

```
;;; =====  
;;; Common Lisp: A Gentle Introduction to Symbolic Computation  
;;; https://www.cs.cmu.edu/~dst/LispBook/   
;;; -----  
;;; Capítulo: 1  
;;; -----  
;;; Por: Abrantes Araújo Silva Filho  
;;;      abrantesasf@pm.me  
;;; =====
```

```
;;; 1.1: Introduction  
;;; -----
```

```
;; Funções x Dados:  
;;   dados    = informações  
;;   funções = processos pelos quais os dados passam  
;;           as funções operam sobre os dados e retornam um resultado
```

```
;;; 1.2: Functions on numbers  
;;; -----
```

```
;; Nós invocamos (chamamos) uma FUNÇÃO que recebe INPUTS. Os inputs são  
;; processados pela função e recebemos um OUTPUT.
```

```
(+ 2 3)  
(- 2 3)  
(* 2 3)  
(/ 8 2)  
(abs -1)  
(sqrt 36)
```

```
;;; 1.2: Three kinds of numbers  
;;; -----
```

```
;; Inteiros, ponto flutuante e racionais (razões: quando chamamos uma função  
;; aritmética com inputs inteiros, Lisp usualmente retornará um inteiro ou uma  
;; razão - na forma reduzida - como resultado). Se misturarmos inteiros e ponto  
;; flutuante o resultado será um ponto flutuante.
```

```
(+ 3 4)  
(sqrt 25)  
(/ 3 6)
```

```
;;; 1.4: Order of inputs is important  
;;; -----
```

```
(/ 8 4)  
(/ 4 8)  
  
(/ 8 10)  
(/ 10 8)
```

```
;;; EXERCISE 1.1  
;;; -----
```

```
;;; 1.5: Symbols
;;; -----

;; Símbolos são outro tipo de dados no Lisp. Seus NOMES podem conter
;; praticamente qualquer combinação de números, letras e alguns caracteres
;; especiais.

;; É fundamental diferenciar inteiros (e outros números) de símbolos:
;; inteiros:      sequência de dígitos (0-9) opcionalmente precedida por +-
;; outros núemros: pontos flutuantes, razões
;; símbolos:      todo o resto que não seja um número

;;; EXERCISE 1.2
;;; -----

;;; 1.6: The special symbols T and NIL
;;; -----

;; t (ou T):      verdadeiro, veracidade, sim
;; nil (ou NIL):  falso, falsidade, vazio, não

;;; 1.7: Predicates; Some simple predicates
;;; -----

;; São FUNÇÕES que retornam T ou NIL são chamadas de PREDICADOS e, geralmente,
;; o nome dessas funções terminam com a letra "p", de predicado.
;;
;; Predicados que recebem T ou NIL como input e fornecem T ou NIL como output
;; são chamados de funções VERDADE.

(numberp 2)
(numberp 'dog)

(symbolp 'cat)
(symbolp 23)

(zerop 0)
(zerop 99)

(evenp 2)
(evenp 1)

(oddp 2)
(oddp 1)

(> 3 2)
(> 2 3)

(< 4 6)
(< 6 4)

;;; 1.8: The EQUAL predicate:
;;; -----

;; Existem vários predicados para comparar se duas coisas são iguais,
```

```
;; como por exemplo: EQUAL, EQUALP, EQ, EQL, =  
;; Por enquanto vamos nos fixar no EQUAL.
```

```
(equal 'cat 'cat)  
(equal 'cat 'mouse)  
(equal 3 3)  
(equal 3 'three)
```

```
;;; EXERCISE 1.3  
;;; -----
```

```
;;; 1.9: Putting functions together:  
;;; -----
```

```
;; As funções que já são definidas (built-in) no Lisp são chamadas de  
;; funções PRIMITIVAS. Podemos criar nossas próprias funções:
```

```
(defun add1 (x)  
  (+ x 1))
```

```
(add1 5)
```

```
(defun add2 (x)  
  (add1 (add1 x)))
```

```
(add2 5)
```

```
(defun twop (x)  
  (equal x 2))
```

```
(twop 2)  
(twop 9)
```

```
;; EXERCISES 1.4 - 1.8  
;; -----
```

```
(defun onemorep (x y)  
  (equal x (add1 y)))
```

```
(onemorep 7 6)  
(onemorep 7 3)  
(onemorep 3 3)
```

```
;; EXERCISES 1.9 - 1.13  
;; -----
```

```
;;; 1.10: The NOT predicate:  
;;; -----
```

```
(not t)  
(not nil)
```

```
;; Atenção: NIL é a única maneira de dizer não/falso em Lisp, mas para dizer  
;; sim/verdadeiro qualquer coisa serve, não apenas o T, ou seja: qualquer coisa  
;; que não seja NIL é tratado como sim/verdade. Desse modo o NOT retorna NIL  
;; para qualquer input que não seja o próprio NIL.
```

```
(not 'a)
(not (not 'a))
(not (not nil))
```

```
;;; EXERCISE 1.14
;;; -----
```

```
;;; 1.11: Negating a predicate:
;;; -----
```

```
(defun not-equal (x y)
  (not (equal x y)))

(not-equal 'pink 'green)
(not-equal 'pink 'pink)
```

```
;;; EXERCISES 1.15 - 1.20
;;; -----
```

```
;;; 1.12: Number of inputs to a function:
;;; -----
```

```
;; Algumas funções exigem um número fixo de argumentos (0, 1, 2, etc.), e
;; outras funções aceitam um número variável de argumentos (0 ou 1 ou 2 ou etc.)
```

```
;; Observações para funções aritméticas:
;;   + e * : aceitam 0, 1 ou mais argumentos
;;   - e / : aceitam 1 ou mais argumentos (nega o input; calcula o recíproco)
```

```
(* 2 3 5)
(- 50 3 4)
(/ 120 3 5)
```

```
(- 4)
(/ 4)
```

```
;;; 1.13: Errors:
;;; -----
```

```
;; Erros comuns:
;;   Wrong type of input
;;   Too few inputs
;;   Too many inputs
;;   Division by zero
```

```
(+ 3 'fred)
(equal 2)
(oddp 4 7)
(/ 3 0)
```

```
;;; EXERCISES 1.21 - 1.27
;;; -----
```