



ICC901 – Introdução à Programação de Computadores
IECo81 – Introdução à Ciência dos Computadores
IECo37 – Introdução à Programação de Computadores

Aula 02 – Estruturas Condicionais Simples e Compostas

Atualização: 6/mar/20



Você tem a liberdade de:



Compartilhar: copiar, distribuir e transmitir esta obra.

Remixar: criar obras derivadas.

Sob as seguintes condições:



Atribuição: você deve creditar a obra da forma especificada pelo autor ou licenciante (mas não de maneira que sugira que estes concedem qualquer aval a você ou ao seu uso da obra).



Uso não comercial: você não pode usar esta obra para fins comerciais.



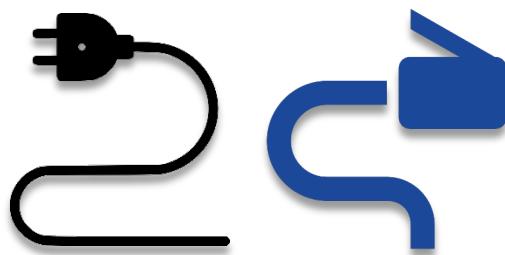
Compartilhamento pela mesma licença: se você alterar, transformar ou criar em cima desta obra, poderá distribuir a obra resultante apenas sob a mesma licença, ou sob uma licença similar à presente.



Conserve o laboratório



Os equipamentos são frágeis:
use-os com cuidado



Não mexa nos cabos



Não consuma alimentos ou bebidas.
Mantenha sua garrafa de água
tampada.

Antes de começar...



Está atento ao **calendário**?



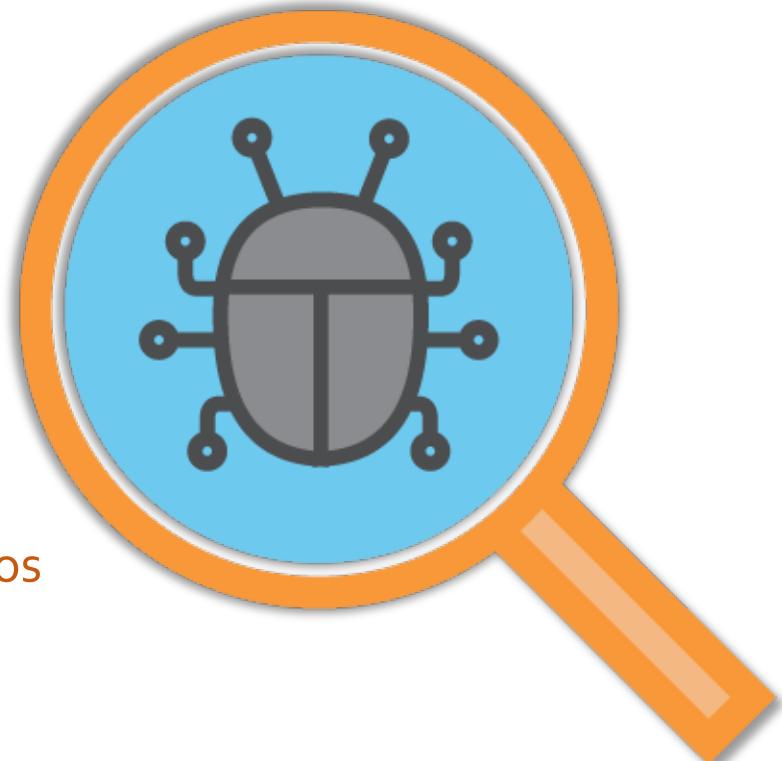
Este módulo tem **Peso 1** na avaliação



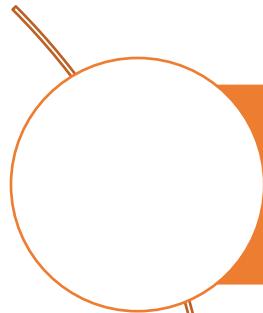
Como está sua **tática** de estudo?

Como lidar com erros de programação?

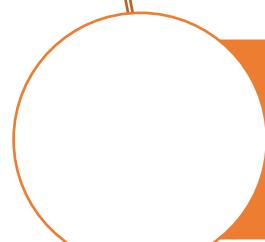
- 🛡️ Não se pergunte: “por que o programa não funciona?”
- 🛡️ Em vez disso, pergunte-se: “por que o programa está funcionando deste jeito?”
- 🛡️ Estratégias:
 - 🛡️ Não se limite ao exemplo: teste vários casos
 - 🛡️ Imprima resultados intermediários
 - 🛡️ Tente explicar o problema para outra pessoa
 - 🛡️ Dê um tempo e tente de novo mais tarde



Conteúdo



Estruturas Condicionais

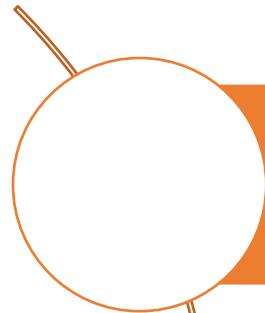


Como montar uma condição



Problemas comentados

Conteúdo



Estruturas Condicionais



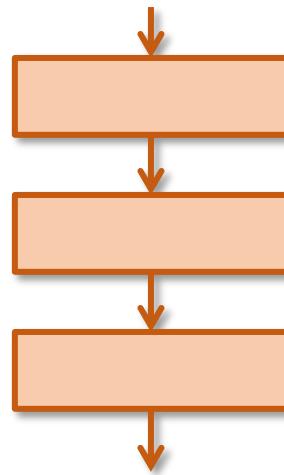
Como montar uma condição



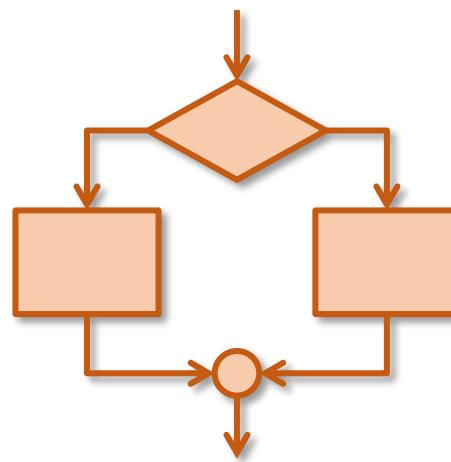
Problemas comentados

Estruturas de Programação

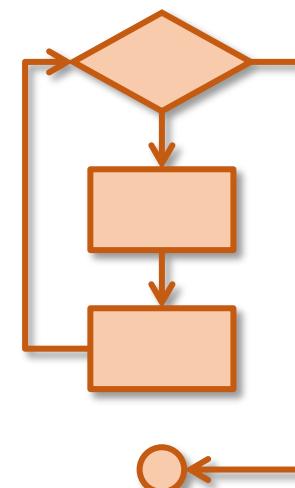
- Qualquer programa de computador pode ser escrito combinando-se os **três tipos básicos de estruturas de programação**:



Sequencial



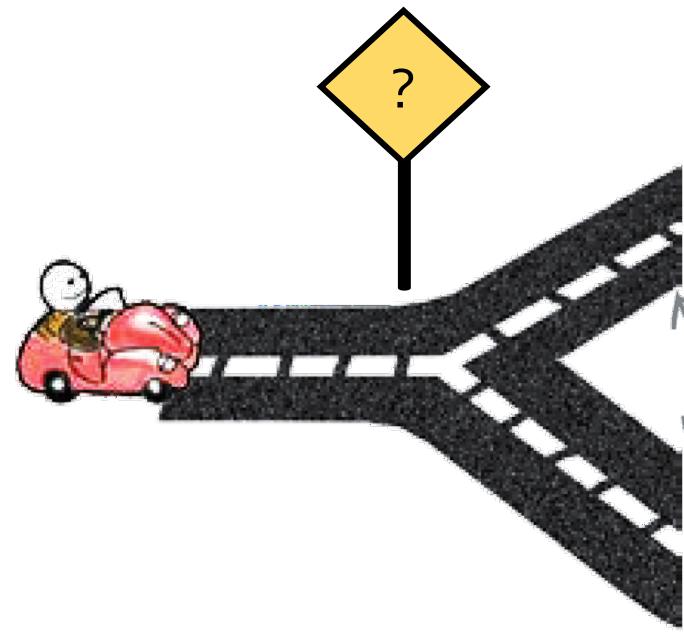
Condicional



Repetição

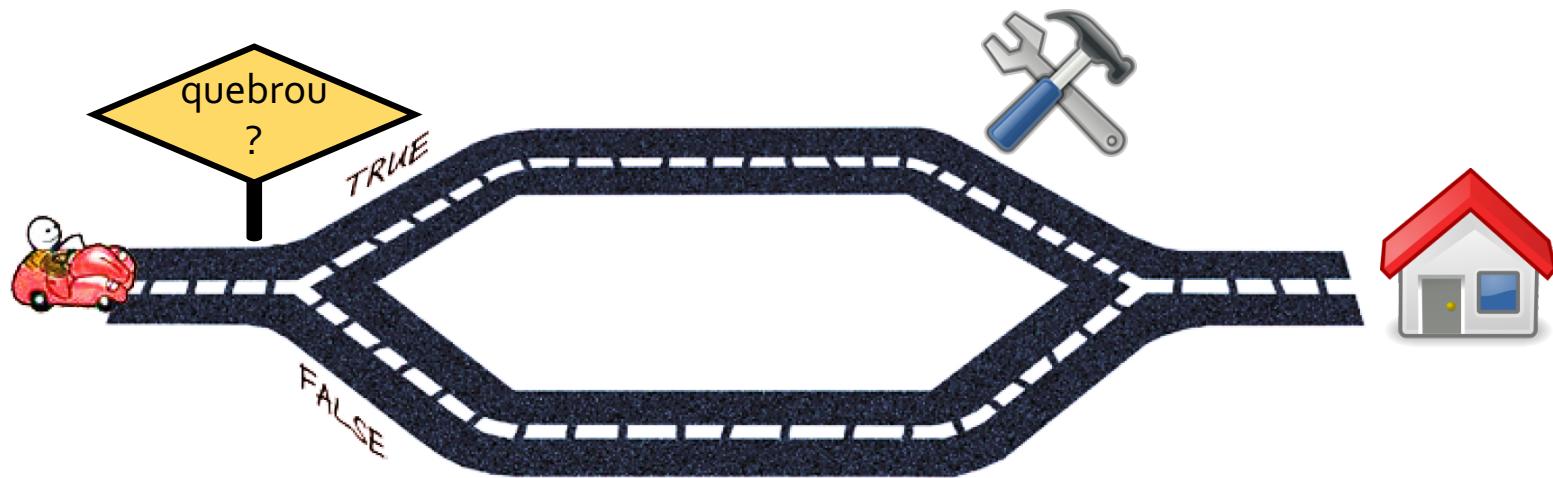
Estrutura Condicional

- Permite **alterar o fluxo de execução**, de forma a **selecionar** qual parte do algoritmo deve ser executada.
- Essa decisão é tomada a partir de uma **condição**, que pode resultar apenas em:
 - **Verdadeiro**, ou
 - **Falso**



Condição verdadeira, condição falsa

- Verdadeiro ou falso é apenas um resultado da **condição**.
- Eles indicam para que lado a execução do seu código vai bifurcar.
- Uma condição falsa **não** indica que seu script está errado.



Estruturas Condicionais

:: Formato

Início da estrutura condicional

if

Condição

Sinal de dois pontos!

:

Ações se a condição
for verdadeira

else:

Outro sinal de
dois pontos!

Ações se a condição
for falsa

Os dois
blocos são
recuados

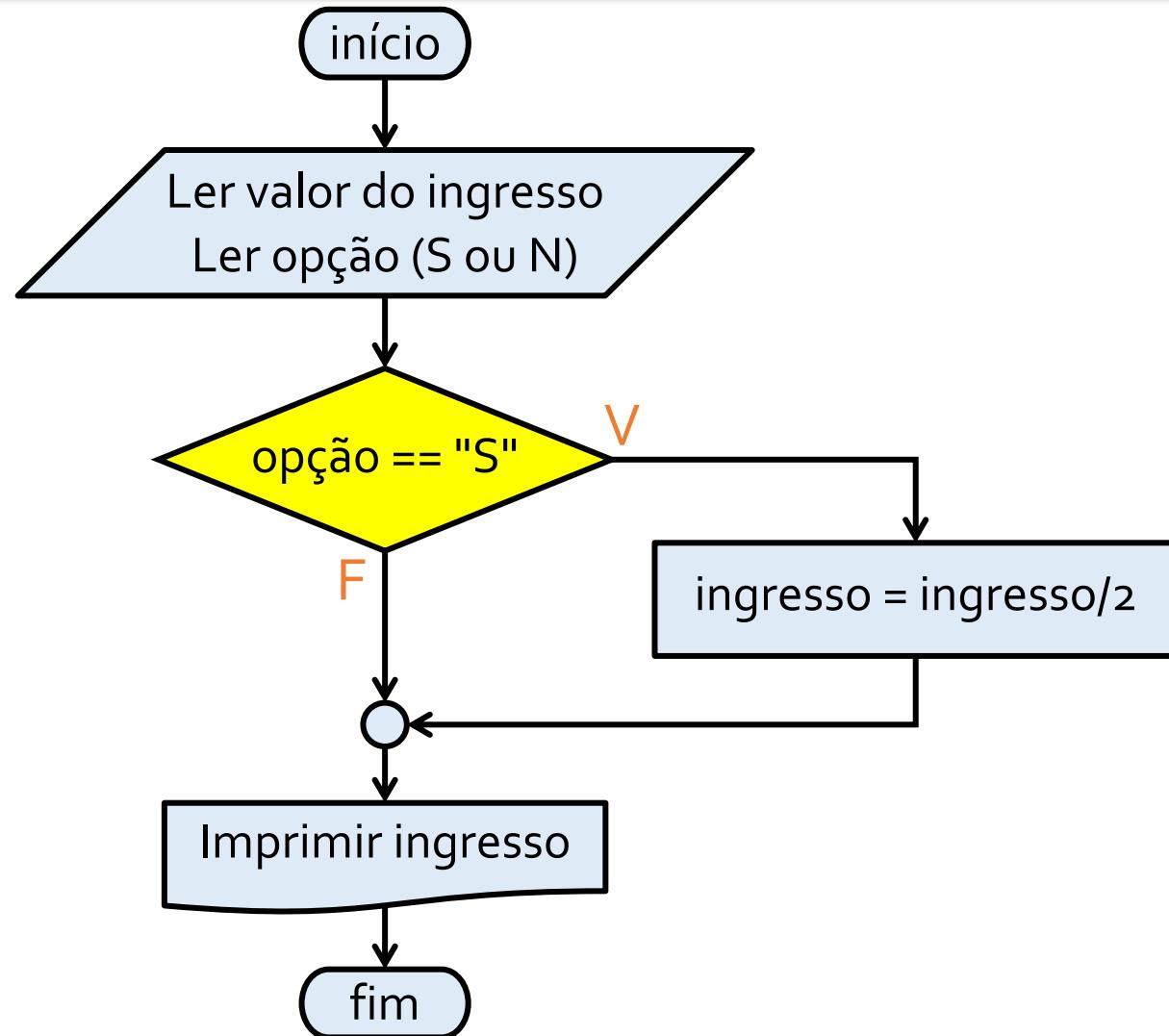
Exemplo A

- Escreva um programa que leia as seguintes entradas:
 - Valor do ingresso (R\$)
 - Mensagem informando se pessoa tem direito a meia entrada
- Como saída, imprima o valor total a ser pago



Exemplo A

:: Pensando no problema



Exemplo A

:: Script de solução

```
# Calcula o valor do ingresso
ingresso = float(input("Valor: "))
opcao = input("Meia entrada? (S/N) ")

if (opcao.upper() == "S"):
    ingresso = ingresso/2

print(round(ingresso, 2))
```

Exemplo A

:: Observações

```
# Calcula o valor do ingresso  
ingresso = float(input("Valor: "))  
opcao = input("Meia entrada? (S/N) ")
```

opcao.upper() == "S"



Ajusta todas as letras de
uma string para maiúsculas

if (opcao.upper() == "S"):

ingresso = ingresso/2



Use dois sinais de igual
na comparação!

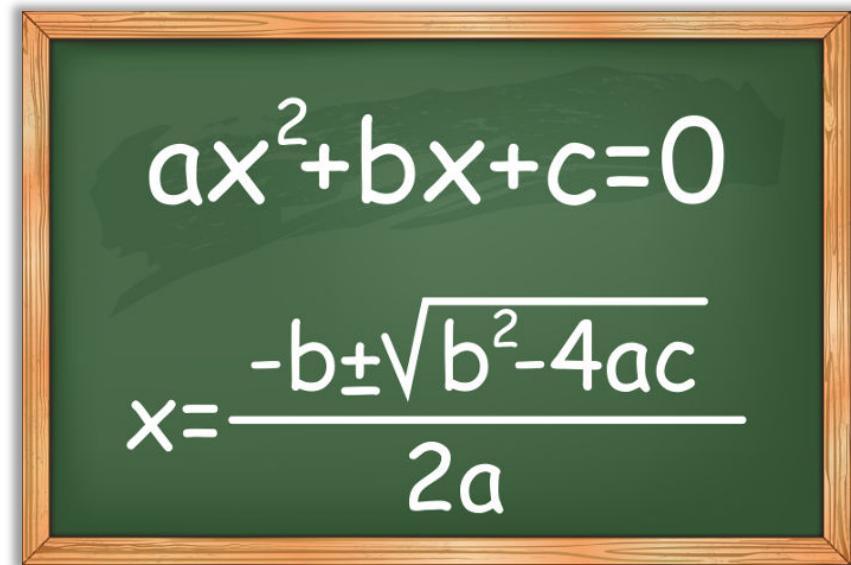
print(round(ingresso, 2))

Condições sempre
terminam com sinal
de dois pontos

Recuo (tecla TAB)

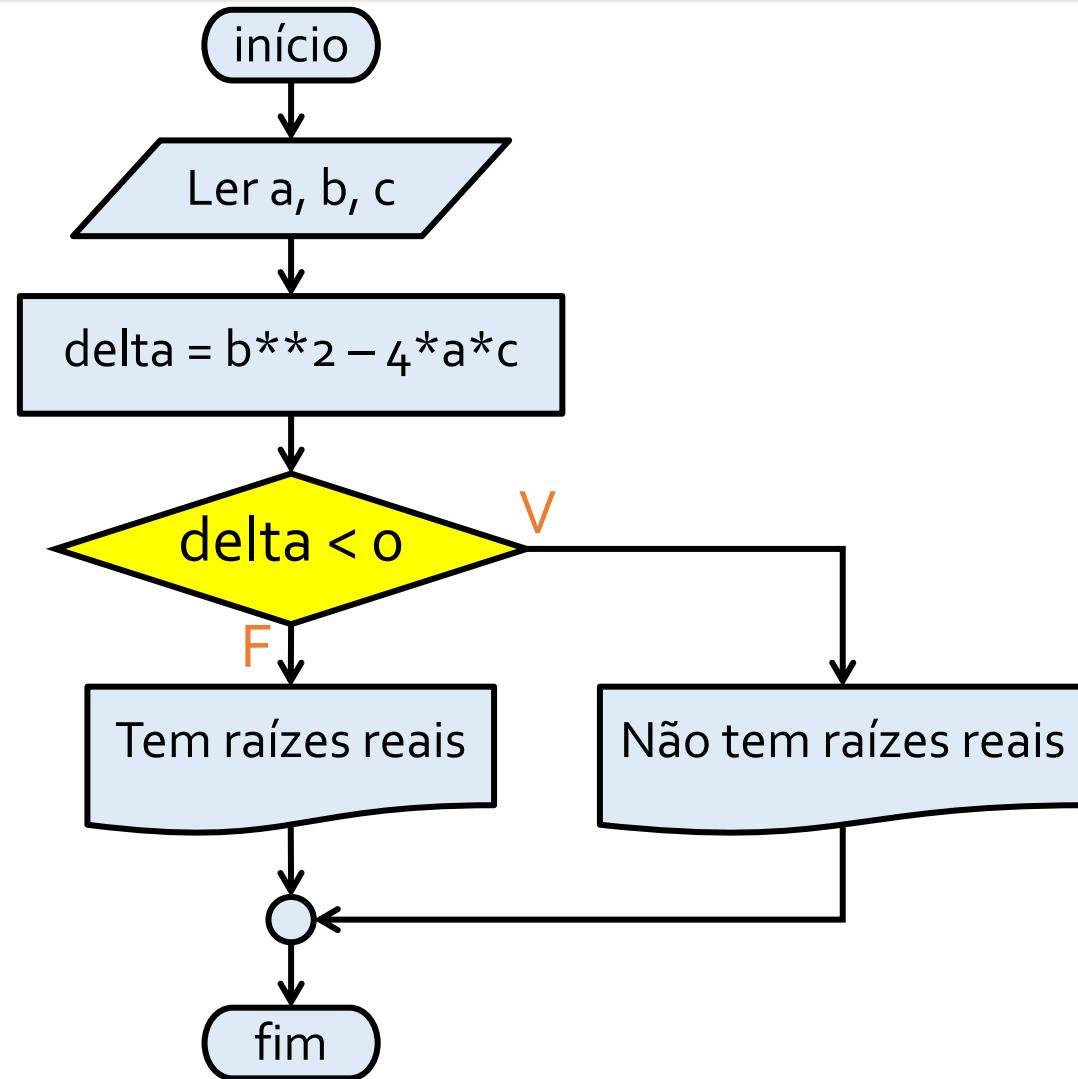
Exemplo B

- Escreva um programa que leia os **três coeficientes** de uma equação de segundo grau.
- Como saída, imprima na tela se essa equação **tem** ou **não tem** raízes reais.


$$ax^2 + bx + c = 0$$
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Exemplo B

:: Pensando no problema



Exemplo B

:: Script de solução

```
a = float(input("Digite a: "))
b = float(input("Digite b: "))
c = float(input("Digite c: "))

delta = b**2 - 4 * a * c

if (delta < 0):
    print("Não tem raiz real")
else:
    print("Tem raiz real")
```

Exemplo B

:: Observações

```
a = float(input("Digite a: "))
b = float(input("Digite b: "))
c = float(input("Digite c: "))
```

```
delta = b**2 - 4 * a * c
```

```
if (delta < 0):
    print("Não tem raiz real")
else:
    print("Tem raiz real")
```

Condição

Comandos internos ao **if** e ao
else devem ser **recuados**

if e **else** sempre terminam
com sinal de **dois pontos**

Indentação

- ⚠ O comando **else** deve estar alinhado com o comando **if** correspondente.
- ⚠ Todos os comandos de um **mesmo bloco** devem ter o **mesmo recuo**.

Indentação Válida

```
if (condição) :  
    comando  
    comando  
else:  
    comando  
    comando
```

Indentação Inválida

```
if (condição) :  
    comando  
    comando  
    .  
    else:  
        comando  
        comando
```

```
if (condição) :  
    comando  
    comando  
    else:  
        comando  
            comando
```

Indentação influencia no resultado!

```
if (temp > 25):  
    print("Quente")  
    print("Ligue o ventilador")  
print("Aproveite")
```



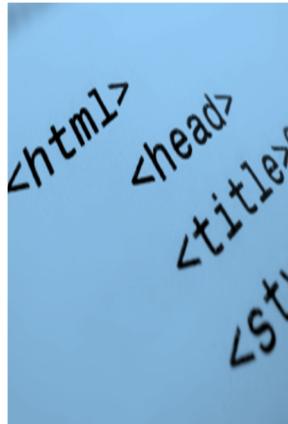
Mensagem
“Aproveite” é
sempre exibida

```
if (temp > 25):  
    print("Quente")  
    print("Ligue o ventilador")  
print("Aproveite")
```



Mensagem
“Aproveite” é exibida
somente quando
temp > 25

Não confunda



Indentação (identação)

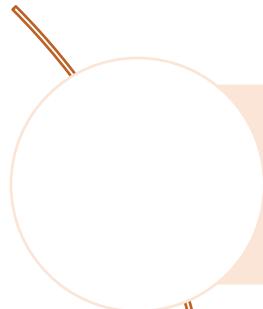
- Inserção de espaços em um código de linguagem de programação



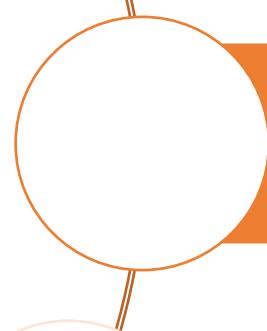
Endentação

- Encaixe dos dentes de uma peça denteada com os de outra

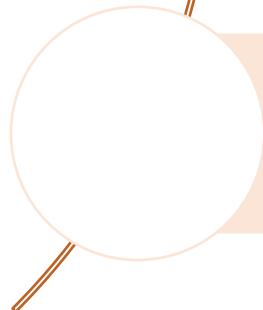
Conteúdo



Estruturas Condicionais

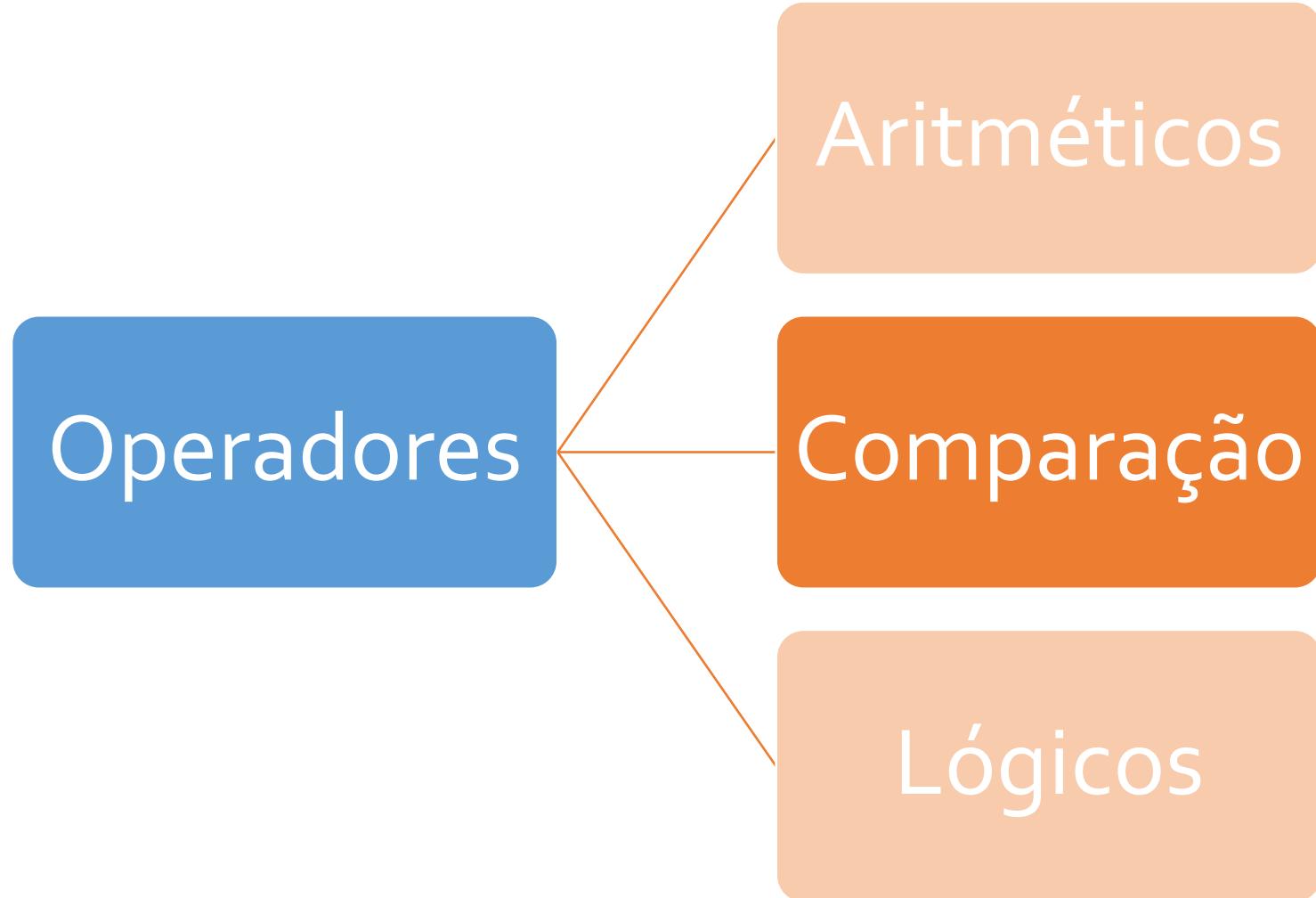


Como montar uma condição



Problemas comentados

Tipos de operadores

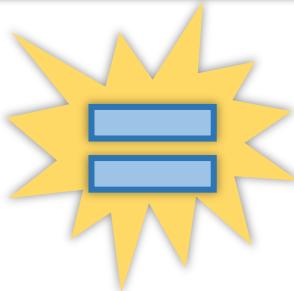


Operadores de Comparação

- 💡 São utilizados para estabelecer relação de comparação entre valores numéricos.

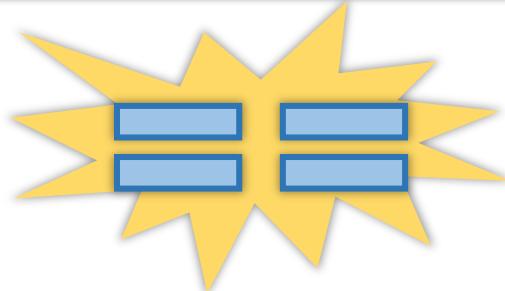
Operador	Operação	Exemplos	
<code>==</code>	Igual a	<code>3 == 3</code>	<code>20 == 18</code>
<code>></code>	Maior que	<code>5 > 4</code>	<code>10 > 11</code>
<code><</code>	Menor que	<code>3 < 6</code>	<code>9 < 7</code>
<code>>=</code>	Maior ou igual a	<code>4 >= 4</code>	<code>3 >= 5</code>
<code><=</code>	Menor ou igual a	<code>3 <= 5</code>	<code>7 <= 1</code>
<code>!=</code>	Diferente de	<code>8 != 9</code>	<code>2 != 2</code>

Os símbolos “=” e “==” são diferentes!



- Indica uma **atribuição**.
- A variável à esquerda **assume** o valor da expressão à direita.

```
x = x + 1
```



- Indica uma **comparação**.
- Nenhum valor é modificado.
- Produz um resultado lógico: **V** ou **F**.

```
if (a + b == 2) :
```

Operadores de Comparaçāo

:: Exemplos

A screenshot of a Python code editor window titled "main.py". The menu bar includes "Arquivo", "Editar", "Buscar", "Executar", and "Ferramentas". The toolbar shows "Python 3" and a file icon. The code area contains the following Python code:

```
1 print(1 == 1)
print(2 != 1)
```

The "Console" tab is selected at the bottom, showing the output:

True
True

A screenshot of a Python code editor window titled "main.py". The menu bar includes "Arquivo", "Editar", "Buscar", "Executar", and "Ferramentas". The toolbar shows "Python 3" and a file icon. The code area contains the following Python code:

```
1 x = 5
print(x > 5)
print(x <= 5)
```

The "Console" tab is selected at the bottom, showing the output:

False
True

Comparação de strings

:: Segue ordem alfabética

A screenshot of a Python code editor window titled "main.py". The code compares three strings: "dragao", "harpia", and "Dragao". It uses print statements to check if "dragao" is greater than "harpia", if "dragao" is less than "harpia", and if "dragao" is equal to "Dragao".

```
Arquivo Editar Buscar Executar Ferramentas
Python 3 main.py
1 nome1 = 'dragao'
2 nome2 = 'harpia'
3 nome3 = 'Dragao'
4 print(nome1 > nome2)
5 print(nome1 < nome2)
6 print(nome1 == nome3)

Console Shell
```

A screenshot of a Python code editor window titled "main.py". The code prints two boolean values: the result of the numerical comparison (2 < 10) and the result of the string comparison ('2' < '10').

```
Arquivo Editar Buscar Executar Ferramentas
Python 3 main.py
1 print(2 < 10)
2 print('2' < '10')

Console Shell
```

Atenção

:: Compare variáveis do mesmo tipo

💡 Neste exemplo, a variável `x` é do tipo inteiro, mas a expressão `"4"` representa um caractere, e não um número.

The screenshot shows a Python code editor interface. The menu bar includes 'Arquivo', 'Editar', 'Buscar', 'Executar', and 'Ferramentas'. The tab bar shows 'Python 3' and 'main.py'. The code in 'main.py' is:

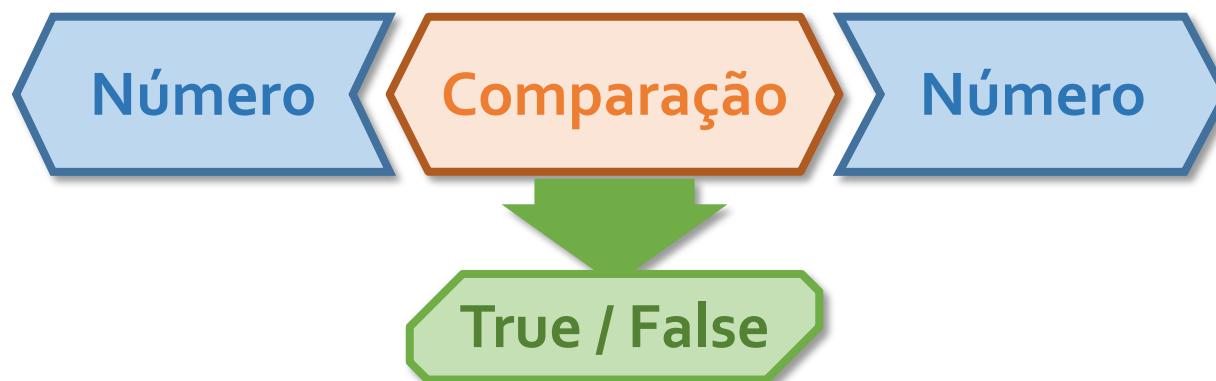
```
Arquivo Editar Buscar Executar Ferramentas
Python 3 main.py
x = 4

if ("4" == x):
    print("igual")
else:
    print("diferente")
```

The 'Console' tab is selected at the bottom, showing the output:

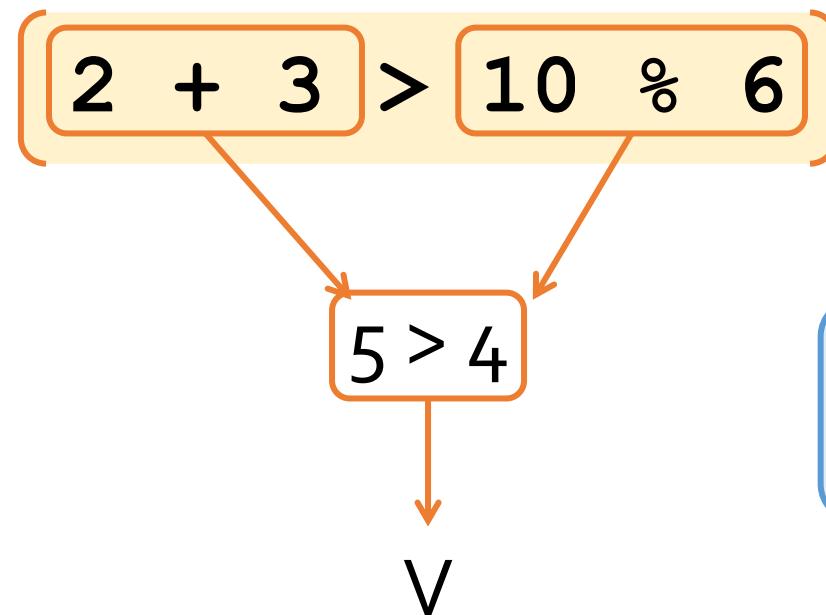
```
Console Shell
diferente
```

Operadores de comparação x aritméticos



Como avaliar uma condição?

- Operadores de comparação têm menor precedência que os operadores aritméticos.
- Por isso, as operações aritméticas são resolvidas antes das operações de comparação.



Na dúvida, use parênteses.

Como avaliar uma condição? :: Exercício

$x = 2$
$y = 3$
$z = 7$

Expressão	Verdadeiro	Falso	Mal formada
$x + y > 6$			
$x - 1 + y == 4$			
$x ** y == x * y$			
$y - 5 = z - 9$			
$1 - z =! 4$			
$x + 8 \% z >= y * 6 - 15$			

Como avaliar uma condição? :: Resposta

$x = 2$
$y = 3$
$z = 7$

Expressão	Verdadeiro	Falso	Mal formada
$x + y > 6$		X	
$x - 1 + y == 4$	X		
$x ** y == x * y$		X	
$y - 5 = z - 9$			X
$1 - z =! 4$			X
$x + 8 \% z >= y * 6 - 15$	X		

Se você negar a condição, inverta os comandos **if/else**

Arquivo Editar Buscar Executar Ferramentas

Python 3 main.py

```
nota = float(input(""))

if (nota >= 5.0):
    print("Passou")
else:
    print("Reprovou")
```

Console Shell

Arquivo Editar Buscar Executar Ferramentas

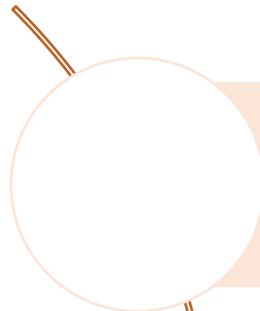
Python 3 main.py

```
nota = float(input(""))

if (nota < 5.0):
    print("Reprovou")
else:
    print("Passou")
```

Console Shell

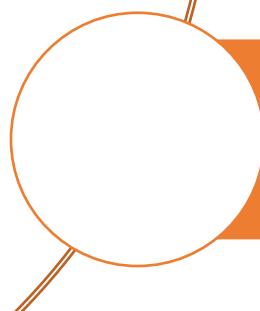
Conteúdo



Estruturas Condicionais



Como montar uma condição



Problemas comentados

Problema 1

- Um radar de trânsito verifica a velocidade dos veículos.
- Caso ultrapassem **60 km/h**, emite-se um registro de multa.
- O valor da multa é de **R\$ 200,00** mais **R\$ 3,00** para cada **1 km/h** acima do limite.
- Escreva um programa para determinar o valor da multa.



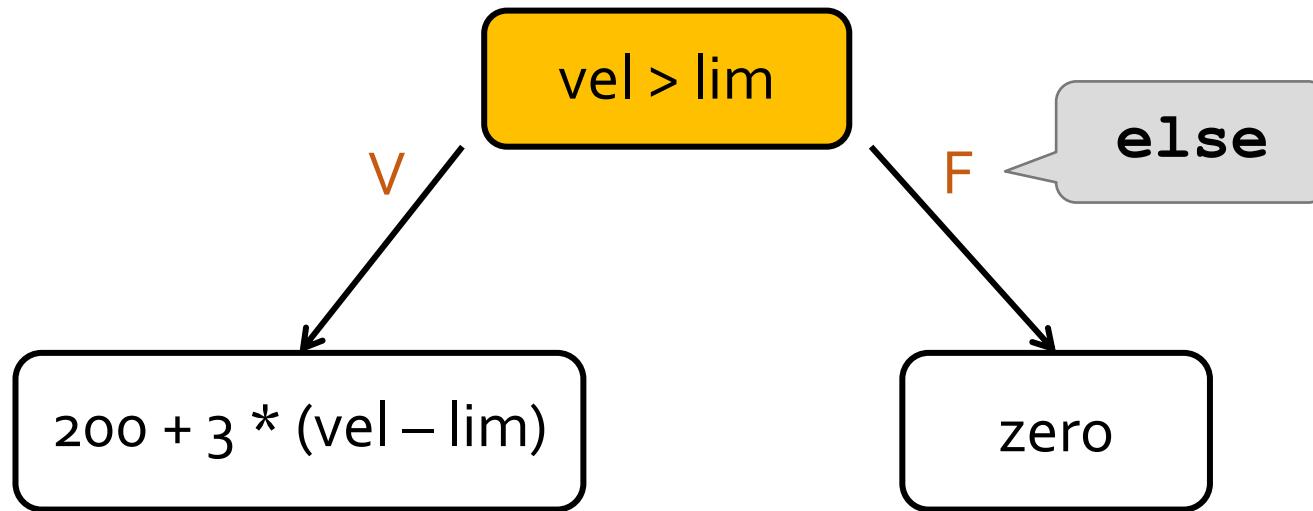
Problema 1

2 – Identifique as entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Velocidade	km/h	≥ 0
Saídas	Multa	R\$	≥ 0

Problema 1

3 – Projete o script



Problema 1

4 – Codifique em Python

```
# Entrada de dados e definicao de constantes  
vel = float(input("Informe a velocidade: "))  
lim = 60 # Limite de velocidade
```

```
# Calculo do valor da multa
```

```
if (vel > lim):  
    multa = 200 + 3 * (vel - lim)  
else:  
    multa = 0
```

```
# Exibicao de resultados  
print(multa)
```

if (vel <= lim):
 multa = 0
else:
 multa = 200 + 3 * (vel - lim)

Alternativa

Problema 1

5 – Teste o script



Valor **menor**
que o limiar

Valor **igual** ao
limiar

Valor **maior**
que o limiar

Problema 2

- 💡 Quantos pontos de força (pf) um esqueleto tira durante um ataque a um personagem com pf_o pontos de força **iniciais** ?
- 💡 O ataque é determinado pela seguinte fórmula:

$$pf = \begin{cases} 0,25 * pf_o + 5, & \text{se } pf_o \leq 50 \\ 0,20 * pf_o + 2, & \text{se } pf_o > 50 \end{cases}$$



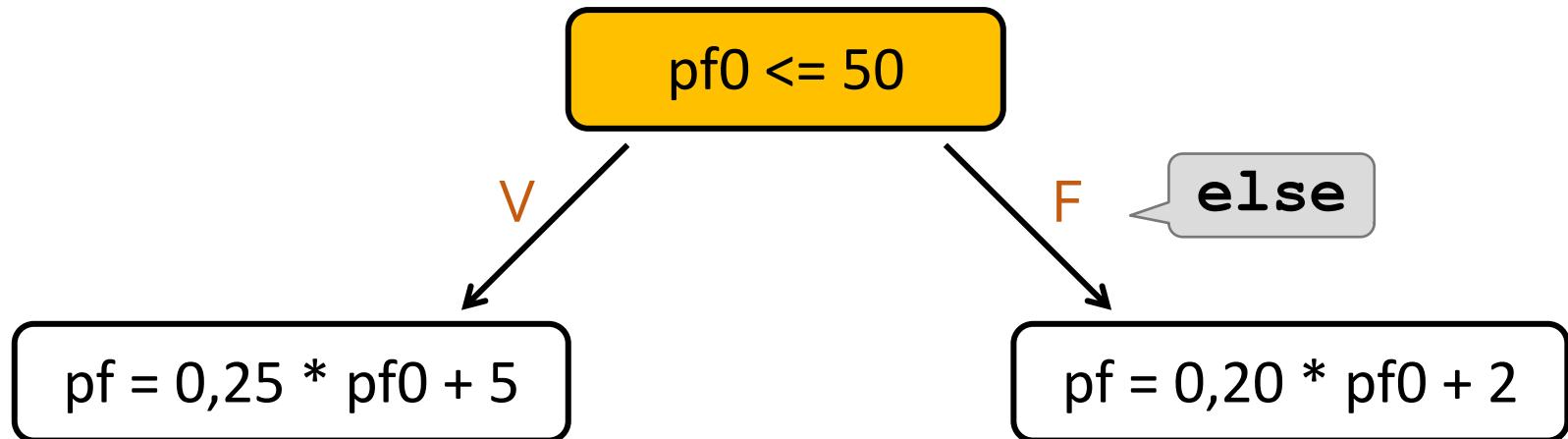
Problema 2

2 – Identifique as entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Ponto de força inicial	---	≥ 0
Saídas	Ponto de força tirado	---	≥ 0

Problema 2

3 – Projete o script



Problema 2

4 – Codifique em Python

```
# Entrada de dados  
pf0 = int(input("Ponto de forca inicial: "))
```

```
if (pf0 <= 50):  
    pf = 0.25 * pf0 + 5  
else:  
    pf = 0.20 * pf0 + 2
```

```
print(pf)
```



```
if (pf0 > 50):      Alternativa  
    pf = 0.20 * pf0 + 2  
else:  
    pf = 0.25 * pf0 + 5
```

Problema 2

5 – Teste o script



Valor **menor**
que o limiar



Valor **igual** ao
limiar



Valor **maior**
que o limiar

Problema 3

- Há duas marcas de leite em pó: A e B.
- Leia o preço de cada marca.
- Leia o peso líquido de cada marca.
- Imprima qual marca tem melhor relação custo/benefício.



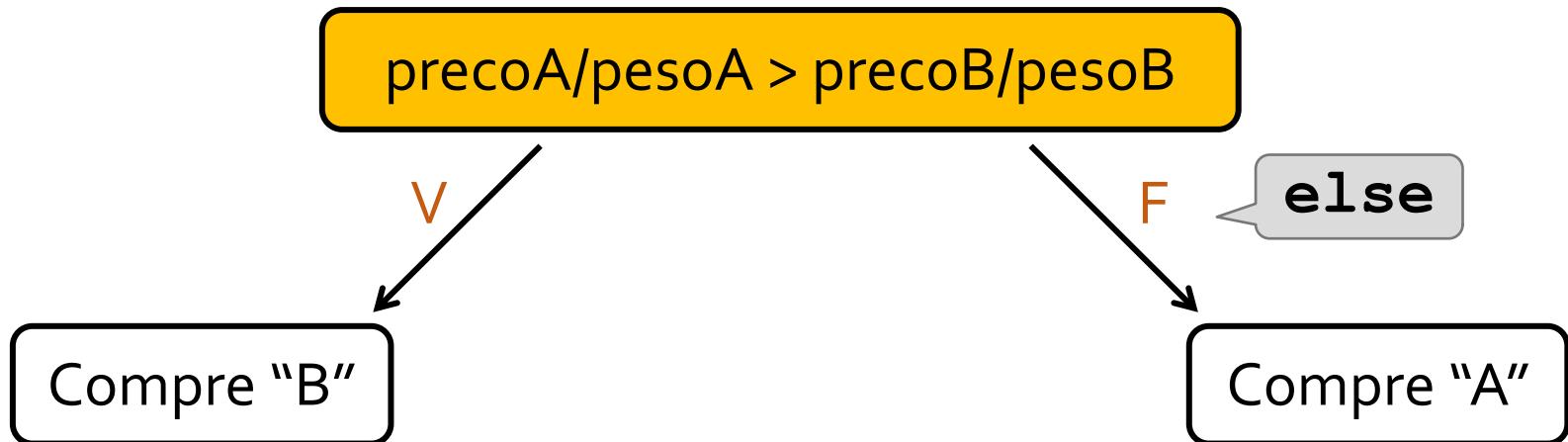
Problema 3

2 – Identifique as entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	PrecoA	R\$	> 0.0
	PesoA	kg	> 0.0
	PrecoB	R\$	> 0.0
	PesoB	kg	> 0.0
Saídas	Marca	---	{"A", "B"}

Problema 3

3 – Projete o script



Problema 3

4 – Codifique em Python

```
# Entrada de dados
prcA = float(input("Digite o preco da marca A: "))
prcB = float(input("Digite o preco da marca B: "))
psA = float(input("Digite o peso da marca A: "))
psB = float(input("Digite o peso da marca B: "))

if (prcA/psA > prcB/psB) :
    marca = "B"
else:
    marca = "A"

msg = "Compre a marca " + marca
print(msg)
```



```
if (prcB/psB > prcA/psA) :
    marca = "A"
else:
    marca = "B"      Alternativa
```

Concatenação
de strings

Problema 3

5 – Teste o script

Preço A: R\$ 10,00

Preço B: R\$ 20,00

Peso A: 400g

Peso B: 400g

Preço A: R\$ 10,00

Preço B: R\$ 10,00

Peso A: 400g

Peso B: 800g

Preço A: R\$ 15,00

Preço B: R\$ 30,00

Peso A: 400g

Peso B: 800g

Marca **A**
claramente
mais barata

Marca **B**
claramente
mais barata

Marcas com
mesma
proporção

Problema 4

- 💡 Escreva um programa que verifique se uma senha de quatro dígitos é válida.
- 💡 A senha é válida se a soma dos dígitos da **primeira** e **da terceira** posição (da esquerda para a direita) for **múltiplo** da soma dos dígitos da **segunda** e da **quarta** posição.



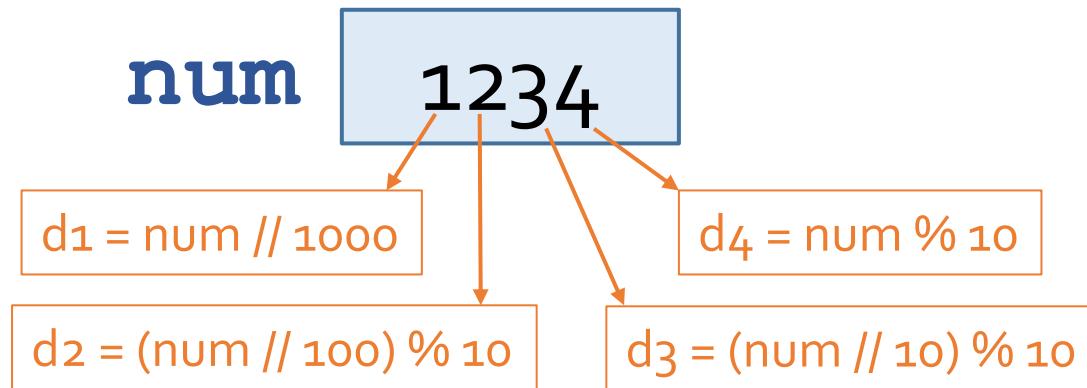
Problema 4

2 – Identifique as entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	senha	---	[0; 9999]
Saídas	Mensagem	---	{"liberado", "negado"}

Problema 4

3 – Projete o script



$$(d_1 + d_3) \% (d_2 + d_4) \neq 0$$

V

F

else

Acesso negado

Acesso liberado

Problema 4

4 – Codifique em Python

```
num = int(input("Digite a senha (4 digitos): "))

# Decompose os quatro digitos da senha
d1 = num // 1000
d2 = (num // 100) % 10
d3 = (num // 10) % 10
d4 = num % 10

if ( (d1 + d3) % (d2 + d4) != 0):
    msg = "negado"
else:
    msg = "liberado"

msg = "Acesso " + msg
print(msg)
```

Problema 4

5 – Teste o script

1 2 3 4

Senha inválida

5 5 5 5

Senha válida

Problema 5



- Um evento começou no horário h_1 e terminou no mesmo dia, no horário h_2 , também medido em horas e minutos.
- Quanto tempo durou o evento?

Problema 5

2 – Identifique as entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Horário 1 (hh ₁ , mm ₁)	horas, minutos	[0; 23], [0; 59]
	Horário 2 (hh ₂ , mm ₂)	horas, minutos	[0; 23], [0; 59]
Saídas	Diferença de tempo (Δh, Δm)	horas, minutos	[0; 23], [0; 59]

Problema 5

3 – Projete o script

Caso 1

$$mm2 \geq mm1$$

Início: 9h 17min
Fim: 15h 43min

Caso 2

$$mm2 < mm1$$

Início: 9h 43min
Fim: 15h 17min

$$\Delta m = 43 - 17 = 26\text{min}$$

$$\Delta m = 17 - 43 = 34\text{min} (-1\text{h})$$

$$\Delta h = 15 - 9 = 6\text{h}$$

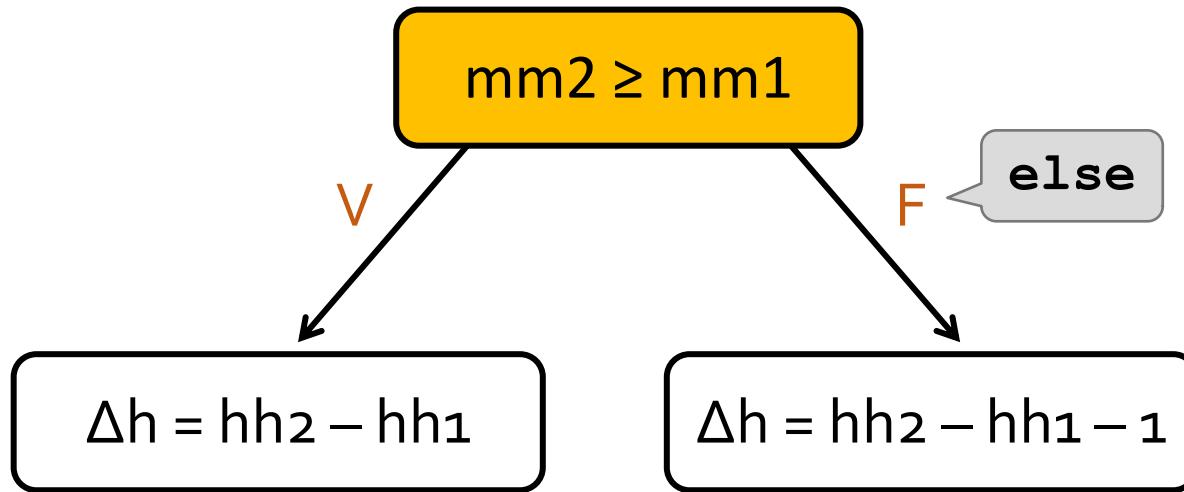
$$\Delta h = 15 - 9 - 1 = 5\text{h}$$

if

$$\Delta m = (mm2 - mm1) \% 60$$

Problema 5

3 – Projete o script



Problema 5

4 – Codifique em Python

```
# Entrada de dados
hh1 = int(input("Hora inicial: "))
mm1 = int(input("Minuto inicial: "))
hh2 = int(input("Hora final: "))
mm2 = int(input("Minuto final: "))

# Diferença de minutos
dm = (mm2 - mm1) % 60

# Diferença de horas
if (mm2 >= mm1):
    dh = hh2 - hh1
else:
    dh = hh2 - hh1 - 1
    print(dh, dm)

if (mm2 < mm1):
    dh = hh2 - hh1 - 1
else:
    dh = hh2 - hh1  Alternativa
```

Problema 5

5 – Teste o script

Caso 1

Início: 9h 17min
Fim: 15h 43min

Caso 2

Início: 9h 43min
Fim: 15h 17min

Referências bibliográficas

-  MENEZES, Nilo Ney Coutinho (2014). *Introdução à Programação com Python*, 2 ed. Editora Novatec.
-  HETLAND, Magnus Lie (2008). *Beginning Python: From Novice to Professional*. Springer eBooks, 2^a edição.
Disponível em: <http://dx.doi.org/10.1007/978-1-4302-0634-7>.
-  GADDIS, Tony (2012). *Starting out with Python*, 2 ed. Editora Addison-Wesley.
-  DIERBACH, Charles. *Introduction to Computer Science using Python*: a computational problem-solving approach. John Wiley & Sons, 2012.

Dúvidas?

