



CENTRO UNIVERSITÁRIO
UNITOP

AGRONOMIA

Introdução à Informática

Prof. Me. **Paulo Augusto**

Palmas, 20 de fevereiro de 2026

Programação

□ Computador

- É extremamente eficiente e rápido para realizar operações e cálculos complexos;
- Não raciocina ou reflete a intenção do programador;
- Interpreta de maneira literal. (como criança);
- **Lenda da programação...**
- Sempre que inicia uma programação começa do Zero.

Programação

Aplicabilidade da Programação

□ Presente em todas as áreas da computação:

- Hardware;
- Sistemas Operacionais;
- Análise de Sistemas;
- Banco de Dados;
- Desenvolvimento Web;
- Redes de Computadores;
- Etc.

Programação

□ Sequência lógica

- Estes pensamentos devem ser descritos como uma *sequência de instruções*, que devem ser seguidas em ordem para se cumprir uma determinada tarefa;
- *Passos* executados até se atingir um objetivo ou solução de um problema

Programação

❑ Instrução

- Cada um dos *passos*, cada uma das ações a tomar (obedecendo a *sequência lógica*) para ir resolvendo o problema, ou para ir executando a tarefa;
- Uma só instrução não resolve problemas.

Programação

☐ Instrução

Ex.: para “fazer omelete”

➤ **InSTRUÇÕES:**

- “quebrar ovos”, “bater ovos”, “pôr sal”, “ligar fogão”, “pôr óleo na frigideira”, “pôr frigideira no fogo”, “fritar ovos batidos”, etc...

➤ **Quanto às instruções isoladas:**

- Só “quebrar ovos”, ou só “pôr óleo na frigideira”, não é suficiente para cumprir a tarefa “fazer omelete”

➤ **Quanto à sequência lógica:**

- Se executarmos “fritar ovos batidos” antes de “bater ovos”, ou pior, antes de “quebrar ovos”, não iremos cumprir a tarefa “fazer omelete”

Programação

❑ Algoritmo

- Sequência finita de passos, seguindo uma sequência lógica que levam à execução de uma tarefa;
- Claro e preciso.

Programação

❑ Algoritmo

Exemplo:

- Acordar;
- Levantar;
- Escovar os dentes;
- Tomar café;
- Sair de casa.

Programação

□ Algoritmo

Ex.:

- Quando uma dona de casa prepara um bolo, segue uma **receita**, que nada mais é do que um **algoritmo** em que cada instrução é um passo a ser seguido para que o prato fique pronto com sucesso:
 1. Bata 4 claras em neve;
 2. Adicione 2 xícaras de açúcar;
 3. Adicione 2 colheres de farinha de trigo, 4 gemas, uma colher de fermento e duas colheres de chocolate;
 4. Bata por 3 minutos;
 5. Unte uma assadeira com margarina e farinha de trigo;
 6. Coloque o bolo para assar por 20 minutos.

Programação

❑ Algoritmo

Envolve pensar de forma estruturada e sequencial para resolver um problema utilizando um computador. Ela é a base para escrever códigos que o computador consiga entender e executar.

- **Entrada (Input):** Dados ou informações que o programa recebe.
- **Processamento:** Ações ou cálculos que o programa faz com os dados.
- **Saída (Output):** Resultado que o programa gera a partir do processamento.

Programação

❑ Algoritmo

FASES para desenvolver o algoritmo:

- Determinar o problema.
- Dividir a solução nas três fases:



➤ Exemplo:

- Problema: calcular a média de três números
- Dados de entrada: os números, X, Y, Z
- Processamento: somar os dois números e dividir a soma por 2

$$\frac{X + Y + Z}{2}$$

- Dados de saída: a média

Programação

❑ Algoritmo

1. Receber o primeiro número
2. Receber o segundo número
3. Receber o terceiro número
4. Somar todos os números
5. Dividir a soma por 3
6. Mostrar o resultado da divisão



$$\frac{X + Y + Z}{2}$$

Programação

□ Algoritmos e Lógica de Programação

Formas de representação de algoritmos:

- Descrição Narrativa;
- Fluxograma;
- Pseudocódigo.

Programação

□ Algoritmos e Lógica de Programação

Descrição Narrativa

➤ Caso 01 – Lavar o cabelo:

1. Molhar o cabelo;
2. Aplicar o xampú;
3. Massagear;
4. Enxaguar.



Programação

□ Algoritmos e Lógica de Programação

Descrição Narrativa

➤ Exercício:

- Caso 02 – Trocar uma lâmpada queimada.



- Caso 03 – Trocar o pneu furado de um carro.

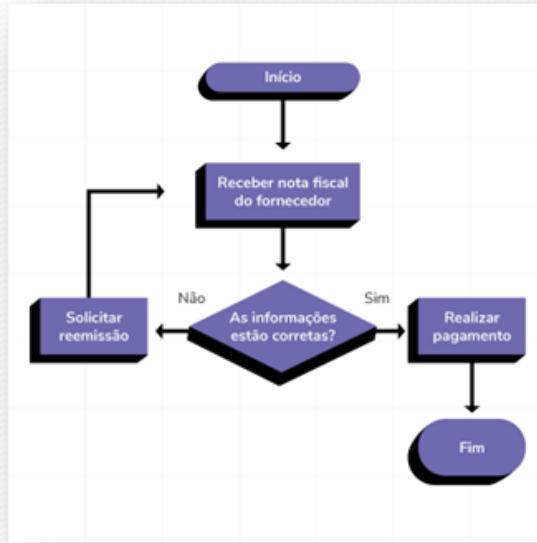


Programação

❑ Algoritmos e Lógica de Programação

Fluxograma

- Representação gráfica, onde formas geométricas diferentes implicam ações distintas



Programação

❑ Algoritmos e Lógica de Programação

Fluxograma

➤ Principais Formas:



= Início e Fim do Fluxograma



= Fluxo de Dados



= Operação de Entrada de Dados



= Operação de Saída de Dados



= Operação de Atribuição (Processamento)



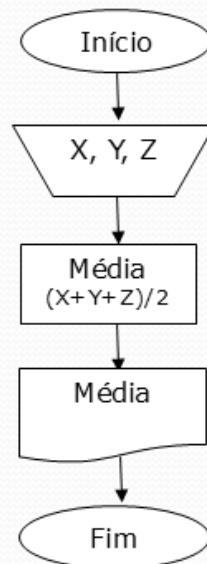
= Decisão

Programação

□ Algoritmos e Lógica de Programação

Fluxograma

➤ Exemplo:



Programação

□ Algoritmos e Lógica de Programação

Pseudocódigo ou Português Estruturado

- Não é uma linguagem;
- Sintaxe.

Algoritmo Soma

Var

n1, n2, S : Inteiro

Início

Escreva ("Entre com o primeiro valor: ")

Leia (N1)

Escreva ("Entre com o segundo valor: ")

Leia (N2)

S <- N1 + N2

Escreva ("Soma = ", S)

Fim.

```
1 programa
2 {
3     funcao inicio()
4     {
5         inteiro a=1, b=0
6         se (a<b)
7         {
8
9     }
10    }
11 }
```

Programação

❑ Variável

- É uma espaço (posição) reservado na memória para guardar algum dado;
- Possui um nome e um valor;
- Durante a execução do algoritmo, pode ter seu valor alterado.

Programação

❑ Variável

Tipos mais comuns:

- **Inteiro (int)**: Números inteiros (ex: 1, -5, 20)
- **Ponto flutuante (float)**: Números com casas decimais (ex: 3.14, -0.5)
- **Texto (string)**: Sequências de caracteres (ex: "Olá, Mundo!")
- **Booleano (bool)**: Verdadeiro ou falso (ex: True, False)

```
idade = 25      # Variável do tipo inteiro
nome = "Carlos" # Variável do tipo string
altura = 1.75   # Variável do tipo float
ativo = True    # Variável do tipo booleano
```

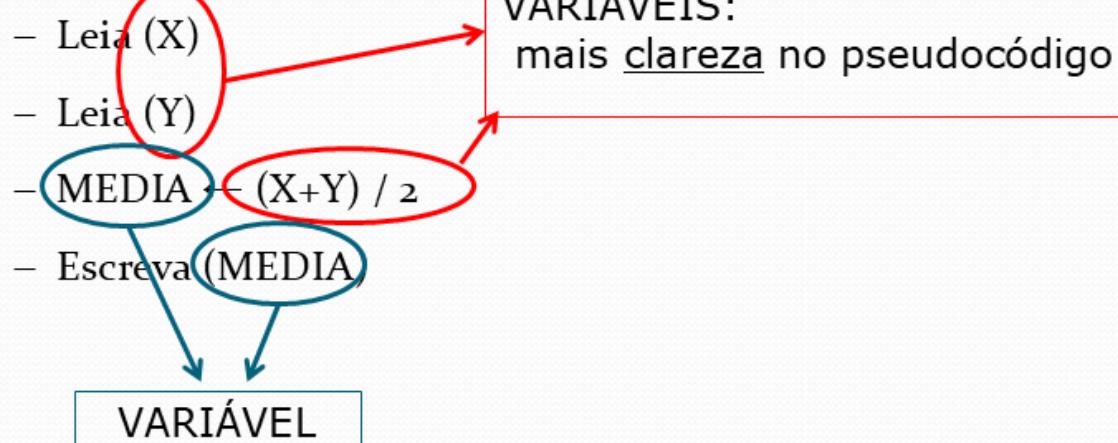
Programação

❑ Variável

Exemplo:

“Calcular a média de quatro números”

➤ PSEUDOCÓDIGO:



Programação

❑ Variável

➤ Uso de Variáveis

- O primeiro caractere do nome de uma variável não poderá ser, em hipótese alguma, um número. Sempre deverá ser uma letra;
- O nome de uma variável não poderá possuir espaços em branco;
- Não poderá ser nome de uma variável uma palavra reservada;
- Não poderão ser utilizados outros caracteres a não ser letras e números, com exceção do caractere *underscore/underline* 

Programação

❑ Variável

➤ Atribuição

- Atribui o valor da direita à variável da esquerda
 - $\text{MEDIA} \leftarrow (\text{N1}+\text{N2}) / 4$
(Lê-se media recebe N1+...)
 - Neste caso, estamos atribuindo o resultado da fórmula à variável média;
- Outros Exemplos:
 - $a \leftarrow 3;$
 - $a \leftarrow x;$

Programação

□ Operadores Aritméticos

OPERAÇÃO	SÍMBOLO
Adição	+
Subtração	-
Multiplicação	*
Divisão	/
Exponenciação	**

Hierarquia das Operações Aritméticas

- 1º () Parênteses
- 2º Exponenciação
- 3º Multiplicação, divisão (o que aparecer primeiro)
- 4º + ou - (o que aparecer primeiro)

Exemplos:

TOTAL = PRECO * QUANTIDADE

1 + 7 * 2 ** 2 - 1 = 28

3 * (1 - 2) + 4 * 2 = 5

MEDIA = (N1+N2+N3+N4) / 4

Programação

❑ Métodos para Construção de Algoritmo:

- Ler atentamente o enunciado;
- Retirar do enunciado a relação das entradas de dados;
- Retirar do enunciado a relação das saídas de dados;
- Determinar o que deve ser feito para transformar as entradas determinadas nas saídas especificadas.
- Construção do algoritmo.

Programação

□ Operadores relacionais

- São muito usados quando temos que tomar decisões nos algoritmos. Com eles fazemos testes, comparações, que resultam em valores lógicos (verdadeiro ou falso):

Exemplo:

tendo duas variáveis, $A = 5$ e $B = 3$:

Descrição	Símbolo
Igual a	=
Diferente de	<> ou #
Maior que	>
Menor que	<
Maior ou igual a	\geq
Menor ou igual a	\leq

Expressão	Resultado
$A = B$	Falso
$A <> B$	Verdadeiro
$A > B$	Verdadeiro
$A < B$	Falso
$A \geq B$	Verdadeiro
$A \leq B$	Falso

Programação

□ Lógica de Programação

➤ Algoritmos:

- Conjunto de passos;
- Sequência lógica;
- Organização coerente das instruções.

Instrução:

- Se a porta estiver aberta:
 - Pegar o lixo;
 - Levar o lixo para fora.

- Se a porta estiver aberta:
 - Pegue a chave;
 - Enfie na fechadura;
 - Gire para esquerda até parar;
 - Abra a porta;
 - Pegue o lixo;
 - Leve o lixo para fora.

Instrução:

- Pegue o lixo;
- Leve o lixo para fora;

Instrução:

- Abra a porta
- Pegue o lixo;
- Leve o lixo para fora;

Instrução:

- Pegue a chave;
- Abra a porta;
- Pegue o lixo;
- Leve o lixo para fora;

Programação

□ Lógica de Programação

➤ Algoritmos:

```
Algoritmo CalcularAreaRetangulo
var
    base, altura, area: real

Inicio
    Escreva("Digite o valor da base: ")
    Leia(base)

    Escreva("Digite o valor da altura: ")
    Leia(altura)

    area <- base * altura

    Escreva("A área do retângulo é: ", area)
Fim
```

```
import java.util.Scanner

public class CalcularAreaRetangulo {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        double base, altura, area;

        System.out.print("Digite o valo da base: ");
        base = sc.nextDouble();

        System.out.print("Digite o valo da altura: ");
        altura = sc.nextDouble();

        System.out.println("A área do retângulo é: " +area);
        sc.close();
    }
}
```

Programação

Lógica de Programação (estrutura básica)

Essas estruturas são fundamentais para controlar o fluxo do programa. As principais são:

- **Sequência;**
- **Seleção (Condicionais);**
- **Repetição (Laços).**

Programação

Estrutura (Sequência)

A sequência é a execução das instruções uma após a outra, na ordem em que são escritas no código.

Ex.:

1. Receber um número;
2. Multiplicar o número por 2;
3. Exibir o resultado.

```
1  Início
2  Receber  número
3      resultado ← número * 2
4  Escreva  resultado
5  Fim
```

Programação

Estrutura (Seleção (Condicionais))

Permite que o programa execute diferentes ações dependendo de uma condição. A estrutura mais comum é o **if**.

```
# Se o número for maior que 10, imprime "Maior"  
# Caso contrário, imprime "Menor ou igual"
```

```
numero = 12  
if numero > 10:  
    print("Maior")  
else:  
    print("Menor ou igual")
```

```
1 Início  
2 Leia número  
3 Se número > 10 então  
4     Escreva "O número é maior que 10."  
5 senão  
6     Escreva "O número é menor ou igual a 10."  
7 Fim
```

Programação

Estrutura (Repetição, Laços)

Laços (ou loops) permitem repetir um bloco de código várias vezes, **for** e **while**.

```
# Imprimir números de 1 a 5

for i in range(1, 6):
    print(i)
```

```
# Imprimir números de 1 a 5

i = 1
while i < 6:
    print(i)
    i = i + 1
```

```
1 Início
2 Para i de 1 até 5 faça
3     Escreva "O número: ", i
4 Fim Para
5 Fim
```

```
1 Início
2 i = 1
3 Enquanto i < 6 faça
4     Escreva "O número: ", i
5     i = i + 1
6 Fim Enquanto
7 Fim
```

Programação

□ Linguagem de programação

- A verdadeira (e única) linguagem de computador é a **linguagem de máquina** (binária), mas para a maioria das pessoas essa linguagem é **ininteligível**.
- Os programadores precisam de um intermediário entre eles e a máquina. Uma linguagem que facilite o processo de programar computadores mas que no final seja traduzida para a linguagem de máquina.

Programação

❑ Linguagem de programação

- Uma linguagem de programação permite desenvolver os conjuntos de instruções que constituem o programa de computador.
- Existem muitas linguagens de programação diferentes, cada uma com seu vocabulário, gramática e usos exclusivos.

Programação

□ Linguagem de programação

➤ Linguagens de Baixo nível:

- **Linguagem de máquina** – Código binário diretamente entendido pelo processador. É extremamente rápida, mas difícil de escrever e entender.;
- **ASSEMBLY** – Linguagem simbólica que representa instruções de máquina com mnemônicos, facilitando a programação em relação à linguagem de máquina..

➤ Linguagens de Alto nível:

- **COBOL** – Voltada para aplicações de negócios e processamento de grandes volumes de dados.;
- **JAVA** – Orientada a objetos, muito usada em aplicativos corporativos, web e móveis.;
- **C** – Linguagem poderosa e eficiente, muito usada em sistemas operacionais e softwares que exigem desempenho.;
- **PYTHON** – Fácil de aprender, com sintaxe clara; usada em ciência de dados, inteligência artificial, web e automação..

Programação

❑ Linguagem de programação

➤ Linguagens Visuais:

- **Programação em blocos** – Linguagens que utilizam blocos visuais para construir programas, facilitando o aprendizado para iniciantes (ex.: Scratch, Blockly).

Programação

ASSEMBLY

```

DOSSEG
.MODEL SMALL
.STACK 100h          ; Define stack size
EXTRN _IntDivide:PROC

.CODE
PUBLIC _Average

_Average PROC
    push bp
    mov bp, sp           ; Configura base pointer para acessar
                           ; parâmetros

    les bx, [bp+4]       ; Carrega ponteiro do array em ES:BX
    mov cx, [bp+8]       ; Carrega tamanho do array em CX
    mov ax, 0             ; Inicializa acumulador AX

    AverageLoop:
        add ax, es:[bx]   ; Soma valor do array em AX
        add bx, 2          ; Passa para o próximo elemento (palavra = 2
                           ; bytes)
        loop AverageLoop ; Decrementa CX e repete enquanto CX > 0

        push cx            ; Push do divisor (tamanho do array)
        push ax            ; Push do dividendo (soma dos elementos)
        call _IntDivide    ; Chama a função de divisão inteira
        add sp, 4           ; Ajusta stack (pop dos parâmetros do call)

        pop bp
        ret
_Average ENDP

END

```

Este programa calcula a média de uma lista de números.

```

DOSSEG
.MODEL SMALL
EXTRN _IntDivide:PROC

.CODE
PUBLIC _Average

_Average PROC
    push bp
    mov bp, sp
    les bx, [bp+4]
    Mov cx, [bp+8]
    mov ax, 0

AverageLoop:
    add ax, es: [bx]
    add bx, 2
    Loop AverageLoop

    push WORD PRG [bp+8]
    push ax
    call _IntDivide
    add sp, 4
    pop bp
    ret
_Average ENDP

END

```

Programação

Linguagem C

```
#include <stdio.h>

int main() {
    float num, den;
    printf("Digite o numerador: ");
    scanf("%f", &num);
    printf("Digite o denominador: ");
    scanf("%f", &den);
    if (den == 0) {
        printf("Numerador não pode ser zero\n");
        return 0;
    }
    printf("Resultado da divisão: %.2f", num/den);
    return 0;
}
```

Este programa calcula a divisão
de dois números quaisquer

Programação

Linguagem de programação em Blocos - Scratch

<https://scratch.mit.edu>



Este programa calcula a área de um retângulo.

Programação

□ Algoritmo para dividir dois números

1. Iniciar o programa
2. Definir variáveis
3. Apresentar mensagem para o usuário digitar o dividendo
4. Ler o dividendo
5. Apresentar mensagem para o usuário digitar o divisor
6. Ler o divisor
7. Se divisor igual a zero então apresentar mensagem e terminar o programa
8. Apresentar o resultado da divisão do dividendo pelo divisor
9. Terminar o programa

Programação

❑ Linguagem C

```
#include <stdio.h>

int main() {
    float num, den;
    printf("Digite o numerador: ");
    scanf("%f", &num);
    printf("Digite o denominador: ");
    scanf("%f", &den);
    if (den == 0) {
        printf("Numerador não pode ser zero\n");
        return 0;
    }
    printf("Resultado da divisão: %.2f", num/den);
    return 0;
}
```

Este programa calcula a divisão
de dois números quaisquer

Programação

Python – O que é?

Python é uma linguagem de programação de alto nível, fácil de aprender, e muito poderosa. É uma das linguagens mais populares no mundo da programação devido à sua simplicidade e versatilidade.

- **Sintaxe simples:** Não precisa de símbolos complexos como {} ou ; para estruturar o código.
- **Usos variados:** é usada para desenvolver websites, automações, jogos, inteligência artificial, ciência de dados, e muito mais.

Programação

Python – *Escrevendo seu Primeiro Código Python*

No Python, você pode usar qualquer editor de texto para escrever seu código (como o **Bloco de Notas**, **Visual Studio Code**, **PyCharm**, dentre outros).

Ex.: **Olá, Mundo!**

```
print ("Olá, Mundo!")
```



<https://www.jdoodle.com/>
<https://www.mycompiler.io/>
<https://www.onlinegdb.com/>

Programação

Python – *Variáveis e Tipos de Dados*

Em Python pode armazenar valores em variáveis. Uma variável é como um "recipiente" para armazenar dados.

Tipos de dados comuns:

- **Inteiros** (int): números inteiros, como 5, 100, -7
- **Pontos flutuantes** (float): números com ponto decimal, como 3.14, -0.001
- **Strings** (str): texto, como "Olá", "Python"
- **Booleanos** (bool): valores True ou False

```
nome = "João"          # String
idade = 25             # Inteiro
altura = 1.75           # Float
is_estudante = True    # Booleano

print(nome)      # Vai exibir "João"
print(idade)     # Vai exibir 25
```

Programação

Python – *Operações Matemáticas*

Python permite realizar operações matemáticas facilmente.

```
a = 10
b = 5

soma = a + b      # Soma
subtracao = a - b # Subtração
multiplicacao = a * b # Multiplicação
divisao = a / b    # Divisão

print(soma)      # 15
print(subtracao) # 5
print(multiplicacao) # 50
print(divisao)    # 2.0
```

Programação

Python – *Estruturas de Controle*

As estruturas condicionais são usadas para tomar decisões no seu código.

Condicionais (if, else)

```
idade = 18

if idade >= 18:
    print("Você é maior de idade!")
else:
    print("Você é menor de idade!")
```

Programação

Python – *Laços de Repetição*

Os laços de repetição permitem executar um bloco de código várias vezes.

Laços de repetição (`for`, `while`):

```
# Exemplo de laço 'for'

for i in range(5):    # Repete de 0 a 4
    print(i)
```

```
# Exemplo de laço 'while'
contador = 0
while contador < 5:
    print(contador)
    contador += 1    # Incrementa contador
```

Programação

Python – *Funções*

São blocos de código que você pode reutilizar. Elas são definidas com a palavra-chave `def`.

```
def saudacao(nome):
    print(f"Olá, {nome}!")

saudacao("Maria") # Chama a função e imprime "Olá, Maria!"

print("Olá, " + nome + "!"")
```

Programação

Python – *Função input()*

Em Python a interação com o usuário é feita principalmente com a função `input()`. Ela permite capturar o que o usuário digita no teclado. O valor capturado sempre é do tipo `string` (texto), então, se for necessário outro tipo de dado (como números), será necessário convertê-lo.

```
# Recebe o nome do usuário
nome = input("Qual é o seu nome? ")

# Exibe uma mensagem para o usuário
print(f"Olá, {nome}!")
```

Programação

Python – *Função input() – Trabalhando com números*

Como mencionado no slide anterior, a função `input()` sempre retorna uma string. Se quisermos trabalhar com números (inteiros ou decimais), precisamos converter a entrada para o tipo apropriado usando `int()` (para números inteiros) ou `float()` (para números decimais)..

```
# Recebe dois números do usuário
numero1 = input("Digite o primeiro número: ")
numero2 = input("Digite o segundo número: ")

# Converte as entradas para números inteiros
numero1 = int(numero1)
numero2 = int(numero2)

# Realiza uma operação (soma)
soma = numero1 + numero2

# Exibe o resultado
print(f"A soma de {numero1} e {numero2} é {soma}.")
```

Programação

- Fazer um programa em python que faça uma divisão de dois números.

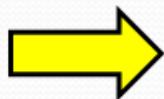
```
dividendo = float(input("Digite o dividendo: "))
divisor = float(input("Digite o divisor: "))

if divisor == 0:
    print("Divisor não pode ser zero!!!")
else:
    resultado = dividendo / divisor
    print("Resultado da divisão: {:.2f}".format(resultado))
```

Programação

```
dividendo = float(input("Digite o dividendo: "))
divisor = float(input("Digite o divisor: "))

if divisor == 0:
    print("Divisor não pode ser zero!!!")
else:
    resultado = dividendo / divisor
    print("Resultado da divisão: {:.2f}".format(resultado))
```



<https://www.jdoodle.com/>

<https://www.mycompiler.io/>

<https://www.onlinegdb.com/>

Programação

- Fazer um programa em python que peça ao usuário para digitar um número, em seguida mostre a tabuada de multiplicar do número digitado.

```
1 # Programa de Tabuada em Python
2 numero = int(input("Digite um número para ver sua tabuada: "))
3
4 print(f"\nTabuada do {numero}:")
5 print("-" * 15) # Linha divisória
6
7 for i in range(1, 11): # De 1 a 10
8     resultado = numero * i
9     print(f"{numero} x {i:2} = {resultado:3}") # :2 e :3 para alinhamento
10
11 print("-" * 15) # Linha divisória
```