

Agenda desta aula!

01

- Ementa/Propósito da Disciplina

02

- Conteúdo Programático

03

- A Ciência da Computação

04

- **Motivação** / Por que estudar Lógica de Programação?

05

- Como aprender a programar?

06

- Por que Python (para auxiliar)?

07

- Compiladores x Interpretadores

08

- Definições e Conceitos Básicos

EMENTA/Propósito

Estudar e entender conceitos de **algoritmos** e sua importância na computação; desenvolver algoritmos usando: **Narrativa, Pseudocódigo, Fluxogramas**; e codificar em Linguagem de programação (Python).

Desenvolver **Lógica** de Programação. Estudo dos Elementos básicos de programação: **variáveis e tipos de dados**; **entrada e saída de dados**. Compreender e aplicar estruturas de controle de fluxo (condicionais e laços de **repetição**); além de introduzir **conceitos básicos de estruturas de dados como vetores e matrizes (manipulação de arranjos estáticos)**, resolver problemas utilizando a lógica de programação e incentivar o desenvolvimento do pensamento computacional.

CONTEÚDO PROGRAMÁTICO

Introdução à Lógica de Programação

- O que é lógica de programação
- Importância dos algoritmos no desenvolvimento de software
- Visão geral da Ciência da Computação e da área de atuação
- Representação de Algoritmos (Narrativa, Pseudocódigo e Fluxograma)

Algoritmos e Pseudocódigo

- Definição e exemplos de algoritmos
- Pseudocódigo: sintaxe básica e estrutura
- Exercícios práticos de escrita de algoritmos simples

Fluxogramas

- Introdução aos fluxogramas: símbolos e convenções.
- Desenvolvimento de algoritmos utilizando fluxogramas
- Ferramentas de softwares para criação de fluxogramas

Variáveis, Tipos de Dados e Operadores

- Variáveis e sua importância.
- Tipos de dados fundamentais (inteiro, real, caractere, lógico)
- Operadores aritméticos, relacionais e lógicos

Estruturas de Controle de Fluxo

- Estruturas condicionais (se/senão, escolha/caso)
- Laços de repetição (para, enquanto, faça/enquanto)
- Exercícios práticos com aplicações reais

Estruturas de Dados Básicas

- Vetores e matrizes: conceitos e aplicações
- Manipulação de vetores e matrizes em algoritmos

Funções e Procedimentos

- Definição e diferenças entre funções e procedimentos.
- Declaração, chamada e passagem de parâmetros
- Exercícios práticos de modularização de algoritmos

Trabalho Prático em Grupo

- Usando linguagem Python

CÁLCULO DA MÉDIA

N1.1 (Nota Um.1)(de 0 a 5) - 16/abr/24

Atividade Avaliativa - Conteúdo para estudo:

- ❖ Introdução à Lógica de Programação
- ❖ Algoritmos: Narrativa, Pseudocódigo e Fluxogramas
- ❖ Variáveis, Tipos de Dados e Operadores

Entrega N1.2 (Nota Dois.2)(de 0 a 5) - 18/abr/24

TRABALHO EM GRUPO (MÁX. 3 estudantes por grupo)

Escolher um problema real e criar um programa em Python para auxiliar o usuário na vida diária com este problema. Sugestão de temas: automatização de processos ou web scraping). Conteúdo: Narrativa, Pseudocódigo, Fluxograma do Projeto.

N2.1 (Nota Dois.1)(de 0 a 5) – 04/jun/24

Atividade Avaliativa - Conteúdo para estudo:

- Estruturas de Dados (Arranjos Bi e Multi dimensionais)
- Funções e Procedimentos

N2.2 (Nota Dois.2)(de 0 a 5) - 18/abr/24

TRABALHO EM GRUPO: APRESENTAÇÃO: código python e demonstrar Programa Final.

$$N1 = N1.1 + N1.2$$

Onde: $0 \leq N1 \leq 10$

$$N2 = N2.1 + N2.2$$

Onde: $0 \leq N2 \leq 10$

$$MF = \frac{3N1 + 6N2}{10}$$

A CIÊNCIA DA COMPUTAÇÃO



O curso de Ciência da Computação é uma graduação que forma profissionais capazes de desenvolver soluções tecnológicas para as mais diversas áreas. Os estudantes aprendem a projetar e implementar sistemas computacionais. Durante a graduação, eles aprofundam em temas como inteligência artificial, segurança de dados, redes de computadores e banco de dados. O curso tem como objetivo formar profissionais capazes de compreender e resolver problemas complexos, utilizando as mais avançadas tecnologias disponíveis.

O profissional de Ciência da Computação atua basicamente na **elaboração de softwares**. Criando desde ferramentas simples, como:
um aplicativo financeiro para lançar despesas pessoais,
até programas mais complexos para:
gerenciamento de produção ou de processamento de informações.

Um cientista da computação, como é chamado o profissional formado em Ciência da Computação, pode ser contratado para atuar como:

- **Analista de Sistemas** – É responsável por elaborar documentos e alguns modelos que especificam os requisitos para o desenvolvimento de um software.
- **Arquiteto de Software** – Modela os sistemas e a infraestrutura que o software precisa para funcionar.
- **Gerente de TI** – Gerencia os projetos de software, coordenando equipes e planejando o desenvolvimento de sistemas.
- **Docência ou Pesquisa** – Trabalha como professor em instituições de ensino ou desenvolve pesquisas tecnológicas.
- Ou no departamento de **Pesquisa e Desenvolvimento (P&D)** de uma empresa

Introdução: Como aprender a Programar?

1 - Compreenda

2 - Retenha

3 – Pratique

4 – Dissemine

5 - Crie

DICAS IMPORTANTES:

- ❖ Não tenha medo de errar! Errar é normal e todos erram, principalmente no começo de um aprendizado ou de uma nova profissão; isso é também importante para sua evolução e crescimento.
- ❖ Nunca desista diante da primeira dificuldade; pergunte, peça auxílio! Como em qualquer jornada de aprendizado você enfrentará dificuldades; porém, não desista, o processo pode até ser longo e cansativo, mas saiba que superar uma dificuldade e alcançar um objetivo é algo que certamente contribuirá para o crescimento pessoal.
- ❖ Seja proativo(a); converse com colegas da área, participe de grupos e comunidades sobre o assunto; isto o ajudará a continuar evoluindo!
- ❖ Estude constantemente e revise os conhecimentos sempre que possível.
- ❖ Participe e desfrute o máximo desta aprendizagem de algo novo; você certamente vivenciará experiências únicas e provavelmente terá a chance de praticar e evoluir suas habilidades socioemocionais (muito valorizadas no mercado de trabalho atualmente). A comunicação e a facilidade de trabalhar em grupo são habilidades muito valorizadas hoje.

Motivação: Algumas frases para inspirar você a programar!

“ "Todos neste país deveriam aprender a programar um computador porque isto te ensina como pensar" (do inglês *"Everybody in this country should learn how to program a computer because it teaches you how to think"* - **Steve Jobs**).

“ "Se você deseja descobrir os segredos do universo ou apenas seguir uma carreira no século 21, a programação básica de computadores é uma habilidade essencial para aprender" (do inglês *"Whether you want to uncover the secrets of the universe, or you just want to pursue a career in the 21st century, basic computer programming is an essential skill to learn"* - **Stephen Hawking**).
























Justificativa: Por que Python como linguagem auxiliar?



- ✓ **Linguagem mais utilizada** hoje globalmente
- ✓ Fácil de aprender
- ✓ **Interoperabilidade** (comunica-se de forma transparente c/ 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

A
L
T
O

N
Í
V
E
L

Introdução: Linguagens de Baixo e Alto Nível

Linguagens de baixo nível

São as mais próximas do hardware do computador. Elas são compostas por instruções essencialmente binárias (Ex.: 0101010001100101). São difíceis para os humanos entenderem, já que são muito específicas e descrevem detalhes técnicos do hardware. A mais popular é a **Assembly**.

Linguagens de alto nível

São mais abstratas e fáceis de entender para os humanos. Elas se concentram em conceitos de alto nível, como variáveis, loops e funções.

As linguagens de alto nível são usadas para a maioria das aplicações de programação. Elas são mais fáceis de aprender e usar do que as linguagens de baixo nível, e permitem que os programadores se concentrem no algoritmo do programa, deixando os detalhes técnicos para a linguagem escolhida.

Exemplos de linguagens de alto nível incluem:

Python Java C C++ JavaScript

Linguagem Assembly

```
.INCLUDE "M32DEF.inc" ;inlui regs
.EQU OP1VAR = 0x109 ;reserva end.
.EQU OP2VAR = 0x10A ;dados p variável
.EQU RESVAR = 0x10B

.ORG 0 ;indica início do código
LDI R16, 5 ;carrega variáveis
STS OP1VAR, R16
LDI R16, 12
STS OP2VAR, R16
...
...
LDS R16, OP1 ;carrega R16 com OP1
LDS R17, OP2 ;carrega R17 com OP2
ADD R16, R17 ;soma, guarda em R16
STS RESVAR, R16 ;devolve res
;para mem. de dados
...
...
```

Linguagem C

```
#include <avr/io.h> //inlui regs

int OP1VAR, OP2VAR, RESVAR;
//declara variáveis

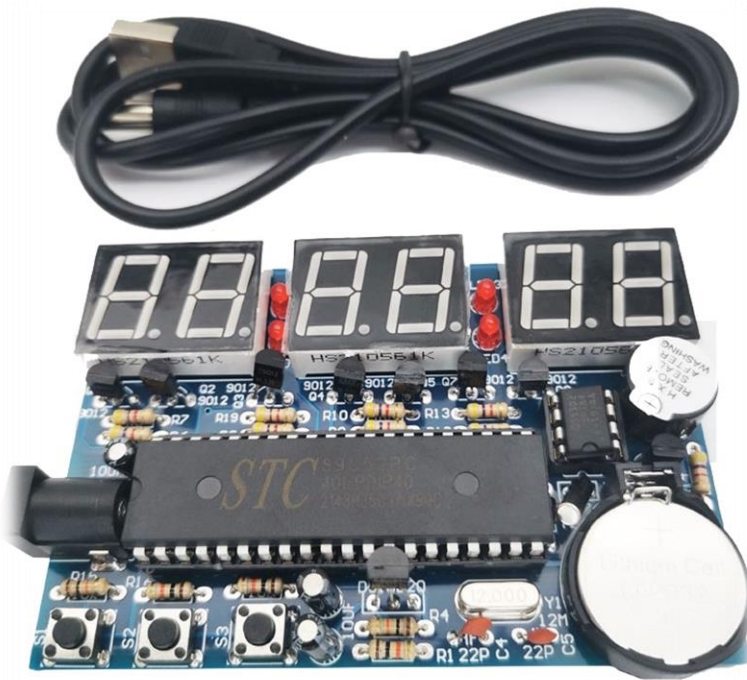
int main(void){ //indica início código
    OP1VAR = 5; //carrega variáveis
    OP2VAR = 12;
    ...
    RESVAR = OP1VAR + OP2VAR;
    //realiza a soma
    ...
    ...
}
```

Fonte: <https://brainly.com.br/tarefa/11562887>

Exemplos: Linguagens de Baixo e Alto Nível

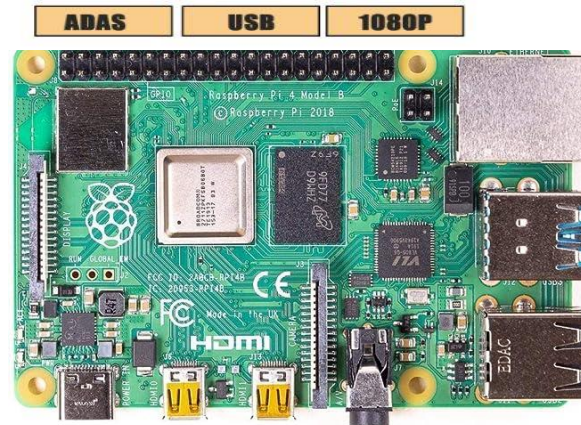
Ex: Programado em linguagem **Assembly**

Relógio de Ponto programado com
Linguagem Assembly
Placa com uControlador 8051



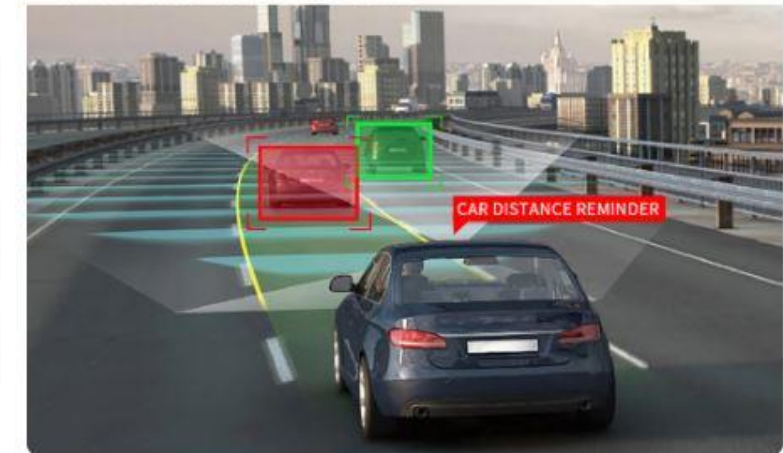
Ex: Programado em linguagem **Python**

Sistema Avançado de assistência ao motorista (do inglês Advanced Driving Assistance System - ADAS)



Front car Distance warning

Keeping the right distance is the key to driving safety. The distance between the vehicle and the front warning system, which automatically detects the distance between the vehicle and the front vehicle. When the distance is less than the system safety distance, an alarm model is issued to remind the user to ensure the safety of the user.



Fonte: <https://research.ijcaonline.org/volume95/number16/pxc3896794.pdf>

Introdução: 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++.

Introdução: Interpretador x Compilador

Os **interpretadores** passam pelo programa **linha a linha** e **executam cada comando**. Enquanto os **Compiladores** **traduzem todo o programa** escrito para formato no qual o computador entenda!

Tabela 1 – Vantagens e Desvantagens dos Interpretadores e Compiladores

	INTERPRETADORES	COMPIADORES
Vantagens	As linguagens interpretadas tendem a ser mais flexíveis ; já que oferecem recursos como digitação dinâmica e tamanho reduzido de programa	Os programas compilados tendem a ser mais rápidos . Pois o processo de traduzir o código em tempo de execução aumenta o tempo do processo.
Desvantagens	A desvantagem mais notável é a menor velocidade de execução em comparação com as linguagens compiladas.	Tempo adicional necessário para concluir toda a etapa de compilação antes dos testes; dependência da plataforma do código binário gerado.
	Ex: PHP, Ruby, Python, R, JS.	Ex: C, C++, C#, Visual Basic

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

Interpretadores
Python:



DEFINIÇÃO: Linguagem de Programação

Conforme o dicionário explica, **Linguagem** é qualquer meio sistemático (sistema de símbolos ou código) capaz de comunicar ideias ou sentimentos.

Por isso,

Linguagem de Programação tem o objetivo de **comunicar, dar instruções à um computador ou máquina** para atingir determinada finalidade.



A linguagem de Programação é um conjunto de **instruções lógicas e símbolos** escritas em um código fonte que permite a nós humanos traduzirmos 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.**



CONCEITOS BÁSICOS: Lógica

E o que são **instruções lógicas**?

Correspondem à **sequência de** passos (**ações**) necessários para que o programa cumpra seu objetivo. Nesse sentido, é importante um breve entendimento sobre **Lógica de Programação**.

Introdução para Lógica de Programação:

Basicamente, **nossas ações costumam seguir uma sequência lógica**; mas sequer damos atenção a isso! Por exemplo, ao fazer uma análise do nosso cotidiano, perceberemos que **nossas ações são consequência de uma cadeia de outras ações menores** que nos levaram até uma atitude final.

Exemplo: O simples **ato de acordar até tomamos café da manhã**.

Então qual seria a **sequência de ações menores** que efetuamos ao

- 1. Acordar;**
- 2. Prepararmos o café com auxílio de uma cafeteira elétrica;**
- 3. Colocarmos o café numa caneca e finalmente**
- 4. tomamos o café?**

Ao detalhar este processo somos capazes de estipular uma sequência de ações menores que nos levaram ao ato final :

Ao acordar:

1. Me espreguiço e abro os olhos;
2. Levanto-me da cama;
3. Calço os chinelos;
4. Caminho até o corredor;
5. Sigo pelo corredor até a cozinha;

Na cozinha,

6. Pego os recipientes do pó de café e do açúcar no armário;
7. Coloco os recipientes ao lado da cafeteira;
8. Pego uma colher de sopa e de café na gaveta;
9. Com a colher de sopa, coloco o pó de café dentro da cafeteira;
10. Pego um copo com água
11. Coloco água no compartimento específico;

Após inserir todos os ingredientes na máquina:

12. Aperto o botão de ligar;

Quando o café está pronto:

13. Pego a caneca e pires;
14. Despejo o café dentro de uma caneca;
15. Adoço o café;
16. Coloco a caneca com café sobre o pires junto à colher de café;

Bebo o café.

Agora é a sua vez!

Agora você está sendo convidada(o) a fazer uma análise do seu cotidiano; use a folha de papel fornecida pelo professor e documente a **sequência de ações que você realizou após tomar seu café da manhã até chegar até aqui.** Faça algo semelhante ao que o professor fez ao documentar a sequência lógica das suas ações ao acordar até tomar o café da manhã.



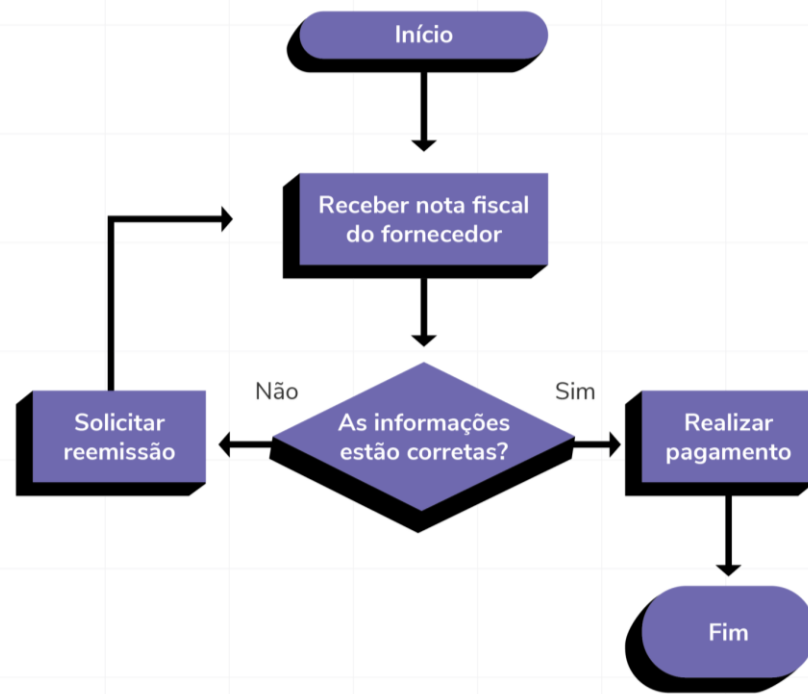
CONCEITOS BÁSICOS: Lógica de Programação

Lógica de programação é a organização coerente (coesa) de uma sequência de instruções voltadas à resolução de um problema, ou à criação de um software ou aplicação. É o primeiro passo antes de qualquer linguagem de programação.



ATENÇÃO

Exemplo: Processo simples



1. **Receber Nota Fiscal**
2. **Verificar** Informações
3. **SE** informações **CORRETAS ENTÃO**
Realizar Pagamento;
SENÃO
Solicitar reemissão;

Fonte: <https://resultadosdigitais.com.br/marketing/fluxograma-de-processo/>

DEFINIÇÃO: Algoritmo

“ Um **algoritmo** corresponde à sequência de passos, regras necessários para se atingir um objetivo; ou seja, é a **organização clara e precisa de uma sequência de instruções e operações** voltadas à resolução de um problema específico. ”

Assim,

O primeiro passo para qualquer programador é criar o algoritmo e avaliar se o resultado obtido condiz com a solução esperada. Em seguida, define-se a linguagem de programação a ser utilizada na implementação (codificação) do algoritmo documentado.

Em outras palavras, entender e **dominar a lógica de programação é a porta de entrada para** tornar-se um(a) programador(a), seja em **front-end** ou em **back-end**.

Os algoritmos são criados para automatizar procedimentos repetitivos!

Introdução: Pensar de maneira lógica possibilita

- Desenvolver códigos mais eficientes.
- Melhor resolução de problemas do cotidiano; já que facilita dividir problema complexo em pequenas partes, e, portanto, auxilia a encontrar uma solução mais eficaz;;
- Ajuda na concentração; pois quanto mais claras e ordenadas as ações melhor será a concentração
- Entender como a tecnologia em geral funciona; já que todos os processos existentes em TI dependem de um código que os sustenta.

INTRODUÇÃO: Representação dos Algoritmos

➡ As principais formas de representação:

➤ **Descrição Narrativa;**

É o que descrevemos, falamos. É a linguagem natural.

➤ **Fluxograma;**

É uma representação gráfica de uma procedimento, problema ou sistema, cujas etapas ou módulos são ilustrados de forma encadeada por meio de símbolos geométricos interconectados. É preciso conhecer e obedecer regras

➤ **Pseudocódigo** (também conhecido como portugol)

É uma linguagem, é como uma português estruturado para simular o programa propriamente dito.

Exercícios para praticar!

- 1) Criar a narrativa e o pseudocódigo de um programa que solicite ao usuário digitar uma temperatura em fahrenheit e convertê-la e exibir o resultado em graus Celsius. Onde $C = (5/9)(F-32)$
- 2) Criar a narrativa e o pseudocódigo de um programa que solicite ao usuário digitar uma medida em polegadas, elaborar um programa para converter o valor dado para milímetros. ($1''=25,4\text{mm}$).
- 3) Criar a narrativa e o pseudocódigo de um programa que solicite ao usuário digitar uma medida em milímetros, elaborar um programa para converter o valor dado para polegadas.
- 4) Criar a narrativa e o pseudocódigo de um programa que solicite ao usuário digitar sua idade e, em seguida, faça a verificação do valor; assim, se idade maior ou igual a 16 anos e menor que 70 anos, o programa deve retornar a mensagem “Você pode votar”. No caso contrário, ou seja, se menor que 16 anos ou maior que 70 anos “Sinto muito, infelizmente você não pode votar!”.

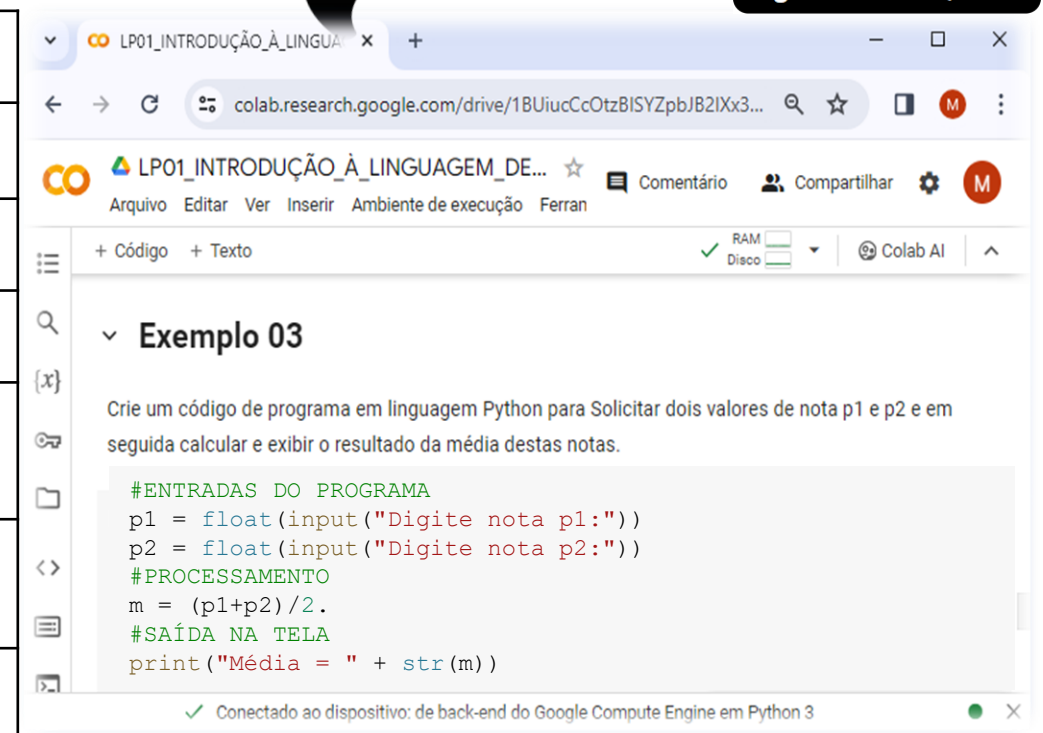
PRÓXIMA AULA PRATICAREMOS!

Fazer a Narrativa, o Fluxograma, o Pseudocódigo e código de um programa em Linguagem de Programação Python!



Aponte seu celular e Digitalize este QRcode

NARRATIVA	FLUXOGRAMA	PSEUDOCÓDIGO	LINGUAGEM PYTHON
	INÍCIO	Início real: p1, p2, m	
Ler nota prova p1	Ler p1	Ler (p1)	<code>p1 = float(input("Digite p1:"))</code>
Ler nota prova p2	Ler p2	Ler (p2)	<code>p2 = float(input("Digite p2:"))</code>
Calcular a Média ($m = (p1+p2)/2$)	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	



```
#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))
```

Ferramenta Útil!



Flowgorithm: permite aos iniciantes programarem baseado em fluxogramas gráficos. Isto permite ao aluno se concentrar nos conceitos de programação, em vez de em todas as nuances de uma linguagem de programação típica. Os programas podem ser executados diretamente no Flowgorithm. Link para acesso: <http://flowgorithm.org/>

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

