

# Operadores e Estruturas de Decisão



**Conteudista:** Prof. Me. Hugo Batista Fernandes

**Revisão Textual:** Prof.<sup>a</sup> Dra. Luciene Oliveira da Costa Granadeiro

## Objetivos da Unidade:

- Estudar os conceitos e a sintaxe das estruturas de decisão no *Python*, bem como exemplos de suas aplicações;
- Explorar os conceitos e a utilização dos operadores aritméticos e suas precedências, operadores relacionais, lógicos e de concatenação.

Material Teórico

Material Complementar

Referências

# Material Teórico

---

## Operadores

Um operador é um símbolo que informa ao programa, quais manipulações matemáticas ou lógicas o programa deve executar. A linguagem *Python* é rica em operadores e fornece o seguinte tipo de operadores: operadores aritméticos; relacionais; lógicos e de concatenação:

### Operadores Aritméticos

Operadores aritméticos são utilizados para elaboração e execução de cálculos matemáticos. Em *Python*, temos os seguintes operadores:

Quadro 1

Operador	Descrição	Exemplo
+	Operador de adição	<code>MinhaVariavel = 2 + 4</code>
-	Operador de subtração	<code>MinhaVariavel = 2 - 4</code>

Operador	Descrição	Exemplo
*	Operador de multiplicação	MinhaVariavel = 2 x 4
/	Operador de divisão	MinhaVariavel = 4 / 2
%	Operador Módulo. Calcula o resto da divisão	Minha Variavel = 8%2
**	Operador de exponenciação	MinhaVariavel = 2**4
//	Operador divisão de números inteiros. Resulta somente na parte inteira da divisão	MinhaVariavel = 9//2

Vejamos a seguir alguns exemplos de utilização desses operadores.

```
1 #atribuindo valores para variáveis a e b
2 a = 2
3 b = 3
4
5 #operação de adição
6 resultado = a + b
7 print(resultado)
8
9 #operação de subtração
10 resultado = a - b
11 print(resultado)
12
13 #operação de multiplicação
14 resultado = a * b
15 print(resultado)
16
17 #operação de divisão
18 resultado = a / b
19 print(resultado)
20
21 #operação de exponenciação
22 resultado = a ** b
23 print(resultado)
24
25 #operação de divisão de inteiros
26 resultado = a // b
27 print(resultado)
28
```

**Figura 1 – Código exemplo 1**

Fonte: Acervo do Conteudista

*Site*

*Jdoodle –Online Python 3 IDE*

O código acima pode ser acessado e testado a seguir.

Clique no botão para conferir o conteúdo.

ACESSE

## Explicando o Código

- **Linha 2:** declaramos uma variável com o nome de “a” e atribuímos, por meio do sinal de “igual” o valor “2”;
- **Linha 3:** declaramos uma variável com o nome de “b” e atribuímos, por meio do sinal de “igual” o valor “3”;
- **Linha 6:** efetuamos a operação aritmética de soma entre os termos representados pelas variáveis a e b, em seguida, é atribuído o resultado dessa operação à variável “resultado”;
- **Linha 7:** utilizamos a função “*print*” para imprimir na tela o valor contido na variável “resultado” (5);
- **Linha 10:** efetuamos a operação aritmética de subtração entre os termos representados pelas variáveis a e b, em seguida, é atribuído o resultado dessa operação à variável “resultado”;
- **Linha 11:** utilizamos a função “*print*” para imprimir na tela o valor contido na variável “resultado” (-1);

- **Linha 14:** efetuamos a operação aritmética de multiplicação entre os termos representados pelas variáveis a e b, em seguida, é atribuído o resultado dessa operação à variável “resultado”;
- **Linha 15:** utilizamos a função “*print*” para imprimir na tela o valor contido na variável “resultado” (6);
- **Linha 18:** efetuamos a operação aritmética de divisão entre os termos representados pelas variáveis a e b, em seguida, é atribuído o resultado dessa operação à variável “resultado”;
- **Linha 19:** utilizamos a função “*print*” para imprimir na tela o valor contido na variável “resultado” (0,6666);
- **Linha 22:** efetuamos a operação aritmética de exponenciação entre os termos representados pelas variáveis a e b, em seguida, é atribuído o resultado dessa operação à variável “resultado”;
- **Linha 23:** utilizamos a função “*print*” para imprimir na tela o valor contido na variável “resultado” (8);
- **Linha 26:** efetuamos a operação aritmética de divisão de inteiros entre os termos representados pelas variáveis a e b, em seguida, é atribuído o resultado dessa operação à variável “resultado”;
- **Linha 27:** utilizamos a função “*print*” para imprimir na tela o valor contido na variável “resultado” (0).

## Precedência de Operadores Aritméticos

Como na matemática, em *Python*, uma expressão numérica é avaliada de acordo com a ordem dos operadores aritméticos, essa regra é chamada de precedência. A precedência do operador determina o agrupamento de termos em uma expressão e afeta a forma como uma expressão é avaliada. A prioridade de avaliação da expressão

está ligada diretamente à precedência do operador. É válido ressaltar que a utilização de parênteses é empregada para a alteração da precedência de uma expressão.

Abaixo temos um exemplo de precedência entre operadores.

Quadro 2

Ordem de Prioridade	Operação	Símbolo
1º	Parênteses	( )
2º	Inversão de sinal	-
3º	Exponenciação	**
4º	Multiplicação, Divisão, Resto da divisão (módulo)	* , /, %, //
5º	Adição e Subtração	+ , -

Em uma expressão com operadores da mesma prioridade, as operações serão executadas da esquerda para a direita.

De acordo com o exposto, podemos destacar, por exemplo, um cenário como:  $a = 8 + 1 * 9$ .

Seguindo a ordem de precedência, a variável “a” resulta em 17, e não 81, pois o operador \* tem precedência **mais alta** do que +, assim, efetua-se primeiro a multiplicação  $1 * 9$  e, depois, a adição por 8.

Outro exemplo. Considere o seguinte código:

```
a = 20  
b = 10  
c = 15  
d = 5  
e = (a + b) * c / d
```

Para resolver essa expressão, é preciso primeiro efetuar o cálculo da expressão entre parênteses, em seguida, a multiplicação e, por fim, a divisão. Seguindo essa ordem, temos:

```
e = (20+10) * 15 / 5  
e = 30 * 15 / 5  
e = 450 / 5  
e = 90
```

## Operadores Relacionais

Operadores relacionais são utilizados para comparar valores entre termos. O resultado dessa comparação sempre irá retornar um valor booleano, ou seja, *true* ou *false*. Em Python, temos os seguintes operadores.

Tabela 1

Operador	Nome	Descrição	Exemplo
<code>==</code>	Igualdade	Verifica se os valores de dois operandos são iguais ou não, se a resposta for <b>sim</b> , a condição torna-se verdadeira ( <i>true</i> ).	<code>if(a==b):</code> <code>print("a é igual a b")</code>
<code>!=</code>	Diferente	Verifica se os valores de dois operandos são iguais ou <b>não</b> , se a resposta for <b>não</b> , a condição torna-se verdadeira ( <i>true</i> ).	<code>if(a!=b):</code> <code>print("a não é igual a b")</code>
<code>&gt;</code>	Maior que	Verifica se o valor do operando esquerdo é maior que o valor do operando à direita, se <b>sim</b> , a condição torna-se verdadeira ( <i>true</i> ).	<code>if(a&gt;b):</code> <code>print("a é maior que b")</code>

Operador	Nome	Descrição	Exemplo
<	Menor que	Verifica se o valor do operando esquerdo é menor que o valor do operando à direita, se sim, a condição torna-se verdadeira ( <i>true</i> ).	<i>if(a&gt;b):</i> <i>print("a é menor que b")</i>
$\geq$	Maior igual	Verifica se o valor do operando à esquerda é maior ou igual ao valor do operando à direita, se sim, a condição torna-se verdadeira ( <i>true</i> ).	<i>if(a&gt;=b):</i> <i>print("a é maior ou igual que b")</i>
$\leq$	Menor igual	Verifica se o valor do operando à esquerda é menor ou igual ao valor do operando à direita, se sim, a condição torna-	<i>if(a&gt;=b):</i> <i>print("a é menor ou igual que b")</i>

Operador	Nome	Descrição	Exemplo
		se verdadeira ( <i>true</i> ).	

---

## Site

[Jdoodle – Online Python 3 IDE](#)

**Podemos verificar esses operadores em execução com o código criado e compartilhado por meio do seguinte *link*:**

Clique no botão para conferir o conteúdo.

ACESSE

## Operadores Lógicos

Um operador lógico é um operador que retorna um resultado booleano (*true* ou *false*) baseado no resultado booleano de uma ou duas outras expressões. O conceito de operadores lógicos é simples. Eles permitem que um programa tome uma decisão com base em múltiplas condições. Cada operando é considerado uma condição que pode ser avaliada de acordo com o valor (verdadeiro ou falso).

Os operadores lógicos em *Python* são *and*, *or* e *not*.

Quadro 3

Operador	Descrição	Exemplo
<i>and</i>	Retorna verdadeiro se ambas as expressões resultarem como verdadeira. Todas as expressões são avaliadas antes que o operador <i>and</i> seja aplicado.	<i>if(a&gt;=b)</i> <i>and (c&gt;d):</i> <i>print("a é maior ou igual que b e c é maior que d")</i>
<i>or</i>	Retorna verdadeiro se pelo menos uma das expressões resultarem como verdadeira. Caso a primeira expressão retorne como verdadeiro, o restante das expressões não é avaliado.	<i>if(a&gt;=b) or (c&gt;d):</i> <i>print("a é maior ou igual que b ou c é maior que d")</i>
<i>not</i>	Retorna verdadeiro se a expressão à direita for avaliada como falsa. Retorna falso se a expressão à direita for verdadeiro.	<i>a,b=10,5</i> <i>if not (a&lt;b):</i> <i>print("a é menor que b")</i>

## Site

[Jdoodle – Online Python 2 IDE](#)

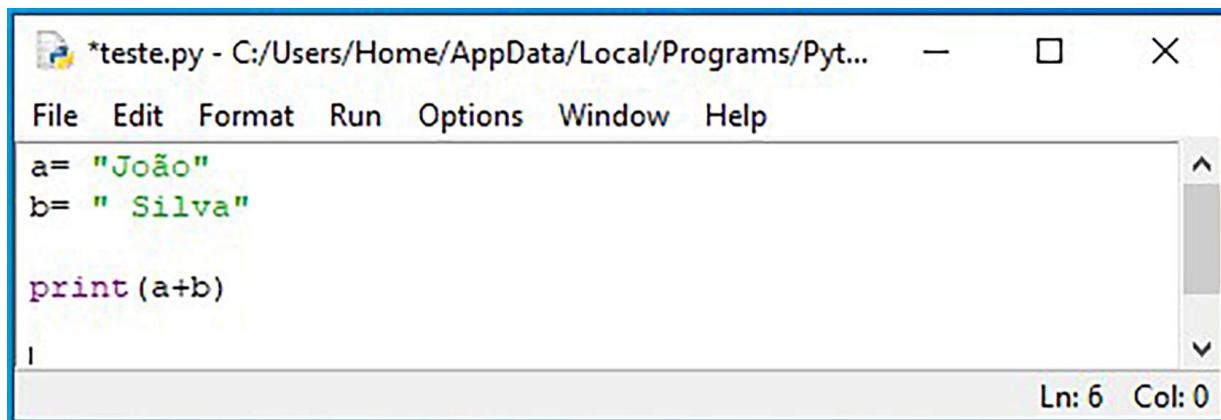
Podemos verificar esses operadores em execução com o código criado e compartilhado por meio do seguinte *link*:

Clique no botão para conferir o conteúdo.

ACESSE

## Operadores de Concatenação

O operador de concatenação de *string* em Python é representado por dois pontos ("+"). Utiliza-se esse operador quando é necessária a junção de dois operandos que contenham um texto.



The screenshot shows a window titled '\*teste.py - C:/Users/Home/AppData/Local/Programs/Pyt...'. The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code editor contains the following Python code:

```
a= "João"
b= " Silva"

print(a+b)
```

The status bar at the bottom right indicates 'Ln: 6 Col: 0'.

**Figura 2**

Fonte: Acervo do Conteudista

The screenshot shows the Python IDLE Shell 3.10.1 interface. The title bar reads "IDLE Shell 3.10.1". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays a command-line session:

```
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec  6 2021, 19:10:37) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> ===== RESTART: C:/Users/Home/AppData/Local/Programs/Python/Python310/teste.py ====
>>> João Silva
>>> |
```

The status bar at the bottom right indicates "Ln: 6 Col: 0".

**Figura 3**  
Acervo do Conteudista

## Leitura

### Operadores e Expressões em Python

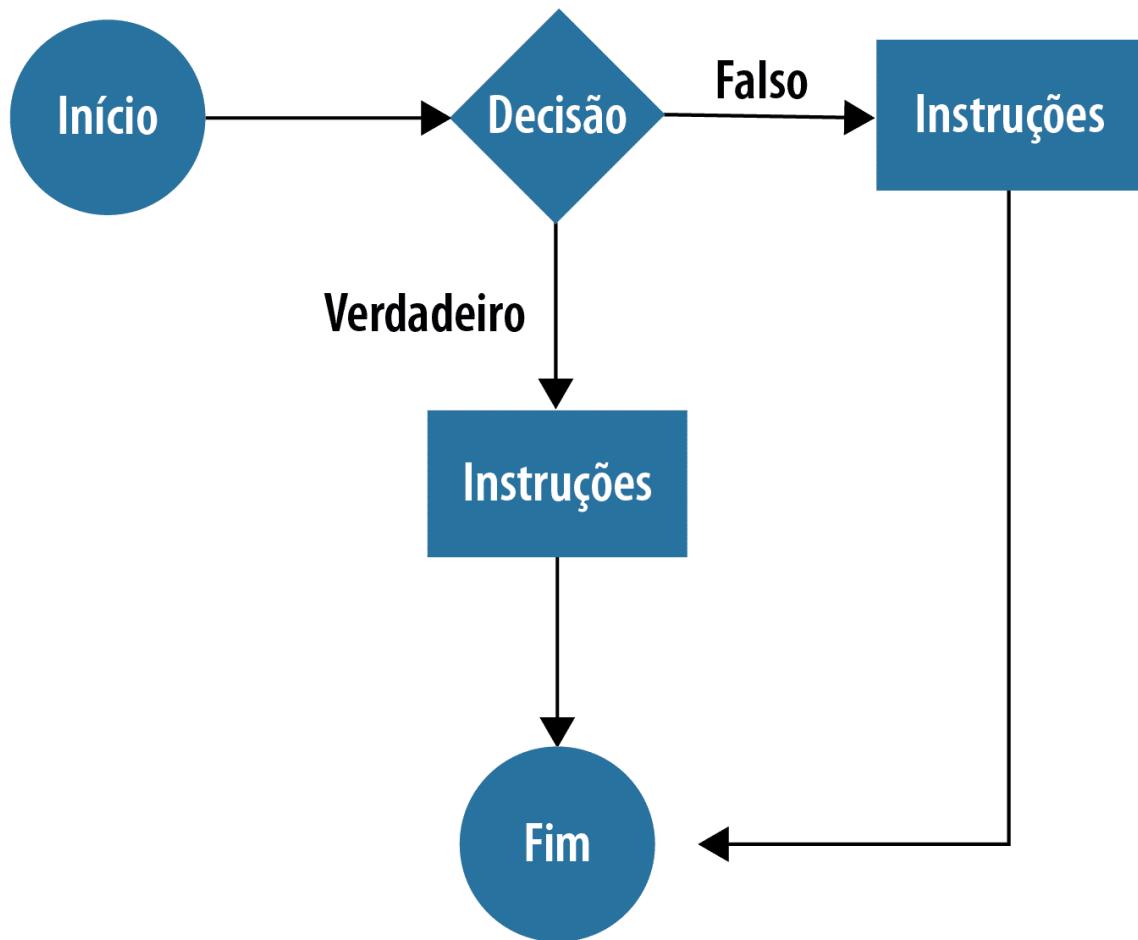
Clique no botão para conferir o conteúdo.

ACESSE

## Estrutura de Decisão

Em muitos momentos, nosso aplicativo deve tomar decisões de acordo com as condições pré-estabelecidas. As estruturas de decisão exigem que o programador

especifique uma ou mais condições a serem avaliadas ou testadas pelo programa, juntamente com uma declaração ou instruções a serem executadas.



**Figura 4**

A Figura acima representa um fluxograma onde temos uma estrutura de decisão, e de acordo com a resposta do teste da **condição, falso ou verdadeiro**, o programa executa um bloco de instruções e segue para a finalização do algoritmo.

## Leitura

### Explicando Algoritmos e Fluxogramas com Exemplos

Clique no botão para conferir o conteúdo.

ACESSE

A linguagem *Python* fornece os seguintes tipos de declarações de tomada de decisão: *if ...; if ... else* e *if... elif*.

#### Estrutura *if ...*

A estrutura *if...* é utilizada para testar uma condição e caso retorne verdadeiro, executa um bloco de instruções. Toda declaração *if* segue a sintaxe padrão: teste uma condição e, se for verdade, execute uma ação.

*if*(condição):  
    ↔ bloco de instruções  
        ↑

Logo abaixo da primeira linha do *if*, deve-se adicionar um “TAB” como indentação para assim, iniciar e seguir com o bloco de instruções.

## Figura 5

---

---

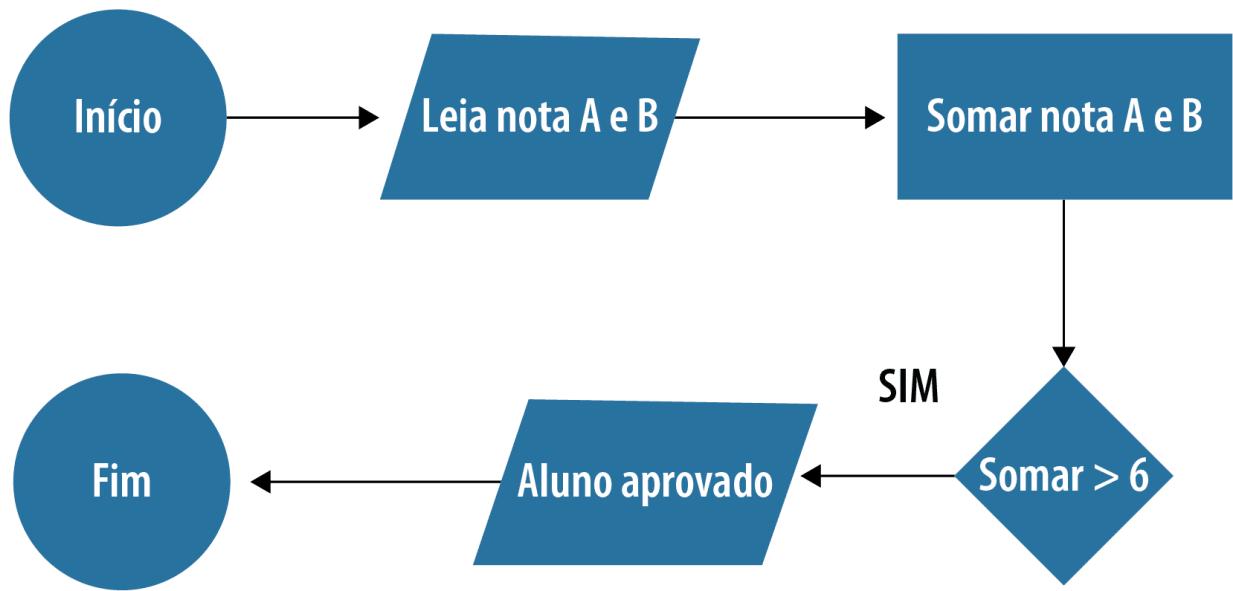
### Leitura

#### Indentação do Código Python

Clique no botão para conferir o conteúdo.

ACESSE

Vejamos o seguinte cenário. Um programa deve obter as notas A e B de um aluno, somar as notas e em seguida, verificar se a nota do aluno é maior que seis (6), se sim, o programa exibe na tela a mensagem “Aluno aprovado”. A seguir, temos um fluxograma descrevendo esse algoritmo.



**Figura 6**

## Leitura

**Desvio Condicional Simples (SE...ENTÃO)**

Clique no botão para conferir o conteúdo.

**ACESSE**

Qual é a condição para que o aluno possa ser aprovado?

A condição é que a soma das notas A e B deve ser maior que seis (6), caso o teste dessa condição retorne verdadeiro, o programa executa as instruções previstas para essa situação.

Implementando o seguinte algoritmo em *Python*, temos:

```
1  a=3
2  b=4
3  soma = a+b
4  if (soma > 6):
5      print("Aluno aprovado")
6
7
```

**Figura 7**

Fonte: Acervo do Conteudista

## Site

*Jdoodle – Online Python 2 IDE*

Clique no botão para conferir o conteúdo.

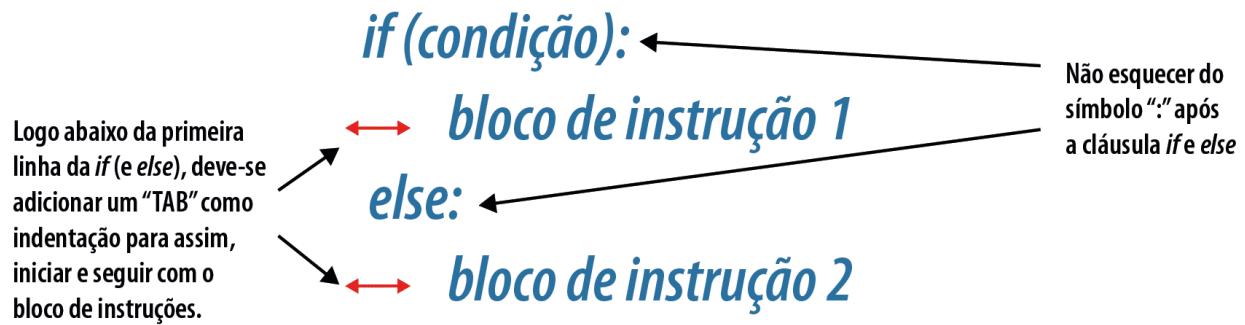
## Explicando o Código

- **Linhas 1 e 2:** declaramos uma variável com o nome de “a” e “b” e em seguida, atribuímos os valores 3 para “a” e 4 para “b”;
- **Linha 3:** efetuamos uma operação de adição entre as variáveis “a” e “b” e atribuímos o resultado para a variável “soma”;
- **Linha 4:** descrevemos a estrutura condicional *if* testando a condição “soma > 6”. Caso o teste retorne verdadeiro, o programa executará a linha 5, do contrário, o programa será finalizado;
- **Linha 5:** utilizamos a função “*print*” para imprimir na tela o texto “Aluno aprovado”.

### Estrutura *if ... else*

Na estrutura *if ... else*, além de testar uma condição e executar um bloco de instruções no caso de retorno do teste ser verdadeiro, também é descrito as instruções para quando o retorno for falso. Utilizamos essa estrutura quando o programa deve tomar duas decisões.

Toda declaração *if ... else*. Segue sintaxe padrão: teste uma condição e, se for verdade, execute o bloco de instrução 1; caso contrário, execute o bloco de instrução 2.



**Figura 8**

---

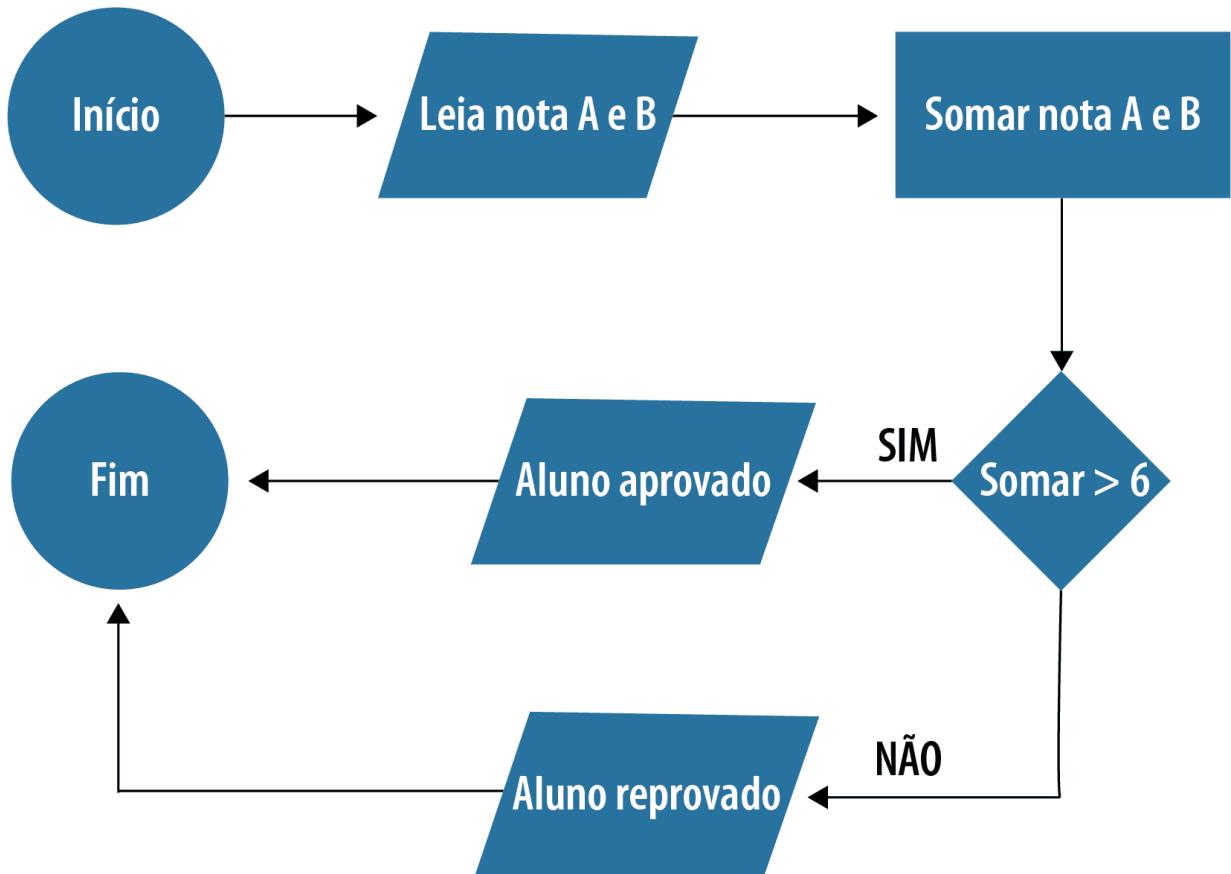
## Leitura

### Estrutura de Decisão SE-ENTÃO-SENÃO

Clique no botão para conferir o conteúdo.

ACESSE

Partindo do exemplo anterior, onde nosso programa recebe duas notas e, em seguida. efetua a soma entre as duas, iremos complementar com mais uma decisão. Caso a condição retorne falsa, o programa irá imprimir na tela o texto “Aluno reprovado”. A seguir, temos um fluxograma descrevendo esse algoritmo.



**Figura 9**

---

Do mesmo modo que o algoritmo anterior, a condição para o aluno ser aprovado é possuir uma nota maior que seis (6). Contudo, nesse novo algoritmo, temos previsto também instruções em caso de um retorno ser falso, ou seja, caso o aluno possua nota menor ou igual a seis (6).

Implementando o seguinte algoritmo em *Python*, temos:

```
1 a=3
2 b=2
3 soma = a+b
4
5 if (soma > 6):
6     print("Aluno aprovado")
7 else:
8     print("Aluno reprovado")
9
```

**Figura 10**

Fonte: Acervo do Conteudista

## *Site*

*Jdoodle – Online Python IDE*

Podemos visualizar esse código por meio do seguinte link:

Clique no botão para conferir o conteúdo.

ACESSE

# Explicando o Código

- **Linhas 1 e 2:** declaramos uma variável com o nome de “a” e “b” e em seguida, atribuímos os valores 3 para “a” e 2 para “b”;
- **Linha 3:** efetuamos uma operação de adição entre as variáveis “a” e “b” e atribuímos o resultado para a variável “soma”;
- **Linha 5:** descrevemos a estrutura condicional *if* testando a condição “soma > 6”. Caso o teste retorne verdadeiro, o programa executará a linha 6. Caso contrário, o programa executará a linha 8;
- **Linha 6:** utilizamos a função “*print*” para imprimir na tela o texto “Aluno aprovado”;
- **Linha 8:** utilizamos a função “*print*” para imprimir na tela o texto “Aluno reprovado”.

## Estrutura *if ... elif*

Utilizamos a estrutura *if ... elif* quando o programa deve testar uma condição quando a primeira condição testada retorna falso.

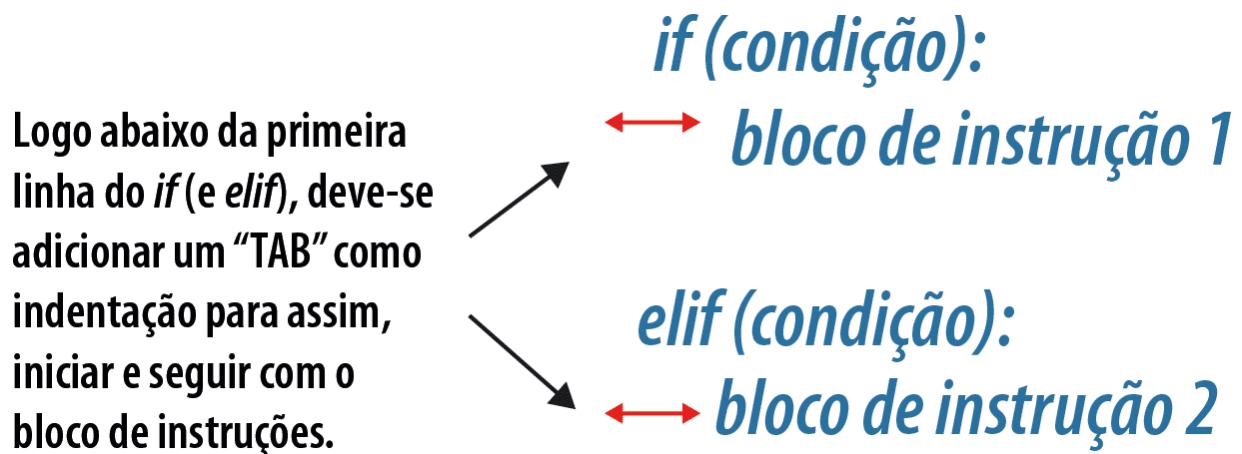


Figura 11

## Leitura

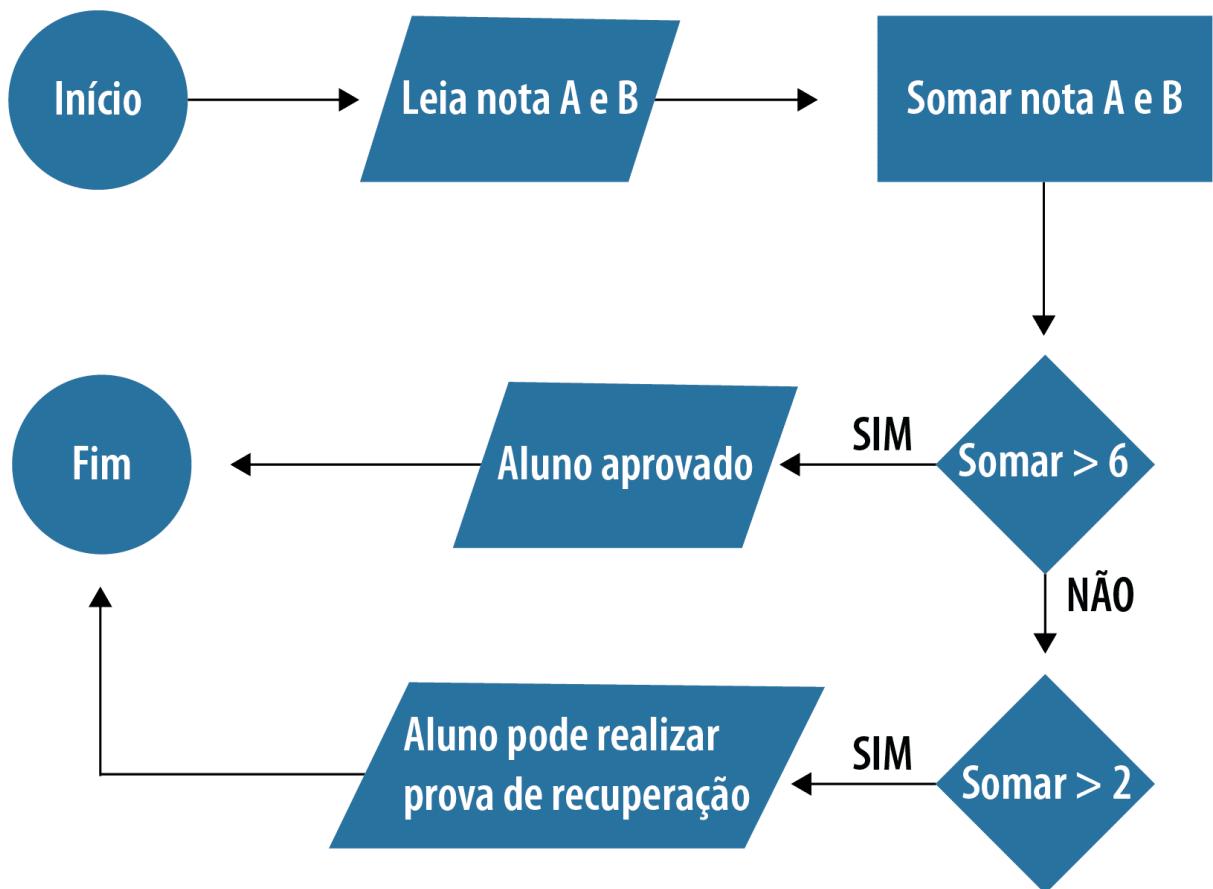
### Desvio Condicional Aninhado (SE...ENTÃO...SENÃO...SE)

Clique no botão para conferir o conteúdo.

ACESSE

Vejamos o seguinte cenário. Um programa deve obter as notas A e B de um aluno, somar as notas e, em seguida, verificar se a nota do aluno é maior que seis (6), se sim, o programa exibe na tela a mensagem “Aluno aprovado”. Caso o aluno não possua uma nota maior que seis (6), porém, possua uma nota maior que dois (2), o aluno poderá

realizar uma prova de recuperação. A seguir, temos um fluxograma descrevendo esse algoritmo.



**Figura 12**

Implementando o seguinte algoritmo em *Python*, temos:

```
1 a=1
2 b=3
3 soma = a+b
4
5 if (soma > 6):
6     print("Aluno aprovado")
7 elif (soma > 2):
8     print("Aluno pode realizar prova de recuperação")
9
```

**Figura 13**

Fonte: Acervo do Conteudista

## *Site*

[Jdoodle – Online Python IDE](#)

Podemos visualizar esse código por meio do seguinte link:

Clique no botão para conferir o conteúdo.

ACESSE

## Explicando o Código

- **Linhas 1 e 2:** declaramos uma variável com o nome de “a” e “b” e, em seguida, atribuímos os valores 1 para “a” e 3 para “b”;
- **Linha 3:** efetuamos uma operação de adição entre as variáveis “a” e “b” e atribuímos o resultado para a variável “soma”;
- **Linha 5:** descrevemos a estrutura condicional *if* testando a condição “soma > 6”. Caso o teste retorne verdadeiro, o programa executará a linha 6. Caso contrário, o programa executará a linha 7;
- **Linha 6:** utilizamos a função “*print*” para imprimir na tela o texto “Aluno aprovado”;
- **Linha 7:** testamos a condição “soma > 2”. Caso o teste retorne verdadeiro, o programa executará a linha 8. Caso contrário, o programa será finalizado;
- **Linha 8:** utilizamos a função “*print*” para imprimir na tela o texto “Aluno pode realizar prova de recuperação”.

## Estruturas de Decisão Encadeadas

Em alguns cenários de nossos algoritmos, devem-se realizar diversos testes e condições para se atingir um objetivo. Em cenários onde é preciso verificar diversas condições, podemos escrever sequências de instruções de estruturas de decisão. Esse cenário comumente é chamado de estruturas de decisão aninhadas ou encadeadas.

Abaixo, temos exemplos de estruturas de decisão encadeadas.

```
if (condição):  
    bloco de instrução 1
```

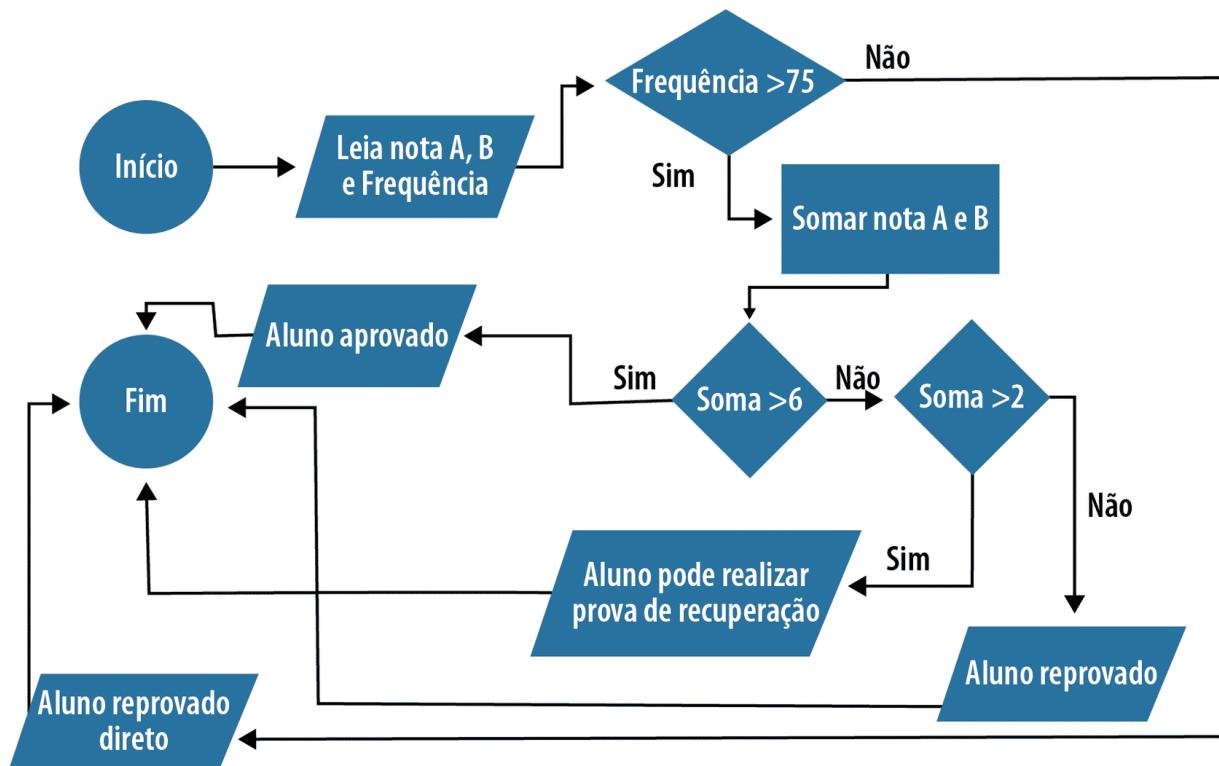
```
elif(condição):
    bloco de instrução 2
else:
    bloco de instrução 3
else:
    bloco de instrução 4
```

```
if(condição):
    if(condição):
        bloco de instrução 1
    else:
        if(condição):
            bloco de instrução 2
        else:
            bloco de instrução 3
    else:
        bloco de instrução 4
```

Vejamos o seguinte cenário. Um programa deve obter as notas A e B e a frequência de um aluno. Para ser considerado aprovado, o aluno deve possuir frequência maior que 75% e nota maior que 6.

Caso o aluno não possua frequência maior que 75%, será considerado reprovado de forma direta.

O aluno que possuir a frequência mínima, porém, nota menor ou igual a 6 e maior que 2, poderá realizar uma prova de recuperação, caso contrário, será reprovado. A seguir, temos um fluxograma descrevendo esse algoritmo.



**Figura 14**

Implementando o seguinte algoritmo em Python, temos:

```

1  a=1
2  b=5
3  frequencia = 80
4
5  if (frequencia > 75):
6      soma = a+b
7      if(soma > 6):
8          print("Aluno aprovado")
9      elif(soma > 2):
10         print("Aluno pode realizar prova de recuperação")
11     else:
12         print("Aluno reprovado")
13 else:
14     print("Aluno reprovado direto")
15

```

## Figura 15

Fonte: Acervo do Conteudista

---

The screenshot shows a Jupyter Notebook interface. At the top, there's a toolbar with various icons. Below it is a code cell containing Python code. The output area shows the results of the code execution, including numerical values and text messages. A horizontal bar at the bottom indicates the progress of the code execution.

**Site**

**Jdoodle – Online Python IDE**

**Podemos visualizar esse código por meio do seguinte link:**

Clique no botão para conferir o conteúdo.

**ACESSE**

## Explicando o Código

- **Linhas 1 e 2:** declaramos uma variável com o nome de “a” e “b” e, em seguida, atribuímos os valores 1 para “a” e 5 para “b”;
- **Linha 5:** descrevemos a estrutura condicional *if* testando a condição “frequencia > 75”. Caso o teste retorne verdadeiro, o programa executará o bloco a partir da linha 6. Caso contrário, o programa executará o bloco de instruções a partir da linha 13 e finalizará a execução do programa;

- **Linha 6:** efetuamos uma operação de adição entre as variáveis “a” e “b” e atribuímos o resultado para a variável “soma”;
- **Linha 7:** descrevemos a estrutura condicional *if* testando a condição “soma > 6”. Caso o teste retorne verdadeiro, o programa executará a linha 8. Caso contrário, o programa executará a linha 9;
- **Linha 8:** utilizamos a função “*print*” para imprimir na tela o texto “Aluno aprovado”;
- **Linha 9:** testamos a condição “soma > 2”. Caso o teste retorne verdadeiro, o programa executará a linha 10. Caso contrário, o programa irá executar o bloco de instruções a partir da linha 12;
- **Linha 10:** utilizamos a função “*print*” para imprimir na tela o texto “Aluno pode realizar prova de recuperação”;
- **Linha 12:** utilizamos a função “*print*” para imprimir na tela o texto “Aluno reprovado”;
- **Linha 13:** utilizamos a função “*print*” para imprimir na tela o texto “Aluno reprovado direto”.

---

## Em Síntese

Nesta Unidade, estudamos a aplicação de operadores (aritméticos, relacionais, lógicos e de concatenação), bem como os conceitos de estruturas de decisão. É importante que assista à videoaula e que leia

**os livros e materiais complementares indicados nesta unidade de estudo. É fundamental que, além dos estudos em *Python*, busque estudar ou retomar conceitos de desenvolvimento de algoritmos, ter uma boa noção desse tema o ajudará na jornada de estudos de programação de computadores.**

**Até a próxima!**

## Material Complementar

---

Indicações para saber mais sobre os assuntos abordados nesta Unidade:

---

### Vídeos

[Raciocínio Lógico: Introdução a Operadores Lógicos](#)



*Python – Estrutura de Decisão Condicional Aninhada –*  
**SE..ENTÃO..SENÃO SE**



---

## Leitura

### Operadores Aritméticos e Lógicos em Python

Clique no botão para conferir o conteúdo.

ACESSE

### Python If Else: Como Usar essa Estrutura Condicional?

Clique no botão para conferir o conteúdo.

ACESSE

 Referências

---

BANIN, S. L. *Python 3 – Conceitos e Aplicações – Uma abordagem didática*. São Paulo: Érica, 2018. (*e-book*)

PERKOVIC, L. *Introdução à Computação Usando Python – Um Foco no Desenvolvimento de Aplicações*. Rio de Janeiro: LTC, 2016. (*e-book*)

WAZLAWICK, R. *Introdução a Algoritmos e Programação com Python – Uma Abordagem Dirigida por Testes*. Rio de Janeiro: Elsevier, 2017. (*e-book*)