



Engenharia de Software

Computacional thinking with Python

Prof. Dr. Francisco Elânio

Resolução dos Exercícios da Aula passada

Resolução dos Exercícios

Ex. 01

Crie um algoritmo para calcular a energia em joules utilizando a equação da teoria da relatividade de Albert Einstein.

$E = m \times c$, em que:

E é energia (J)

m é massa (kg)

c é velocidade da luz - 300000 km/h

Resolução dos Exercícios

Ex. 01

```
m = float(input("Inserir valor da massa em kg: "))
```

→ Condição para faixa adolescente

```
energia = (m * 3 * pow(10,8)) / 1000000
```

→ Calculo da energia

```
print("A energia resultante é de", energia, "Mjoules")
```

→ Saída da variável energia

Resolução dos Exercícios

Ex. 02

Agora crie um algoritmo que imprima a mensagem criança, adolescente, adulto-jovem, meia-idade ou idoso de acordo com as seguintes faixas de idade:

Criança - nascimento até os 11 anos de idade

Adolescência - entre 12 e 20 anos

Adulto-jovem - entre 21 e 40 anos

Meia-idade - entre 40 e 65 anos

Idoso - acima dos 65 anos de idade

Resolução dos Exercícios

Ex. 02

```
idade = int(input("Digite a idade: "))  
if idade <= 11:  
    print("crianca")  
elif idade >= 12 and idade <= 20:  
    print("adolescente")  
elif idade >= 21 and idade <= 40:  
    print("adulto_jovem")  
elif idade >= 41 and idade <= 65:  
    print("meia_idade")  
else:  
    print("Idoso")
```

→ Digitar idade

→ Condição para faixa criança

→ Condição para faixa adolescente

→ Condição para jovem adulto

→ Condição para meia idade

→ Condição para faixa idoso

Resolução dos Exercícios

Ex. 03

Uma determinada faculdade utiliza quatro atividades para avaliar o desempenho do seu aluno. No entanto, para verificar o seu desempenho no semestre utiliza apenas as três maiores notas. Crie um algoritmo para calcular a média deste aluno.

Resolução dos Exercícios

Ex. 03

Inserir quatro notas

```
nota1 = float(input("Digite a primeira nota: "))  
nota2 = float(input("Digite a segunda nota: "))  
nota3 = float(input("Digite a terceira nota: "))  
nota4 = float(input("Digite a quarta nota: "))
```

Calcular a média descartando a menor nota

```
notas = [nota1, nota2, nota3, nota4]  
nota_minima = min(notas)  
notas.remove(nota_minima)  
media = sum(notas) / 3  
  
print(f"A média final do aluno é: {media:.2f}")
```


Resolução dos Exercícios

Ex. 04

Crie um algoritmo para calcular a potência em watts de um motor em corrente contínua de acordo com as equações abaixo. Repare que o usuário pode inserir tensão V e corrente I , ou Resistência R e Corrente I , ou Tensão V e Resistência R .

$$P = V \times I$$

$$P = R \times I \times I$$

$$P = V^2 / R$$

Resolução dos Exercícios

Ex. 04

```
corrente = float(input("Inserir valor de corrente: "))
resistencia = float(input("Inserir valor de resistência: "))
tensao = float(input("Inserir valor de tensão: "))

print("valor da corrente", corrente)
print("valor da resistência", resistencia)
print("valor de tensão", tensao)

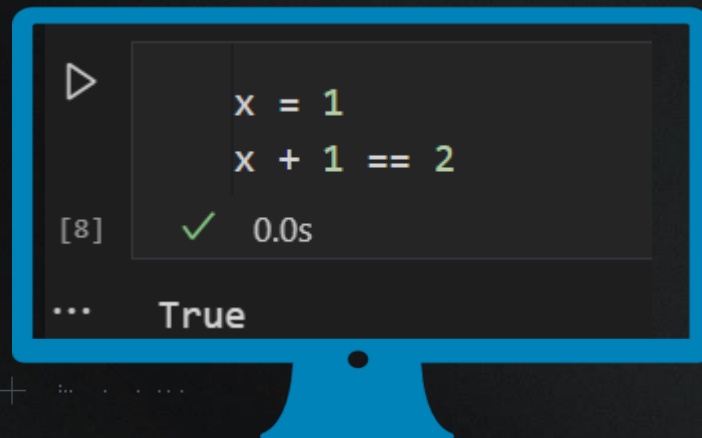
if tensao == 0 and corrente == 0:
    print("Não é possível calcular a potência, pois tensão e corrente são iguais a zero.")
elif corrente == 0 and resistencia == 0:
    print("Não é possível calcular a potência, pois corrente e resistência são iguais a zero.")
elif resistencia == 0 and tensao == 0:
    print("Não é possível calcular a potência, pois resistência e tensão são iguais a zero.")
elif tensao == 0:
    print((corrente ** 2) * resistencia)
elif corrente == 0:
    print((tensao ** 2) / resistencia)
elif resistencia == 0:
    print(tensao * corrente)
else:
    print("Valores de corrente, resistência e tensão não podem ser todos zero.")
```

**Conceitos de variáveis
e atribuições, operadores lógicos, operações com
strings, sequências e tempo, entrada e conversão
de entrada de dados**

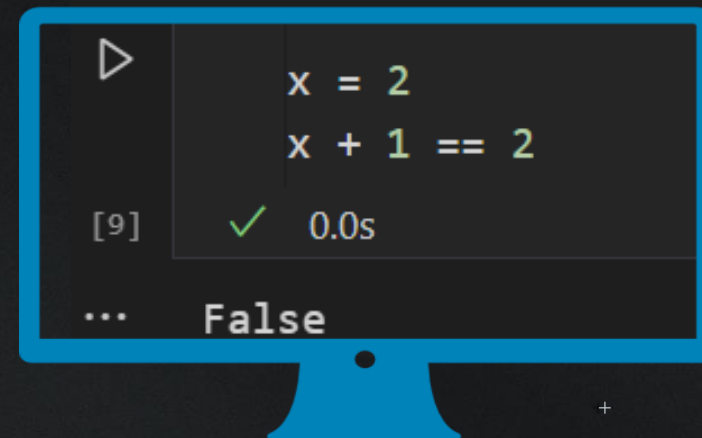
Conceitos de variáveis e atribuições

Em matemática, aprendemos o conceito de variável para representar incógnitas em equações do tipo $x + 1 = 2$, nas quais devemos determinar o valor de x , resolvendo a equação.

Em programação, variáveis são utilizadas para armazenar valores e para dar nome a uma área de memória do computador onde armazenamos dados.



```
▶ x = 1
  x + 1 == 2
[8] ✓ 0.0s
... True
```



```
▶ x = 2
  x + 1 == 2
[9] ✓ 0.0s
... False
```


Conceitos de variáveis e atribuições

O comando de atribuição altera o valor armazenado em uma variável. O valor atribuído torna-se o novo valor da variável, substituindo o anterior. Quando lermos nosso programa, as operações de atribuição serão chamadas de "recebe"; ou seja, uma variável recebe um valor.



```
x = 2
```

```
y = 4
```

```
z = 6
```



```
a = x + y
```

```
b = x + z
```

```
c = y + z
```



```
print(x + y)
```

```
print(x + z)
```

```
print(y + z)
```

Exercícios

1. Escreva um programa que exiba o resultado de $2a \times 3b$, em que a vale 3 e b vale 5.
2. Escreva um programa que calcule a soma de três variáveis e imprima o resultado na tela.
3. Modifique de aumento de salário, de forma que ele calcule um aumento de 15% para um salário de R\$ 750.

Conceitos de variáveis e atribuições

```
salario = 1500
```

```
aumento = 5
```

```
print(salario + (salario * aumento / 100))
```

```
print(1500 + (1500 * 5 / 100 ))
```

Variável salário

Variável aumento

Comando de saída

Forma mais direta

Variáveis Numéricas

Quando armazena números inteiros ou de ponto flutuante.

Os números inteiros são aqueles sem parte decimal, como 1, 0, -5, 550, -47, 30000.

Números de ponto flutuante ou decimais são aqueles com parte decimal, como 1.0, 5.478, 10.478, 30000.4.

Observe que 1.0, mesmo tendo zero na parte decimal, é um número de ponto flutuante.

Nome das Variáveis

Nome	Válido	Comentários
A1	Sim	Embora contenha um número, o nome a1 inicia com letra.
Velocidade	Sim	Nome formado por letras.
Velocidade90	Sim	Nome formado por letras e números, mas iniciado por letra.
Salário_médio	Sim	O símbolo sublinha U é permitido e facilita a leitura de nomes grandes.
Salário médio	Não	Nomes de variáveis não podem conter espaços em branco.
_b	Sim	O sublinha U é aceito em nomes de variáveis, mesmo no início.
1a	Não	Nomes de variáveis não podem começar com números.

Variáveis do tipo numérica

Base 10

$$\begin{aligned} &5 \times 10^2 + 3 \times 10^1 + 1 \times 10^0 \\ &5 \times 100 + 3 \times 10 + 1 \times 1 \\ &500 + 30 + 1 \\ &531 \end{aligned}$$

Python

$(5 * 10^{**}2) + (3 * 10^{**}1) + (1 * 10^{**}0)$

$5 * \text{pow}(10,2) + 3 * \text{pow}(10,1) + 1 * \text{pow}(10,0)$

Base 2

$$\begin{aligned} &1 \times 2^2 + 0 \times 2^1 + 1 \times 10^0 \\ &4 + 0 + 1 \\ &5 \end{aligned}$$

Python

$(1 * 2^{**}2) + (0 * 2^{**}1) + (1 * 10^{**}0)$

$1 * \text{pow}(2,2) + 0 * \text{pow}(2,1) + 1 * \text{pow}(2,0)$

Variáveis do tipo lógico

Verdadeiro ou Falso: variável chamado tipo lógico ou booleano. Em Python, escreveremos True para verdadeiro e False para falso. Observe que o T e o F são escritos com letras maiúsculas:

Resultado = True

Aprovado = False

Operadores Relacionais

Operador	Operação	Símbolo matemático
==	Igualdade	=
>	Maior que	>
<	Menor que	<
!=	Diferente	≠
>=	Maior ou Igual	≥
<=	Menor ou Igual	≤

Variáveis do tipo lógico

Também podem ser utilizadas para armazenar o resultado de expressões e comparações:

```
nota = 8
```

```
média = 7
```

```
aprovado = nota > média
```

```
print(aprovado )
```

Se uma expressão contém operações aritméticas, estas devem ser calculadas antes que os operadores relacionais sejam avaliados.

Operadores Lógicos

Python suporta três operadores básicos:
not (não), and (e), or (ou).

Esses operadores devem ser traduzidos
como não (negação), e (conjunção) e ou
(disjunção).

Operador Python	Operação
Not	Não
And	E
Or	Ou

V1	Não V1
V	F
F	V

V1	V2	V1 E V2
V	V	V
V	F	F
F	V	F
F	F	F

V1	V2	V1 OU V2
V	V	V
V	F	V
F	V	V
F	F	F

Exercício

Complete a tabela a seguir em Python utilizando `a = True`, `b = False` e `e = True`

Expressão	Resultado
A e A	
B e B	
Não C	
Não B	
Não A	
A e B	
B e C	

Expressão	Resultado
A ou C	
B ou C	
C ou A	
C ou B	
C ou C	
B ou B	
-	-

Expressões Lógicas

Os operadores lógicos podem ser combinados em expressões lógicas mais complexas. Quando uma expressão tiver mais de um operador lógico, avalia-se o operador not (não) primeiro, seguido do operador and (e) e, finalmente, or (ou).

True or False and not True

True or False and False

True or False

True

Expressões Lógicas

Os operadores relacionais também podem ser utilizados em expressões com operadores lógicos.

Salário > 1000 and idade > 18

Esses casos, os operadores relacionais devem ser avaliados primeiro.

Expressões Lógicas

Exemplos

Façamos salário = 100 e idade = 20

Salário > 1000 and idade > 18

100 > 1000 and 20 > 18

False and True

False

Façamos salário = 1200 e idade = 20

Salário > 1000 and idade > 18

1200 > 1000 and 20 > 18

True and True

True

Expressões Lógicas

Exercício 1

Escreva uma expressão para determinar se uma pessoa deve ou não pagar imposto. Considere que pagam imposto pessoas cujo salário é maior que R\$ 1.200,00.

Crie o algoritmo Pseudocódigo, Fluxograma e código em Python

Código em Python

```
# Salário da pessoa
salario = float(input("Digite o salário da pessoa: "))

# Verifica se o salário é maior que R$ 1.200,00
if salario > 1200.00:
    print("A pessoa deve pagar imposto.")
else:
    print("A pessoa está isenta de pagar imposto.")
```

Expressões Lógicas

Exercício 2

Escreva uma expressão que será utilizada para decidir se um aluno foi ou não aprovado. Para ser aprovado, todas as médias do aluno devem ser maiores que 7. Considere que o aluno cursa apenas três matérias, e que a nota de cada uma está armazenada nas seguintes variáveis: Matéria1, Matéria2 e Matéria3.

Código em Python

```
# Notas do aluno
```

```
Materia1 = float(input("Digite a nota da Matéria 1: "))
```

```
Materia2 = float(input("Digite a nota da Matéria 2: "))
```

```
Materia3 = float(input("Digite a nota da Matéria 3: "))
```

```
# Verifica se todas as notas são maiores que 7
```

```
if Materia1 > 7 and Materia2 > 7 and Materia3 > 7:
```

```
    print("O aluno foi aprovado em todas as matérias.")
```

```
else:
```

```
    print("O aluno não foi aprovado em todas as matérias.")
```

Variáveis do tipo string

Armazenam cadeias de caracteres como nomes e textos. Chamamos cadeia de caracteres uma sequência de símbolos como números, sinais de pontuação etc.

String

J o ã o e M a r i a c o m e m p ã o

Função len em Python

A função len retorna um valor do tipo inteiro, representando a quantidade de caracteres contidos na string.

```
len("joão e maria comem pão")
```

22 caracteres

```
len("O rato roeu a roupa")
```

19 caracteres

```
len("Nosso país é muito bonito,  
porém precisa evoluir")
```

48 caracteres

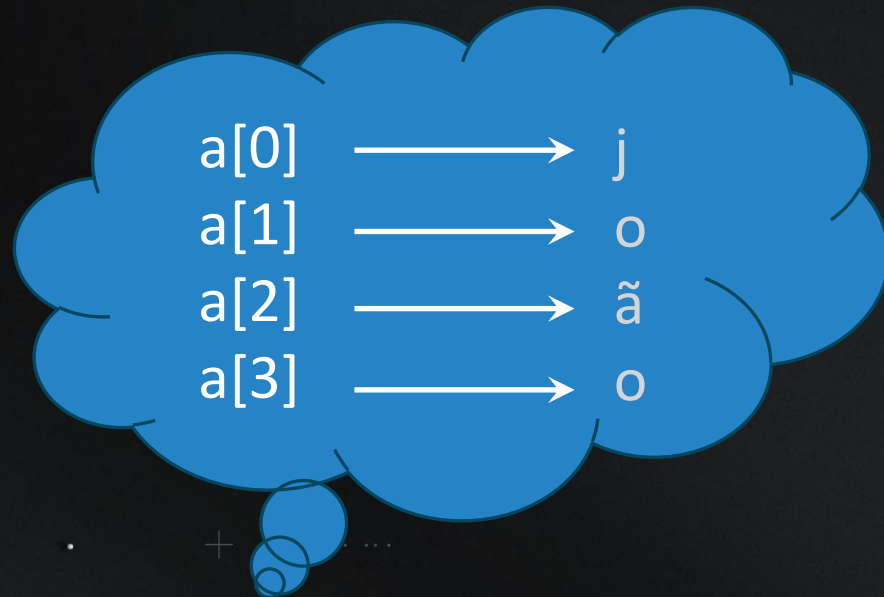
```
len("A Covid 19 trouxe danos  
catastróficos para a raça  
humana")
```

56 caracteres

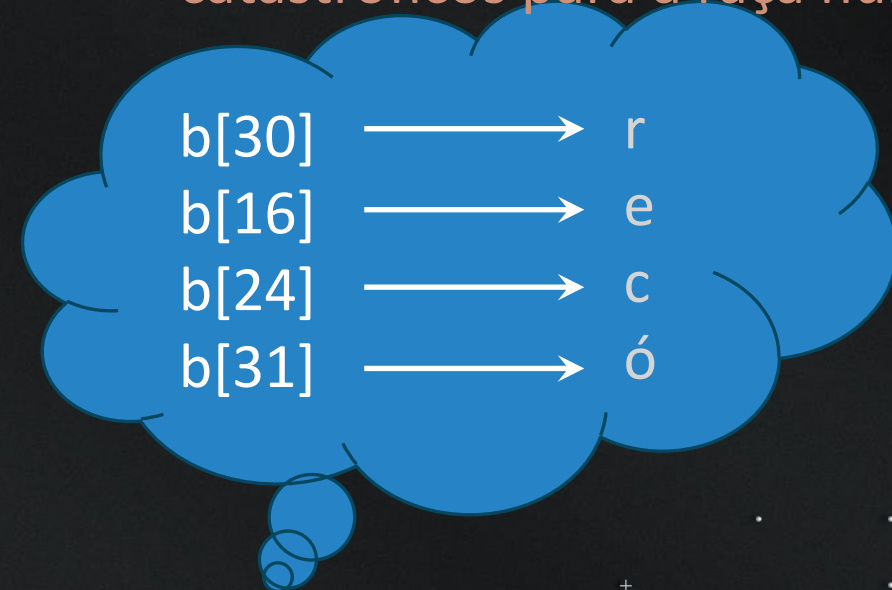
Acessar os caracteres de uma string

Para acessar os caracteres de uma string, devemos informar o índice ou posição do caractere entre colchetes ([]).

a = ("joão e maria comem pão")



b = ("A Covid 19 trouxe danos
catastróficos para a raça humana")

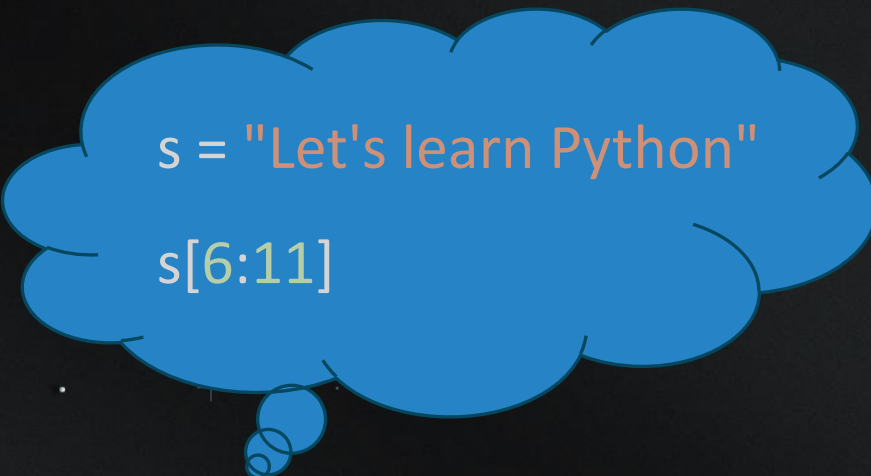


Operações com strings

Fatiamento

As variáveis de tipo string suportam operações como fatiamento, concatenação e composição.

Por fatiamento, podemos entender a capacidade de utilizar apenas uma parte de uma string, ou uma fatia.



```
s = "Let's learn Python"  
s[6:11]
```



```
s = "Let's learn Python"  
s[6:15]
```

Operações com strings

Fatiamento

Omitir os caracteres da esquerda

```
s = "Let's learn Python"
```

```
s[10:]
```

n Python

```
s = "Let's learn Python"
```

```
s[-2:]
```

on

Omitir os caracteres da direita

```
s = "Let's learn Python"
```

```
s[:10]
```

Let's lear

```
s = "Let's learn Python"
```

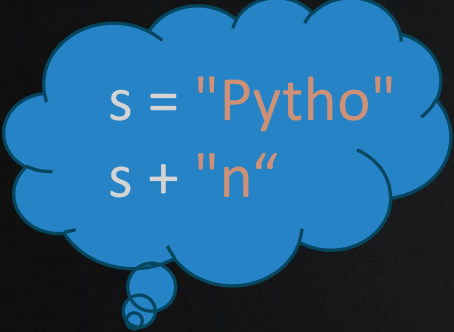
```
s[:-2]
```

Let's learn Pyth

Operações com strings

Concatenação

A concatenação nada mais é que poder juntar duas ou mais strings em uma nova string maior.



```
s = "Pytho"  
s + "n"
```

Python



```
s = "Python "  
s + "é " + "impressionante"
```

Python é impressionante



```
s = "Python "  
t = "é "  
u = "impressionante"
```

s + t + u

Operações com strings

Composição

Exibir que "João tem X anos ", em que X é uma variável numérica. Usando a composição de strings do Python, podemos escrever de forma simples e clara: "João tem %d anos "% X

Marcador	Resultado
%d	Números inteiros
%s	Strings
%f	Números decimais

"%1f" % 5

"%2f" % 5

"%.1d" % 3

"%.2d" % 3

"%.3d" % 3

"%s" % sun

"%s" % mon

"%s" % day

"%s" % week

"%1.2f" % month

Operações com strings

Composição – Ex1

João tem 22 anos e apenas R\$51.340000 no bolso

`"%s tem %d anos e apenas R$%f no bolso"% ("João", 22, 51.34)`

decimal

string

float

Operações com strings

Composição – Ex2

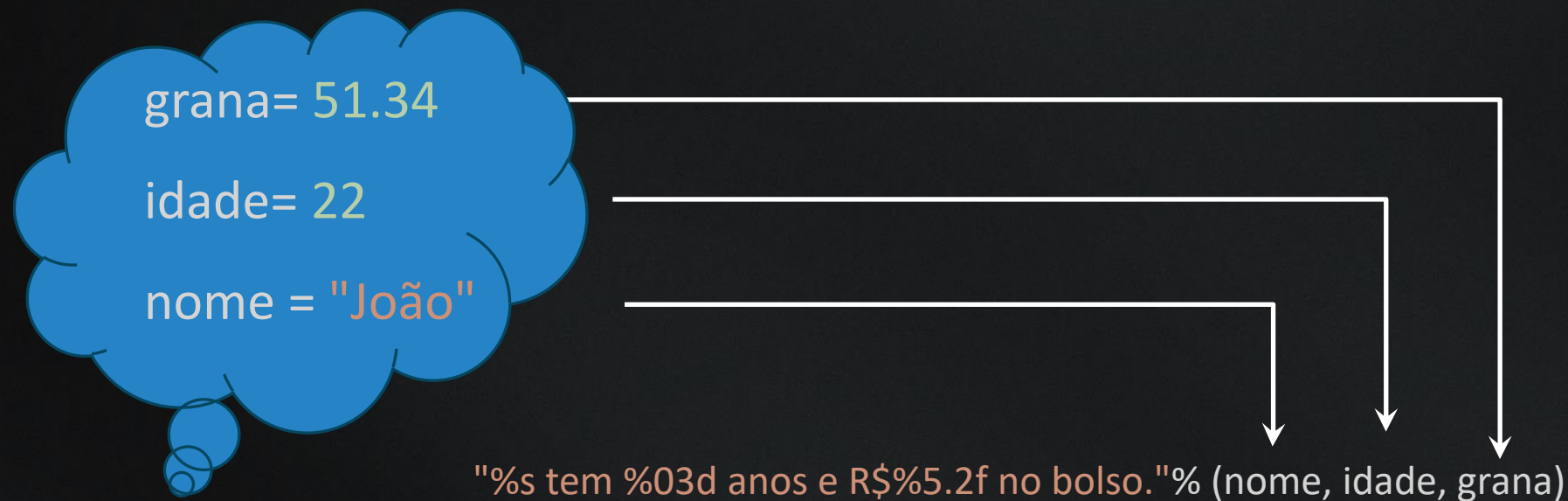
Para este exemplo foram criadas as variáveis nome, idade e grana.



Operações com strings

Composição – Ex3

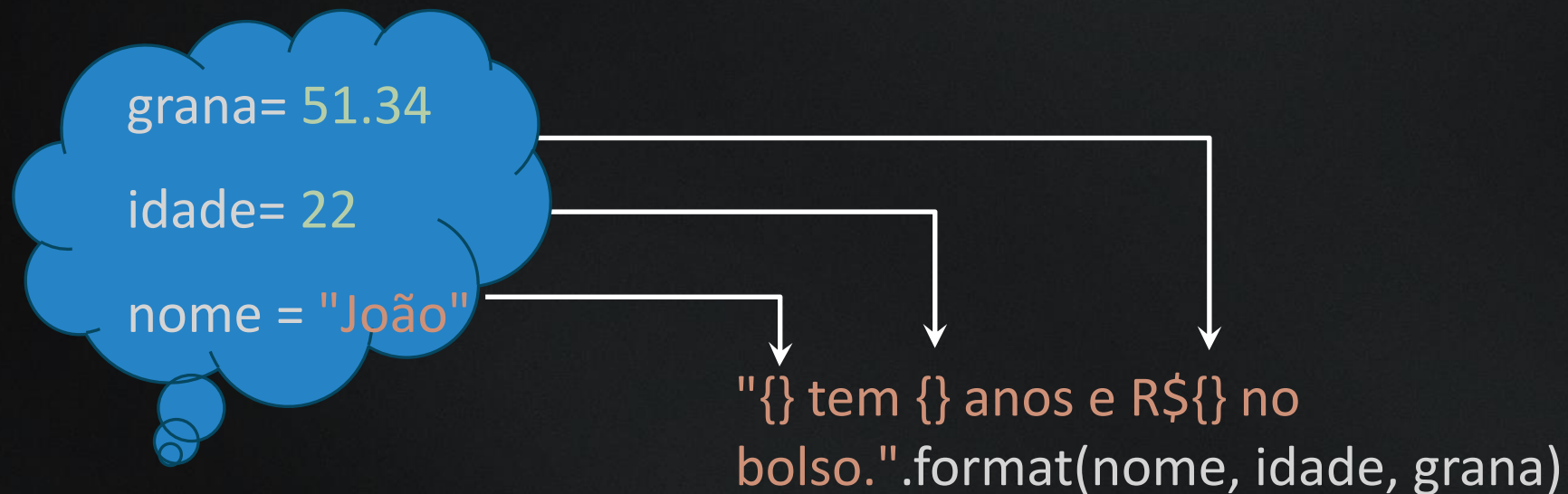
Para este exemplo foram criadas as variáveis nome, idade e grana.



Operações com strings

Composição – Ex5

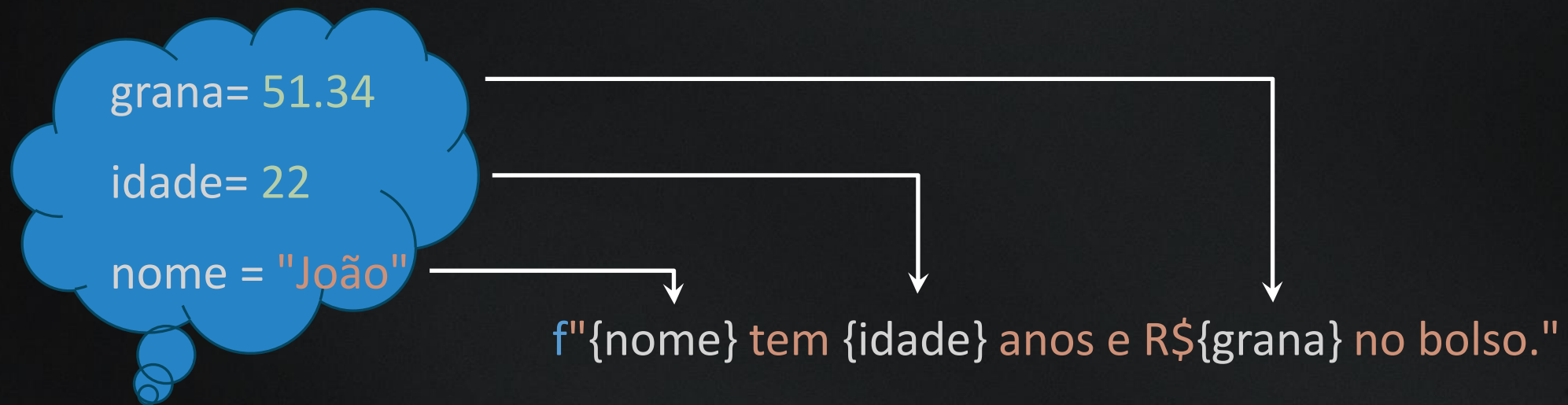
Para este exemplo foram criadas as variáveis nome, idade e grana.



Operações com strings

Composição – Ex5

Para este exemplo foram criadas as variáveis nome, idade e grana.



Sequências e Tempo

Um programa é executado linha por linha pelo computador.

O conteúdo de uma variável pode mudar com o tempo. A cada vez que alteramos o valor de uma variável, o valor anterior é substituído pelo novo.

Tempo	Sequência
1	dívida = 0
2	compra = 100
3	dívida = dívida + compra
4	compra = 200
5	dívida = dívida + compra
6	compra = 300
7	dívida = dívida + compra
8	compra = 0
9	print(dívida)

Código em Python

```
# Inicializa a dívida
divida = 0
print("Dívida Inicial:", divida)

# Define a primeira compra
compra = 100
print("Compra:", compra)

# Adiciona o valor da primeira compra à dívida
divida = divida + compra
print("Dívida após a primeira compra:", divida)

# Define a segunda compra
compra = 200
print("Compra:", compra)

# Adiciona o valor da segunda compra à dívida
divida = divida + compra
print("Dívida após a segunda compra:", divida)
```

```
# Define a terceira compra
compra = 300
print("Compra:", compra)

# Adiciona o valor da terceira compra à dívida
divida = divida + compra
print("Dívida após a terceira compra:", divida)

# Define uma compra com valor zero
compra = 0
print("Compra:", compra)

# Adiciona o valor da compra com valor zero à dívida
divida = divida + compra
print("Dívida após a compra com valor zero:", divida)

# Imprime o valor total da dívida
print("Dívida total:", divida)
```

Entrada de Dados

A função `input` é utilizada para solicitar dados do usuário. Ela recebe um parâmetro, que é a mensagem a ser exibida, e retorna o valor digitado pelo usuário.

```
x = input( "Digite um número: ")  
print(x)
```

```
nome = input( "Digite seu nome:")  
print(f"Você digitou {nome}")  
print(f"Olá, {nome}!")
```


Conversão da Entrada de Dados

A função `input` sempre retorna valores do tipo `string`, ou seja, não importa e digitamos apenas números, o resultado sempre é `string`.

Observação

A função `int` converte o valor retornado em um número inteiro, e a função `float` para convertê-lo em número decimal ou de ponto flutuante.

```
anos = int(input("Anos de serviço: "))  
valor_por_ano = float(input("Valor por ano: "))  
bônus = anos * valor_por_ano  
print(f"Bônus de R$ {bônus:5.2f}")
```

Bônus de R\$ 250.00

Exercícios

Desenvolva os algoritmos em Python

1. *Escreva um programa que leia um valor em metros e o exiba convertido em milímetros.*
2. *Escreva um programa que leia a quantidade de dias, horas, minutos e segundos do usuário. Calcule o total em segundos.*
3. *Faça um programa que solicite o preço de uma mercadoria e o percentual de desconto. Exiba o valor do desconto e o preço a pagar.*
4. *Escreva um programa que calcule o tempo de uma viagem de carro. Pergunte a distância a percorrer e a velocidade média esperada para a viagem.*

Exercícios

Desenvolva os algoritmos em Python

5. *Escreva um programa que converta uma temperatura digitada em °C em °F. A fórmula para essa conversão é:*

$$F = ((9 \times C) / 5) + 32$$

6. *Escreva um programa que pergunte a quantidade de km percorridos por um carro alugado pelo usuário, assim como a quantidade de dias pelos quais o carro foi alugado. Calcule o preço a pagar, sabendo que o carro custa R\$ 60 por dia e R\$ 0,15 por km rodado.*

*O importante é não parar
de questionar (Einstein)*

Referências

ASCENCIO, A. F. G, CAMPOS, E. A. V. Fundamentos da Programação de Computadores: algoritmos, Pascal, C/C++ e Java, 2ª Edição, São Paulo: Pearson 2007.

SALVETTI, Dirceu Douglas; BARBOSA, Lisbete Madson. Algoritmos. São Paulo: Pearson, 2004.