```
;;=============================================================================;;
;; How to Design Programs, 2ª edição                                           ;;
;; https://htdp.org/                                                           ;;
;;-----------------------------------------------------------------------------;;
;; Capítulo 02: Functions and Programs                                         ;;
;;-----------------------------------------------------------------------------;;
;; Resumo por: Abrantes Araújo Silva Filho                                     ;;
;;             abrantesasf@gmail.com                                           ;;
;;=============================================================================;;
```

## 2.1) Funções e variáveis:

Grosso modo, programas são funções:
   a) cujo input são dados diversos (números, imagens, booleans, strings, etc.)
   b) que são "disparadas" por eventos do mundo real
   c) cujo output afetam o mundo real

Uma função determina uma nova operação nos dados, e cada função pode ter 0, um ou mais inputs (argumentos ou parâmetros) que serão "consumidos" pela função.

Variáveis ou constantes não são dados, elas representam dados.

Definição de funções em BSL:

```
(define nome
  (lambda (parametros)
    (corpo)))
```

Definição de constantes em BSL:

```
(define nome expressao)
```

## 2.2) Como as funções são calculadas:

É necessário saber como uma função é calculada após sua aplicação, para poder descobrir o que deu errado quando alguma coisa der errado (e elas darão errado com certeza em algum momento).

A função é avaliada de acordo com as normas de avaliação da Lisp: primeiro resolve os argumentos recursivamente e, depois, passa para a função.

Use o STEPPER da DrRacket para visualizar como as funções são calculadas!

## 2.3) Composição de funções:

Tipicamente um programa não é composto apenas por uma única função mas, sim, por uma composição de funções diferentes. Em geral temos uma função PRINCIPAL e uma ou mais funções AUXILIARES ("helpers"). Isso facilita o entendimento e a manutenção posterior de um programa.

Uma regra importante é: DEFINA 1 FUNÇÃO PARA CADA TAREFA!

## 2.4) Constantes globais:

O "nome" de uma constante é uma variável global, e sua criação é chamada de definição da constante. Elas introduzem nomes para todos os tipos de dados. Por convenção elas são escritas em LETRAS MAIÚSCULAS.

Podem ser literais ou calculadas através de uma expressão.

Uma regra importante é: PARA CADA CONSTANTE MENCIONADA EM UM PROBLEMA, INTRODUZA UMA DEFINIÇÃO DE CONSTANTE NO PROGRAMA (elimine os números mágicos)!

## 2.5) Programas:

Da perspectiva de uma programa, temos 2 tipos principais: programas em BATCH e programas INTERATIVOS.

### 2.5.1) Programas em BATCH:

Programas em batch, em geral:
- Consomem todos os inputs ao mesmo tempo
- Realizam o processamento
- Fornecem o output
- Não há interação com o usuário

A função MAIN em um programa em batch:
- Geralmente chama várias outras funções auxiliares para o processamente e exibição do output.

Na BSL programas em batch geralmente precisam da biblioteca:
- 2htdp/batch-io: fornece as funções:
  - read-file
  - write-file

Para interação, pode-se usar 'stdin e 'stdout.

### 2.5.2) Programas INTERATIVOS:

Programas interativos, em geral:
- Consomem input, processam, dão o output e repetem tudo de novo
- Há interação com o usuário (ou outros sistemas)
- São dirigidos por EVENTOS (event-driven)
- Um evento ocorre quando um input válido ocorre. Os eventos geralmente são:
  - teclado
  - mouse
  - gerados pelo computador (tempo do relógio interno, por exemplo).

A função MAIN em um programa interativo:
- Geralmente descreve funções que vão lidar com cada tipo de evento que ocorrer e é considerado válido, chamadas de EVENT HANDLERS.

A mecânima de escrever programas interativos é, grosso modo, a seguinte:
- Definir o "estado inicial" do programa
- Definir os eventos válidos no programa
- Definir quais funções lidam com quais eventos (definir os event handlers)
- Definir como um "estado" do programa será transformado no próximo
- Manter o rastreamento do "estado atual" (e seus atributos) no programa
- Definir como o "estado atual" será exibido ao usuário

Para cumprir todas as tarefas acimas, a BSL e a DrRacket têm uma biblioteca especial chamada de 2htdp/universe, que contém a função BIG-BANG. Essa função big-bang descreve como o programa se conecta a um pequeno segmento do mundo, ou seja, a função big-bang descreve um pequeno segmento do mundo e seus estados (os programas com big-bang são, as vezes, chamados de world programs).

Uma expressão BIG-BANG tem a seguinte forma, em geral:

```
(big-bang <estado-inicial>
  [on-tick   <event handler que responde ao clock>]
  [on-key    <event handler que responde ao teclado>]
  [on-mouse  <event hander que responde ao mouse>]
  [to-draw   <render: como o estado atual será apresentado ao mundo>]
  [stop-when <boolean: indica quando parar>]
  ... )
```

Observações:

- O render definido em "to-draw" é a única maneira de uma expressão apresentar dados ao mundo. Na BSL espera-se que ele produza uma imagem!
- Os event handlers são executados a cada vez que ocorrerem os eventos
- A avaliação de uma expressão big-bang é feita da seguinte maneira:
  a) O estado inicial é avaliado e exibido com o to-draw;
  b) Os eventos ocorridos são capturados pelos event handlers, e o valor retornado pelo event handler) é considerado como o PRÓXIMO estado;
  c) Os eventos são processados na ordem em que ocorrem mas, se houver empate, DrRacket desempata a ordem de ocorrência.
  d) O valor do PRÓXIMO estado é avaliado por stop-when:
     - se a expressão for #true o big-bang encerra imediatamente a execução, não considerando esse valor como o estado ATUAL, portanto o valor de PRÓXIMO não é exbido por to-draw.
     - se a expressão for #false o big-bang considerará o PRÓXIMO como o estado ATUAL, produzirá a saída com o to-draw, e continuará a responder por outros eventos.
- O big-bang NÃO ALTERA o estado atual: ele repassa o estado atual para os event handlers (o estado atual é passado como argumento para os event handlers) e recebe um valor que é considerado como o PRÓXIMO estado. Esse valor será convertido (ou não) no estado ATUAL dependendo da experessão stop-when.