Product example: calculate product of a list of numbers.

Step 1. Define the type:

\[
\text{product} : [\text{Int}] \rightarrow \text{Int}
\]

(we may generalize later; we "start simple")

Step 2. Enumerate the cases:

\[
\text{product}([]) = \text{(empty list)}
\]

\[
\text{product}([x \cdot x \cdot x]) = \text{(non-empty list)}
\]
Step 3. Define the simple cases

\[ \text{product}(\cdot) = 1 \]

\[ \text{product}(1 \times x \times 3) = \]

(Usually, the simple cases become base cases, as it happens here.)
Step 4 Define the other case

\[ \text{product}([ ] ) = 1 \]

\[ \text{product}(x; \text{xs}) = x \ast \text{product} \text{xs} \]

(As in this example, the "other" case often become the certain cases)
Step 5 Generalize and simplify

\[ \text{product } \colon \text{Num } a \Rightarrow [a] \rightarrow a \]

The function `defn` is unchanged when we generalize from `Int` to any type of the `Num` class.

`foldr (*) 1` when applied to `[ ]`

the function that replaces `cons` `"++"`
foldr is actually very similar to
FP's `insert ()`

`(*)` is the FP equivalent to

foldr (\x\y -> x)

it works only on lists with at least two values

foldr 1 (\x\y -> x) is actually closer to (!).