Hello, Haskell!

- **Redexes** are reducible expressions. (Evaluation, reduction: "*normalizing*" or "executing" an expression)
- **Functions** are a specific type of expression.
- **Currying.** Apply 1 argument to each of nested functions.

1 Evaluation

Evaluating an expression is reducing the terms until the expression reaaches its simplest form.

Definition : Lazy Evaluation

Defer evaluation of terms until being forced.

- **Infix operators.** `(+) 100 100` or `100 ‘add‘ 100`

2 Associativity and precedence

- **Precedence.** Higher number, higher precedence.
- **Associativity**: left or right.

```
❯ :info *
type Num :: * → Constraint
class Num a where
  ...
  (*) :: a → a → a
  ...
        -- Defined in ‘GHC.Num’
infixl 7 *
```

- Use `space`, not Tab.
- All declarations in the module must start at the **same column**. The first declaration in a module defines the remaining ones.
- **Sectioning** `(+1), (*30), ...`

3 Let and where

- `Let` introduces an expression
- `where` is a declaration.
- **Scope.** The area of *source code* where a binding of a variable applies.
- The **order** in code is unimportant.

`where`: syntactic sugar for a declaration.

Syntactic sugar is syntax within a programming language designed to make expressions easier to write or read.