



Prof. Dr. Juliano Schimiguel

Prof. Me. Massaki Igarashi

Prof. Amadeu Vinicius

Prof. Clayton Valdo

Programação Orientada a Objetos

CONTEÚDO PROGRAMÁTICO

	DOM	SEG	TER	QUA	QUI	SEX	SÁB	AULA	CONTEÚDO
JUL				24	25	26	27	-	24/07: Início do semestre letivo – CAI - CT
	28	29	30	31	01	02	03	-	
AGO	04	05	06	07	08	09	10	Aula0	Planejamento e RECEPÇÃO
	11	12	13	14	15	16 01	17	Aula01	Apresentação - Conceitos do paradigma da programação orientada a objetos. - Introdução a classes e objetos. - Atributos, métodos e interação entre objetos.
	18	19	20	21	22	23 01	24	Aula02	- Sintaxe de linguagem de programação orientada a objetos - O que é abstração em orientação a objetos
	25	26	27	28	29	30 02	31	Aula 03	Como utilizar Interfaces. -Agregação e Composição de objetos.
	01	02	03	04	05	06 03	07	Sem.TI	Semana de TI
SET	08	09	10	11	12	13 04	14	Aula 05	- Encapsulamento - Herança e Polimorfismo.
	15	16	17	18	19	20 05	21	Aula 06	- Tratamento de Exceções.
	22	23	24	25	26	27 06	28	Aula 07	- Revisão e dúvidas para Prova P1
	29	30	01	02	03	04 07	05	P1	PROVA P1

	DOM	SEG	TER	QUA	QUI	SEX	SÁB	AULA	CONTEÚDO
OUT	06	07	08	09	10	11 08	12	Aula 08	POO usando Framework Streamlit Python
	13	14	15	16	17	18 09	19	Aula 09	Análise e Projeto orientado a objetos – Parte 1 15/09: Feriado dia dos Professores
	20	21	22	23	24	25 10	26	Aula 10	Análise e Projeto orientado a objetos – Parte 2
	27	28	29	30	31	01 11	02	Aula 11	Introdução ao teste de software.
	03	04	05	06	07	08 12	09	Aula 12	- Teste de software aplicado ao paradigma de orientação a objetos.
NOV	10	11	12	13	14	15	16	Aula 13	14/11: Dia Não Letivo – Reunião Pedagógica 15/11: Feriado Nacional: Proclamação República
	17	18	19	20	21	22 P2	23	P2	PROVA P2
	24	25	26	27	28	29 15	30	Aula15	Devolutiva Prova P2
	01	02	03	04	05	06 16	07	Aula16	APRESENTAÇÃO DOS PROJETOS
DEZ	08	09	10	11	12	13 17	14	Aula17	09/12: PROVA RECUPERAÇÃO
	15	16	17	18				-	18/12: ENCERRAMENTO SEMESTRE

Cálculo da Média

$$\textit{Média} = \frac{4N1 + 4N2 + 2N3}{10}$$

ALGORITMO

Os algoritmos são criados para automatizar procedimentos repetitivos!

“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**.

BENEFÍCIOS

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.

- **Descrição Narrativa;**

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

- **Fluxograma;**

É uma representação gráfica de um 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 português)

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

O QUE É?

PROGRAMAÇÃO ORIENTADA A OBJETOS?

A **Programação Orientada a Objetos** - POO é um **paradigma de programação**

Ou seja, é um exemplo que serve como modelo; se baseia na criação de **objetos** para organizar o código, que **são instâncias de classes**. Cada objeto representa uma **entidade** do mundo real, **com suas próprias propriedades (atributos)** e **comportamentos (métodos)**. Essa abordagem facilita a compreensão, manutenção e reutilização do código.

O QUE É?

- **Classe:** Modelo ou plano para criar objetos.
- **Objeto:** Instância concreta de uma classe.
- **Atributos:** Características de um objeto (variáveis).
- **Métodos:** Comportamentos de um objeto (funções).

CLASSE

Uma classe define as propriedades e comportamentos que um objeto pode ter. Ao criar um objeto, estamos instanciando uma classe e atribuindo valores aos seus atributos. Os métodos definem as ações que um objeto pode realizar.

- **Imagem:** Um diagrama UML representando uma classe com seus atributos e métodos, e um objeto sendo criado a partir dessa classe.

ENCAPSULAMENTO

- Proteção dos dados internos de um objeto.
- Acessadores (getters) e modificadores (setters).

O encapsulamento garante que os dados de um objeto só sejam modificados através de métodos específicos, evitando alterações indesejadas e facilitando o controle sobre o estado do objeto. Os getters e setters são métodos que permitem acessar e modificar os atributos de um objeto de forma controlada.

- **Imagem:** Um diagrama UML representando uma classe com atributos privados e métodos públicos para acessá-los.

- Criação de hierarquias de classes.
- Classes pai (base) e classes filhas (derivadas).
- Reutilização de código.
- Polimorfismo de inclusão.

A herança permite criar novas classes a partir de classes existentes, herdando seus atributos e métodos. Isso promove a reutilização de código e a organização de classes em uma hierarquia. O polimorfismo de inclusão ocorre quando uma classe filha pode ser utilizada no lugar de uma classe pai.

• **Imagem:** Um diagrama UML representando uma hierarquia de classes com herança simples e múltipla.

POLIMORFISMO

- Objetos de classes diferentes podem ser tratados como mesma classe.
- Polimorfismo de sobrecarga e sobreescrita.
- Importância para a flexibilidade e extensibilidade do código.

O polimorfismo permite que um mesmo método tenha comportamentos diferentes em classes diferentes. Isso torna o código mais flexível e adaptável a diferentes situações. A sobrecarga ocorre quando um método tem o mesmo nome, mas diferentes parâmetros. A sobreescrita ocorre quando um método em uma classe filha tem a mesma assinatura de um método em sua classe pai, mas uma implementação diferente.

•**Imagem:** Um exemplo de código demonstrando polimorfismo com sobrecarga e sobreescrita.

EXEMPLO PYTHON

```
import numpy as np

class ClasseMSK:

    def __init__(self, ListaNUM):
        self.num_list = ListaNUM
        self.MEDIA = np.mean(self.num_list)
        self.MAX = np.max(self.num_list)
        self.MIN = np.min(self.num_list)
        self.STD = np.std(self.num_list)
```

self : Class Name we will use WITHIN the Class

Input: In this case a list

Class Attribute

Class Name

The screenshot shows a Jupyter Notebook window titled 'Classe_Objeto_Atributos_e_Método...'. The code in the cell is as follows:

```
import numpy as np
class ClasseMSK:
    def __init__(self, ListaNUM):
        self.num_list = ListaNUM
        self.MEDIA = np.mean(self.num_list)
        self.MAX = np.max(self.num_list)
        self.MIN = np.min(self.num_list)
        self.STD = np.std(self.num_list)
    def Somatorio(self):
        sum1 = 0
        for i in range(len(self.num_list)):
            sum1 = sum1 + self.num_list[i]
        return sum1
#TESTE DE UTILIZAÇÃO
Herdeiro = ClasseMSK([43, 3, 9, 54, 23, 56, 675, 23])
print("Média = ", Herdeiro.MEDIA)
print("Valor Máximo = ", Herdeiro.MAX)
print("Valor Mínimo = ", Herdeiro.MIN)
print("Desvio Padrão = ", Herdeiro.STD)
print("Somatório = ", Herdeiro.Somatorio())
```

The output of the code execution is:

```
Média = 110.75
Valor Máximo = 675
Valor Mínimo = 3
Desvio Padrão = 214.0530016140862
Somatório = 886
```

- ✓ <https://medium.com/analytics-vidhya/understanding-classes-and-methods-in-python-2a81a051226>
- ✓ <https://www.tutorialsteacher.com/python/property-function>

POR QUE PYTHON?

PROGRAMAÇÃO ORIENTADA A OBJETOS

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

ALTO NÍVEL

- ✓ **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);



INTRODUÇÃO

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

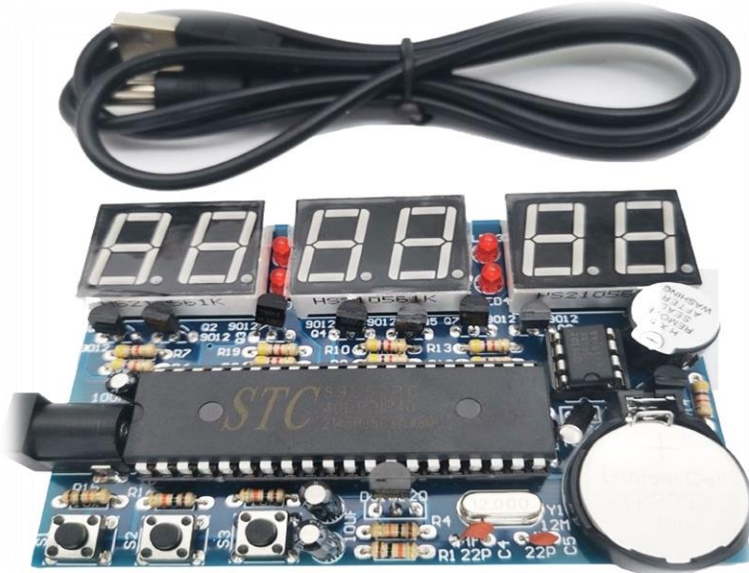
```
#include <avr/io.h> //inlui regs

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>

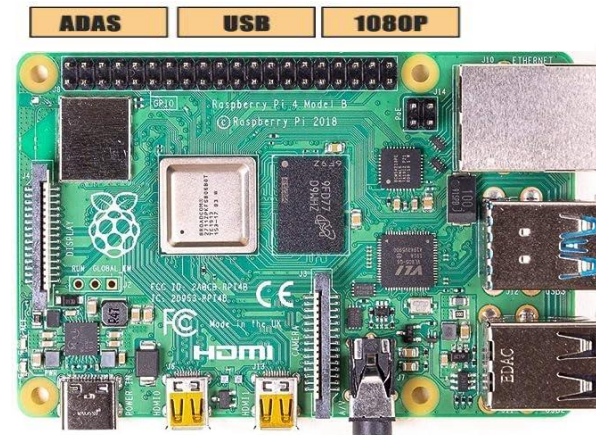
INTRODUÇÃO

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)



Fonte: <https://research.ijcaonline.org/volume55/number10/pacs050754.pdf>



ABSTRAÇÃO

- Foco nas características essenciais de um objeto.
- Classes abstratas e interfaces.
- Modelagem de conceitos genéricos.



A abstração permite criar classes que representam conceitos genéricos, sem se preocupar com os detalhes de sua implementação. As classes abstratas definem um contrato que as classes filhas devem implementar. As interfaces definem um conjunto de métodos que uma classe deve implementar, sem fornecer uma implementação padrão.

• **Imagem:** Um diagrama UML representando uma classe abstrata e uma interface.

INTRODUÇÃO

PROGRAMAÇÃO ORIENTADA A OBJETOS



	Vs CARACTERÍSTICAS	
Estaticamente Tipada	TIPO DE CÓDIGO	Dinamicamente tipada
Relativamente mais rápido que Python	PERFORMANCE	Mais lento que o Java em várias implementações
Convenções complexas e maior tempo de desenvolvimento	CURVA DE APRENDIZAGEM	Fácil de aprender e de usar
Spring, Blade	BIBLIOTECAS /FRAMEWORKS	TensorFlow, Pandas, Django, Flask, Streamlit

Fonte: Elaborado pelo autor

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

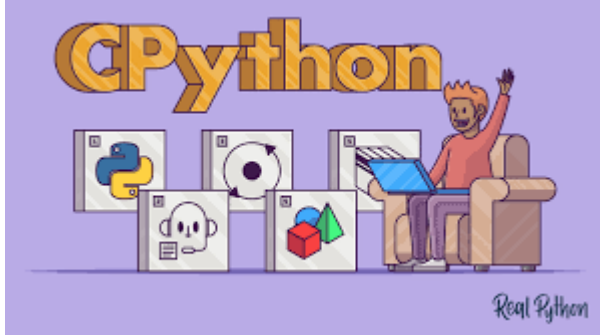
O acrônimo **IDE** (***Integrated Development Environment***) significa software ou ambiente de desenvolvimento integrado que une ferramentas de desenvolvimento em uma única interface gráfica do usuário para escrever e testar códigos escritos em diferentes linguagens de programação.

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++.

INTERPRETADORES PYTHON

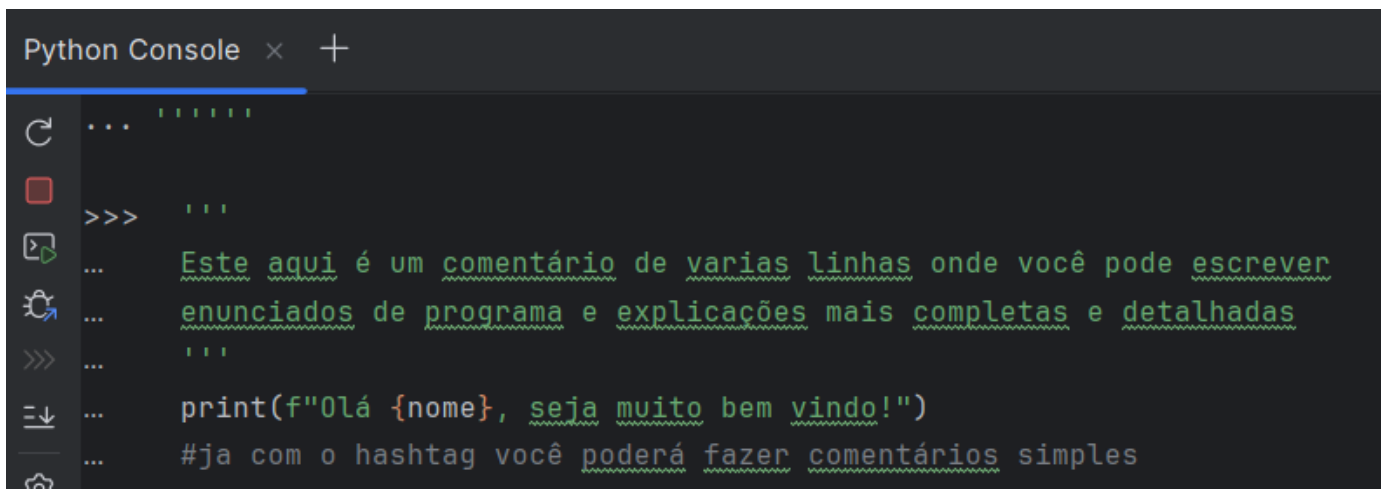


COMENTÁRIOS EM PYTHON

Saber como fazer comentários em uma linguagem de programação é um detalhe muito importante.

Símbolo

#Comentário sem função de execução



```
Python Console x +
... """
>>> '''
... Este aqui é um comentário de varias linhas onde você pode escrever
... enunciados de programa e explicações mais completas e detalhadas
... '''
>>> ...
... print(f"Olá {nome}, seja muito bem vindo!")
... #ja com o hashtag você poderá fazer comentários simples
```

Ao usar # na linguagem PYTHON você fará um comentário de uma linha no código; isto permite ao programador documentar a funcionalidade(s) de cada linha ou inserir observações importantes ao longo do código.

Mas se você precisar fazer um comentário de várias linhas, basta usar ''' OU """ no início e finalizar com ''' OU """.

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 retornar ao código anterior caso sua modificação não funcione!

TIPOS DE DADOS

Saber sobre variável e operadores apenas não basta; você precisa saber sobre tipos de dados para programar!

Tipos de dados em Python

Inteiro (do inglês, **integer**)

int → Tipo para definir números inteiros: **8** / **80**

Decimal ou ponto flutuante (do inglês, **float** point)

float → Tipo para definir números decimais até 7 dígitos: **3.14** / **5.50**

Caracteres/Cadeia de Caracteres (do inglês **String**)

String → Tipo para definir caracteres: **"Exemplo de String"** / **"12"**

Booleano (do inglês, **Bool**)

bool → Tipo para definir valores booleanos: **True** / **False**

OPERADORES ARITMÉTICOS

Operadores aritméticos: são símbolos especiais que realizam as operações aritméticas básicas

Operadores Aritméticos

OPERAÇÃO	SÍMBOLO	CÓDIGO EM PYTHON	DESCRIÇÃO
ADIÇÃO	+	$x + y$	Efetua a soma $x + y$
SUBTRAÇÃO	-	$x - y$	Efetua a subtração $x - y$
MULTIPLICAÇÃO	*	$x * y$	Efetua a multiplicação $x * y$
DIVISÃO	/	x / y	Efetua a divisão x / y
DIVISÃO INTEIRA	//	$x // y$	Efetua a divisão $x // y$
RESTO DA DIVISÃO	%	$x \% y$	Obtém o resto da divisão entre dois inteiros
INCREMENTO	++	+=	Incrementa o valor da variável em 1 unidade
DECREMENTO	--	--	Decrementa o valor da variável em 1 unidade

OPERADORES RELACIONAIS

O símbolo **=** é considerado operador de atribuição; usado para atribuir valores para constantes ou atribuir cálculos. **Exemplo:**

X = 2
Y = 3
soma = X + Y



OPERAÇÃO	SÍMBOLO	CÓDIGO EM PYTHON	DESCRIÇÃO
Menor que	<	x < y	Verdadeiro se x Menor y
Maior que	>	x > y	Verdadeiro se x Maior y
Igual	==	x == y	Verdadeiro se x Igual a y
ATRIBUIÇÃO	= ←	X = 2	Atribui um valor ou resultado de cálculo à uma constante
Diferente	!=	x != y	Verdadeiro se x Diferente de y
Menor igual	<=	x <= y	Verdadeiro se x Menor ou Igual a y
Maior Igual	>=	x >= y	Verdadeiro se x Maior ou Igual a y

UM SISTEMA!



O Conceito de
é um
Sistema!

```
# Solicita ao usuário que insira dois valores
valor1 = float(input("Digite o primeiro valor: "))
valor2 = float(input("Digite o segundo valor: "))

# Calcula a média dos dois valores
media = (valor1 + valor2) / 2

# Exibe o resultado
print(f"A média entre {valor1} e {valor2} é {media:.2f}")
```

Entrada(s)

```
valor1 = float(input("Digite o primeiro valor: "))
valor2 = float(input("Digite o segundo valor: "))
```

Processamento

```
media = (valor1 + valor2) / 2
```

Saída

```
print(f"A média entre {valor1} e  
{valor2} é {media:.2f}")
```

EXEMPLO 01

Algoritmo e Código em Python

Programa para calcular média aritmética entre dois valores

NARRATIVA	FLUXOGRAMA	PSEUDOCÓDIGO
		Início real: valor1, valor2, m
Ler valor1		Ler (valor1)
Ler valor2		Ler (valor2)
Calcular a Média $m = (\text{valor1} + \text{valor2})/2$		$m \leftarrow (\text{valor1} + \text{valor2})/2$
Exibir a média calculada		Escrever (m)
		Fim

EXEMPLO 01

Algoritmo e Código em Python

Programa para calcular média aritmética entre dois valores

Solicita dois valores reais ao usuário

```
valor1 = float(input("Digite o primeiro valor: "))
```

```
valor2 = float(input("Digite o segundo valor: "))
```

Calcula a média aritmética

```
media = (valor1 + valor2) / 2
```

Exibe o resultado

```
print(f"A média aritmética entre {valor1} e {valor2} é {media}.")
```

EXEMPLO 02

Narrativa, Pseudocódigo e fluxograma do processo calcular área triângulo. Definir variáveis reais: base, altura e área; Ler base, Ler altura; Calcular Área = Base*Altura/2; Exibir Área; Fim).

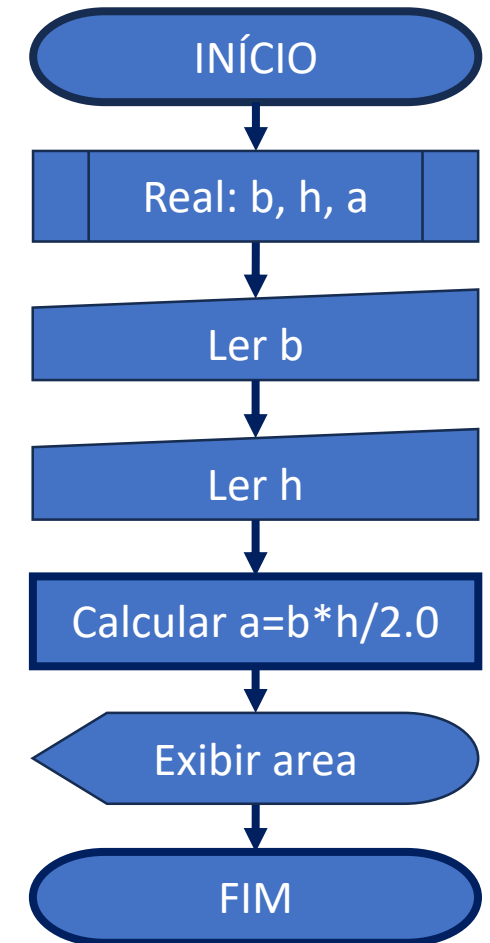
Narrativa:

Ler b, h,
Calcular $a = b * h / 2.0$
Exibir area

Pseudocódigo:

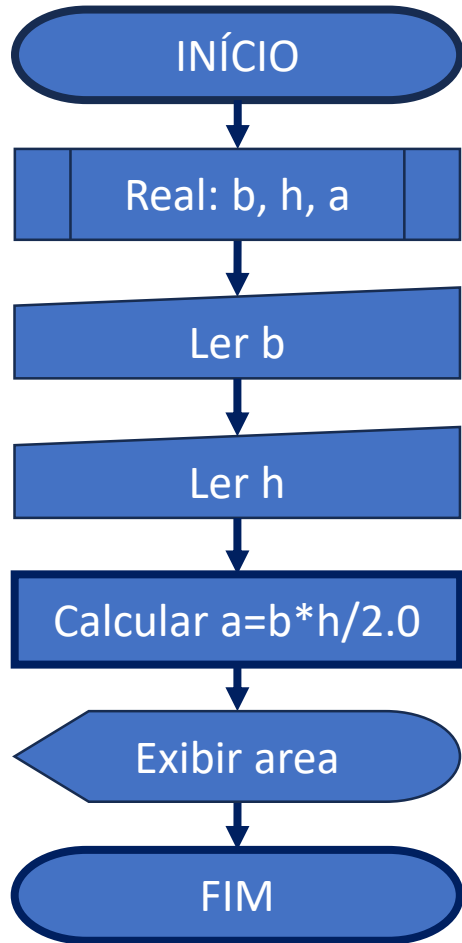
Início:
Real: base, altura, area
Ler base
Ler altura
Calcular $area = base * Altura / 2.0$
Escrever área
Fim

Fluxograma:



EXEMPLO 02

Fluxograma:



Fluxograma e código do programa p/ calcular área triângulo(definir variáveis reais: base, altura e área; Ler base, Ler altura; Calcular Área = Base*Altura/2; Exibir Área; Fim).

Definir variáveis reais: base, altura e área

```
base = float(input("Digite o valor da base do triângulo: "))
```

```
altura = float(input("Digite o valor da altura do triângulo: "))
```

Calcular a área do triângulo

```
area = (base * altura) / 2
```

Exibir a área calculada

```
print(f"A área do triângulo com base {base} e altura {altura} é {area}.")
```

Fim do programa

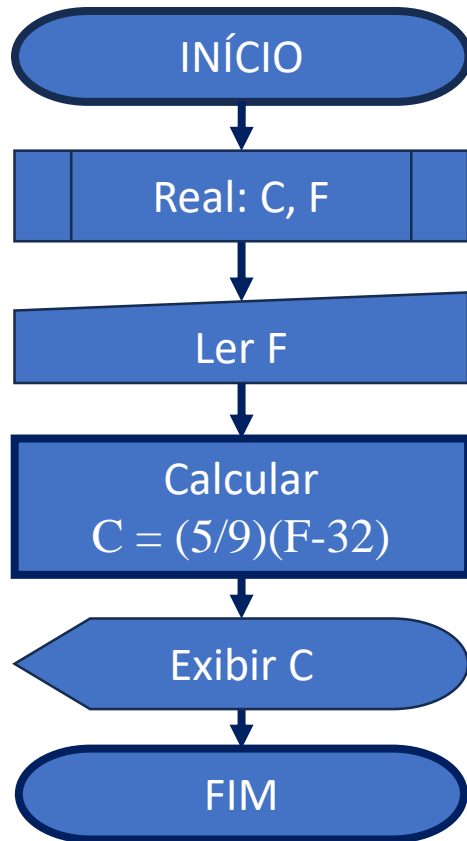
EXERCÍCIOS PARA PRATICAR

- 1) Criar a narrativa, o pseudocódigo, o fluxograma e código 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, o pseudocódigo, o fluxograma e código 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, o pseudocódigo, o fluxograma e código 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, o pseudocódigo, o fluxograma e código 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!".

EX. 01

Criar a narrativa, o pseudocódigo e o fluxograma de um programa que solicite ao usuário digitar uma temperatura em fahrenheit e convertê-la e exibir o resultado em graus Celsius. Onde $C = (5/9)(F-32)$

Fluxograma:



Narrativa:

Ler Temp. Fahrenheit
Calcular $C = (5/9)(F-32)$
Exibir C

Pseudocódigo:

Início:

Real: C, F

Ler F

Calcular $C = (5/9)(F-32)$

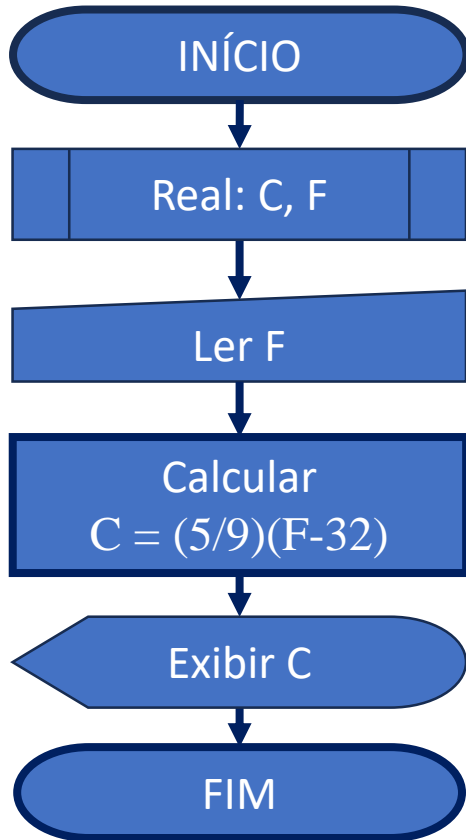
Exibir C

FIM

EX. 01

Criar a narrativa, o pseudocódigo e o fluxograma de um programa que solicite ao usuário digitar uma temperatura em fahrenheit e convertê-la e exibir o resultado em graus Celsius. Onde $C = (5/9)(F-32)$

Fluxograma:



```
# Solicita ao usuário que insira uma temperatura em Fahrenheit
fahrenheit = float(input("Digite a temperatura em Fahrenheit: "))

# Converte a temperatura para Celsius usando a fórmula C = (5/9) * (F - 32)
celsius = (5/9) * (fahrenheit - 32)

# Exibe o resultado
print(f"A temperatura em Celsius é {celsius:.2f}°C")
```

EX. 02

Criar a narrativa, pseudocódigo, fluxograma e código de um programa que solicite ao usuário digitar medida em polegadas e converter o valor dado para milímetros. (1"=25,4mm).

Narrativa:

Ler P

Calcular $\text{mm} = 25.4 * P$

Exibir mm

Pseudocódigo:

Início:

Real: P, mm

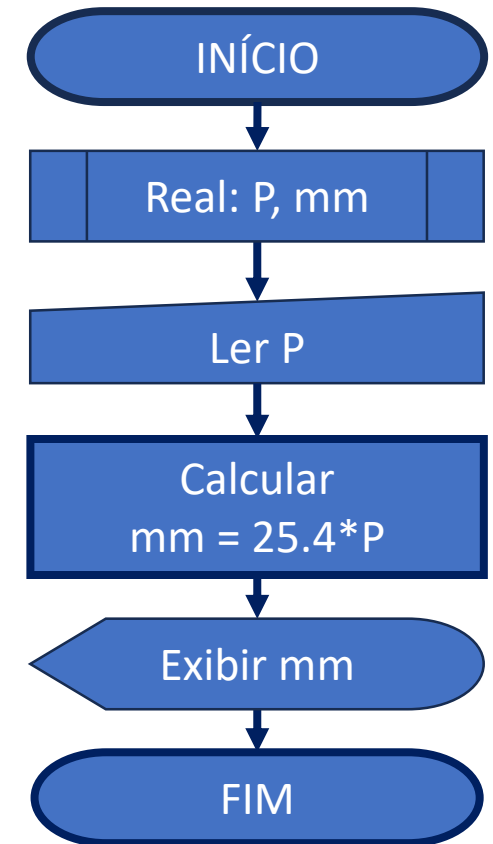
Ler P

Calcular $\text{mm} = 25.4 * P$

Exibir mm

FIM

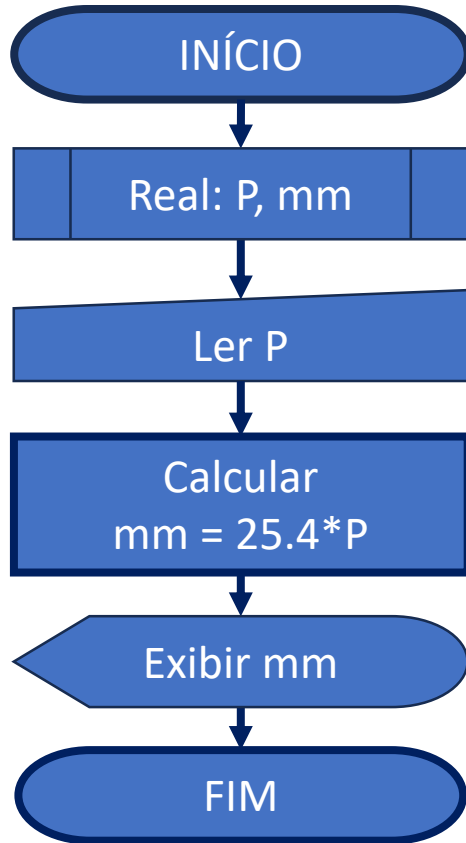
Fluxograma:



EX. 02

Criar a narrativa, pseudocódigo, fluxograma e código de um programa que solicite ao usuário digitar medida em polegadas e converter o valor dado para milímetros. (1"=25,4mm).

Fluxograma:



```
# Solicita ao usuário que insira uma medida em polegadas
polegadas = float(input("Digite a medida em polegadas: "))

# Converte a medida para milímetros (1 polegada = 25,4 mm)
milímetros = polegadas * 25.4

# Exibe o resultado
print(f"{polegadas} polegadas é igual a {milímetros:.2f} milímetros.")
```

EX. 03

Narrativa, pseudocódigo fluxograma e código de um programa que solicite ao usuário digitar uma medida em milímetros, elaborar um programa para converter o valor dado para polegadas.

Narrativa:

Ler mm

Calcular $P = \text{mm} / 25.4$

Exibir P

Pseudocódigo:

Início:

Real: P, mm

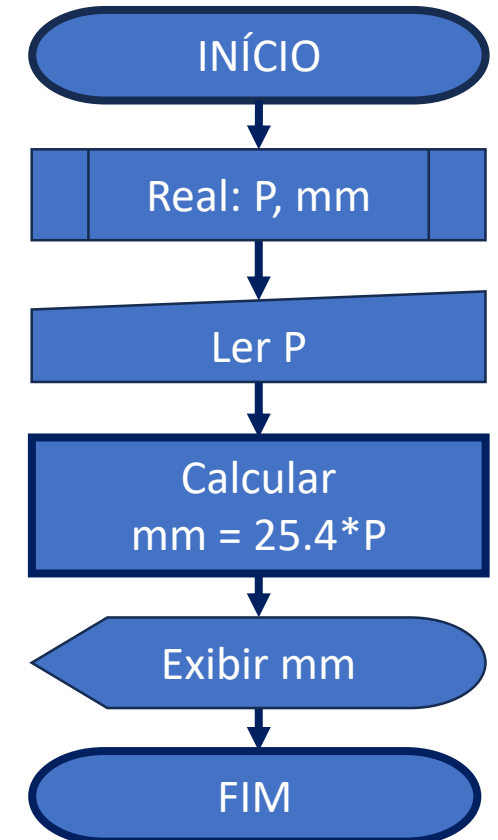
Ler mm

Calcular $P = \text{mm} / 25.4$

Exibir P

FIM

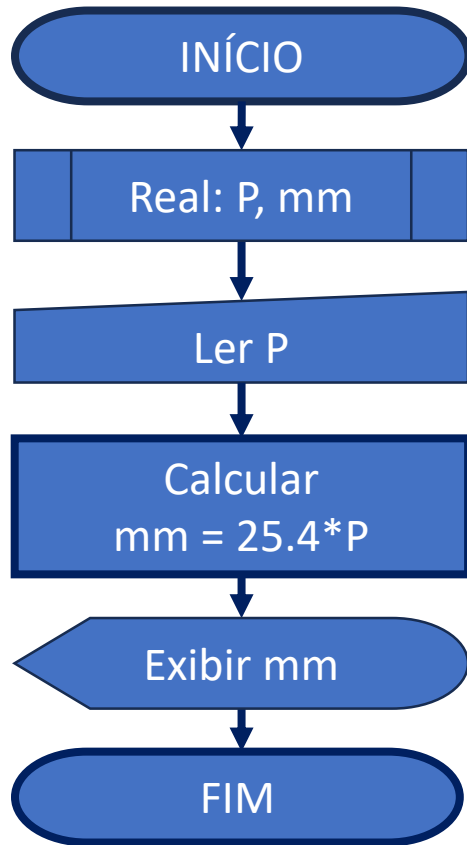
Fluxograma:



EX. 03

Narrativa, pseudocódigo fluxograma e código de um programa que solicite ao usuário digitar uma medida em milímetros, elaborar um programa para converter o valor dado para polegadas.

Fluxograma:



```
# Solicita ao usuário que insira uma medida em milímetros
milímetros = float(input("Digite a medida em milímetros: "))

# Converte a medida para polegadas (1 polegada = 25,4 mm)
polegadas = milímetros / 25.4

# Exibe o resultado
print(f"{milímetros} milímetros é igual a {polegadas:.2f} polegadas.")
```

EX. 04

Narrativa, pseudocódigo, fluxograma e código 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 “Pode votar”. No caso contrário, ou seja, se menor que 16 anos ou maior que 70 anos “Não pode votar!”.

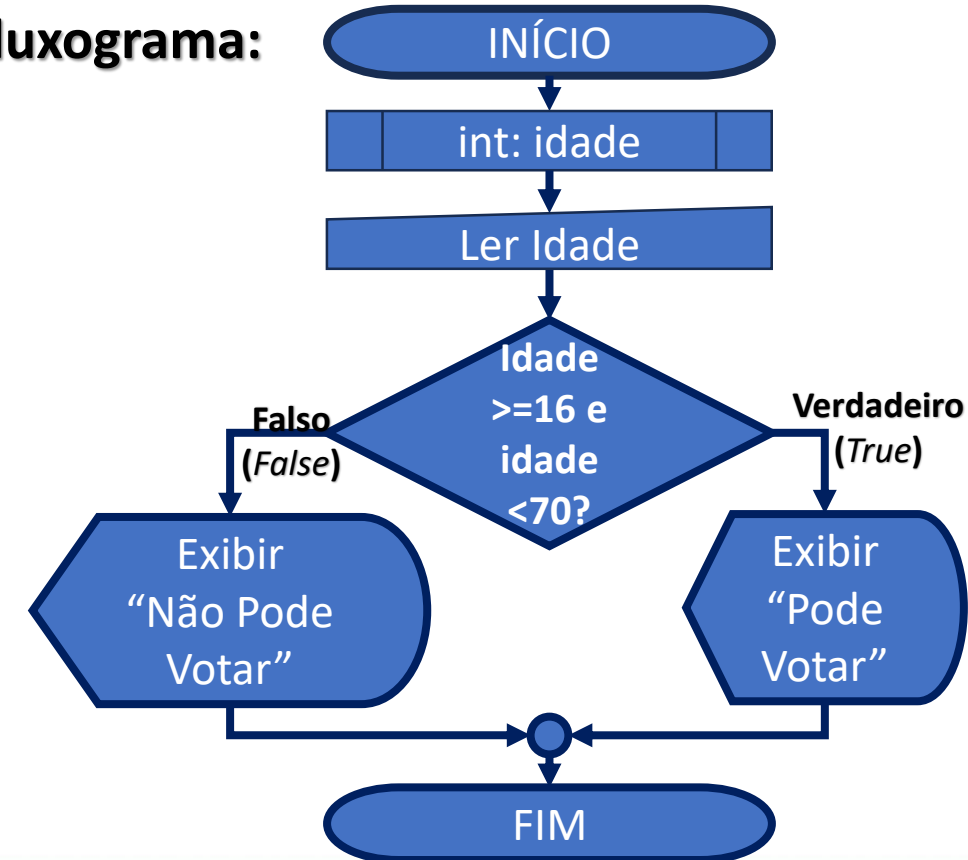
Narrativa:

Ler idade
se idade ≥ 16 e idade < 70 :
Exibir “Pode votar!”
senão:
Exibir “Não pode votar!”

Pseudocódigo:

Início:
inteiro: idade
Ler idade
se idade ≥ 16 e idade < 70 então:
Exibir “Pode votar!”
senão:
Exibir “Não pode votar!”
FIM

Fluxograma:



EX. 04

Narrativa, pseudocódigo, fluxograma e código 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,

```
# Solicita ao usuário que insira sua idade
idade = int(input("Digite a sua idade: "))

# Verifica se a idade é maior ou igual a 16 anos e menor que 70 anos
if 16 <= idade < 70:
    print("Você está na faixa etária entre 16 e 70 anos.")
else:
    print("Você está fora da faixa etária entre 16 e 70 anos.")
```

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.

[illegible]