Processamento da Informação
– Teoria –

Algoritmos
e
Tipos de dados

Semana 01
Prof. Jesús P. Mena-Chalco

24/04/2013

(*) Slides adaptados das aulas do Prof. Harlen Costa Batagelo

Algumas definições de algoritmo

- **Sequência ordenada de passos** que deve ser seguida para a realização de uma tarefa (ASCENCIO, 1999).

- **Regras formais** para a obtenção de um resultado ou **solução de um problema**, englobando formulas de expressões aritméticas (MANZANO, 1997).

- **Sequência finita de instruções ou operações** cuja execução, em tempo finito, resolve um problema computacional, qualquer que seja sua instância (SALVETTI, 1999)

Exemplo de algoritmo

Sacar dinheiro de um caixa eletrônico (2012):

(1) Insira o cartão eletrônico.

(2) Digite a senha.

(3) Selecione a opção “saque”.

(4) Digite a quantidade desejada.

(5) Se não houver saldo suficiente, continue no passo (7).

(6) Retire o dinheiro

(7) Retire o cartão do caixa eletrônico

Exemplo de algoritmo

Sacar dinheiro de um caixa eletrônico (2012):

(1) Insira o cartão eletrônico.

(2) Digite a senha.

(3) Selecione a opção “saque”.

(4) Digite a quantidade desejada.

(5) Se não houver saldo suficiente, continue no passo (7).

(6) Retire o dinheiro

(7) Retire o cartão do caixa eletrônico

Exemplo de algoritmo

Sacar dinheiro de um caixa eletrônico (2012):

(1) Insira o cartão eletrônico.

(2) Digite a senha.

(3) Selecione a opção “saque”.

(4) Digite a quantidade desejada.

(5) Se não houver saldo suficiente, continue no passo (7).

(6) Retire o dinheiro

(7) Retire o cartão do caixa eletrônico

Exemplo de algoritmo

Sacar dinheiro de um caixa eletrônico (2015):

(1) ~~Insira o cartão.~~ Use o sensor biométrico (iris, face, digitais).

(2) ~~Digite a senha.~~

(3) Selecione a opção “saque”.

(4) Digite a quantidade desejada.

(5) Se não houver saldo suficiente, continue no passo (7).

(6) Retire o dinheiro

(7) ~~Retire o cartão.~~ Transação finalizada.

Exemplo de algoritmo

A sequência de passos **depende do tipo do tecnologia utilizada.**

A sequência de passos **é limitada pela quantidade finita de possíveis operações.**

Exemplo de algoritmo

Calcular (a+b) * (a+b):

(1) Multiplique a*a.

(2) Multiplique a*b.

(3) Multiplique b*a.

(4) Multiplique b*b.

(5) Some os resultados obtidos nos passos anteriores.

Exemplo de algoritmo

Calcular (a+b) * (a+b):

(1) Multiplique a*a.

(2) ~~Multiplique a*b.~~

(3) Multiplique b*a.

(4) Multiplique b*b.

(5) Some os resultados obtidos nos passos anteriores, da seguinte forma: a*a + 2(b*a) + b*b

Exemplo de algoritmo

Verificar se o número x é par ou impar:

Exemplo de algoritmo

Verificar se o número x é par ou impar:

(1) Se o resto da divisão de x por 2 for igual a 0, então

O número é par

(2) Senão, então

O número é impar

O resto da divisão:
3/2 = 1
4/2 = 0
7/2 = 1
17/2 = 1
26/2 = 0

Algoritmos e programas

Para que um computador desempenhe uma tarefa é necessário que uma sequência de ações **(algoritmo)** seja especificada de uma forma compreensível pela máquina.

Um programa de computador nada mais é que um algoritmo escrito de forma compreensível pelo computador.

Ações especificadas de forma “formal”

Algoritmo

Mostrar uma sequência de números de 1 até n:

(1) Peça o valor de n.

(2) Faça a = 1.

(3) Mostrar na tela o valor de a.

(4) Adicione 1 ao valor de a.

(5) Se a é menor ou igual a n, então volte para o passo (3).

Programa na linguagem Basic

```
10   REM MOSTRA SEQUENCIA DE NUMEROS de 1 A N
20   INPUT N
30   A = 1
40   PRINT A
50   A = A + 1
60   IF A <= N THEN GOTO 40
```

Programa na linguagem C

```
/* Mostra uma sequencia de numeros de 1 a n */
main()
{
    int a, n;
    scanf("%d", &n);
    for(a = 1; a <= n; a++)
    {
        printf("%d", a);
    }
}
```

Programa na linguagem Java

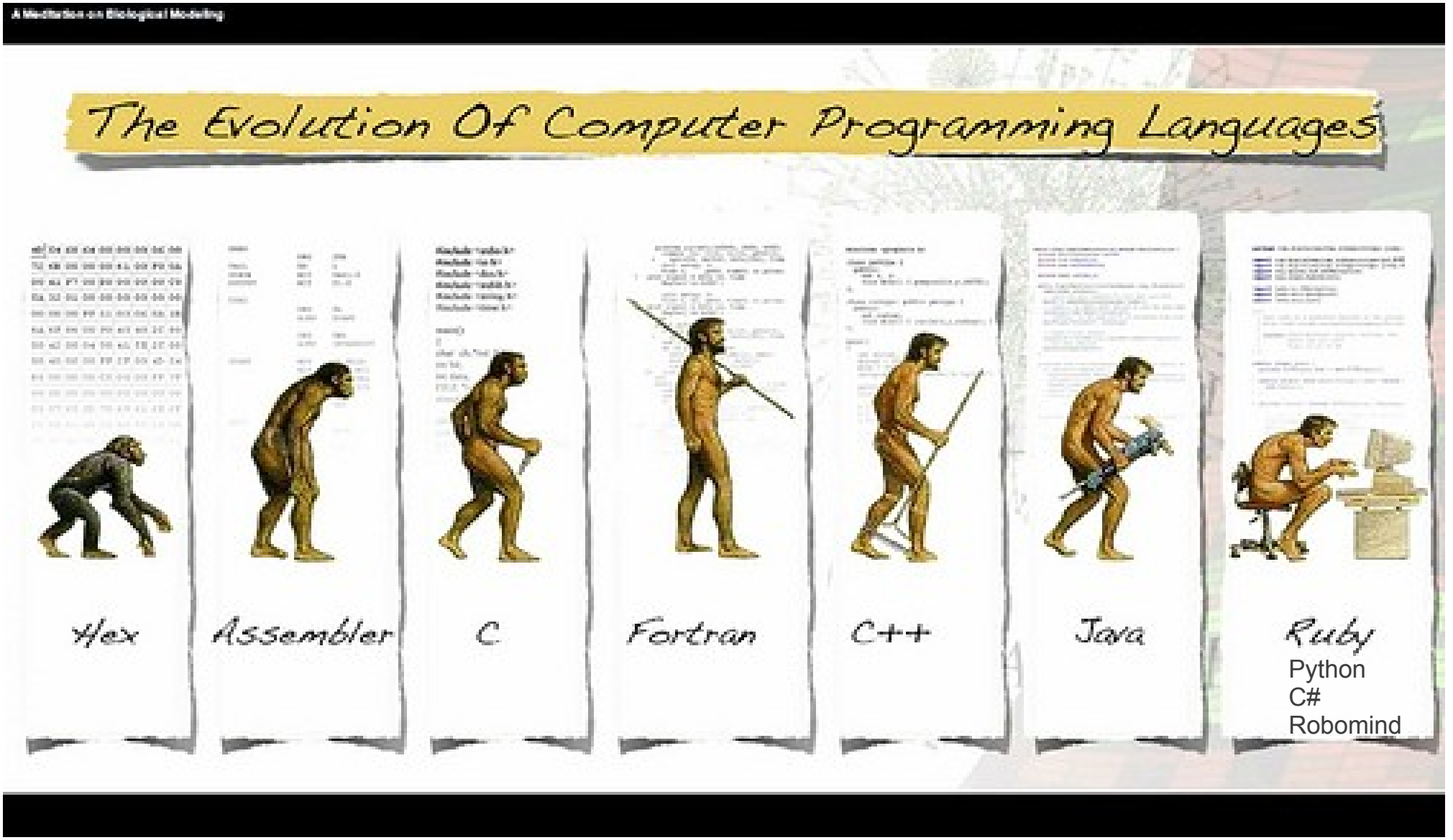
```
//Mostra uma sequencia de numeros de 1 a n
import java.util.Scanner;
class MeuPrograma
{
    public static void main (String[] args)
    {
        Scanner entrada = new Scanner(System.in);
        int n = entrada.nextInt();
        for(int a = 1; a <= n; a++)
        {
            System.out.print(a);
        }
    }
}
```

Programa em Python

```
# Mostra uma sequencia de numeros de 1 a n
n = input()
for a in range(1, n+1):
    print(a)
```

(*) Linguagem de Programação que vamos a utilizar nesse quadrimestre!

Linguagens de programação



Desenvolvendo um programa de computador

Análise:
Estuda-se o enunciado do problema para definir os dados de entrada, o processamento e os dados de saída

Algoritmo:
Cria-se uma sequência de passos para resolver o problema.

Codificação:
Traduz-se essa sequência de passos em um programa escrito em uma linguagem que o computador entenda.

Desenvolvendo um programa de computador

Análise:
Estuda-se o enunciado do problema para definir os dados de entrada, o processamento e os dados de saída

Algoritmo (projeto):
Cria-se uma sequência de passos para resolver o problema.

Codificação (implementação)**:**
Traduz-se essa sequência de passos em um programa escrito em uma linguagem que o computador entenda.

Formas de representação de algoritmos

- Descrição narrativa.
- Fluxograma.
- Pseudocódigo ou Portugol.

Descrição narrativa

Consiste em analisar o enunciado do problema e escrever, utilizando uma linguagem natural, os passos a serem seguidos para sua resolução.

Prós:
Não é necessário aprender conceitos novos pois já entendemos a linguagem natural.

Contras:
Uma linguagem natural pode ser interpretada de várias maneiras.
Isso pode dificultar a transição do algoritmo em um programa

Exemplo de descrição narrativa

Algoritmo para saber se um aluno for aprovado em PI:

(1) Obter as notas da P1 e P2.

(2) Calcular a média aritmética entre as duas.

(3) Se a média for menor ou igual a 5, o aluno foi **re**provado.

Caso contrário, ele foi aprovado.

Exemplo de descrição narrativa

Algoritmo para saber se um aluno for aprovado em PI:

(1) Obter as notas da P1, P2 e média listas de exercícios.

~~(2) Calcular a média aritmética entre as duas.~~

(2) Calcular m = 0,3*P1 + 0,4*P2 + 0,3*(média daslistas)

(3) Se m for menor ou igual a 5, o aluno foi **re**provado.

Caso contrário, ele foi **aprovado**.

Exemplo de descrição narrativa

Algoritmo para saber se um aluno for aprovado em PI:

(1) Obter as notas da P1, P2 e média listas de exercícios.

~~(2) Calcular a média aritmética entre as duas.~~

(2) Calcular m = 0,3*P1 + 0,4*P2 + 0,3*(média das listas)

(3) Se m for menor ou igual a 5, o aluno foi **re**provado.

Caso contrário, ele foi **aprovado**.

Como modificar este algoritmo para verificar se o aluno for reprovado por faltas?

Exemplo de descrição narrativa

Algoritmo para saber se um aluno for aprovado em PI:

(1) Obter as notas da P1, P2 e média listas de exercícios.

(2) Calcular m = 0,3*P1 + 0,4*P2 + 0,3*(média das listas)

(3) Se m for menor ou igual a 5 ou a frequência for menor que 75%, o aluno foi reprovado. Caso contrário, ele foi aprovado.

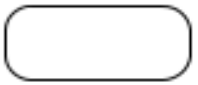
Fluxograma

Consiste em analisar o enunciado do problema a escrever, utilizando símbolos gráficos predefinidos, os passos seguidos para sua resolução.

Prós:
O entendimento de elementos gráficos é mais simples que o entendimento de textos.

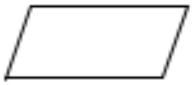
Contras:
É necessário aprender a simbologia dos fluxogramas e, além disso, o algoritmo resultante pode não ter detalhes suficientes para sua transição em um programa.

Fluxograma



Início ou fim do algoritmo.

Atribuição ou cálculo.



Entrada de dados.

Saída de dados.

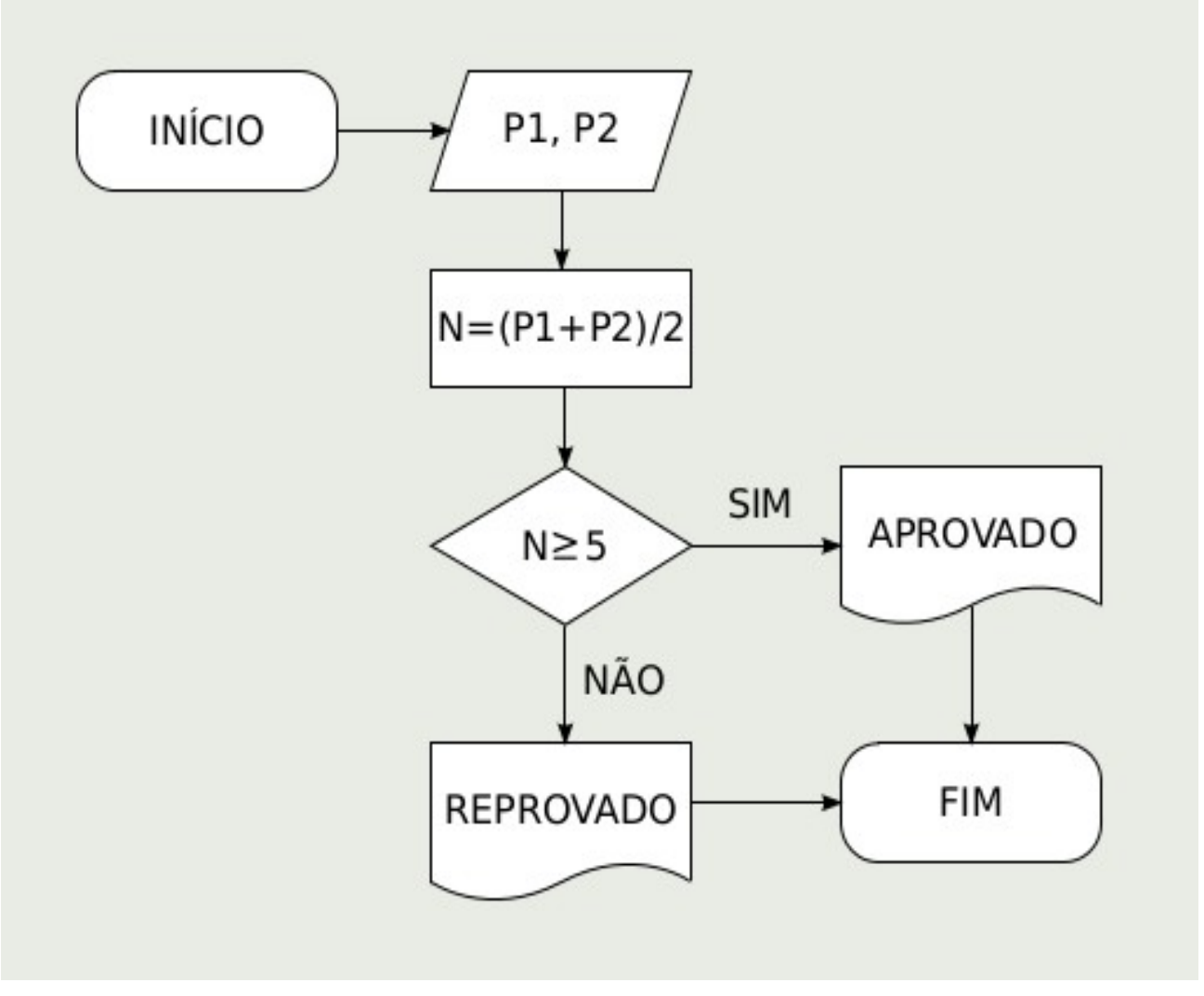
Tomada de decisão.

Sentido do fluxo de dados e ordem de execução dos passos.

Símbolos utilizados em um fluxograma

Exemplo de fluxograma

Algoritmo para saber se um aluno for aprovado em PI:



Pseudocódigo ou Portugol

Consiste em analisar o enunciado do problema e escrever, por meio de regras predefinidas, os passos a serem seguidos para sua resolução.

Prós:
A passagem do algoritmo para qualquer linguagem de programação é quase imediata, bastando conhecer as palavras reservadas da linguagem que será utilizada.

Contras:
É necessário aprender as regras de pseudocódigo.

Pseudocódigo ou Portugol

Forma geral de um pseudocódigo

```
ALGORITMO "nome do algoritmo"
VAR
     <declaração_de_variáveis>
INICIO
     <corpo_do_algoritmo>
FIMALGORITMO
```

As palavras em **negrito** são palavras reservadas do pseudocódigo.

Na declaração <declaração_de_variáveis> escrevemos quais as variáveis a utilizar.

No <corpo_do_algoritmo> escrevemos a sequência de instruções usando as variáveis declaradas.

Pseudocódigo ou Portugol

Exemplo de pseudocódigo

Algoritmo para saber se um aluno foi aprovado em PI:

```
ALGORITMO "Média"
VAR
     P1, P2, N : REAL
INICIO
     LEIA(P1)
     LEIA(P2)
     N ← (P1+P2)/2
     SE N ≥ 5 ENTAO
          ESCREVA("Aprovado")
     SENAO
          ESCREVA("Reprovado")
     FIMSE
FIMALGORITMO
```

Tipos de dados

Todo o trabalho realizado por um computador consiste em manipular informações contidas em sua memória. São elas:

• As **instruções**, que comandam o funcionamento da máquina e determinam como os dados são tratados.

• Os **dados**, que correspondem à porção das informações a serem processadas (entradas e saídas).

Em geral os dados são classificados em três tipos: *numéricos, literais, e lógicos.*

Tipos de dados

Todo o trabalho realizado por um computador consiste em manipular informações contidas em sua memória. São elas:

• As **instruções**, que comandam o funcionamento da máquina e determinam como os dados são tratados.

• Os **dados**, que correspondem à porção das informações a serem processadas (entradas e saídas).

Em geral os dados são classificados em três tipos: *numéricos*, *literais*, e *lógicos*.

Python: Int, float, long, str, bool

Dados numéricos no computador

Na atual arquitetura dos computadores, os dados numéricos são divididos em dois tipos:

- Inteiros: Subconjunto finito de $\mathbb{Z}$: {…,-2,-1,0,1,2,...}

- Reais: Subconjunto finito de $\mathbb{R}$ em uma notação de ponto flutuante.

Dados numéricos inteiros

Os números **inteiros** podem ser positivos ou negativos e não possuem parte fracionária.

Exemplos de inteiros
23
0
-12
982

Dados numéricos inteiros

Dependendo da linguagem de programação e arquitetura do computador, os números inteiros podem ser subdivididos segundo diferentes faixas de valores.

Subconjuntos de inteiros

- Inteiros de 16 bits com sinal: [-32768, 32767].
- Inteiros de 16 bits sem sinal: [0, 65535].
- Inteiros de 32 bits com sinal: [-2147483648, 2147483647].

Em Python, os inteiros em geral são de 32 bits com sinal e dizemos que eles são do tipo `int` (mas também podem ser armazenados em 64 bits ou mais).

Dados numéricos reais

Os números **reais** podem ser positivos ou negativos e podem possuir componentes decimais ou fracionários.

Exemplos de reais
23.01
144.0
-13.3
0.0

- Em programação usa-se a convenção inglesa de empregar o ponto (.) como separador decimal.
- Observe que '144.0' é de um tipo de dado diferente de '144'. O primeiro é real; o segundo é inteiro. Usamos o ponto para mostrar essa diferença.

Dados numéricos reais

Dependendo da linguagem de programação e arquitetura do computador, os números reais também podem ser subdivididos segundo diferentes faixas de valores.

Subconjuntos de reais

- Ponto flutuante de 32 bits: $[-3.4 \times 10^{-38}, 3.4 \times 10^{38}]$.
- Ponto flutuante de 64 bits: $[-1.7 \times 10^{-308}, 1.7 \times 10^{308}]$.

Em Python, os números reais são geralmente números em ponto flutuante de 64 bits e dizemos que eles são do tipo **`float`**.

Dados literais

Um dado do tipo **literal** é constituído por uma sequência de caracteres (letras, dígitos e/ou símbolos). Essa sequência é comumente chamada de **cadeia** de caracteres ou **string**.
São representados sempre entre aspas.
O *comprimento* de uma string é o seu número de caracteres.

Exemplos de dados do tipo literal
"Olá"
"oLá ?!$"
"AbCdefGHi"

Dados literais

Exemplos de dados do tipo literal

" "

"1.0"

"1-2+3"

- Uma string pode ser composta apenas de espaços (cada espaço é por si só um caractere).
- Note que "1.0" representa uma string de comprimento 3 e é constituída pelos caracteres "1", "." e "0", diferindo de 1.0 que é um dado do tipo real.

Em Python, dizemos que dados literais são do tipo **`str`** e são sempre especificados entre aspas.

Dados lógicos

Dados do tipo **lógico**, ou **booleano** (devido ao matemático George Boole, 1815–1864), só podem conter dois valores: **verdadeiro** ou **falso**.

Outras notações possíveis para dados lógicos

sim/não
1/0
true/false

Em Python, dados lógicos são chamados de dados do tipo **`bool`** e admitem os valores **`True`** e **`False`**.