



**Prof. Me. Massaki Igarashi**

[massaki.igarashi@anchieta.br](mailto:massaki.igarashi@anchieta.br)

**Algoritmos e Ling. de Programação**

# Cálculo da Média

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

# 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**  $mm = 25.4 * P$

**Exibir** mm

## Pseudocódigo:

**Início:**

Real: P, mm

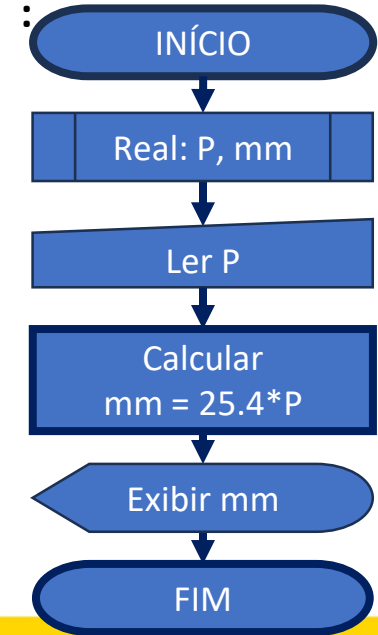
**Ler P**

Calcular  $mm = 25.4 * P$

Exibir mm

**FIM**

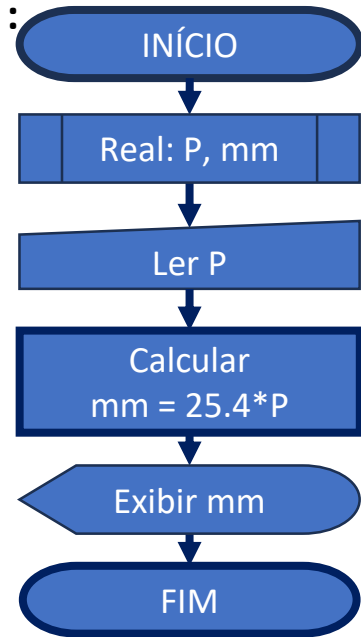
## Fluxograma



## 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)
milimetros = polegadas * 25.4

# Exibe o resultado
print(f"{polegadas} polegadas é igual a {milimetros:.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

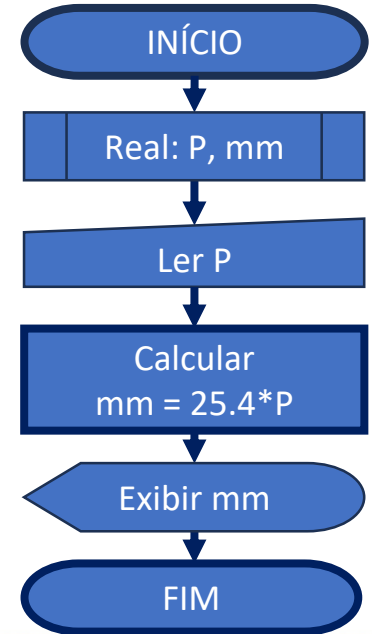
**Calcular**  $P = \text{mm} / 25.4$

Exibir P

**FIM**

## Fluxograma

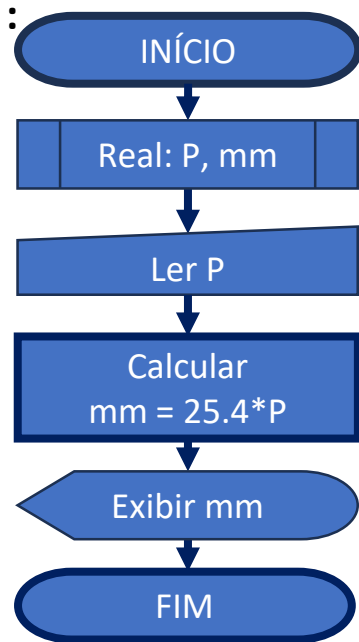
:



## 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  $\geq 16$  e idade  $< 70$ :

Exibir “Pode votar!”

**senão:**

Exibir “Não pode votar!”

**Pseudocódigo:**

**Início:**

inteiro: idade

**Ler idade**

**se** idade  $\geq 16$  e idade  $< 70$  **então:**

Exibir “Pode votar!”

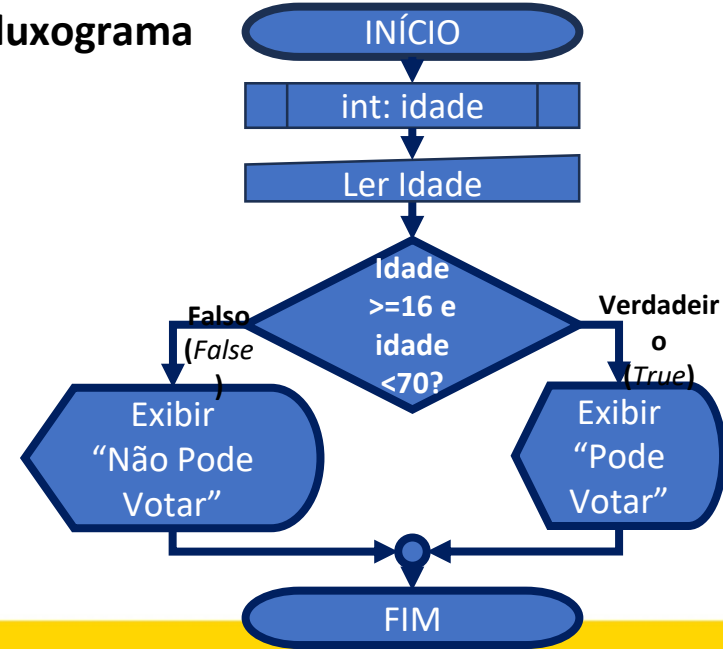
**senão:**

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.")
```



# EXERCÍCIOS P/ PRATICAR

- 5) Criar a narrativa, pseudocódigo, fluxograma e código para um programa que receba o raio ( $r$ ) de uma esfera, e em seguida calcule e exibir o seu volume ( $v$ ) desta esfera.
- 6) Criar a narrativa, pseudocódigo, fluxograma e código para um programa que receba um ângulo ( $g$ ) em graus, e converta este valor para radianos ( $r$ ).
- 7) Criar a narrativa, pseudocódigo, fluxograma e código para um programa que receba o valor de um ângulo em radianos, e converta este valor para graus.
- 8) Criar a narrativa, pseudocódigo, fluxograma e código para um programa que receba ângulo em radianos, e converta este valor para graus. Lembrar que  $400$  graus equivalem a  $2 \cdot \pi$  radianos.

# OPERADORES LÓGICOS

Já vimos os operadores **Aritméticos** e também os **Relacionais**.

Hoje, como tratamos das estruturas de condição, mais especificamente os testes de condição (ou Estrutura de Decisão), necessitaremos, dos **operadores lógicos** na verificação da condição para execução:

OPERADOR	SÍMBOLO	CÓDIGO EM PYTHON	CÓDIGO EM JAVA	DESCRIÇÃO
E	&&	exp1 <b>and</b> exp2	exp1 <b>&amp;&amp;</b> exp2	Verdadeiro tanto exp1 quanto exp2 forem verdadeiras
OU	OR	exp1 <b>or</b> exp2	exp1 <b>  </b> exp2	Verdadeiro se exp1 verdadeira OU exp2 verdadeira
Não	NOT	<b>not</b> exp1	<b>!=</b> exp1	Verdadeiro se exp1 FALSA

# PARA PRATICAR EM CASA

5) Criar a narrativa, pseudocódigo, fluxograma e código um programa que receba o raio ( $r$ ) de uma esfera, e em seguida calcule e exibir o seu volume ( $v$ ) desta esfera.

## Narrativa:

Atribuir PI

=3.14159265358979323846

Ler  $r$

Calcular  $v = (4/3.0)*PI/(r*r*r)$

Exibir  $r$

## Pseudocódigo:

Início:

Real:  $r, v$

Real PI = 3.14159265358979323846

Ler  $r$

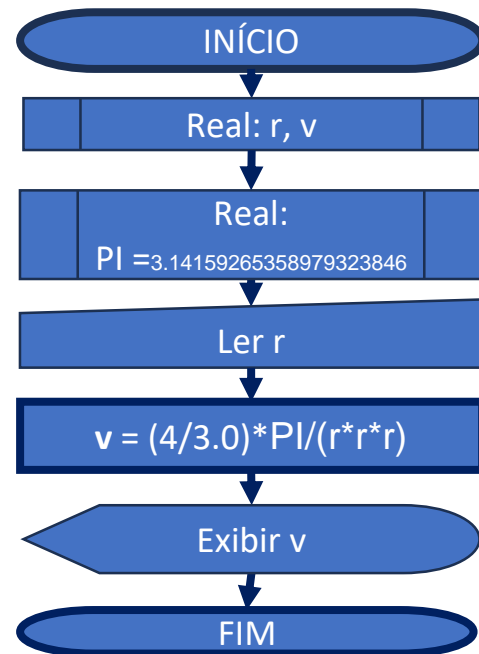
Calcular  $v = (4/3.0)*PI/(r*r*r)$

Exibir  $v$

FIM

## Fluxograma

:



# PARA PRATICAR EM CASA

6) Criar a narrativa, o pseudocódigo e o fluxograma para um programa que receba um ângulo ( $g$ ) em graus, e converta este valor para radianos ( $r$ ).

```
import math

# Solicita ao usuário que insira o raio da esfera
raio = float(input("Digite o raio da esfera em unidades de comprimento: "))

# Calcula o volume da esfera usando a fórmula  $V = (4/3) * \pi * r^3$ 
volume = (4/3) * math.pi * (raio ** 3)

# Exibe o resultado
print(f"O volume da esfera com raio {raio} é {volume:.2f} unidades cúbicas.")
```

# PARA PRATICAR EM CASA

6) Criar a narrativa, o pseudocódigo e o fluxograma para um programa que receba um ângulo ( $g$ ) em graus, e converta este valor para radianos ( $r$ ).

## Narrativa:

Atribuir PI

=3.14159265358979323846

Ler  $g$

Calcular  $r = g * PI / 180.0$

Exibir  $r$

## Pseudocódigo:

Início:

Real:  $g, r$

Real PI = 3.14159265358979323846

Ler  $g$

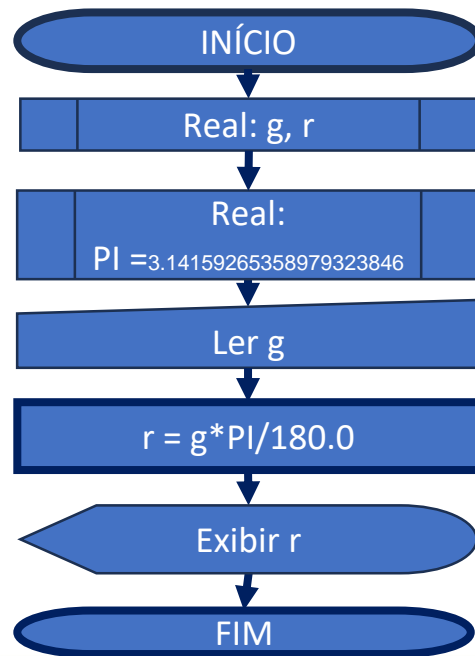
Calcular  $r = g * PI / 180.0$

Exibir  $r$

FIM

## Fluxograma

:



# PARA PRATICAR EM CASA

6) Criar a narrativa, o pseudocódigo e o fluxograma para um programa que receba um ângulo ( $g$ ) em graus, e converta este valor para radianos ( $r$ ).

```
import math

# Solicita ao usuário que insira o ângulo em graus
graus = float(input("Digite o ângulo em graus: "))

# Converte o ângulo para radianos usando a fórmula  $r = g * (\pi / 180)$ 
radianos = graus * (math.pi / 180)

# Exibe o resultado
print(f"O ângulo de {graus} graus é igual a {radianos:.4f} radianos.")
```

# PARA PRATICAR EM CASA

7) Criar a narrativa, pseudocódigo, fluxograma e código para um programa que receba o valor de um ângulo em radianos, e converta este valor para graus.

## Narrativa:

Atribuir PI

=3.14159265358979323846

Ler r

Calcular  $g = r * 180.0 / PI$

Exibir g

## Pseudocódigo:

Início:

Real: g, r

Real PI = 3.14159265358979323846

Ler r

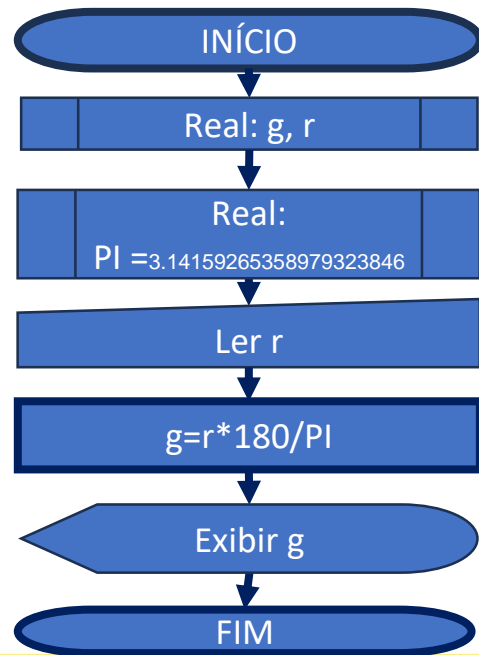
Calcular  $g = r * 180.0 / PI$

Exibir g

FIM

## Fluxograma

:



# PARA PRATICAR EM CASA

7) Criar a narrativa, pseudocódigo, fluxograma e código para um programa que receba o valor de um ângulo em radianos, e converta este valor para graus.

```
import math

# Solicita ao usuário que insira o ângulo em radianos
radianos = float(input("Digite o ângulo em radianos: "))

# Converte o ângulo para graus usando a fórmula  $g = r * (180 / \pi)$ 
graus = radianos * (180 / math.pi)

# Exibe o resultado
print(f"O ângulo de {radianos:.4f} radianos é igual a {graus:.2f} graus.")
```



# PARA PRATICAR EM CASA

8) Criar a narrativa, pseudocódigo, fluxograma e código para um programa que receba ângulo em radianos, e converta este valor para graus. Lembrar que 400 graus equivalem a 27.

## Narrativa:

Atribuir PI

=3.14159265358979323846

Ler Rad

Calcular  $Gr = 400 * Rad / (2 * PI)$

Exibir Gr

## Pseudocódigo:

Início:

Real: Rad, Gr

Real PI = 3.14159265358979323846

Ler Rad

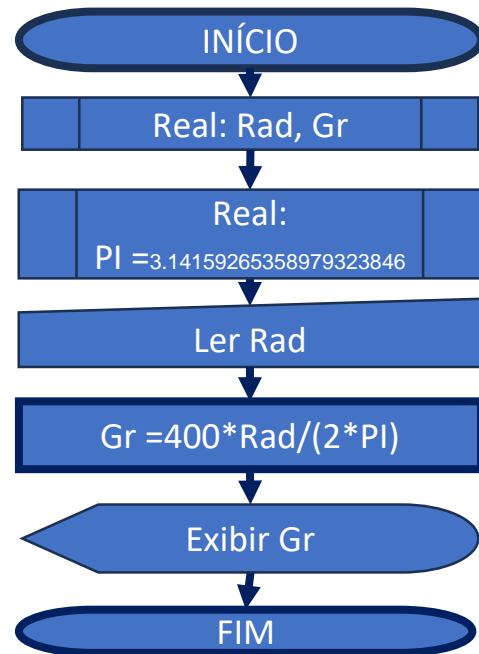
Calcular  $Gr = 400 * Rad / (2 * PI)$

Exibir Gr

FIM

## Fluxograma

:



# PARA PRATICAR EM CASA

8) Criar a narrativa, pseudocódigo, fluxograma e código para um programa que receba ângulo em radianos, e converta este valor para graus. Lembrar que 400 graus equivalem a 27.

```
import math

# Solicita ao usuário que insira o ângulo em radianos
radianos = float(input("Digite o ângulo em radianos: "))

# Converte o ângulo para grados usando a fórmula  $g = r * (200 / \pi)$ 
grados = radianos * (200 / math.pi)

# Exibe o resultado
print(f"O ângulo de {radianos:.4f} radianos é igual a {grados:.2f} grados.")
```

# Identificadores: VARIÁVEIS

Nas últimas aulas aprendemos sobre algoritmo e suas formas de representação, inclusive sobre a **função do algoritmos de representar uma solução para um problema** no mundo real escrito em linguagem que o computador compreende. Com esse mesmo pensamento, podemos entender que cada item tratado nesse problema pode ser representado por uma **variável**.

# Identificadores: VIARIÁVEIS

Pode-se dizer que **você certamente irá lidar com variáveis na maior parte do seu tempo como profissional de programação**, mas as variáveis não costumam vir sozinhas; elas costumam aparecer com **operadores aritméticos e relacionais**, principalmente quando o programa tem a função de processar algum cálculo específico ou algum sinal de entrada para tomar uma determinada decisão.

# Identificadores: VIARIÁVEIS



Em programação, um **identificador** é um nome atribuído a uma **variável** de memória, uma estrutura de dados, uma classe, um objeto, um procedimento, uma função, um comando ou palavra reservada da linguagem.

Variável é tudo que está **sujeito a variações**, que é **incerto, instável ou inconstante**. Quando se fala de **computadores**, é preciso ter em mente que o volume de dados a serem tratados é grande e diversificado. Dessa forma, os **dados a serem processados são bastante variáveis**.

**Todo dado a ser armazenado na memória de um computador deve ser previamente identificado segundo seu tipo**, ou seja, primeiramente é necessário saber o tipo do dado para depois fazer seu armazenamento adequado. **Armazenado o dado, ele pode ser utilizado e processado a qualquer momento**.

## Regras para identificadores:

Um identificador pode ter até 32 caracteres de comprimento.

- ❖ O 1º caractere deve ser uma letra do alfabeto.
- ❖ Os demais caracteres podem ser letras, números ou sinal *underscore* (`_`)
- ❖ Não usar sinais de pontuação, caracteres acentuados, cedilha (ç) ou espaço em branco.
- ❖ Exemplos de identificadores: **Nome, Nome\_Cliente, Nome\_Completo, Data, Data\_Emissao, Mensagem, Aviso, 1aSemana, Endereco, a, b, c, ..., X, Y, Z.**

# Identificadores: VIARIÁVEIS

Uma variável representa um contêiner ou espaço na memória física ou virtual de um computador, onde diferentes tipos de dados (valores) são armazenados durante a execução de um programa. A cada variável é atribuído um nome descritivo ou identificador que se refere ao valor salvo.

Elas são importantes para o funcionamento de programas e aplicações que lidam com cálculos, condições, repetições e qualquer outro dado mutável durante o seu funcionamento.

# CONTROLE DE FLUXO

Os comandos de controle de fluxo são essenciais a qualquer linguagem de programação; já que eles determinam a ordem dos comandos e especificam, numa dada condição, quais destes devem ser executados (comandos de **teste de condição**) e se devem ser repetidos (comandos de **controle de loop**).

Aula de  
Hoje!

1. **Testes de condição (ou Estrutura de Decisão):** popularmente conhecida como estrutura condicional ou de seleção, realiza diferentes ações dependendo se a condição for verdadeira ou falsa. A condição é uma expressão processada em um valor booleano.

Próx.  
Aula

2. **Controle de loop (ou Estrutura de Repetição):** realiza e repete uma ou mais ações dependendo da condição (se verdadeira ou falsa). Condição esta que é processada em um valor booleano como no teste de condição.

# TESTE CONDICIONAL

1. **Testes de condição (ou Estrutura de Decisão):** São importantes na programação pois **reproduzem a ação humana da tomada de decisão** em função de condições lógicas explícitas. Elas podem conter um único comando ou blocos de

Na linguagens de programações a estrutura condicional é popularmente conhecida por usar os termos em inglês: **if (SE)...****else (SENÃO);** **else if (SENÃO SE)** ou **elif (SENÃO SE** na linguagem python).

A estrutura condicional avalia uma condição específica e **divide a lógica do programa em dois caminhos:** um que será seguido **quando a condição for verdadeira** e **outro** que será **seguido quando a condição for falsa**.

Veja a sintaxe a seguir:



# SE(If)...ENTÃO

Realiza diferentes ações dependendo se a condição for verdadeira ou falsa.

A condição é uma expressão processada em um valor booleano.

## Uma única linha de instrução:

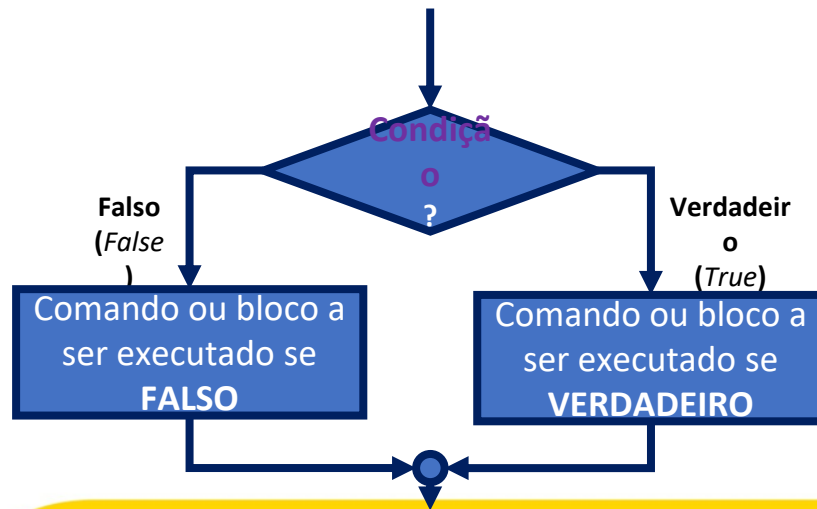
Se (expressão lógica) então:  
Instrução

## Bloco de instruções:

Se (expressão lógica) então:  
Instrução 1  
Instrução 2  
.  
.  
.  
Instrução n

FIM

## Fluxograma da Estrutura Condicional:



# SE(If)...SENÃO (Else)

Uma variação da estrutura apresentada anteriormente é **SE/ENTÃO/SENÃO**; em que temos uma cláusula que permite a execução de outra instrução ou bloco de instrução no caso de a avaliação for falsa.

## Uma única linha de instrução:

**Se** (expressão lógica) **então**:  
Instrução\_Verdadeira

**Senão**:  
Instrução\_Falsa

## Bloco de instruções:

**Se** (expressão lógica) **então**:  
Instrução\_Verdadeira1  
....  
Instrução\_VerdadeiraN

**Senão**:  
Instrução\_Verdadeira1  
....  
Instrução\_VerdadeiraN

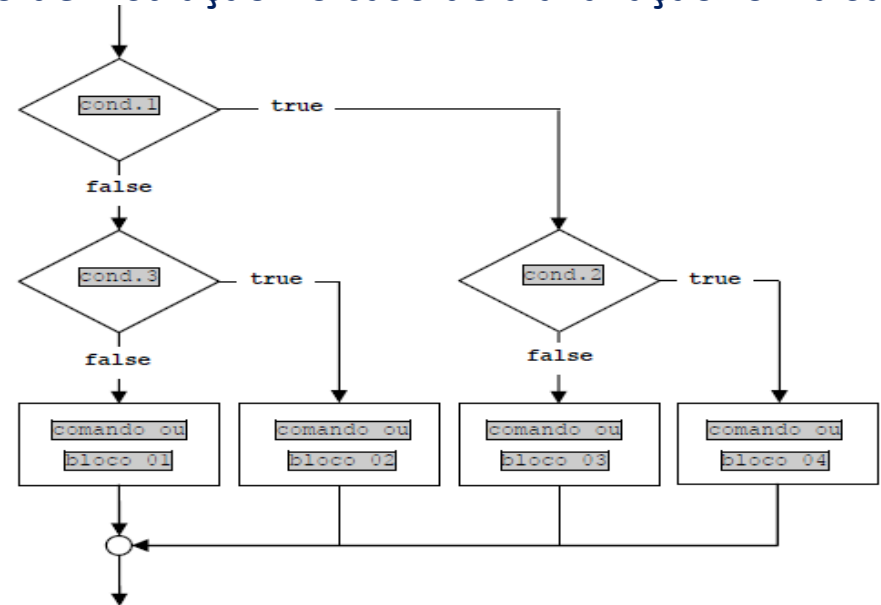


FIGURA 6-6: FLUXOGRAMA COM ESTRUTURAS CONDICIONAIS ANINHADAS.

# EXEMPLO 03

A seguir, a narrativa, o pseudocódigo, fluxograma e código e um programa para ler um número digitado pelo usuário no seu teclado e verificar e informar se o número digitado é par ou ímpar.

## Narrativa:

Início:

Ler num:

se  $\text{num} \% 2 == 0$ :  
Exibir "Par".

senão:  
Exibir "Ímpar".

Fim

## Pseudocódigo:

Início:

inteiro: num

Ler num

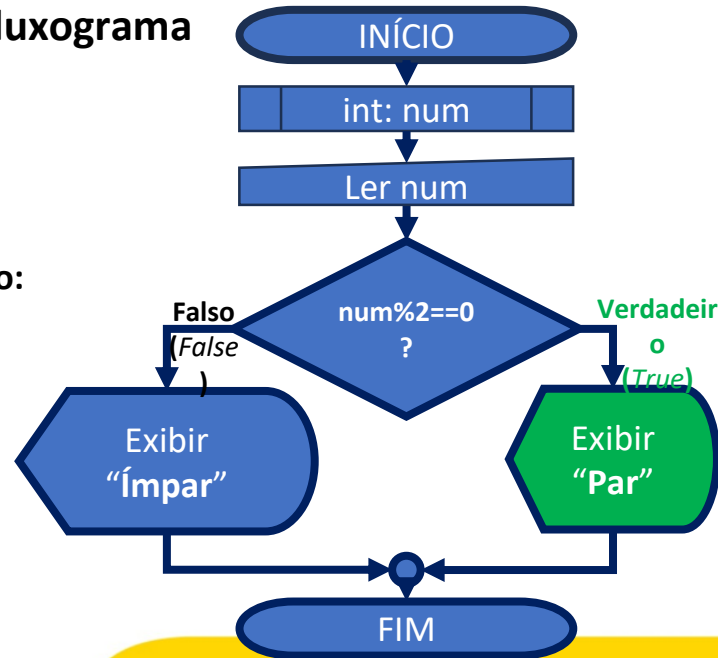
se  $\text{num} \% 2 == 0$  então:  
Exibir "Par"

senão:  
Exibir "Ímpar"

FIM

## Fluxograma

:



# EXEMPLO 03

A seguir, a narrativa, o pseudocódigo, fluxograma e código de um programa para ler um número digitado pelo usuário no seu teclado e verificar e informar se o número digitado é par ou ímpar.

```
# Solicita ao usuário que insira um número
numero = int(input("Digite um número inteiro: "))

# Verifica se o número é par ou ímpar
if numero % 2 == 0:
    print(f"O número {numero} é par.")
else:
    print(f"O número {numero} é ímpar.")
```

# EXEMPLO 04

Pseudocódigo, fluxograma e código de um programa para ler um número digitado pelo usuário no seu teclado e verificar e calcular raiz quadrada para valores Maiores ou iguais a zero; caso exibir a mensagem “Não existe raiz real para este número!”.

## Pseudocódigo:

### Início:

Real: num

Ler num

**se** num  $\geq 0$  **então:**

    Calcular Raiz Quadrada

    Exibir Raiz Quadrada

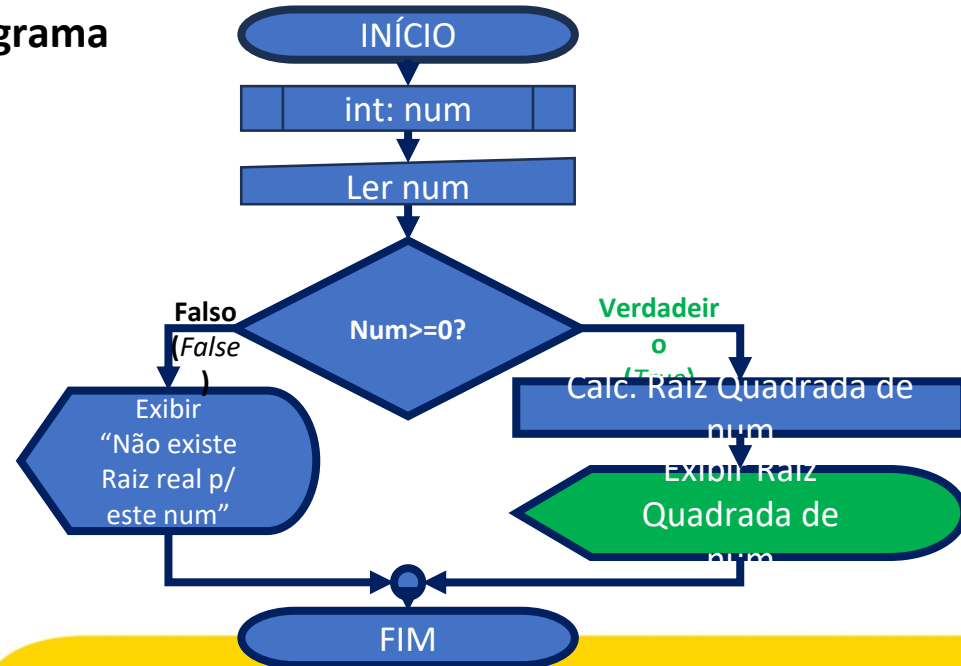
**senão:**

    Exibir “Não existe Raiz real p/ este num”

**FIM**

## Fluxograma

:



# EXEMPLO 04

Pseudocódigo, fluxograma e código de um programa para ler um número digitado pelo usuário no seu teclado e verificar e calcular raiz quadrada para valores Maiores ou iguais a zero; caso exibir a mensagem “Não existe raiz real para este número!”.

```
import math

# Solicita ao usuário que insira um número
numero = float(input("Digite um número: "))

# Verifica se o número é maior ou igual a zero
if numero >= 0:
    raiz_quadrada = math.sqrt(numero)
    print(f"A raiz quadrada de {numero} é {raiz_quadrada:.2f}.")
else:
    print("Não existe raiz real para este número!")
```

# EX.05

Medidas Aproximadas :

Camiseta manga curta			
	Comp	Larg	
PP	66 cm	50 cm	
P	70 cm	54 cm	
M	72 cm	56 cm	
G	74 cm	58 cm	
GG	76 cm	60 cm	
XGG	79 cm	63 cm	

\*as medidas podem variar em até 2cm

Elaborar o Pseudocódigo, fluxograma e código de um programa para exibir o tamanho da camiseta manga curta, “PP”, “P”, “M”, “G”, “GG” ou “XGG”, ao solicitar ao usuário que ele insira os valores de comprimento e Largura.

# EX.05

A seguir o **código** de um programa para exibir o tamanho da camiseta ao solicitar ao usuário que ele insira os valores de comprimento e

```
# Solicita ao usuário que insira o comprimento e a largura da camiseta
comprimento = float(input("Digite o comprimento da camiseta em centímetros: "))
largura = float(input("Digite a largura da camiseta em centímetros: "))

# Determina o tamanho da camiseta com base no comprimento e largura
if comprimento <= 60 and largura <= 45:
    tamanho = "pp"
elif comprimento <= 65 and largura <= 48:
    tamanho = "p"
elif comprimento <= 70 and largura <= 51:
    tamanho = "M"
elif comprimento <= 75 and largura <= 54:
    tamanho = "G"
elif comprimento <= 80 and largura <= 57:
    tamanho = "GG"
else:
    tamanho = "XGG"

# Exibe o tamanho da camiseta
print(f"O tamanho da camiseta é {tamanho}.")
```



# EXERCÍCIOS EXTRAS

01) Elaborar um programa em python que solicite ao usuário um valor inteiro e em seguida informe ao usuário se o valor é múltiplo de 3 ou não. A seguir o pseudocódigo referente ao algoritmo do programa:

```
início
    inteiro: x, r
    Ler x
    r = resto da divisao de x por 3
    se (r==0) então:
        escrever "Múltiplo"
    senão:
        escrever "Não múltiplo"
    fim se
fim
```

# EXERCÍCIOS EXTRAS

02) Dadas as notas a e b de um aluno, com pesos respectivamente 2 e 3; pede-se elaborar um programa em Python para calcular e exibir a média ponderada do aluno, verificar e informar, junto ao valor da sua média, se foi aprovado ou não (OBS: Considere aprovado se obtiver média igual ou superior a 5).

03) Dado o algoritmo a seguir, elabore o código python do programa em questão:

Início

Leia os três valores: a, b e c.

Compare os valores:

Se a é menor ou igual a b e a é menor ou igual a c, então

a é o menor valor.

Caso contrário, se b é menor ou igual a a e b é menor ou igual a c, então

b é o menor valor.

Caso contrário,

c é o menor valor.

Exiba o menor valor.

Fim

# CONTROLE DE FLUXO

Os comandos de controle de fluxo são essenciais a qualquer linguagem de programação; já que eles determinam a ordem dos comandos e especificam, numa dada condição, quais destes devem ser executados (comandos **de teste de condição**) e se devem ser repetidos (comandos de **controle de loop**).

Aula de  
LP06

1. **Testes de condição (ou Estrutura de Decisão)**: popularmente conhecida como estrutura condicional ou de seleção, realiza diferentes ações dependendo se a condição for verdadeira ou falsa. A condição é uma expressão processada em um valor booleano.

Próx.  
Aula

2. **Controle de loop (ou Estrutura de Repetição)**: realiza e repete uma ou mais ações dependendo da condição (se verdadeira ou falsa). Condição esta que é processada em um valor booleano como no teste de condição.

# LOOP FOR

## ESTRUTURA DE REPETIÇÃO FOR

**2. Controle de loop (ou Estrutura de Repetição):** realiza e repete uma ou mais ações dependendo da condição (se verdadeira ou falsa). Condição esta que é processada em uma valor booleano como no teste de condição.

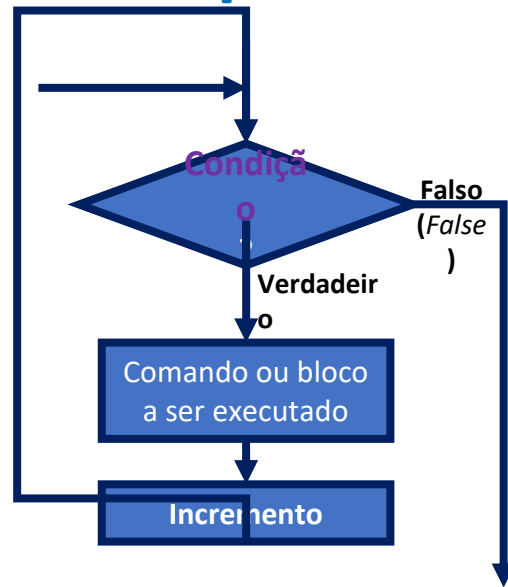
Necessita de uma **variável de controle** que define o **número de execuções** do laço logico.



### Atenção!

A estrutura de repetição **for** realiza essencialmente uma **CONTAGEM NUMÉRICA** (por isso necessita um valor inicial, um valor final da contagem e o passo)

## Fluxograma do Loop FOR:



# LOOP FOR

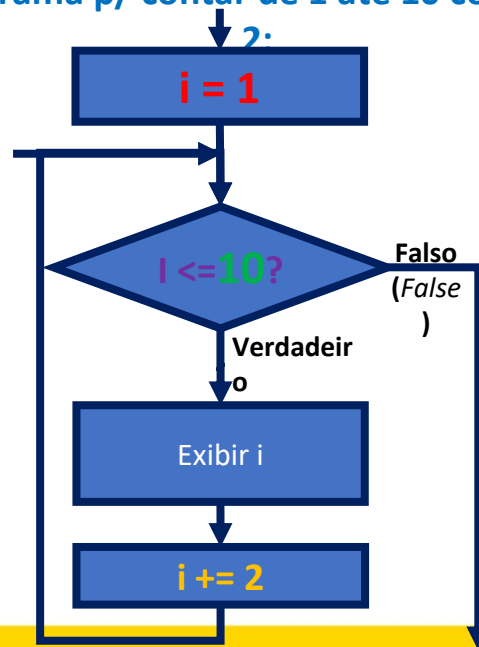
## ESTRUTURA DE REPETIÇÃO FOR

Quando você souber exatamente quantas vezes deseja percorrer um bloco de código, use o loop for em vez de um loop while. Neste exemplo a variável de controle i definirá o número de execuções do loop.

```
for variável (Valor_Inicial, Valor_Final, Passo)
{
    Instrução ou Bloco de instruções;
}
```

Exemplo de código:

Fluxograma p/ contar de 1 até 10 com passo



# EXEMPLO 06

Elaborar um programa para tabelar todos os valores dos ângulos de 0º a 360º com intervalo de 5º e seus respectivos senos.

## Narrativa:

Início:

Inteiro: i

Atribuir 0 a variável i

i <= 0?

Se verdadeiro então:

Converter i para radiano

Exibir seno de i (radiano)

Incrementar i de 5 unidades

Senão

Sair do Loop

Fim

## Código em Python:

```
from math import pi, sin
def Gaus2Rad(Vg):
    return Vg*pi/180

for i in range(0, 360, 5):
    Vrad = Gaus2Rad(i)
    print(sin(Vrad))
```

## Pseudocódigo:

Início:

inteiro: i

i=0

i<=360?

se verdadeiro então:

Exibir seno (i em radiano)

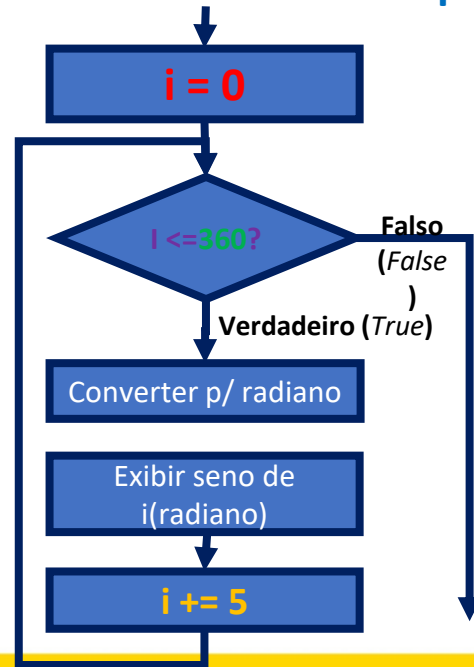
i+=5

senão:

Sair do Loop

FIM

## Fluxograma p/ contar de 0 até 360 com passo 5:



# EXEMPLO 06

Elaborar um programa para tabelar todos os valores dos ângulos de 0º a 360º com intervalo de 5º e seus respectivos senos.

## Pseudocódigo:

Início:

inteiro: i

i=0

i<=360?

se verdadeiro então:

Exibir seno (i em radiano)

i+=5

senão:

Sair do Loop

FIM

## Código

```
import math

# Cabeçalho da tabela
print(f'Ângulo (graus):<15} {'Seno':<10}')

# Gera os valores dos ângulos de 0º a 360º com intervalos de 5º
for angulo in range(0, 361, 5):
    # Converte o ângulo de graus para radianos
    radianos = math.radians(angulo)
    # Calcula o seno do ângulo
    seno = math.sin(radianos)
    # Exibe o ângulo e seu seno correspondente
    print(f'{angulo:<15} {seno:<10.4f}')
```

# EXEMPLO 07

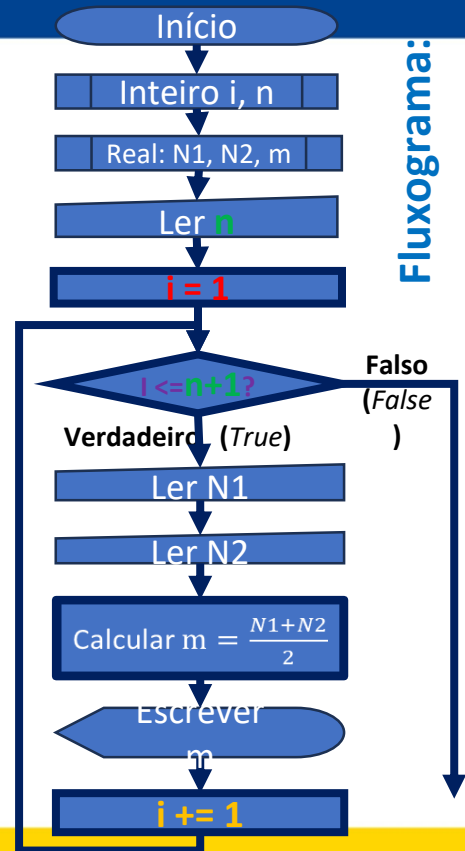
Calcular e exibir a **média de n alunos**. Considerar que cada aluno possui duas **notas (N1 e N2) de mesmo peso**; e que a **quantidade de alunos (n)**, assim como as **notas de cada aluno** devem ser **fornecidas pelo usuário**.

```
# Solicita ao usuário que insira a quantidade de alunos
n = int(input("Digite a quantidade de alunos: "))

# Inicializa a variável para armazenar a soma das médias
soma_das_medias = 0

# Laço para calcular a média de cada aluno
for i in range(1, n + 1):
    print(f"\nAluno {i}:")
    n1 = float(input("Digite a nota N1: "))
    n2 = float(input("Digite a nota N2: "))

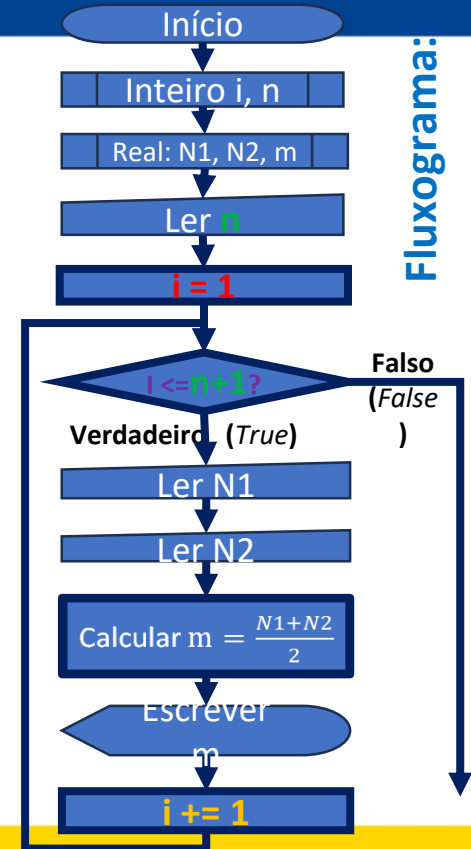
    # Calcula a média do aluno
    media = (n1 + n2) / 2
```





# EXEMPLO 07

Calcular e exibir a **média de n alunos**. Considerar que cada aluno possui duas **notas (N1 e N2) de mesmo peso**; e que a **quantidade de alunos (n)**, assim como as **notas de cada aluno** devem ser **fornecidas pelo usuário**.



# LOOP WHILE

## ESTRUTURA DE REPETIÇÃO WHILE

2. Controle de loop ENQUANTO (WHILE): O comando de repetição **ENQUANTO (while)** permite que um comando ou bloco de comandos seja executado **enquanto uma determinada condição for verdadeira**. A condição pode ser verificada no **início de cada repetição**, quando utilizamos o comando **while**, ou no **final**, quando utilizamos o comando **FAÇA .... ENQUANTO (do...while)**.

Necessita de uma **condição** que **enquanto for atendida, será executada!**

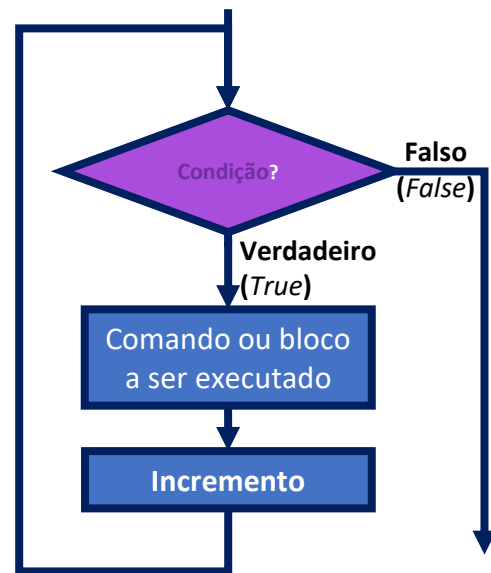


### ATENÇÃO

**Atenção!**

A estrutura de repetição **while** diferente do loop for, foca na condição, que será executada por inúmeras vezes enquanto for atendida.

## Fluxograma do Loop WHILE:



# EXEMPLO 08

Programa que utilize estrutura de **repetição while** para perguntar ao usuário se deseja ou não realizar a soma de 2 variáveis;

\*Enquanto responder Sim o programa continua solicitando os valores e calculando a soma.\*

```
while True:
    # Pergunta ao usuário se deseja realizar a soma de duas variáveis
    resposta = input("Deseja realizar a soma de 2 variáveis? (S/N): ").strip().upper()

    # Se a resposta for 'S', realiza a soma
    if resposta == 'S':
        # Solicita ao usuário que insira os valores das variáveis
        var1 = float(input("Digite o valor da primeira variável: "))
        var2 = float(input("Digite o valor da segunda variável: "))

        # Realiza a soma das variáveis
        soma = var1 + var2

        # Exibe o resultado da soma
        print(f"A soma de {var1} e {var2} é: {soma:.2f}\n")

    # Se a resposta for 'N', termina o loop
    elif resposta == 'N':
        print("Operação finalizada.")
        break

    # Caso o usuário insira uma resposta inválida, solicita novamente
    else:
        print("Resposta inválida! Por favor, digite 'S' para Sim ou 'N' para Não.\n")
```

# EXEMPLO 09

Programa em que utilize estrutura de **repetição DO WHILE** para perguntar ao usuário se deseja ou não realizar a soma de 2 variáveis; enquanto responder Sim o programa continua solicitando os valores e calculando a soma.

```
while True:
    resposta = input("Deseja realizar uma soma? (sim/não): ").lower()

    if resposta == "sim":
        num1 = float(input("Digite o primeiro número: "))
        num2 = float(input("Digite o segundo número: "))
        soma = num1 + num2 # Cálculo da soma
        print("A soma é:", soma)
    else:
        print("Encerrando o programa.")
        break
```

# 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.

## LINKS ÚTEIS:

- ❖ <https://colab.research.google.com/notebooks/intro.ipynb>
- ❖ <https://docs.oracle.com/en/java/javase/22/index.htm>
- ❖ [https://www.alura.com.br/artigos/java?utm\\_term=&utm\\_campaign=%5BSearch%5D+%5BPerformance%5D+-+Dynamic+Search+Ads+-+Artigos+e+Conte%C3%BAdos&utm\\_source=adwords&utm\\_medium=ppc&hsa\\_acc=7964138385&hsa\\_cam=11384329873&hsa\\_grp=111087461203&hsa\\_ad=687448474447&hsa\\_src=g&hsa\\_tgt=dsa-2276348409543&hsa\\_kw=&hsa\\_mt=&hsa\\_net=adwords&hsa\\_ver=3&gad\\_source=1&gclid=Cj0KCQjwLZixBhCoARIsAlC745B7qhj3m-fOX0DTfGIGtaYH2jAy1rUdKgJLk0sPkYYBGvDtDCdMG7MaAmc1EALw\\_wcB](https://www.alura.com.br/artigos/java?utm_term=&utm_campaign=%5BSearch%5D+%5BPerformance%5D+-+Dynamic+Search+Ads+-+Artigos+e+Conte%C3%BAdos&utm_source=adwords&utm_medium=ppc&hsa_acc=7964138385&hsa_cam=11384329873&hsa_grp=111087461203&hsa_ad=687448474447&hsa_src=g&hsa_tgt=dsa-2276348409543&hsa_kw=&hsa_mt=&hsa_net=adwords&hsa_ver=3&gad_source=1&gclid=Cj0KCQjwLZixBhCoARIsAlC745B7qhj3m-fOX0DTfGIGtaYH2jAy1rUdKgJLk0sPkYYBGvDtDCdMG7MaAmc1EALw_wcB)

# MODULARIDADE

A Modularidade consiste em **dividir um problema grande em problemas menores**; ou seja, dividi-lo **em pequenas partes**, ou **pequenos módulos**, isto é **modularidade**.

Cada parte menor ou módulo tem um algoritmo mais simples, o que facilita chegar à grande solução (de maior complexidade). Este conceito tem sido adotado desde meados da década de 1950 (PRESSMAN, 1995, p. 427).

**Módulo** é um bloco de programa que **geralmente efetua operações** computacionais **de entrada, processamento e saída**. Ao dividir um problema complexo em módulos, automaticamente usa-se a **ideia de abstração**.

**Logo, abstrair** um algoritmo **significa considerar isoladamente** um ou mais elementos de seu todo; significa, de forma geral, separar o todo em partes. A operação de funcionalidade de um módulo em um programa de computador baseia-se na existência de três características operacionais (adaptado de SEBESTA, 2003, p. 330).

# MODULARIDADE

Cada módulo possui um único ponto de entrada. Toda unidade de programa chamadora (**unidade mestre**) é suspensa durante a execução da unidade de módulo chamada (**unidade escravo**), o que implica a existência de somente um módulo em execução num determinado momento, exceto qdo se trabalha c/ paralelismo ou cor-rotinas.

Quando a execução da unidade escravo (módulo) é encerrada, o controle do fluxo de execução do programa volta para a primeira linha de instrução após a chamada do módulo na unidade mestre, quando se tratar da chamada de um procedimento.

No caso de chamada de uma função, o retorno ocorre exatamente na mesma linha de código que efetuou a chamada. Ao trabalhar com essa técnica, pode ocorrer a necessidade de dividir um módulo em outros tantos módulos quantos forem necessários, buscando uma solução mais simples de uma parte do problema maior.

A divisão de um módulo em outros módulos denomina-se **refinamento** (WIRTH, 1971). Tanto os **módulos de procedimentos como os de funções** são formas de estender os **recursos de abstração da técnica de programação estruturada**. Cada uma das formas de sub-rotinas será explanada mais adiante.

A **abstração** na programação de computadores é **semelhante a abstração na filosofia**.

O dicionário Aurélio (2010) define abstração como "**ato de separar mentalmente um ou mais elementos de uma totalidade complexa** (coisa, representação, fato), os quais só mentalmente podem subsistir fora dessa totalidade".

# PROCEDIMENTO

Um módulo de **procedimento** (sub-rotina) é um bloco de programa c/ início e fim, identificado por um nome que

referência seu uso em qualquer parte do programa principal ou do programa chamador da sub-rotina.

O uso de uma sub-rotina em um diagrama de blocos é idêntico às formas de desenho já utilizá-las. No caso do desenho da ação da sub-rotina, a diferença está na identificação dos rótulos utilizados nos símbolos terminal, em que as identificações tradicionais de início (primeiro símbolo) e fim (segundo símbolo) são substituídas pelo nome do procedimento no primeiro símbolo de terminal e pelo rótulo de identificação RETORNA no segundo símbolo de terminal. Na representação **da chamada da sub-rotina (ou procedimento) usa-se o símbolo processo predefinido**, idêntico ao símbolo de processamento com linhas paralelas à borda esquerda e também à borda direita, o qual tem por finalidade representar a chamada de um subprograma. A Figura 1 demonstra a estrutura básica dos diagramas de bloco que representam o programa chamador (a) e o módulo de procedimento (b) propriamente dito. Em seguida, apresenta-se a estrutura geral do código escrito em português estruturado.



# EXEMPLO 01: PROCEDIMENTO SOMADOR

## PSEUDO-CÓDIGO: Programa SOMADOR

{\*\*\* Trecho dos módulos para efetivação dos cálculos \*\*\*}

### procedimento ROTSOMA

var

R, A, B: real

início

escreva "OPERAÇÃO ADIÇÃO:"

Escreva "Digite valor A: "

leia (A)

escreva "Digite valor B: "

leia (B)

$R1 \leftarrow A + B$

escreva ("Resultado = ", R)

Fim

{\*\*\* Trecho da parte principal do programa \*\*\*}

var

escolha : inteiro

início

escreva("CALCULAR SOMA? 0 = NÃO e SAIR; 1 - SIM")

leia (escolha)

enquanto (escolha !=0)

se escolha ==1 então:

faça ROTSOMA

senão:

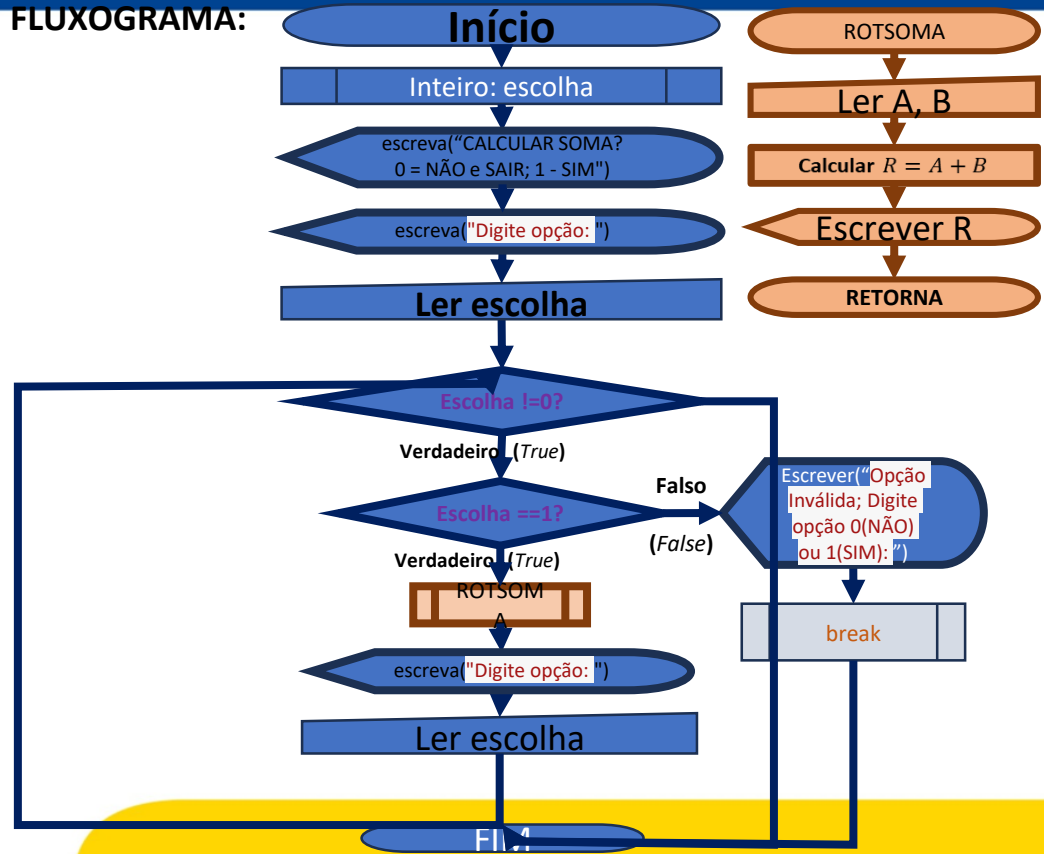
escreva("Opção Inválida; Digite opção 0(NÃO) ou 1(SIM): \n1")

fim\_se

fim\_enquanto

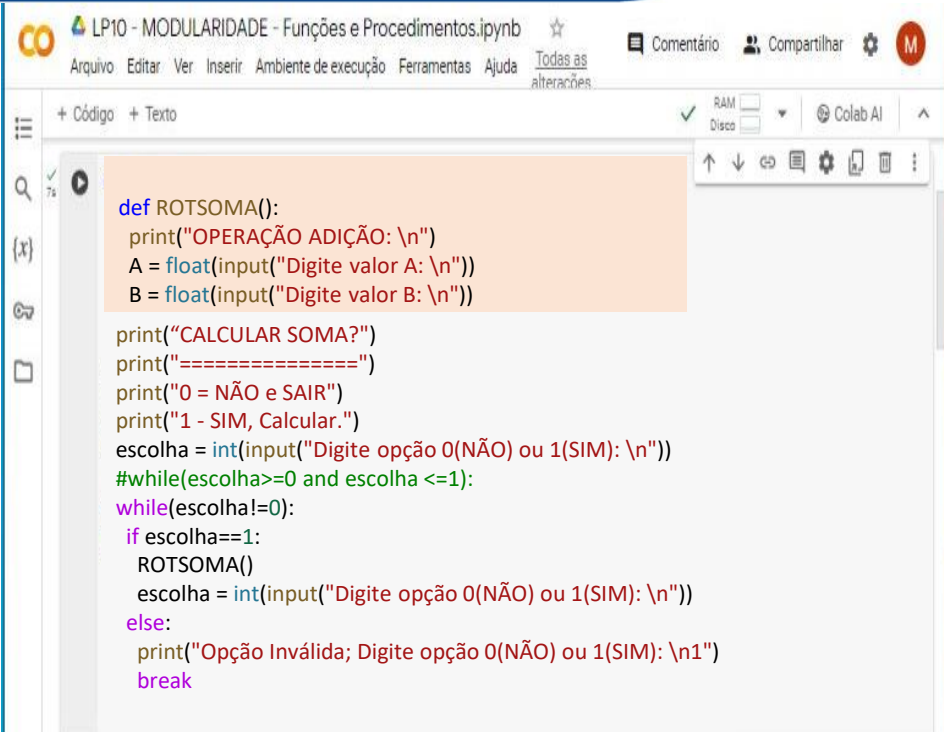
fim

## FLUXOGRAMA:



# EXEMPLO 01:

## PROCEDIMENTO SOMADOR



The screenshot shows a Jupyter Notebook window titled "LP10 - MODULARIDADE - Funções e Procedimentos.ipynb". The code is as follows:

```
def ROTSOMA():  
    print("OPERAÇÃO ADIÇÃO: \n")  
    A = float(input("Digite valor A: \n"))  
    B = float(input("Digite valor B: \n"))  
  
    print("CALCULAR SOMA?")  
    print("=====  
    print("0 = NÃO e SAIR")  
    print("1 - SIM, Calcular.")  
    escolha = int(input("Digite opção 0(NÃO) ou 1(SIM): \n"))  
    #while(escolha>=0 and escolha <=1):  
    while(escolha!=0):  
        if escolha==1:  
            ROTSOMA()  
            escolha = int(input("Digite opção 0(NÃO) ou 1(SIM): \n"))  
        else:  
            print("Opção Inválida; Digite opção 0(NÃO) ou 1(SIM): \n1")  
            break
```



Digitalize para  
acessar o código

# EXEMPLO 02:

## Programa principal

1. Apresentar um menu de seleção com cinco opções:
  1. Adição
  2. Subtração
  3. Multiplicação
  4. Divisão
  5. Fim de programa
2. Ao selecionar uma opção, a rotina correspondente deve ser executada.
3. Ao escolher o valor 5, o programa deve ser encerrado.

## Rotina 1 – Adição

1. Ler dois valores, no caso variáveis A e B.
2. Efetuar a soma das variáveis A e B, colocando o resultado na variável R.
3. Apresentar o valor da variável R.
4. Retornar ao programa principal.

## Rotina 2 – Subtração

1. Ler dois valores, no caso, variáveis A e B.
2. Efetuar a subtração das variáveis A e B, colocando o resultado na variável R.
3. Apresentar o valor da variável R.
4. Retornar ao programa principal.

## Rotina 3 – Multiplicação

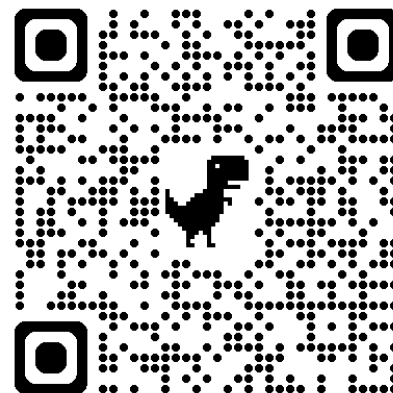
1. Ler dois valores, no caso, variáveis A e B.
2. Efetuar a multiplicação das variáveis A e B, colocando o resultado na variável R.
3. Apresentar o valor da variável R.
4. Retornar ao programa principal.

## Rotina 4 – Divisão

1. Ler dois valores, no caso, variáveis A e B.
2. Efetuar a divisão das variáveis A e B, colocando o resultado na variável R.
3. Apresentar o valor da variável R.
4. Retornar ao programa principal.



Figura 10.3 Hierarquia das rotinas do programa calculadora.



# EXEMPLO 02:

## CÓDIGO PYTHON:

```
def ROTSOMA():
```

```
    print("OPERAÇÃO ADIÇÃO: \n")
    A = float(input("Digite valor A: \n"))
    B = float(input("Digite valor B: \n"))
    R = A+B
    return R
```

```
def ROTSUBTRACAO():
```

```
    print("OPERAÇÃO SUBTRAÇÃO: \n")
    A = float(input("Digite valor A: \n"))
    B = float(input("Digite valor B: \n"))
    R = A-B
    return R
```

```
def ROTMULTIPLICACAO():
```

```
    print("OPERAÇÃO MULTIPLICAÇÃO: \n")
    A = float(input("Digite valor A: \n"))
    B = float(input("Digite valor B: \n"))
    R = A*B
    return R
```

```
def ROTDIVISAO():
```

```
    print("OPERAÇÃO DIVISÃO: \n")
    A = float(input("Digite valor A: \n"))
    B = float(input("Digite valor B: \n"))
    R = A/B
    return R)
```

```
print("MENU DE OPÇÕES:")
print("=====")
print("1 - ADIÇÃO")
print("2 - SUBTRAÇÃO")
print("3 - MULTIPLICAÇÃO")
print("4 - DIVISÃO")
print("5 - Fim do Programa")
```

```
escolha = int(input("Digite opção entre 1 e 5: \n"))
```

```
while(escolha !=5):
```

```
    if escolha==1:
```

```
        R = ROTSOMA()
```

```
        print("Resultado = ", R)
```

```
        escolha = int(input("Digite opção entre 1 e 5: \n"))
```

```
    if escolha==2:
```

```
        R = ROTSUBTRACAO()
```

```
        print("Resultado = ", R)
```

```
        escolha = int(input("Digite opção entre 1 e 5: \n"))
```

```
    if escolha==3:
```

```
        R = ROTMULTIPLICACAO()
```

```
        print("Resultado = ", R)
```

```
        escolha = int(input("Digite opção entre 1 e 5: \n"))
```

```
    if escolha==4:
```

```
        R = ROTDIVISAO()
```

```
        print("Resultado = ", R)
```

```
        escolha = int(input("Digite opção entre 1 e 5: \n"))
```

```
    else:
```

```
        print('Escolha errada! Você precisa escolher uma opção entre 1 e 5!')
```



Digitalize para  
acessar o código

# PASSAGEM DE PARÂMETRO

## Função **FATORIAL** c/ Passagem Parâmetro

Ocorre quando o **\*\*parâmetro formal da sub-rotina recebe do parâmetro real do trecho de programa chamador\*\*** um determinado conteúdo. Assim, o conteúdo passado pelo parâmetro real é copiado para o parâmetro formal, que neste caso assume o papel de variável local da sub-rotina. Qualquer modificação no conteúdo do parâmetro formal não afeta o valor do conteúdo do parâmetro real correspondente, ou seja, o processamento é executado dentro da sub-rotina, ficando o resultado dessa operação “preso” na própria sub-rotina. Como exemplo desse tipo de passagem de parâmetro, considere a apresentação do resultado do cálculo da fatorial de um número qualquer.

Quando se utiliza passagem de parâmetros em uma sub-rotina, é necessário mencionar no diagrama de bloco ou blocos da sub-rotina o uso desse parâmetro entre parênteses, conforme o exemplo da Figura 10.10. Note a indicação **FATORIAL(N)** no símbolo terminal do diagrama da sub-rotina.

### FLUXOGRAMA:

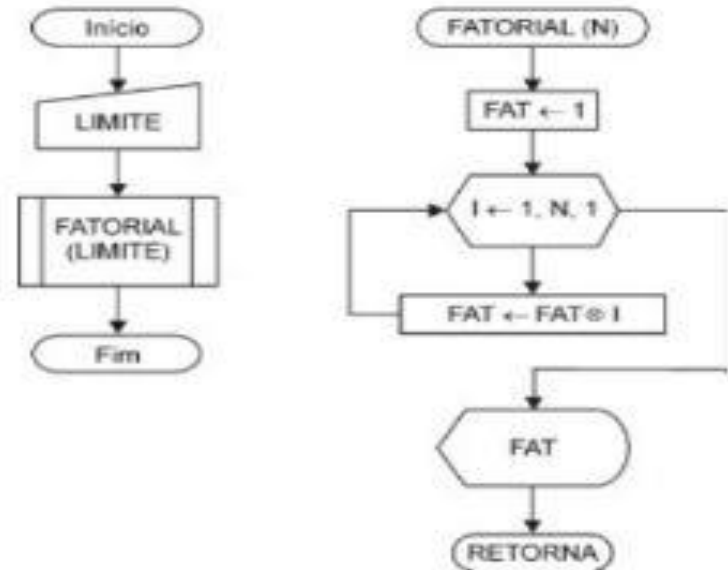


Figura 10.10 Diagramas de blocos com a sub-rotina FATORIAL.

# PASSAGEM DE PARÂMETRO

**Exemplo 03:** Função **FATORIAL** c/ Passagem Parâmetro. Ver (MANZANO, 2019. p.245)

**Pseudocódigo** do programa CALC\_FAT\_V1

**procedimento** FATORIAL(N : inteiro)

**var**

i, FAT : inteiro

**início**

FAT  $\leftarrow$  1

**para** i **de** 1 **até** N+1 **passo** 1 **faça:**

FAT  $\leftarrow$  FAT \* i

**fim\_para**

**escreva** (FAT)

**Fim**

**var**

LIMITE : inteiro

**início**

**escreva** "Qual fatorial: "

**leia** FATORIAL(LIMITE)

**fim**

7:11

**CÓDIGO EM PYTHON:** `print("Programa CALC_FAT_V1")`

```
def FATORIAL(N):
```

```
    fat = 1
```

```
    for i in range(1, N + 1):
```

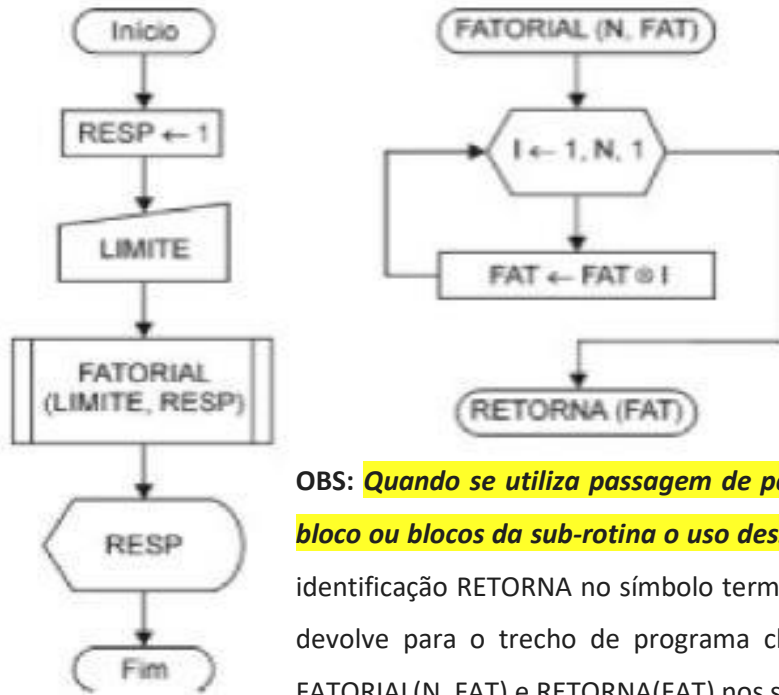
```
        fat *= i
```

```
    return fat
```

```
LIMITE = int(input("Qual fatorial: "))
```

```
print(f'O fatorial de {LIMITE} é  
{FATORIAL(LIMITE)}')
```

# PASSAGEM DE PARÂMETRO



Ocorre quando o **parâmetro real do trecho de programa chamador recebe o conteúdo do parâmetro formal de uma sub-rotina**. Após certo processamento dentro da sub-rotina, o parâmetro formal **reflete a alteração de seu conteúdo no parâmetro real**. Qualquer modificação feita no conteúdo do parâmetro formal implica alteração imediata do conteúdo do parâmetro real correspondente. **A alteração efetuada é devolvida ao trecho do programa chamador**. Como exemplo desse tipo de passagem de parâmetro, considere a apresentação do resultado do cálculo da fatorial de um número qualquer; observe o uso do comando `var` na declaração do parâmetro formal da sub-rotina em português estruturado.

**OBS:** Quando se utiliza passagem de parâmetros por referência em uma sub-rotina, é necessário mencionar no diagrama de bloco ou blocos da sub-rotina o uso desse parâmetro entre parênteses, tanto no início da sub-rotina como no fim, no rótulo de identificação `RETORNA` no símbolo terminal. Essa forma é necessária para esclarecer o uso de uma passagem de parâmetro que devolve para o trecho de programa chamador algum conteúdo, conforme o exemplo da Figura 10.11. Note as indicações `FATORIAL(N, FAT)` e `RETORNA(FAT)` nos símbolos de terminal do diagrama da sub-rotina.

**Figura 10.11** Diagramas de blocos com a sub-rotina `FATORIAL`.

# PASSAGEM DE PARÂMETRO

**Pseudocódigo** do programa CALC\_FAT\_V1

```
procedimento FATORIAL(N : inteiro, var FAT: inteiro)
var
    i: inteiro
início
    Fat ← FAT[0]
    para i de 1 até N passo 1 faça:
        FAT ← FAT * i
    fim_para
    escreva (FAT)
Fim
var
    LIMITE, RESP : inteiro
início
    escreva "Qual fatorial: "
    leia LIMITE
    FATORIAL(LIMITE, RESP)
    Escreva (RESP)
```

fim

7:11

**Exemplo 04:** Função **FATORIAL** c/ Passagem Parâmetro por Referência. Ver (MANZANO, 2019. p.247)

## CÓDIGO EM PYTHON:

```
def FATORIAL(N, FAT):
    fat = FAT[0]
    for i in range(1, N + 1):
        fat *= i
    FAT[0] = fat

LIMITE = int(input("Qual fatorial: "))
RESP = [1]
FATORIAL(LIMITE, RESP)
print(f'O fatorial de {LIMITE} é {RESP[0]}')
```

A alteração efetuada é devolvida ao trecho do programa chamador.



# 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.

DANKSCHEEN  
 SPASIBO  
 TASHAKKUR  
 ATU  
 SUKSAMA  
 EKHMET  
 SHUKURIA  
 GOZAIMASHITA  
 EFCHARISTO  
 JUSPAXAR  
 KOMAPSUMNIDA  
 GRAZIE  
 MEHRBANI  
 PALDIES  
 BOLZIN  
 MERCİ  
 BİYAN  
 SHUKRIA  
 TINGKI  
 THANK  
 YOU

# Linguagem Python ou Java?



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

**Fonte:** Elaborado pelo autor