



Compreender lógica  
de programação



**Aula 1 – Entrada e Saída de Dados**

**Aula 2 – Aula 02 – Web (HTTP / HTTPS)**

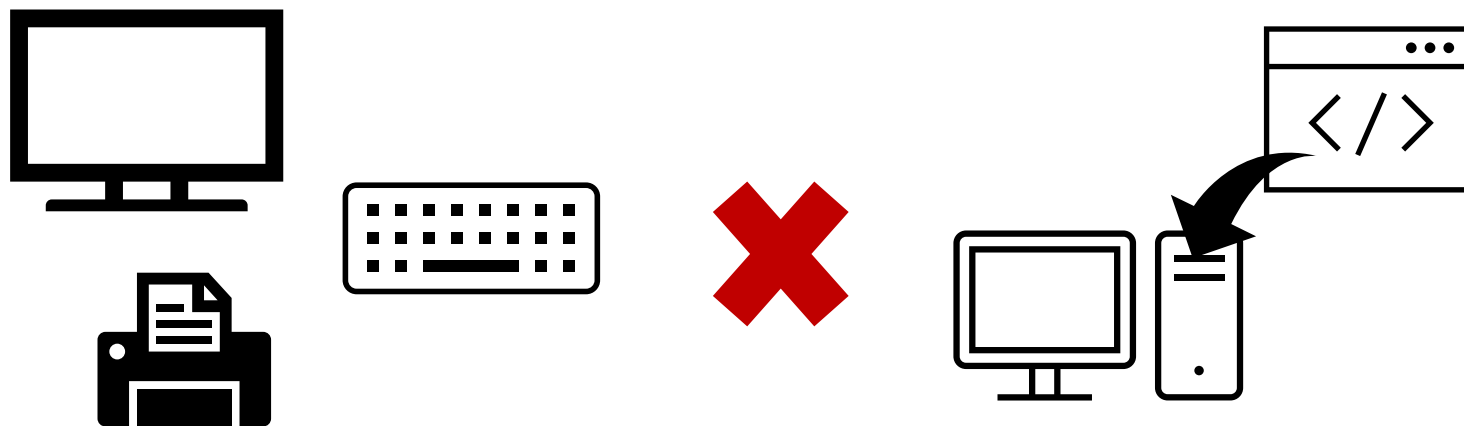
**Aula 3 – DNS**

**Aula 4 – DHCP**



# Aula 01 – Entrada e Saída de Dados

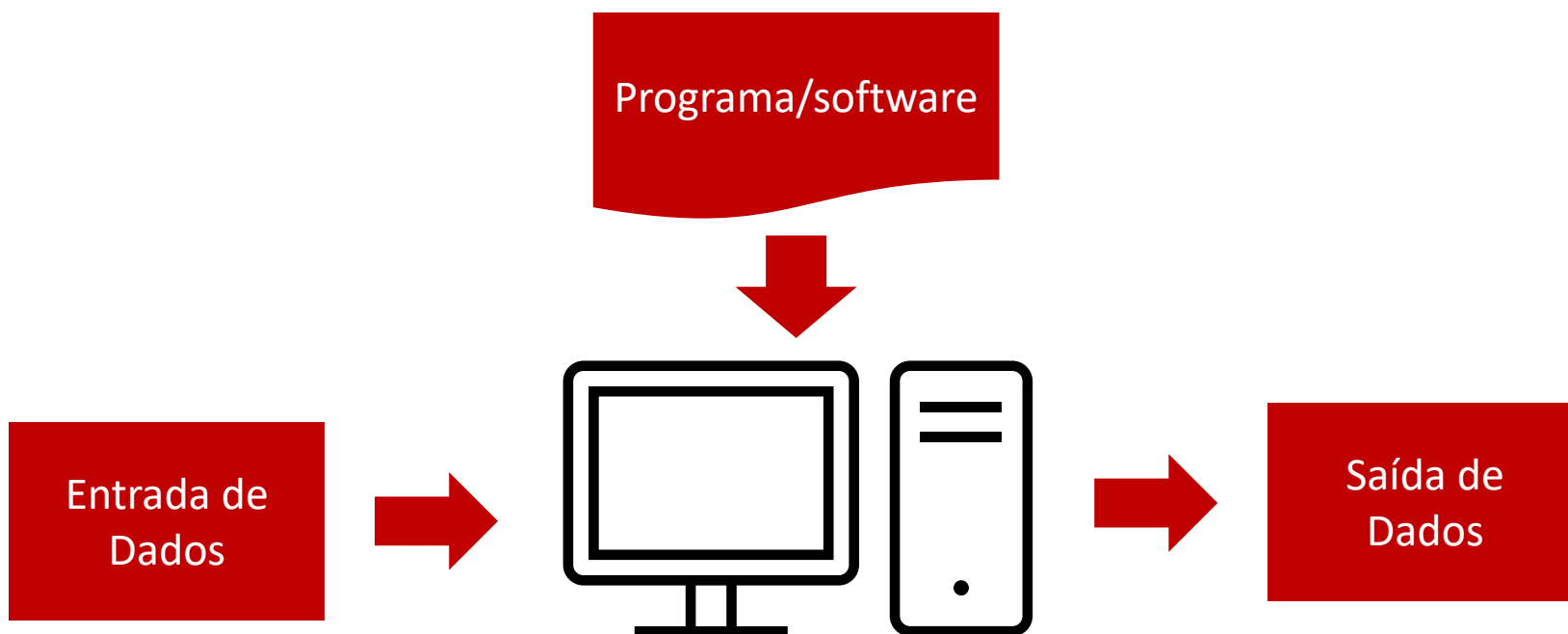
- **Hardware:** é a parte física do dispositivo (impressora, monitor, teclado, entre outros);
- **Software:** são os conjuntos de programas que controlam o dispositivo para tarefas as quais eles foram especificados



**Hardware vs Software**

Fonte: Adaptado do Microsoft Power Point (Autor)

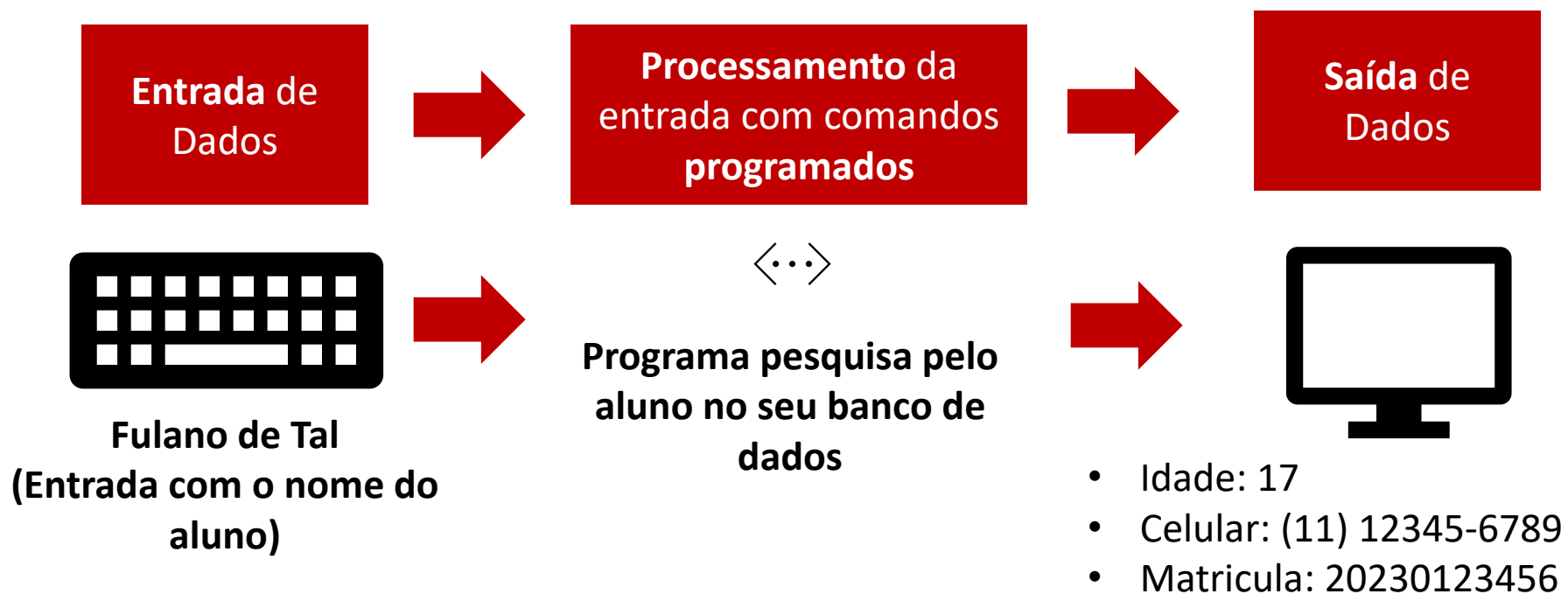
O programa ou *software* é quem determina como o computador irá responder as **entradas** e qual a finalidade que queremos dele



***Processamento de Dados***

*Fonte: Adaptado do Microsoft Power Point (Autor)*

Exemplo de um programa de uma consulta de dados de um aluno:



*Exemplo de Programa de Computador*

*Fonte: Adaptado do Microsoft Power Point (Autor)*

Como os programas são criados?

Como os  
programas são  
criados?



***Como Programas são criados?***

*Fonte: Adaptado do Microsoft Power Point (Autor)*

Como os  
programas são  
criados?



Os programas são  
criados usando a  
*linguagem* de  
*programação!*



***Como Programas são criados?***

*Fonte: Adaptado do Microsoft Power Point (Autor)*



O que é a linguagem de programação?

Uma linguagem de programação possui:

- **Símbolos reservados** (ex.: +, -, \*, /, etc);
- **Comandos/palavras reservadas** (ex.: if, else, for, while, etc)

Como criar um  
programa com a  
linguagem de  
programação?



→ Sequência de  
ideias/comandos para  
alcançar o objetivo final do  
seu programa



***Como usar a linguagem de programação?***  
*Fonte: Adaptado do Microsoft Power Point (Autor)*

Como os programas são criados?



Exemplo de solução de um problema usando o raciocínio de um programador:

**Como trocar uma lâmpada queimada?**

Como os programas são criados?



Qual o resultado ou a **saída** esperada?

Como os programas são criados?



Qual o resultado ou a **saída** esperada?

**Uma nova lâmpada instalada e que esteja funcionando!**

O que eu preciso para trocar a lâmpada?

**Dados de entrada:**

- Lâmpada;
- Escada;

Modo para troca da lâmpada ou **processamento**:

- Posicione a escada embaixo da lâmpada queimada;
- Pegue a nova lâmpada;
- Suba na escada;
- Remover a lâmpada queimada;
- Rosquear a nova lâmpada;
- Descer da escada
- Testar a nova lâmpada instalada.

Sequência de ideias para criarmos um programa:

- Identificar o que deve ser apresentado pelo programa  
**(saída de dados)**
- Identificar quais são os dados que são necessários para o problema **(dados de entrada)**
- Determinar o procedimento para a resolução do problema  
**(processamento)**



Como os programas são criados?



**Exemplo de programa para realizar a soma de dois números**

**Saída de dados:**

Exibir na tela do computador o resultado da soma de dois números

# Como os programas são criados?



## **Exemplo de programa para realizar a soma de dois números**

### **Entrada de dados:**

Dois números digitados pelo usuário

## Exemplo de programa para realizar a soma de dois números

**Processamento** necessário (considerando as sequências de um programa):

*(Um programa de computador irá executar cada linha como uma instrução por vez!)*

1. Receber o primeiro número do usuário (num\_1)
2. Receber o segundo número do usuário (num\_2)
3. Somar num\_1 e num\_2
4. Exibir o resultado da soma na tela do computador.

## Exemplo de programa para realizar a soma de dois números

**Processamento** necessário (considerando as sequências de um programa):

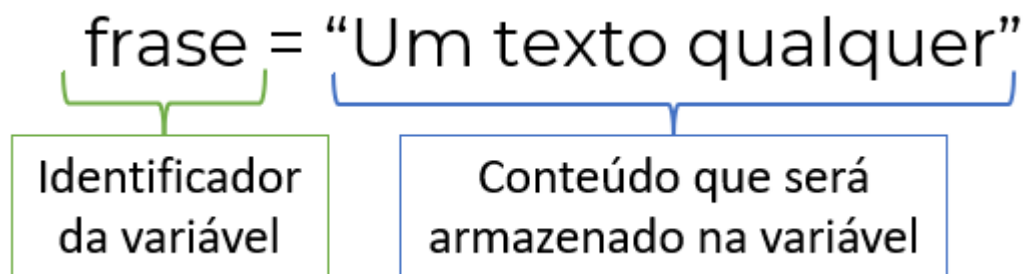
*(Um programa de computador irá executar cada linha como uma instrução por vez!)*

- |  |   |                  |
|--|---|------------------|
| 1. Receber o primeiro número do usuário (num_1)      | } | Entrada de dados |
| 2. Receber o segundo número do usuário (num_2)       |   |                  |
| 3. Somar num_1 e num_2                               |   | Processamento    |
| 4. Exibir o resultado da soma na tela do computador. |   | Saída            |

# Aula 02 – Variáveis

Prática em Python

Para armazenar valores na memória do computador podemos usar **variáveis**.



*Exemplo de atribuição de variáveis*

*Fonte: Adaptado do Microsoft Power Point (Autor)*

Fazer exemplo prático no Colab!

Variáveis podem assumir diferentes tipos, os básicos são:

- **Inteiro (int):** Armazena números inteiros (Exemplos: 1; 2; 3; etc);
- **Real (float):** Armazena números com casas decimais (Exemplos: 1,1; 10,5; 77,3; etc);
- **Lógico (bool):** Armazena apenas dois tipos diferentes de valor: **VERDADEIRO** ou **FALSO**;
- **String (str):** Em python pode armazenar um ou mais conjunto de caracteres.

Variáveis podem assumir diferentes tipos, os básicos são:

- **Inteiro (int)**: Armazena números inteiros (Exemplos: 1; 2; 3; etc);
- **Real (float)**: Armazena números com casas decimais (Exemplos: 1,1; 10,5; 77,3; etc);
- **Lógico (bool)**: Armazena apenas dois tipos diferentes de valor: **VERDADEIRO** ou **FALSO**;
- **String (str)**: Em python pode armazenar um ou mais conjunto de caracteres.

**Fazer exemplo prático no Colab!**



### Operadores Matemáticos:

Operação	Operador
Adição	+
Subtração	-
Multiplicação	*
Divisão	/

Operadores Matemáticos:

Operação	Operador
Adição	+
Subtração	-
Multiplicação	*
Divisão	/

**Fazer exemplo prático no Colab!**

# Aula 03 – Estruturas Condicionais



Operadores  
Relacionais

Os operadores relacionais são usados para fazer comparações entre valores.

Seu resultado sempre será uma variável do tipo lógica (True ou False)

Os operadores relacionais são:

Operador	Símbolo
Igualdade	==
Diferente	!=
Maior	>
Maior ou igual	>=
Menor	<
Menor ou igual	<=

**Exemplo:**

**10 == 6 → False**

Os operadores relacionais são usados para fazer comparações entre valores.

Seu resultado sempre será uma variável do tipo lógica (True ou False)

Os operadores relacionais são:

Operador	Símbolo
Igualdade	==
Diferente	!=
Maior	>
Maior ou igual	>=
Menor	<
Menor ou igual	<=

**Fazer exemplo prático!**



# Operadores Lógicos

Os operadores lógicos também são usados para fazer comparações. Entretanto, eles fazem comparações entre duas variáveis lógicas. Os operadores lógicos são:

Operador	Instrução
E	and
Ou	or
Não	not



O operador **and** terá a sua saída igual a **True** apenas quando as todas as suas entradas que estão sendo comparadas forem igual a **True**

Exemplo: entrada\_1 **and** entrada\_2

entrada_1	entrada_2	Saída
False	False	False
False	True	False
True	False	False
True	True	True

**Fazer exemplo prático!**

O operador **or** terá a sua saída igual a **True** apenas quando uma das suas entradas que estão sendo comparadas forem igual a **True**

Exemplo: entrada\_1 **or** entrada\_2

entrada_1	entrada_2	Saída
False	False	False
False	True	True
True	False	True
True	True	True

**Fazer exemplo prático!**

O operador **not** é aplicado a apenas uma entrada. Ele sempre irá inverter o valor da entrada, caso essa entrada seja igual a **True**, a saída será **False** e vice-versa

Exemplo: **not** entrada

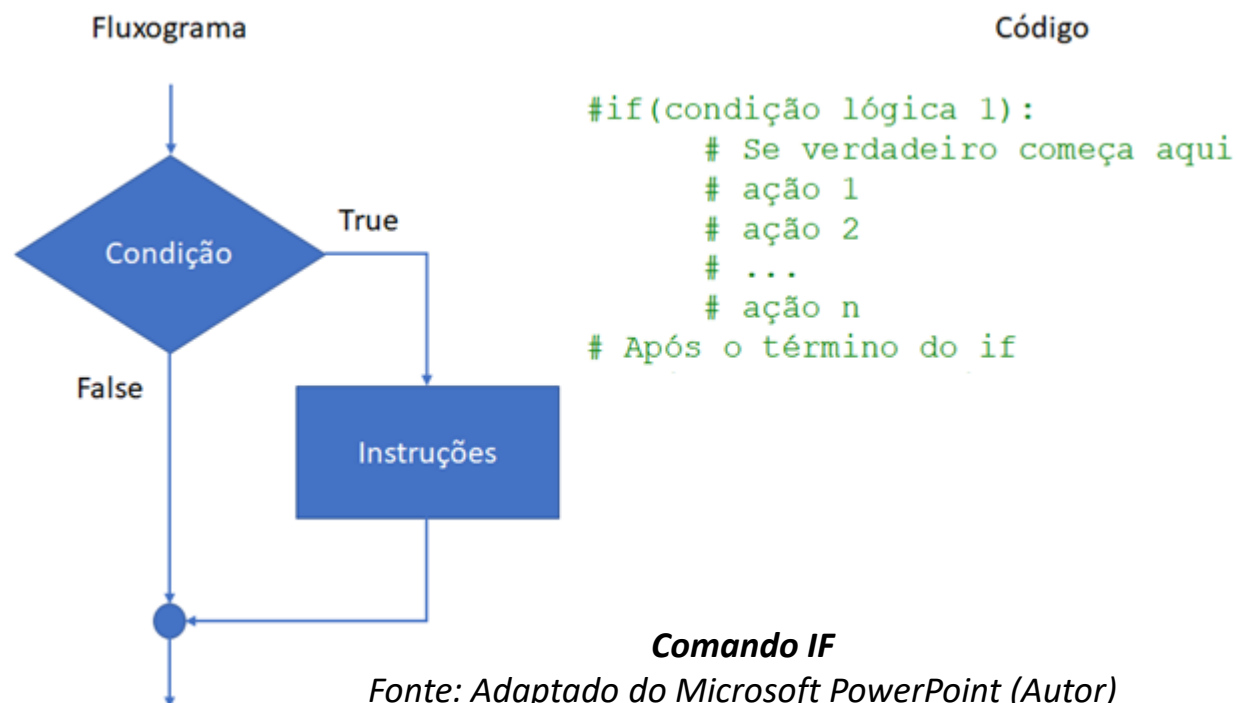
entrada	Saída
False	True
True	False

Fazer exemplo prático!



# Estruturas Condicionais

A estrutura de seleção simples (*if*), temos uma condição lógica que será avaliada. Caso o resultado for *True* um conjunto de instruções serão executados. Do contrário, não será executado esse conjunto de instruções



Caso o if for verdadeiro, tudo que está com uma indentação será executado

Fim do if: Quando os códigos não tiverem uma indentação

```
#if(condição lógica 1):  
→ # Se verdadeiro começa aqui  
→ # ação 1  
→ # ação 2  
→ # ...  
→ # ação n  
# instrução 1  
# instrução 2
```

Indica o início do if

**Exemplo de programação (IF)**

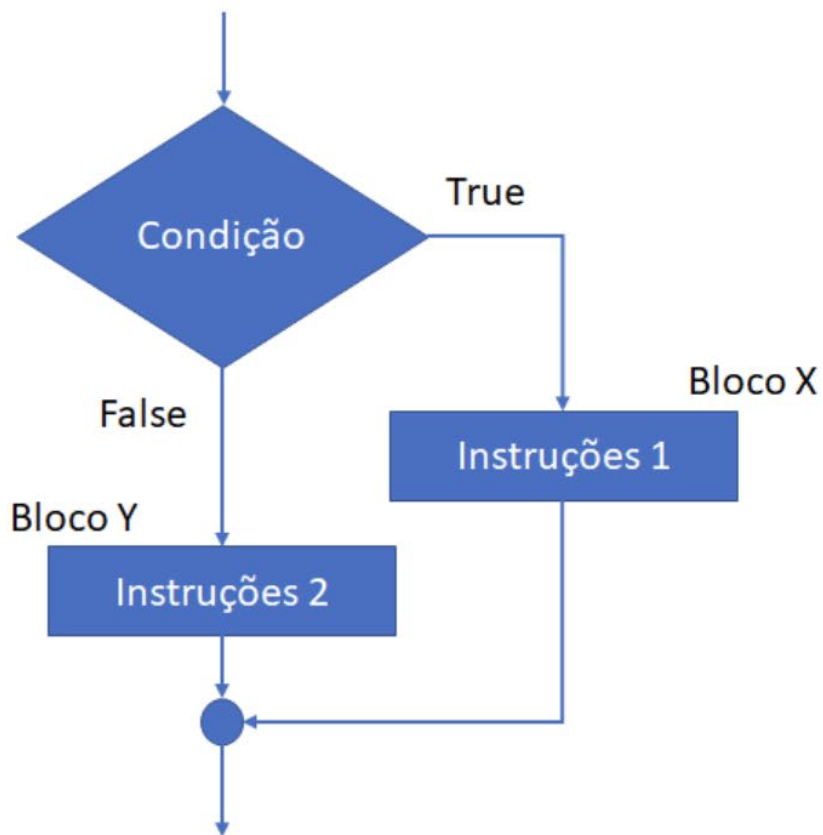
Fonte: Adaptado do Microsoft PowerPoint (Autor)

**Fazer exemplo prático!**



Estruturas  
Condicionais Compostas

Fluxograma



Código

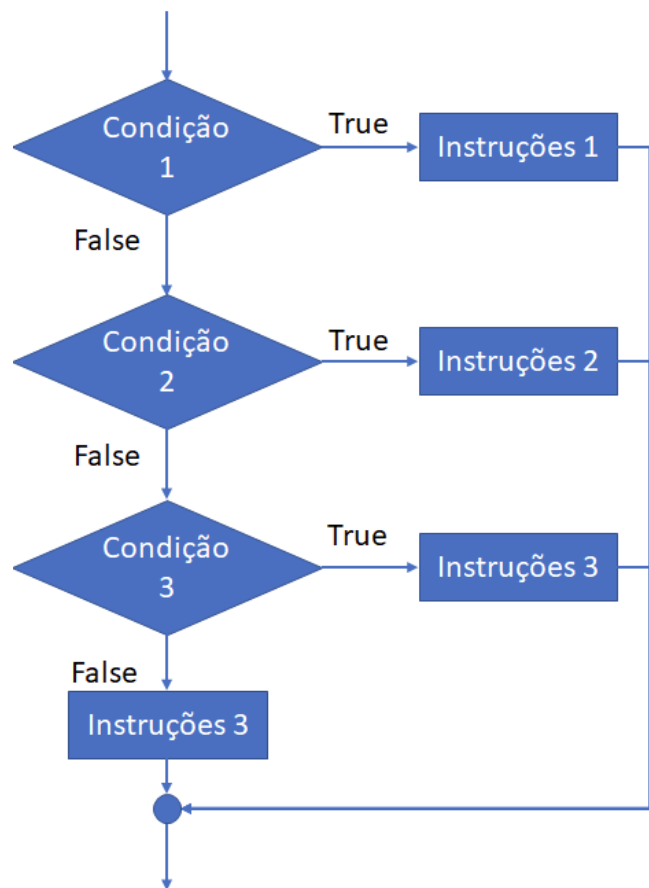
```
#if(condição lógica 1):  
    # Se verdadeiro começa aqui  
    # ação v1  
    # ação v2  
    # ...  
    # ação vn  
#else:  
    # Se a condição não for verdadeira:  
    # ação f1  
    # ação f2  
    # ...  
    # ação fn  
# Após o término do if ou após as  
# n instruções continua daqui
```





# Estruturas Condicionais Encadeadas

Fluxograma



Código

```
#if(condição lógica 1):  
    # Se verdadeiro começa aqui  
    # ação 1  
    # ...  
    # ação n  
#elif(condição lógica 2):  
    # Se a condição lógica 1 não é verdadeira, mas  
    # a condição lógica 2 é:  
    # ação 1  
    # ...  
    # ação n  
#elif(condição lógica 3):  
    # Se a condição lógica 3 não é verdadeira, mas  
    # a condição lógica 3 é:  
    # ação 1  
    # ...  
    # ação n  
#else:  
    # Se nenhuma outra for verdadeira:  
    # ação 1  
    # ...  
    # ação n
```

# Aula 04 – Estruturas de Repetição

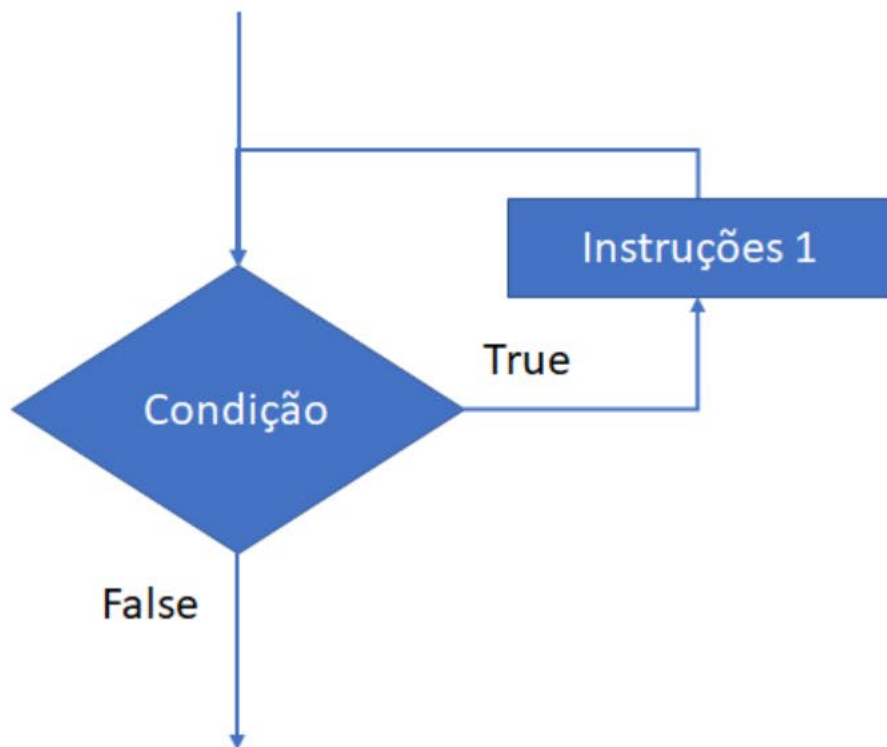


Comando *while*

- Quando desejamos repetir um conjunto de instruções repetidas vezes podemos usar as estruturas de repetição para nos auxiliar
- Essas estruturas devem avaliar uma condição para verificar se devemos continuar repetindo a mesma.

# Comando while

## Fluxograma laço de repetição



```
#while (condição lógica):  
# Enquanto verdadeiro repetira:  
# ação v1  
# ação v2  
# ...  
# ação vn  
#termino do laço e continuação do programa
```

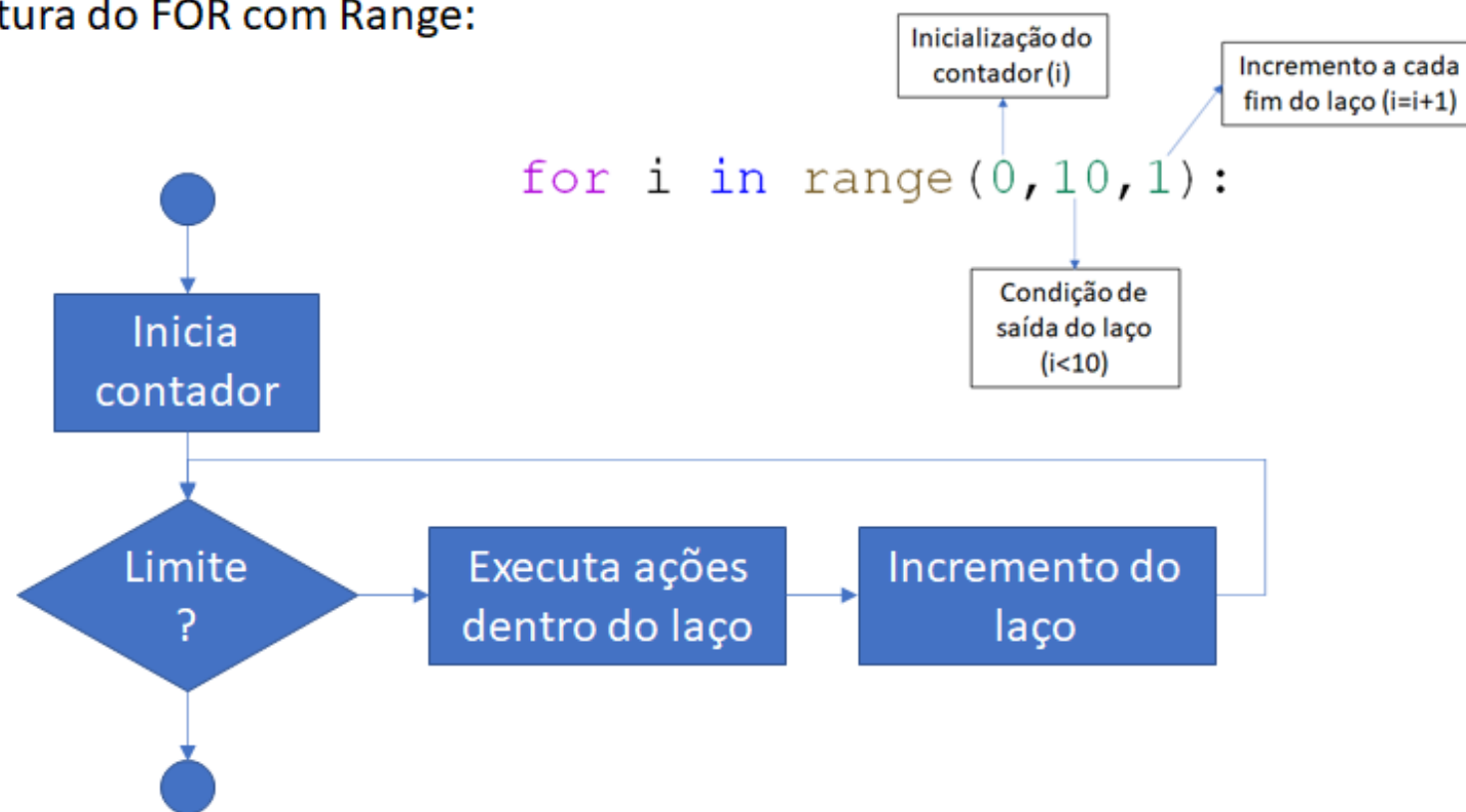
### Comando while

Fonte: Adaptado do Microsoft PowerPoint (Autor)



Comando *for*

Estrutura do FOR com Range:



**Comando for**

Fonte: Adaptado do Microsoft PowerPoint (Autor)



