



# [Aula 2-B] Linguagem de Programação

## Análise e Desenvolvimento de Sistemas

Operadores e Estrutura de Decisão – Prof. Jean Zahn

[jeanozahn@gmail.com](mailto:jeanozahn@gmail.com)

# Na aula de hoje

---

## ▶ Operadores

- ▶ Aritméticos (usados em contas)
- ▶ Relacionais (usados em comparações numéricas)
- ▶ Lógicos (usados em comparações lógicas)
- ▶ De atribuição (armazenamento de valores em variáveis)

## ▶ Estruturas de decisão

- ▶ *if...*
- ▶ *if...else*
- ▶ *if...elif...*



# Operadores aritméticos

---

Operador	Exemplo	Prioridade
(x)	$(1 + 2) * 3 \rightarrow 9$	1
**	$2 ** 3 \rightarrow 8$	2
+x	+15	3
-x	$-(5+3) \rightarrow -8$	3
*	$5 * 3 \rightarrow 15$	4
/	$5 / 3 \rightarrow 1.66$	4
//	$5 // 3 \rightarrow 1$	4
%	$5 \% 3 \rightarrow 2$	4
+	$5 + 3 \rightarrow 8$	5
-	$5 - 3 \rightarrow 2$	5



# Operadores aritméticos

---

- ▶ Operadores com a mesma prioridade (precedência) são analisados da esquerda para a direita
- ▶ Divisão de inteiros (//)
  - ▶ Resultado é somente a parte inteira da divisão
- ▶ Divisão (/)
  - ▶ Resultado fracionário



# Exemplo

---

## ► Considerando

$$x = 512$$

$$y = 9.2 - (x // 10 - 14 / 5) + 14 * 0.1$$

## ► Resolução de y

$$y = 9.2 - (512 // 10 - 14 / 5) + 14 * 0.1$$

$$y = 9.2 - (51 - 14 / 5) + 14 * 0.1$$

$$y = 9.2 - (51 - 2.8) + 14 * 0.1$$

$$y = 9.2 - 48.2 + 14 * 0.1$$

$$y = 9.2 - 48.2 + 1.4$$

$$y = -39 + 1.4$$

$$y = -37.6$$



# Conversão de Tipos

---

- ▶ Em algumas situações o programador deseja transformar o tipo de uma expressão
  - ▶ Para isso, basta envolver a variável a ser transformada por “tipo(variável)”

- ▶ **Exemplo: transformar um real em um inteiro**

```
a = 5.1
```

```
x = int(a)
```

```
x vale 5
```

- ▶ **Exemplo: transformar um inteiro em um real**

```
b = 5
```

```
y = float(b)
```

```
y vale 5.0
```



# Exemplo

---

```
x = int(3.3 / ( 5/2 ) - 5)
```

```
y = int(3.3) / ( 5/2 ) - 5
```

## ► Resolução de x

```
x = int(3.3 / ( 5/2 ) - 5)
```

```
x = int(3.3 / 2.5 - 5)
```

```
x = int(1.32 - 5)
```

```
x = int(-3.68)
```

```
x = -3
```

## ► Resolução de y

```
y = int(3.3) / ( 5/2 ) - 5
```

```
y = int(3.3) / 2.5 - 5
```

```
y = 3 / 2.5 - 5
```

```
y = 1.2 - 5
```

```
y = -3.8
```



# Funções matemáticas: números e suas representações

Método	Descrição	Exemplo
<code>math.ceil(x)</code>	Arredonda para cima	<code>math.ceil(5.3) → 6</code>
<code>math.copysign(x, y)</code>	Obtém um float com o valor absoluto de x, mas com o sinal de y	<code>math.copysign(-5.3, 1) → 5.3</code>
<code>math.fabs(x)</code>	Valor absoluto de x	<code>math.fabs(-5.3) → 5.3</code>
<code>math.floor(expr)</code>	Arredonda para baixo	<code>math.floor(5.3) → 5</code>
<code>math.fmod(x, y)</code>	Resto da divisão de x por y (usar quando x ou y forem float, caso contrário usar %)	<code>math.fmod(5.4, 2) → 1.4</code>
<code>math.trunc(x)</code>	Parte inteira de x	<code>math.trunc(5.6) → 5</code>

Constantes:

`math.pi` → 3.141592...

`math.e` → 2.718281...

Para usar essas funções ou constantes, colocar **import math** no início do programa



# Funções matemáticas: potência e funções logarítmicas

Método	Descrição	Exemplo
<code>math.exp(x)</code>	$e^{**}x$	<code>math.exp(2)</code> → 7.38905609893065
<code>math.log(x)</code>	Logaritmo natural de x (base e)	<code>math.log(2)</code> → 0.6931471805599453
<code>math.log(x, y)</code>	Logaritmo de x na base y	<code>math.log(2, 10)</code> → 0.30102999566398114
<code>math.pow(x, y)</code>	$x^{**}y$	<code>math.pow(2, 3)</code> → 8.0
<code>math.sqrt(x)</code>	Raiz quadrada de x	<code>math.sqrt(16)</code> → 4.0

Para usar essas funções, colocar **import math** no início do programa



# Exemplo – Distância entre dois pontos

```
import math
```

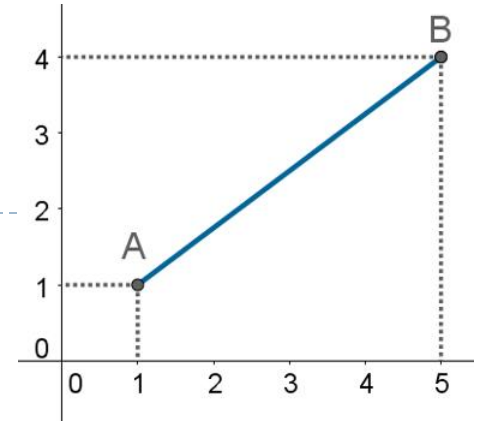
```
x1 = int(input("Entre com a coordenada x do 1o. ponto:"))  
y1 = int(input("Entre com a coordenada y do 1o. ponto:"))  
x2 = int(input("Entre com a coordenada x do 2o. ponto:"))  
y2 = int(input("Entre com a coordenada y do 2o. ponto:"))
```

```
cateto1 = math.fabs(y2-y1)
```

```
cateto2 = math.fabs(x2-x1)
```

```
hipotenusa = math.sqrt(cateto1 ** 2 + cateto2 ** 2)
```

```
print("A distancia entre os dois pontos é", hipotenusa)
```



# Funções matemáticas: trigonometria

---

Função	Descrição	Exemplo
<code>math.sin(x)</code>	Seno	<code>math.sin(0) → 0.0</code>
<code>math.asin(x)</code>	Arco seno	<code>math.asin(1) → 1.5707963267948966</code>
<code>math.cos(x)</code>	Cosseno	<code>math.cos(0) → 1.0</code>
<code>math.acos(x)</code>	Arco cosseno	<code>math.acos(-1) → 3.141592653589793</code>
<code>math.tan(x)</code>	Tangente	<code>math.tan(1) → 1.5574077246549023</code>
<code>math.atan(x)</code>	Arco tangente	<code>math.atan(1) → 0.7853981633974483</code>
<code>math.degrees(x)</code> )	Converte radianos para graus	<code>math.degrees(math.pi) → 180.0</code>
<code>math.radians(x)</code>	Converte graus para radianos	<code>math.radians(180) → 3.141592653589793</code>



# Números aleatórios

---

- ▶ **Algumas aplicações necessitam que o computador sorteie um número**
  - ▶ Função `random.random()`
  - ▶ Gera número pseudo aleatório entre  $[0,1]$
- ▶ **A partir desse número, é possível gerar números em outros intervalos**
  - ▶  $\text{inicio} + (\text{fim} - \text{inicio}) * \text{random.random()}$
- ▶ **Para usar, seguir esses passos**

```
import random  
  
y = random.random()  
# y conterá um número real sorteado  
# entre 0 e 1
```



# Números aleatórios (exemplo)

---

- ▶ **Número entre 0 e 1**

```
print(random.random())
```

- ▶ **Número entre 5 e 6**

```
print(5 + random.random())
```

- ▶ **Número entre 0 e 10**

- ▶ 

```
print(random.random() * 10)
```

- ▶ **Número entre 50 e 70**

- ▶ 

```
print(50 + random.random() * 20)
```



# Números aleatórios inteiros

---

- ▶ **É possível gerar números aleatórios inteiros**

```
import random
```

```
y = random.randint(3, 9)
```

```
# y conterá um número inteiro sorteado
```

```
# entre 3 e 9
```



# Operadores relacionais

---

Operador	Exemplo	Prioridade
$x < y$	$5 < 3 \rightarrow \text{False}$	6
$x \leq y$	$5 \leq 3 \rightarrow \text{False}$	6
$x > y$	$5 > 3 \rightarrow \text{True}$	6
$x \geq y$	$5 \geq 3 \rightarrow \text{True}$	6
$x == y$	$5 == 3 \rightarrow \text{False}$	6
$x \neq y$	$5 \neq 3 \rightarrow \text{True}$	6

- ▶ Prioridade sempre **inferior** aos operadores aritméticos
- ▶ Sempre têm **resultado booleano**



# Operadores lógicos

---

Operador	Exemplo	Prioridade
not x	not True $\rightarrow$ False	7
x and y	True and False $\rightarrow$ False	8
x or y	True or False $\rightarrow$ True	9

- ▶ Prioridade sempre **inferior** aos operadores aritméticos
- ▶ Sempre têm **resultado booleano**





# Tabela verdade

---

<b>a</b>	<b>b</b>	<b>not a</b>	<b>a and b</b>	<b>a or b</b>
<b>True</b>	<b>True</b>	False	True	True
<b>True</b>	<b>False</b>	False	False	True
<b>False</b>	<b>True</b>	True	False	True
<b>False</b>	<b>False</b>	True	False	False



# Atribuição

---

- ▶ Variável do lado esquerdo, valor ou expressão do lado direito

```
x = 0
```

- ▶ Pode-se atribuir valor a várias variáveis ao mesmo tempo

```
x = y = z = 0
```

```
# x, y e z terão valor 0
```

- ▶ Pode-se também atribuir valores diferentes para variáveis diferentes ao mesmo tempo

```
x, y = 1, 2
```

```
# x terá o valor 1, e y terá o valor 2
```



# Exemplo

---

```
x = 10
```

```
y = -2
```

```
z = 5
```

```
w = x * y < z / x or x / y > z * x and z * y < x
```

- ▶ Como o valor de w seria avaliado pelo interpretador Python?



## Resolução de w

---

$w = x * y < z / x \text{ or } x / y > z * x \text{ and } z * y < x$

$w = 10 * -2 < 5 / 10 \text{ or } 10 / -2 > 5 / 10 \text{ and } 5 * -2 < 10$

$w = -20 < 5 / 10 \text{ or } 10 / -2 > 5 / 10 \text{ and } 5 * -2 < 10$

$w = -20 < 0.5 \text{ or } 10 / -2 > 5 / 10 \text{ and } 5 * -2 < 10$

$w = -20 < 0.5 \text{ or } -5 > 5 / 10 \text{ and } 5 * -2 < 10$

$w = -20 < 0.5 \text{ or } -5 > 5 / 10 \text{ and } 5 * -2 < 10$

$w = -20 < 0.5 \text{ or } -5 > 0.5 \text{ and } 5 * -2 < 10$

$w = -20 < 0.5 \text{ or } -5 > 0.5 \text{ and } -10 < 10$

$w = \text{True or } -5 > 0.5 \text{ and } -10 < 10$

$w = \text{True or False and } -10 < 10$

$w = \text{True or False and True}$

$w = \text{True or False}$

$w = \text{True}$

---



# Referência sobre operadores e prioridades

---

- ▶ Tutorial do Python 3

- ▶ <https://docs.python.org/3.3/reference/expressions.html#operator-precedence>



# Decisão

---

## Mecanismos de decisão:

- ▶ *if ...*
  - ▶ Executa algo somente quando uma condição é verdadeira
- ▶ *if... else*
  - ▶ Bifurca a execução do código em função de uma condição
- ▶ *if... elif...*
  - ▶ Executa apenas o bloco em que a condição é verdadeira



# Decisão do tipo if...

---

## Pseudocódigo

```
...  
se CONDIÇÃO então  
    INSTRUÇÃO 1  
    INSTRUÇÃO 2  
    ...  
    INSTRUÇÃO N  
fimse  
...
```

## Python

```
...  
if CONDIÇÃO:  
    INSTRUÇÃO 1  
    INSTRUÇÃO 2  
    ...  
    INSTRUÇÃO N  
...
```



## Decisão do tipo *if*...

---

- ▶ Executa o bloco de instruções somente se a condição for verdadeira
- ▶ A condição é uma expressão booleana que pode fazer uso de quaisquer operadores
- ▶ O bloco de instruções é delimitado por endentação





## Exemplo de *if*...

---

- ▶ Programa para informar quando um número inteiro é par:

```
numero = int(input("Entre com um numero: "))  
if (numero % 2 == 0):  
    print("O número é par")
```

if com instrução simples



## Exemplo de *if*...

---

- ▶ Programa para somar dois números, se o usuário desejar:

```
op = input("Deseja somar? (S/N) ")
if (op == "S") :
    x = int(input("Digite o primeiro numero:"))
    y = int(input("Digite o segundo numero:"))
    resultado = x + y
    print("O resultado da soma é", resultado)
print("Até a próxima! ")
```

if com bloco de instruções



# Decisão do tipo if... *else*

---

## Pseudocódigo

```
...  
Se CONDIÇÃO então  
    INSTRUÇÃO 1  
    INSTRUÇÃO 2  
    ...  
    INSTRUÇÃO N  
Senão  
    INSTRUÇÃO 1  
    INSTRUÇÃO 2  
    ...  
    INSTRUÇÃO N  
Fimse
```

## Python

```
...  
if CONDIÇÃO:  
    INSTRUÇÃO 1  
    INSTRUÇÃO 2  
    ...  
    INSTRUÇÃO N  
else:  
    INSTRUÇÃO 1  
    INSTRUÇÃO 2  
    ...  
    INSTRUÇÃO N  
...
```



## Decisão do tipo *if... else*

---

- ▶ Executa um ou o outro bloco de instruções em função da condição ser verdadeira ou falsa
- ▶ Valem as mesmas regras para **if...**
- ▶ Qualquer combinação de instrução individual ou em bloco é aceita no corpo do **if** ou do **else**
- ▶ Podem ser aninhados com outras estruturas



## Exemplo de *if... else*

---

- ▶ Programa para informar se um número é par ou impar:

```
numero = int(input("Entre com um número: "))  
if numero % 2 == 0:  
    print("O número é par.")  
else:  
    print("O número é impar.")
```



## Exemplo de *if... else*

---

### ► Programa para somar ou multiplicar dois números

```
op = input("Deseja somar (S) ou multiplicar (M) ?")
x = int(input("Digite o primeiro numero:"))
y = int(input("Digite o segundo numero:"))
if (op == "S") :
    r = x + y
    print("O resultado da soma é", r)
else:
    r = x * y
    print("O resultado da multiplicação é", r)
```



## Exemplo de *if... else*

---

- ▶ Programa para somar ou multiplicar dois números

```
op = input("Deseja somar (S) ou multiplicar (M) ?")
x = int(input("Digite o primeiro número: "))
y = int(input("Digite o segundo número: "))
if (op == 'S'):
    r = x + y
    print("O resultado da soma é", r)
else:
    r = x * y
    print("O resultado da multiplicação é", r)
```

Problema: a multiplicação será realizada mesmo se o usuário digitar algo diferente de M



## Exemplo de *if... else*

### ► Programa para somar ou multiplicar dois números

```
op = input("Deseja somar (S) ou multiplicar (M) ?")
x = int(input("Digite o primeiro numero:"))
y = int(input("Digite o segundo numero:"))
if (op == "S") :
    r = x + y
    print("O resultado da soma é", r)
else:
    r = x * y
    print("O resultado da multiplicação é", r)
```

Caso não seja "S", execute  
o que está abaixo!





# Decisão do tipo *if... elif...*

---

## Pseudocódigo

```
...
Se CONDIÇÃO então
    INSTRUÇÃO 1
    INSTRUÇÃO 2
    ...
    INSTRUÇÃO N
Senão Se CONDIÇÃO então
    INSTRUÇÃO 1
    INSTRUÇÃO 2
    ...
    INSTRUÇÃO N
fimse
```

## Python

```
...
if CONDIÇÃO:
    INSTRUÇÃO 1
    INSTRUÇÃO 2
    ...
    INSTRUÇÃO N
elif CONDIÇÃO:
    INSTRUÇÃO 1
    INSTRUÇÃO 2
    ...
    INSTRUÇÃO N
```



## Decisão do tipo *if... elif...*

---

- ▶ Apenas o bloco no qual a condição é verdadeira é executado
- ▶ É possível colocar tantos *elif* quantos forem necessários
- ▶ Qualquer combinação de instrução individual ou em bloco é aceita no corpo do *if* ou do *elif*
- ▶ É possível adicionar um *else* ao final de tudo
  - ▶ Nesse caso, se nenhuma condição for verdadeira, o bloco do *else* será executado

```
...  
if CONDIÇÃO:  
    INSTRUÇÃO 1  
    ...  
    INSTRUÇÃO N  
elif CONDIÇÃO:  
    INSTRUÇÃO 1  
    ...  
    INSTRUÇÃO N  
elif CONDIÇÃO:  
    INSTRUÇÃO 1  
    ...  
    INSTRUÇÃO N  
else:  
    ...
```



# Exemplo de *if... else*

---

## ► Programa para somar ou multiplicar dois números

```
op = input("Deseja somar (S) ou multiplicar (M)?")
x = int(input("Digite o primeiro numero:"))
y = int(input("Digite o segundo numero:"))
if (op == "S"):
    r = x + y
    print("O resultado da soma é", r)
elif (op == "M"):
    r = x * y
    print("O resultado da multiplicação é", r)
else:
    print("Opção inválida")
```



## Exemplo de *if... else*

- ▶ Programa para somar ou multiplicar dois números

```
op = input("Deseja somar (S) ou multiplicar (M)?")
x = int(input("Digite o primeiro numero:"))
y = int(input("Digite o segundo numero:"))
if (op == "S"):
    r = x + y
    print("O resultado é:", r)
elif (op == "M"):
    r = x * y
    print("O resultado é:", r)
else:
    print("Opção inválida")
```

Problema: x e y serão lidos mesmo se a opção for inválida

# Exemplo de *if... else*

---

## ► Programa para somar ou multiplicar dois números

```
op = input("Deseja somar (S) ou multiplicar (M)?")
if (op == "S"):
    x = int(input("Digite o primeiro numero:"))
    y = int(input("Digite o segundo numero:"))
    r = x + y
    print("O resultado da soma é", r)
elif (op == "M"):
    x = int(input("Digite o primeiro numero:"))
    y = int(input("Digite o segundo numero:"))
    r = x * y
    print("O resultado da multiplicação é", r)
else:
    print("Opção inválida")
```



## Solução mais elegante: faz a leitura de x e y uma única vez

---

### ► Programa para somar ou multiplicar dois números

```
op = input("Deseja somar (S) ou multiplicar (M)?")
if (op == "S" or op == "M"):
    x = int(input("Digite o primeiro numero:"))
    y = int(input("Digite o segundo numero:"))
if (op == "S"):
    r = x + y
    print("O resultado da soma é", r)
elif (op == "M"):
    r = x * y
    print("O resultado da multiplicação é", r)
else:
    print("Opção inválida")
```



# Exemplo

---

## ► Programa para informar o número de dias de um mês qualquer

```
mes = int(input('Entre com um mês (1 a 12): '))
if (mes==1) or (mes==3) or (mes==5) or (mes==7) or (mes==8) or (mes==10) or (mes==12):
    print('Esse mes tem 31 dias')
elif (mes==4) or (mes==6) or (mes==9) or (mes==11):
    print('Esse mes tem 30 dias')
elif (mes==2):
    ano = int(input('Entre com o ano (4 dígitos): '))
    if (ano % 400 == 0) or (ano % 4 == 0) and (ano % 100 != 0):
        print('Esse mes tem 29 dias')
    else:
        print('Esse mes tem 28 dias')
else:
    print('Mês inválido')
```



# Uso de variáveis booleanas

---

```
imprimeMensagem = True
n = int(input("Digite um numero: "))
if (imprimeMensagem):
    print("O numero digitado foi", n)
else:
    print(n)
```





# Uso de variáveis booleanas

---

```
imprimeMensagem = True
n = int(input("Digite um numero: "))
if (imprimeMensagem):
    print("O numero digitado foi", n)
else:
    print(n)
```

Note que NÃO usei  
**if (imprimeMensagem == True):**  
pois seria redundante!



## Uso de not

---

```
imprimeMensagem = True
n = int(input("Digite um numero: "))
if not(imprimeMensagem):
    print(n)
else:
    print("O numero digitado foi", n)
```



# Condições Simplificadas

---

- ▶ Python permite simplificar condições, adicionando um AND implicitamente

```
a = int(input('Digite um numero: '))
b = int(input('Digite um numero: '))
c = int(input('Digite um numero: '))
if (a == b == c):
    print('Os 3 números são iguais')
else:
    print('Os 3 números não são iguais')
```



# Condições Simplificadas

---

- ▶ Python permite simplificar condições, adicionando um AND implicitamente

```
a = int(input('Digite um numero: '))
b = int(input('Digite um numero: '))
c = int(input('Digite um numero: '))
if (a == b == c):
    print('Os 3 números são iguais')
else:
    print('Os 3 números não são iguais')
```



**Isso equivale à condição**  
(a == b and b == c)

# Condições Simplificadas

- ▶ Python permite simplificar condições, adicionando um AND implicitamente

```
a = int(input('Digite um numero: '))
b = int(input('Digite um numero: '))
c = int(input('Digite um numero: '))
if (a == b == c):
    print('Os 3 números são iguais')
else:
    print('Os 3 números não são iguais')
```

Nada se pode  
afirmar sobre  
a == c

Isso equivale à condição  
(a == b and b == c)

# Condições Simplificadas

---

- ▶ Para igualdades isso pode ser garantido por **transitividade**, mas operadores não **transitivos** apresentam problema

```
a = int(input('Digite um numero: '))
b = int(input('Digite um numero: '))
c = int(input('Digite um numero: '))
if (a != b != c):
    print('Os 3 números são diferentes')
else:
    print('Os números são iguais')
```



# Condições Simplificadas

- ▶ Para igualdades isso pode ser garantido por **transitividade**, mas operadores não **transitivos** apresentam problema

```
a = int(input('Digite um numero: '))
b = int(input('Digite um numero: '))
c = int(input('Digite um numero: '))
if (a != b != c):
    print('Os 3 números são diferentes')
else:
    print('Os números são iguais')
```

Nesse caso  
não é possível  
garantir que  
 $a \neq c$

Operador  $\neq$  não é transitivo

# Condições Simplificadas

---

- ▶ Solução: não usar condição simplificada nesses casos

```
a = int(input('Digite um numero: '))
b = int(input('Digite um numero: '))
c = int(input('Digite um numero: '))
if (a != b and b != c and a != c):
    print('Os 3 números são diferentes')
else:
    print('Os números são iguais')
```





# Escopo de variáveis

---

- ▶ Variável só é visível dentro do seu “escopo”
- ▶ Variável declarada (usada pela primeira vez) fora de um bloco
  - ▶ Pode ser acessada e modificada de qualquer lugar
- ▶ Variável declarada (usada pela primeira vez) dentro de um bloco
  - ▶ Só existe se esse bloco for executado
- ▶ Revisitaremos esse assunto mais adiante na disciplina



## Exemplo com Erro

---

```
nome = input('Digite o nome da pessoa: ')\nsexo = input('Digite o sexo da pessoa (F/M): ')\nif (sexo == 'M'):\n    idade = input('Digite a idade da pessoa: ')\nprint(nome, 'tem', idade, 'anos')
```



## Exemplo com Erro

---

```
nome = input('Digite o nome da pessoa: ')\nsexo = input('Digite o sexo da pessoa (F/M): ')\nif (sexo == 'M'):\n    idade = input('Digite a idade da pessoa: ')\nprint(nome, 'tem', idade, 'anos')
```



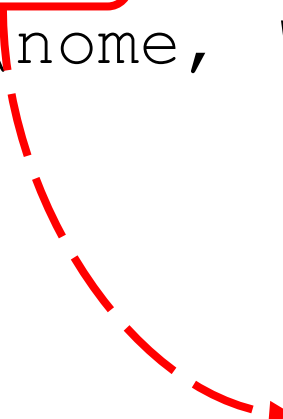
**nome e sexo** podem ser acessadas  
em qualquer lugar do programa



## Exemplo com Erro

---

```
nome = input('Digite o nome da pessoa: ')\nsexo = input('Digite o sexo da pessoa (F/M): ')\nif (sexo == 'M'):\n    idade = input('Digite a idade da pessoa: ')\nprint(nome, 'tem', idade, 'anos')
```

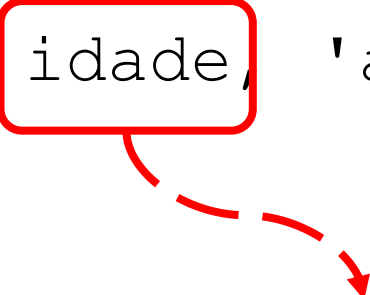


**idade** só existe se o código dentro  
do `if` for executado

## Exemplo com Erro

---

```
nome = input('Digite o nome da pessoa: ')\nsexo = input('Digite o sexo da pessoa (F/M): ')\nif (sexo == 'M'):\n    idade = input('Digite a idade da pessoa: ')\nprint(nome, 'tem', idade, 'anos')
```



Se **sexo** for **F**, esse comando dará erro,  
pois variável **idade** não terá sido criada  
pelo Python



# Exercícios

---

- ▶ Faça um programa que calcule o IMC de uma pessoa ( $\text{IMC} = \text{massa em kg} / \text{altura em metros elevado ao quadrado}$ ) e informe a sua classificação segundo a tabela a seguir, obtida na Wikipédia

IMC	Classificação
< 18,5)	Abaixo do Peso
[18,5 – 25)	Saudável
[25 – 30)	Peso em excesso
[30 – 35)	Obesidade Grau I
[35 – 40)	Obesidade Grau II (severa)
>= 40	Obesidade Grau III (mórbida)



# Exercícios

---

- ▶ Faça um programa que leia três coordenadas num espaço 2D e indique se formam um triângulo, juntamente com o seu tipo (equilátero, isósceles e escaleno)
  - ▶ Equilátero: todos os lados iguais
  - ▶ Isósceles: dois lados iguais
  - ▶ Escaleno: todos os lados diferentes
- ▶ DICA: Condição de existência de triângulo de lados  $a$ ,  $b$  e  $c$ :
  - ▶  $|b - c| < a < b + c$
  - ▶  $|a - c| < b < a + c$
  - ▶  $|a - b| < c < a + b$



# Exercícios

---

- ▶ Faça um programa que leia um número inteiro de 5 dígitos e indique se ele é palíndromo
  - ▶ Um número palíndromo é aquele que se lido da esquerda para a direita ou da direita para a esquerda possui o mesmo valor (ex.: 15451)





# Exercícios

---

- ▶ Faça um programa que leia um número inteiro entre 0 e 99 e escreva o seu valor por extenso



# Vocês já podem ler

---

- ▶ Capítulo I do livro Use a Cabeça: Programação, até a página 25



# Referências

---

- ▶ Slides baseados no curso dos Professores Leonardo Murta e Vanessa Braganholo, Instituto de Computação – Universidade Federal Fluminense





# [Aula 2-B] Linguagem de Programação

## Análise e Desenvolvimento de Sistemas

Operadores e Estrutura de Decisão – Prof. Jean Zahn

[jeanozahn@gmail.com](mailto:jeanozahn@gmail.com)