streamlit hello

Welcome to Streamlit. Check out our demo in your browser.

Local URL: <http://localhost:8501>
Network URL: <http://192.168.53.59:8501>

Ready to create your own Python apps super quickly?
Head over to <https://docs.streamlit.io>

May you create awesome apps!

Hello

localhost:8501

Welcome to Streamlit! 🙌

Streamlit is an open-source app framework built specifically for Machine Learning and Data Science projects. 🎉 Select a demo from the sidebar to see some examples of what Streamlit can do!

Want to learn more?

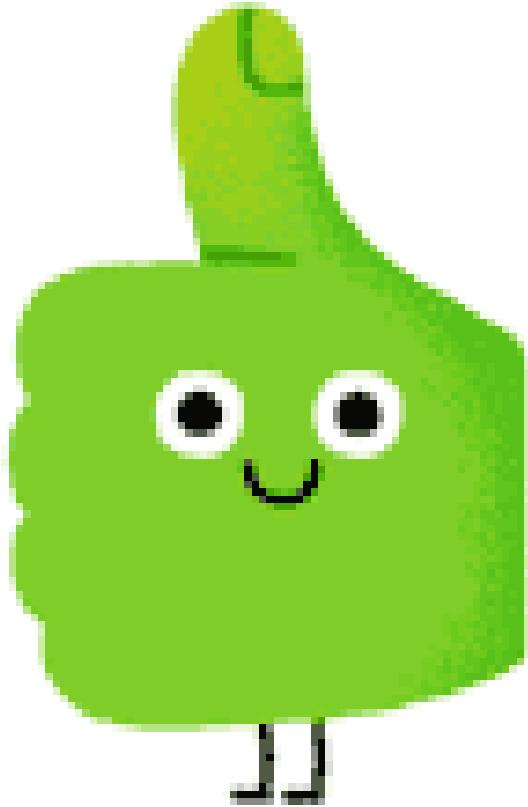
- Check out [streamlit.io](#)
- Jump into our [documentation](#)
- Ask a question in our [community forums](#)

See more complex demos

- Use a neural net to [analyze the Udacity Self-driving Car Image Dataset](#)
- Explore a [New York City rideshare dataset](#)



Dicas para o desenvolvimento do trabalho!



Parabéns!

**O ambiente está pronto
para iniciar!**



Crie seu próprio Git Hub



A screenshot of the GitHub sign-up page (github.com/signup?source=login). The page features a large, bold headline: "É muito simples criar um Git Hub!" (It's very simple to create a Git Hub!). Below it is the sub-headline: "Crie o seu agora!". A central input field is labeled "Enter your email" with a placeholder arrow icon. To the right of the input field is a "Continue" button. In the top right corner, there are links for "Sign in" and "Sign up". A red arrow points to the "Sign up" link, with the text "Clique!" (Click!) written vertically next to it. The bottom of the page contains a small note about account terms and privacy practices.

Join GitHub · GitHub

github.com/signup?source=login

Already have an account? [Sign in](#) [Sign up](#)

É muito simples criar um Git Hub!

Crie o seu agora!

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.



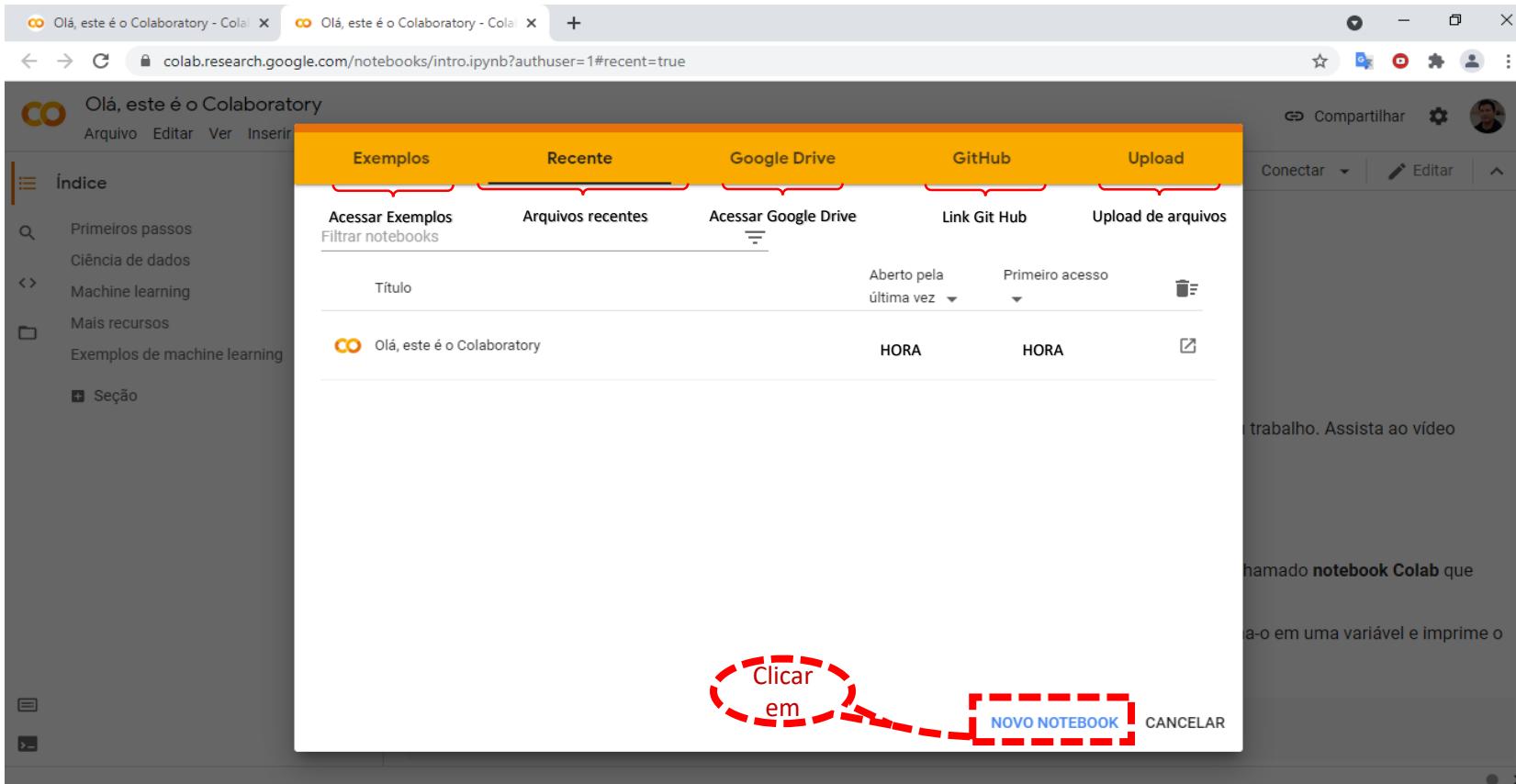
Dicas para o desenvolvimento do trabalho!

Defina o interpretador python a ser utilizado



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

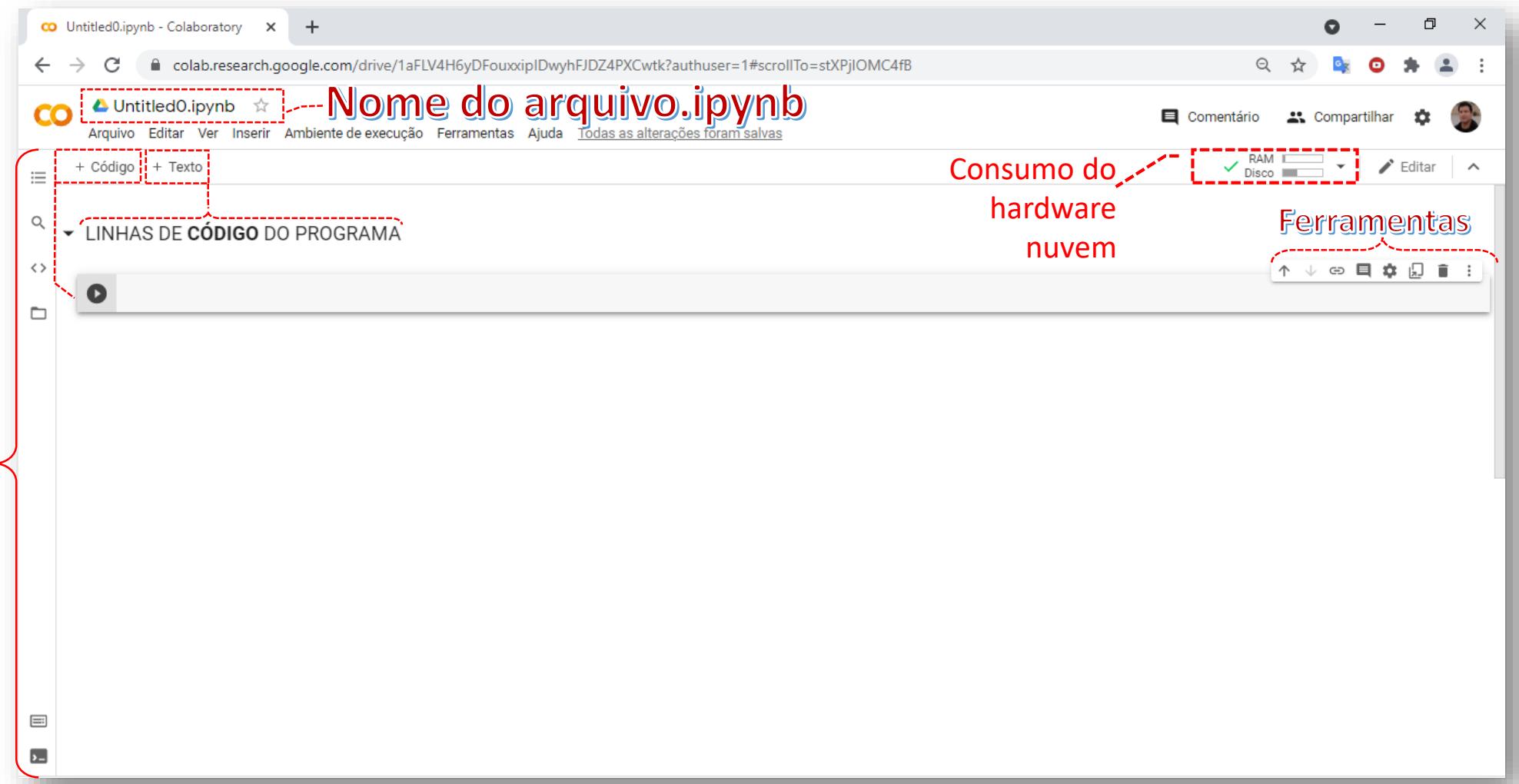
OBS: Seu navegador exibiria a seguinte tela (desde que você esteja logado em seu Google Drive e usando navegador Chrome):





O Google Colab em 12 Passos

Passo 01: Ao Clicar em NOVO NOTEBOOK o novo bloco de anotações do colab abrirá:





O Google Colab em 12 Passos

Passo 02: Saiba criar 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 desejar 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 Google Colab notebook titled "LP01_INTRODUÇÃO_À_LINGUAGEM_DE_PROGRAMAÇÃO.ipynb". The code cell contains the following Python code:

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

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

The code cell has a green checkmark icon indicating it was successfully run. The status bar at the bottom right shows "6s conclusão: 22:43".



O Google Colab em 12 Passos

Passo 03: saiba instalar e importar as bibliotecas necessárias

Agora suponha que você desenvolveu um aplicativo muito grande que inclui muitos módulos. À medida que o número de módulos aumenta, torna-se difícil controlá-los todos se forem despejados em um único local. Isto é especialmente verdade se eles tiverem nomes ou funcionalidades semelhantes. Você pode desejar um meio de agrupá-los e organizá-los.

Os pacotes permitem uma estruturação hierárquica do namespace do módulo usando **notação de ponto**. Da mesma forma que os módulos ajudam a evitar colisões entre nomes de variáveis globais, os pacotes ajudam a evitar colisões entre nomes de módulos. Criar um pacote é bastante simples, pois utiliza a estrutura hierárquica de arquivos inerente ao sistema operacional. Considere o seguinte arranjo:

Instalador pip

[pip](#) é o gerenciador de pacotes de referência Python. É usado para instalar e atualizar pacotes. Você precisará certificar-se de que possui a versão mais recente do pip instalada.

Você também pode instalar o pip por conta própria para garantir que possui a versão mais recente. É recomendado usar o sistema pip para inicializar uma instalação de usuário do pip:

```
python3 -m pip install --user --upgrade pip  
python3 -m pip --version
```

```
python3 -m pip install requests
```



The screenshot shows the Google Colab interface. At the top, there's a navigation bar with 'Arquivo', 'Editar', 'Ver', 'Inserir', 'Ambiente de execução', 'Ferramentas', and 'Ajuda'. Below the navigation bar, there are two buttons: '+ Código' and '+ Texto'. In the main workspace, there's a search bar with a magnifying glass icon and a play button icon. To the right of the search bar, the command '!pip install update pandas' is typed into the text area.



O Google Colab em 12 Passos

Passo 03: saiba instalar e importar as bibliotecas necessárias

Agora, vá até o **Google Colab**, lá você precisará usar eventualmente o instalador pip, mas principalmente, precisará instalar as bibliotecas: **pandas, urllib3 e io e requests**.

Dentro do comando **requests.get('')** insira o **link de compartilhamento** do Google Planilhas.



The screenshot shows a Google Colab notebook titled "Aula06_ImportDataFromGoogleSpreadSheet.ipynb". The code cell contains the following Python code:[] !pip install update pandas
{x} [7] import pandas as pd
import urllib3
from urllib3 import request
from io import BytesIO
import requests

[8] http = urllib3.PoolManager()
rD = requests.get('https://docs.google.com/spreadsheets/d/e/2PACX-1vTe21GxyDLN7cDFuV50_fkqoeMV8TZ5g3SMBsajcgnbzWhm6FFU8qGlurFSrYn_kzTIufwRlVyxH5-e/pub?gid=1623706195&single=true&output=csv')
dataD = rD.content
df = pd.read_csv(BytesIO(dataD), index_col=0)
df

Below the code, there is a feedback section:

Sua opinião sobre este evento é: Em poucas palavras, resuma este evento

Carimbo de data/hora

10/09/2023 19:20:29	uma Sugestão	Mais prática
10/09/2023 19:20:50	um Elogio	Excelente



O Google Colab em 12 Passos

Passo 04: sempre indentar corretamente o programa

Na programação de computadores , o ato de indentar o programa (criar recuo) é uma convenção que governa o recuo de blocos de código para transmitir a estrutura e hierarquia do programa. A indentação não é uma obrigação da maioria das linguagens de programação, onde é usada como notação secundária.

Mas na linguagem Python, ela tem uma função muito importante, podendo, inclusive, ser causa de erros de execução de muitos programas.

Ao criar um looping para que uma variável i conte de 0 até < 11 e imprima seu valor na tela,

```
for i in range(11):
    print(i)      #observe que esta linha ficou com recuo de 1 TAB
```

**Espaço feito com a
tecla TAB do teclado**

ou seja, a de 0 até 10, é preciso indentar os comandos que você executará no looping:



O Google Colab em 12 Passos

Passo 05: saiba usar os operadores básicos

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



O Google Colab em 12 Passos

Passo 06: 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:

A screenshot of the Google Colab interface. On the left, there's a code cell containing Python code. On the right, the output of the code is shown. The code defines two variables, n3 and n4, performs a multiplication, and prints the result. The output shows the expected numerical result followed by the string 'Mack'.

```
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



O Google Colab em 12 Passos

Passo 06: Declaração de variáveis no Python

The screenshot shows a Google Colab notebook cell. The code defines various variables with their types:

```
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)
```

The output below the code cell shows the results of the print statements:

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



O Google Colab em 12 Passos

Passo 07: Entenda a ferramenta Markdown para formatação de texto.

A screenshot of a Google Colab notebook titled "Untitled0.ipynb - Colaboratory". The notebook interface includes a toolbar with file operations, a sidebar with a search bar and a "+ Código" button, and a main workspace with a code editor and a preview pane. In the code editor, there is a single cell containing the following Markdown code:

```
## Título em **negrito**.  
## Sub-título em *itálico*.  
Texto ~tachado~.
```

The preview pane shows the rendered output of this Markdown:

Título em **negrito**.
Sub-título em *itálico*.
Texto ~~tachado~~.



O Google Colab em 12 Passos

Passo 07: inserindo equações matemáticas no seu texto Markdown.

Passo 03) Inserindo equações matemáticas nos textos de anotações

The screenshot shows a Google Colab notebook interface. On the left, a code cell contains various examples of mathematical expressions and equations. On the right, the same expressions are shown as rendered text and equations.

Code Cell Content:

```
## 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:  
- $3x_1 + 6x_2 + x_3 <= 28$  
- $7x_1 + 3x_2 + 2x_3 <= 37$  
- $4x_1 + 5x_2 + 2x_3 <= 19$  
- $x_1, x_2, x_3 \geq 0$  
  
## 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}
```

Renders on the Right:

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



O Google Colab em 12 Passos

Passo 08: Salvando seu bloco de notas python

The screenshot shows a Google Colab notebook titled "Aula01.ipynb". The "Arquivo" (File) menu is open, with "Salvar" (Save) highlighted. The main workspace contains two code cells. The first cell displays a matrix equation:

$$\begin{matrix} \text{matrix} \\ & \& a_{\{1,2\}} \& \cdots \& a_{\{1,n\}} \\ & \& a_{\{2,2\}} \& \cdots \& a_{\{2,n\}} \\ & \vdots \& \vdots \& \ddots \& \vdots \\ & \& a_{\{m,2\}} \& \cdots \& a_{\{m,n\}} \end{matrix}$$

The second cell contains mathematical examples:

Exemplo 5: Vetores

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

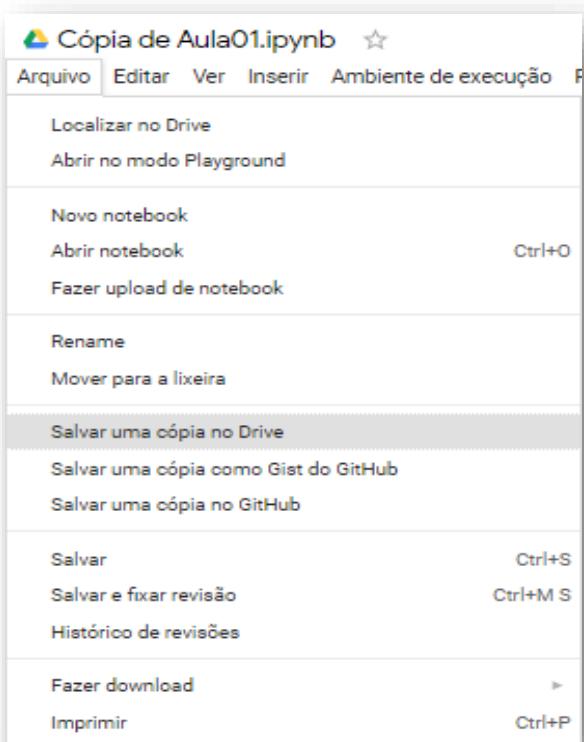
Exemplo 6: Matriz

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



O Google Colab em 12 Passos

Passo 09: Compartilhar seu Bloco de notas Colab



Cópia de Aula01.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Fazer download Imprimir

Índice

LINHAS DE CÓDIGO DO PROGRAMA

Título em negrito.

Passo 03) Inserindo equações matemáticas nos textos de anotações

Exemplo 1:

Título em negrito.

Sub-título em itálico.

Texto tachado.

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:

- $3x_1 + 6x_2 + x_3 = < 28$
- $7x_1 + 3x_2 + 2x_3 = < 37$

Clicar em
Compartilhar



O Google Colab em 12 Passos

Passo 10: Executando seu Código Python

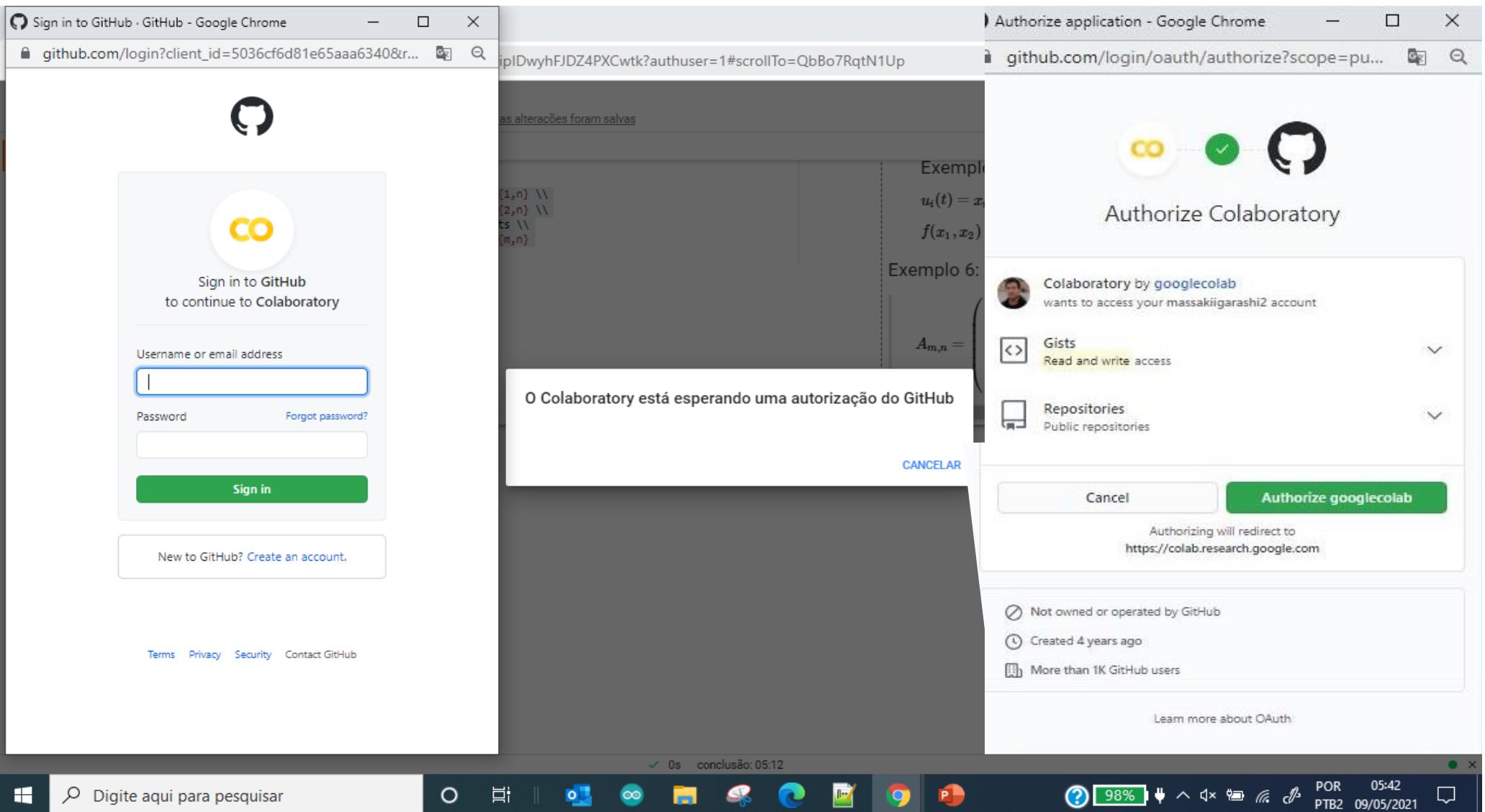
Para executar um código python, clique na célula de código e depois **pressione o botão Play à esquerda do código** ou use o atalho do teclado "**Command/Ctrl+Enter**". Para editar o código, basta clicar na célula e começar a editar.

▼ Passo 09) Princípios básicos do Python

A screenshot of the Google Colab interface. On the left, there is a code cell containing Python code. The first line of code, which contains a play button icon, is highlighted with a red dashed box. The code itself is:n1 = int(input("Digite o valor 1: "))
n2 = int(input("Digite o valor 2: "))
n3 = 20
n4 = n1 + n2 + n3
print ("Soma: " + str(n4))On the right side of the interface, the output of the code is shown in a light gray box. It displays three lines of text: "Digite o valor 1: 3", "Digite o valor 2: 2", and "Soma: 25". Above the code cell, there is a toolbar with various icons for navigating and managing the notebook.

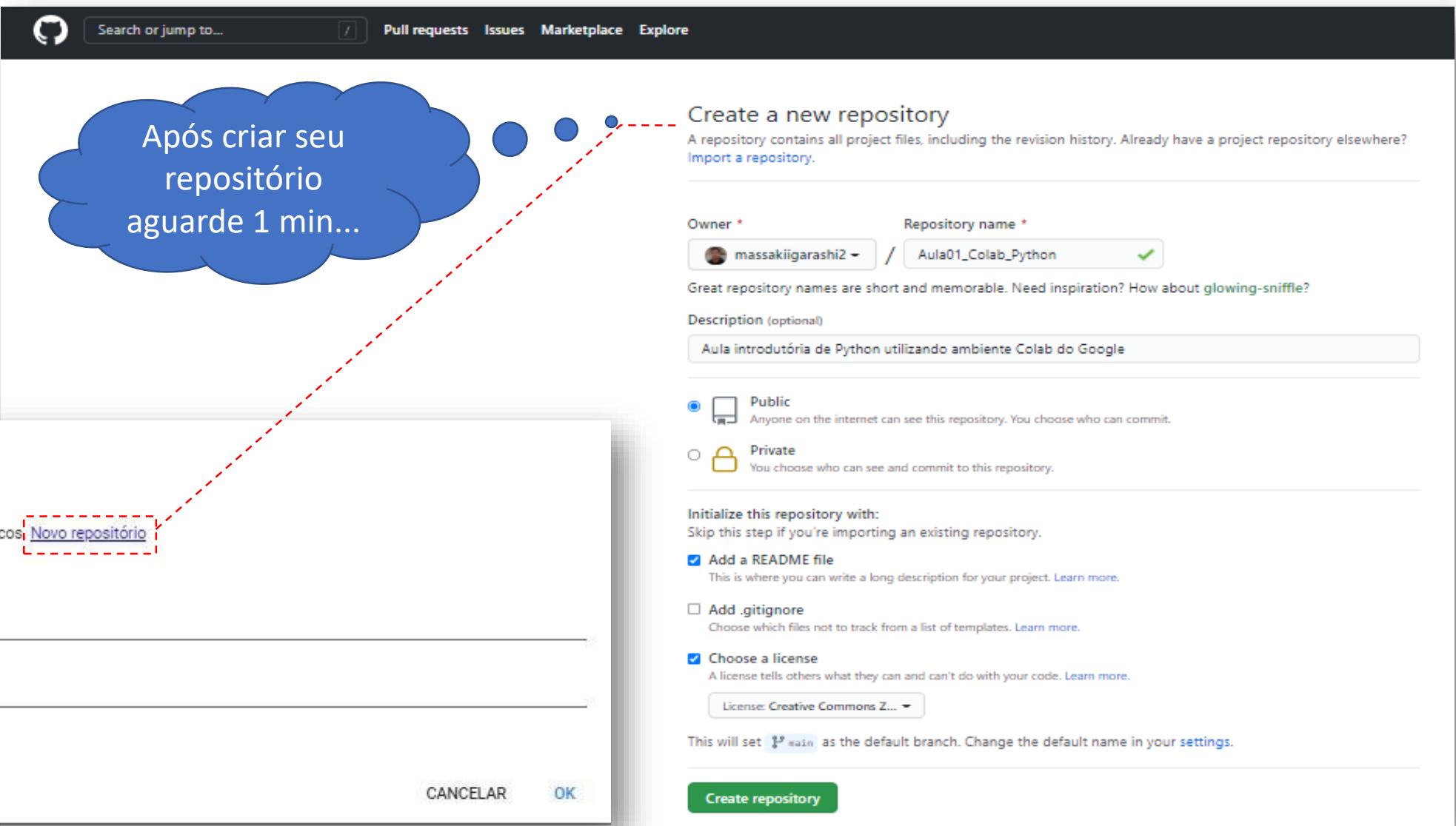
O Google Colab em 12 Passos

Passo 11: Salvando seu bloco de notas python No Git Hub



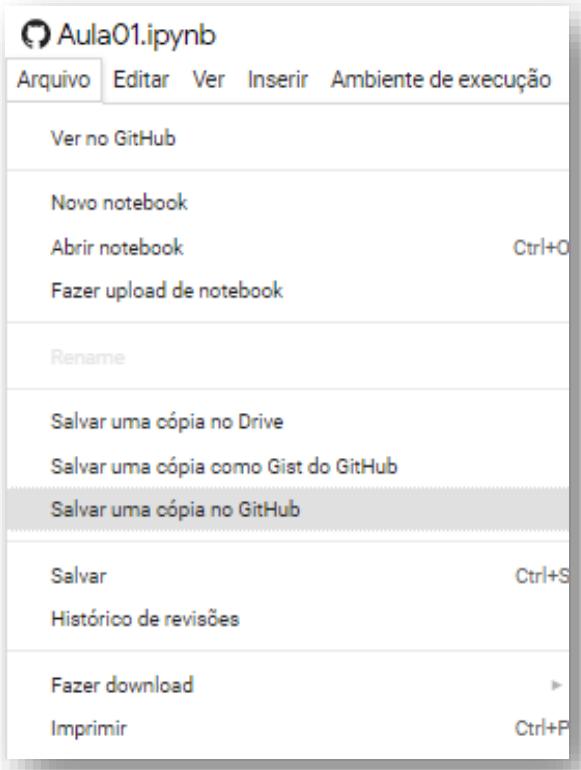
O Google Colab em 12 Passos

Passo 11: Configurando seu repositório Git Hub



O Google Colab em 12 Passos

Configurando seu repositório Git Hub



Aula01.ipynb

Arquivo Editar Ver Inserir Ambiente de execução

Ver no GitHub

Novo notebook

Abrir notebook Ctrl+O

Fazer upload de notebook

Rename

Salvar uma cópia no Drive

Salvar uma cópia como Gist do GitHub

Salvar uma cópia no GitHub

Salvar Ctrl+S

Histórico de revisões

Fazer download

Imprimir Ctrl+P

LINHAS DE CÓDIGO DO PROGRAMA

Título em negrito.

Passo 03) Inserindo equações matemáticas nos textos de anotações

Exemplo 1:

Exemplo 2:

Exemplo 3:

Exemplo 4:

Exemplo 5:

Exemplo 6: Matriz

Copiar para o GitHub

Repositório: massakiigarashi2/Aula01_Colab_Python

Ramificação: main

Caminho do arquivo
Aula01.ipynb

Mensagem de confirmação
Criado usando o Colaboratory

Incluir um link para o Colaboratory

CANCELAR OK

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

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

https://github.com/massakiigarashi2/Aula01_Colab_Python



O Google Colab em 12 Passos

Passo 12: Entender que Python faz muito uso de Funções e Módulos

Basicamente um Módulo python é um arquivo.py contendo **uma ou mais funções** que você deseja incluir em seu aplicativo.



Uma função é um bloco de código independente. As linguagens de programação fornecem inúmeras funções criadas pelos fabricantes das mesmas e também por terceiros. As funções são guardadas em módulos mas também podem estar em bibliotecas disponíveis no repositório Pypi.

def mediaA2(v1, v2):
return (v1+v2)/2.0

The screenshot shows a Google Colab notebook titled 'aul06_ModuloPython.ipynb'. On the left, there's a sidebar with file navigation (Arquivos, sample_data, FuncoesMassaki.py) and creation options (Upload, Atualizar, Novo arquivo, Nova pasta). The main area contains four code cells:

- [25] `import FuncoesMassaki
FuncoesMassaki.greeting("Maria")`
Output: Olá, Maria
- [26] `import FuncoesMassaki as fm
fm.greeting("Maria")`
Output: Olá, Maria
- [27] `fm.mediaA2(3,5)`
Output: 4.0
- [28] `from FuncoesMassaki import mediaA2
mediaA2(3,5)`
Output: 4.0

On the right, a sidebar shows memory usage: RAM 100% and Disco 81.49 GB. A preview of the file 'FuncoesMassaki.py' is shown with the following content:

```
1 def greeting(name):  
2 | print("Olá, " + name)  
3  
4 def mediaA2(v1, v2):  
5 | return (v1+v2)/2.0
```

A palavra reservada **def** *inicia a definição de uma função*. Ela deve ser **seguida do nome da função** e da lista de parâmetros formais entre parênteses. Os comandos que formam o corpo da função começam na linha seguinte e devem ser indentados. A instrução **return** finaliza a execução e *retorna um valor da função*.

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

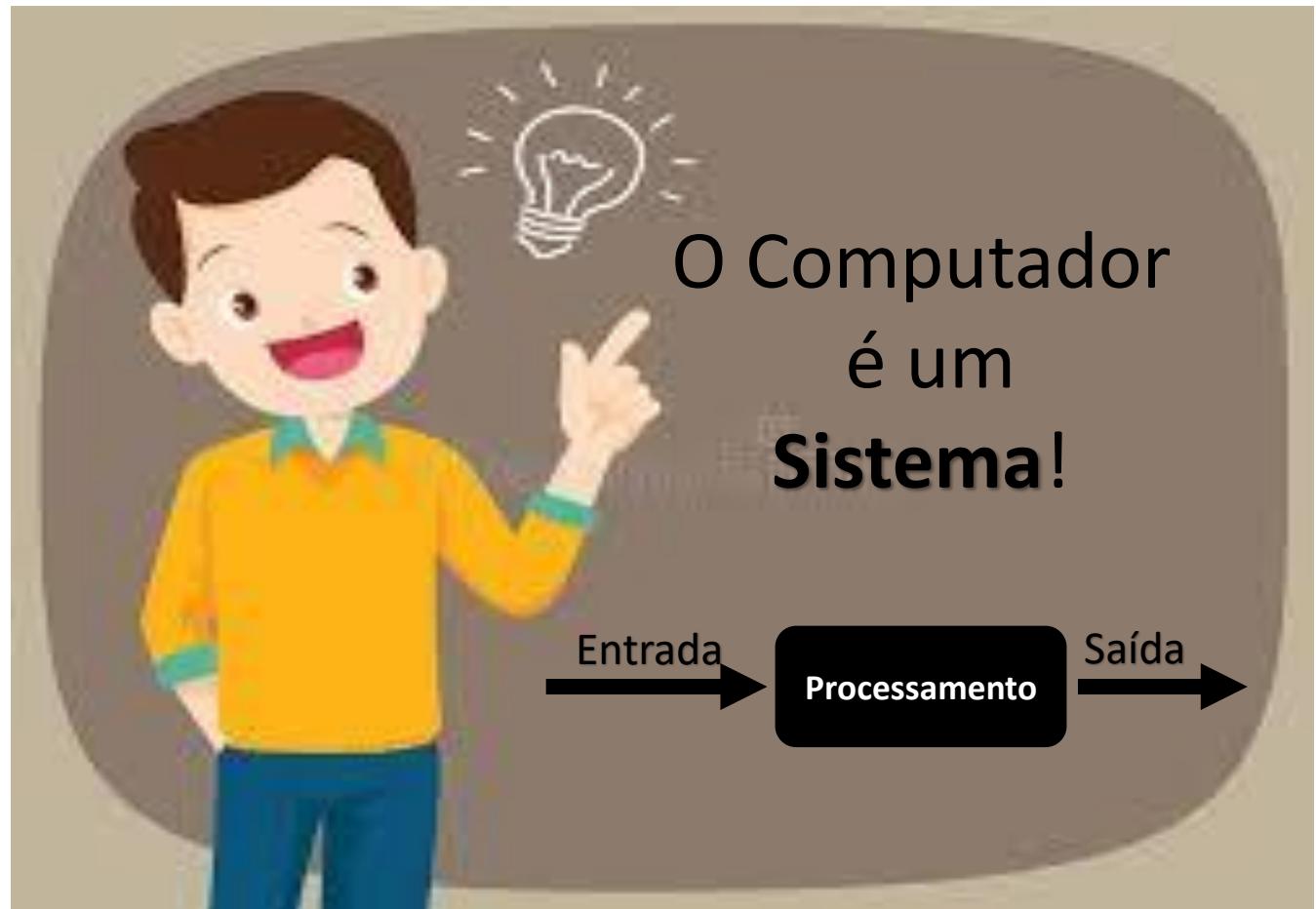
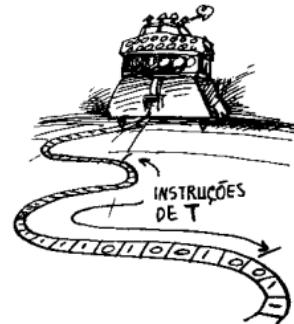
print(y)

Introdução: Você sabe 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



Introdução: O Sistema processando um programa

Exemplo 01: Crie um código em linguagem Python p/ Solicitar 2 valores a e b e em seguida calcular e exibir o resultado.

Entrada(s)

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

Processamento

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

Saída

```
print(soma)
```

1º Passo: Criar a Narrativa ou pseudo-código

```
Ler valor a  
Ler valor b  
Calcular soma=a+b  
imprimir soma
```

2º Passo: criar o Algoritmo do programa

```
início:  
    imprimir("Digite a: ") e ler_e_guardar(a)  
    imprimir("Digite b: ") e ler_e_guardar(b)  
    Processar soma = real(a) + real(b)  
    Imprimir(soma)
```

Fim

3º Passo: A partir do algoritmo, codificar na Linguagem de Programação escolhida (Neste caso, Python):

```
#ENTRADAS DO PROGRAMA  
# Imprimir("Digite a: ") e ler_e_guardar(a)  
a = input("Digite a:")  
# Imprimir("Digite b: ") e ler_e_guardar(b)  
b = input("Digite b:")  
#PROCESSAMENTO  
# Processar soma = real(a) + real(b)  
soma = float(a) + float(b)  
#SAÍDA NA TELA  
#Imprimir(soma)  
print(soma)
```



EXEMPLO 02

Narrativa ou pseudo-código

Ler valor a Ler valor b

Calcular soma=a+b

Calcular subtracao=a-b

Calcular multiplicacao=a*b

Calcular divisao=a/b

Calcular resto=a%b

Calcular potencia=a**b

imprimir soma

imprimir Subtração

imprimir Multiplicação

imprimir Divisão

imprimir Resto

imprimir Potência



**Crie o algoritmo
a partir da
Narrativa ao lado**

LP02_INTRODUÇÃO_A_LINGUAGEM_DE_PROGRAMAÇÃO_PYTHON.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda

+ Código + Texto Copiar para o Drive

Exemplo 02

Crie um código de programa em linguagem Python para Solicitar dois valores a e b e em seguida calcular e exibir o resultado de:

```
#ENTRADAS DO PROGRAMA
a = float(input("Digite a:"))
b = float(input("Digite b:"))

#PROCESSAMENTOS
soma = a + b
subtracao = a - b
multiplicacao = a * b
divisao = a / b
resto = a % b
potencia = a ** 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))
```

SCAN ME

EXEMPLO 03

NARRATIVA	FLUXOGRAMA	ALGORITMO	LINGUAGEM PYTHON
	<pre> graph TD INICIO([INÍCIO]) --> LER_P1[Ler p1] LER_P1 --> LER_P2[Ler p2] LER_P2 --> CALCULAR[Calcular m<=(p1+p2)/2.] CALCULAR --> EXIBIR_M[Exibir m] EXIBIR_M --> FIM([FIM]) </pre>	Início real: p1, p2, m	
Ler nota na prova p1	Ler p1	Ler (p1)	<code>p1 = float(input("Digite nota p1:"))</code>
Ler nota na prova p2	Ler p2	Ler (p2)	<code>p2 = float(input("Digite nota p2:"))</code>
Calcular a Média (Onde $m = (p1+p2)/2$)	Calcular $m \leftarrow (p1+p2)/2.$	$m \leftarrow (p1+p2)/2.$	<code>m = (p1 + p2)/2.0</code>
Exibir a média calculada	Exibir m	Escrever (m)	<code>print(m)</code>
	FIM	Fim	

EXEMPLO 03

CO LP02_INTRODUÇÃO_À_LINGUAGEM_DE_PROGRAMAÇÃO_PYTHON.ipynb

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

+ Código + Texto Copiar para o Drive

RAM Disco

Compartilhar M

Exemplo 03 Acessar Fluxograma

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))
```

Digite nota p1:7
Digite nota p2:8
Média = 7.5



REFERÊNCIAS BIBLIOGRÁFICAS

MANZANO, José Augusto Navarro G.; OLIVEIRA, Jayr Figueiredo de. **Algoritmos - Lógica para Desenvolvimento de Programação de Computadores.** [Digite o Local da Editora]: Editora Saraiva, 2019. *E-book*. ISBN 9788536531472. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788536531472>. Acesso em: 10 mar. 2024.

GOOGLE COLABORATORY:

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